mathematics

MDPI

Review

The Unfolding: Origins, Techniques, and Applications within Discrete Event Systems

Younes Rouabah and Zhiwu Li *

Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau SAR, China
* Correspondence: zwli@must.edu.mo

Abstract: This article aims to provide a perspective on the foundations and developments of the net unfolding techniques and their applications to discrete event systems. The numerous methods applied to concurrency presented in the literature can be roughly divided into two classes: those that assume concurrency can be represented by means of a non-deterministic form, and those that represent concurrency by means of causal relations. This study serves as an ideal starting point for researchers interested in true concurrency semantics by offering a concise literature review of one of the major streams of research towards concurrency and interleaving problems. In order to cope with the state-explosion problem, the unfolding approach is used. Based on the findings of concurrency theory, interleaving semantics are replaced with a unique partially ordered occurrence net. In this paper, we aim to provide a comprehensive review on the history of net unfoldings, the methods that are based on these unfoldings, and how they are used in discrete event systems for automatic verification and compact representations purposes.

Keywords: discrete event system; state-explosion problem; theory of concurrency; state space reduction technique; partial-order method; true concurrency semantics; net unfolding

MSC: 93-02; 93-03; 93-08

Citation: Rouabah, Y.; Li, Z. The Unfolding: Origins, Techniques, and Applications within Discrete Event Systems. *Mathematics* **2023**, *11*, 47. https://doi.org/10.3390/math11010047

Academic Editor: Frank Werner

Received: 15 November 2022
Revised: 14 December 2022
Accepted: 15 December 2022
Published: 23 December 2022

1. Introduction

In [1], the term Discrete Event Dynamic Systems (DEDSs) first appeared to recognize the ubiquitous boom of a deluge of man-made systems that are usually computer-integrated and highly automated, thanks to the stupendous developments of information technology and network communication. As an alias of DEDS, a discrete event system (DES) is a dynamic system, whose behavior is identified by sudden and unexpected shifts in the values of its states, which takes many values from a potentially infinite set. The state evolution of a DES relies entirely on the occurrence of asynchronous discrete events over time. Simply put, a DES is defined as a discrete-state and event-driven system. The theory and engineering of DES began taking shape through numerous complementary research fields. Since the birth of notion of discrete event systems, many inter- or cross-disciplinary methodologies have technically and conceptually formulated to address the modeling, analysis, and synthesis of various contemporary human-made systems that are usually highly computer-integrated, including manufacturing, transportation, communication networks, software systems, and logistics [2].

Back to the early sixties of the last century, C. A. Petri originally founded the primary idea and concept of Petri nets (PNs) [3]. From then the area has been massively developed in both theory and applications. Net theory gradually becomes an important branch of computer science, and the methodologies based on Petri nets become extensively applied to the control area, accompanying with the blooming of discrete event systems. Even though several other models of concurrent and distributed systems, e.g., various programming languages, queues, and formal languages, were developed at that time, Petri nets along

with their various variants remain an outstanding model for such systems thanks to their graphical representation and modeling capability.

One of the most appealing aspects of Petri nets is the way that the fundamental aspects of concurrent systems are identified conceptually and mathematically. Petri nets are the model of choice in many applications due to their simplicity of conceptual modeling that is centered on a natural graphical notation. The intrinsic process of Petri nets, which permits a formal capture of many basic notions and issues of concurrent systems, and contributes significantly to the foundation of a robust theory of concurrent systems based on Petri nets. In fact, the "Petri nets" terminology refers to net-based models that may be categorized into three primary layers: Elementary net systems, place/transitions systems, and high-level nets. These types of models differ in what and how they can process the formal verification of concurrent systems [4].

The analytical study of concurrent DESs with synchronizations is vital because of the diversity of their potential behaviors (i.e., the causal/concurrent relationships). Due to its practical cruciality and theoretical significance, the verification of event-driven systems drags a lot of attention from researchers and practitioners. Technically and specifically, one can formulate a verification problem using a given nondeterministic DES model M of a real or mathematical system and a logical specification S, which denotes a specific property (associated with the system behavior) that the system must satisfy. The verification problem comes to decide whether or not the behavior of M complies with the specification S. The boundedness of a concurrent system represented through its model results in a state space projected into a finite state machine, formally known as the "reachability graph", an interleaving representation of the behavior that serves as the most standard semantics of Petri nets. This reachability graph can be thought of as a finite folding of infinite "computation trees" containing all possible executions of events [5]. However, the model could be unbounded, leading to a possibly computed overridden representation denoted as the "coverability graph" [2].

A rigorous state exploration approach is applied to construct a formally defined reachability graph, which ends with astronomical numbers of states in most cases, particularly for real-world networked systems, leading to the curse of dimensionality or the so-called "state explosion problem". This state explosion problem is the consequence of enumerating all the possible interleaving of event sequences, where the concurrency between events is discarded and replaced with non-determinism in exchange for efficiency and simplicity. Many researchers have efficiently and sufficiently used the "interleaving semantics" approach for fully automated verification purposes. However, representing concurrency by interleaving is usually semantically adequate but often extremely limited (due to the state explosion problem) [6]. This limitation plagues finite-state verification techniques since it creeps in when constructing a reachability graph corresponding to the concurrent system with highly concurrent tasks.

In order to examine a simpler and preferably "equivalent" model on the basis of verification, one alternative strategy is the formulation of abstractions. This may be conducted by dividing the set of transitions of a Petri net model into "silent/unobservable" and "tangible/observable" sets that are disjoint. This is accomplished using formal language techniques, transformation, or even rewriting techniques on the structure of models. For example, in [7,8], an abstraction of the basic reachability graph is presented as the so-called "basis reachability graph" (i.e., a compact representation that contains a condensed set of pillar markings; such a representation is feasible on the premise that there is no unobservable events cycles in a system structure) to verify concurrent systems properties such as fault diagnosis [7], security in terms of opacity [8], and the enforcement of these properties. However, these approaches, whose dependency is on the "interleaving semantics", usually fall into the state-explosion problem trap due to a lack of assumptions or the heavily concurrent activities within a modeled system.

Modeling concurrency by means of causal independence, usually called "true concurrency semantics", has been developed from the contributions made in [9–11], raising

various data structures and techniques to manipulate sets of runs (defined as the sequences of system states and events) efficiently (e.g., stubborn sets [12], persistent sets [13], etc.), where some of these methods tackle the effects of the state explosion, while others prefer to tackle its causes. These methods, also known as "partial-order methods", mainly define concurrent executions as partial-orders for unordered concurrent transitions since the order of their occurrences is irrelevant [14].

Net unfolding is a partial-order method founded in [15] and renovated in [16] to be used as a state explosion avoidance or mitigation technique. It mainly concentrates on the true concurrency of event occurrences, which leads to generating an acyclic net known as the "occurrence net". The latter is a mathematical structure that explicitly represents concurrency and causal dependence between events. Branching processes are another technical term, where unfoldings are known for the original contribution in [17] as a link between the theory of processes and the unfolding rather than the theory of event structures used in [15]. Like a reachability graph of a net system, the method of branching processes records all potential behaviors of a system, and only a finite partial behavior needs to be examined to address specific system-related queries. However, unlike a reachability graph, it does not make interleavings explicit; thus, it can be exponentially compact [5].

This paper scopes the net unfolding technique and its application within the community of DESs. It delivers the early foundations of unfoldings along the way with encapsulation and a walk-through of the various developed partial-order methods that fall into automatic verification. Several DES properties are examined, where the net unfolding technique is applied.

This paper consists of six sections, of which this is the first. Section 2 consists of an opening to DES automatic verification and the various properties studied over the years of research, and Section 3 recalls the basics of Petri nets. The foundations of net unfolding and its formal definitions are detailed in Section 4. In Section 5, the applications of partial-order methods for verification purposes are presented and reviewed. Finally, Section 6 serves as a conclusion for this paper.

2. Concurrent Systems

Throughout the years of research in computer science, the analysis of concurrent systems has been one of the hardest practical issues. Nowadays, it has been recognized that formal development methods are almost the best vehicle to guarantee software quality, in which a formal software specification expressed in a mathematical language must be well defined, based on mathematical concepts, whose properties can be easily understood and checked such that no ambiguities and contradictions are found. It has been demonstrated that formal specifications using Petri nets offer the tangible yet significant benefits of concentrating on the challenging features of these systems, such as concurrency, synchronization, and conflict aspects. In this section, we introduce the concurrent systems along with their properties, explain how these properties are verified in a general way. Readers may refer to [2,18,19] for an extensive coverage.

According to [13], one may think of concurrent systems as a set of interconnected components with the ability to operate concurrently and communicatively. Each element of these components can be considered a reactive system (i.e., a system with constant interaction with its environment). The environment of a single component is shaped by the rest of the components of the concurrent system, which is hence assumed to be closed. The behavior of a reactive system is determined by its evolving behavior over time. This is entirely different from the classical transformational view of programs, which holds that the functional connection between the input and the output state determines a program's meaning. Reactive systems are not devoted to transforming data such as traditional programs but to controlling processes. Many industrial examples of such concurrent reactive systems exist, such as networks, asynchronous circuits, and various types of plant-controller systems (e.g., telecom, digital circuits, manufacturing-plant controllers, etc.).

Concurrent systems can exhibit a staggering variety of diverse behaviors, making them particularly challenging to design, analyze and verify many properties of concerns. This is resulted from the combinatorial explosion brought on by all the potential interactions between the many concurrent components of the system and the numerous competing conditions that arose between them. This fact makes the design of concurrent systems an incredibly delicate undertaking. Applying these concurrent systems industrially for testing purposes before the final deployment also comes with a high cost that is usually considered impossible in most real-world cases. The latter makes the developments around concurrent systems extremely essential since they are used to control safety-critical devices as well as economically-crucial systems.

Verification ensures the correctness of the design of concurrent systems. It checks whether or not a system description complies with its expected properties. Such properties differ from the various types of consistency to complex correctness specifications that are pre-defined, for example, in a logical language. In this way, the verification guarantees that the accuracy of the formal description of the system meets the problem specifications.

A verification framework for a concurrent system consists of four essential components:

– A representation of the system;
– A representation of the properties to be checked;
– Semantics according to which the representations of the system and of the properties are compared; and
– A method (in most cases automated) for performing this comparison.

A verification process relies heavily on mathematical expressions to prove that a system is correct and matches certain specifications. Verifying a property does not mean testing it using the methods that prove a system is approximately correct. To prove that a system complies with a specific property, every possible behavior of the system needs to be checked to decide whether all of them are consistent with the given property.

One of the effective methods for the purpose of analysis and verification for concurrent systems is state-space exploration. It implies examining the behavior of all concurrent components in the system through a global representation known as the reachability graph. This is achieved through the enumeration of all the states yielded from all the encountered states, by firing all enabled transitions in each state starting from a given initial state.

State-space exploration techniques are able to analyze and verify various types of properties of a designed system, such as safety, liveness, deadlock-freedom, and forbidden states. The usage of such techniques can even be extended to more complicated properties; such security properties include non-interference, observability, and opacity, which are nowadays considered crucial requirements for many contemporary cyber-physical systems where information is extensive and message transformations in a network environment are rather frequent, possibly under some external malicious network invasion and critical information intrusion. Furthermore, these techniques show massive reliability within the control theory community [20], e.g., for supervisory control and fault diagnosis of discrete event systems, or more generally, networked infrastructure systems of human beings in contemporary society.

Numerous studies have aimed toward verification using state-space exploration. The flexibility of such a technique leads to robust and, hence, efficient implementations. Furthermore, verification using state-space exploration techniques is known to be entirely automatic (no human involvement is required). This is a critical aspect of an industrial verification approach. Real world systems are frequently constructed under time constraints, so verification processes that would take the designer an excessive amount of time are not feasible. Moreover, a verification approach should be easy to use by practitioners such that it can be accepted in practice.

These factors mainly contribute to the widespread usage of such an approach in the existing verification tools. Such tools vary from one another in terms of the languages of the formal description they rely on to represent systems and properties, as well as the conformation yardstick used to evaluate these representations. However, even though

these tools show a noticeable diversity, they still rely on state-space exploration algorithms, by any means possible, for conducting the verification itself.

The constant development of the tools made state-exploration techniques for checking concurrent systems become widely accepted, due to their efficiency and conveniences for practitioners. The blooming of well-established cases from the industrial size systems has demonstrated an increasing rate over the last decades, see [21]. Various studies on concurrent systems with a high degree of complexity are analyzed and verified by adopting state-space exploration techniques. These techniques are frequently successful in identifying delicate design flaws.

The critical constraint on state-space exploration verification approaches is the massive size of the state space. Due to elementary combinations, this size can be exponential with respect to the scale of the description of a system being examined. The state-explosion problem is the term given to this exponential growth. In theory, the notorious state-explosion problem is a consequence of coping with the true concurrency by interleaving, or, more precisely, of exploring every possible interleaving of concurrent transitions. For example, executing n concurrent transitions requires enumerating all $n!$ interleavings of these transitions [13].

3. Prerequisites on Petri Nets

In this section, we recall the essential definitions pertaining to Petri nets. The reader is referred to [22] for more details on Petri nets.

A net is a quadruple $N = (P, T, {}^\bullet(), ()^\bullet)$ such that P and T, respectively, refer to the sets of Places and Transitions, and ${}^\bullet() : T \to 2^P$ is a preset relation and $()^\bullet : T \to 2^P$ is a postset relation. Places and transitions are generally called nodes. A marking M of N is a multi-set of places, i.e., $M : P \to \mathbb{N}$. We pick the common standard rules for net pictorialization, viz., circles for places, boxes for transitions, arcs for the flow relation, and tokens as black dots placed within the circles to represent markings.

A net system is a pair $\Sigma = (N, M_0)$ that consists of a finite net $N = (P, T, {}^\bullet(), ()^\bullet)$ and an initial marking M_0. Figure 1a shows a 1-bounded net system, with $M_0 = [M(p_1), M(p_2), M(p_3), M(p_4), M(p_5), M(p_6), M(p_7)]^T = [1100000]^T$ as its initial marking. Different markings can be reached by firing transition sequences.

A transition $t \in T$ is enabled at a marking M if $M \geq {}^\bullet(t)$. Such a transition may occur, yielding a new marking $M' = M - {}^\bullet t + t^\bullet$, denoted by $M[t\rangle M'$ or $M[t\rangle$ in the case that M' is of no interest. When t is not of concern, write $M \xrightarrow{*} M'$.

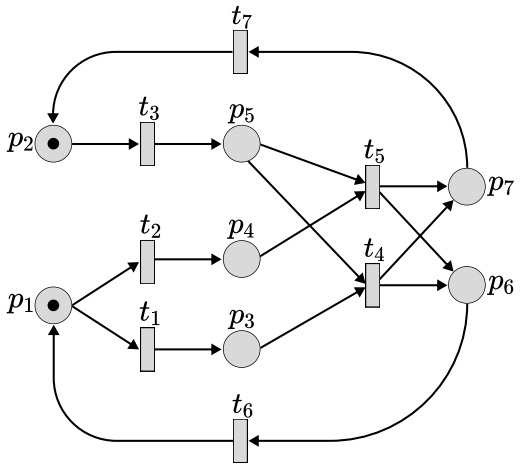
We denote by $M[\sigma\rangle$ the enabling of the transitions sequence $\sigma = t_{1j} \dots t_{jk} \in T^*$ at M, and $M[\sigma\rangle M'$ the occurrence of σ that yields marking M' from M. Figure 1b illustrates one of the possible occurrence sequences of the net system in Figure 1. Two transitions t_1 and t_2 are concurrently enabled at a marking M if $M \geq {}^\bullet(t_1) + {}^\bullet(t_2)$.

A marking M is a reachable marking if there exists an occurrence sequence σ such that $M[\sigma\rangle M'$. The set of all markings reachable from M_0 raises up the reachability set of Σ, denoted by $R(N, M_0) = \{M | M_0 \xrightarrow{*} M\}$, i.e., $R(N, M_0) = \{M \mid \exists \sigma \in T^* : M_0[\sigma\rangle M\}$.

Let $k \in \mathbb{N}$ be a natural number, where $\mathbb{N} = \{0, 1, 2, \dots\}$. A Petri net system is k-bounded if, for any reachable marking M and any place $p \in P$, $M(p) \leq k$. Moreover, Σ is said to be bounded if it is k-bounded for some $k \in \mathbb{N}$.

A transition t is reachable from a marking M if there exist a marking M' and an occurrence sequence σ such that $M[\sigma\rangle M'$ and M' enables t. In other words, a transition is reachable from a marking if the firing of a feasible transition sequence at the marking enables the transition.

Analogously, a place p is reachable from a marking M if there exists a transition t reachable from M and $p \in (t)^\bullet$. Furthermore, a set of transitions T_1 is reachable from a marking M if every transition $t \in T_1$ is reachable from M. A set of places P_1 is reachable from a marking M if every place $p \in P_1$ is reachable from M.



(**a**)

$$\binom{p_1}{p_2} t_1 \binom{p_2}{p_3} t_3 \binom{p_3}{p_5} t_4 \binom{p_6}{p_7} t_6 \binom{p_1}{p_7} t_7 \binom{p_1}{p_2}$$

(**b**)

Figure 1. Self-explanatory example of net system mechanics. (**a**) A simple net system. (**b**) An occurrence sequence of the net system in (**a**).

4. Unfolding

Back to the late 1970s and 1980s of the last century, several researchers noticed the necessity to distinguish between concurrency and nondeterminism, see [23–26]. By the 1990s, this insight served as the foundation for a sizable collection of studies on concurrency models known as "partial-order models". These models contributed heavily to the progress made on the state-explosion problem. In this section, we detail the earliest history of net unfolding (i.e., more precise partial-orders methods) and introduce branching processes. As mentioned in Section 1, branching processes are another term for the unfolding, linked to the theory of processes within the net theory. The reader can think of a branching process as the mathematical model relied on a process to "unfold" net systems. Simply put, branching processes are partial-order semantics of Petri nets. Readers can check [5,13,17] for further extensive coverage.

4.1. History of Net Unfolding

The studies concerning concurrency models incorporate the partial order models and vary in the way they deal with the state-explosion problem. Among these studies, there is the "stubborn sets" of Valmari [27], the "persistent sets" of Godefroid [14], and the "ample sets" of Peled [28]. These approaches vary on the implementation; however, they serve various common ideas. For instance, the stubborn sets of Valmari and the persistent sets of Godefroid are the closest to each other, since they both developed a method for computing a compact representation of the state space, which is used later on to verify the properties of concurrent systems. However, despite this overall resemblance, there are a number of distinctions that set Valmari's work apart from Godefroid's. For example, Valmari does not depend on any partial-order semantics to verify and validate the correctness of his algorithms, while Godefroid does by using "Mazurkiewicz's traces [24]". In fact, Godefroid even claimed that his work generalized and improved Valmari's stubborn set method [13].

Similar findings about the relationship between the partial and total order models of execution were also exploited by other techniques, such as the "sleep sets" of Godefroid [6], the "branching process" of Engelfriet [17], and the "unfolding technique" of McMillan [16]. Among these studies, we are interested in the work of Engelfriet and that of McMillan, which contribute heavily to the advancement of the verification algorithms using partial-order methods that we observe in the current decade as evidenced by the literature maps (litmaps) shown in Figure 2.

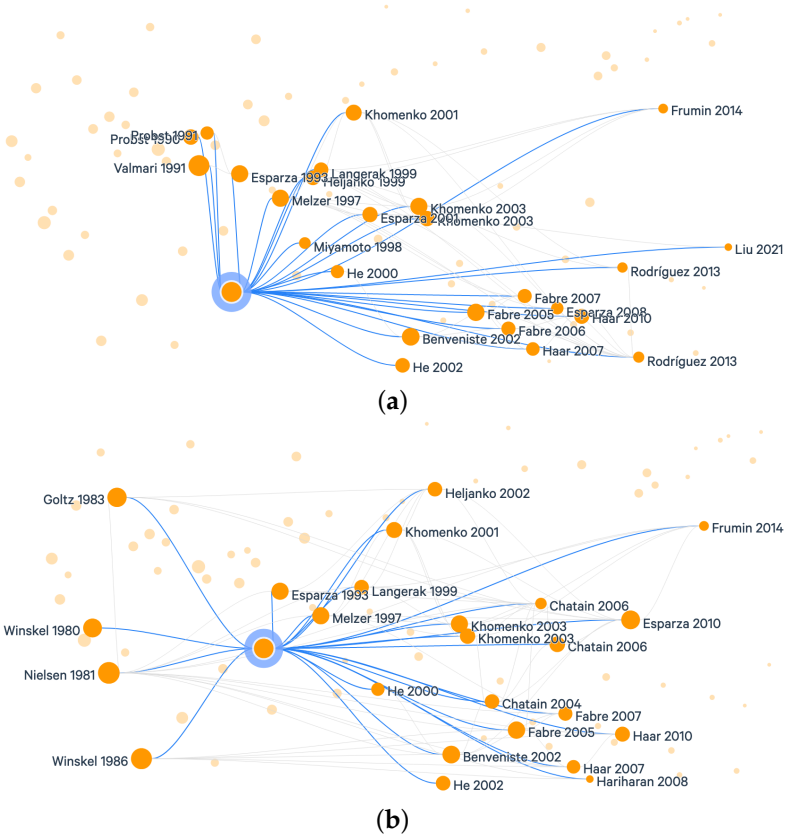


Figure 2. Litmaps representing the impact of both McMillan and Engelfriet studies on the partial-order models developed in the current decade (e.g., readers may refer to Tables 1 and 2 for more details). (**a**) McMillan's unfolding impact on the current decade approaches. (**b**) Engelfriet's Branching processes impact on the current decade approaches.

Let us go back to [9] in order to embark on the branching processes and unfolding of net systems, where non-sequential processes were introduced by Petri to represent the true concurrency of nets. A partial run of a net system is referred to as a non-sequential process, also commonly known as a causal or occurrence net. It consists of the global state of the fired transitions and their causal dependencies, in addition to the identified concurrent transitions, which may be fired separately. Several studies have intensively explored the theory of non-sequential processes since its inception, such as the work of Best et al., Goltz and Reisig, and others [29–31].

A net system may generate a variation of non-sequential processes. In other words, every single process reflects a possible solution to a conflict. A conflict represents a case where two completely different transitions are enabled at the same marking; however, yielding one of them leads to disabling the other. Net unfolding was first introduced by Nielsen et al. in [15] as an attempt to combine the events causality of Petri [9,32] and the domains of Scott [33,34] and Stoy [35] for the purpose of enumerating all potential full runs (i.e., the full branching run of a net system) in terms of a single object. In [36,37] Nielsen et al. presented classifying and axiomatic formulations of the Elementary Net Systems (i.e., a class of Petri nets we mentioned in Section 1) and investigated their properties. Net unfolding was indeed the motivating force behind the theory of event structures of Winskel [26,38].

Although no objects had been specified to express "partial branching runs", Engelfriet noticed in [17] that non-sequential processes represented partial runs; however, the unfolding only explained the unique full branching run of a net system. For such an observation, he comes up with an axiomatic definition of branching processes, where branching processes are defined as a set of Petri nets that meet a variety of conditions.

A wider range of Petri nets classes, such as high-level nets [39], colored nets [40,41], and contextual nets [42] among others, have been incorporated into the theory of non-sequential processes and branching processes.

In [16], McMillan investigated the unfoldings from an algorithmic perspective and not only the semantic side. He introduced an algorithm to verify the deadlock-freeness of net systems by generating a finite complete prefix of the overall unfolding, and applied it to multiple testing scenarios such as the famous "dining philosophers" paradigm to demonstrate the effectiveness of his approach in avoiding or mitigating the state-explosion problem within asynchronous circuits.

Several studies have demonstrated the reliability of complete prefixes as being relatively compact for various experimented net systems; see [16,43–47]. Taking the complete prefix as input, other works have studied the verification of net systems properties. The deadlock property was initiated by McMillan in [16,48], and then followed by Melzer and Römer, Heljanko, and Khomenko and Koutny in [49–51], respectively. Furthermore, the reachability problem has been explored by Schröter and Esparza in [52]. Probst and Li in [44,53] relied on "pomtrees" (i.e., a form of partial-orders) to verify delay-insensitive VLSI systems using what is called "behavior machines".

These early-stage foundations of the algorithmic perspective for the unfolding techniques depend on different implementations to reach the desired goal. In [16], McMillan adopted a branch and bound technique. Melzer and Römer proposed in [49] an elegant approach to resolve deadlocks and reachability problems using Integer Linear Programming (ILP), and acquired quite promising results using existing ILP tools. Unlike Melzer and Römer, Khomenko and Koutny developed their custom algorithm in [51] for solving ILP equations and found higher performance results compared to using the already existing tools. In [50], Heljanko combined logic programs reduction with stable model semantics, which in itself has an NP-complete complexity. Schröter and Esparza in [52] provided an extensive coverage of various algorithms from a performance and complexity point of view.

A polynomial verification approach using the theory of non-sequential processes for net systems without conflicts was also developed in a previous study by Best and Esparza in [54], but the applied representation of the system (i.e., persistent nets) suffered from a constrained level of expressiveness. McMillan is considered to be the first to successfully apply the unfoldings method for the purpose of verification.

The true concurrency of net systems was also the pillar motivation to avoid the state-space explosion problem by using partial-order compaction techniques. Nevertheless, the adopted strategy differs from the unfolding technique. Given a net system, the unfolding techniques exploit the unfolding (i.e., the true concurrency semantics) of the system representation rather than the state graph representation (i.e., the interleaving semantics). Partial-order compaction techniques compact the overall state space that needs to be enumerated by using concurrency semantics. For such a reason, given a reachable marking, the partial-order compaction techniques generate the partial set of transitions leaving the marking, and then only investigate the transitions of this set. For further coverage, comprehensive details about the reduced sets can be found in the investigation of Valmari about the state space problem [55].

Given a branching process, at each iteration, a set of events needs to be generated and added to it; this problem was studied by Esparza and Römer in [47]. It was later improved in [51] by Khomenko and Koutny. The unfolding technique was demonstrated to be NP-hard by McMillan in [48] when it comes to deadlocks' verification. The set of possible extensions computation causes the NP-hardness complexity and was extensively covered in [52] by Schröter and Esparza.

By this, we enclose this section with a summary Table 1 representing the earliest studies (i.e., from 1970–2000), which had a great impact on the verification approaches of the recent decades using the partial-order techniques.

Table 1. Reported work on the earliest motivations to the foundation of the unfolding in discrete event systems.

Title	Refs.
Mathematical theory of computation	[33]
Data types as lattices	[34]
Net theory	[32]
The non-sequential processes	[9]
TIme, clocks, and the ordering of events	[23]
Denotational semantics and programming language theory	[35]
Petri nets, event structures and domains	[15]
The non-sequential behavior of Petri nets	[29]
Trace theory	[24]
Modeling concurrency with partial orders	[25]
Event structures	[26]
Sequential and concurrent behaviors in net theory	[30]
Non-sequential processes	[31]
Elementary net systems and behaviours	[36]
Pomtrees and behavior machines	[44]
The reduced sets "Stubborn sets"	[27]
The reduced sets "Persistent sets"	[14]
Branching processes of Petri nets	[17]
Persistent sets and non-sequential processes	[54]
The unfolding technique and deadlock checking (McMillan algorithm)	[16]
The reduced sets "Ample sets"	[28]
Transition systems, event structures, and unfoldings	[37]
Trace verification using unfoldings	[43]
Partial-order methods and concurrent systems	[13]
The state explosion problem	[55]
Foata normal forms (ERV algorithm)	[45,46]
Net unfolding and deadlock verification	[49]
Products and unfoldings	[47]
Stable model semantics for deadlock and reachability problems	[50]
Integer Linear Programming with partial-order methods for deadlock verification	[51]
Net unfolding as an approach for reachability analysis	[52]

4.2. Occurrence Nets

Prior to any formal definition, we first provide a few intuitive thoughts. Given a directed graph G with a root node, one can "unfold" it into a labeled tree. The latter takes as nodes the available paths in G starting from the root. The labels of the nodes of the graph are then projected into the nodes of the tree. The unfolding process can be stopped at any chosen step, resulting in different trees; however, in most cases, the process is infinite and may yield an infinite unique tree. The latter is the full unfolding of the graph.

Correspondingly, net systems are unfolded into labeled "occurrence nets", a particular class of nets with a tree-like structure. The places and transitions labeling of the net system is projected into the nodes of the occurrence net. The unfolding of a net system yields a set of labeled occurrence nets known as branching processes. The unfolding process can be stopped at any chosen step, resulting in different trees. However, in most cases, the process is infinite and may yield an infinite unique tree. This latter is the maximal unfolding of the net system.

Foremost, we need to define the causal, conflict, and concurrency relations between net nodes. We denote a sequence of arcs $(x_1, x_2), (x_2, x_3), \ldots, (x_{N-2}, x_{N-1}), (x_{N-1}, x_N)$ as a path of the net N. We distinguish the net acyclicity if there is no path with $x_1 = x_N$.

Let $N = (P, T, {}^\bullet(), ()^\bullet)$ be an acyclic net and $x_1, x_2 \in P \cup T$ be two *nodes* of N. The ordering relations between x_1 and x_2 are defined in the following way.

– We say that x is a causal predecessor of y, denoted by $x < y$, if there is a path of arcs from x to y.

– We say that x and y are in conflict, denoted by $x\#y$, if there is a place z, different from x and y, from which one can reach x and y.
– We say that x and y are concurrent, denoted by $x||y$, if x and y are neither causally related nor in conflict.

An occurrence net is a net $\mathcal{O} = (B, E, {}^\bullet(), ()^\bullet)$ such that:

(1) For every $b \in B$, $|{}^\bullet b| \leq 1$;
(2) $\mathcal{O}$ is acyclic, or, equivalently, the causal relation is a partial order;
(3) $\mathcal{O}$ is finitely preceded, i.e., for every $x \in B \cup E$, the set of elements $y \in B \cup E$ such that $x < y$ is finite;
(4) No element is in conflict with itself.

The occurrence net properties make the detection of causal, conflict, and concurrency relations between any two nodes straightforward. The elements of B and E are commonly referred to as conditions ("Bedingungen" in Petri's original notation) and events, respectively. $Min(\mathcal{O})$ signifies the set of minimal elements of $B \cup E$ with respect to the causal relation (i.e., the elements with an empty preset). Since the transitions of nets typically have a nonempty preset, the elements of $Min(\mathcal{O})$ are conditions.

The acyclic net in Figure 3b represents the true concurrency semantics through the occurrence net corresponding to the occurrence sequence in Figure 1b, while Figure 3a shows the interleaving semantics through the sub-reachability graph of the same occurrence sequence.

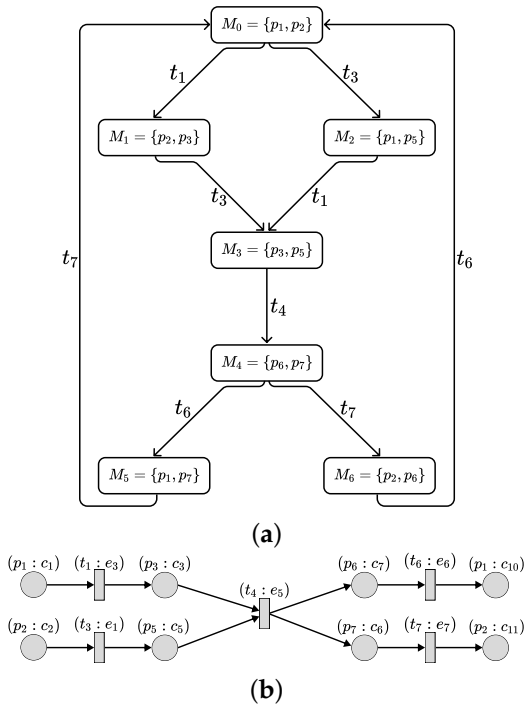


Figure 3. A pictorial representation of the different nets semantics. (**a**) A sub-reachability graph representing the interleaving semantics of the occurrence sequence in Figure 1b. (**b**) An occurrence net representing the true concurrency semantics of the occurrence sequence in Figure 1b.

4.3. Branching Processes

The occurrence nets generated from net systems by "unfolding" are paired with a net "homomorphism" mapping λ to form a branching process β. Before we give the formal definition of branching processes, we define the homomorphism properties between nets.

Let $\Sigma_i = (P_i, T_i, {}^\bullet()_i, ()_i^\bullet, M_{0i})$ be net systems, with $i = 1, 2$. A net homomorphism, $\lambda : P_1 \cup T_1 \to P_2 \cup T_2$ is a mapping between the nodes of Σ_1 and Σ_2 such that:

(1) $\lambda(P_1) \subseteq P_2$ and $\lambda(T_1) \subseteq T_2$;

(2) for every $t \in T_1$, the restriction of λ to ${}^\bullet(t)$ is a bijection between ${}^\bullet(t)$ (in Σ_1) and ${}^\bullet\lambda(t)$ (in Σ_2). The same goes for postset $(t)^\bullet$ and $\lambda(t)^\bullet$. Simply put, λ preserves the environments of transitions (the preset and postset of transitions);
(3) The restriction of λ to M_{01} is a bijection between M_{01} and M_{02}. In other words, a net homomorphism also preserves the initial marking.

A branching process β of a net system $\Sigma = (N, M_0)$ is a pair $\beta = (\mathcal{O}, \lambda)$, such that:

(1) $\mathcal{O} = (B, E, {}^\bullet(), ()^\bullet)$ is an occurrence net;
(2) λ is a net homomorphism from $\mathcal{O}$ to N such that for every $e_1, e_2 \in E$, if ${}^\bullet e_1 = {}^\bullet e_2$ and $\lambda(e_1) = \lambda(e_2)$, then $e_1 = e_2$ (β does not duplicate the transactions of Σ).

Two branching processes $\beta_1 = (\mathcal{O}_1, \lambda_1)$ and $\beta_2 = (\mathcal{O}_2, \lambda_2)$ are said to be isomorphic if there is a bijective homomorphism λ between them such that $\lambda_2 \circ \lambda = \lambda_1$. In other words, $\mathcal{O}_1$ and $\mathcal{O}_2$ are different only in the naming of their nodes and arcs.

Given two occurrence nets $\mathcal{O}_1 = (B_1, E_1, {}^\bullet()_1, ()_1^\bullet)$ and $\mathcal{O}_2 = (B_2, E_2, {}^\bullet()_2, ()_2^\bullet)$, we say that $\mathcal{O}_1$ contains $\mathcal{O}_2$, denoted as $\mathcal{O}_2 \subseteq \mathcal{O}_1$, if $B_2 \subseteq B_1$, $E_2 \subseteq E_1$ and for all $t \in T_2$, ${}^\bullet(t)$ is the same in $\mathcal{O}_2$ as in $\mathcal{O}_1$; the same goes for $(t)^\bullet$.

Figure 4 illustrates the unfolding process of the net system in Figure 1a. The first iteration $\mathcal{O}_0$ represents the mapping of the initial marking M_0 of the original net system to the set of minimal conditions $Min(\mathcal{O})$. The rest of the iterations are as follows: for each iteration, we pick a co-set of conditions $B' \subseteq B$ and check the original net system for any enabled transitions (backward mapping thanks to the homomorphism function λ). If a transition t is enabled, we add a new event e labeled with $(t_j : e_i)$, where j is the transition identifier from the original net system and i is the incremental counter for the unique events' mapping. The new conditions labeled with $(p_m : c_n)$ represent the transition t postset $(t)^\bullet$, where m is the place identifier from the original net system and n is the incremental counter for the unique conditions mapping. Next, we connect the newly added event e with the chosen co-set B' as its preset and with the newly generated conditions as its postset. Since the events and conditions have an incremental unique mapping, the occurrence net may contain several events, conditions mapped to the same transitions, and places of the original net system. For example, even though the conditions $(p_7 : c_6), (p_6 : c_7)$ are computed in $\mathcal{O}_4$, they are also generated in $\mathcal{O}_5$ due to the firing of t_5, which yields $(p_7 : c_8)$ and $(p_6 : c_9)$ as a postset.

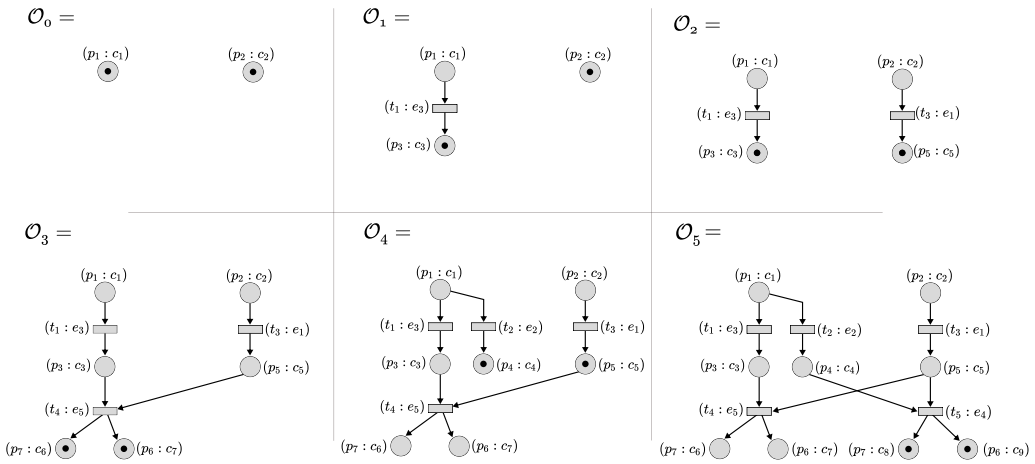


Figure 4. An iterative illustration of the unfolding process of the net system in Figure 1a.

We say that a branching process $\beta_1 = (\mathcal{O}_1, \lambda_1)$ contains $\beta_2 = (\mathcal{O}_2, \lambda_2)$, if $\mathcal{O}_2 \subseteq \mathcal{O}_1$. A branching process is maximal if it contains all other branching processes of a net N. An unfolding of a net system Σ is the maximal branching process of Σ. The unfolding of the net system in Figure 1a is infinite due to the infinite occurrence sequences, as shown in Figure 5.

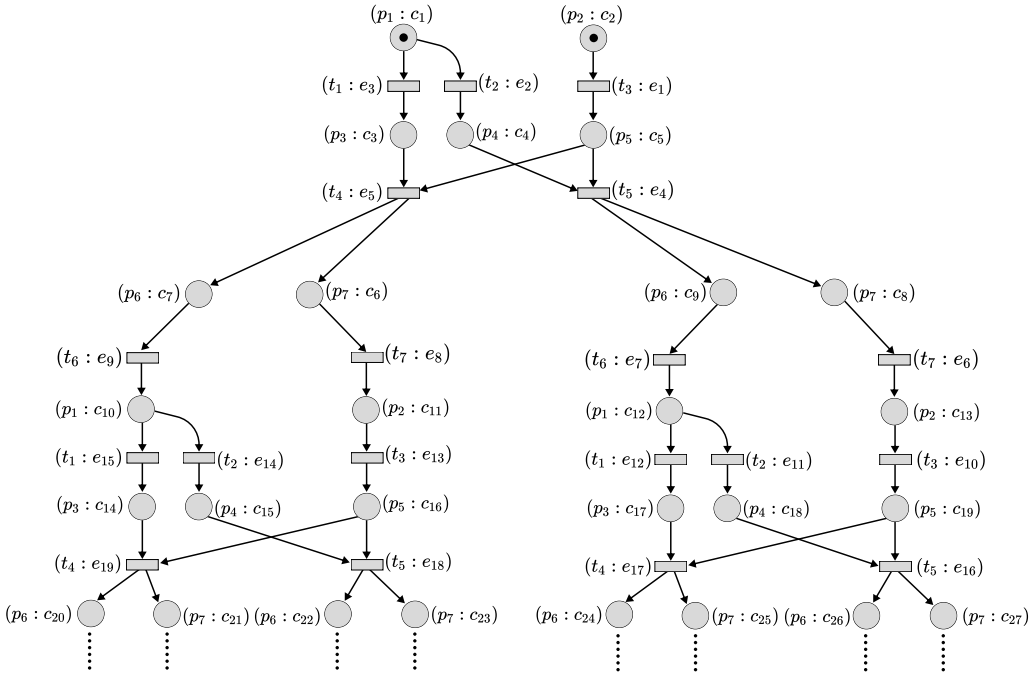


Figure 5. A graphical representation of the maximal branching process (unfolding) of the net system in Figure 1a.

4.4. Cuts and Configurations

The maximal branching process is an equivalent net tended to be used in the verification of net systems. Although the branching process for a cyclic net system is infinite, as illustrated in Figure 5 above, it is always possible to truncate such a branching process up to a finite "complete" prefix. The latter possesses all information about the original net system (i.e., it contains the set of reachable markings of the original net system). This complete prefix is called a "Finite Complete Prefix" (FCP). Various approaches have been developed to generate the FCP for different classes of PNs, starting from [16,56] for generic P/T net systems, and now to a higher level of PN models such as high-level nets [39], contextual nets [42], and colored nets [40,41]. Before we jump into the FCP construction, let us first introduce several essential notions.

Let $\mathcal{O} = (B, E, {}^\bullet(), ()^\bullet)$ be an occurrence net obtained from the branching process of a net system Σ. A set of events $\mathcal{C} \subseteq E$ is a configuration if the following statements hold:

(1) if $e \in \mathcal{C}$, then $e' < e$ implies $e' \in \mathcal{C}$;
(2) $\mathcal{C}$ contains no mutually conflict events.

Figure 4 shows the step-by-step occurrence nets obtained from the unfolding of the net system in Figure 1a. In the occurrence net $\mathcal{O}_5$, the events in $\{(t_1 : e_3), (t_3 : e_1), (t_4 : e_5)\}$ form a configuration.

Based on the definition of configurations, if an event e is in a configuration $\mathcal{C}$, then all the events preceding e should be contained within the same configuration as well. Furthermore, any event concurrent with e is allowed to be included in the same configuration.

Let $e \in E$ be an event of $\mathcal{O}$. The minimal configuration that contains e and all the rest of events preceding e is called a local configuration of the event e, denoted by $[e] = \{e' \in E | e' \leq e\}$. The notion of the local configuration originated in [16] signifies the cause of e to occur. For instance, in the occurrence net $\mathcal{O}_5$ in Figure 4, $[t_4 : e_5] = \{(t_1 : e_3), (t_3 : e_1), (t_4 : e_5)\}$ represents the local configuration of the event e_5.

A cut of a configuration $\mathcal{C}$ corresponds to a marking of the original net system reachable from the initial marking M_0. Following the original notation in [16], a cut is the final marking of $\mathcal{C}$, denoted by $FM(\mathcal{C})$. A final marking of a local configuration of an event e is called a basic marking of e, denoted by $BM(e)$.

Let $\mathcal{C}$ be a configuration of an occurrence net obtained from a branching process β of a net system Σ. A final marking $FM(\mathcal{C})$ is a cut reachable from the set of minimal conditions $Min(\mathcal{O})$ after all the events from $\mathcal{C}$ and only those events are fired. This final marking (Cut) is denoted by $FM(\mathcal{C}) = (Min(\mathcal{O}) \cup \mathcal{C}^\bullet) \backslash {}^\bullet\mathcal{C}$.

The association between the final markings of the branching process β and the reachable markings of the original net system Σ can be formalized as follows:

Let $M : \lambda(FM(\mathcal{C})) \rightarrow N^{|P|}$ be a mapping such that $M(\lambda(FM(\mathcal{C})))_i = |\{b \in FM(\mathcal{C}) \mid \lambda(b) = p_i\}|$, where i is the i-th element of the vector $M(\lambda(FM(\mathcal{C})))$ representing the number of copies of the place p_i from the original net system Σ in $FM(\mathcal{C})$. It has been proved in [56] that with a given configuration $\mathcal{C}$, $M(\lambda(FM(\mathcal{C})))$ is a reachable marking of the original net system. For example, in Figure 5, the configuration $\mathcal{C} = \{(t_1 : e_3), (t_3 : e_1), (t_4 : e_5)\}$ yields the following final marking: $FM(\mathcal{C}) = FM(\{(t_1 : e_3), (t_3 : e_1), (t_4 : e_5)\}) = \{(p_1 : c_1), (p_2 : c_2)\} \cup \{(p_3 : c_3), (p_5 : c_5), (p_6 : c_7), (p_7 : c_6)\} \backslash \{(p_1 : c_1), (p_2 : c_2), (p_3 : c_3), (p_5 : c_5)\} = \{(p_6 : c_7), (p_7 : c_6)\}$. Now, since the final marking is derived, we have the reachable marking of the original net system $M = M(\lambda(\{(p_6 : c_7), (p_7 : c_6)\})) = M(\{p_6, p_7\}) = [0000011]^T$, which is equal to M_5, as Figure 3a shows. Note that, since $\mathcal{C}$ is the local configuration of the event e_5, the final marking $FM(\mathcal{C})$ is also the basis marking $BM(e_5)$.

4.5. Finite Complete Prefix

Since the unfolding of a cyclic net system is infinite, as demonstrated previously, targeting the construction of a finite and complete prefix is inevitable. The study in [16] came up with an elegant idea to truncate the unfolding into one single prefix, which represents the set of all reachable markings without the need to entirely unfold the net system into a maximal branching process as long as it is bounded in most cases. The idea was about stopping the unfolding construction when a particular break-off condition is fulfilled. This is accomplished by introducing the notion of cut-off events. These events serve as redundancy detectors (i.e., if a reached marking has already existed, the current event is considered a cut-off event). In this part, we go through the cut-off events criteria and the construction of the finite complete prefix (FCP).

Let $\mathcal{C}$ be a configuration in a branching process β and e be an event belonging to $\mathcal{C}$. Then $|\mathcal{C}|$ stands for the size (number of events) of the configuration $\mathcal{C}$. $||[e]||$ stands for the size of the local configuration $[e]$ of the event e. In the occurrence net, the depth of a condition b is the number of events preceding b. Accordingly, if $e < b$, then $||[e]||$ also stands for the depth of b. Conditions with the same depth are called a tier. A tier covers a set of final markings. A new tier is formed by listing all the markings reached from a marking in the previous tier after firing one event.

Let $\mathcal{O} = (B, E, {}^\bullet(), ()^\bullet)$ be an occurrence net of the branching process β of a net system Σ. An event e of the occurrence net $\mathcal{O}$ is a cut-off event, if there exists another event $e' \in E$ such that

(1) $M(\lambda(BM(e))) = M(\lambda(BM(e')))$, or $M(\lambda(BM(e))) = M_0$;
(2) $||[e]|| > ||[e']||$.

Figure 6 shows the finite complete prefix β_f of the maximal branching process β_m in Figure 5, which contains only the first tier of cut-off events and allows the detection of cycles, serving as an essential criterion for net systems verifications [16]. The events $\{(t_4 : e_5), (t_4 : e_{17}), (t_5 : e_{16})\}$ are considered as cut-off events and depicted by black boxes. An infinite maximal branching process contains infinite copies of the same exact finite complete prefix.

Local configurations are critical in the construction of FCPs, and their size defines the order in which different events are generated. Since the occurrence net is basically a partially ordered set of nodes, it makes sense that the selection decision between e and e' relies entirely on the basic markings isomorphism and local configurations' size. The unfolding process is continued tier by tier until the events enabled by the most recent tier are cut-off events.

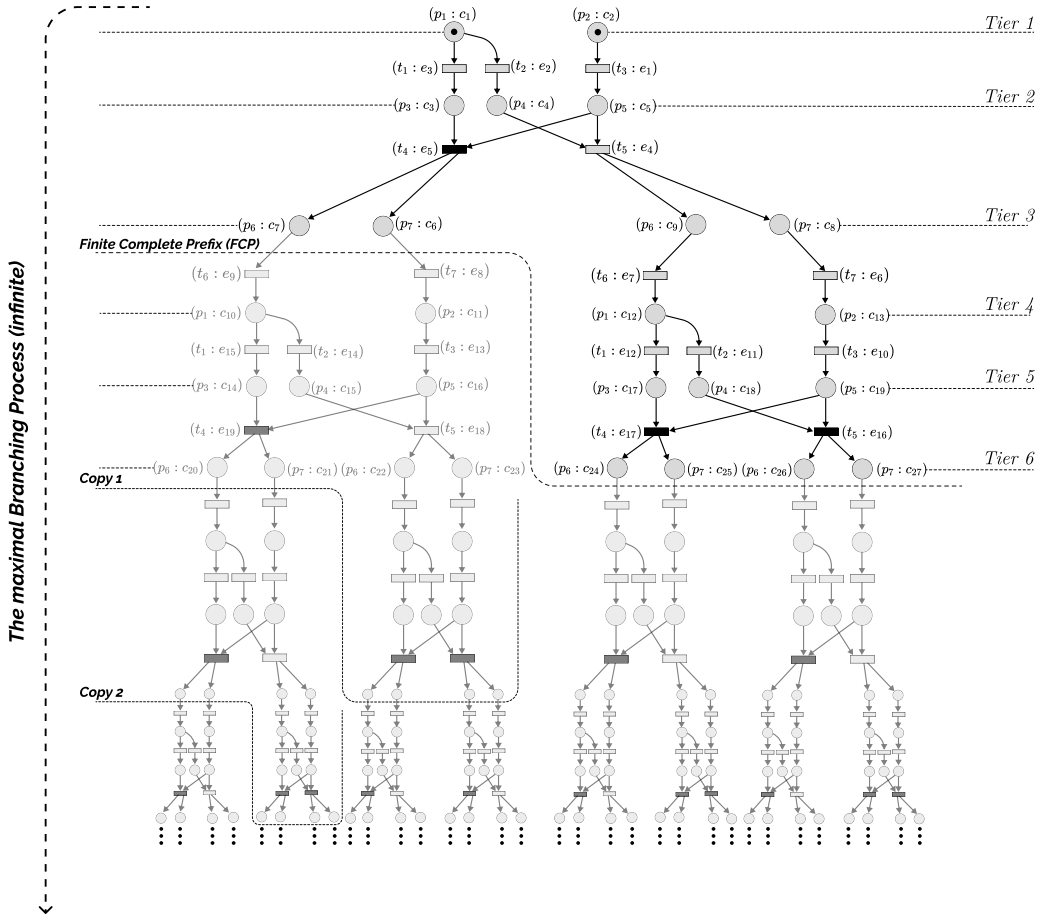


Figure 6. An illustration representing the truncation process of the final complete prefix of the maximal branching process in Figure 5.

A branching process $\beta = (\mathcal{O}, \lambda)$ is a finite complete prefix of the unfolding $\beta_m = (\mathcal{O}_m, \lambda_m)$ if $\mathcal{O}$ is contained by $\mathcal{O}_m$. An FCP β_f is obtained from the branching process β_m by removing all the conditions and events which succeed cut-off events.

A marking M is a reachable marking of the original net system Σ if and only if there is a configuration $\mathcal{C}$ of β_f such that $M = M(\lambda_f(FM(\mathcal{C})))$. For any transition t of Σ that is reachable from M_0, there exists an event e of β_f such that $\lambda_f(e) = t$.

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two configurations of β_f and $\mathcal{C}_1 \subseteq \mathcal{C}_2$. Then, in the original net system Σ, $M(\lambda_f(FM(\mathcal{C}_2)))$ is reachable from $M(\lambda_f(FM(\mathcal{C}_1)))$.

For bounded net systems an FCP β_f is a finite acyclic net. It is proved in [16] that all the reachable markings of the original net systems exist in the FCP. The key concept behind this is the absence of any newly generated final marking for a cut-off event, along with its causal history.

We close this section with an example to demonstrate the truncation process. Consider the original net system in Figure 1a. The initial marking M_0 forms the first tier of the FCP, as Figure 6 illustrates. The second tier is generated by enumerating the final markings of all configurations corresponding to the events enabled by the first tier. In this example, there are three events, namely $\{(t_1 : e_3), (t_2 : e_2), (t_3 : e_1)\}$, where $(t_3 : e_1)$ is concurrent with either $(t_2 : e_2)$ or $(t_1 : e_3)$; thus, every element in the power set of $\{(t_3 : e_1), (t_1 : e_3)\}$ and the power set of $\{(t_3 : e_1), (t_2 : e_2)\}$ represents a feasible configuration. Firing all these configurations forms the second tier of the finite complete prefix. Unlike the reachability graph, the enumeration of arbitrary interleavings is not required in the unfolding. For instance, transitions $\{t_1, t_3\}$ form a diamond effect due to the interleaving semantics in Figure 3a, which is the main cause of the state explosion problem, while in Figure 6 the properties

of partial-orders preserve the concurrency between events and avoid enumerating the unnecessary interleavings.

In the same way, the third tier is generated by enumerating the final markings of all configurations containing new events that are enabled at a final marking from the second tier. For example, we add event $(t_4 : e_5)$ and conditions $\{(p_6 : c_7), (p_7 : c_6)\}$ to the occurrence net $\mathcal{O}_f$, since the event e_5 is enabled by $FM(\{(t_1 : e_3), (t_3 : e_1)\})$ and $\lambda_f((t_1 : e_3), (t_3 : e_1), (t_4 : e_5)) = \{p_6, p_7\}$. Since the marking $\{p_6, p_7\}$ is already presented by $\{(p_6 : c_9), (p_7 : c_8)\}$ (i.e., the event $(t_5 : e_4)$ is explored before $(t_4 : e_5)$ and $M(\lambda(BM(e_4))) = M(\lambda(BM(e_5))) = M_5$), the event $(t_4 : e_5)$ is considered a cut-off event. The rest of the tiers are obtained in the same way. In constructing the sixth tier, we stop the truncation process with the cut-off events $\{(t_4 : e_{17}), (t_5 : e_{16})\}$, where both of these events represent the basic marking of the event $(t_5 : e_4)$ with a longer size of local configurations: $M(\lambda(BM(e_{17}))) = M(\lambda(BM(e_{16}))) = M(\lambda(BM(e_4))) = M_5$.

5. Net Unfolding Applications

After the previous sections, readers might assume that DESs are fundamentally complex and challenging to analyze, despite the chosen framework. It might also be reasonable to investigate whether there are any properties in DESs that we can examine to construct mathematical approaches for analysis, design, and control. The partial-order approach and its early foundations, such as the branching processes of Engelfriet and the McMillan unfolding technique, are one of the adopted approaches for such purposes. Due to its true concurrency nature, the partial-order approach has observed massive adoption in recent decades when it comes to the automatic verification of the DES properties. In this section, we recall the various types of verification methods and the different specifications that net system properties could take, and then we highlight the significant recent studies concerning the automatic verification of DESs properties using the interleaving semantics (i.e., reachability graph). After that, we focus on the methods based on the partial-order reduction, all of which are from a conceptual and technical perspective. Readers may refer to [5,19,57] for more details.

5.1. Properties & Formal Verification

Designing net system models from imprecise specifications is a complex process that necessitates extensive modeling expertise in the field as well as enough awareness of model development strategies. Consequently, a net system model might deviate significantly from its initial description. This is particularly the case when working with massive net models of advanced systems that are tough to make their abstractions.

Model synthesis by experience is not fully reliable and feasible in most cases. As a result, using formal approaches for problem-solving makes the implementation of a robust model a crucial task, especially for a model that is correct in relation to a given specification. The correlation of both an initial specification and its net system representation gives designers enough feedback that in many cases allows them to refine their understanding of the system.

There are several interpretations of correctness. Essentially, a system is correct when the model representing the specification and the model representing the implementation are consistent, or when the system shows a set of desirable properties chosen from a predefined set (e.g., liveness, finiteness, and forbidden states) [22]. These properties enable the designers to determine the existence or absence of the application-domain-specific practical properties of the system being designed.

The challenge that arises when choosing properties for net systems is identical to the case when choosing the net system model itself. While a limited range of properties is unable to capture the many aspects of protocols or distributed solutions, defining a large set of properties prevents the construction of reliable defined methods.

If one is going to impose constraints on the properties, then these properties need to be generic in the way that they reflect the behavior of the modeled system across

a wide variety of interpretations, for instance, deadlock-freeness, liveness, and finiteness. Regarding the generality of the previously mentioned properties, there will still be certain behavioral characteristics that cannot be described by a static set of properties. Therefore, it is preferable to use a property language that is specialized for dynamic systems, particularly those with a higher degree of concurrency. The temporal logic-based framework is one of these languages that has demonstrated extensive adoption for net systems [58]. This advancement is due to the fact that most properties of concurrency can be described by straightforward formulations, and that the verification related to such formulas can be simply transferred to the reachability graph. Among other forms, there are also the bisimulation-based equivalence (i.e., imitating what a net system executes by means of reachable markings and/or firing sequences), process algebra (i.e., transferring the net model into process algebra throughout the design of a system to simplify mathematical deduction), and finally the refinement of equivalence by discriminating between true concurrency and interleaving concurrency.

It might be difficult for an engineer to determine the best ways to address his or her problem, given the wide variety of verification methods established for Petri nets and their expansions. One way to maintain complexity is to use highly expressive models such as high-level Petri nets. However, this level of expressiveness comes with its own set of challenges in the verification phase. Meanwhile, some elevated models incorporate flexible tuning that significantly expands the scope of findings.

The methods may be categorized based on certain fundamental characteristics, for example, supported nets and properties, structure and behavior adaptation, and automation level. Moreover, if one can crucially recognize how a method could potentially benefit from the findings of others, then it is possible to combine such methods. For example, even if Petri nets are used to model a system, they are not necessarily required for the verification process. In this sense, transferring into a process algebra could prove to be useful to avoid any exploiting difficulties. Nevertheless, the most crucial requirement are the Petri nets' aspects, where the way they are exploited provides feasible information about the verification method that should be adopted. For instance, one could consider a Petri net to be a graph with token circulation, leading to the choice of graph theory as the adopted method. As an alternative, one could consider the net to be a linear transformation of numerical vectors; in this case, linear algebra is the closest choice. Another type of verification method is the state-based approach, such as mimicking behaviors with colors, which is referred to as "color-based analysis", or relying on event structures where causal relations are well founded, i.e., the "partial-order approach". Moreover, formal logic can be used to reason out the properties of nets; this is the logic verification method.

With the formal specifications of the net system and the property, the verification phase is set to be automated under what is known as the model checking process, see [21]. Verifying a property requires accurate determinations about the net system runs, and whether they are consistent with the specified property or not, supposing that the state space of the net system has a finite size.

Although model checking techniques provide a comprehensive foundation for systems verification, the complexity involved rises exponentially with the number of expressed objects. Concurrency and, more specifically, the interleaving semantics adopted to express each feasible action sequence are largely the consequence of this limitation. For such a reason, space reduction techniques are usually embedded with the model checking to deliver an industrial size implementation.

The verification of a net system involves determining if it meets a set of properties generated from its specification. Properties of concurrent systems may be categorized based on the behavior they reflect. The two major types of properties are liveness and safety. A liveness property indicates that good things do occur. All system runs need to conform these properties. A safety property indicates that bad things do not occur during runtime. Other properties are claimed to be variations of decomposition between the safety and liveness properties.

There are three main partial-order-based techniques that serve state-space reductions and property verification using the trace theory of Mazurkiewic. The first is the persistent set technique with its complementary technique of the sleep set of Godefroid [6,13,14]. The second one is the stubborn set technique of Valmari [12,27]. Both of these techniques rely on the partial orders to reduce the reachability graph computation by deleting the firing sequences that are considered redundant for the trace computation. The third technique, which is the one we are interested in and covering throughout this survey, is the approach proposed by Nielsen et al. in [15,26,38] (known now as the branching processes method; see Section 4.1), which attempts to explicitly express the partial firing order of transitions by using the concepts of causal relations (i.e., concurrency, conflict). The objective of the branching process approach is to generate a direct Petri net model of the possible partial order of system transitions. The translation of the original net system into its equivalent branching process is noted as an unfolding, because all transitions and places of the original net might have multiple instances within a process, based on the potential executions of the transitions. In this model, independent events are modeled by independent transitions, reflecting their interleavings without computing them.

Partial-order methods and their early foundations enable the blooming of verification methods to verify a wide variety of system properties and extend their application to various types of Petri net models. In the next sections, we represent and investigate the latest achieved contributions within the automatic verification of DESs that we are aware of based on the specification of properties they are implemented for. We first take up the verification methods based on the interleaving semantics representation, and then we focus on the true concurrency ones. We also review the application of such methods within the supervisory control and fault diagnosis communities. Furthermore, we detail the net unfolding generalizations for different Petri net models.

5.1.1. Fundamental Properties' Verification

Verifying fundamental properties such as safety, liveness, and deadlocks using interleaving semantics drew the interest of many researchers. To achieve efficient verification algorithms using interleaving semantics, the majority of these methods rely on supervisory control theory. We highlight these studies in Section 5.1.3, where we cover the supervisory control applications. For the time being, we will concentrate on net unfolding-based methods for verifying fundamental properties.

In [16], McMillan proposed the first unfolding-based algorithm to verify deadlocks and reachability properties, where the notion of cut-off events was introduced to generate complete finite prefixes as a solution to terminate the infinite unfolding process. The application of such an approach in the verification of asynchronous circuits using hash tables and branch and bound techniques for deadlock detection was demonstrated. Several types of tests (such as distributed mutual exclusion, the dining philosophers paradigm, and so on) were included.

However, Esparza et al. in [45,46] claimed that the McMillan algorithm is only applicable for systems in which the local configurations sizes are guaranteed to be different; otherwise, the cut-off events would not be detected, yielding fatal errors concerning FCPs' construction. As a result, they introduced an improvement to the McMillan algorithm known as the "ERV unfolding". The latter relies on the branching processes of [17] to avoid the generated unfolding being larger than necessary, and even exponentially larger in the worst case. They introduced the concept of "Foata normal forms" and total adequate orders as a way to improve the possible extensions' *pe* of processes (i.e., enabled transitions that could eventually be fired, yielding a new reachable marking). The space complexity of this technique is claimed to be linear. However, the time complexity heavily depends on the possible extensions' computation. Thus, possible extensions are considered the most critical aspect of the unfolding construction.

On the other hand, in [49], Melzer and Römer introduced a faster implementation of McMillan's algorithm. They claimed that the backtracking approach in McMillan's work

is considered as a drawback for widely increased unfoldings. The enhanced approach was tested using the same tests that are used by McMillan in his work, resulting in better performance using the linear algebra approach if the number of cut-off events is high. They also provided an optimized version of the McMillan algorithm to clarify the existing gap between the LISP implementation of McMillan and the new optimized version.

A similar study was due to Khomenko and Koutny in [51], where they developed a mixed integer linear programming reduction method for deadlock detection based on "Contejean-Devie's algorithm" to solve systems of linear constraints by transferring the causality relations of the events into corresponding integer variables. The new approach is claimed to have higher performance indicators due to the specified implementation of ILP problem solving, while the worst case is the lack of concurrent events merged with a minimal number of cut-off events, leaving the problem to be NP-complete due to the time complexity taken to solve the linear constraints. Furthermore, in 2006, Khomenko et al. proposed a new compact representation of net systems, called a "merged process". The authors claimed that it significantly copes with non-boundedness and different choices. The concept behind this approach is to merge specific nodes in the FCP of the original net system and apply the verification purposes on the resulting compacted net. The technique was also applied to deadlock detection and demonstrated that the merged process could potentially be significantly smaller than the original unfolding representation.

Moreover, Heljanko introduced another approach using stable model semantics with SAT-solvers as a logic programming base to transform the deadlock detection from prefixes into a mathematical reasoning problem using constraints and linear size projections. The "smodels" system, which is a logic programming framework based on constraints, is used in this method to find a stable model that fits the deadlock detection problem.

Other studies focused on the reachability problem, such as Schröter and Esparza in [52]. They introduced a new solution with its time complexity being $O(n^k)$ by reducing the reachability problem to a "CLIQUE" (i.e., a graph theory concept for induced subgraphs), where n is the size of the FCP of the maximal unfolding containing all reachable markings and k is the number of places that should be concurrently marked. In other words, the algorithm computes a set of "complementary markings" using concurrency relations. They also provide a comparison between three kinds of algorithms that we mentioned above (i.e., ERV unfolding [45,46], ILP approach [49], and SAT-solvers approach [51]) and the one they developed. As a result, the ERV unfolding and SAT-solvers approaches had better efficiency compared with the CLIQUE reduction approach. Unlike Schröter and Esparza, Chatain and Paulevé in [59] focused on the reachability of one single marking called the "goal". They proposed an algorithm for FCPs computations dedicated to reaching a specified reachable marking by identifying minimal configurations, that is, with a combination of the net unfolding technique and a model reduction approach. Such a technique is highly required in the biological community, where biologists need to verify the reachability of a specified node with biological networks.

It is worth noting that, to the best of our knowledge, none of the preceding studies included unbounded network systems in their development. The majority of these studies used 1-bounded net systems, and some of them claimed the appliance of the techniques to k-bounded net systems. However, in 2000, Abdulla et al. represented the unfoldings for models based on unbounded Petri nets to verify the safety properties [60]. The proposed approach relies on backward reachability analysis based on the well-quasi-orders theory. The symbolic algorithm uses constraints rather than the traditional cuts (final markings) computation. A constraint denotes upward-closed sets of final markings used to generate a "Reverse Occurrence Net". The algorithm terminates by computing the postfix of the reversed net. At the same year, He and Lemmon proposed a liveness verification method for DESs modeled with k-bounded Petri nets in [61]. The method was reviewed by Xie and Giua with sufficient counter-examples in [62] to refine some of the essential concepts. The developments made by He, Lemmon, Xie, and Giua are covered under Section 5.1.3 since they are more concerned with the control theory community.

5.1.2. Security

Non-interference has been proposed to describe the lack of unwanted information flows within safety critical systems [63], covering a large range of human-being infrastructures. The core principle is straightforward: A system is thought of as a composition of elements with various degrees of secrecy, in the basic form with a high-level component H, which should naturally be kept private, and a low-level component L considered to be public. Opacity, another form of security properties [64] has been investigated in the context of labeled Petri nets (i.e., a class of Petri nets that relies on the observability of transitions to make system state estimation and further preform verification purposes) with interleaving semantics as a general notion that may capture several types of privacy properties, such as non-interference with degradation. In [8,65,66], Tong et al. focused on solving and optimizing the opacity problem within DESs, such as current and initial state opacity and language-based opacity.

Starting from [8], Tong et al. proposed a verification method based on the interleaving representation of the state space of a system that is modeled with labeled Petri nets, where they describe the "secret" behavior of a subjected net system as a set of states. The approach generates a set of short-path-related markings, called "basis markings", which represent what is called the "basis reachability graph (BRG)". The latter is used to verify the initial and current state opacity and is claimed to be efficient since the computation avoids the exhaustive enumeration of the state space. The basic idea of verifying whether a net system is secure or not is to check whether a set of estimated markings consistent with a given observation (i.e., an eligible sequence of transitions that may be fired yielding a reachable marking) fully belongs to the secret or not. If the set is not fully included, then the system is said to be secure or opaque. The approach was later followed by several optimizations, including weakly exposable basis markings and basis observer construction, where the net system secret is described by integer linear programming constraints. Furthermore, the problem of opacity within net systems modeled by labeled Petri nets was demonstrated to be undecidable in [65] for initial and current state opacity as well as language-based opacity. Using supervisory control to enforce the opacity property has also been reported in [66], where Tong et al. proposed a finite structure called the "augmented 1-observer" to design a local optimal supervisor. The novel approach claims lower complexity than existing methods and employs finite automata as a system representation model. Moreover, in [67] Cong et al. proposed a centralized and decentralized online approach for verifying current-state opacity. The approach captures the occurring events (i.e., more precisely, observable events) and uses ILP to determine the opacity of the system. The method was also extended to decentralized architectures, where a coordinator is required to synchronize the communications between local intruders and provide final outcomes about the current-state opacity of a given observation.

For the verification approaches based on true concurrency semantics, Baldan and Carraro in [68] proposed the first net unfolding approach for the Bisimilarity-based Non-Deducibility on Composition (BNDC) property. A form of non-interference says that a process $\mathcal{P}$ is secure from interferences if $\mathcal{P}$ is operating in isolation. The authors established a causal formulation of non-interference by concentrating on Petri nets and the BNDC property, and furthermore they defined BNDC in terms of causalities and conflicts over high and low level behaviors for 1-bounded Petri nets. They also claimed the applicability of their approach for unbounded Petri nets. As a result, Baldan and Carraro developed a verification algorithm for the BNDC property based on the unfolding technique. Furthermore, they implemented a tool for interference checking called the "Unfolding-Based Interference Checker" (UBIC) for short, which demonstrates impressive results compared with the already existing tools when it comes to time efficiency. Later on, Baldan et al. in [69] extended their work to the Bisimilarity-based Intransitive Non-interference (BINI) property, where the approach also has a high-level H, a low-level L and an additional concept called the "downgrading" actions D. It declassifies all previously occurring secret actions that are formerly regarded as secret. Similar to their original work, the authors

established a formulation of non-interference by concentrating on Petri nets and the BINI property. They also defined BINI in terms of causalities and conflicts over high and low levels of behaviors for 1-bounded Petri nets. A UBIC-2 tool was also developed to implement the Intransitive Non-interference property.

5.1.3. Supervisory Control

The development of supervisory control theory for discrete event systems and their logic controller design can be traced back to the early 1980s. Ramadge, Wonham, and their team at the University of Toronto distinguished between the controller and the controlled DES plant, resolving a problem similar to the typical control of continuous variable dynamic systems [20]. This ended up leading to the formalization of controllable/observable events and uncontrollable/unobservable events, resulting in control problem synthesis for discrete event systems at the logic level [20]. As we mentioned in Section 5.1.1, before we jump to the verification using true concurrency semantics, we mention some of the recent achievements acquired in recent decades, when it comes to interleaving semantics. In [19,70–78], the authors conduct intensive research studies covering the potential deadlocks of net systems modeled by Petri nets. Li et al. in [73] followed a divide-and-conquer strategy to achieve a deadlock prevention policy for flexible manufacturing systems (FMSs) modeled by a class of Petri nets, called S^3PRs. They represent a plant model with three separate subnets, namely the "idle, autonomous, and toparchies" subnets, respectively. Then, a set of toparches supervisors is designed for each toparchy and merged with the autonomous subnet, resulting in a new net called the "monarch". The latter is claimed to be the overall liveness-enforcing supervisor for the plant model. Furthermore, in [74], Chen and Li designed a deadlock prevention policy for flexible manufacturing systems using a vector covering approach. The proposed method generates a pair of marking sets representing the minimal covering of legal markings and the minimal covered forbidden markings. The latter is used to design a maximally permissive supervisor for deadlock prevention using integer linear programming (ILP). The method clarifies the obstacle of the state explosion problem due to the full enumeration of the set of reachable markings and the ILP computation. Moreover, the work in [19] contains a comprehensive guide to resolving the deadlock property in automated manufacturing systems using a novel Petri nets approach. It is worth noting that deadlock problems are critical and significant in highly automated resource allocation systems such as semiconductor manufacturing. As mentioned, the complete state enumeration remains the root of the difficulty in finding an optimal control policy to enforce deadlock-freedom given a resource allocation system. Net unfolding could be an effective solution to this notorious problem.

For the concurrency-based methods, we mentioned in Section 5.1.1 that He and Lemmon proposed a liveness verification and enforcing method for DESs modeled with *k*-bounded Petri nets using net unfolding. They provided an approach for cycles detection, since liveness is directly connected to the cyclical behavior of a net system. Furthermore, they demonstrated how cycles could be obtained from the occurrence net and the possibility of verifying the liveness property by checking the liveness of every event cycle, in addition to the occurrence of cyclic locks (i.e., a lock leads to a deadlock) within a specific set of cycles. Moreover, He and Lemmon expanded their work by formulating the necessary and sufficient conditions for the design of a maximally permissive supervisor for liveness enforcement. The approach computes a set of event invariants called the "base configurations". The latter are used to obtain every basis execution of a net system, which forms what is known as the "cut graph". Each node in a cut graph represents a basis execution and the possible interleavings to other basis executions as outgoing arcs. A cut graph is claimed to represent a high degree of abstraction when it comes to system executions. However, in [62], Xie and Giua provided a set of counterexamples to the approach proposed by He and Lemmon. These counterexamples concern liveness analysis, liveness enforcing supervision, and the maximally permissive supervisor [79]. In addition, the fundamental theorem proposed in [61] is concerned with liveness verification conditions.

The control problem of DESs with net unfolding has also been visited by Giua and Xie in [80], where, under a set of assumptions, the authors applied the unfolding technique to design a maximally permissive supervisor to prevent reaching a set of prohibited markings for safe Petri nets. The approach consists of adding a set of controlling places to the unfolding. The applied assumptions include detecting prohibited markings specified with the so-called "REACH" property, which causes the markings reached from a REACH marking to be prohibited as well. The latter is efficiently implemented due to the configuration properties of the unfolding (i.e., a prohibited configuration results in extended configurations being prohibited as well). The authors claimed that the proposed approach needs an exhaustive enumeration of the prohibited markings set. It does, however, have the advantage of allowing the construction of controlled occurrence nets. Later on, in [81] the approach was improved by adopting linear algebra techniques to design a maximally permissive supervisor by satisfying a set of linear inequalities.

Another work by Buy et al. exploited supervisory control using the net unfolding technique; however, the approach was applied to timed Petri nets [82]. The approach consists of deadline enforcement for firing transitions. Given a net system with a target transition and a deadline, the method generates a controller that, with very general assumptions, triggers the transition to fire per each deadline time unit. The online supervisory controller of this method is comprised of two sub-nets added to the controlled net to impose deadlines on the firing of targeted transitions. The first sub-net is a "clock sub-net", which monitors the remaining time until the deadline expires, and the second one is a "supervisor sub-net", which is responsible for disabling transitions with expired deadlines. After the targeted transition is fired, the disabled transitions are enabled normally. The method is claimed to be polynomial and completely depends on the size of the unfolding.

5.1.4. Fault Detection and Diagnosis

Fault detection and diagnosis in DESs have gathered considerable interest (see [83] for an introduction to the topic) from researchers, using either interleaving semantics or true concurrency semantics. For the studies based on interleaving semantics, multiple researchers and practitioners have been intensively investigating the fault detection and diagnosis area [84–88]. Specifically, in [86], Zhu et al. adopted the partially observed Petri nets as models to develop a model-based fault identification approach for DESs. The model representing the system is considered to be free of fault events, called the "nominal" net with its partial places being observable. The computation of the "observed evolution" (i.e., a sequence related to transitions and markings of the observable places) determines whether a fault has occurred or not. The approach depends on ILP solving to detect the faults that occurred within the system, which are consistent with the nominal net and the observed evolution. Furthermore, Cong et al. in [84], investigated decentralized diagnosis within systems modeled by Petri nets combined with ILP to provide a novel on-line fault diagnosis approach. The proposed method follows a decentralized strategy constructed from a set of local sites exchanging information with a central coordinator, which is responsible for deciding whether a system behavior is free of faults or not. The approach uses two protocols defined by the diagnostic information and operation managed by local sites.

True concurrency semantics, on the other hand, led a group of French researchers from INRIA and the University of Rennes (France) to exhibit a series of studies on the usage of net unfolding technique to diagnose distributed asynchronous systems that are usually partially observed systems [89–96]. The first appearance was from Benveniste et al. in [89], where they introduced a true concurrency-based approach to diagnosing asynchronous DESs (i.e., distributed systems) modeled by labeled Petri nets. The approach mainly supports distributed faults, event propagation, and distributed sensor setups in addition to fault management in telecommunications networks. Furthermore, rather than constructing an observer for a net system, the approach followed here is based on computing an estimated set of explanations in terms of scenarios. These explanations are generated using "Pattern

Matching Rules" to construct the diagnosis nets, which are later used to diagnose a net system based on the unfolding approach.

In [91,92], Fabre and Benveniste introduced the "Trellis processes", serving as a new compact representation of the unfolding technique for the executions of concurrent systems. The core of unfoldings is their factorization characteristic. This characteristic asserts that the unfolding of a composed system can be described as the product of the unfoldings of its components, by assuming a suitable definition of a product. This factorization offers an additional tool for further condensing the data structure reflecting all concurrent system executions, as the factorized representation is typically more condensed than the developed form. Its key advantage, however, is that it lays the foundation for modular processing by synchronizing many online diagnosis activities performed at the level of individual components. A net trellis representation can be generated directly using the developed approach or by a partial refolding of an already computed FCP (i.e., generated from the original net system and then condensed to a trellis process). The cost of having such a condensed representation is the elevated difficulty of the recovering configurations, such as system executions.

On the other hand, Haar in [93–95] introduced the facet-based and logical covering concepts of q-diagnosability for the purpose of qualitative diagnosability. These "Facets" are considered sub-nets of the overall unfolding, where every pair of events covers another. As a result, if one event in a facet occurs, all other events in that facet must inevitably occur for proper execution. Moreover, in [94], Haar proposed logical formulation and verification methods for the observability and diagnosis properties. He introduced the notion of "reveal-relation" due to the weak diagnosability verification purposes, which is claimed to be polynomial in the size of the second order unfolding and exponential in the size of the overall unfolding. Finally, Haar et al. proposed the first fault diagnosis verification based on the unfolding of partially observed net systems in [95]. The approach provided formal foundations and algorithms for the diagnosis problem (i.e., weak diagnosis verification) using SAT-solvers. The weak diagnosis exploits indirect dependencies, obtained by the derived relations, to examine system runs that explain a given observation pattern and decide whether an unobservable fault is unavoidable.

Partially observable systems were also diagnosed in [41] by Chatain and Jard. The authors proposed a symbolic approach to diagnosis and supervisory control for colored Petri nets systems using a set of explanations consistent with a given observation. The partially ordered explanations are used to construct a causal diagnosis graph to solve inference problems. A similar study is found in [96] by Hélouët and Marchand. The proposed approach uses disambiguation mechanisms and a cost model that is analogous to the energy model, where transitions can consume or restore the energy of the system. It uses observation-guided unfolding to generate a partial-order representation for the set of consistent states with the given observation. After that, it is used to establish a criterion for determining when a diagnosis can be made based on the characteristics of those processes explaining the observation. No fault means that the set of explanations processes is fault-free. Furthermore, if the overall unfolding includes both faulty and non-faulty processes, it is determined to be ambiguous. Using this approach, a net system is said to be diagnosable if and only if it can break from the ambiguity after a certain amount of time (within a limited number of observations or steps).

5.2. Generalizations

The technique of net unfolding is also extended to various kinds of Petri net models to benefit from the rich parametrization of these formal models. Previously, in Section 5.1.4, we mentioned the work of Benveniste, Haar, and Hélouët and Marchand, etc. These approaches rely on labeled Petri nets (i.e., fully or partially observable), since they are required to generate the set of consistent explanations of the executions of the concurrent systems. Furthermore, a universal characterization for colored Petri nets is found in [40], where Liu et al. introduced a technique to efficiently unfold colored Petri nets such that the

constraint satisfaction approach and pattern matching are applied to enhance the unfolding performance. The algorithm implementation showed effective results when unfolding large colored nets.

High-level Petri nets were generalized due to Khomenko and Koutny in [39,97–100], where a series of generalizations of an unfolding-based model checking framework were introduced in a Ph.D. thesis by Khomenko at Newcastle University, United Kingdom. It starts with improving the already existing algorithms for FCPs' computations by enhancing the possible extensions computation with the all-in-once approach rather than the classical one-by-one approach generalization [97]. In [98], Khomenko et al. introduced the "canonical prefixes" approach, which is dedicated to the unfolding trimming. The approach consists of terminating the unfolding based on a new definition of cut-off events called the "cutting context" to generate a finite and complete canonical prefix of the original unfolding. Furthermore, in [39], Khomenko and Koutny proposed the branching processes for high-level Petri nets, where they introduced a new approach that allows one to generate the FCP directly from a high-level net model known as the "M-net". Using the previous findings and a parallel unfolding approach, they achieved a concrete connection between the branching processes of M-nets and the branching processes of its low-level version. The approach is claimed to be applicable to the classical net unfoldings when it comes to the verification tools, and shows effective results compared with the classical ones. Moreover, in [100], Khomenko et al. reported a new compaction representation of the net unfolding called "Merged Processes" as a solution to the state-explosion problem. The approach combines the earlier mentioned canonical prefixes and uses "mp-conditions, mp-configurations" to represent the merged parts of the unfolding. As we mentioned in Section 5.1.1, the technique is claimed to effectively cope with non-boundedness and free choices.

Baldan et al. in [42], on the other hand, took the net unfoldings to another kind of Petri nets model, called contextual Petri nets, a type of nets dedicated to systems with read only resources. The approach applies a modified version of McMillan's unfolding technique to a finite-bound semi-weighted contextual net to generate a contextual FCP. Due to the existence of asymmetric conflicts, the approach is based on the idea of the various histories that events may have. During construction, the subsets of critical event histories are stored in the prefix. Moreover, a new concept of cut-off events was introduced by adopting contextual Petri nets. As a result, the approach was proven to generate more condensed FCPs than the ones generated using the original technique for the class of contextual Petri nets. Another study concerning contextual nets by Schwoon and Rodríguez is found in [101]. The authors combined contextual nets with a SAT-based verification to develop an efficient technique for properties, verification, such as the reachability problem.

A class of nets called "general Petri nets", characterized by multisets of places and arcs, gained the interest of Hayman and Winskel in [102], who proposed a generalization of the net unfolding technique to the general nets. The approach takes advantage of the implicit symmetry between paths in the unfolding to obtain cofreeness up to this symmetry, thus generalizing the unfolding for the general Petri nets.

Several other Petri nets classes have been covered for the unfolding appliance, such as timed Petri nets in [103], nested nets in [104], and reset nets in [105]. As an encapsulation for this section, Table 2 represents a summary of the different unfolding representations and verification algorithms discussed throughout this section.

Table 2. A list of the major contributions found within the concurrency community for recent decades concerning the net unfolding approach.

Title	Refs.
McMillan's unfolding technique	[16]
ERV unfolding technique	[45,46]
ILP-based approaches	[49,51]
SAT-based, Symbolic-based approaches	[41,50,60,94,95]
Graph-based (CLIQUES)	[52]
Goal-driven	[59]
Canonical Prefixes	[98]
Merged processes	[100]
Trellis processes	[91]

6. Conclusions

In this paper, we have recalled the origins of net unfolding and the various developed methods in recent decades, along with their application to a number of typical problems in discrete event systems (DESs). Net unfolding techniques are risk-free because they often minimize the size of the state space while retaining the capacity to test a wide range of properties, from fundamentals such as deadlocks and reachability to more complex properties such as non-interference. These techniques rely on dependency relations. Unlike the classical reachability graph, the net unfolding approach provides a direct representation of the partial order of firing sequences. However, the construction must be finished before the verification can begin. We have also provided different kinds of condescending representations based on net unfolding (i.e., merged processes and trellis processes) and a wide variety of verification algorithms implemented for different classes of Petri net models. Most of these techniques deal with 1-bounded or k-bounded net systems, opening doors for potential solutions for unbounded nets that are considered practically critical and theoretically significant for industrial systems.

Author Contributions: Conceptualization, Y.R.; methodology, Y.R.; validation, Z.L.; formal analysis, Y.R.; writing—original draft preparation, Y.R.; writing—review and editing, Z.L.; visualization, Y.R.; supervision, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding and was partially supported by the Science and Technology Fund, FDCT, Macau SAR, under Grant No. 0064/2021/A2.

Data Availability Statement: Not applicable.

Acknowledgments: This research was partially supported by the Science and Technology Fund, FDCT, Macau SAR, under Grant No. 0064/2021/A2.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ho, Y.C. Introduction to special issue on dynamics of discrete event systems. *Proc. IEEE* **1989**, *77*, 3–6. [CrossRef]
2. Silva, M. On the history of discrete event systems. *Annu. Rev. Control* **2018**, *45*, 213–222. [CrossRef]
3. Petri, C.A. Kommunikation Mit Automaten. Ph.D. Thesis, University of Bonn, Bonn, Germany, 1962.
4. Rozenberg, G. Behaviour of elementary net systems. In *Petri Nets: Central Models and Their Properties*; Springer: Berlin/Heidelberg, Germany, 1987; pp. 60–94.
5. Esparza, J.; Heljanko, K. *Unfoldings: A Partial-Order Approach to Model Checking*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2008.
6. Godefroid, P.; Wolper, P. Using partial orders for the efficient verification of deadlock freedom and safety properties. In Proceedings of the International Conference on Computer Aided Verification, Aalborg, Denmark, 1–4 July 1991; pp. 332–342.
7. Cabasino, M.P.; Giua, A.; Seatzu, C. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica* **2010**, *46*, 1531–1539. [CrossRef]
8. Tong, Y.; Li, Z.; Seatzu, C.; Giua, A. Verification of state-based opacity using Petri nets. *IEEE Trans. Autom. Control* **2016**, *62*, 2823–2837. [CrossRef]

9. Petri, C.A. Nichtsequentielle Prozesse Arbeitsberichte des IMMD, Bd. 9, Heft. 8, p.57ff. In *Non-Sequential Processes*; Krause, P., Low, J., Translators; Internal Report GMD-ISF-77-05; Universität Erlangen-Nürnberg: Bonn, Germany, 1977.
10. Mazurkiewicz, A. Concurrent Program Schemes and their Interpretations. *DAIMI Rep. Ser.* **1977**, *6*, 1–50. [CrossRef]
11. Winskel, G. Events in Computation. Ph.D. Thesis, University of Edinburgh, Edinburgh, UK, 1980.
12. Valmari, A. Stubborn sets for reduced state space generation. In Proceedings of the Advances in Petri Nets, Bonn, Germany, 13 March 1990 ; Rozenberg, G., Ed.; Springer: Berlin/Heidelberg, Germany, 1991; pp. 491–515.
13. Godefroid, P. *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*; Springer: Berlin/Heidelberg, Germany, 1996.
14. Godefroid, P. Using partial orders to improve automatic verification methods. In Proceedings of the International Conference on Computer Aided Verification, New Brunswick, NJ, USA, 18–21 June 1990; pp. 176–185.
15. Nielsen, M.; Plotkin, G.; Winskel, G. Petri nets, event structures and domains, part I. *Theor. Comput. Sci.* **1981**, *13*, 85–108. [CrossRef]
16. McMillan, K.L. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In Proceedings of the International Conference on Computer Aided Verification, Montreal, QC, Canada, 29 June–1 July 1992; pp. 164–177.
17. Engelfriet, J. Branching processes of Petri nets. *Acta Inform.* **1991**, *28*, 575–591. [CrossRef]
18. Cassandras, C.G.; Lafortune, S. *Introduction to Discrete Event Systems*; Springer: New York, NY, USA, 2008.
19. Li, Z.; Zhou, M. *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*; Springer Science & Business Media: London, UK, 2009.
20. Ramadge, P.J.; Wonham, W.M. The control of discrete event systems. *Proc. IEEE* **1989**, *77*, 81–98. [CrossRef]
21. Grumberg, O.; Veith, H. *25 Years of Model Checking: History, Achievements, Perspectives*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5000.
22. Murata, T. Petri nets: Properties, analysis and applications. *Proc. IEEE* **1989**, *77*, 541–580. [CrossRef]
23. Lamport, L. Time, clocks, and the ordering of events in a distributed system. In *Concurrency: The Works of Leslie Lamport*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 179–196.
24. Mazurkiewicz, A. Trace theory. In Proceedings of the Advanced Course on Petri Nets, Bad Honnef, Germany, 8–19 September 1986; pp. 278–324.
25. Pratt, V. Modeling concurrency with partial orders. *Int. J. Parallel Program.* **1986**, *15*, 33–71. [CrossRef]
26. Winskel, G. Event structures. In Proceedings of the Advanced Course on Petri Nets, Bad Honnef, Germany, 8–19 September 1986; pp. 325–392.
27. Valmari, A. A stubborn attack on state explosion. In Proceedings of the International Conference on Computer Aided Verification, New Brunswick, NJ, USA, 18–21 June 1990; pp. 156–165.
28. Peled, D. Combining partial order reductions with on-the-fly model-checking. In Proceedings of the International Conference on Computer Aided Verification, Stanford, CA, USA, 21–23 June 1994; pp. 377–390.
29. Goltz, U.; Reisig, W. The non-sequential behaviour of Petri nets. *Inf. Control.* **1983**, *57*, 125–147. [CrossRef]
30. Best, E.; Devillers, R. Sequential and concurrent behaviour in Petri net theory. *Theor. Comput. Sci.* **1987**, *55*, 87–136. [CrossRef]
31. Best, E.; Fernandez, C.C. *Nonsequential Processes: A Petri Net View*; EATCS Monographs on Theoretical Computer Science; Springer: Berlin/Heidelberg, Germany, 1988.
32. Petri, C. General net theory. communication disciplines. In Proceedings of the Joint IBM University of Newcastle upon, Tyne Seminar, Newcastle, GB, USA, 6–9 September 1977.
33. Scott, D. *Outline of a Mathematical Theory of Computation*; Oxford University Computing Laboratory, Programming Research Group Oxford: Oxford, UK, 1970.
34. Scott, D. Data types as lattices. *Siam J. Comput.* **1976**, *5*, 522–587. [CrossRef]
35. Stoy, J.E. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*; MIT press: Cambridge, MA, USA, 1981.
36. Nielsen, M.; Rozenberg, G.; Thiagarajan, P.S. Behavioural notions for elementary net systems. *Distrib. Comput.* **1990**, *4*, 45–57. [CrossRef]
37. Nielsen, M.; Rozenberg, G.; Thiagarajan, P. Transition-systems, event structures, and unfoldings. *Inf. Comput.* **1995**, *118*, 191–207. [CrossRef]
38. Winskel, G. An introduction to event structures. In Proceedings of the Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems), Noordwijkerhout, The Netherlands, 30 May–3 June 1988; pp. 364–397.
39. Khomenko, V.; Koutny, M. Branching processes of high-level Petri nets. In Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Warsaw, Poland, 7–11 April 2003; pp. 458–472.
40. Liu, F.; Heiner, M.; Yang, M. An efficient method for unfolding colored Petri nets. In Proceedings of the 2012 Winter Simulation Conference (WSC), Berlin, Germany, 9–12 December 2012; pp. 1–12.
41. Chatain, T.; Jard, C. Symbolic diagnosis of partially observable concurrent systems. In Proceedings of the International Conference on Formal Techniques for Networked and Distributed Systems, Madrid, Spain, 27–30 September 2004; pp. 326–342.
42. Baldan, P.; Corradini, A.; König, B.; Schwoon, S. McMillan's complete prefix for contextual nets. In *Transactions on Petri Nets and Other Models of Concurrency I*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 199–220.

43. McMillan, K.L. Trace theoretic verification of asynchronous circuits using unfoldings. In Proceedings of the International Conference on Computer Aided Verification, Liege, Belgium, 3–5 July 1995; pp. 180–195.
44. Probst, D.K.; Li, H.F. Using partial-order semantics to avoid the state explosion problem in asynchronous systems. In Proceedings of the International Conference on Computer Aided Verification, New Brunswick, NJ, USA, 18–21 June 1990; pp. 146–155.
45. Esparza, J.; Römer, S.; Vogler, W. An improvement of McMillan's unfolding algorithm. In Proceedings of the International Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Passau, Germany, 27–29 March 1996; pp. 87–106.
46. Esparza, J.; Römer, S.; Vogler, W. An improvement of McMillan's unfolding algorithm. *Form. Methods Syst. Des.* **2002**, *20*, 285–310. [CrossRef]
47. Esparza, J.; Römer, S. An unfolding algorithm for synchronous products of transition systems. In Proceedings of the International Conference on Concurrency Theory, Eindhoven, The Netherlands, 24–27 August 1999; pp. 2–20.
48. McMillan, K.L. Symbolic model checking. In *Symbolic Model Checking*; Springer: Dordrecht, The Netherlands, 1993; pp. 25–60.
49. Melzer, S.; Römer, S. Deadlock checking using net unfoldings. In Proceedings of the International Conference on Computer Aided Verification, Haifa, Israel, 22–25 June 1997; pp. 352–363.
50. Heljanko, K. Using logic programs with stable model semantics to solve deadlock and reachability problems for 1-safe Petri nets. *Fundam. Inform.* **1999**, *37*, 247–268. [CrossRef]
51. Khomenko, V.; Koutny, M. LP deadlock checking using partial order dependencies. In Proceedings of the International Conference on Concurrency Theory, University Park, PA, USA, 22–25 August 2000; pp. 410–425.
52. Schröter, C.; Esparza, J. Reachability analysis using net unfoldings. In Proceedings of the International Workshop on Concurrency, Specification and Programming (CS&P 2000), University Park, PA, USA, 22–25 August 2000.
53. Probst, D.K.; Li, H.F. Partial-order model checking: A guide for the perplexed. In Proceedings of the International Conference on Computer Aided Verification, New Brunswick, NJ, USA, 18–21 June 1991; pp. 322–331.
54. Best, E.; Esparza, J. Model checking of persistent Petri nets. In Proceedings of the International Workshop on Computer Science Logic, Berne, Switzerland, 7–11 October 1991; pp. 35–52.
55. Valmari, A. The state explosion problem. In Proceedings of the Advanced Course on Petri Nets, Bad Honnef, Germany, 8–19 September 1996; pp. 429–528.
56. Esparza, J. Model checking using net unfoldings. *Sci. Comput. Program.* **1994**, *23*, 151–195. [CrossRef]
57. Girault, C.; Valk, R. *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
58. Bradfield, J.C. Proving temporal properties of Petri nets. In Proceedings of the International Conference on Application and Theory of Petri Nets, Bonn, Germany, 1 June 1989 ; pp. 29–47.
59. Chatain, T.; Paulevé, L. Goal-driven unfolding of Petri nets. *arXiv* **2016**, arXiv:1611.01296.
60. Abdulla, P.A.; Iyer, S.P.; Nylén, A. Unfoldings of unbounded Petri nets. In Proceedings of the International Conference on Computer Aided Verification, Chicago, IL, USA, 15–19 July 2000; pp. 495–507.
61. He, K.X.; Lemmon, M.D. Liveness verification of discrete event systems modeled by n-safe ordinary Petri nets. In Proceedings of the International Conference on Application and Theory of Petri Nets, Aarhus, Denmark, 26–30 June 2000; pp. 227–243.
62. Xie, X.; Giua, A. Counterexamples to "liveness-enforcing supervision of bounded ordinary Petri nets using partial-order methods". *IEEE Trans. Autom. Control* **2004**, *49*, 1217–1219. [CrossRef]
63. Goguen, J.A.; Meseguer, J. Security policies and security models. In Proceedings of the 1982 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 26–28 April 1982; p. 11.
64. Bryans, J.W.; Koutny, M.; Ryan, P.Y. Modelling opacity using Petri nets. *Electron. Notes Theor. Comput. Sci.* **2005**, *121*, 101–115. [CrossRef]
65. Tong, Y.; Li, Z.; Seatzu, C.; Giua, A. Decidability of opacity verification problems in labeled Petri net systems. *Automatica* **2017**, *80*, 48–53. [CrossRef]
66. Tong, Y.; Li, Z.; Seatzu, C.; Giua, A. Current-state opacity enforcement in discrete event systems under incomparable observations. *Discret. Event Dyn. Syst.* **2018**, *28*, 161–182. [CrossRef]
67. Cong, X.; Fanti, M.P.; Mangini, A.M.; Li, Z. On-line verification of current-state opacity by Petri nets and integer linear programming. *Automatica* **2018**, *94*, 205–213. [CrossRef]
68. Baldan, P.; Carraro, A. Non-interference by unfolding. In Proceedings of the International Conference on Applications and Theory of Petri Nets and Concurrency, Tunis, Tunisia, 23–27 June 2014; pp. 190–209.
69. Baldan, P.; Burato, F.; Carraro, A. Intransitive non-interference by unfolding. In Proceedings of the International Conference on Formal Aspects of Component Software, Bertinoro, Italy, 10–12 September 2014; pp. 269–287.
70. Li, Z.; Zhang, J.; Zhao, M. Liveness-enforcing supervisor design for a class of generalised Petri net models of flexible manufacturing systems. *IET Control. Theory Appl.* **2007**, *1*, 955–967. [CrossRef]
71. Li, Z.; Zhao, M. On controllability of dependent siphons for deadlock prevention in generalized Petri nets. *IEEE Trans. Syst. Man-Cybern. Part Syst. Hum.* **2008**, *38*, 369–384.
72. Wang, A.; Li, Z.; Jia, J.; Zhou, M. An effective algorithm to find elementary siphons in a class of Petri nets. *IEEE Trans. Syst. Man-Cybern. Part Syst. Hum.* **2009**, *39*, 912–923. [CrossRef]
73. Li, Z.; Zhu, S.; Zhou, M. A divide-and-conquer strategy to deadlock prevention in flexible manufacturing systems. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2009**, *39*, 156–169.

74. Chen, Y.; Li, Z. Design of a maximally permissive liveness-enforcing supervisor with a compressed supervisory structure for flexible manufacturing systems. *Automatica* **2011**, *47*, 1028–1034. [CrossRef]
75. Chen, Y.; Li, Z. On structural minimality of optimal supervisors for flexible manufacturing systems. *Automatica* **2012**, *48*, 2647–2656. [CrossRef]
76. Chen, Y.; Li, Z.; Al-Ahmari, A. Nonpure Petri net supervisors for optimal deadlock control of flexible manufacturing systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2012**, *43*, 252–265. [CrossRef]
77. Liu, G.; Li, Z.; Barkaoui, K.; Al-Ahmari, A.M. Robustness of deadlock control for a class of Petri nets with unreliable resources. *Inf. Sci.* **2013**, *235*, 259–279. [CrossRef]
78. Uzam, M.; Li, Z.; Gelen, G.; Zakariyya, R.S. A divide-and-conquer-method for the synthesis of liveness enforcing supervisors for flexible manufacturing systems. *J. Intell. Manuf.* **2016**, *27*, 1111–1129. [CrossRef]
79. He, K.X.; Lemmon, M.D. Liveness-enforcing supervision of bounded ordinary Petri nets using partial order methods. *IEEE Trans. Autom. Control.* **2002**, *47*, 1042–1055. [CrossRef]
80. Giua, A.; Xie, X. Control of safe ordinary Petri nets with marking specifications using unfolding. *IFAC Proc. Vol.* **2004**, *37*, 63–68. [CrossRef]
81. Giua, A.; Xie, X. Control of safe ordinary Petri nets using unfolding. *Discret. Event Dyn. Syst.* **2005**, *15*, 349–373. [CrossRef]
82. Buy, U.; Darabi, H.; Lehene, M.; Venepally, V. Supervisory control of time Petri nets using net unfolding. In Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05), Scotland, UK, 25–28 July 2005; Volume 2, pp. 97–100.
83. Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; Teneketzis, D. Diagnosability of discrete-event systems. *IEEE Trans. Autom. Control* **1995**, *40*, 1555–1575. [CrossRef]
84. Cong, X.; Fanti, M.P.; Mangini, A.M.; Li, Z. Decentralized diagnosis by Petri nets and integer linear programming. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *48*, 1689–1700. [CrossRef]
85. Tong, Y.; Li, Z.; Giua, A. On the equivalence of observation structures for Petri net generators. *IEEE Trans. Autom. Control* **2015**, *61*, 2448–2462. [CrossRef]
86. Zhu, G.; Li, Z.; Wu, N. Model-based fault identification of discrete event systems using partially observed Petri nets. *Automatica* **2018**, *96*, 201–212. [CrossRef]
87. Ma, Z.; Li, Z.; Giua, A. Characterization of admissible marking sets in Petri nets with conflicts and synchronizations. *IEEE Trans. Autom. Control* **2016**, *62*, 1329–1341. [CrossRef]
88. Zhu, G.; Li, Z.; Wu, N.; Al-Ahmari, A. Fault identification of discrete event systems modeled by Petri nets with unobservable transitions. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *49*, 333–345. [CrossRef]
89. Benveniste, A.; Fabre, E.; Haar, S.; Jard, C. Diagnosis of asynchronous discrete-event systems: A net unfolding approach. *IEEE Trans. Autom. Control* **2003**, *48*, 714–727. [CrossRef]
90. Fabre, E.; Benveniste, A.; Haar, S.; Jard, C. Distributed monitoring of concurrent and asynchronous systems. *Discret. Event Dyn. Syst.* **2005**, *15*, 33–84. [CrossRef]
91. Fabre, E. Trellis processes: A compact representation for runs of concurrent systems. *Discret. Event Dyn. Syst.* **2007**, *17*, 267–306. [CrossRef]
92. Fabre, E.; Benveniste, A. Partial order techniques for distributed discrete event systems: Why you cannot avoid using them. *Discret. Event Dyn. Syst.* **2007**, *17*, 355–403. [CrossRef]
93. Haar, S. Unfold and cover: Qualitative diagnosability for Petri nets. In Proceedings of the 2007 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 12–14 December 2007; pp. 1886–1891.
94. Haar, S. Types of asynchronous diagnosability and the reveals-relation in occurrence nets. *IEEE Trans. Autom. Control* **2010**, *55*, 2310–2320. [CrossRef]
95. Haar, S.; Rodríguez, C.; Schwoon, S. Reveal your faults: It's only fair! In Proceedings of the 2013 13th International Conference on Application of Concurrency to System Design, Barcelona, Spain, 8–10 July 2013; pp. 120–129.
96. Hélouët, L.; Marchand, H. On the cost of diagnosis with disambiguation. In Proceedings of the International Conference on Quantitative Evaluation of Systems, Berlin, Germany, 5–7 September 2017; pp. 140–156.
97. Khomenko, V.; Koutny, M. Towards an efficient algorithm for unfolding Petri nets. In Proceedings of the International Conference on Concurrency Theory, Aalborg, Denmark, 20–25 August 2001; pp. 366–380.
98. Khomenko, V.; Koutny, M.; Vogler, W. Canonical prefixes of Petri net unfoldings. *Acta Inform.* **2003**, *40*, 95–118. [CrossRef]
99. Khomenko, V. Model Checking Based on Prefixes of Petri Net Unfoldings. Ph.D. Thesis, Newcastle University, Newcastle upon Tyne, UK, 2003.
100. Khomenko, V.; Kondratyev, A.; Koutny, M.; Vogler, W. Merged processes: A new condensed representation of Petri net behaviour. *Acta Inform.* **2006**, *43*, 307–330. [CrossRef]
101. Schwoon, S.; Rodríguez, C. Construction and SAT-based verification of contextual unfoldings. In Proceedings of the International Workshop on Descriptional Complexity of Formal Systems, Giessen/Limburg, Germany, 25–27 July 2011; pp. 34–42.
102. Hayman, J.; Winskel, G. The unfolding of general Petri nets. In Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, Bangalore, India, 9–11 December 2008.
103. Benito, F.C.V.V.; Kunzle, L.A. Relaxed unfolding for time Petri nets. In Proceedings of the 2013 International Conference on Computer Sciences and Applications, Washington, DC, USA, 14–15 December 2013; pp. 833–839.

104. Frumin, D.; Lomazova, I.A. Branching processes of conservative nested Petri nets. *VPT@ CAV* **2014**, *19*, 35.
105. Jezequel, L.; Chatain, T.; Comlan, M.; Delfieu, D.; Roux, O.H. Pomsets and unfolding of reset Petri nets. In Proceedings of the LATA 2018-12th International Conference on Language and Automata Theory and Applications, Ramat Gan, Israel, 9–11 April 2018.