Workshop on Structure Theory of Petri Nets
STRUCTURE 2017
Zaragoza, Spain, 26 June, 2017

Session honouring Manuel Silva on the occasion of his 65th birthday

Organizing Committee
Serge Haddad
Jetty Kleijn
Jörg Desel
J.M. Colom

Invited Speakers
Manuel Silva
Jörg Desel
Gianfranco Balbo
Serge Haddad
David de Frutos-Escrig
Jetty Kleijn
Maciej Koutny
Alessandro Giua
Wil van der Aalst
Jordi Cortadella
Preface

This booklet constitutes the proceedings of the Workshop on Structure Theory of Petri Nets (STRUCTURE). This Workshop is a satellite event of the 38th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (PETRI NETS 2017).

The main topic of the Workshop is Structure Theory of Petri nets. This field is one of the intrinsic facets to this family of models. Developed from the dawn of Petri nets throughout, for example, marking linear invariants or the structural definition of subclasses of nets, today this theory has become a very large and highly specialized domain for many reasons.

The primary purpose of the STRUCTURE Workshop is to present new visions of old specialties, and to help, practitioners and researchers, stay abreast in all areas of Structure Theory of Petri Nets or its use in theory and practice. STRUCTURE focuses on integrating and adding understanding/value to the existing results in the literature. This goal is intended to be accomplished, for this first edition of the Workshop, by surveys, tutorials, and open problems or challenges on special topics of interest to the communities around Petri Nets and other models of concurrency.

STRUCTURE Workshop does not intend to publish “new” research of Structure Theory of Petri Nets. This is left to the PETRI NETS Conference. Instead, STRUCTURE focuses on surveys and tutorials that integrate the existing literature and put its results in context. The main topics are:

1. Concepts: Structural objects and their computation
2. Structure and behavior: Properties guaranteed by the structure
3. Models defined from the structure: Subclasses of Petri Nets
4. Structure based analysis techniques
5. Application domains exploiting structure
6. Structure Theory in other models of Concurrency
7. Future trends and Challenges

The STRUCTURE Workshop was held at the University Zaragoza, Zaragoza, Spain during June 26, 2017. We would like to express our deepest thanks to the Organizing Committee for the time and effort invested in the local organization of the Workshop. The number of presentations amounted to 10. The authors are all reputed researchers within the field of Petri Nets and, in general, Models of Concurrency, which throughout their trajectories have made significant contributions within the domain of the Structure Theory of Petri Nets.

With this Workshop we would like to pay tribute to Professor Manuel Silva on the occasion of his recent 65th birthday, for whom the Structure Theory of Petri nets has played a central role in his research work.

June 2017

Organizing Committee of STRUCTURE
Organization

Organizing Committee

J. Desel, Germany
S. Haddad, France
J. Kleijn, The Netherlands
J.M. Colom, Spain

Speakers

Wil v.d. Aalst Eindhoven University of Technology (TU/e), The Netherlands
Gianfranco Balbo Università di Torino, Italy
Jordi Cortadella Universitat Politècnica de Catalunya, Spain
Jörg Desel Fernuniversität in Hagen, Germany
David de Frutos-Escrig Universidad Complutense de Madrid, Spain
Alessandro Giua Università di Cagliari, Italy
Serge Haddad École Normale Supérieure de Cachan, France
Jetty Kleijn Leiden University, The Netherlands
Maciej Koutny Newcastle University, UK
Manuel Silva University of Zaragoza, Spain

Sponsoring Institutions

University of Zaragoza (Vicerrectorado de Investigación) [N.Ref. 171360]
Workshop on Structure Theory of Petri Nets  
STRUCTURE 2017

Monday, 26 June 2017 (Lecture Room A.14)

09:00-10:30 Session A: Definition of Structure

- Manuel Silva. Models, modeling paradigm and structure
- Jörg Desel. What is a Petri Net? Does Structure Provide an Added Value?

10:30-11:00 Coffee Break

11:00-12:30 Session B: The Theoretical Framework

- Serge Haddad. An Algorithmic View on Continuous Petri Nets.
- David de Frutos-Escrig. Recalling some decidability and complexity results on (structurally) constrained and general P/T nets.

12:30-13:45 Lunch Break

13:45-14:45 Session C: The Application Framework

- Maciej Koutny. Structuring nets and their behaviours.
- Alessandro Giua. On the control and estimation of discrete event systems using Petri net structural approaches.

14:45-15:15 Coffee Break

15:15-16:45 Session D: Future Trends and Challenges

- Jordi Cortadella. Making Petri nets friendlier to engineers.

16:45-17:15 Coffee Break

17:15-18:15 Session E: Homage to Prof. Manuel Silva

- Short communications about the figure and the work of Prof. Silva
# Table of Contents

## Regular Presentations

Models, modeling paradigm and structure ........................................... 1
   *Manuel Silva*

What is a Petri Net? Does Structure Provide an Added Value? .......... 11
   *Jörg Desel*

Stochastic Petri Nets: A Structured Approach to the Performance
Evaluation of Complex Systems .................................................. 17
   *Gianfranco Balbo*

An Algorithmic View on Continuous Petri Nets ............................... 19
   *Serge Haddad*

Recalling some decidability and complexity results on (structurally)
constrained and general P/T nets .............................................. 27
   *David de Frutos Escrig*

Structure of Behaviour in Extended EN Systems ............................. 33
   *Jetty Kleijn*

Structuring nets and their behaviours ....................................... 35
   *Maciej Koutny*

On the control and estimation of discrete event systems using Petri net
structural approaches .......................................................... 37
   *Alessandro Giua*

Making Petri nets friendlier to engineers .................................. 43
   *Jordi Cortadella*

Structure Theory in a Dynamic Data-Driven World: Applications in
Process Mining and Business Process Management ........................ 45
   *Wil v.d. Aalst*

## Author Index
................................................................. 55
Models, modeling paradigm and structure  
(Extended Abstract)

Manuel Silva  
Instituto de Investigación en Ingeniería de Aragón (I3A),  
Universidad de Zaragoza, Spain  
(e-mail: silva@unizar.es)

1 Models, formalisms and modeling paradigm

Models are abstract views of existing or (being) designed systems. As abstract views they just represent some features of the system. Depending on the characteristics to be represented, models may be of very different nature; for example, they may focus on some geometrical aspects (3D-architectural models; floor plans, eventually with relative flow circulations; etc.), or they try to emphasize certain aspects of its dynamic behavior (a wide variety of dynamic models may exist). Among the diversity of models a few can be said to be formal in a mathematical sense. Formal models can be validated and using some proof techniques allow to establish its correctness in a given pre-established sense.

Formal models are expressed in some formal languages or formalisms; that is, in some means of representation/communication with clearly defined syntax and semantics, powerful enough to express certain abstract views (or descriptions) of a given class of system. Each peculiar instance of a given formalism may describe a specific model. Therefore, formalisms may be seen as meta-models. The expressivity of formalisms may be considered from theoretical or practical perspectives. In the first case, it is concerned by the variety of systems that can be represented; in the second one, provided the fact that the system can be represented, the main concerns are on conciseness and easiness of use. As a basic trade-off between expressive power and analyzability, the more a given formalism can express, the harder it becomes to analyze.

Formalisms derive from a conceptual framework where a kind of formal model of a system can be expressed. For instance, ordinary differential equations constitute a formalism to model the dynamic behavior of (space and time) continuous systems with lumped parameters. Conceived for diverse purposes, some examples of formalisms for Discrete Event Systems (DES) are different kinds of State Diagrams; Markov Chains and many types of Queuing Networks (for performance evaluation); PERT and conjunctive/disjunctive graphs (for scheduling); different kinds of Petri Nets (PNs) or Process Algebra; etc.

Petri Nets are bipartite structures based on places (local state variables) and transitions (local state transformers). Provided with some initial state (marking) they can model very complex dynamic DES. Place/Transition systems (PT-systems) are able to model infinite state (i.e., unbounded) systems, and constitute a “frontier formalism”. By this we mean that they are certainly expressive,
while most classical properties are decidable on untimed systems. For example, reachability [14, 16, 21]; nevertheless, “small” extensions (as inhibitor arcs, or firing priorities) transform them into formalisms able to simulate Turing machines. From a quite different perspective, PT-net models provided with a timed interpretation—even if continuous under infinite server semantics—are also able to simulate Turing machines [19].

In fact, almost three decades ago in the report about Future directions in control theory: a mathematical perspective [9] it was stated that:

there exist no formalisms for DES mathematically as compact or computationally as tractable as the differential equations are for continuous systems, particularly with the goal of control.

A quarter of century later, the idea is reformulated as “Unity in Diversity, Diversity in Unity” [1]. In view of the long life-cycle of many systems (conception and modeling; analysis and synthesis from different perspectives; implementation and operation) and the diversity of application domains, it seems desirable to have a family of formalisms rather than a collection of “unrelated” or weakly related formalisms. Nevertheless, when dealing with complex systems, multiformalisms approaches are frequently used. “Unrelated formalisms” (eventually one per phase of the life-cycle), based on different points of view and using particular underlying theories, are frequently used; for example, some kind of automata (for functional specification), QNs (for performance evaluation), PERTs (for basic scheduling/control), different coding schemes (for software implementation), etc. Eventually, provided with appropriate interpretations, PNs can do an analogous kind of job: autonomous PNs, time-stochastic PNs (many possibilities), marking diagrams, etc, till the (semi)automatic generation of code, eventually fault-tolerant to a certain level.

Following Thomas S. Kuhn’s ideas, a paradigm is: “[...] the total pattern of perceiving, conceptualizing, acting, validating, and valuing associated with a particular image of reality that prevails in a science or a branch of science” [15]. The conceptual seeds proposed by Carl Adam Petri do not just flourish as a single formalism, initially the so-called Condition/Event nets, where the state variables (one per place) are Boolean. His work, and the improvements and extensions developed by many other scientists and engineers, inspired a set of “related” formalisms that constitute a modeling paradigm, a conceptual framework that allows to obtain “derived” formalisms from some common concepts and principles.

Nevertheless, even if trying to reduce the diversity of formalisms to a common framework (modeling paradigm) may be perceived as a “natural goal”, different languages, calculi and theories may be needed or be convenient in order to deal with alternative perspectives. The use of distinct formalisms usually represents different concerns and subcultures on the users (preferred languages, tools, etc.). In certain cases, choices may be application-domain dependent. Otherwise stated, the complexity and variety of systems may suggest the interest of having multi-paradigm environments. Provided the existence of concept mappings
and model transformations techniques, it is possible to convert models in one paradigm into models in an alternative one. This fact asks for the existence of sound and efficient bridges connecting different modeling paradigms, a major issue [25]. Like many other questions in human societies, the number of alternative formalisms tends to fluctuate. The field of DES is immense, truly multifaceted; in particular conceptual paradigms should deal inter alia with modeling, identification, logical analysis, performance evaluation, parametric optimization, dynamic control, diagnosis and opacity, and implementation issues. Therefore, it is important to remark that, proceeding within a given modeling paradigm may be really a multidisciplinary task!

Inside a formal modeling paradigm, coherence among models usable at different phases, economy in the transformations, and synergy in the development of analysis and synthesis theories and techniques are among the expected advantages. As a very basic example of synergy between the theories of formalisms belonging to different levels of abstraction in the PNs framework, let us just mention that basic concepts and techniques for the analysis were immediately transferred from PT-nets to Colored PNs, even if nice developments were needed.

Strictly speaking, to interpret a net system is merely to assign a meaning or interpretation to the various entities: places, transitions and tokens (in some cases, also to the arcs). In this sense it can be said that autonomous PN systems are partially-interpreted bipartite and animate graphs. But this does not alter at all the behavior of the net model; it remains “autonomous”. The marking informs about what may happen, but does not condition what (for example, among alternatives) actually happens and when it will happen. Nevertheless, in control problems the model of a controller should be in close loop (otherwise stated, concurrently composed) with the plant being controlled. Thus inputs and outputs synchronize the formal model with the external world; moreover, means are needed to express time dependent evolutions. Interpreted extensions of “autonomous” systems, lead to non-autonomous models, and its dynamical evolution depends not only on the net marking, but also on the state of the environment being considered (i.e., a non-autonomous DES is “reactive” with respect to its environment).

Petri net is a generic term used to designate a family of related DES formalisms, all sharing some basic relevant features as minimality in the number of primitives, non-determinism, and locality of the states and actions (with consequences over model building —top-down or bottom-up— and structuration, or temporal realism). In our view, the PN modeling paradigm derives from the “cross-product” of the different levels of abstraction of autonomous PNs formalisms (Condition/Event; Place/Transition; Colored...) and the different interpreted extensions (e.g., extended by considering external events or conditions and actions, adding quantitative time in one of the many possible ways, etc.; see Fig. 1).

Engineering is an art and a science. In practice, engineering disciplines combine scientific-based knowledge, frequently created, adapted or completed in their own field, with less formal, but often very creative activities, partially
based on experience and intuition and supported through trial and error approaches. \textit{Maturity} in an engineering discipline requires formal methods, and the existence of paradigmatic or standard models (idea of patterns for reuse). In this context, the availability of powerful analysis and synthesis techniques is very important. In this sense, it can be said that PN theory and applications is a mature field. Moreover, in order to efficiently benefit in practice from PN theory, software tools are necessary. As a final remark, let us point a very important issue in most industrial contexts: standardization. The existence of standard norms adopted by international organizations such as the ISO (International Organization for Standardization) or the IEC (International Electrotechnical Commission) is important for many reasons, not least because such norms derive from technical consensus.

2 Structure and behavior

From early times, M. H. T. Hack clearly pointed out that the graphical appeal of Petri net methods permits a better grasp for intuitive arguments, which can help enormously to find rigorous proofs of various facts [12]. Even before and in a more general context, A.W. Holt and F. Commoner explicitly said that

we are closest in spirit to operations research techniques, but with an insistence on conceptual economy and rigor more common in purer branches of mathematics. Also, it is necessary that our descriptions be built up part by part in analogy to the way in which the systems being described are built up part by part. [13]
According to J. L. Peterson [18], the work of Petri, Holt, and many European researchers, “emphasizes the fundamental concepts of systems”, what represent a “more philosophical [than mechanistic] approach”. Moreover, C.A. Petri persistently claimed that formal languages (in the automata theory sense) was not appropriate to deal with the expressiveness of net systems models. He search for some kind of “isomorphism” between the described system and the model, what contribute to the “faithfulness and understandability” of those formal constructions (for a more detailed reference to those early works and a perspective of the evolution of the discipline, see [23]).

In reality, the previous comments points out the importance of structure in net models: a PN system is a net-based structure (essentially objects as places and transitions, and eventually weighted or annotated relations) provided with an initial condition (marking). Modeling, formal analysis (eventually constrained enumeration, transformation and structural techniques) and simulation are basic pillars for the understanding and design of complex DES. Of course, good theories can overcome or reduce the need of massive simulations, what may be very expensive both in cost and time, even of uncertain interest. The judicious cooperation of such kind of approaches is frequently a central issue.

2.1 On structural properties and components

Based on the separation into a structure and a marking, Structure Theory is a branch of Net Theory devoted to investigate the relationship between the structure and the behavior of net system models. Systems properties (i.e., those depending of a given initial marking) are behavioral; in structural properties the initial marking is abstracted. Total abstraction leads to properties as structural boundedness (the systems that can be defined are always bounded, i.e. independently of the initial marking) or structural non-liveness (if, independently of the initial marking, the system is non-live; more classically stated: a net is structurally live if an initial marking exist such the net system is live). Concepts and techniques for the structural analysis and synthesis of complex systems are frequently parametrized by the initial marking and are computationally much more efficient than purely behavioral approaches. The weakness is that, except for certain net systems subclasses, only necessary or sufficient conditions are frequently available. Nevertheless, the more important aspect is that structural concepts and techniques allows to improve our “understanding” of the systems under consideration.

PN models are bipartite, weighted (or annotated) and animated graphs; therefore graph-based components have always been considered for their analysis and synthesis. Circuits, strongly connected components, siphons and traps (defined from some inclusions of transitions subsets related to some subsets of places), handles, bridges, allocations and configurations, etc., prove the interest of graph-based concepts and techniques. From the perspective of the fundamental or state-transition equation, many other structural properties and components have been derived. The two more important problems with this structural approach are the facts that those equations should be solved in the naturals (in
particular, it is not a vector space!) and spurious solutions (i.e., non-reachable solutions) may exist. This has important consequences over computational complexities, but not only. Despite what has been said, the P- or T- trios of semiflows (vectors), laws (invariants) and components (subnets), for example, are fundamental in understanding many net systems behaviors.

Interestingly, graph-defined structural objects as siphons and traps also lead to alternative trios of related concepts (inclusions on some subsets; some stable predicates; and the corresponding net components, subnets); moreover, even if defined in graph-based terms, siphons and traps can be computed using convex geometry-based algorithms (similar to those used to compute P-semiflows; see, for example, [8]). Also, the consideration of (behavioral) implicit places (i.e., those that never are the unique to constraint the behavior of a net system) are very useful in several respects. For example, their elimination allows to reduce net systems in order to prove behavioral properties, or to simplify the implementation; moreover, some additions may be very interesting, as should be briefly pointed out later.

Like in ordinary differential equations, for example, the important expressive power of PNs makes interesting the consideration of net systems subclasses. Their smaller practical expressive power translates into easier to analyze models. By using only structural constraints, Marked Graph (MG), Free-Choice (FC) or Asymmetric Choice (AC, or simple nets), and many generalizations have been defined. Combining some structurally defined modules and connecting elements (representing shared resources, private buffers, etc.) together with some marking constraints, net systems as Deterministically Synchronized Sequential Processes (DSSP) or Systems of Simple Sequential Processes with Resources (S3PR) have been defined, the first subclass focusing on distributed cooperation among sequential processes, the second on competition (resource allocation).

2.2 Some examples of structural characterizations at the intersection of autonomous and timed net systems

Using structural approaches, let us briefly point out some examples of synergy at the conceptual and technical levels, in all cases rooted at the interleaving of functional and performance analysis and control [22]. Because only a few examples are very informally mentioned, let us basically constraint to some of the developments of our group at Zaragoza, frequently in cooperation with other colleagues.

The starting point was the structural computation (through some LPP) of performance bounds for stochastic PNs [3]; form performance evaluation, new results for checking a logical property (liveness) are derived. The visit ratio is a well defined concept in QNs; its computation is difficult in stochastic (even if Markovian) Petri Net models, because it depends not only on the net structure and routing rates, but also on the initial marking and the firing rates associated to transitions. Nevertheless, reducing the attention to strongly connected and structurally bounded free-choice nets, the visit ratio can be efficiently computed (polynomial time) at the structural level. This turns out to be very important
because it opens a new algebraic and full characterization of structural liveness, a rank theorem [2] (later, [6, 4]). Among its consequences: (1) for any structurally bounded PT-net, a necessary polynomial time computation for structural liveness is derived; and (2) for structurally live and bounded free-choice nets, follows a new, easy and compact proof of the well-known duality theorem [11]. Later, this kind of results and several others have been generalized in different ways, for example to Equal Conflict nets or DSSP systems\(^1\). Moreover, analogous results hold for continuous PNs.

Centered now in timed mono-T-semiflow net systems (conservative and consistent with a unique minimal T-semiflow), it is easy to observe that the firing flow in steady-state may be non-monotonous. For example, against intuition, increasing the firing rate of some transitions, slower throughput may be obtained. The key point is that under the existence of non-monotonicities (sometimes “cached” from certain points of view), throughput discontinuity may appear in continuous PNs under infinite server semantics. This fact and a full characterization of the existence of non-monotonicities is something that may be addressed considering structural arguments and objects as configurations. A configuration is a set of \((p, t)\) arcs, one per transition, covering all transitions; if the support of a P-semiflow is contained in the set of places associated with a configuration (its \(T\)-coverture), then it is said to be suitable; otherwise, it is problematic. Given a continuous mono-T-semiflow net, if all configurations are suitable, for any initial marking the throughput of the transitions is monotonic and there exist no discontinuity at all [17].

Changing the perspective, in Automatic Control, concepts as observability in timed continuous PNs under infinite server semantics can be generalized to structural observability (the timed net is observable for all possible values of the firing rates of transitions; i.e., the observability only depends on the structure of the net) [24]. In addition, weak structural (or generic) observability abstracts the value of the firing rates of transitions, except in a variety of lower dimensions, a concept “close” to observability in linear structured systems [5].

Let us just end this text returning to (structurally) implicit places. For example, the addition of so called cutting implicit places remove spurious solutions. Thus, if all spurious deadlocks are removed, this allows to structurally conclude about deadlock-freeness of the net-based DES under consideration [26], or to greatly improve the throughput upper bounds for those models. Moreover, the addition of implicit places allow new decompositions of the net system model (for example, in order to use some “divide and conquer” iterative approaches for computing performance figures).

Using rather different structural objects or properties, all the previous kind of results mentioned in this subsection, deal with analysis. Nevertheless, implicit places can also be used to improve the implementation or to control the system. For example, the addition of well selected places allows net system decompo-

\(^1\) The results were generalized in a pure untimed framework [27]. Taking into account the construction process of the net (i.e., not viewing it as a flat structure), those kind of results were generalized to a class of multi-level nets [20].
sitions that makes easier their implementation, or that increase the *Hamming distance* of the reachability set interpreted as a code of markings (this open the window to the use of *error-detection* and *error-correction codes* techniques), what allows to improve the safety of the implementation. Moreover, if a place is *structurally implicit*, by using a *deficient marking*, some reachable but undesirable states (i.e., non-spurious solutions) may be removed; in other words, constraining the behavior they implement some *control* functions. In this context, so called *Generalized Mutual Exclusion Constraints* (GMEC) were defined [10]. For example, if the places of the siphons in a S3PR net system are constrained to remain marked, the system become deadlock-free [7].

The main purpose of this note has been just to highlight that *Structure Theory*, built and used dealing with many classes of net-based formalisms and properties—usable with purely DES and derived hybrid or continuous models—is a powerful indigenous feature of the *Petri Net modeling paradigm*. 
References


What is a Petri Net?  
Does Structure Provide an Added Value?  

Extended Abstract

Jörg Desel  
FernUniversität in Hagen, Germany  
joerg.desel@fernuni-hagen.de

1 Introduction

Structure Theory of Petri nets aims at establishing relations between the structure and the behavior of a Petri net, given e.g. by its occurrence sequences. The analysis of structural properties is in general much more efficient than the analysis of behavior, because the latter requires a finite representation of potentially infinitely many occurrence sequences. Such a representation does not necessarily exist; if it exists, it has often exponential size compared to the size of the net. Therefore, structure theory provides means for the analysis of a Petri net’s behavior, which would otherwise be impossible or at least very expensive.

The first question of the title of this contribution can be rephrased as: What is the structure of a Petri net? It is possible to consider any compact representation of a distributed system’s behavior as a structure. However, Petri nets are more than that [1, 2]. As Carl Adam Petri has suggested in his early work, components of a Petri net model components of the modeled system. In particular, signals and messages between system components are reflected as connectors in the net. More importantly, if two components of a system are not directly connected, then the same holds for the respective Petri net components. As a consequence, if all system components have only a bounded number of direct connections to other components, then the Petri net components have a bounded fan-in and fan-out. Therefore, even very complex systems built from elementary components can be adequately represented by Petri nets; the complexity is given by the overall structure, whereas the local vicinity of each Petri net element is simple. One of the reasons for the great success of Petri nets is that Petri nets need no more than a very simple firing rule (dependent on the respective class of Petri nets) which only refers to transitions and their immediate environment. As this environment is typically bounded in distributed systems, the environment of a single Petri net transition does not depend on the complexity of the overall model. Structure theory exploits this fact and studies the communication structure of a modeled system by analyzing the structure of the related net.

There are many well-known results of structure theory. Most of them provide strong links between structural properties like connectedness of a Petri net graph and behavioral properties such as deadlock-freedom, liveness, or boundedness of a marked Petri net, reflecting according relations between the physical or logical...
structure of the modeled system and its behavior. This contribution will provide one particular example of such a relation, which goes back to the early roots of Petri net theory.

Carl Adam Petri has defined the notions of synchrony and of synchronic distances between transitions in his early papers. Synchrony is a purely structural concept, i.e., the synchrony between two transitions can be derived from the structure of the net, whereas synchronic distance refers to the behavior of the net, i.e., to all its occurrence sequences. Synchrony provides an upper limit to synchronic distance. This relation between synchrony and synchronic structure is one of the first contributions to structure theory of Petri nets. Conversely, as Petri has shown by means of a counter example, the synchronic distance between two transitions can be strictly smaller than the synchrony between the transitions.

Synchronic distances strongly depend on the initial marking of a net, whereas synchrony does not. We will study the question whether synchrony more strongly relates to a marking-independent variant of synchronic distance. In other words, we study whether, for each given pair of transitions, there is an initial marking of the net such that the synchrony between the transitions coincides with the synchronic distance between the transitions.

2 Petri's Concept of Synchrony

In [3], page 42, Carl Adam Petri postulated:

*Even speed of clocks can only be established by communication.*

and used the argument that any tiny difference of clock speeds will eventually lead to an arbitrary difference between their mutual number of ticks. He then discussed that even speed has no obvious definition, but can only be defined by means of a closed signal chain, containing both clocks. Since Petri did not assume the existence of time as an objective and available entity, no clock speed can be named accurate (with respect to what?). Hence, according to Petri, we can compare the speed of a clock only with the speed of another clock. If we want both clocks to run with a limited error w.r.t. the respective other clock, we must provide communication means between the two clocks, in both directions. In other words: Two clocks have a limited divergence only if there are mutual signals between the clocks. One obvious consequence of these considerations is that synchronicity between two distinct clocks with its traditional interpretation “ticks at the same time” is meaningless.

We will consider in the sequel two machines tick and tack, that can be considered clocks, but might have any interpretation. The clock tick is repeatedly producing “tick” and, after each “tick” sends a message *I said “tick” to tack*. After receiving this message, tack is allowed to “tack”, and then sends back a message *I said “tack” to tick*, which can only then produce the next “tick”.

In [4], Petri considered one-safe Petri nets (which he always called just nets) and defined a binary relation $D$ between sets of transitions. Two such sets $A$
and $B$ are in the relation $D$ if, in any process representing a run, occurrences of transitions from $A$ strictly alternate with occurrences of transitions from $B$. The clocks *tick* and *tack* (more precisely, the sets \{*tick*\} and \{*tack*\}) are in the relation $D$.

Now let us consider a third clock *tock* that is connected to *tack* in the same way as *tack* is connected to *tick*, i.e., *tack* and *tock* exchange messages and are thus in the relation $D$, see Figure 1.

![Figure 1. Three communicating clocks](image)

The clocks *tick* and *tock* are not as closely related as the other pairs of clocks, because the sequence of events

$\text{tick } \text{tack } \text{tick } \text{tock} \ldots$

is possible, whence we can see two “tick”s before the first “tock”.

Since *tick* $D$ *tack* (i.e., *tick* and *tack* are in the relation $D$) and *tack* $D$ *tock*, we can write *tick* $D^2$ *tock*. In [4], Petri defined the smallest number $m$ such that $A \ D^m \ B$ as the synchrony of two transition sets $A$ and $B$ and also used the notion synchronous with distance $m$. It is worth mentioning that he defined synchronous transitions neither by synchrony 0, which only applies to identical sets of transitions, nor by synchrony 1, which represents strict alternation, but by synchrony 2! This becomes meaningful when we consider our clock *tack* from above as a controller of the clocks *tick* and *tock*. Repeatedly, *tack* sends to both other clocks its message, then both clocks occur concurrently, and so on.

3 Petri’s Synchronic Distance

For an illustration of the concept “synchronic distance”, consider two equally sized chain wheels as in Figure 2. Assume that every turn of the first chain wheel causes an event $a$ and that every turn of the second causes $b$. Since the two wheels are mutually connected by a circular chain, the number of turns of the two wheels will almost be the same, no matter how long they proceed to turn. If the chain is relatively tight, we expect for any future situation that the numbers of $a$-events and $b$-events occurred so far is almost equal. Now assume a longer chain connecting the two wheels. Still, the numbers of occurred $a$-events and $b$-events are related at any time, but the possible mutual deviation grows with the length of the chain. For example, if we can observe at most two more $a$-events than $b$-events and at most one more $b$-events than $a$-events, then $\sigma(a, b) = 2 + 1 = 3$. 
In [5], Petri defined synchronic distance as the maximal deviation between occurrences of two transitions in a run, where here a run was a feasible sequence of transition occurrences:

$$\sigma(a, b) = \max_{\tau \text{ is a run}} |(\text{number of } a \text{ in } \tau) - (\text{number of } b \text{ in } \tau)|$$

It is important to notice that a run in the sense of Petri can start with any reachable marking, not necessarily with a given initial one. Consider, for example, a net with two transitions $a$ and $b$ such that

$$a \ldots b \ldots a \ldots a \ldots b \ldots$$

is a possible run. In none of its finite prefixes, constituting also runs, the absolute difference between the number of $a$-occurrences and $b$-occurrences exceeds 1. However, starting with the marking reached after the first $a$, we observe a subsequence with two $b$s and no $a$, which is a run proving $\sigma(a, b) \geq 2$.

In the same paper, Petri referred to synchrony between transitions, i.e., to the definition based on the alternation relation $D$. The two definitions do not coincide, as Petri showed by means of a counter example, see Figure 3.

From the structure of this net, we obtain $a \, D \, b \, D \, c \, D \, d \, D \, e \, D \, f \, D \, a$ and thus $a \, D^3 \, d$, but not $a \, D^2 \, d$. However, since the outer cycle of this net contains only two tokens, we have $\sigma(a, d) = 2$. 

**Fig. 2. A chain**

**Fig. 3. A counter example**
4 Relations between Synchrony and Synchronic Distance

A straightforward generalization of synchronic distance to sets of transitions $A$ and $B$ counts the respective numbers of transition occurrences of $A$ and $B$, respectively. Using this generalization, we can compare the synchrony between $A$ and $B$ with its synchronic distance $\sigma(A,B)$.

For 1-safe nets, $\sigma(A,B)$ does never exceed the synchrony between $A$ and $B$, if it exists: Assume that $A D^m B$. Then there is a path $A = A_0 A_1 \ldots A_m = B$ such that, for $i \in \{0, \ldots, m-1\}$, $A_i$ and $A_{i+1}$ are the pre-set and the post-set of a place. Let us denote by $\#(X,\sigma)$ the number of transition occurrences of transitions from $X$ in $\sigma$. By 1-safeness, for each occurrence sequence $\sigma$ we have $\#(A_i,\sigma) - 1 \leq \#(A_{i+1},\sigma) \leq \#(A_i,\sigma) + 1$. Hence

$$\#(A_0,\sigma) - m \leq \#(A_m,\sigma) \leq \#(A_i,\sigma) + m,$$

whence $\sigma(A,B) = \sigma(A_0, A_m) \leq m$.

Coming back to the example from Petri shown in Figure 3, we can observe that the synchrony between transitions only depends on the structure of the net, whereas the synchronic distances between transitions also depend on the initial marking. In Figure 4, we see on the left the same net with only one token on the outer circle. The only behavior of this net is given by the occurrence sequence $af edcbaf\ldots$ and its finite prefixes, whence the synchronic distance between $a$ and $d$ is one.

On the right, we have three tokens on the outer circle and three tokens on the inner circle. As a result, we have (among many others) the following occurrence sequence: $af e d c b a f b a f \ldots$ and its finite prefixes, whence the synchronic distance between $a$ and $d$ is one. The sequence contains three occurrences of transition $a$, whence for this initial marking both the synchrony between $a$ and $d$ and the synchronic distance between $a$ and $d$ are equal to 3. Hence in fact the synchrony provides the exact value of synchronic distance, but only for a particular initial marking.

In the complete version of this contribution we study net classes for which this is always the case, i.e., for which, given any two transition sets, the synchrony between these sets coincides with their synchronic distance.

![Fig. 4. The example from Figure 3 with different markings](image-url)
5 Conclusions

We have discussed an early result of structure theory of Petri nets: the relation between the structurally defined synchrony between two transitions and the behaviorally defined synchronic distance, which very clearly shows the merits of structure theory. Petri did not present any theorems relating synchrony and synchronic distance, and no other means to calculate synchronic distance from the structure of a net. However, many years later it was shown by various authors that, for particular classes of Petri nets, the existence of a finite synchronic distance between two sets of transitions can be deduced from the incidence matrix of the net and that, in some cases, the exact value can be deduced from the initial marking (see e.g. [7]).

There are analogue results in other areas of structure theory. For example, T-semiflows are multisets of transitions that, fired in any order, lead from a marking to itself. Every such cyclic occurrence sequence has a T-semiflow as Parikh-vector. However, given a T-semiflow, an according cyclic occurrence sequence only exists for particular initial markings. Similarly, P-semiflows can prove boundedness of nets, but a net can be bounded for some markings and unbounded for other markings, whence P-semiflows cannot be used to decide boundedness for a particular marking.

The notion of synchrony can be extended in various ways, and for different classes of nets. For example, the relation between input and output transitions of a general (unbounded) place of a Petri net has a different interpretation: output transitions cannot fire infinitely often without the occurrence of an input transition, no matter how many tokens the place carries initially. This type of fairness relations have been defined in terms of Temporal Logics. For similar behavioral notions within Petri net theory, see for example [6].

References

Stochastic Petri Nets: A Structured Approach to the Performance Evaluation of Complex Systems

Gianfranco Balbo
Dipartimento di Informatica
Università di Torino
Torino, Italy

Stochastic Petri Nets (SPNs) are among the most widely used formalism for modeling and analyzing complex systems with concurrency and synchronization.

Queuing networks, where customers move between stations to request service, gained widespread acceptance in the 1970s as a compact formalism well suited for studying the behaviors of complex systems when the focus of the analysis was the competition for the use of shared resources and for the consequent arising of congestion situations. An initial list of possible scheduling policies (defining the order in which customers are served) and routing policies (defining where a customer goes after his current service has been completed) were sufficient to describe systems, for which the most important goal was dimensioning (e.g., deciding how many CPUs and disks where needed to obtain an acceptable response time while maintaining a high utilization). However, as systems and analysis needs grew complex, the ease of specification and solution offered by these ad-hoc formalisms and their specialized algorithms could not offset their inability to model many increasingly common situations. For example, sophisticated scheduling policies, non-memoryless routing mechanisms, fork-and-join behaviors, or blocking and operational dependencies among different components cannot be modeled by a classic queueing network often requiring also the addition of textual annotations that make the construction of the mathematical model difficult and error prone.

What was needed was a general purpose high-level formalism that could describe these and future systems in a convenient manner. In the 1980s stochastic extensions to Petri Nets (SPNs) were introduced as a viable solution to this problem, and we believe that they are still an excellent choice several decades later.

After some initial reluctance to accept this new formalism, SPNs are now widely used for performance evaluation of many practical systems by researchers with considerably different backgrounds. Important features that made SPNs popular for performance evaluation are: Graphical representation - SPNs have a clear, precise, and compact semantic, but also a graphical representation which helps discussing and communicating models in an intuitive manner; Generality - SPNs can model complex systems independently of the application field, and are not limited to choose from sets of predefined building blocks.
as required by more specialized formalisms: **Fidelity** - The automatic construction of the underlying stochastic model relieves the analyst from the burden of specifying it by hand, ensuring him that the stochastic model is truly equivalent to the specification of the system, and avoids errors due to misinterpreting or oversimplifying details of the system; **Structural analysis** - The analysis of the PN obtained from the SPN by ignoring timing and probabilistic information can provide valuable insight into the structure of the system; **Performance Evaluation** - The mathematical model obtained from the translation of the SPN can be conveniently analyzed using numerical and simulation techniques implemented in many available tools.

Arguably the most successful extension of classical (untimed) Petri Nets (PN), and the most appropriate for performance and analysis, is **stochastic PNs** (SPNs), where the time between the enabling a PN transition and its firing is a random variable. If these *firing times* are exponentially distributed (or, more generally, have continuous phase-type distributions), the SPN defines a CTMC whose state space is isomorphic to that of the untimed model. This is an important advantage as the many results and algorithms for logical (structural) analysis of PNs can be applied to SPNs in this case, yielding important insights into the structural or logic aspects of the system behavior, such as the presence of deadlocks, the reachability of certain conditions, the liveness of portions of the system, or the boundedness of the state space.

This ability of performing logical verification is quite new for performance studies, mostly because the formalisms traditionally used for these purposes are limited, while SPNs can describe much more accurately the behavior of complex systems, which may have logical design flaws worth discovering prior to performing a timing and probabilistic analysis.

Many SPN extensions have been proposed in the literature, building upon basic SPNs principles with the aim of simplifying the construction of models for complex systems without completely loosing the connection between the timed model and the untimed one.

During this talk, we will discuss the cases of the *generalized SPNs* (GSPNs) and of the *Stochastic Symmetric Nets* (SSNs)- previously known as *Stochastic Well-formed Nets* - in which the relationships between the timed and untimed models has been explicitly exploited also at the level of the solution methods used to compute performance indices.

Despite the extensive amount of research conducted in recent years to improve the efficiency of the performance evaluation methods exploiting results coming from a structural analysis of the untimed models, many open questions remains that will be addressed at the end of the talk. In particular, problems related to the construction of a model via de composition of sub-models will be discussed with reference to the solution techniques that can be used in these cases to exploit the structure of the model also during the performance evaluation performed for studying both the transient and steady state behaviors of these systems.
An Algorithmic View on Continuous Petri Nets

Serge Haddad

LSV, CNRS & ENS Paris-Saclay & Inria, France

1 Introduction

From Petri nets to continuous Petri nets. Continuous Petri nets (CPN) were introduced in [2] by considering continuous states (specified by nonnegative real numbers of tokens in places) where the dynamics of the system is triggered either by discrete events or by a continuous evolution ruled by the speed of firings. In the former case, such nets are called autonomous CPNs while in the latter they are called timed CPNs. In both cases, the evolution is due to a fractional transition firing (infinitesimal and simultaneous in the case of timed CPNs).

Modelling and analysis with CPNs. CPNs have been used in several significant application fields like fault diagnosis, biological regulatory networks, water distribution systems, traffic in urban networks, etc. While several analysis methods have been developed for timed CPNs, there is no hope for fully automatic techniques in the general case since standard problems of dynamic systems are known to be undecidable even for bounded nets [5]. Contrary to the timed case, the analysis of autonomous CPNs (that we simply call CPNs in the sequel) appears to be less complex than the one of discrete Petri nets. In [4], exponential time decision procedures are proposed for the reachability problems for general CPNs. In [6] assuming additional hypotheses on the net, the authors design polynomial time decision procedures for reachability and boundedness. In [5], deadlock-freeness and liveness are shown to belong in coNP. These procedures are based on structural characterisations of the properties.

This presentation. Here I will give a brief overview of results related to the complexity of decision problems of CPNs and their application to (discrete) Petri nets, obtained with several groups of researchers [5,3,1]. In particular I would thank Professor Manuel Silva that invited me at Zaragoza in 2006 and stimulated my interest for CPNs.

2 Presentation and properties

Syntactically a continuous Petri net (CPN) is an ordinary Petri net defined by $P$, a finite set of places, $T$ a finite set of transitions with $P \cap T = \emptyset$, and $\text{Pre}, \text{Post} \in \mathbb{N}^{P \times T}$, the backward and forward incidence matrices, respectively. As usual, $^t\!\!\!\!\!\!\!$ denotes the set of places with $\text{Pre}(p, t) > 0$ (resp. $\text{Post}(p, t) > 0$). The incidence matrix $C = \text{Post} - \text{Pre}$ plays here an important role. In CPN, place markings consist of real (or rational when given as input) numbers and in which

* Supported by ERC project EQualIS (FP7-308087).
transitions may be fired a fractional number of times. Formally, a marking of a
continuous Petri net is a vector \( m \in \mathbb{R}^P_+ \). Let \( t \in T \), the \textit{enabling degree} of \( t \)
with respect to \( m \) is a function \( \text{enab}(t, m) \in \mathbb{R} \cup \{\infty\} \) defined by:

\[
\text{enab}(t, m) \triangleq \begin{cases} 
\min\{m(p)/\text{Pre}(p, t) : p \in \cdot t\} & \text{if } \cdot t \neq \emptyset, \\
\infty & \text{otherwise.}
\end{cases}
\]

We say that \( t \) is \textit{enabled at} \( m \) if \( \text{enab}(t, m) > 0 \). When no transition is enabled at
\( m \), we say that \( m \) is a \textit{dead marking}. If \( t \) is enabled it may be \textit{fired} by any amount
\( q \in \mathbb{R}^+ \) such that \( 0 \leq \alpha \leq \text{enab}(t, m) \), leading to a new marking \( m' \) such that for
all places \( p \in P \), \( m(p) \triangleq m(p) + \alpha \cdot C(p, t) \). In this case, we write \( m \xrightarrow{\alpha \cdot t} m' \). Let
\( \sigma = \alpha_1 t_1 \cdots \alpha_k t_k \in (\mathbb{R}^+ \times T)^* \). We say that \( \sigma \) is a \textit{firing sequence} enabled at
\( m \) and that \( m' \) is reachable from \( m \). The \textit{Parikh image} of the firing sequence \( \sigma \)
is the vector \( \pi(\sigma) \in \mathbb{R}^T_+ \) such that \( \pi(\sigma)(t) \triangleq \sum_{i=1}^{k} \alpha_i \).

\textbf{Example 1.} Let us consider the net of Figure 1 equipped with initial marking
\( m_0 = 1 \cdot p_1 + 1 \cdot p_3 \). Only transition \( t_1 \) is enabled at \( m_0 \) with enabling degree 1.
Assume that one fires \( 0.6 \cdot t_1 \), then ones reach \( m_1 = 0.4 \cdot p_1 + 0.6 \cdot p_2 + 1 \cdot p_3 \).
Due to place \( p_3 \), the enabling degree of \( t_2 \) at \( m_1 \) is 0.5 and \( \sigma = 0.5 t_2 0.5 t_3 \) is a
firing sequence enabled at \( m_1 \) leading to \( m_2 = 0.4 \cdot p_1 + 0.6 \cdot p_2 + 0.5 \cdot p_3 \).

\begin{center}
\begin{tikzpicture}
\node (p1) at (0,0) [shape=circle,fill,draw] {};
\node (p2) at (1,0) [shape=circle,draw] {};
\node (p3) at (2,0) [shape=circle,fill,draw] {};
\node (p4) at (1,-1.5) [shape=circle,draw] {};
\node (t1) at (0.75,0.5) [shape=circle,draw] {};
\node (t2) at (1.75,0.5) [shape=circle,draw] {};
\node (t3) at (1.75,-0.5) [shape=circle,draw] {};
\draw (p1) -| (t1) -| (p2);
\draw (p1) -| (t2) -| (p3);
\draw (p2) -| (t3) -| (p4);
\end{tikzpicture}
\end{center}

\textbf{Fig. 1.} A (continuous) Petri net.

Most of the properties defined for Petri nets can also be defined and studied
in the framework of CPN. In this short overview, we only focus on two key
decision problems. Given a CPN \( N \) and rational initial and final markings \( m_0 \)
and \( m_f \), the \textit{reachability problem} asks whether \( m_f \) is reachable from \( m_0 \) in \( N \).
Given a CPN \( N \) and a rational initial marking \( m_0 \), the \textit{deadlock problem} asks
whether there exists a dead marking \( m_f \) reachable from \( m_0 \) in \( N \).

\textbf{Example 2.} The reachability set of the net of Figure 1 is presented in Figure 2
where marking \( m(p_1) \) can be recovered by the linear invariant \( m(p_1) = 1 - m(p_2) \). It consists of the interior of a polyhedron enlarged with some of its faces
or their interiors. In fact, this characterisation is valid for any CPN. Here the
reachability set includes neither the markings of the polyhedron with \( m(p_2) = 0 \)
(except the initial marking) nor the markings with \( m(p_3) = m(p_4) = 0 \). Thus all
reachable markings (except the initial marking) fulfill \( m(p_2) > 0 \) and \( m(p_3) > 0 \) or \( m(p_4) > 0 \) which implies that either \( t_2 \) or \( t_3 \) is enabled. So the net is deadlock-
free.
3 Analysis

The design of efficient algorithms for checking the properties of a CPN follows a two-step scheme. First, one characterises a property by a structural condition. Then one provides a tricky algorithm to verify whether this structural characterisation. We illustrate this principle for reachability analysis.

3.1 Reachability in CPN

Let us exhibit some necessary conditions for reachability. Assume that $m_0 \xrightarrow{\sigma} m_1$. Then the first necessary condition is based on the incidence matrix:

$$\exists v \in \mathbb{R}^{|T|}_{+} \quad m_1 = m_0 + C \cdot v$$  \hspace{1cm} (1)

since $\pi(\sigma)$ is a solution of this equation.

The two other conditions are based on the notion of firing set. A firing set is a subset of transitions which is the support of a firing sequence: $T' \in fs(N, m_0)$ if there exists $\sigma$ a firing sequence enabled at $m_0$ such that $T' = \{t \mid \pi(\sigma)(t) > 0\}$ (i.e. $T' = \text{Supp}(\pi(\sigma))$).

Example 3. As seen previously, $T$ is a firing set of the net of Figure 1 as well as $\{t_1\}$ or $\{t_1, t_2\}$, but neither $\{t_1, t_3\}$ nor $\{t_2, t_3\}$ are firing sets.

So we can strengthen our necessary condition as:

$$\exists v \in \mathbb{R}^{|T|}_{+} \quad m_1 = m_0 + C \cdot v \land \text{Supp}(v) \in fs(N, m_0)$$

Another condition relies on the reverse net $N^{-1}$ which consists in reverting the edges of the net and reverse sequence $\sigma^{-1}$ which consists in reverting the firing order. A straightforward reasoning establishes that $m_0 \xrightarrow{\sigma_{N^{-1}}} m_1$ iff $m_1 \xrightarrow{\sigma^{-1}_{N^{-1}}} m_0$. So we can apply the second condition on the reverse net and observing that $\pi(\sigma) = \pi(\sigma^{-1})$, one gets as next strengthening:

$$\exists v \in \mathbb{R}^{|T|}_{+} \quad m_1 = m_0 + C \cdot v \land \text{Supp}(v) \in fs(N, m_0) \land \text{Supp}(v) \in fs(N^{-1}, m_1)$$
Example 4. Let us check that in the net of Figure 1, $1 \cdot p_2$ is not reachable. The only vector fulfilling the linear equation is $1 \cdot t_1 + 1 \cdot t_2 + 1 \cdot t_3$ whose support is a firing set of the net. However this support is not a firing set of $N^{-1}$ since one can only fire (some fraction of) $t_1^{-1}$ from $1 \cdot p_2$ in $N^{-1}$.

We now study how to exploit this characterisation. The first issue is to check whether a subset of transitions is a firing set. Given a firing sequence $\sigma$ and $\alpha > 0$ define $\sigma \cdot \alpha$ the sequence where all fractions are multiplied by $\alpha$. Assume that $\sigma$ and $\sigma'$ are firing sequences, a straightforward reasoning establishes that $(0.5\sigma)(0.5\sigma')$ is a firing sequence. Thus firing sets are closed by union and so there exists a maximal firing set denoted $maxfs(N, m_0)$. $maxfs(N, m_0)$ can be computed in polynomial time as follows. The algorithm manages a subset of places $Markable$, initially set to the support of $m_0$ and a subset of transitions $Firable$, initially set to the empty set. It proceeds iteratively by enlarging $Firable$ with transitions $t$ such that $t \subseteq Markable$ and then enlarging $Markable$ with $t^*$. It stops when no more places or no more transitions can be added and returns $Firable$.

Example 5. Let us apply this algorithm on the net of Figure 1. On initialisation, $Markable = \{p_1, p_3\}$ and $Firable = \emptyset$. After the first iteration, $Markable = \{p_1, p_2, p_3\}$ and $Firable = \{t_1\}$. After the second one, $Markable = \{p_1, p_2, p_3, p_4\}$ and $Firable = \{t_1, t_2\}$. After the third one, $Markable = \{p_1, p_2, p_3, p_4\}$ and $Firable = \{t_1, t_2, t_3\}$. Since $Markable$ is unchanged, the algorithm stops and returns $\{t_1, t_2, t_3\}$.

This algorithm can be easily adapted to check whether $T' \in fs(N, m_0)$. One builds in linear time the net $N_{T'}$ restricted to transitions of $T'$ and checks whether $T' = maxfs(N_{T'}, m_0)$. This shows that the reachability problem belongs to $NP$: one guesses a subset of transitions $T'$ and checks whether (1) there exist $v$ such that $Supp(v) = T'$ and $m_1 = m_0 + C \cdot v$ using linear programming, (2) $T' \in fs(N, m_0)$, and (3) $T' \in fs(N^{-1}, m_1)$.

In fact there is a better algorithm which proceeds by a greatest fixed point computation. Assuming that $m_1 \neq m_0$, it manages a subset of transitions $T'$ initially set to $T$. Then the iterative step consists to:

- Find the maximal support of a solution of the equation $v \in \mathbb{R}^{T'_+}$
  $m_1 = m_0 + C_{T'} \cdot v$ where $C_{T'}$ is $C$ restricted to columns indexed by $T'$.
  If there is no solution, the algorithm returns false.
  Otherwise $T'$ is this maximal support;
- $T'$ is set to $maxfs(N_{T'}, m_0)$;
- $T'$ is set to $maxfs(N_{T'}^{-1}, m_1)$.

If $T'$ is unchanged by the last two steps, the algorithm returns true.

Example 6. Let us apply this algorithm on the net of Figure 1 with target marking $1 \cdot p_2$. During the first iteration it finds the solution of the linear equation system $1 \cdot t_1 + 1 \cdot t_2 + 1 \cdot t_3$ leading to $T' = \{t_1, t_2, t_3\}$. $T'$ is unchanged by the assignement $maxfs(N_{T'}, m_0)$ but is set to $T' = \{t_1\}$ by the assignement $maxfs(N_{T'}^{-1}, m_1)$. During the second iteration, it does not find any solution of the linear equation system and so returns false.
3.2 Deadlocks in CPN

A marking $m$ is a deadlock if there is a subset of places $P'$ such that for all transition $t$, $\bullet t \cap P' \neq \emptyset$ and for all $p \in P'$, $m(p) = 0$. So by exploiting the characterisation of reachability, one obtains a characterisation of existence of deadlocks.

$$\exists \mathbf{v} \in \mathbb{R}_{\geq 0}^T \exists \mathbf{m}_1 \in \mathbb{R}_{\geq 0}^P \mathbf{m}_1 = \mathbf{m}_0 + C \cdot \mathbf{v}$$
$$\land \text{Supp}(\mathbf{v}) \in \text{fs}(N, \mathbf{m}_0)$$
$$\land \text{Supp}(\mathbf{v}) \in \text{fs}(N^{-1}, \mathbf{m}_1)$$
$$\land \forall t \in T \exists p \in \bullet t \mathbf{m}_1(p) = 0$$

This characterisation shows that the deadlock problem belongs to NP. One guesses a set of places $P'$ intended to be the support of the dead marking and a set of transitions $T'$ intended to be the support of the Parikh image of the firing sequence leading to this dead marking. Then:

- one checks whether for all $t \in T$, $\bullet t \not\subseteq P'$;
- one checks whether $T' \in \text{fs}(N, \mathbf{m}_0)$;
- one checks whether $T' \in \text{fs}(N^{-1}, P')$. The change of definition for $\text{fs}$ is justified since the firing set only depends on the support of the initial marking;
- one checks there is a solution of the linear equation system over unknowns vectors $\mathbf{v}$ and $\mathbf{m}_1$: $\mathbf{m}_1 = \mathbf{m}_0 + C \cdot \mathbf{v}$ with $\text{Supp}(\mathbf{v}) = T'$ and $\text{Supp}(\mathbf{m}_1) = P'$.

Contrary to reachability there is little hope to find a polynomial time algorithm since the deadlock problem is in fact NP-complete. The proof is done via a reduction from the 3SAT problem. Figure 3 illustrates this reduction for the 3CNF formula $\varphi = \neg x_1 \lor \neg x_3 \land (x_1 \lor \neg x_2 \lor x_3) \land (x_2 \lor \neg x_3)$. Let us give some intuition about this reduction on the example. Transitions $t_i$ (resp. $f_i$) puts tokens to places corresponding to literals $\neg x_i$ (resp. $x_i$) occurring in a clause. Transition $nc_j$ corresponds to the $j$th clause and is firable if all places corresponding to its literals are marked. If $\varphi$ is satisfiable then firing one unit of $t_i$ or $f_i$ according to the interpretation $\nu$ satisfying $\varphi$ ensures that (1) places $b_i$ are unmarked and (2) for all $j$ there is an unmarked place of $\bullet nc_j$. So one has reached a dead marking. One can show that there is no deadlock when $\varphi$ is not satisfiable.

4 Application to Petri nets

As discussed in the introduction, the most relevant decision problems of Petri nets are decidable but with high complexity. Given a net $N$, an initial marking $\mathbf{m}_0$ and a marking $\mathbf{m}_c$, $\mathbf{m}_c$ is coverable from $\mathbf{m}_0$ if there is a reachable marking $\mathbf{m} \geq \mathbf{m}_c$. The coverability problem is EXPSPACE-complete and there exist several algorithms to solve it.
Fig. 3. From 3SAT to existence of deadlocks.

The **backward algorithm** operates as follows. It manages a finite set $MinCov$ of markings initially set to $m_c$. At the $i$th iteration, $MinCov$ fulfills the following property:

$$\{m | \exists m' \in MinCov \ m \geq m'\} = \{m | \exists j \leq i \ \exists \sigma \in T^j \ m \stackrel{\sigma}{\rightarrow} m' \geq m_c\}$$

An iteration consists to:

- for all $t \in T$ and all $m \in MinCov$ enlarge $MinCov$ with $m'$ defined by:
  for all $p \in P$, $m'(p) = \max(\text{Pre}(p,t), m'(p) - C(p,t))$
- to delete all non minimal markings of $MinCov$;

Due to well quasi-order theory, $MinCov$ stabilises and on stabilisation,

$$\{m | \exists m' \in MinCov \ m \geq m'\} = \{m | \exists \sigma \in T^* \ m \stackrel{\sigma}{\rightarrow} m' \geq m_c\}$$

Thus the algorithm returns true if there exists $m \in MinCov$ such that $m \leq m_0$.

By definition of the firing rule in CPN, a marking is coverable in CPN if it is coverable in the discrete Petri net (but not vice versa). So one may design a simple heuristic that on-the-fly prunes markings $m \in MinCov$ that are not coverable in the CPN. Since coverability in CPN is done in polynomial time, the additional computation is worthwhile. This heuristic is shown to be very efficient on benchmarks provided by the other tools [1].
References

Recalling some decidability and complexity results on (structurally) constrained and general P/T nets

David de Frutos Escrig
defrutos@sip.ucm.es
Dpto. Sistemas Informáticos y Computación - Facultad CC. Matemáticas
Universidad Complutense de Madrid, Spain.

Abstract. I recall several important milestones in the development of algorithms to decide (structural) properties of (structured) Petri nets. Although I have separated in two sections those related with the Structural Theory, and those for general nets, after some recent results by Leroux on the (general) reachability problem, I conclude that now we can state that “structure is everywhere”, when we are coping with (finite) nets.

1 Introduction

Place/transition (P/T) nets, \( N = (P,T,F) \), define the local variability of a system where transitions \( t \in T \) can be fired whenever their precondition places (such that \((p,t) \in F\)) are activated. When a transition is fired the state changes by deactivating its preconditions and reactivating its postconditions (such that \((t,p) \in F\)). But in order to precise this definition we need to state when a place is activated. This is made by means of markings, that are functions \( M : P \rightarrow \mathbb{N} \). Then, activated means that \( M(p) > 0 \), and when deactivating (resp. reactivating) we substract (resp. add) one unit to that value. In this way the dynamics of systems is defined by the (local) firing rules of transitions, but we also need some initial marking from which the story wil start.

In the beginning we could expect that this initial marking should not be so important in order to establish the behaviour of the defined system and its properties. This hurried reasoning would be somehow supported by the monotonicity of P/T nets, since it is clear that any behaviour from some marking \( M_0 \) remains valid when starting from some bigger marking \( M'_0 \geq M_0 \). But, on the contrary, it is clear that starting from such a marking \( M'_0 \) we will be able to get some totally new behaviours. Moreover, even in the former case, when firing some sequence of transitions \( \sigma \) from \( M'_0 \), the reached state \( M'_1 \) will not certainly be the same that when firing it from \( M_0 \). Instead, we would have the algebraic invariant \( M'_1 - M_1 = M'_0 - M_0 \).

At the same time, it is not clear at all that “the more” behaviours we have, “the better”. When we have more options, also the risks to get some undesired (wrong) behaviour increase. The consequence is that, in general, the properties of the systems defined by a net do not only depend on this underlying net, but
also on the precise initial marking. We can suppose that this flexibility of P/T nets was not intended when they were defined stressing their (static) structure, but now we perfectly know that in the general case the considerations about the initial state could be equally important in order to guarantee that our systems will satisfy the required properties.

Even so, when the main properties of P/T nets were studied, starting with reachability, it was certainly a big surprise to find that a formal proof of its decidability had to wait for several years, and when discovered it was extremely complicated. That decidability was definitely expected, once the “loss” of Turing completeness implied by the monotonicity of P/T nets was taken into account. However, the dependency of the initial marking sometimes produce extremely involved reachability states, even for some “apparently simple” (and small) nets.

But as stated above, in the beginning that complexity was not expected, and several researchers tried to induce “any” structure at the reachability space from the structure (as a graph) of the underlying net. This was made by trying to characterize this structure in a “step by step” way. In this way, several simple classes of nets were considered: S-nets and T-nets, that totally forbid either synchronization or conflicts, trying to avoid the situations that could make more complicated the behaviour of the corresponding systems; and later free-choice, that tried to encompass these two classes in a symmetric way. Informally, only very simple “local components” are allowed, and in this way the expected simplicity was obtained and proved in some cases, although in some others it was finally proved that even for these very restrictive classes, some properties remain very difficult to analyze.

There was another complementary approach, looking for nets such that the desired properties will be uniformly guaranteed, whichever the initial marking is. We also talk about “structural” properties in this case, even if in general we cannot precise “where” that structure “is hidden”. Even so, there must be something in the definition (e.g. at its structure!) of the net that makes good the systems defined “on top” of it, whichever the initial marking is. This is why we will also recall in the following some results that were obtained using this approach to “structureness”.

Once reachability was proved to be decidable for general P/T nets, but at the same very difficult to decide, the study of the restricted classes cited above remained interesting now looking for the limits in the expressive power of them, so that the analysis of the interesting properties of systems will remain affordable. Thus the obtained complexity results sometimes show that we can freely use some classes of nets, but some other times we must be careful, because in the general case the corresponding systems could remain quite difficult to analyze.

The second part of this short paper is devoted to the decidability of properties of general P/T nets. Even if the topic of this meeting is Structure Theory, I had two reasons to include some results that are not directly related to “the structure” of nets. The first is totally personal, my own works in decidability of properties of Petri nets do not consider (in a direct way) any structural constraint, but even so I wanted to cite at least some of them here. The second
reason is more solid: we will see that based on some of these results we can now
claim that when we are working with some finite mechanism to define possibly
infinite behaviours, as Petri nets are, “there will be some kind of structure every-
where”. In particular, the new proofs of decidability of reachability for general
nets by Jerome Leroux try to enlight the structure of the reachability space of
any P/T net, and support this structure by means of an ideal structure defined
by some underlying well quasi order.

My personal interpretation of these recent results is that finally it has been
really established a connection between the restricted cases that impose some
structure, and the general case, where now the structure that makes possible
simpler proofs. Besides, as stated by Leroux, now we can expect that some
adequate variants of these proofs could be applied to other more general classes
of nets, as unordered data Petri nets.

2 Decidability and complexity issues in Structure Theory

By no means you can expect here a full survey of hundreds of papers devoted
to this subject in the past, and for the same reason, I am not claiming that the
papers that I will cite in the following are the most important contributions in
this area, but simply they represented important milestones for my own learning.

I think that I must start going 40 years back, when Jones, Landweber and
Lien published the first monography devoted to the study of complexity of prob-
lems in Petri nets [12]. In particular, they proved that reachability for conflict
free nets is NP-hard, thus showing that the problem is likely to be “very difficult”,
even for this highly constrained class of nets. In fact, it was already known that
reachability was decidable for this class of nets, but by means of an algorithm
that was extremely costly. Also, for 1-conservative free choice, reachability re-
mains “difficult”, although in this case the problem was proved to be P-SPACE
complete. More complexity results for other properties such as boundedness and
persistency were also presented.

Landweber and Robertson focused on conflict free and persistent nets in [13].
In this case, it is important to note that they introduced a new feature that is the
key to develop many algorithms to decide properties: semilinearity. In particular
they proved that the reachability space of a persistent net is always semilinear.
With respect to conflict free nets, they provided an algorithm to decide bounded-
ness, that works in EXPTIME. To conclude their paper they shown that a single
pair of non-persistent transitions is enough to get the reachability space of any
net, so that any partial proof of persistency could be absolutely useless, when
we are trying to show that our net is “regular enough", so that its properties
could be checked in an efficient way.

Just two years later, H.Müller proved that reachability was decidable for
persistent nets [20]. The main idea was to “adapt” the coverability tree by con-
sidering the periods that can be fired at each state. A period is a reproducible
sequence, because its effect is non-negative at any place. The algorithm then
applies a decomposition mechanism by means of which the reachability tree can be represented in a certain manageable finite way.

Next year E. Mayr continued the study of persistence, by showing that it is decidable [19]. It is true that persistency is not technically a structuring property, since it is defined directly on the reachability space, but its study has been connected in the past to some structural properties that guaranteed it, and this is why I have included its study in this section. Moreover, once again, the reachability set becomes semilinear for persistent nets, and it is by trying to construct a finite presentation of that set that the algorithm works.

Let us next jump ten years to the future, to find the contributions by Howell, Rosier and Yen, that continued the research on conflict free nets, showing in [9] that reachability was in \( NP \), thus completing the proof of NP-completeness for this problem. Their algorithm used an algebraic characterization “guessing” a linear programming problem whose solution would prove the checked reachability. Once again semilinear sets play a key role at their proof. The paper also studies the containment and equivalence problems comparing the reachability spaces of two nets, showing in these cases that they are \( \Pi^p_2 \)-complete. Liveness and a few fairness properties were proved to be (only) \( P \)-complete in [10], while some other fairness properties remained \( NP \)-complete. Finally, reachability for bounded conflict free nets was solved in polynomial time.

In [11] this group studied normal nets, that are the structural version of sinkless nets. They were introduced because they were proved to be much more flexible than conflict free nets. They are defined in terms of the circuits and sinks in the structure of the net, but considering whether these sinks are marked, so that the original class of sinkless nets is not structural. Besides the modeling power of these nets, they show that reachability remains \( NP \)-complete, while boundedness is \( co-NP \)-complete, as it is also the case for the normality check.

There are many important decision algorithms based on the algebraic characterization of nets. In particular deadlocks and traps can be characterized by means of \( P \)-semiflows. This was fully proved by Esparza and Silva in [8], after a preliminary result by Lauttenbach. Several applications of the derived algorithm were presented at that paper, showing in particular that liveness of bounded free choice nets can be decided in polynomial time. As a matter of fact, there has been an enormous number of publications developing algorithms for Petri nets based on their algebraic characterization, and many of them came from the group led by Manuel Silva in Zaragoza. For sure they would merit by themselves a detailed monography, but here I have prefered to make a more general presentation, thus talking about the different techniques that have contributed to the development of these algorithms.

Another popular survey that also presents some new results on 1-safe nets, and recalls others previously known for particular subclasses, is [3]. Liveness, reachability and the deadlock problem were proved to be \( PSPACE \)-complete, while the latter is only \( NP \)-complete, when we restrict ourselves to the class of free choice nets. It was thought for a long time that reachability for live, safe free-choice nets, certainly a quite restrictive class, should not be “too complicated”,
but Esparza shown that this was not the case [7]. The reduction from CNF-SAT was made in two steps, using as intermediate step an ad-hoc class of constrained systems that forced certain number of executions of some transitions at the allowed sequences to reach the desired target marking. The paper concludes by presenting an extension to the bounded case, although its correctness is based on a “difficult to check” previous result by Lee, Kumagai and Kodama.

We conclude this section with a very recent contribution by Drewes and Leroux [6], where structurally cyclic nets are analyzed. In this case, the monotonicity implies that a net is structurally cyclic iff the empty marking can be recovered by some non-empty sequence of transitions. The original general cyclicity problem is proved to be EXPSPACE-complete by means of a reduction from the reachability problem for lossy nets. On the contrary, the structural cyclicity problem is only P-complete. It is clear that the minimality character of the empty marking makes the problem much easier to analyze than in the general case.

3 Some Decidability results for General nets

In this short section I start by briefly recalling my own result on the decidability of home states [4], that later was expanded to cover home spaces in a joint work with C.Johnen [5]. Once again, semilinear sets played an important role, together with a finite decomposition theorem which has a compactness flavour. The algorithm makes calls to the reachability algorithm, and it seems that this dependency cannot be “essentially” removed. In fact, very recently Best and Esparza has shown that the existence of home states is decidable but at least as hard as the reachability problem [2].

Reversibility is the particular instance of the home state property where the target marking is just the initial marking of the system. Leroux proved in [14] that this problem is only EXPSPACE-complete and therefore it is reasonable to expect that it is strictly easier than the general home state problem. We need to mention here that as early as in 1977 Araki and Kasami “proved” [1] that reversibility was decidable, even before reachability was proved to be too, and that meant that they never checked (general) reachability all along their proof. It is curious to see that Presburger sets appear several times along their reasonings, exactly as it is the case for the recent results by Leroux [16]. Unfortunately, when I tried to trace their proofs, I found so many verbose arguments and also some “suspiciously too short” proofs of some auxiliary results that seem to require careful and lengthy proofs, that personally I cannot accept as proved their results at that paper, although for sure I have to credit its authors for foreseeing many years in advance some quite interesting results that were correct after all.

And next I conclude this section with a fast review of the papers where Leroux has shown along the last years that the reachability problem (and its solution) was not really so “strange” as the reading of the successive presentations of the “classical” algorithm to solve it made us to think. In [17], jointly with Schmitz, he makes an “inspired” presentation of that algorithm that illuminates the steps in it, thus trying to desmystify it. We can make a new step in order to better
understand the reachability problem, looking to the alternative proof in [15], where almost semilinear relations were introduced and combined with Presburger relations to obtain the new decidability proof. Finally, [18] provides the ideal decompositions that again are telling us that “structure is everywhere” when we consider P/T nets (for sure, due to their finiteness!).

References

Structure of Behaviour in Extended EN Systems

Jetty Kleijn

Leiden Institute of Advanced Computer Science
Leiden University
Leiden, The Netherlands

Abstract. As an example of the connections that may exist between structural and behavioural properties of Petri net models, we turn in this presentation our attention to Elementary Net (EN) Systems and the structure of their behaviour. EN systems are the classical net model to study the interaction of transitions at a certain state. In fact, for this model, the three basic relations of concurrency, causality, and conflict, are determined by the structure of the net. Starting from EN systems and their partial order semantics, we extend the set-up and consider generalised order structures. Based on a classification of these behavioural structures, we identify some structural extensions of elementary net systems.
Structuring nets and their behaviours

Maciej Koutny

School of Computing Science
Newcastle University
Newcastle upon Tyne, UK

Abstract. In this talk I will discuss a range of issues that arise when process algebras and Petri nets are combined. In particular, I will focus on compositionality of structure and behaviour, refinement, and equivalence notions. The talk will explain how one can introduce a generic algebra of nets and process expressions which is equipped with two types of equivalent semantics: a Petri net semantics based on step sequences and causal partial orders, and a structural operational semantics based on a system of derivation rules. As a concrete example of this algebraic framework I will use the Petri Box Calculus (PBC).
On the control and estimation
of discrete event systems
using Petri net structural approaches

Alessandro Giua (♭, ♫), Carla Seatzu (♫)

(♭) Aix-Marseille Universit´e, CNRS, ENSAM, Universit´e de Toulon, LSIS UMR 7296, Marseille 13397, France. Email: alessandro.giua@univ-amu.fr
(♫) DIEE, University of Cagliari, Cagliari 09124, Italy. Email: \{giua,seatzu\}@diee.unica.it

Abstract. The interest in Petri nets has grown within the automatic control community in parallel with the development of the theory of discrete event systems. In this presentation our goal is that of recalling the main features that make Petri nets a powerful model for discrete event systems. In particular we discuss how the knowledge of the net structure can be used to derive practically efficient methods for analysis and control and point out the areas where Petri nets have offered the most significant contributions.

1 Introduction

The object of the study of traditional control theory have been time-driven systems, i.e., systems of continuous and synchronous discrete variables, modeled by differential or difference equations. However, as the scope of control theory is being extended into the domains of manufacturing, robotics, computer and communication networks, and so on, there is an increasing need for different models, capable of describing systems that evolve in accordance with the asynchronous occurrence, at possibly unknown irregular intervals, of physical events. Such systems, whose states have logical or symbolic, rather than numerical, values that change in response to events which may also be described in nonnumerical terms, are called discrete event systems (DESs) and the corresponding models are called discrete event models (Cassandras and Lafortune, 2008).

As dynamical systems, discrete event systems qualify as a proper subject for control theory. Hence a fundamental issue arises: we need classes of formal models that are capable of capturing the essential features of discrete, asynchronous and possibly nondeterministic systems and that are endowed with efficient mathematical tools for analysis and control.
2 Petri nets as models for discrete event systems

We claim that Petri nets (PNs) are a powerful discrete event model and, in fact, the interest for this model has grown, within the automatic control community, in parallel with the development of the theory of discrete event systems (Holloway et al., 1997; Cassandras and Lafortune, 2008; Seatzu et al., 2012). In this presentation the goal is not that of providing a comprehensive survey of the research in this area, but rather that of giving a flavor of the structural features that make Petri nets a good model for systems theory.

Let us first point out what are the two main challenges in the application of system theory approaches to discrete event systems.

– A first problem is due to the absence of reference models. Unlike classical control theory where there are well accepted models, such input-output or state variables models that can be used in different contexts (analysis, control, etc.), in the DES domain we still find a series of different modeling formalisms, each one used to solve a number of particular problems. Examples of these modeling formalisms include finite state automata, formal languages, max-plus algebra, predicate algebra, semimarkovian processes, queuing networks, and so on.

– A second problem, which has so far represented the main obstacle to the transfer of theoretical results to real word applications, is the so called state space explosion that originates from the combinatorial nature of DESs. Consider, as an example, a system composed by $k$ modules each one having a state space of cardinality $n$: the cardinality of the state space of the overall system will be $N \leq n^k$ and the upper bound is strict in the sense that in many cases it is attainable. This exponential growth of the state space size with the number of composed systems makes all those approaches that are based on an exhaustive exploration of the state space of little practical interest in many applications.

The success of Petri nets as a discrete event model can be explained observing that they provide an answer (albeit partial) to both problems.

Petri nets can be considered as a reference model because they represent a general paradigm — with a unique underlying well defined structure — that gives rise to a large family of powerful models (David and Alla, 2005). Place/transition nets can be used to describe logical systems — possibly with an infinite state space — providing several primitives for modeling sequentiality, concurrency, rendez-vous, choice, resource allocation, etc. Labeled nets are formal languages generators that can also
be used to study problems related to observability and identification: the
class of Petri net languages are (analogously to context-free languages)
a superset of regular languages and a subset of context-sensitive lan-
guages but the introduction of inhibitor arcs extends the modeling power
of Petri nets to that of Turing machines. Models of timed nets have al-
so been defined: if the timing structure is deterministic we get nets that
can describe, among others, max-plus algebra models, while determin-
istic nets generalize semi-markovian processes and queueing networks.
Finally, more recently continuous and hybrid Petri nets combining both
event-driven and timed-driven dynamics have also be defined. Although
some of these models require their own analysis techniques that are not
applicable to all the family, there exist other approaches such as those
based on reachability, state equation, incidence matrix analysis that are
applicable to all models, thus providing a common unifying framework.

Petri nets provide several features to alleviate the obstacle created by
the state space explosion. The modular synthesis of complex models is
done through operators — such as the synchronous product also called
concurrent composition operator — that work on the net structure and
not on the reachability space (Peterson, 1981; Giua, 2012). This is an
enormous advantage, because the net structure typically grows linearly
with the number of composed modules, as opposed to the state space that
grows exponentially.

Some PN analysis techniques, such as those based on the reachability
graph, require the exhaustive enumeration of the net state space, thus
they do not fully exploit the representational advantage of this model.
However, there also exist a wide range of approaches based on structural
analysis (Colom and Silva, 1990; Silva et al., 1992), that take into account
the net structure. This has made possible the development of efficient
techniques for reachability, liveness and deadlock analysis, state estimation,
observability and diagnosis. It must be said, however, that some of
these techniques are only applicable to restricted classes of nets.

In this presentation, in particular, we will discuss a recent and quite
general approach that exploits the notion of basis marking to practically
reduce the computational complexity of solving a reachabilty problem.
The advantage of this technique is that only part of the reachability space
— i.e., the set of the so-called basis markings — is enumerated. All other
markings are reachable from a basis marking by firing only unobservable
transitions and can be characterized by the integer solutions of a system
of linear equations. This method has originally been introduced to solve
problems of state estimation under partial observation (Corona et al.,
2007) but has later been extended to address fault diagnosis (Cabasino et al., 2010), state-based opacity (Tong et al., 2017) and general reachability problems (Ma et al., 2017).

3 Structural approaches for control and estimation

In the last two decades an increasing number of researchers from the automatic control community have devoted their effort to the study of Petri nets: for a recent survey see Seatzu et al. (2012). Major contributions have appeared in the following two main areas of discrete event systems and will be reviewed in this presentation.

Control. This is the core issue in systems theory and consists in applying suitable control inputs to a plant to ensure its behavior satisfies a given specification. For Petri nets a typical solution is that of designing a supervisor that disables the firing of some transitions as a feedback law of the plant’s observed behavior or state: the controlled system (plant and supervisor) is also called closed-loop system. The most interesting approach to the control of discrete event systems, that has directly or indirectly shaped much of the research in this area, is Supervisory Control Theory (SCT), originated by the work of Ramadge and Wonham (1989). Petri nets can be used within this framework for enforcing both language specifications and state specifications. Of particular interest are a class of state specifications called Generalized Mutual Exclusion Constraints (GMECs) (Giua et al., 1992) that can essentially be solved by structural analysis and, thanks to their simplicity, have been studied by a large number of researchers: see (Iordache and Antsaklis, 2005) for a survey.

Estimation and related problems. The estimation of partially observed systems, i.e., systems whose complete evolution can only be reconstructed from the measurement of a set of outputs, is considered the dual problem of control and is also a fundamental issue in systems theory. The problem of estimating the state or, more generally, the evolution of a dynamical system, has attracted the attention of several researchers in the DES community. In this framework possible choices of outputs are event labels (consider Mealy automata) or state labels (consider Moore automata). Labeled Petri nets, where labels are associated to transitions, have been successfully used as models of partially observable DESs but often the token content of some places is also assumed to be measurable (Ramirez-Trevino et al., 2003). Several are the motivations behind estimation. The most natural one in a systems theory setting is the need to implement feedback control but other motivations have also driven much
of the research in the area of DESs in general and Petri nets in particular. Among the problems where structural approaches have been proved successful we recall the following.

- **Fault diagnosis**, i.e., detecting the occurrence of a fault. In the DES framework it is commonly assumed that both the *nominal model* and the *fault model* of the system are given. Following the theory developed by Sampath *et al.* (1995), faults are usually modeled by unobservable events whose occurrence must be reconstructed based on the system’s observation: thus it is not surprising that many approaches derived for state estimation can also be tailored for diagnosis. In particular, we recall the works of Benveniste *et al.* (2003), Wu and Hadjicostis (2005), Basile *et al.* (2009), Dotoli *et al.* (2009), Cabasino *et al.* (2010), Cabasino *et al.* (2011).

- **Computation of synchronization sequences**, i.e., sequences of inputs that drive the system to a unique final state independently of the initial state and do not require the observation of the systems outputs. This is a particular instance of the more general class of testing problems, which have been usually addressed using automata models. Preliminary results showing the advantages of Petri net structural approaches have been recently presented by Pocci *et al.* (2014).

- **Opacity verification and enforcing**. A system is said to be opaque if a given secret behavior remains opaque (uncertain) to an intruder which can partially observe its evolution through a mask. Interesting state-based opacity properties, as defined in Petri nets by Bryans *et al.* (2006), can be efficiently verified for bounded nets using structural approaches, as shown by Tong *et al.* (2017).

**References**


Making Petri nets friendlier to engineers

Jordi Cortadella

Department of Computer Science
Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract. For more than 30 years, Petri nets have been used for the specification, synthesis and verification of asynchronous circuits [2]. However, there is still a perpetual reluctance to adopt this formalism outside academia. On one hand, many electrical engineers, either in the digital or analog world, are not familiar with the background required to understand the fundamentals of concurrency. On the other hand, academicians sometimes work at a level of abstraction far from the actual problems engineers need to solve.

This talk will review the main formalisms used for the specification of asynchronous controllers and will focus on Signal Transition Graphs (STGs), which is a minimalist interpretation of Petri nets where each transition represents the toggle of a digital signal.

The talk will also present a new formalism called Waveform Transition Graphs (WTGs) [1], that was recently proposed as a means to make Petri nets friendlier to engineers. WTGs can be specified with a simple graphical representation. However, they sacrifice some of the expressive power of Petri nets: concurrency and choice cannot be enabled simultaneously. This constraint brings interesting structural properties that can be effectively used for synthesis and verification.

References


Structure Theory in a Dynamic Data-Driven World

Applications in Process Mining and BPM

(extended abstract)

Wil M.P. van der Aalst

Eindhoven University of Technology, PO Box 513, Eindhoven, The Netherlands

Abstract. Until the turn of the century, most Petri nets were made by hand or generated from another model (e.g., though synthesis). Such Petri nets where mostly used to provide a specification or design of a system (as-is or to-be). Analysis of these nets aimed at detecting behavioral anomalies like deadlocks and livelocks (through verification) and understanding performance (through simulation or analytical techniques). Structure theory provided unique ways to facilitate such analysis by exploiting the structure of (subclasses of) Petri nets. However, over the last decade one could witness a dramatic change in the way we analyze the behavior of discrete processes and systems. Model-driven approaches are replaced or complemented by data-driven approaches. The abundance of event data makes it feasible to study processes and systems directly. Process mining techniques allow for the discovery of Petri nets from event data and can be used to check conformance. Through process mining we are able to connect Petri nets to mainstream developments related to Big data, data science, and machine learning. The direct confrontation between modeled and observed behavior is valuable, but also provides many challenges. For example, one needs to deal with huge event logs and processes that change over time and exhibit deviating behavior. Can structure theory also play a key role in such data-driven analysis? The answer is affirmative. Elements of structure theory are already widely used in process mining and Business Process Management (BPM). Moreover, further breakthroughs are possible by tailoring structure theory towards more data-driven problems.

1 Introduction

Traditionally, Petri nets are made by hand or generated from other models [23, 32, 34, 35]. Petri nets can be used to design or specify discrete dynamic systems. Most Petri nets described in literature were created manually. However, program code, lower-level models (e.g., transition systems), and higher-level models (e.g., BPMN or UML models) can be transformed into Petri nets. Most of these transformations are quite straightforward, although the devil is often in the details and abstractions are needed. For example, the de facto standard for business process modeling—BPMN (Business Process Model and Notation) [33]—uses token passing. Also UML activity diagrams use token-based semantics and a notation similar to Petri nets. Examples of transformations that are more involved include the Petri net synthesis techniques known under the name of "region theory" [25, 20, 10]. State-based region theory starts from a transition system and aims to produce a Petri net that has the same behavior while capturing con-
currency. For example, in [20] it is shown that any transition system can be transformed into a bisimilar Petri net.

Given a Petri net, one can apply verification and performance analysis techniques. Verification is concerned with the correctness of a system or process. Verification techniques may be used to find deadlocks, livelocks, and other anomalies. It is also possible to define desirable properties in some temporal logic and then check whether the model has these properties. Performance analysis focuses on flow times, waiting times, utilization, and service levels. Typically, three dimensions of performance are identified: time, cost and quality. For each of these performance dimensions different Key Performance Indicators (KPIs) can be defined. Simulation, queueing models, or Markov models can be used to analyze systems with respect to such KPIs.

Mainstream analysis techniques do not exploit the structure of the model. For example, verification techniques may try to exhaustively traverse the state space and simulation approaches randomly sample behavior independent of the model’s structure. One of the key advantages of using Petri nets is that knowledge about the structure can be exploited during analysis [16]. The marking equation can be used to rule out markings that cannot be reachable [37, 19]. Siphons and traps can be used to reason about deadlocks [21, 22]. Place and transition invariants are used to identify properties that are preserved because of the net’s structure [27, 32, 31, 18]. Reduction rules can be used to make problems smaller while guaranteeing the same outcome [13, 14, 22, 46, 41].

Free-choice nets [15, 22, 39, 42], Petri nets without conflicting splits and joins [26], and marked graphs [32] are well-known subclasses of Petri nets. These subclasses can be identified based on their structure and often analysis becomes easier, e.g., one can decide whether a free-choice net is live and bounded in polynomial time [22]. Performance analysis may also benefit from structural theory [11, 17], e.g., one can compute performance bounds for marked graphs and free-choice nets.

Petri nets representing workflows or other types of business processes can also benefit from knowledge about the structure of the model. Consider for example the class of workflow nets (WF-nets) and the corresponding soundness notion [1]. A WF-net is a Petri net with a dedicated source place where the process starts and a dedicated sink place where the process ends. Moreover, all nodes are on a path from source to sink. A WF-net is sound if it is always possible to terminate and there are no dead parts in the model. Soundness can be checked in polynomial time for several subclasses, including free-choice WF-nets [7, 40].

The examples above show that structure theory allows for the identification of Petri nets whose structure strongly influences their behavior. Moreover, structure theory can also be used to compute bounds or shown the (im)possibility of particular behaviors.

Structure theory developed over the last fifty years with a strong focus on model-based analysis [16]. However, the spectacular growth of data is rapidly changing the way we analyze behavior. Rather than analyzing modeled behavior, we can now analyze the actual behavior of processes and systems!

Data are collected about anything, at any time, and at any place. It has become possible to record, derive, and analyze events at an unprecedented scale. Events may take place inside a machine (e.g., an X-ray machine or baggage handling system), inside an enterprise information system (e.g., an order placed by a customer or the submission of
a tax declaration), inside a hospital (e.g., the analysis of a blood sample), inside a social network (e.g., exchanging e-mails or twitter messages), inside a transportation system (e.g., checking in, buying a ticket, or passing through a toll booth), etc. [5]. Events may be “life events”, “machine events”, or “organization events”. The term Internet of Events (IoE), coined in [4], includes (1) the Internet of Content (traditional web pages, articles, encyclopaedia like Wikipedia, YouTube, e-books, newsfeeds, etc.), (2) the Internet of People (all data related to social interaction, including e-mail, Facebook, Twitter, forums, LinkedIn, etc.), (3) the Internet of Things (physical objects connected to the network), and (4) the Internet of Locations (data that have a geographical or geospatial dimension, e.g., generated by smartphones and cars). The IoE provides a new and extremely valuable source of information for analyzing processes and systems.

The abundance of event data triggers the question: Do we need structure theory in this dynamic data-driven world? We believe that, more than ever, there is a need to use and develop structure theory. This extended abstract only provides a few pointers in this direction. However, structure theory is already used in areas such as Business Process Management (BPM) and process mining. Moreover, in the era of Big data, there is a need to analyze processes efficiently. This can only be done by exploiting the structure of process models.

2 Process Mining and Business Process Management

Developments in Business Process Management (BPM) over the last two decades have resulted in a well-established set of principles, methods and tools that combine knowledge from information technology, management sciences and industrial engineering for the purpose of improving business processes [3, 24, 45]. Until recently, mainstream BPM approaches were predominantly model-driven without considering the “evidence” hidden in the data [3]. However, this changed dramatically with the uptake of process mining.

Process mining aims to exploit event data in a meaningful way, for example, to provide insights, identify bottlenecks, anticipate problems, record policy violations, recommend counter-measures, and streamline processes [5].

The interest in process mining is reflected by the growing number of commercial process mining tools available today. There are over 25 commercial products supporting process mining (Celonis, Disco, Minit, myInvenio, ProcessGold, QPR, etc.). All support process discovery and can be used to improve compliance and performance problems. For example, without any modeling, it is possible to learn process models clearly showing the main bottlenecks and deviating behaviors.

Starting point for any process mining effort is a collection of events commonly referred to as an event log (although events can also be stored in a database). Each event is characterized by:

- a case (also called process instance), e.g., an order number, a patient id, or a business trip,
- an activity, e.g., “submit form” or “make decision”,
- a timestamp, e.g., “2017-06-30T09:56:30+00:00”,

47
– additional (optional) attributes such as the resource executing the corresponding event, the type of event (e.g., start, complete, schedule, abort), the location of the event, or the costs of an event.

The lion’s share of process mining research focused on the discovery of process models from event data [5]. The process model should be able to capture causalities, choices, concurrency, and loops. Process discovery is a notoriously difficult problem because event logs are often far from complete and there are at least four competing quality dimensions: (1) fitness, (2) simplicity, (3) precision, and (4) generalization. Most discovery algorithms described in the literature (e.g., the α-algorithm [8], the region-based approaches [12, 38, 44], and the inductive mining approaches [28, 29, 30]) produce formal models having clear semantics. All of these approaches use Petri nets as a representation or the results they return can easily be converted into Petri nets [5].

We strongly believe that the communities working on BPM and process mining can benefit more from structure theory. Moreover, we also believe that process mining provides novel and exciting challenges for people working on structure theory. Given the developments sketched before, it is important to use the abundantly available data. Purely model-driven analysis only makes sense when designing a completely new system of process.

In the remainder, we briefly sketch two examples where structure theory could play a more prominent role. In this extended abstract, we only highlight some of the opportunities and challenges without going into detail.

3 Process Discovery

The goal of process discovery is to learn a process model from event data. Typically, an event log \( L \in B(A^*) \) is used as input. \( L \) is a non-empty multiset of traces over some activity set \( A \). A process model \( Mod \subseteq A^* \) defines a set of traces over some activity set \( A \). Different representations can be used to describe \( Mod \). One can use for example a so-called accepting labeled Petri net described by the triplet \( AN = (N, M_{\text{init}}, M_{\text{final}}) \) where \( N = (P, T, F, A, l) \) is a labeled Petri net, \( M_{\text{init}} \in B(P) \) is the initial marking, and \( M_{\text{final}} \in B(P) \) is the final marking. \( P \) is the set of places, \( T \) is the set of transitions, and \( F \) is the flow relation. Transitions can have a label as defined by labeling function \( l \in T \not\rightarrow A \). Transition \( t \in T \) has a label \( l(t) \in A \) if \( t \in \text{dom}(l) \). Otherwise, \( t \) is silent (i.e., its occurrences are not recorded). Any firing sequence leading from \( M_{\text{init}} \) to \( M_{\text{final}} \) corresponds to an accepting trace \( \sigma \in A^* \). The set of all possible accepting traces defines the behavior of \( AN \): \( Mod_{AN} \subseteq A^* \).

A discovery algorithm can be described as a function \( disc \in B(A^*) \rightarrow \mathcal{P}(A^*) \). Note that \( \mathcal{P}(A^*) \) denotes the powerset of traces over \( A \), i.e., \( disc(L) \subseteq A^* \). Ideally, the discovered model allows for all traces observed, i.e., \( \{ \sigma \in L \} \subseteq disc(L) \). However, it is easy to define degenerate solutions like \( disc_{\text{overfit}}(L) = \{ \sigma \in L \} \) and \( disc_{\text{underfit}}(L) = A^* \) that do not provide any insights. \( disc_{\text{overfit}} \) basically enumerates

\[ \text{Note that one needs to apply the labeling function to each transition occurrence in the firing sequence. Transitions without a visible label are skipped.} \]
the event log and is likely to severely overfit the data. $disc_{\text{underfit}}$ allows for any behavior involving activities $A$. Discovery function $disc$ should generalize over the input data that consists of examples only. At the same time, we may want to abstract from infrequent behavior.

The representation of the discovered process model plays an important role in balancing between overfitting and underfitting. The so-called representational bias defines the class of model that can be returned by the discovery algorithm. Accepting labeled Petri nets form such a class. One can impose additional restrictions on the class of accepting labeled Petri nets. For example, one can limit the representational bias to free-choice nets, WF-nets, or sound WF-nets. Such constraints may aid the understandability of the resulting process models, e.g., free-choice nets separate choice and synchronization and WF-nets have a clear begin and end.

Discovery algorithms producing Petri nets may return a model that is not a WF-net or that is not sound. This makes the interpretation of the discovered process model very difficult. The $\alpha$ miner [8] and heuristic miner [43] aim to return a sound WF-net, but often do not. Parts of the model may be disconnected and cases may get stuck in the middle of the process. Discovered Petri nets having deadlocks and and livelocks are difficult to interpret: They should describe the observed behavior but confuse the analyst instead. The deadlocking or livelocking paths do not contribute to the set of accepting traces $\text{Mod}_{AN} \subseteq A^*$. Region-based approaches [12, 38, 44] provide more control over the result. However, without special provisions the set of accepting traces is ill-defined or hard to interpret. The family inductive mining approaches [28, 29, 30] produce process trees which form a subclass of sound WF-nets. However, the output of these techniques is limited to process trees: a small and very particular subclass of process models.

We would like to discover process models with a configurable representational bias and therefore see many opportunities for structure theory. The representational bias, i.e., the class of models that can be discovered, should not be accidental. The class should be defined based on desirable (1) structural properties and (2) behavioral properties. Structural properties include possible constraints like:

- There is one source place and one sink place marking the start and completion of a case (i.e., a WF-net) [1, 7].
- There should be no mixtures of choice and synchronization (i.e., the net is free-choice) [22].
- Splits and joins should match (i.e., there are no PT- and PT-handles) [26].
- The sort-circuited Petri net should have an S-cover and/or a T-cover [22].
- Places cannot be a split and a join at the same time (for any $p \in P$: $| \bullet p | \leq 1$ or $| p \bullet | \leq 1$).
- Places have at most $k$ inputs and outputs for any $p \in P$: $| \bullet p | + | p \bullet | \leq k$.
- Etc.

Behavioral properties include [7]:

\(^{2}\) Loops can only be unfolded a finite number of times in the event log. Moreover, in case of concurrency, one cannot expect to see all interleavings in the log.
– Soundness: there are no dead parts and it is always possible to reach the final marking and when it is reached the rest of the net is empty.
– Generalized soundness: the same as soundness but with any number of tokens in the source place.
– Relaxed soundness: there is at least one execution that ends up in the final marking.
– Deadlock free: the only reachable dead marking is the final marking.
– Etc.

As shown in [2, 40, 7] there are interesting relations between structure and behavior. These are key to limit the search space to the desired class of models. It is not very effective to generate models first and subsequently check whether they match the desired representational bias. Therefore, structural techniques are needed to limit the search space during discovery.

4 Conformance Checking

After discussing the (possible) role of structure theory in control-flow discovery, we now look at the situation in which both a process model and an event log are given. The model may have been constructed by hand or may have been discovered. Moreover, the model may be normative or descriptive. Conformance checking relates events in the event log to activities in the process model and compares both. The goal is to find commonalities and discrepancies between the modeled behavior and the observed behavior.

For conformance checking an event log \( L \in B(A^*) \) and a process model \( Mod \subseteq A^* \) are used as input. Here we assume that process model \( Mod \) was specified in terms of accepting labeled Petri net \( AN = (N, M_{\text{init}}, M_{\text{final}}) \) with \( N = (P, T, F, A, l) \). The result of conformance checking is a diagnosis identifying and explaining discrepancies. Hence, a conformance checking algorithm can be described as a function \( conf \in B(A^*) \times P(A^*) \rightarrow D \) where \( D \) is the set of possible diagnostics. For example, we may compute the fraction of cases in the log that fit the model perfectly. Formally: 

\[
conf(L, Mod) = \frac{|\{\sigma \in L | \sigma \in Mod\}|}{|L|} \quad \text{(note that L is a multiset and Mod is a set)}.
\]

Simply counting the fraction of fitting cases is useful, but does not provide detailed diagnostics. Moreover, one cannot distinguish between cases that deviate just a bit and cases that are completely unrelated. Therefore, more advanced techniques have been developed. The token-based conformance checking approach proposed in [36] counts the number of missing and remaining tokens. State-of-the-art techniques in conformance checking are often based on the notion of alignments [6, 9]. Alignments relate events in the log to transition occurrences in the model. An alignment is a sequence of moves.

There are three types of moves: synchronous moves (model and log agree), moves on model only (the model needs to make a move that is not matched by the event log), and moves on log only (an event in the log cannot be matched by the model). Here we cannot give the details. However, the construction of an optimal alignment can be formulated as a shortest path problem in the state space obtained by taking the synchronous products of both the model and log. This shortest path problem greatly benefits from the marking equation which can be used to (1) prune the state-space by removing

50
paths that cannot lead to the final marking and (2) to compute underestimates for the remaining distance [6, 9]. This is a wonderful example of using structure theory in the context of process mining.

Apart from alignments there may be other opportunities for structure theory. If there is a clear relation between structure and behavior, then there are opportunities to speed-up conformance checking.

5 Outlook

In this extended abstract, we positioned structure theory in the context of more data-driven challenges. Structure theory has been applied to verification questions in Business Process Management (BPM). For example, the soundness notion for WF-nets can be related to a variety of “structural ingredients”, e.g., by using properties specific for free-choice WF-nets or by applying the marking equation to get initial diagnostics. However, even more promising are the applications of structure theory in process mining. We provided two example questions (process discovery and conformance checking) where structure theory could play a prominent role. Process discovery is probably the most important and most visible intellectual challenge related to process mining. It is far from trivial to construct a process model based on event logs that are incomplete and noisy. New process mining approaches should reconsider the representational bias to be used. However, this is only feasible for real-life event logs if the structure can be related to behavior. Alignments are a powerful tool to relate modeled and observed behavior. However, computing optimal alignments requires solving large optimization problems for every trace in the event log. Fortunately, the marking equation can been used to prune the search space and guide the search algorithms.

We hope that this extended abstract will encourage people working on structure theory to consider the many interesting and challenging problems in process mining. There are great opportunities for original research and a need to better cope with the abundance of event data. Clearly, it does not make sense to consider only models when analyzing existing processes and systems. We should also take into account the data to remain relevant for the stakeholders.

References


Author Index

Aalst, Wil.v.d., 45
Balbo, Gianfranco, 17
Cortadella, Jordi, 43
de Frutos Escrig, David, 33
Desel, Jörg, 17
Giua, Alessandro, 43
Haddad, Serge, 19
Kleijn, Jetty, 33
Koutny, Maciej, 35
Silva, Manuel, 1