DOI 10.1007/s00165-014-0329-4
BCS © 2015
Formal Aspects of Computing (2015) 27: 753–788

Formal Aspects of Computing

Untanglings: a novel approach to analyzing concurrent systems

Artem Polyvyanyy[1], Marcello La Rosa[1,2], Chun Ouyang[1], Arthur H. M. ter Hofstede[1,3]

[1] Queensland University of Technology, P Block (Level 8), Gardens Point Campus, GPO Box 2434, Brisbane, QLD 4001, Australia
[2] NICTA Queensland Lab, Brisbane, Australia
[3] Eindhoven University of Technology, Eindhoven, The Netherlands

Abstract. Substantial research efforts have been expended to deal with the complexity of concurrent systems that is inherent to their analysis, e.g., works that tackle the well-known state space explosion problem. Approaches differ in the classes of properties that they are able to suitably check and this is largely a result of the way they balance the trade-off between analysis time and space employed to describe a concurrent system. One interesting class of properties is concerned with behavioral characteristics. These properties are conveniently expressed in terms of computations, or *runs*, in concurrent systems. This article introduces the theory of *untanglings* that exploits a particular representation of a collection of runs in a concurrent system. It is shown that a *representative* untangling of a bounded concurrent system can be constructed that captures all and only the behavior of the system. Representative untanglings strike a unique balance between time and space, yet provide a single model for the convenient extraction of various behavioral properties. Performance measurements in terms of construction time and size of representative untanglings with respect to the original specifications of concurrent systems, conducted on a collection of models from practice, confirm the scalability of the approach. Finally, this article demonstrates practical benefits of using representative untanglings when checking various behavioral properties of concurrent systems.

Keywords: Concurrency, Concurrent systems, Analysis, Untanglings, Representative untanglings

1. Introduction

Concurrent systems are systems in which multiple computations, or *runs*, are executed simultaneously by independent processing units, which usually interact with each other, in order to achieve a common goal. Naturally, concurrent systems can provide a better 'throughput', i.e., produce more computed output relative to the number and size of given tasks over a certain period of time, than systems in which all operations are performed in a centralized manner by a single processing unit. The design of a concurrent system, thus, deals with finding a suitable coordination of operations that encourages efficiency of individual computations.

Correspondence and offprint requests to: A. Polyvyanyy, E-mail: artem.polyvyanyy@qut.edu.au

Concurrent systems are complex to analyze. The complexity is primarily due to the amount of uncertainty about the order in which atomic operations can be executed by different processing units of a system. When studying this fact in detail, one can immediately identify the following three reasons that are inherent to the complexity of concurrent systems:

- Firstly, the number of runs described in a concurrent system is exponential in the number of choices that one can make when carving a way through the system into a desired computation.
- Secondly, the number of all possible interleavings of operations described in a concurrent system grows at a factorial rate with respect to the number of simultaneously enabled operations—a phenomenon which leads to the well-known state space explosion problem.
- Thirdly, a concurrent system can encode an infinite number of runs in the case of loops.

Consequently, approaches to analyzing concurrent systems constitute one of the most complex classes of practical problems in computer science. Most of the techniques for verification of behavioral characteristics of concurrent systems, i.e., properties over potential runs of concurrent systems, are doomed to explore immense portions of operation interleavings that are encoded in models of concurrent systems. Besides, the complexity of these explorations depends on the particular formalism that is used to describe systems.

A number of formalisms for modeling and analyzing concurrent systems have been developed and systematized [SNW96, LSV98]. Some of them can be used through the entire development cycle while others target some particular phase, e.g., analysis. One of the first formalisms for describing concurrent systems, and by now one of the most studied, was proposed in Carl Adam Petri's seminal work on Petri nets [Pet77]. A concurrent system captured in the Petri net notation, or a *net system*, is often reminiscent of a mass of highly interwoven models of individual runs—a *tangle* of computations. Every time a Petri net gets instantiated it 'works' its way through a maze of pre-designed options into a collection of desired computations.

This article proposes a theory that leads to a novel characterization of runs encoded in concurrent systems, which is specifically suitable for analysis purposes. Intuitively, we propose to 'untie' Petri net models of concurrent systems into their *untanglings*—collections of abstract models capturing different portions of the behavior of the original system. Precisely, an *untangling* of a concurrent system is a set of its *processes* [Pet77, NPW81, GR83], i.e., conflict-free Petri nets, such that each process represents a (possibly infinite) subset of runs of the original system. A *representative untangling* of a net system describes the *exact* behavior of the net system, is usually larger than the net system (with respect to model size), but is easier to analyze. Representative untanglings provide a unique balance between the time required for extraction of various behavioral properties and the space required to describe concurrent systems.

This article discusses a baseline algorithm for building representative untanglings of Petri net models of concurrent systems and presents results on its termination and correctness analysis. The baseline algorithm requires an input net system to be *bounded*, i.e., to have a finite number of reachable states. No restrictions are put on the Petri net class; we do not require that net systems are captured as workflow nets [vdA97] or free-choice nets [DE95]. Furthermore, the article presents an improvement of the baseline algorithm that can be used to efficiently construct representative untanglings with small space footprints. This improvement is designed as a framework consisting of one or more structural transformations of input net systems and untanglings produced by the baseline algorithm. In particular, we discuss two reduction rules, inspired by Murata's rules [Mur89], that fit the framework and demonstrate that the properties of representative untanglings are all preserved by the improved algorithm.

To prove the feasibility of the proposed approach, we implemented the improved untangling algorithm and measured its performance in terms of time and space using a large collection of net systems from practice. The results confirm that representative untanglings can be constructed efficiently (in the order of a few milliseconds) and that the duplication factor w.r.t. the original net system is negligible (a representative untangling is on average only 3.5 times larger than the original system). Moreover, we demonstrate the practical advantages of representative untanglings over other formalisms for describing concurrent systems when computing various behavioral properties, e.g., executability, deadlock freedom, and mutual exclusiveness.

The remainder of this article is organized as follows: The next section introduces preliminary notions that will be used to impart the findings. Section 3 proposes process set systems—abstract models capable of representing chunks of behaviors of concurrent systems. Section 4 is devoted to the main artifact which is developed in this article—a representative untangling of a concurrent system; it proposes the baseline algorithm for constructing representative untanglings. Section 5 discusses the transformation-based optimization framework and uses this framework to propose an improved untangling algorithm. The article closes with a discussion of related work, a demonstration of different applications of representative untanglings to analyzing concurrent systems, an evaluation on a collection of concurrent systems from practice, and conclusions.

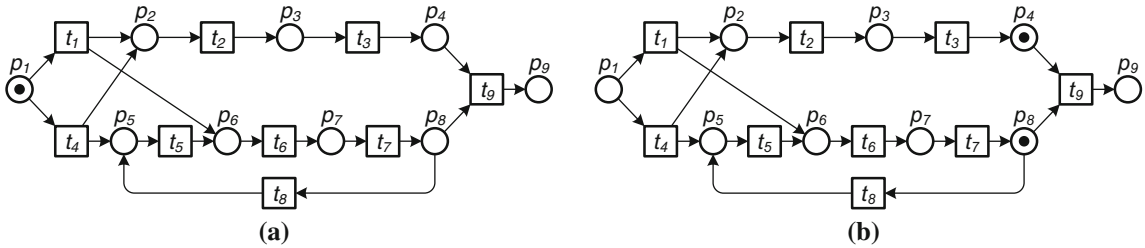

Fig. 1. Two net systems (the same net at two different markings)

2. Preliminaries

This section introduces formalisms that will be used to support discussions in the subsequent sections. Section 2.1 presents *Petri nets* and *net systems*—well-established formalisms for describing concurrent systems. Section 2.2 talks about *processes* of net systems—abstract models that are capable of describing the non-sequential behavior of concurrent systems.

2.1. Petri nets and net systems

Petri nets are a well-known formalism for modeling concurrent systems. This section introduces the basic Petri net terminology and notations.

Definition 2.1 (*Petri net*) A *Petri net*, or a *net*, is a triple $N := (P, T, F)$, where P and T are finite disjoint sets of *places* and *transitions*, respectively, and $F \subseteq (P \times T) \cup (T \times P)$ is the *flow* relation. ⌟

An element $x \in P \cup T$ is a *node* of N. A node $x \in P \cup T$ is an *input* node of a node $y \in P \cup T$ iff x and y are in the flow relation, i.e., $(x, y) \in F$. Similarly, a node $x \in P \cup T$ is an *output* node of a node $y \in P \cup T$ iff $(y, x) \in F$. By $\bullet x, x \in P \cup T$, we denote the *preset* of x, i.e., the set of all input nodes of x, $\bullet x := \{y \in P \cup T \mid (y, x) \in F\}$. Likewise, by $x\bullet$, $x \in P \cup T$, we refer to the *postset* of x, i.e., the set of all output nodes of x, $x\bullet := \{y \in P \cup T \mid (x, y) \in F\}$. For a set of nodes $X \subseteq P \cup T$, $\bullet X := \bigcup_{x \in X} \bullet x$ and $X\bullet := \bigcup_{x \in X} x\bullet$. For technical convenience, we require all nets to be T-restricted. A net N is *T-restricted* iff the preset and postset of every transition is non-empty, i.e., $\forall\, t \in T : \bullet t \neq \varnothing \neq t\bullet$.

A node $x \in P \cup T$ is a *source* node of N iff x has no input nodes, i.e., $\bullet x = \varnothing$. A node $x \in P \cup T$ is a *sink* node of N iff x has no output nodes, i.e., $x\bullet = \varnothing$. Given a net $N := (P, T, F)$, by $Min(N)$ we denote the set of all source nodes of N, i.e., $Min(N) := \{x \in P \cup T \mid \bullet x = \varnothing\}$, whereas by $Max(N)$ we denote the set of all sink nodes of N, i.e., $Max(N) := \{x \in P \cup T \mid x\bullet = \varnothing\}$. Observe that source and sink nodes of a T-restricted net are places.

The execution semantics of a Petri net is based on the notion of a *marking*. A *marking* of a Petri net is a distribution of *tokens* over the net's places. Let $\mathbb{K}$ be a universe of tokens.

Definition 2.2 (*Marking of a net*) A *marking* of a net $N := (P, T, F)$ is a pair $M := (K, \mu)$, where $K \subseteq \mathbb{K}$ is a set of *tokens* and $\mu : K \to P$ is a function that assigns each token to its place. ⌟

Finally, a net system is a net at a certain marking.

Definition 2.3 (*Net system*)
A *net system*, or a *system*, is a pair $S := (N, M)$, where N is a net and M is a marking of N. ⌟

In the graphical notation, a common practice is to visualize places as circles, transitions as rectangles, the flow relation as directed edges, and tokens as black dots inside the assigned places; see Fig. 1 for examples of visualizations of net systems. Both net systems in Fig. 1 are defined by the same net at different markings. The net system (N, M), $M := (K, \mu)$, in Fig. 1a is at the marking that puts one token at place p_1 and no tokens elsewhere, i.e., $\mu(p_1) = 1$ and $\mu(p_i) = 0$, $i \in [2\,..\,9]$, whereas the net system in Fig. 1b is at the marking with one token at place p_4, one token at place p_8, and no tokens elsewhere.

Whether a transition is *enabled* depends on tokens in its input places. An enabled transition can *occur*, which leads to a fresh marking.

Definition 2.4 (*Semantics of a net system*)[1]
Let $S := (N, M)$, $N := (P, T, F)$, $M := (K, \mu)$, be a net system.

- A transition $t \in T$ is *enabled* in S, denoted by $S[t\rangle$, iff every input place of t contains at least one token, i.e., $\forall\, p \in \bullet t\ \exists\ k \in K : \mu(k) = p$.
- If a transition $t \in T$ is enabled in S then t can *occur*, which leads to a net system $S' := (N, M')$, where $M' := (K', \mu')$ is a marking obtained by 'consuming' one token from every input place of t and 'producing' one fresh token at every output place of t as follows:
  Let $\hat{K} \subseteq \mathbb{K} \backslash K$ be a set of tokens such that $|\ \hat{K}\ | = |\ t\bullet\ |$ and let $g : \hat{K} \to t\bullet$ be a bijection between $\hat{K}$ and $t\bullet$. Let $f \subseteq \mu \rhd \bullet t$, with f a bijection and $|\ dom(f)\ | = |\ \bullet t\ |$. Then, $K' := K \vartriangle (dom(f) \cup dom(g))$ and $\mu' := \mu \vartriangle (f \cup g)$.[2] ⌟

The net system in Fig. 1a enables transitions t_1 and t_4, whereas the net system in Fig. 1b enables transitions t_8 and t_9. An occurrence of transition t_1 in Fig. 1a leads to a system with one token at place p_2, one token at p_6, and no tokens at other places. In turn, an occurrence of transition t_4 in Fig. 1a leads to a system with one token at place p_2, one token at place p_5, and no tokens elsewhere. By $S[t\rangle S'$, we denote the fact that there exists an occurrence of transition t that leads from S to S'. Observe that the execution semantics of net systems relies on holdings of tokens at places and is independent of the tokens' identities. This fact gives rise to the next relation on markings. Two markings $M_1 := (K_1, \mu_1)$ and $M_2 := (K_2, \mu_2)$ of a net are *equivalent*, denoted by $M_1 \equiv M_2$, if and only if there exists a bijection $\beta : K_1 \to K_2$ such that $\mu_1(k) = (\mu_2 \circ \beta)(k)$, $k \in K_1$. Clearly, the '$\equiv$' relation is an equivalence relation. Every equivalence class of this relation specifies a *state* of the net (system), which is often identified as a multiset of the net's places.

Definition 2.5 (*State of a net*) The *state of a net* $N := (P, T, F)$ *induced by a marking* $M := (K, \mu)$, $\mu : K \to P$, is a multiset H of places of N, where every place $p \in P$ appears $|\ \{k \in K \mid \mu(k) = p\}\ |$ times in H. ⌟

The state of a net system $S := (N, M)$ is the state of N induced by M. We shall write $[p_1\ p_1\ p_2]$ to denote the state induced by a marking that puts two tokens at place p_1, one token at place p_2, and no tokens elsewhere. Hence, the net system in Fig. 1a is at the state $[p_1]$, whereas the net system in Fig. 1b is at $[p_4\ p_8]$.

Let M_1 and M_2 be two markings of a net and let H_1 and H_2 be two states of the net induced by M_1 and M_2, respectively. It is easy to see that $H_1 = H_2$ iff $M_1 \equiv M_2$. Let M_1 and M_2 be equivalent and let $S_1 := (N, M_1)$ and $S_2 := (N, M_2)$ be two net systems, where $N := (P, T, F)$. Then, it holds that $(N, M_1)[t\rangle$ iff $(N, M_2)[t\rangle$, $t \in T$. Finally, let (N, M_1') and (N, M_2') be two net systems obtained after an occurrence of transition t in S_1 and S_2, respectively. Clearly, it holds that M_1' is equivalent to M_2'. The above observations lead to the notion of a *step* that describes 'equivalent' transition occurrences. A triple $\chi := (H_1, t, H_2)$ is a *step* in a net $N := (P, T, F)$, iff it holds that $(N, M_1)[t\rangle(N, M_2)$, where H_1 and H_2 are the states of net systems (N, M_1) and (N, M_2), respectively, and $t \in T$.

A net system induces a set of its reachable states/markings/systems.

Definition 2.6 (*Run*, *Reachable state*, *Occurrence sequence*)
Let $S_0 := (N, M_0)$, $N := (P, T, F)$, be a net system.

- A sequence of steps $\delta := (H_0, t_1, H_1)(H_1, t_2, H_2)\ldots$, either finite or infinite, in N is a *run* in S_0, iff $\delta = \langle\rangle$, i.e., δ is the empty sequence, or H_0 is the state of S_0.
- A state H of N is *reachable* from S_0, denoted by $H \in [S_0\rangle$, iff H is induced by M_0 or there exists a run $(H_0, t_1, H_1)(H_1, t_2, H_2)\ldots(H_{n-1}, t_n, H_n)$, $n \in \mathbb{N}_0$, in S_0 such that $H = H_n$. A marking M of N is *reachable* from S_0 iff there exists a state $H \in [S_0\rangle$ such that H is induced by M. A net system (N, M) is *reachable* from S_0 iff M is reachable from S_0. Similarly, we shall often refer to a state, marking, and net system that is reachable from a net system S as the *reachable* state, marking, and net system of S, respectively.
- A sequence of transitions $\sigma := t_1 t_2 \ldots$, either finite or infinite, of N is an *occurrence sequence* in S_0, iff σ is empty or there exists a run $(H_0, t_1, H_1)(H_1, t_2, H_2)\ldots$ in S_0. ⌟

Let $S := (N, M)$ be a net system. A step (H, t, H') in N is a step in S iff $H \in [S\rangle$.

In the sequel, we shall often write $p_1\,[t_1\rangle\ p_2\ p_6$ to denote the step $([p_1], t_1, [p_2\ p_6])$, or $p_1\,[t_1\rangle\ p_2\ p_6\,[t_2\rangle\ p_3\ p_6$ to refer to the sequence of steps $([p_1], t_1, [p_2\ p_6])([p_2\ p_6], t_2, [p_3\ p_6])$. For example, the net system in Fig. 1b is reachable from the net system in Fig. 1a, among others, via these runs:

[1] While unusual, the use of token identities in this definition allows substantial simplification of formalisms in subsequent sections.

[2] $f \rhd X$ denotes the range restriction of function $f : Y \to Z$ to the subset of its codomain $X \subseteq Z$, i.e., $f \rhd X := \{(y, z) \in f \mid z \in X\}$. $dom(f)$ denotes the domain of function f. Finally, $X \vartriangle Y$ denotes the symmetric difference of sets X and Y, i.e., the union of the two sets minus their intersection: $X \vartriangle Y := (X \cup Y) \backslash (X \cap Y)$.

- $p_1\,[t_1\rangle\,p_2\,p_6\,[t_2\rangle\,p_3\,p_6\,[t_6\rangle\,p_3\,p_7\,[t_7\rangle\,p_3\,p_8\,[t_3\rangle\,p_4\,p_8$,
- $p_1\,[t_1\rangle\,p_2\,p_6\,[t_2\rangle\,p_3\,p_6\,[t_3\rangle\,p_4\,p_6\,[t_6\rangle\,p_4\,p_7\,[t_7\rangle\,p_4\,p_8$,
- $p_1\,[t_4\rangle\,p_2\,p_5\,[t_5\rangle\,p_2\,p_6\,[t_6\rangle\,p_2\,p_7\,[t_7\rangle\,p_2\,p_8\,[t_2\rangle\,p_3\,p_8\,[t_3\rangle\,p_4\,p_8$, and
- $p_1\,[t_4\rangle\,p_2\,p_5\,[t_2\rangle\,p_3\,p_5\,[t_5\rangle\,p_3\,p_6\,[t_6\rangle\,p_3\,p_7\,[t_7\rangle\,p_3\,p_8\,[t_8\rangle\,p_3\,p_5\,[t_5\rangle\,p_3\,p_6\,[t_6\rangle\,p_3\,p_7\,[t_3\rangle\,p_4\,p_7\,[t_7\rangle\,p_4\,p_8$.

A net system $S := (N, M)$ is *k-bounded*, or *bounded*, iff there exists a number $k \in \mathbb{N}_0$ such that for every marking M reachable from S it holds that the amount of tokens at each place of N is at most k, which implies that the set $[S\rangle$ is finite. Observe that both net systems in Fig. 1 are bounded.

This concludes the section. For more information on Petri nets and systems, the reader can refer to [Mur89].

2.2. Processes of net systems

A common way to trace dependencies between transition occurrences in net systems is by means of runs, cf. Definition 2.6. Runs suit well when it comes to describing *orderings* of transition occurrences. Let $\delta := \chi_1 \dots \chi_n$, $n \in \mathbb{N}_0$, be a run in a net system $S := (N, M_0)$, $N := (P, T, F)$, let H_0 be the state of S, and let $H_1 \dots H_n$ be states of N such that for every position i in δ there exists $t_i \in T$ for which it holds that $\chi_i = (H_{i-1}, t_i, H_i)$. Then, every two subsequent steps χ_{i-1} and χ_i, $i \in [2\,..\,n]$, represent two subsequent transition occurrences, such that an occurrence of t_{i-1} leads to a marking that induces H_{i-1} and an occurrence of t_i takes place at a marking that induces H_{i-1}. However, runs of net systems cannot be used to capture such behavioral phenomena as *causality* and *concurrency*.

In this section, we discuss *processes* of net systems [Pet77, NPW81, GR83]. One can rely on processes to adequately represent causality and concurrency relations on transition occurrences. A process of a net system is a net of a particular kind, called a *causal net* (or sometimes an *occurrence net*, see for instance [GR83]), together with a mapping from nodes of the causal net to nodes of the net system. The mapping allows interpreting the causal net as a *concurrent run* of the net system.

Definition 2.7 (*Causal net*) A net $N := (B, E, G)$, where $B \subseteq \mathbb{K}$, is a *causal net*,[3] iff:

- for every $b \in B$ it holds that $|\bullet b\ | \leq 1$ and $|\ b\bullet| \leq 1$, i.e., N is conflict-free, and
- G^+ is irreflexive, i.e., N is acyclic.[4]

⌟

Given a causal net $N := (B, E, G)$, elements of B and E are called *conditions* and *events*, respectively.

One can use events of causal nets to represent transition occurrences. Consequently, conditions in the preset and postset of an event must be interpreted as tokens that get consumed and produced, respectively, by an occurrence of the transition that corresponds to the event. Thus, we suggest that conditions of causal nets are always drawn from the universe of tokens $\mathbb{K}$. The above discussed intuition can be formalized in the notion of a *process* of a net system as follows.

Definition 2.8 (*Process of a net system*) A *process* of a net system $S := (N, M_0)$, $N := (P, T, F)$, is a pair $\pi := (N_\pi, \rho)$, where $N_\pi := (B, E, G)$ is a causal net and $\rho : B \cup E \to P \cup T$ is such that:

- $\rho(B) \subseteq P,\ \rho(E) \subseteq T$, i.e., ρ preserves the nature of nodes,
- $M_0 \equiv (Min(N_\pi), \rho\,|_{Min(N_\pi)})$, i.e., π starts at M_0, and
- for every event $e \in E$ and for every place $p \in P$ it holds that $|\ \{(p, t) \in F \mid t = \rho(e)\}\ | = |\ \rho^{-1}(p) \cap \bullet e\ |$ and $|\ \{(t, p) \in F \mid t = \rho(e)\}\ | = |\ \rho^{-1}(p) \cap e\bullet\ |$, i.e., ρ respects the environment of transitions.

⌟

Given a process $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, an event $e \in E$ represents an occurrence of transition $\rho(e)$, whereas conditions in $\bullet e$ and $e\bullet$ relate to tokens that get consumed and produced by the occurrence of $\rho(e)$. More precisely, for every condition $c \in \bullet e$ one token must be removed from place $\rho(c)$ and for every condition $c \in e\bullet$ one token must be put at place $\rho(c)$ during an occurrence of $\rho(e)$.

Figure 2 shows three processes of the net system in Fig. 1a.When visualizing processes, we employ elements $e_i\ e_i'\ e_i'' \dots$ to denote events that refer to transition t_i in the corresponding net system. Similarly, conditions $c_i\ c_i'\ c_i'' \dots$ are used to refer to place p_i.

[3] For technical convenience, we require that every element of B is an element of $\mathbb{K}$.

[4] R^+ denotes the transitive closure of a binary relation R.

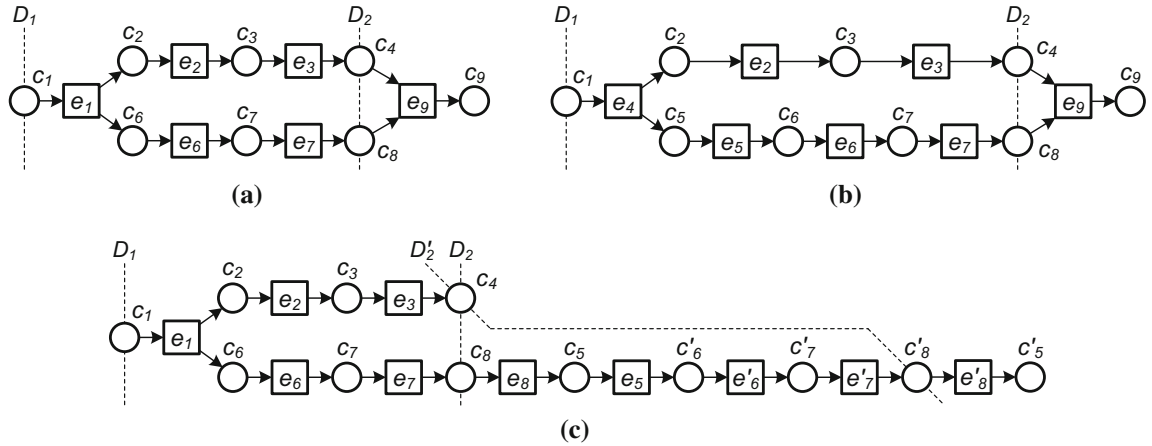

Fig. 2. Three processes of the net system in Fig. 1a

For example, events e_6 and e'_6 in Fig. 2c refer to transition t_6 of the net in Fig. 1a, i.e., $\rho(e_6) = t_6 = \rho(e'_6)$, whereas conditions c_6 and c'_6 in Fig. 2c refer to place p_6, i.e., $\rho(c_6) = p_6 = \rho(c'_6)$.When visualizing several processes we often use identical element names, e.g., event e_1 in the processes in Fig. 2a, c. Though it is not required, in what follows we shall always assume, for technical convenience, that elements with identical names coming from different processes are distinct. Thus, event e_1 of the process in Fig. 2a and event e_1 of the process in Fig. 2c are two different events.

Processes capture dependencies between transition occurrences as follows. Two nodes x and y of a causal net $N := (B, E, G)$ are *causal*, or x is a *cause* for y, iff $(x, y) \in G^+$, whereas x and y are *concurrent* iff $(x, y) \notin G^+$ and $(y, x) \notin G^+$. Intuitively, two distinct events e_1 and e_2 of a causal net of a process (N_π, ρ) of a net system S being concurrent hints at the fact that starting from S one can reach a net system S' in which transitions $\rho(e_1)$ and $\rho(e_2)$ are enabled. Moreover, an occurrence of one of the two transitions in S' leads to a net system in which the other transition is enabled. In contrast, the fact that event e_1 is a cause for event e_2 in N_π indicates the presence of an order, i.e., starting from S one can reach a net system S' which enables transition $\rho(e_1)$ such that an occurrence of $\rho(e_1)$ in S' leads to a net system from which one can reach a net system that enables transition $\rho(e_2)$. For instance, in Fig. 2a, event e_1 is a cause for event e_9, which indicates that an occurrence of transition t_1 can act as a prerequisite for an occurrence of transition t_9 in the net system in Fig. 1a. On the other hand, events e_2 and e_7 are concurrent, i.e., transitions t_2 and t_7 can be enabled at the same time in some net system reachable from the one shown in Fig. 1a.

In addition to transition occurrences, processes encode reachable markings of corresponding net systems. This is accomplished by means of cuts of causal nets. A *cut* of a causal net is a maximal (with respect to set inclusion) set of its pairwise concurrent conditions.

Theorem 2.9 (Cuts and reachable markings, [GR83, Theorem 3.5]) *Let $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, be a process of a net system S. If $C \subseteq B$ is a cut of N_π, then $M := (C, \rho\,|_C)$ is a reachable marking of S.* ⌟

Theorem 2.9 hints at the fact that processes of net systems can be employed as space-efficient data structures for storing reachable markings of corresponding net systems. Figure 2a shows two cuts: D_1 and D_2. Each of these cuts is composed of conditions that intersect the respective dashed line. Hence, cut D_1 is equal to the singleton $\{c_1\}$ and cut D_2 is equal to the set of conditions $\{c_4, c_8\}$. Cuts D_1 and D_2 in Fig. 2a induce markings that are equivalent to markings in Fig. 1a and Fig. 1b, respectively; these are markings $M_1 := (D_1, \rho\,|_{D_1})$ and $M_2 := (D_2, \rho\,|_{D_2})$ (note that D_1 and D_2 are sets of tokens from $\mathbb{K}$). Both M_1 and M_2 are indeed reachable markings of the net system in Fig. 1a. Similarly, cuts D_1 and D_2 in Fig. 2b induce markings that are equivalent to those in Fig. 1a and Fig. 1b, respectively. Observe that the net in Fig. 2c has two cuts that induce markings which are equivalent to the marking in Fig. 1b; these are cuts $D_2 = \{c_4, c_8\}$ and $D'_2 = \{c_4, c'_8\}$. Again, similar to events and conditions of causal nets, in the rest of the article, two cuts with the same name coming from two different processes will be considered to be different. If there is a potential for confusion, we shall explicitly mention the net that possesses the cut.

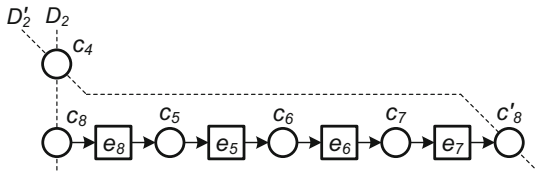

Fig. 3. A reproduction process of the net system in Fig. 1b

Let $\pi := (N_\pi, \rho)$ be a process of a net system $S := (N, M)$. It is known that $Min(N_\pi)$ and $Max(N_\pi)$ are cuts [GR83]. Moreover, by Definition 2.8, $Min(N_\pi)$ is a cut that induces a marking that is equivalent to M. For example, cut D_1 in the process (N_π, ρ) in Fig. 2a is equal to $Min(N_\pi)$ and induces the marking that is equivalent to the marking of the net system in Fig. 1a, i.e., $(D_1, \rho\,|_{D_1})$ is equivalent to the marking with one token at place p_1 and no tokens elsewhere, which is depicted in Fig. 1a.

This concludes the section. For more information on processes of net systems, the reader can refer to [GR83].

3. Process (set) systems

A process of a net system is often interpreted as a model that describes a *finite* portion of the corresponding net system's behavior. In [Des00], Jörg Desel suggests to enhance a causal net N_π of a process $\pi := (N_\pi, \rho)$ of a net system $S := (N, M)$ with a marking M_π that puts one token at every source place of N_π and no tokens elsewhere. Then, every run in the net system (N_π, M_π) 'represents' (via mapping ρ) a run in S. For example, consider the net system S_π composed of the net in Fig. 2a and the marking that puts one token at condition c_1 and no tokens elsewhere.[5] Then, the run $c_1\,[e_1\rangle\,c_2\,c_6\,[e_2\rangle\,c_3\,c_6\,[e_6\rangle\,c_3\,c_7\,[e_7\rangle\,c_3\,c_8\,[e_3\rangle\,c_4\,c_8\,[e_9\rangle\,c_9$ in S_π represents the run $\rho(c_1)[\rho(e_1)\rangle\rho(c_2)\rho(c_6)[\rho(e_2)\rangle\rho(c_3)\rho(c_6)[\rho(e_6)\rangle\rho(c_3)\rho(c_7)[\rho(e_7)\rangle\rho(c_3)\rho(c_8)[\rho(e_3)\rangle\rho(c_4)\rho(c_8)[\rho(e_9)\rangle\rho(c_9) = p_1\,[t_1\rangle\,p_2\,p_6\,[t_2\rangle\,p_3\,p_6\,[t_6\rangle\,p_3\,p_7\,[t_7\rangle\,p_3\,p_8\,[t_3\rangle\,p_4\,p_8\,[t_9\rangle\,p_9$ in the net system in Fig. 1a. In this way, S_π represents 26 different runs in the net system in Fig. 1a. Observe that it is not required that every run leads to the state $[p_9]$, e.g., the process in Fig. 2a represents, among others, the empty run and run $p_1\,[t_1\rangle\,p_2\,p_6$.

The simple structure of processes (processes are static models captured as causal nets) permits simple analysis. For example, processes allow precise reasoning about *causality* and *concurrency* of transition occurrences. However, this simple analysis comes at a price, as time is traded for space. A net system can often have an infinite number of processes making any type of analysis on the set of all processes an infeasible task, e.g., both net systems in Fig. 1 induce infinitely many processes. This discussion triggers a question: Is it possible to represent the behavior of a net system with a finite number of processes? Clearly, one can give a positive answer to this question only if the space-efficiency of each individual process is 'high', i.e., it should be possible to interpret a process in such a way that it captures an infinite portion of the net system's behavior. The initial insights into the feasibility of such an interpretation of a process come from the notion of a *reproduction process* [BD90]. A reproduction process (N_π, ρ) captures repetitive behavior as its minimal and maximal cuts, i.e., $Min(N_\pi)$ and $Max(N_\pi)$, induce equivalent markings.

Figure 3 shows a reproduction process $\pi := (N_\pi, \rho)$ of the net system in Fig. 1b. Indeed, cuts $D_2 := Min(N_\pi)$ and $D_2' := Max(N_\pi)$ induce equivalent markings, i.e., $(D_2, \rho\,|_{D_2}) \equiv (D_2', \rho\,|_{D_2'})$. By following the ideas from [Des00], the process in Fig. 3 enhanced with a marking that puts one token at each of its source conditions can be used to represent five runs in the net system in Fig. 1b: the empty run, $p_4\,p_8\,[t_8\rangle\,p_4\,p_5$, $p_4\,p_8\,[t_8\rangle\,p_4\,p_5\,[t_5\rangle\,p_4\,p_6$, $p_4\,p_8\,[t_8\rangle\,p_4\,p_5\,[t_5\rangle\,p_4\,p_6\,[t_6\rangle\,p_4\,p_7$, and $p_4\,p_8\,[t_8\rangle\,p_4\,p_5\,[t_5\rangle\,p_4\,p_6\,[t_6\rangle\,p_4\,p_7\,[t_7\rangle\,p_4\,p_8$. The last run from the list above starts and ends at the same state, i.e., the state $[p_4\,p_8]$. This fact signifies that one can replay this run over and over again and, hence, interpret the process in Fig. 3 as encoding an infinite number of runs in the corresponding net system. For example, one can conclude that the process in Fig. 3 represents, among others, the run $p_4\,p_8\,[t_8\rangle\,p_4\,p_5\,[t_5\rangle\,p_4\,p_6\,[t_6\rangle\,p_4\,p_7\,[t_7\rangle\,p_4\,p_8\,[t_8\rangle\,p_4\,p_5\,[t_5\rangle\,p_4\,p_6$.

[5] Note that tokens for markings of N_π must be drawn from $\mathbb{K}\backslash B$, where B is the set of conditions of N_π.

The process in Fig. 2c is not a reproduction process. However, this process has cuts that induce equivalent markings, e.g., cuts D_2 and D'_2. Some other pairs of cuts that induce equivalent markings and are not explicitly shown in Fig. 2c are $\{c_2, c_6\}$,$\{c_2, c'_6\}$ and $\{c_3, c_7\}$,$\{c_3, c'_7\}$. In what follows, we use this observation to propose a new interpretation for processes of net systems. Our new interpretation maximizes the portion of the corresponding net system's behavior that processes represent. We formalize this interpretation in the notion of a *process system*, which is discussed in detail in Section 3.1. Then, Sect. 3.2 proposes *process set systems* that generalize process systems to interpret behaviors encoded in sets of processes.

3.1. Process systems

Given a process $\pi := (N_\pi, \rho)$ of a net system S, one can 'explain' the behavior described in π in terms of a 'restricted' semantics of S. That is, instead of enhancing N_π with a marking, as suggested in [Des00], we propose that the somewhat modified 'token game' is played, as usual, in S. Formally, we implement the above intuition by means of *process systems*.

Definition 3.1 (*Process system*) A *process system* is a triple $\mathcal{S}_\pi := (N, M, \pi)$, where N is a net, π is a process of a net system (N, M'), and M, M' are two markings of N. ⌟

We say that $\mathcal{S}_\pi := (N, M, \pi)$, $\pi := (N_\pi, \rho)$, is a *process system of the net system* (N, M) *induced by* π if and only if $M \equiv (Min(N_\pi), \rho\ |_{Min(N_\pi)})$, i.e., π starts at M, refer to Definition 2.8. The execution semantics of a process system—similarly to the execution semantics of a net system, cf. Definition 2.4—consists of the transition enablement and transition occurrence rules. The enablement rule of a net system (N, M) depends on the structure of the net, i.e., on tokens in presets of transitions of N. In contrast, the enablement rule of a process system (N, M, π), $\pi := (N_\pi, \rho)$, relies on the structure of N_π; specifically, on cuts and events of N_π. The exact formulation of the enablement rule is due to Theorem 2.9 and the next result.

Proposition 3.2 (Process restricted transition enablement) *Let* $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, *be a process of a net system* $S := (N, M)$, *let* $D \subseteq B$ *be a set of conditions, and let* $e \in E$ *be an event. If* $\bullet e \subseteq D$, *then transition* $t := \rho(e)$ *is enabled in* (N, M'), *where* $M' := (D, \rho\ |_D)$, *i.e., it holds that* $(N, M')[t\rangle$. ⌟

Proposition 3.2 follows immediately from the fact that ρ preserves the nature of nodes and environment of transitions, cf. Definition 2.8. Finally, we propose the execution semantics of a process system.

Definition 3.3 (*Semantics of a process system*)
Let $\mathcal{S}_\pi := (N, M, \pi)$, $N := (P, T, F)$, $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, be a process system.

- A transition $t \in T$ is *enabled* in $\mathcal{S}_\pi$, denoted by $\mathcal{S}_\pi[t\rangle$, iff there exist a cut $C \subseteq B$ of N_π and an event $e \in E$ such that $M \equiv (C, \rho\ |_C)$, $\bullet e \subseteq C$, and $t = \rho(e)$.
- If a transition $t \in T$ is enabled in $\mathcal{S}_\pi$ then t can *occur*, which leads to a process system (N, M', π), where M' is a marking of N for which it holds that $(N, M)[t\rangle(N, M')$. ⌟

Intuitively, the semantics of a process system (N, M, π), $N := (P, T, F)$, $\pi := (N_\pi, \rho)$, is obtained by 'restricting' the semantics of the net system (N, M) to those markings that are induced by cuts of N_π and to those transition occurrences that are described by events of N_π. Let $\mathcal{S}_\pi$ and $\mathcal{S}'_\pi$ be two process systems. By $\mathcal{S}_\pi[t\rangle\mathcal{S}'_\pi$, we denote the fact that there exists an occurrence of transition $t \in T$ that leads from $\mathcal{S}$ to $\mathcal{S}'$. Let $\mathcal{S}_0 := (N, M, \pi)$, $N := (P, T, F)$, be a process system. Similar to net systems, we say that the state H of N induced by M is the state of $\mathcal{S}_0$. A sequence of steps $\delta := (H_0, t_1, H_1)(H_1, t_2, H_2)\ldots$, either finite or infinite, is a *run* in $\mathcal{S}_0$, iff δ is empty or there exists a sequence of process systems $\mathcal{S}_1\,\mathcal{S}_2\ldots$ such that for every position i in δ it holds that $\mathcal{S}_{i-1}[t_i\rangle\mathcal{S}_i$ and H_{i-1} and H_i are the states of $\mathcal{S}_{i-1}$ and $\mathcal{S}_i$, respectively. The notions of a reachable state, reachable marking, reachable (process) system, and that of an occurrence sequence in a (process) system are defined (mutatis mutandis) as the respective notions for net systems, cf. Definition 2.6; the only difference is that they rely on the notion of a run in a process system instead of that in a net system.

As an example, consider the process system $\mathcal{S}_\pi$ of the net system (N, M) in Fig. 1a induced by process π in Fig. 2c. Process system $\mathcal{S}'_\pi := (N, M', \pi)$, where $S' := (N, M')$ is the net system in Fig. 1b, is reachable from $\mathcal{S}_\pi$, e.g., via the run $p_1\,[t_1\rangle\,p_2\,p_6\,[t_2\rangle\,p_3\,p_6\,[t_6\rangle\,p_3\,p_7\,[t_7\rangle\,p_3\,p_8\,[t_3\rangle\,p_4\,p_8$. The net system in Fig. 1b enables transitions t_8 and t_9. $\mathcal{S}'_\pi$ restricts the semantics of S' by enabling transition t_8 only. Indeed, there exist two cuts of the causal net in Fig. 2c that induce markings equivalent to M'; these are cuts D_2 and D'_2. Further, there exist exactly two events e_8 and e'_8 for which it holds that $\bullet e_8 \subseteq D_2$ and $\bullet e'_8 \subseteq D'_2$. Finally, note that events e_8 and e'_8 both refer to transition t_8. An occurrence of transition t_8 in $\mathcal{S}'_\pi$ leads to the process system $\mathcal{S}''_\pi := (N, M'', \pi)$, where M''

induces the state $[p_4\ p_5]$. Observe that $\mathcal{S}''_\pi$ enables transition t_5 by means of the cut $\{c_4, c_5\}$ and event e_5 of the causal net in Fig. 2c.

It is easy to see that the process system $\mathcal{S}_\pi$ of the net system in Fig. 1a induced by the process in Fig. 2c describes infinite runs (occurrence sequences), which in turn signifies that there exist infinitely many runs (occurrence sequences) in $\mathcal{S}_\pi$. For example, the regular expression $t_1\ t_2\ t_6\ t_7\ t_3\ (t_8\ t_5\ t_6\ t_7)^*$ matches an infinite number of occurrence sequences in $\mathcal{S}_\pi$.[6]

3.2. Process set systems

A process system usually captures only part of the behavior of the corresponding net system. For example, the process system of net system S in Fig. 1a induced by the process in Fig. 2c does not represent any run in S that describes at least one occurrence of transition t_9. In order to overcome this limitation, we introduce a *process set system* that describes behavior of a net system restricted by a collection of processes, rather than by a single process.

Definition 3.4 (*Process set system*) A *process set system* is a triple $\mathcal{S} := (N, M, \Pi)$, where N is a net, Π is a set of processes, or an *untangling*, of a net system (N, M'), and M, M' are two markings of N. ⌟

Again, we say that $\mathcal{S} := (N, M, \Pi)$ is a *process set system of the net system* (N, M) *induced by* Π if and only if every process in Π starts at M, refer to Definition 2.8.

The semantics of a process set system is 'composed' of the semantics of several process systems.

Definition 3.5 (*Semantics of a process set system*)
Let $\mathcal{S} := (N, M, \Pi)$, $N := (P, T, F)$, be a process set system.

- A transition $t \in T$ is *enabled* in $\mathcal{S}$, denoted by $\mathcal{S}[t\rangle$, iff there exists a process $\pi \in \Pi$ for which it holds that $(N, M, \pi)[t\rangle$.
- If a transition $t \in T$ is enabled in $\mathcal{S}$ then t can *occur*, which leads to a process set system (N, M', Π'), where M' is a marking of N for which it holds that $(N, M)[t\rangle(N, M')$ and $\Pi' := \{\pi \in \Pi \mid (N, M, \pi)[t\rangle\}$. ⌟

Let $\mathcal{S}$ and $\mathcal{S}'$ be two process set systems. By $\mathcal{S}[t\rangle\mathcal{S}'$, we denote the fact that there exists an occurrence of transition $t \in T$ that leads from $\mathcal{S}$ to $\mathcal{S}'$. Let $\mathcal{S}_0 := (N, M, \Pi)$, $N := (P, T, F)$, be a process set system. Similar to net systems and process systems, the state H of N induced by M is the state of $\mathcal{S}_0$. All the standard notions are lifted to process set systems accordingly. A sequence of steps $\delta := (H_0, t_1, H_1)(H_1, t_2, H_2)\ldots$, either finite or infinite, is a *run* in $\mathcal{S}_0$, iff δ is empty or there exists a sequence of process set systems $\mathcal{S}_1\,\mathcal{S}_2\ldots$ such that for every position i in δ it holds that $\mathcal{S}_{i-1}[t_i\rangle\mathcal{S}_i$ and H_{i-1} and H_i are the states of $\mathcal{S}_{i-1}$ and $\mathcal{S}_i$, respectively. Again, the notions of a reachable state, reachable marking, reachable (process set) system, and occurrence sequence in a (process set) system are defined in the same way as to those for net systems, cf. Definition 2.6, but considering runs in process set systems.

As an example, consider the process set system $\mathcal{S}$ of the net system (N, M) in Fig. 1a induced by the set of processes $\Pi := \{\pi_1, \pi_2, \pi_3\}$, where π_1, π_2, and π_3 are the processes in Fig. 2a, Fig. 2b and Fig. 2c, respectively. The process set system $\mathcal{S}$ enables transitions t_1 and t_4. Transition t_1 is enabled via cuts D_1 and events e_1 in processes π_1 and π_3. Transition t_4 is enabled via cut D_1 and event e_4 in process π_2. An occurrence of transition t_1 in $\mathcal{S}$ leads to the process set system $\mathcal{S}' := (N, M', \Pi')$, where M' induces the state $[p_2\ p_6]$ and $\Pi' := \{\pi_1, \pi_3\}$; note that Π' does not contain process π_2 as it was not used to enable t_1. From $\mathcal{S}'$ one can reach, e.g., via the run $p_2\ p_6\,[t_2\rangle\ p_3\ p_6\,[t_3\rangle\ p_4\ p_6\,[t_6\rangle\ p_4\ p_7\,[t_7\rangle\ p_4\ p_8$, the process set system $\mathcal{S}'' := (N, M'', \Pi')$, where M'' induces the state $[p_4\ p_8]$. $\mathcal{S}''$ enables transitions t_8 and t_9. An occurrence of transition t_9 in $\mathcal{S}''$ leads to the process set system $\mathcal{S}''' := (N, M''', \{\pi_1\})$, where M''' induces the state $[p_9]$. Alternatively, an occurrence of transition t_8 in $\mathcal{S}''$ leads to the process set system $\bar{\mathcal{S}} := (N, \bar{M}, \{\pi_3\})$, where $\bar{M}$ induces the state $[p_4\ p_5]$. Note that $\mathcal{S}'''$ does not enable any transition, while $\bar{\mathcal{S}}$ enables transition t_5.

[6] In the proposed regular expression, parentheses are used to define the scope and the asterisk is used to indicate that there is zero or more of the preceding element.

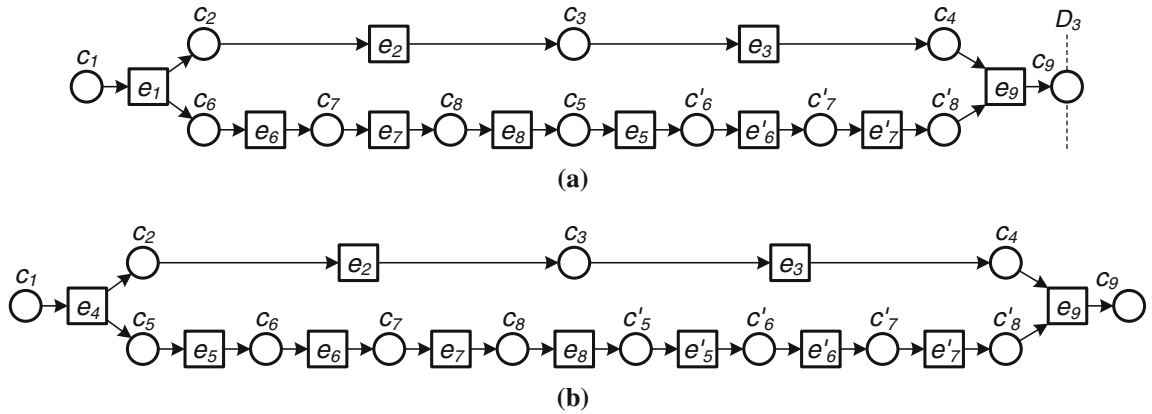

Fig. 4. A representative untangling of the net system in Fig. 1a

4. Representative untanglings

The process set system $\mathcal{S}$ of the net system S in Fig. 1a induced by the set of three processes in Fig. 2 represents a big share of runs in S, where a run δ in S is considered to be represented in $\mathcal{S}$ if and only if δ is a run in $\mathcal{S}$. Still, $\mathcal{S}$ does not represent any run in S in which transitions t_4 and t_8 both occur.

This section proposes the notion of a *representative untangling* of a net system. A representative untangling of a net system S is a set of processes of S that induces a process set system of S that describes all the behavior of S and disallows any other. Section 4.1 defines the notion of a representative untangling of a net system. Then, Sect. 4.2 proposes an algorithm for constructing representative untanglings.

4.1. Definition

A net system can be 'untangled' into a set of its processes in many different ways. Every set of processes of a net system contains information on some portion of the net system's behavior. The precise definition of this portion depends on the particular semantics that one associates with processes. If one expects to employ untangled processes for analysis, it is essential that they capture all and only the behavior of the corresponding net system (see the discussion in Sect. 3). Next, we characterize those untanglings that according to the semantics proposed in Sect. 3.2 represent the *exact* behavior of the corresponding systems.

Definition 4.1 (*Representative untangling*)
Let Π be a set of processes, i.e., an *untangling*, of a net system $S := (N, M)$.

- A process $\pi \in \Pi$ *represents* a step (H, t, H') in S iff it holds that $(N, M', \pi)[t\rangle$, where M' is a marking of N that induces H.
- A process $\pi \in \Pi$ *represents* a run δ in S, either finite or infinite, iff π represents every step of δ.
- An untangling Π of S is *representative* iff for every run δ in S there is a process $\pi \in \Pi$ that represents δ. ⌟

Figure 4 shows two processes of net system S in Fig. 1a that constitute a representative untangling of S.

Let Π be a representative untangling of a net system $S := (N, M)$. Then, it holds that S and the process set system $\mathcal{S} := (N, M, \Pi)$ are in a strong behavior equivalence relation. In fact, from the point of view of an external observer, S and $\mathcal{S}$ specify the same system. Both S and $\mathcal{S}$ induce occurrences of transitions of N. Thus, whenever a transition occurs in either of the two systems, it occurs in the same environment, i.e., the same preset and postset of places. Because a run δ in S is represented by some process $\pi \in \Pi$, it holds that δ is a run in $\mathcal{S}$. The converse, i.e., the result that a run δ in $\mathcal{S}$ is also a run in S, can be obtained by appealing to Proposition 3.2 at each step of δ. The above observations lead to the conclusion that S and $\mathcal{S}$ are two *occurrence net equivalent* systems [vGV87], which is the strongest behavioral equivalence notion for models of concurrent systems [Tar97]; it states that *unfoldings* [McM92, McM95] of both systems are isomorphic.

A representative untangling Π of a net system S induces a process set system that allows all the behavior captured in S and disallows any other. Moreover, Π provides an alternative specification of the behavior that is described in S which is particularly interesting for analysis purposes (refer to Sect. 7 for details); The unique analysis experience for S that stems from the use of Π is mainly due to the special relation between runs in S and processes in Π. For example, one can test if there exists a run in S with certain properties by checking if there exists a process in Π that allows some run with these properties. Alternatively, it is possible to check if some property holds for all runs in S by validating it against all processes in Π. Finally, as processes in Π adequately represent the ordering, causality, and concurrency relations on transition occurrences, one can rely on these relations when analyzing the described behavior.

4.2. Construction

This section proposes an algorithm that given a bounded net system S builds a representative untangling of S. It starts by suggesting an encoding for describing processes in pseudocode. The encoding is used to define an algorithm that given a run in a net system constructs its induced process. The algorithm for building representative untanglings relies on this construction as it proceeds by discovering those runs in the given net system that induce a set of processes with the desired characteristics. After having presented the untangling algorithm, this section closes with its termination and correctness analysis.

4.2.1. Encoding of processes

One can encode processes in many different ways. For example, one can adopt the approach proposed in [ERV02]. A process can be captured as a set of nodes. A *node* is either a condition or an event. A *condition* is a pair of a place it refers to and the only event in its preset, or `NIL` if the condition is a source condition. An *event* is a pair of a transition it refers to and a set of conditions in its preset. This encoding is particularly convenient for describing nets without backward conflicts. Processes are captured as causal nets and, thus, forbid backward conflicts. However, causal nets forbid forward conflicts as well, cf. Definition 2.8. Consequently, we take a different approach and represent conditions (B) and events (E) as tokens and sets of tokens, respectively, i.e., $B \subseteq \mathbb{K}$ and $E \subseteq \mathcal{P}_{\geq 1}(\mathbb{K})$.[7] A binary relation $\rho \subseteq (B \times P) \cup (E \times T)$ is used to specify the mapping of conditions to places (P) and events to transitions (T). As usual, the structure of a process is given by a flow relation $G \subseteq (B \times E) \cup (E \times B)$. This proposed process encoding follows the intuition that every condition describes a holding of a token at a certain place and every event describes an occurrence of a transition at a certain marking. Next, we implement this intuition.

4.2.2. From runs to processes

A natural way to address construction of a process is by iteratively appending fresh events to a causal net. The input to such a procedure is a run in a net system, i.e., a sequence of transition occurrences. Algorithm 1 summarizes a procedure that given a run in a net system constructs its induced process. The algorithm follows the intuition of the proof of Theorem 3.6 in [GR83].

The starting point for the construction is an 'empty' process, i.e., a process composed of conditions that correspond to tokens in the marking of the net system and no events, cf. lines 1–2 in Algorithm 1. The construction proceeds by iteratively appending events to the process via the `for` loop of lines 4–11. Every appended event corresponds to a transition occurrence described by some step of the given run. Events are appended in the order in which the respective steps appear in the run. Every fresh event that gets appended to the process and has transition t as its corresponding transition is appended together with output conditions that correspond to output places of t, refer to lines 6–9 in the algorithm. Observe that line 6 ensures that appended conditions are fresh with respect to the history of the run. The flow relation gets completed at line 8. Note that at the start of every iteration of the `for` loop, the pair (K, μ) describes a marking that enables transition t_i, i.e., induces state H_{i-1}; refer to the input run of the algorithm.

The construction proposed in Algorithm 1 is not unique, but is always possible. Moreover, a process that results from a call to this algorithm represents the run it was constructed from.

[7] $\mathcal{P}_{\geq 1}(X)$ denotes the set of all non-empty subsets of a set X, including X itself.

Algorithm 1: *Process*(S, δ) – Construct a process induced by a run

Input: A run $\delta := (H_0, t_1, H_1)(H_1, t_2, H_2)\ldots(H_{n-1}, t_n, H_n)$, $n \in \mathbb{N}_0$, in a net system $S := (N, M_0)$, $N := (P, T, F)$, $M_0 := (K_0, \mu_0)$, $t_i \in T$, $i \in [1 .. n]$
Output: A process π of S induced by δ

1 $B := K_0$; $E := \varnothing$; $G := \varnothing$; `// initialize conditions, events, and the flow relation`
2 $\rho := \mu_0$; `// initialize mapping of nodes`
3 $K := K_0$; $\mu := \mu_0$;
4 **for** $i := 1$ **to** n **do** `// iterate over positions in` δ
  `// prepare`
5 $f \subseteq \mu \rhd {\bullet t_i}$ is a bijection such that $|\ dom(f)\ | = |\ {\bullet t_i}\ |$;
6 $X \subseteq \mathbb{K} \setminus B$ is a set of tokens such that $|\ X\ | = |\ t_i{\bullet}\ |$;
7 $g : X \to t_i{\bullet}$ is a bijection between X and $t_i{\bullet}$;
  `// construct`
8 $B := B \cup X$; $E := E \cup \{K\}$; $G := G \cup (dom(f) \times \{K\}) \cup (\{K\} \times dom(g))$;
9 $\rho := \rho \cup \{(K, t_i)\} \cup g$;
10 $\mu := \mu\ \triangle (f \cup g)$; $K := dom(\mu)$;
11 **end**
12 **return** $\pi := ((B, E, G), \rho)$;

Proposition 4.2 (Processes and runs)
If δ is a finite run in a net system S, then it holds that Process(S, δ) represents δ. ⌟

Proof Let $\delta := (H_0, t_1, H_1)(H_1, t_2, H_2)\ldots(H_{n-1}, t_n, H_n)$, $n \in \mathbb{N}_0$, be a run in a net system $S := (N, M)$. For every start of the i-th iteration of the `for` loop of lines 4–11, the set $C := Max((B, E, G))$ is a cut of (B, E, G) and $(C, \rho\ |_C)$ is a marking of N that induces H_{i-1}, cf. [GR83] for details. By the end of the i-th iteration of the `for` loop, a fresh event that corresponds to transition t_i is appended to the process under construction with its preset completely in C. Hence, *Process*(S, δ) represents every step of δ, cf. Definition 4.1. ■

4.2.3. Algorithm

This section proposes an algorithm for constructing representative untanglings. The algorithm expects a bounded net system S as input and is a state space search algorithm that discovers those runs in S that induce a representative untangling of S. Searching a state space means systematically observing transition occurrences so as to visit all the reachable states of the net system. Subsequent transition occurrences make up runs of the net system. If one attempts to use some runs of S to induce a representative untangling, one must ensure that these runs contain a sufficient amount of information on the behavior that is captured in S. Construction of a run can terminate when the run leads to a net system that does not enable any transition. Additionally, we propose to terminate construction of a run if it contains a subsequence that encodes a repetitive behavior that can be 'reconstructed' from other steps of the run, i.e., the run is not *significant* with respect to repetitive behaviors which it specifies.

Definition 4.3 (*Significant run*)
A run $\delta := \chi_1\ \chi_2 \ldots$, either finite or infinite, in a net system is *repetition significant*, or *significant*, iff:

$$\forall i \in [1 .. n)\ \forall\ j \in (i .. n] : \chi_i = \chi_j \Rightarrow \exists\ k \in (i .. j)\ \forall\ m \in [1 .. i) \cup (j .. n] :\ \chi_k \neq \chi_m.$$ ⌟

Intuitively, a run δ is significant if every subsequence γ of δ composed of all steps between a pair of identical steps at two distinct positions of δ is non-empty and cannot be 'reconstructed' from steps at positions of δ that are not part of γ. We shall refer to runs that are not significant as *insignificant*. It is easy to see that every run having length less than or equal to one is significant, whereas a run that is composed of two identical steps is insignificant. As a further example consider the sequence of steps $\delta := p_1\,[t_1\rangle\ p_2\ p_6\,[t_6\rangle\ p_2\ p_7\,[t_7\rangle\ p_2\ p_8\,[t_8\rangle\ p_2\ p_5\,[t_5\rangle\ p_2\ p_6\,[t_6\rangle\ p_2\ p_7\,[t_7\rangle\ p_2\ p_8\ [t_8\rangle\ p_2\ p_5$. This sequence is a significant run in net system S in Fig. 1a. Although step $\chi := ([p_2\ p_6], t_6, [p_2\ p_7])$ appears twice in δ (at positions $i = 2$ and $j = 6$), observe that the step $([p_2\ p_5], t_5, [p_2\ p_6])$ at position $k = 5$ is unique within δ and resides between the two occurrences of χ. A similar reasoning applies to steps $([p_2\ p_7], t_7, [p_2\ p_8])$ and $([p_2\ p_8], t_8, [p_2\ p_5])$ of δ. Note, however, that sequence $\delta' := \delta + ([p_2\ p_5], t_5, [p_2\ p_6])$ is an *insignificant* run in S. [8]

[8] Let α and β be two sequences. Then, $\alpha + \beta$ is the sequence obtained by concatenating α and β, i.e., joining them end-to-end.

Algorithm 2: *BaselineRepresentativeUntangling*(S)

Input: A bounded net system $S := (N, M_0)$
Output: A representative untangling of S

```
1  Δ := {⟨⟩}; // Δ is a set of runs in S, R := {⟨⟩}, A := ∅
2  R_b^S := ∅; // initialize result with the empty set
3  while Δ ≠ ∅ do
4      Δ := Δ\{δ}, δ ∈ Δ; // δ is a run in S
5      if PE(S, δ) = ∅ then R_b^S := R_b^S ∪ {Process(S, δ)}; // construct result
6      else
7          allExtIns := true;
8          foreach χ ∈ PE(S, δ) do
9              if δ + χ is significant then
10                 Δ := Δ ∪ {δ + χ}; // R := R ∪ {δ + χ}, A := A ∪ {(δ, δ + χ)}
11                 allExtIns := false;
12             end
13         end
14         if allExtIns then R_b^S := R_b^S ∪ {Process(S, δ)}; // construct result
15     end
16 end
17 return R_b^S;
```

Let Σ be the set of all steps in a net system S and let $\delta := \chi_1 \chi_2 \ldots \chi_n$, $n \in \mathbb{N}_0$, be a run in S. By $PE(S, \delta)$ we denote the set of all *possible extensions* of δ, i.e., the set that contains every step $\chi \in \Sigma$ for which it holds that $\delta + \chi$ is a run in S. Finally, Algorithm 2 exploits the significance property of runs to construct a representative untangling of a bounded net system. Note that the code in the comments to lines 1 and 10 will be used later to reason over loop invariants.

4.2.4. Termination analysis

The `while` loop of lines 3–16 in Algorithm 2 iterates as long as there are runs in set Δ. Prior to the first iteration of the loop, at line 1, set Δ is initialized with the empty run. At every iteration of the `while` loop, one run δ is drawn from Δ at line 4. If δ has no possible extensions (see the check at line 5), then no runs are added to Δ in the same iteration of the loop. Otherwise, in the `foreach` loop of lines 8–13, every possible extension of δ that results in a significant run (refer to the check at line 9) triggers insertion of this extended run into set Δ at line 10. Observe that no run is added to set Δ within a single iteration of the `while` loop of lines 3–16 if every possible extension of δ leads to an insignificant run in S.

Given a bounded net system as input, Algorithm 2 systematically explores its runs. Let R be a set that collects runs that are added to set Δ at lines 1 and 10 along a course of execution of Algorithm 2—a set of *visited runs*; R can be constructed as specified in the comments of the applicable lines of the algorithm (if executed). It is easy to see that the `while` loop of lines 3–16 maintains the next invariant.

Invariant 4.4 (Fresh runs—the `while` loop of lines 3–16) Let R be the set of visited runs up to the current execution point in Algorithm 2. Prior to every execution of line 10, it holds that the run $\delta + \chi$ is not in R. ⌟

Proof Assume that before some execution of line 10, it holds that $\delta + \chi$ is in R. Then, in the trace of the same execution of Algorithm 2 one can observe two iterations of the `while` loop of lines 3–16 that select the same run as δ at line 4. This means that there exists an iteration of the `while` loop in the course of this execution such that before line 10 it holds that δ is in R. By applying the above reasoning iteratively, one can conclude that there exist two iterations of the `while` loop of lines 3–16 that select the empty run as δ at line 4. However, one can clearly see from the structure of the algorithm that the empty run is always selected at line 4 during the first iteration of the `while` loop and never again at any subsequent iteration. ■

The logic of Algorithm 2 makes it apparent that its execution will never terminate if the input net system has an infinite significant run. Next, we show that there are no infinite significant runs in a bounded net system.

Lemma 4.5 (Infinite runs) *An infinite run in a bounded net system is insignificant.* ⌟

Proof Let $\delta := \chi_1, \chi_2 \ldots$ be an infinite run in a bounded net system $S := (N, M)$, $N := (P, T, F)$. Assume that δ is significant. Let Σ be the set of all steps (H, t, H'), $t \in T$, in S. Because S is bounded and T is finite, it holds that Σ is finite. There exists a step $\chi \in \Sigma$ such that χ occurs infinitely often in δ; otherwise δ is finite. Then, there also exists an infinite sequence γ of positions in δ (in ascending order) such that for every element n in γ it holds that $\chi = \chi_n$. Consequently, for every pair of subsequent elements i and j in γ, i.e., $i < j$, there exists $k \in (i \,..\, j)$ such that for all $m \in [1 \,..\, i) \cup (j \,..+\infty]$ it holds that $\chi_k \neq \chi_m$; otherwise δ is insignificant. Then, it holds that χ occurs at most $\mid \Sigma \mid$ times in δ and, thus, γ is finite. ■

In other words, all significant runs in a bounded net system are finite. Another threat that can cause Algorithm 2 to run forever stems from the necessity to explore an infinite number of finite significant runs. Next, we show that there is a finite number of significant runs in a bounded net system.

Lemma 4.6 (Significant runs) *The set of all significant runs in a bounded net system is finite.* ⌟

Proof Let $S := (N, M)$, $N := (P, T, F)$, be a bounded net system and let Σ be the set of all steps (H, t, H'), $t \in T$, in S. As S is bounded and T is finite, it holds that Σ is finite. As per Lemma 4.5, every significant run in S is finite. Then, there is $n \in \mathbb{N}_0$ such that every significant run in S is a sequence of at most n steps. Thus, the number of significant runs in S cannot be larger than the size of a dictionary of all words of length at most n such that every word is composed of symbols drawn from the fixed alphabet Σ. Such a dictionary is finite. ■

Finally, Theorem 4.7 collects all the above results to show that Algorithm 2 indeed terminates for every input bounded net system.

Theorem 4.7 (Termination) *Algorithm 2 always terminates.* ⌟

Proof Algorithm 2 terminates if the `while` loop of lines 3–16 terminates. We associate a measure with every iteration of the `while` loop. Let R be a set of visited runs that gets constructed at lines 1 and 10 in the course of some execution of Algorithm 2. Let the measure of the i-th iteration of the `while` loop be a pair (Δ_i, R_i), where Δ_i and R_i are the values of Δ and R, respectively, at the time of the check at line 3 (prior to executing the body of the loop). For example, Δ_1 and R_1 are both equal to $\{\langle\rangle\}$. One can classify every scenario of a single iteration of the `while` loop into two cases according to which relation between the measure of this iteration and the measure of the next iteration this scenario leads:

(i) $\mid \Delta_{i+1} \mid - \mid \Delta_i \mid = n$, $n \in N_0$, and $R_i \subset R_{i+1}$. This case corresponds to the scenario when a run δ chosen at line 4 has at least one significant extension and, hence, line 10 is executed one or more times during the i-th iteration of the `while` loop. Observe that both $\mid \Delta_i \mid \leq \mid \Delta_{i+1} \mid$ and $R_i \subset R_{i+1}$ hold because of Invariant 4.4.

(ii) $\Delta_{i+1} \subset \Delta_i$ and $R_i = R_{i+1}$. This case corresponds to the scenario when a run δ chosen at line 4 has no extensions or all its extensions result in insignificant runs and, thus, line 10 is not executed during the i-th iteration of the `while` loop.

The key observation here is that execution of the `while` loop can follow the scenario that falls under case (i) only a finite number of times. It is easy to see that the algorithm preserves the invariant of R being composed of significant runs and the set of all significant runs is finite, refer to Lemma 4.6. Every time the `while` loop is executed according to scenario (i), the size of set Δ stays unchanged or is increased by some number n. Therefore, there exists an iteration of the `while` loop from which on every subsequent iteration of this loop will follow scenario (ii); note that at that moment in time set Δ is finite. Thus, set Δ will eventually become empty and the condition at line 3 will evaluate to false. ■

4.2.5. Correctness analysis

Let δ be a run in a net system S. If an untangling Π of S is representative, then there exists a process $\pi \in \Pi$ that represents every step of δ, cf. Definition 4.1. According to Proposition 4.2, a process that is constructed from a run using Algorithm 1 represents all steps of that run. Next, we show that for every run in a bounded net system

S it holds that it is composed of steps that also participate in some run that is used to induce a process in set $\mathcal{R}_b^S$ constructed using Algorithm 2.

Every fresh run that gets explored by Algorithm 2 is obtained from the one previously observed, refer to lines 4 and 10 of the algorithm. This fact gives rise to the next relation on runs.

Definition 4.8 (*Graph of runs*) Let R be the set of visited runs constructed by Algorithm 2 for an input bounded net system S. A *graph of runs* of S is a pair $\mathcal{G} := (R, A)$, where A is the set of pairs $(\delta, \delta + \chi)$ constructed for every pair of values for δ and χ observed at line 10 during execution of Algorithm 2 for input S. ⌟

The principles for constructing sets R and A are specified in the comments to lines 1 and 10 of Algorithm 2. Next, we point out two interesting invariants of the `while` loop of lines 3–16 that relate to graphs of runs.

Invariant 4.9 (Tree of runs—the `while` loop of lines 3–16)
Let Δ_i, $\mathcal{R}_{n,i}^S$, and $\mathcal{G}_i := (R_i, A_i)$, be the values for Δ, the set of runs used to induce the set of processes $\mathcal{R}_b^S$, and the graph of runs, respectively, at the start of the i-th iteration of the `while` loop of lines 3–16 in Algorithm 2. Then, these two statements hold at the start of the i-th iteration of the `while` loop:
(i) $\mathcal{G}_i$ is a tree, (ii) $\Delta_i \cup \mathcal{R}_{n,i}^S$ are all the leaves of the tree $\mathcal{G}_i$ which is rooted at $\langle\rangle \in R_i$ (the empty run). ⌟

Proof We show that statements (i) and (ii) both hold prior to the first iteration of the loop, and if statements (i) and (ii) both hold before some iteration of the loop, then they also hold before the next iteration.

Initialization: It is immediate that $\mathcal{G}_1 := (\{\langle\rangle\}, \varnothing)$ is a tree and $\Delta_1 \cup \mathcal{R}_{n,1}^S = \{\langle\rangle\}$ are all the leaves of $\mathcal{G}_1$.

Maintenance: (i) Because of Invariant 4.4, for every two subsequent iterations of the `while` loop of lines 3–16 it holds that $|\, R_{i+1} \,| - |\, R_i \,| = |\, A_{i+1} \,| - |\, A_i \,|$. Therefore, because $\mathcal{G}_1$ consists of a single vertex and no edges, it holds that $|\, A_{i+1} \,| = |\, R_{i+1} \,| - 1$. In addition, it clearly holds that $\mathcal{G}_{i+1}$ is connected. (ii) A run δ that is selected from Δ_i at line 4 is a leaf node of $\mathcal{G}_i$, where $\mathcal{G}_i$ is rooted at $\langle\rangle$. Note that δ is removed from Δ_{i+1} at line 4. If δ has an extension χ that leads to a significant run, then, because of Invariant 4.4, a fresh node $\delta + \chi$ is added to $\mathcal{G}_{i+1}$ at line 10, which is a leaf node of $\mathcal{G}_{i+1}$ (run $\delta + \chi$ is also added to Δ_{i+1}); otherwise no fresh run is added to R_{i+1} and, thus, δ is a leaf node of $\mathcal{G}_{i+1}$, and it is added to $\mathcal{R}_{n,i+1}^S$ at line 14. ■

Moreover, it is immediate to see that at every moment in the course of execution of Algorithm 2, set $\mathcal{R}_b^S$ is composed of processes induced by maximal significant runs, where a significant run is *maximal* if every one of its extensions leads to an insignificant run. Indeed, a run induces a process in $\mathcal{R}_b^S$ either if it has no extensions (line 5) or all its extensions are insignificant (*allExtIns* flag is set to true at line 14). Next, we propose several results on the significance of runs that we shall later orchestrate to show that Algorithm 2 is correct. We start with the claim that each extension of an insignificant run leads to an insignificant run.

Proposition 4.10 (Insignificance invariant)
Let δ be an insignificant run in a net system S. Then, a run $\delta + \chi$, where $\chi \in PE(S, \delta)$, is insignificant. ⌟

The proof of Proposition 4.10 follows immediately from Definition 4.3.

Proposition 4.11 (Infinite and finite runs) *Let δ be an infinite run in a bounded net system S. Then, there exists a finite run δ' in S such that for every step χ of δ there is a step χ' of δ' for which it holds that $\chi = \chi'$.* ⌟

Proof Let $\delta := \chi_1, \chi_2 \ldots$ be an infinite run in a bounded net system $S := (N, M)$, $N := (P, T, F)$. Let Σ be the set of all steps of δ. Because S is bounded and T is finite, it holds that Σ is finite. Let Ω be the set that for every step $\chi \in \Sigma$ contains the smallest position i in δ at which χ occurs. Then, a sequence δ' derived from δ by deleting all the elements of δ after position $\max \Omega$ is finite. Moreover, it is guaranteed that for every step χ of δ there exists a step χ' of δ' for which it holds that $\chi = \chi'$. ■

An important observation is that for every finite insignificant run δ there exists a significant run that has all the unique steps of δ as its elements.

Lemma 4.12 (Significant and insignificant runs)
Let δ be a finite insignificant run in a net system S. Then, there exists a significant run δ' in S such that for every step χ of δ there is a step χ' of δ' for which it holds that $\chi = \chi'$. ⌟

Proof Let $\delta := \chi_1 \dots \chi_n$, $n \in \mathbb{N}_0$, be a finite insignificant run in S. Then, these two conditions hold: (i) there exist two positions i and j in δ such that $i < j$ and $\chi_i = \chi_j$, and (ii) for every $k \in (i \,..\, j)$ there exists $m \in [1 \,..\, i) \cup (j \,..\, n]$ such that $\chi_k = \chi_m$. Let δ' be a sequence obtained from δ via a reduction operation that deletes all elements of δ after position i up to and including the element at position j, i.e., $\delta' := (\chi_1 \dots \chi_i) + (\chi_{j+1} \dots \chi_n)$. Clearly δ' is a run in S. Moreover, because of condition (ii) above, for every step χ of δ there is a step χ' of δ' for which it holds that $\chi = \chi'$. Observe that the length of δ' is strictly smaller than the length of δ. Assume that δ' is always insignificant. Then, one can construct an infinite sequence of finite insignificant runs in S that starts with δ and every other run in the sequence is obtained from the previous run via the reduction operation proposed above, which is impossible. ■

The next corollary is an immediate consequence of Proposition 4.11 and Lemma 4.12.

Corollary 4.13 (Runs and significant runs)
Let δ be a run, either finite or infinite, in a bounded net system S. Then, there exists a significant run δ' in S such that for every step χ of δ there is a step χ' of δ' for which it holds that $\chi = \chi'$. ⌟

Finally, Theorem 4.14 collects all the above results to demonstrate that Algorithm 2 is correct in the sense that given a bounded net system S it, indeed, computes a representative untangling of S.

Theorem 4.14 (Correctness) *Let S be a bounded net system. Then, $\mathcal{R}_b^S$ is a representative untangling of S.* ⌟

Proof Upon termination of Algorithm 2, it holds that $\Delta = \varnothing$ and, hence, set $\mathcal{R}_b^S$ contains processes induced by all the runs that correspond to the leaf nodes in the tree of runs $\mathcal{G} := (R, A)$ of S, refer to Invariant 4.9. Therefore, for every run δ in S that is composed of steps that also participate in some leaf run in $\mathcal{G}$ it holds that there exists a process in $\mathcal{R}_b^S$ that represents δ, refer to Proposition 4.2. Next, we show that the above statement holds for every significant run in S. Observe that R is composed of significant runs, see the check at line 9 of Algorithm 2, and every internal run in $\mathcal{G}$ is a subrun (a subsequence) of some leaf run in $\mathcal{G}$. Moreover, as leaf runs in $\mathcal{G}$ are maximal and because of Proposition 4.10, it holds that R is the set of all significant runs in S. Finally, according to Corollary 4.13, it holds that every insignificant run is composed from steps of some significant run that, in turn, is composed from steps of some leaf run in $\mathcal{G}$. ■

5. Transformation-based untangling

Algorithm 2 is primarily of theoretical value. As the name *BaselineRepresentativeUntangling* suggests, it should be used as a point of reference that specifies basic principles for constructing representative untanglings. In practice, this algorithm is often 'expensive' to execute as it relies on an extensive search over state spaces induced by input net systems. For example, set $\mathcal{R}_b^S$ of the net system S in Fig. 1a comprises 352 processes! In this section, we propose an enhancement to the baseline algorithm that can yield results that suit practical applications. The enhancement is based on structural transformations of input net systems that 'inform' subsequent structural transformations of resulting untanglings. The proposed enhancement is 'orthogonal' to the baseline technique as it proposes pre- and post-processing measures to be taken in combination with Algorithm 2. To set the scene, Sect. 5.1 describes the overall design of the transformation-based untangling technique and develops requirements for individual transformations. Section 5.2 proposes two transformations that fit the proposed design. Finally, Sect. 5.3 collects all the results of this section to explain the transformation-based untangling technique.

5.1. Design

The proof of correctness of Algorithm 2 exploits the observation that an untangling of S is representative if it represents every maximal significant run in S. We formalize this observation in the next statement.

Lemma 5.1 (Representative untangling) *An untangling Π of a bounded net system S is representative if and only if for every maximal significant run δ in S there exists a process $\pi \in \Pi$ that represents δ.* ⌟

The proof of Lemma 5.1 is immediate due to Definition 4.1 and the proof of Theorem 4.14.

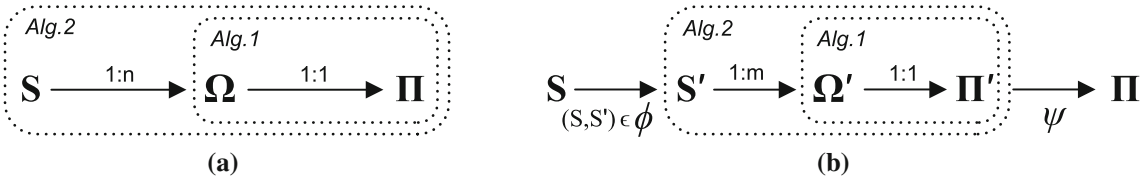

Fig. 5. Schematic visualizations of two untangling techniques: **a** Algorithm 2 only, and **b** the enhanced version of Algorithm 2 that relies on structural transformations

Figure 5a gives a schematic visualization of Algorithm 2. Starting with an input bounded net system S, the set Ω of all maximal significant runs in S is discovered; let n be the cardinality of Ω, $|\ \Omega\ | = n$. Every maximal significant run that appears in Ω triggers construction of one process in the resulting representative untangling Π of S. In the figure, boxes with dotted borderlines denote scopes of the respective algorithms.

As pointed out in the previous section, Algorithm 2 suffers from the fact that the number n of all maximal significant runs in a bounded net system can be immense. Consequently, constructed representative untanglings can contain extremely large numbers of processes, which restricts their practical applicability. To address this issue, we propose an enhancement to the baseline untangling technique. Given a bounded net system S, we suggest that one is as far as three steps away from a representative untangling of S:

1. S gets transformed into a bounded net system S' using some structural transformation rule ϕ.[9]
2. A representative untangling Π' of S' is discovered by constructing processes that represent all maximal significant runs Ω' in S', where $|\ \Omega'\ | = m$, using Algorithm 2.
3. To compensate for the effects of the transformation at step 1, every process in Π' gets transformed using some rule ψ (an 'inverse' of ϕ) to obtain Π—a representative untangling of S.

The logic of this three-step approach is summarized in the schematic drawing in Fig. 5b. It is designed to benefit from the decrease in the number m of maximal significant runs in S' as compared to those in S; it is expected that m is much less than n, i.e., $m \ll n$. It is reasonable to expect such a decrease when at step 1 of the outlined procedure one applies a structural reduction on a given net system, as many well-established structural reduction rules do indeed lead to a noticeable decrease in the number of runs in the reduced system [Mur89].

Clearly, not every pair of rules ϕ and ψ fits the proposed algorithm design. We noticed that rules ϕ and ψ fit the design if there exists a special relation between runs in a given net system S and runs in a net system S' (the net system that S is transformed into). More precisely, given a transformation $(S, S') \in \phi$ and a transformation rule ψ between processes of S and S', one can construct a representative untangling of S by relying on S' and ψ, as suggested in Fig. 5b, if there exists a relation γ between a superset Δ of all the maximal significant runs Ω in S and a superset Λ' of all the maximal significant runs in S' such that S, S', ψ, and γ all relate to each other in a certain way, which will be detailed below. Figure 6 shows an augmented version of the schematic drawing in Fig. 5b; γ is introduced to characterize cases when processes in Π represent runs in Ω and, thus, all runs in S (cf. Lemma 5.1).

Firstly, rule ψ must *correspond* to the transformation $(S, S') \in \phi$.

Definition 5.2 (*Corresponding transformation rule: processes*) A transformation rule ψ between processes of net systems *corresponds* to a net system transformation $(S, S') \in \phi$ iff for every process π' of S' that represents at least one maximal significant run in S' it holds that there exists a process π of S such that $(\pi, \pi') \in \psi$. ⌟

Intuitively, correspondence of a rule ψ to a transformation (S, S') hints at the possibility of using ψ to transform every process in Π' into a process in Π.

Secondly, rule γ must *correspond* to the transformation $(S, S') \in \phi$.

Definition 5.3 (*Corresponding transformation rule: runs*)
A transformation rule γ between runs in net systems *corresponds* to a net system transformation $(S, S') \in \phi$ iff: (i) for every maximal significant run δ in S it holds that there is a run δ' in S' such that $(\delta, \delta') \in \gamma$, and (ii) for all $(\delta, \delta') \in \gamma$ it holds that if δ is a maximal significant run in S then δ' is a significant run in S'. ⌟

[9] A *transformation rule*, or a *rule*, α between two sets A and B is a binary relation between A and B. A pair $(a, b) \in \alpha$ is a *transformation*. The fact that $(a, b) \in \alpha$ denotes that a can be transformed into b using α, and vice versa.

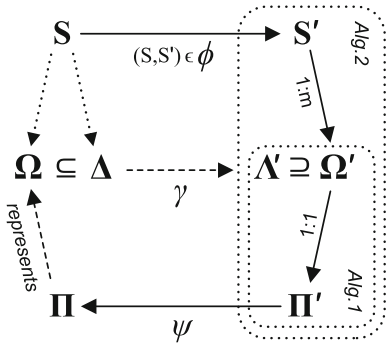

Fig. 6. Augmented version of the schematic visualization of the enhanced version of Algorithm 2 in Fig. 5b

Intuitively, correspondence of a rule γ to a transformation (S, S') hints at the possibility of using γ to transform every maximal significant run in S into a significant run in S'.

Finally, rules ψ and γ must be *compatible*.

Definition 5.4 (*Compatible transformations*) A transformation rule ψ between processes of net systems and a transformation rule γ between runs in net systems are *compatible* if and only if for all $(\pi, \pi') \in \psi$ and for all $(\delta, \delta') \in \gamma$ it holds that if π' represents δ' then π represents δ. ⌟

A net system transformation rule ϕ is *sound*, iff for every $(S, S') \in \phi$ it holds that S and S' are bounded and there exist transformation rules ψ and γ between processes of S and S', and runs in S and S', respectively, such that ψ and γ correspond to (S, S'), and ψ and γ are compatible.

Every transformation taken from a sound rule fits the above proposed three-step untangling approach.

Lemma 5.5 (Representativeness inheritance) *Let $(S, S') \in \phi$ be a transformation between two bounded net systems. Let ψ be a transformation rule between processes of S and S' and let γ be a transformation rule between runs in S and S'. Finally, let Π' be a representative untangling of S'. If ψ and γ correspond to the transformation (S, S') and are compatible, then $\Pi := \psi^{-1}(\Pi')$ is a representative untangling of S.* ⌟

Proof According to Lemma 5.1, it suffices to show that Π represents every maximal significant run in S. Let δ be a maximal significant run in S. Because γ corresponds to the transformation (S, S'), there exists a significant run δ' in S' such that $(\delta, \delta') \in \gamma$. It trivially holds that there exists a maximal significant run δ'' in S' such that δ' is a subrun of δ''; δ'' is a maximal extension of δ' that is significant. By definition, Π' represents every run and, thus, every maximal significant run in S'. Therefore, there exists a process π' of S' that represents δ'' and, thus, it represents δ'. Because ψ corresponds to the transformation (S, S'), there exists a process π of S such that $(\pi, \pi') \in \psi$. Finally, because ψ and γ are compatible, it holds that π represents δ. ■

A net system transformation that can be expressed as a sequence of transformations, each taken from a sound rule, also fits the above proposed design of the untangling algorithm.

Lemma 5.6 (Compositionality) *Let $(S_0, S_1) \cdots (S_{n-1}, S_n)$, $n \in \mathbb{N}$, be a sequence of transformations of bounded net systems. Let $\psi_1 \cdots \psi_n$ and $\gamma_1 \cdots \gamma_n$ be transformation rules such that for every position $i \in [1 \,..\, n]$ it holds that ψ_i and γ_i are rules between processes of S_{i-1} and S_i, and runs in S_{i-1} and S_i, respectively, that correspond to the transformation (S_{i-1}, S_i) and are compatible. If Π_n is a representative untangling of S_n, then $\Pi_0 := (\psi_1^{-1} \circ \cdots \circ \psi_n^{-1})(\Pi_n)$ is a representative untangling of S_0.* ⌟

Proof By induction on the length $k \in \mathbb{N}$ of a given sequence of transformations.

Base: Let (S_0, S_1) be a sequence of net system transformations of length $k = 1$. Then, the statement holds because of Lemma 5.5 and the fact that ψ_1 and γ_1 correspond to (S_0, S_1) and are compatible, i.e., $\Pi_0 := \psi_1^{-1}(\Pi_1)$ is a representative untangling of S_0, where Π_1 is a representative untangling of S_1.

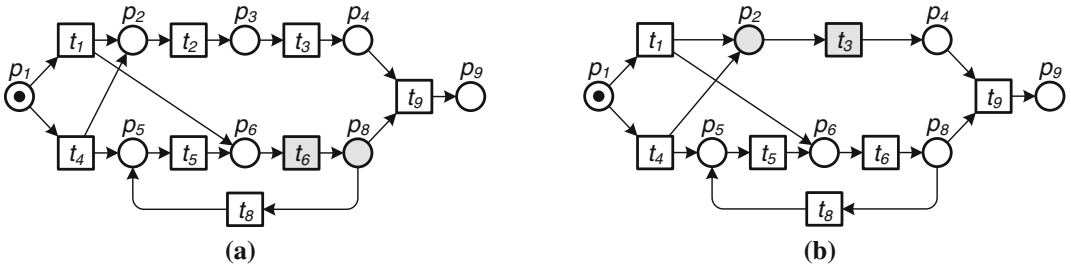

Fig. 7. a A net system obtained from the net system in Fig. 1a via the FST-reduction from t_6 to p_8, and **b** a net system obtained from the net system in Fig. 7a via the FSP-reduction from p_2 to t_3

Step: Assume that the statement holds for a sequence of net system transformations $(S_0, S_1) \dots (S_{k-1}, S_k)$ of length $1 \leq k < n$, i.e., it holds that $\Pi_0 := (\psi_1^{-1} \circ \dots \circ \psi_k^{-1})(\Pi_k)$ is a representative untangling of S_0, where Π_k is a representative untangling of S_k. Then, the statement must also hold for a sequence of net system transformations $(S_0, S_1) \dots (S_k, S_{k+1})$. Indeed, according to Lemma 5.5, it holds that $\Pi'_k := \psi_{k+1}^{-1}(\Pi_{k+1})$ is a representative untangling of S_k, where Π_{k+1} is a representative untangling of S_{k+1}. ■

5.2. Structural reductions

This section proposes two sound *reduction* rules on net systems. These rules allow transforming net systems into systems with fewer nodes. The names of the proposed rules are due to the names of the rules by Murata, as they are inspired by transformation rules in [Mur89].

5.2.1. Fusion of series transitions

Fusion of series transitions is a reduction rule on net systems that can be used to 'skip' a sequence of nodes composed of a place followed by a transition. The rule is defined as follows.

Definition 5.7 (*Fusion of series transitions (FST): net systems*) Let S and S' be two net systems, where $S := (N, M)$, $N := (P, T, F)$, $M := (K, \mu)$, and $S' := (N', M)$, $N' := (P', T', F')$. A pair (S, S') is an *FST-reduction from a transition $t \in T$ to a place $p \in P$* iff:

- there exist a place $p' \in P$ and a transition $t' \in T$ such that $\bullet p' = \{t\}$, $p'\bullet = \{t'\}$, $\bullet t' = \{p'\}$, $t'\bullet = \{p\}$, $(t, p) \notin F$, $\forall\ k \in K : \mu(k) \neq p'$, and
- $P' = P \backslash \{p'\}$, $T' = T \backslash \{t'\}$, $F' = (F \backslash \{(t, p'), (p', t'), (t', p)\}) \cup \{(t, p)\}$.

The *FST-reduction rule ϕ_A on net systems* consists of all FST-reductions on all net systems. ⌟

For example, a transformation (S_0, S_1), where S_0 and S_1 are the net systems in Fig. 1a and Fig. 7a, respectively, is an FST-reduction between S_0 and S_1 from transition t_6 to place p_8 (highlighted with grey background in Fig. 7a). Given an FST-reduction $(S, S') \in \phi_A$ on net systems, the next transformation rule can be used (as it will be demonstrated later) as a rule that corresponds, see Definition 5.2, to (S, S').

Definition 5.8 (*Fusion of series transitions (FST): processes*) Let $(S, S') \in \phi_A$ be an FST-reduction on net systems, where $S := (N, M)$, $N := (P, T, F)$, and $S' := (N', M)$, $N' := (P', T', F')$. Let $\pi' := (N_{\pi'}, \rho')$, $N_{\pi'} := (B', E', G')$, be a process of S', let $p \in P$ and $t \in T$ be such that $P' = P \backslash \{p\}$ and $T' = T \backslash \{t\}$, and let $\mathcal{G} := \{(e, b) \in G' \mid (\rho'(e), \rho'(b)) \notin F\}$. A pair (π, π') is an *FST-reduction to process π' induced by* (S, S') if and only if $\pi := (N_\pi, \rho)$, where $N_\pi := (B, E, G)$, is a pair such that:

- $B = B' \cup \{b_g \mid g \in \mathcal{G}\}$, $E = E' \cup \{e_g \mid g \in \mathcal{G}\}$, $G = (G' \backslash \mathcal{G}) \cup \{(e, b_{(e,b)}) \mid (e, b) \in \mathcal{G}\} \cup \{(b_g, e_g) \mid g \in \mathcal{G}\} \cup \{(e_{(e,b)}, b) \mid (e, b) \in \mathcal{G}\}$, and
- $\rho = \rho' \cup \{(b_g, p) \mid g \in \mathcal{G}\} \cup \{(e_g, t) \mid g \in \mathcal{G}\}$.

The *FST-reduction rule ψ_A on processes induced by* (S, S') consists of all FST-reductions to all processes of S'. ⌟

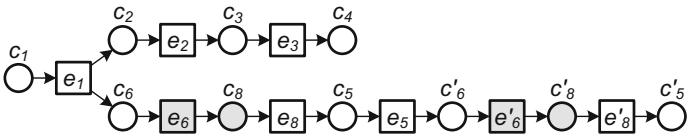

Fig. 8. A process of the net system in Fig. 7a

Let π and π' be processes in Fig. 2c and Fig. 8, respectively. Then, it holds that $(\pi, \pi') \in \psi_A$, where ψ_A is the FST-reduction rule on processes induced by (S_0, S_1). Again, S_0 and S_1 are the net systems in Fig. 1a and Fig. 7a, respectively. The reduction takes place between the nodes that are highlighted in Fig. 8. Indeed, according to Definition 5.8, an FST-reduction on processes can affect several places in the processes under transformation. Finally, we show that given an FST-reduction $(S, S') \in \phi_A$ on net systems, one can rely on rule ψ_A on processes to transform a process of S' into a process of S.

Proposition 5.9 (Correctness of ψ_A) *Let $(S, S') \in \phi_A$ be an FST-reduction on net systems. If π' is a process of S' and $(\pi, \pi') \in \psi_A$ is an FST-reduction on processes induced by (S, S'), then π is a process of S.* ⌟

Proof (Sketch) Because $\pi' := (N_{\pi'}, \rho')$, $N_{\pi'} := (B', E', G')$ is a process of $S' := (N', M)$, $N' := (P', T', F')$, and because of Definition 5.8, it holds that N_π, where $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, is a causal net. Moreover, π satisfies all the three criteria of a process of $S := (N, M)$, $N := (P, T, F)$, cf. Definition 2.8:

- (ρ preserves the nature of nodes). Because π' is a process of S', it holds that ρ' preserves the nature of nodes. Because of Definition 5.8, it holds that $\rho\ |_{dom(\rho')} = \rho'$. Again, because of Definition 5.8, there exist $p \in P$ and $t \in T$ such that $\forall\ b \in B \backslash B' : \rho(b) = p$ and $\forall\ e \in E \backslash E' : \rho(e) = t$.
- (π starts at M). Because of Definition 5.8, i.e., it trivially holds that $Min(N_\pi) = Min(N_{\pi'})$ and $\rho\ |_{Min(N_\pi)} = \rho'\ |_{Min(N_{\pi'})}$. Finally, observe that M is the marking in both S and S'.
- (ρ respects the environment of transitions). Let $t, t' \in T$ and $p, p' \in P$ be such that $T = T' \backslash \{t\}$, $P = P' \backslash \{p\}$, $\{t'\} = {\bullet p}$, and $\{p'\} = t\bullet$. Assume that there exists an event $e \in E$ for which ρ does not respect the environment of the corresponding transition. We consider these two cases:
  1. $(\rho(e) = t)$. Because of Definitions 5.7 and 5.8, it holds that ${\bullet \rho(e)} = \{p\} = \rho({\bullet e})$ and $\rho(e)\bullet = \{p'\} = \rho(e\bullet)$.
  2. $(\rho(e) \neq t)$. Again, because of Definitions 5.7 and 5.8, it holds that ${\bullet \rho(e)} = \rho({\bullet e})$ and $\rho(e)\bullet = \rho(e\bullet)$. ■

Given an FST-reduction $(S, S') \in \phi_A$ on net systems, one can use the FST-reduction rule on processes induced by (S, S') as a rule that *corresponds*, cf. Definition 5.2, to (S, S').

Lemma 5.10 (Correspondence of ψ_A to $(S, S') \in \phi_A$) *Let $(S, S') \in \phi_A$ be an FST-reduction on net systems and let ψ_A be the FST-reduction rule on processes induced by (S, S'). Then, it holds that ψ_A corresponds to (S, S').* ⌟

Proof By Definition 5.8, it holds that for every process π' of S' and, thus, for every process of S' that represents at least one maximal significant run in S', there exists $(\pi, \pi') \in \psi_A$. Moreover, according to Proposition 5.9, π is a process of S. ■

Next, we propose a transformation rule between runs in net systems that can be used as a rule that *corresponds*, cf. Definition 5.3, to a given FST-reduction on net systems and is *compatible*, cf. Definition 5.4, with the respective rule ψ_A on processes of net systems. We shall make several 'stops' on the way to this desired transformation. The first stop is the notion of a run adaptation rule.

Definition 5.11 (*Run adaptation*) Let $(S, S') \in \phi_A$ be an FST-reduction on net systems, where $S := (N, M)$, $N := (P, T, F)$, and $S' := (N', M)$, $N' := (P', T', F')$. Let $p \in P$ and $t \in T$ be such that $P' = P \backslash \{p\}$ and $T' = T \backslash \{t\}$. Finally, let δ be a run in S. A pair (δ, δ') is a *run adaptation on δ induced by* (S, S') if and only if δ' is a sequence of steps obtained from δ in two stages (in the specified order):

1. Construct sequence δ'' from δ by removing all steps (H, t', H') of δ for which it holds that $t' = t$, without changing the order of the remaining steps.
2. Construct sequence δ' from δ'' by replacing every element p with a fresh element p' in H and H' of every step (H, t', H') of δ'', where $t\bullet = \{p'\}$.

The *run adaptation rule γ_α on runs induced by* (S, S') consists of all run adaptations on all runs in S. ⌟

Given an FST-reduction $(S, S') \in \phi_A$ on net systems, one can use γ_α to transform a run in S into a run in S'.

Proposition 5.12 (Correctness of γ_α) *Let $(S, S') \in \phi_A$ be an FST-reduction on net systems. If δ is a run in S and $(\delta, \delta') \in \gamma_\alpha$ is a run adaptation induced by (S, S'), then δ' is a run in S'.* ⌟

Proof (Sketch) By induction on the length $k \in \mathbb{N}_0$ of δ.

Base: Let δ be empty, i.e., $k = 0$. Then, δ' is also empty and, thus, is a run in S'.

Step: Let us assume that the statement holds for $\delta := \chi_1 \cdots \chi_k$, $k \geq 0$, i.e., it holds that δ' is a run in $S' := (N', M)$, $N' := (P', T', F')$. Let $\kappa := \delta + \chi_{k+1}$, where $\chi_{k+1} := (H, t', H')$, be a run in $S := (N, M)$, $N := (P, T, F)$, and let $(\kappa, \kappa') \in \gamma_\alpha$. Finally, let $p \in P$ and $t \in T$ be such that $P' = P \backslash \{p\}$ and $T' = T \backslash \{t\}$. We consider these three cases:

1. $(t = t')$. If $t = t'$, then because of Definition 5.11 it holds that $\kappa' = \delta'$ and, thus, κ' is a run in S'.
2. $(t \neq t', p \in H$, and $p \notin H')$. This case is not possible as it must hold that $t = t'$.
3. $(t \neq t'$, and $p \notin H$ or $p \in H')$. Because of Definitions 5.7 and 5.11, it holds that κ' is a run in S'. ■

The next stop towards the desired transformation on runs is a transformation that given a run δ in a net system S constructs a significant run in S which is composed of all distinct steps of δ.

Definition 5.13 (*Insignificance reduction*) Let $\delta := \chi_1 \cdots \chi_n$, $n \in \mathbb{N}_0$, be a run in a net system.

- A *one step insignificance reduction* (between i and j) of δ is *applicable* if and only if there exist positions i and j in δ, $i < j$, such that $\chi_i = \chi_j$, and for each position k in δ, $i < k < j$, it holds that step χ_k is equal to some step χ_m of δ, where $m < i$ or $j < m$.
- An application of the reduction between i and j results in a fresh run $\delta' := (\chi_1 \cdots \chi_i) + (\chi_{j+1} \cdots \chi_n)$. ⌟

It is easy to see that one eventually obtains a significant run by utilizing one step insignificance reductions on an input run (one after another as long as they are applicable). Observe that one may end up with different significant runs by applying different (in terms of different positions between which reductions are applied) sequences of one step insignificance reductions. By Λ_δ, we denote the set of all significant runs that can be obtained from a given run δ through different sequences of one step insignificance reductions.

Considering the above, a suitable transformation rule on runs can be defined as follows.

Definition 5.14 (*Run adjustment*) Let $(S, S') \in \phi_A$ be an FST-reduction on net systems and let δ be a run in S.

- A pair (δ, δ') is a *run adjustment on run δ induced by* (S, S') if and only if there exists a run δ'' in S' such that $(\delta, \delta'') \in \gamma_\alpha$ is a run adaptation and it holds that $\delta' \in \Lambda_{\delta''}$.
- The *run adjustment rule γ_β on runs induced by* (S, S') consists of all run adjustments on all runs in S. ⌟

Given an FST-reduction $(S, S') \in \phi_A$ on net systems, one can use the run adjustment rule induced by (S, S') as a rule that *corresponds*, cf. Definition 5.3, to (S, S').

Lemma 5.15 (Correspondence of γ_β to $(S, S') \in \phi_A$) *Let $(S, S') \in \phi_A$ be an FST-reduction on net systems and let γ_β be the run adjustment rule induced by (S, S'). Then, it holds that γ_β corresponds to (S, S').* ⌟

Proof By Definition 5.14, it holds that for every run δ in S and, thus, for every maximal significant run in S, there exists a run δ' in S' such that $(\delta, \delta') \in \gamma_\beta$. According to Proposition 5.12 and Definition 5.14, δ' is a significant run in S'. ■

The next statement claims that transformation rules ψ_A and γ_β are *compatible*, cf. Definition 5.4.

Lemma 5.16 (Compatibility of ψ_A and γ_β) *Let $(S, S') \in \phi_A$ be an FST-reduction on net systems, and let ψ_A and γ_β be the FST-reduction rule on processes induced by (S, S') and the run adjustment rule induced by (S, S'), respectively. Then, it holds that ψ_A and γ_β are compatible.* ⌟

Proof (Sketch) Let us assume that ψ_A and γ_β are incompatible. Then, there exist $(\pi, \pi') \in \psi_A$ and $(\delta, \delta') \in \gamma_\beta$ such that π' represents δ' but π does not represent δ, i.e., there exists a step $\chi := (H, t, H')$ of δ that π does not represent. Let $S := (N, M)$, $N := (P, T, F)$, $S' := (N', M)$, $N' := (P', T', F')$, and let $p', p'' \in P$ and $t' \in T$ be such that $P' = P \backslash \{p'\}$, $T' = T \backslash \{t'\}$, and $\{p''\} = t' \bullet$. We consider these two cases:

1. (χ is a step of δ'). It follows immediately from Definition 5.8 that π represents χ.
2. (χ is *not* a step of δ'). We consider these two subcases:
   (a) (There exists a step χ' of δ' that represents an occurrence of t).
   Let $\chi' := (Q, t, Q')$. We consider these three subcases:
   (i) ($H = Q$, $H' \neq Q'$, and Q' can be obtained from H' by replacing every element p' with p''). Because of Definition 5.8, π represents χ. Then, there exists a cut C and an event e in π such that C induces H and e represents an occurrence of t.
   (ii) ($H' = Q'$, $H \neq Q$, and Q can be obtained from H by replacing every element p' with p''). It must hold that $t = t'$ and, thus, because of Definition 5.14, there exists no step χ' of δ'.
   (iii) ($H \neq Q$, $H' \neq Q'$, and Q and Q' can be obtained from H and H', respectively, by replacing every element p' with p''). Because of Definition 5.8, π represents χ. Then, there exists a cut C and an event e in π such that C induces H and e represents an occurrence of t.
   (b) (There exists *no* step χ' of δ' that represents an occurrence of t). It must hold that $t = t'$. There exists a step $\chi' := (Q, t^*, Q')$ of δ' such that either $Q = H'$ or $Q' = H'$. Thus, there exists a cut C in π' that induces H'. Then, there exists a cut C' in π that induces H'. Finally, because of Definition 5.8, it is easy to see that π represents χ. ■

We conclude this section with a claim that the FST-reduction rule on net systems is *sound*.

Lemma 5.17 (Soundness of ϕ_A) *The FST-reduction rule ϕ_A on net systems is sound.* ⌟

Proof Let $(S, S') \in \phi_A$ be an FST-reduction on net systems. Let ψ_A be the FST-reduction on processes of net systems and let γ_β be the run adjustment rule; both induced by (S, S'). According to Lemmas 5.10 and 5.15, rules ψ_A and γ_β correspond to (S, S'). Finally, according to Lemma 5.16, rules ψ_A and γ_β are compatible. ■

5.2.2. *Fusion of series places*

Fusion of series places is a reduction rule on net systems that can be used to 'skip' a sequence of nodes composed of a transition followed by a place. The precise definition of the rule proceeds as follows.

Definition 5.18 (*Fusion of series places* (*FSP*): *net systems*)
Let S and S' be two net systems, where $S := (N, M)$, $N := (P, T, F)$, $M := (K, \mu)$, and $S' := (N', M)$, $N' := (P', T', F')$. A pair (S, S') is an *FSP-reduction from a place $p \in P$ to a transition $t \in T$* iff:

- there exist a place $p' \in P$ and a transition $t' \in T$ such that $\bullet t' = \{p\}$, $t' \bullet = \{p'\}$, $\bullet p' = \{t'\}$, $p' \bullet = \{t\}$, $(p, t) \notin F$, ($|\, p \bullet \,| = 1$ or $|\, \bullet t \,| = 1$), $\forall\ k \in K : \mu(k) \neq p'$, and
- $P' = P \backslash \{p'\}$, $T' = T \backslash \{t'\}$, $F' = (F \backslash \{(p, t'), (t', p'), (p', t)\}) \cup \{(p, t)\}$.

The *FSP-reduction rule ϕ_B on net systems* consists of all FSP-reductions on all net systems. ⌟

For example, a transformation (S_1, S_2), where S_1 and S_2 are the net systems in Fig. 7a and Fig. 7b, respectively, is an FSP-reduction from place p_2 to transition t_3 (highlighted with grey background in Fig. 7b). Given an FSP-reduction $(S, S') \in \phi_B$ on net systems, the next transformation rule can be used as a rule that corresponds, see Definition 5.2, to (S, S').

Definition 5.19 (*Fusion of series places* (*FSP*): *processes*) Let $(S, S') \in \phi_B$ be an FSP-reduction on net systems, where $S := (N, M)$, $N := (P, T, F)$, and $S' := (N', M)$, $N' := (P', T', F')$. Let $\pi' := (N_{\pi'}, \rho')$, $N_{\pi'} := (B', E', G')$, be a process of S', let $p \in P$ and $t \in T$ be such that $P' = P \backslash \{p\}$ and $T' = T \backslash \{t\}$, let $\mathcal{G} := \{(b, e) \in G' \,|\, (\rho'(b), \rho'(e)) \notin F\}$, and let $\mathcal{B} := \{c \in B' \,|\, c \bullet = \varnothing \ \wedge\ t \in \rho(c) \bullet\}$. A pair (π, π') is an *FSP-reduction* to process π' induced by (S, S') if and only if $\pi := (N_\pi, \rho)$, where $N_\pi := (B, E, G)$, is a pair such as:

- $B = B' \cup \{b_g \,|\, g \in \mathcal{G}\} \cup \{b_c \,|\, c \in \mathcal{B}\}$, $E = E' \cup \{e_g \,|\, g \in \mathcal{G}\} \cup \{e_c \,|\, c \in \mathcal{B}\}$, $G = (G' \backslash \mathcal{G}) \cup \{(b, e_{(b,e)}) \,|\, (b, e) \in \mathcal{G}\} \cup \{(e_g, b_g) \,|\, g \in \mathcal{G}\} \cup \{(b_{(b,e)}, e) \,|\, (b, e) \in \mathcal{G}\} \cup \{(c, e_c) \,|\, c \in \mathcal{B}\} \cup \{(e_c, b_c) \,|\, c \in \mathcal{B}\}$, and
- $\rho = \rho' \cup \{(b_g, p) \,|\, g \in \mathcal{G}\} \cup \{(e_g, t) \,|\, g \in \mathcal{G}\} \cup \{(b_c, p) \,|\, c \in \mathcal{B}\} \cup \{(e_c, t) \,|\, c \in \mathcal{B}\}$.

The *FSP-reduction rule ψ_B on processes induced by (S, S')* consists of all FSP-reductions to all processes of S'. ⌟

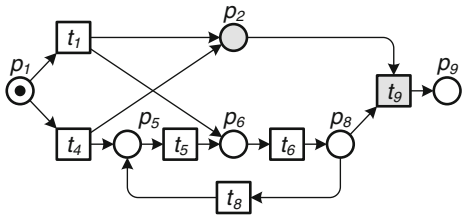

Fig. 9. A net system obtained from the net system in Fig. 7b via the FSP-reduction from p_2 to t_9

The proof of the fact that the FSP-reduction rule ϕ_B on net systems is sound proceeds similar to the proof of Lemma 5.17 with a minor difference that it relies on rule ψ_B on processes, cf. Definition 5.19, and rule γ_β on runs in net systems, cf. Definition 5.14. Note that one can prove that rules ψ_B and γ_β correspond to an FSP-reduction on net systems and that they are compatible similar to the proofs of Lemmas 5.10, 5.15, and 5.16, changing those things that need to be changed.

5.3. Algorithm

This section organizes all the above proposed results in an alternative technique for computing representative untanglings. Given a *repertoire*, i.e., a set, of sound transformation rules Φ on net systems, the baseline construction of a representative untangling of a bounded net system S_0 can be augmented as follows. Firstly, S_0 gets transformed into a net system S_n via a sequence of transformations $\kappa := (S_0, S_1) \dots (S_{n-1}, S_n)$, $n \in \mathbb{N}$, where each transformation belongs to some rule in Φ. Secondly, a representative untangling $\mathcal{R}_b^{S_n}$ of S_n is constructed using Algorithm 2. Finally, the set $(\psi_1^{-1} \circ \dots \circ \psi_n^{-1})(\mathcal{R}_b^{S_n})$ is a representative untangling of S_0, denoted by $\mathcal{R}_\kappa^{S_0}$, where $\psi_1 \dots \psi_n$ correspond to transformations $(S_0, S_1) \dots (S_{n-1}, S_n)$, respectively. Termination of the outlined algorithm is guaranteed by the finiteness of κ and the proof of termination of Algorithm 2, whereas correctness is guaranteed by Lemma 5.6 and the proof of correctness of Algorithm 2.

Given a bounded net system S, the construction of $\mathcal{R}_\kappa^S$ depends on κ, i.e., different sequences of transformations usually lead to different representative untanglings. However, it is easy to see that if one constructs κ as long as possible by using rules from the repertoire $\Phi := \{\phi_A, \phi_B\}$, in any order, one always ends up with the same set $\mathcal{R}_\kappa^S$ (if ψ_A and ψ_B are used as rules that correspond to ϕ_A and ϕ_B, respectively).

Next, we exemplify the transformation-based untangling technique by constructing a representative untangling of the net system in Fig. 1a. Firstly, one needs to come up with a sequence of transformations on the given net system, where each transformation must belong to a sound transformation rule. It is intended that the transformations reduce the net system structurally. For example, the sequence $\kappa := (S_0, S_1)(S_1, S_2)(S_2, S_3)$, where S_0, S_1, S_2, and S_3 are the net systems in Fig. 1a, Fig. 7a, Fig. 7b and Fig. 9, respectively, is a sequence of transformations that reduces S_0 to S_3. It holds that $(S_0, S_1) \in \phi_A$ and $(S_1, S_2), (S_2, S_3) \in \phi_B$, refer to Sect. 5.2. Observe that S_3 is obtained from S_2 via the FSP-reduction from place p_2 to transition t_9 (highlighted with grey background in Fig. 9). Thus, all transformations of κ belong to sound transformation rules. Secondly, a representative untangling $\mathcal{R}_b^{S_3}$ of S_3 must be constructed. We rely on Algorithm 2 to accomplish this step. Algorithm 2 proceeds by discovering all the five maximal significant runs in S_3:

$\delta_0 := p_1\,[t_1\rangle\,p_2\,p_6\,[t_6\rangle\,p_2\,p_8\,[t_9\rangle\,p_9$,
$\delta_1 := p_1\,[t_4\rangle\,p_2\,p_5\,[t_5\rangle\,p_2\,p_6\,[t_6\rangle\,p_2\,p_8\,[t_9\rangle\,p_9$,
$\delta_2 := p_1\,[t_1\rangle\,p_2\,p_6\,[t_6\rangle\,p_2\,p_8\,[t_8\rangle\,p_2\,p_5\,[t_5\rangle\,p_2\,p_6\,[t_6\rangle\,p_2\,p_8\,[t_9\rangle\,p_9$,
$\delta_3 := p_1\,[t_4\rangle\,p_2\,p_5\,[t_5\rangle\,p_2\,p_6\,[t_6\rangle\,p_2\,p_8\,[t_8\rangle\,p_2\,p_5\,[t_5\rangle\,p_2\,p_6\,[t_6\rangle\,p_2\,p_8\,[t_9\rangle\,p_9$, and
$\delta_4 := p_1\,[t_1\rangle\,p_2\,p_6\,[t_6\rangle\,p_2\,p_8\,[t_8\rangle\,p_2\,p_5\,[t_5\rangle\,p_2\,p_6\,[t_6\rangle\,p_2\,p_8\,[t_8\rangle\,p_2\,p_5$.

Algorithm 2 uses the above five runs to construct five process in $\mathcal{R}_b^{S_3}$. Finally, the set $\mathcal{R}_\kappa^{S_0}$ that is defined as $(\psi_A^{-1} \circ \psi_B^{-1} \circ \psi_B^{-1})(\mathcal{R}_b^{S_3})$, where ψ_A and ψ_B are transformations on processes of net systems that correspond to transformations drawn from rules ϕ_A and ϕ_B, respectively (refer to Sects. 5.2.1 and 5.2.2 for details), is a representative untangling of S_0. The set $\mathcal{R}_\kappa^{S_0}$ contains the five processes shown in Figs. 2 and 4. Recall that the set $\mathcal{R}_b^{S_0}$ contains 352 processes. Indeed, it holds that $|\,\mathcal{R}_\kappa^{S_0}\,| \ll |\,\mathcal{R}_b^{S_0}\,|$.

6. Related work

An untangling of a concurrent system is a mathematical formalism that can be used to describe and analyze the behavior of that system. In this section, we discuss several other formalisms that address the same problem and point out some of the differences between these formalisms and the one of untangling.

One can use state space techniques when it comes to the automatic analysis and verification of concurrent systems. Rather than performing analysis directly on a net system, state space techniques explore an induced representation, called a *transition system*. In a nutshell, a transition system induced by a net system S is a directed graph with all the states reachable from S as its nodes and directed edges that encode possible state transitions, i.e., there is an edge in the graph from state H to state H' whenever there is a step in S that leads from H to H'. Unfortunately, state space techniques suffer from the well-known state space explosion problem. In the worst case, all nodes of the induced transition system must be explored to accomplish the envisioned analysis task. This often turns out to be infeasible due to the large size of the transition system.

An *unfolding* of a concurrent system is a mathematical structure (often infinite) that explicitly represents causality and concurrency relations on operations of the system, as well as all the points of choice between qualitatively different behaviors of the system. Unfoldings are special acyclic graphs that describe states that are reachable from a net system as combinations of nodes rather than dedicating every node to a single state. In [McM92, McM95], Ken McMillan observed that one can represent all the information contained in an unfolding in its truncated finite initial part, which is often called a *finite prefix* of the unfolding. If adequate construction algorithms are employed, the size of a finite prefix of the unfolding of a concurrent system is never larger—and in practice is by far smaller—than the size of the transition system induced by the same net system. A classic condition for prefix truncation is *completeness*, where a prefix induced by a concurrent system is complete if it encodes all the steps (reachable states and possible state transitions) in the system. Completeness of prefixes helps to 'unveil' some of the behavioral properties of concurrent systems by allowing their validation in time that is linear in the size of a given finite complete prefix, e.g., any executability problem [ERV02, EH08] can be solved efficiently using a finite complete prefix of the net system, while keeping other properties sufficiently 'concealed', e.g., the problem of deciding deadlock freedom [MR97] of a net system using a finite complete prefix of this system is known to be NP-complete [McM95]. One can rely on special unfolding truncation criteria in order to unveil other behavioral properties in finite parts of unfoldings. In [EH08], truncation criteria that address executability, repeated executability, livelock, and properties expressible in linear temporal logic, are systematized. The strong coupling of behavioral properties of concurrent systems with different configurations of finite prefixes of their unfoldings is mainly due to implicit dependencies between transition occurrences which stem from unfolding truncations.

Merged processes, which are studied in [KKKV06, KM11, RSK13], are compressed representations of finite prefixes of unfoldings with most of the discussed above advantages and disadvantages of finite prefixes of unfoldings. The compression in merged processes is achieved by addressing such causes of the state space explosion as sequences of choices and unsafeness. Many analysis results that were initially proposed for finite prefixes of unfoldings can be transferred to merged processes.

In [Des00], Jörg Desel suggests using processes of net systems for analysis of concurrent systems. As net systems can induce infinite processes, Desel proposes several termination conditions to limit the size of processes. One option is to use a bound in terms of the number of created events and conditions or in terms of construction time. Unfortunately, this simple method does not guarantee that all the behavior of the net system is explored before triggering the termination condition. As a consequence, any analysis on the set of all—discovered in such a way—processes cannot be considered exhaustive. Alternatively, if one decides to distinguish between internal and external actions of a concurrent system, one can further decide to terminate construction of a process of the corresponding net system if the last appended event describes an occurrence of a transition that leads to a state that does not enable any internal action (this corresponds to a situation when the concurrent system waits for an external trigger in order to progress). Unfortunately, given a net system, it is unknown beforehand whether this termination condition will eventually be fulfilled in the system and, hence, it only fits very specific analysis situations, or must be further constrained to guarantee termination. Finally, Desel proposes to terminate construction of a process if its last appended event corresponds to an occurrence of a transition that leads to a state which was already observed during construction of this process or some other process of the same net system. If the net system is bounded, this condition guarantees termination. The same and stronger termination conditions were previously used for truncating infinite unfoldings of net systems [McM95, ERV02]. In fact, every finite complete prefix of the unfolding of a net system S is a compact representation of a set of all processes of S truncated according to this, or a stronger, termination condition and exhaustively merged based on their isomorphic histories.

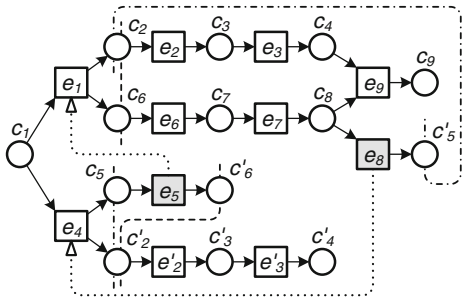

Fig. 10. A finite complete prefix of the unfolding of the net system in Fig. 1a

Every process in an untangling can be seen as a *scenario*, i.e., a behavioral requirement that describes interdependencies between actions that can occur in the system. In [Fah10, Rei13], scenarios are proposed as a feature to specify behavior of a system. In this context, our work suggests a technique to decompose a given system into a set of its scenarios that collectively specify all and only the behavior of the system.

Both unfoldings and untanglings are acyclic graphs that encode causality and concurrency relations on transition occurrences. Thus, fundamentally, untanglings address the state space explosion problem to a similar extent as unfoldings. Furthermore, similar to a finite prefix of the unfolding of a net system S, a representative untangling of S is a finite model that describes all the steps in S, i.e., all the transition occurrences that are possible from all the states reachable from S. However, in contrast to a finite prefix of the unfolding of S, a representative untangling of S suggests dedicated scopes for analysis of properties that target individual runs in S, i.e., every run in S is represented by some process in a representative untangling of S. This result is due to a special termination condition for process construction that limits the size of processes in representative untanglings, cf. Definition 4.3. Several benefits of using representative untanglings for describing concurrent systems are explored in the next section.

7. Evaluation

This section ascertains the value of representative untanglings. It illustrates their unique properties, as compared to artifacts studied by the theory of unfoldings, i.e., *finite complete prefixes* (of unfoldings) of net systems, and demonstrates practical benefits that stem from the use of representative untanglings. The next section contrasts representative untanglings with finite complete prefixes by comparing running times and output artifacts of the reduction-based untangling algorithm (see Sect. 5.3) and the finite complete prefix algorithm for 1-bounded net systems proposed in [ERV02]. In the subsequent sections, we propose several methods for deciding various behavioral properties of net systems using their representative untanglings and compare computational complexities and running times of these methods with computational complexities and running times of their counterpart techniques that rely on the use of finite complete prefixes. In particular, the behavioral properties addressed are executability, deadlock freedom, and mutual exclusiveness. The results of this evaluation confirm the feasibility and benefits of using representative untanglings in practice.

The evaluation was carried on a real-life collection of 448 net systems (more precisely workflow nets) that model processes from the domains of finance and telecommunication. They were selected from the BIT Library [FFK+11]. Net systems that are unsafe were filtering out; a net system is *safe* iff it is 1-bounded. All the experiments were implemented in Java and performed on a laptop with Intel Core i7-3667U CPU 2.00 GHz, 8 GB of memory, running Windows 7 and Sun JVM 1.7 (with standard allocation of memory).

7.1. Size and construction time

In Sect. 6, we pointed out that representative untanglings have commonalities with finite complete prefixes of unfoldings. Similar to a representative untangling, a finite complete prefix of a net system is a finite acyclic net that encodes the behavior of the corresponding net system. Figure 10 shows a finite complete prefix of the net system S in Fig. 1a. This and all the finite complete prefixes constructed in our experiments are generated using the total adequate order for safe systems proposed in [ERV02]. In the prefix, event e_5 is a *cutoff* event (highlighted with grey background). Event e_5 and its *corresponding* event e_1 (the relation is denoted by a dotted arrow) are

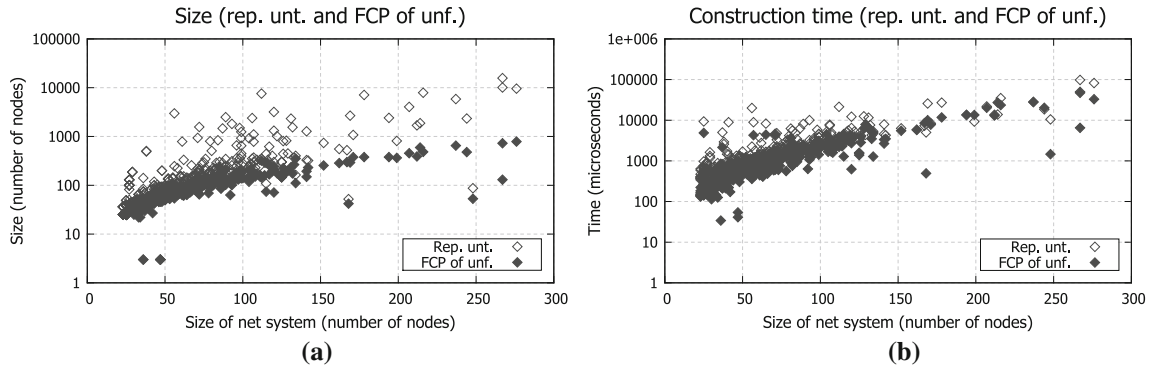

Fig. 11. Experimental results of computing finite complete prefixes and representative untanglings of 448 net systems taken from practice: **a** number of nodes in constructed artifacts and **b** construction time

associated with equivalent reachable markings that put one token at place p_2, one token at place p_6, and no tokens elsewhere. In the figure, these markings are encoded as sets of conditions that overlap with dashed lines; refer to [McM95, ERV02] for more information on how events of unfoldings are associated with reachable markings of the corresponding net system. Intuitively, the finite complete prefix in Fig. 10 can be constructed starting from the initial marking of S, i.e., from condition c_1 that encodes the marking with the only token at place p_1, then iteratively appending events that encode occurrences of transitions (in the order that they can occur in the system), and terminating the construction at cutoff event e_5 as information on transition occurrences that follow e_5 is already contained in the part of the prefix that follows its corresponding event e_1. Similarly, cutoff event e_8 and its corresponding event e_4 encode equivalent markings that put one token at place p_2, one token at place p_5, and no tokens elsewhere; the respective sets of conditions are denoted by dash-dotted lines in the figure.

Both the baseline and the reduction-based untangling techniques have been implemented and are publicly available as part of the `jBPT` initiative [PW13].[10] Using this implementation, we conducted an experiment to assess performance of the reduction-based untangling technique in terms of construction times and sizes of the produced outputs. To eliminate load times from the measurements, each experiment was executed six times, and we recorded average times of the second to sixth executions.

Figure 11 summarizes the results of this test, which can be downloaded and reproduced.[11] Figure 11a reports on the sizes (as measured by the number of nodes) of constructed representative untanglings and finite complete prefixes of the 448 net systems. The non-shaded diamonds encode sizes of representative untanglings, while the shaded ones specify sizes of finite complete prefixes. The sizes of net systems from the collection range from 23 to 276 nodes, with the average size being 63.75 nodes. The average size of a finite complete prefix is 91.24 nodes, whereas the average size of a representative untangling is 371.1 nodes. On average, constructed finite complete prefixes and representative untanglings are 1.3 and 3.54 times larger than the corresponding net systems, respectively. The size of a finite complete prefix of a net system is on average 67.39 % of the size of a representative untangling of the same net system. Furthermore, we observed that on average, constructed representative untanglings contain 6.22 processes. Similarly, Fig. 11b reports on the times of constructing finite complete prefixes and representative untanglings. According to our tests, the average time spent on constructing a finite complete prefix is 1.57 ms, while the average time spent on constructing a representative untangling is 2.83 ms.[12]

7.2. Executability

Finite complete prefixes have proven their advantage when verifying behavioral properties of concurrent systems. For example, given a net system S, the *executability problem*, which consists of deciding whether there exists an

[10] http://code.google.com/p/jbpt.
[11] http://code.google.com/p/jbpt/wiki/UntanglingsExperiment.
[12] Uma is used to construct finite complete prefixes of unfoldings: http://service-technology.org/tools/uma.

occurrence sequence in S of which every transition out of a given set of transitions is an element, can be solved in a time that is linear in the size of a finite complete prefix of S [EH08]. More precisely, if a transition t is an element of some occurrence sequence in S, then a finite complete prefix of S contains an event that describes an occurrence of t. For example, using events of the finite complete prefix in Fig. 10 and the above described rationale, one can trivially conclude that for every transition t of the net system in Fig. 1a there exists an occurrence sequence in this net system of which t is an element.

The executability problem is a fundamental problem in concurrency theory as many other problems can be reduced to it, e.g., a solution to the executability problem can help deciding *reachability* and *safety* [EH08, GW93]. The executability problem can be formulated as follows.

Definition 7.1 (*Executability*) A net system $S := (N, M)$, $N := (P, T, F)$, *can execute some transition* in $U \subseteq T$ iff there exist an occurrence sequence σ in S and a transition $t \in U$ such that t is an element of σ. ⌟

One can use representative untanglings to solve the executability problem.

Lemma 7.2 (Executability) *Let* Π *be a representative untangling of a net system* $S := (N, M)$, $N := (P, T, F)$. *Then*, S *can execute some transition in* $U \subseteq T$ iff *there exist a process* $\pi \in \Pi$, $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, *and an event* $e \in E$ *such that* $\rho(e) \in U$. ⌟

Proof We prove each direction of the statement separately.

($\Rightarrow$) Assume that there exists an occurrence sequence that contains $t \in U$ as an element. Then, there exists a run δ in S that contains a step $\chi := (H, t, H')$. According to Definition 4.1, there exists a process $\pi \in \Pi$, $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, that represents every step of δ. Then, π represents χ and, thus, contains an event $e \in E$ such that $\rho(e) \in U$.

($\Leftarrow$) Assume that there exist a process $\pi \in \Pi$, $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, and an event $e \in E$ such that $\rho(e) \in U$. According to Lemma 1 in [Des00], there is a run in S which contains a step that describes an occurrence of $\rho(e)$. Thus, there is an occurrence sequence in S which contains $\rho(e)$ as an element. ■

Therefore, similar to finite complete prefixes, representative untanglings can be employed to solve the executability problem efficiently.

Proposition 7.3 (Executability) *Given a representative untangling* Π *of a net system* $S := (N, M)$, $N := (P, T, F)$, *the following problem can be solved in linear time on* Π*: To decide if* S *can execute some transition in* $U \subseteq T$. ⌟

The proof of Proposition 7.3 is a direct consequence of Lemma 7.2. Given a net system S and a subset of its transitions U, one can verify whether S can execute some transition in U by visiting events of a representative untangling of S (such that each event is visited at most once) until none of the visited events describes an occurrence of some transition in U. The linear time complexity reported in Proposition 7.3 assumes that the set membership operation can be performed in constant time, e.g., as a search in a hash table. For example, using events of the five processes in Figs. 2 and 4, one can decide that the net system S in Fig. 1a can execute some transition in every non-empty subset of transitions of S; recall that the five processes in Figs. 2 and 4 constitute a representative untangling of S.

Figure 12 reports average times spent when solving executability problems on representative untanglings (rep. unt.) and on finite complete prefixes of unfoldings (FCP of unf.) of the 448 net systems under evaluation. Fig. 12a, Fig. 12b, and Fig. 12c, show the average times spent on solving executability problems on input sets that consist of one, six, and eleven transitions, respectively; note that the smallest net in the collection has eleven transitions. For a given number n of transitions and a given net system S, n transitions of S are randomly selected and the time required for solving the executability problem for these transitions is computed. The average value for this combination of number n and net system S is determined by taking the average of 1,000 repetitions of this procedure. Figure 12d shows the average times spent on solving the executability problems for all 448 net systems in the collection for different sizes of input sets of transitions. Again, for a given number n of transitions and a net system S, n transitions of a net system are randomly selected and the time required for solving the executability problem for these transitions is computed. The average value for this number n is obtained by taking the average of 448 values of time spent on executing the above described procedure (one value for every net system in the collection).

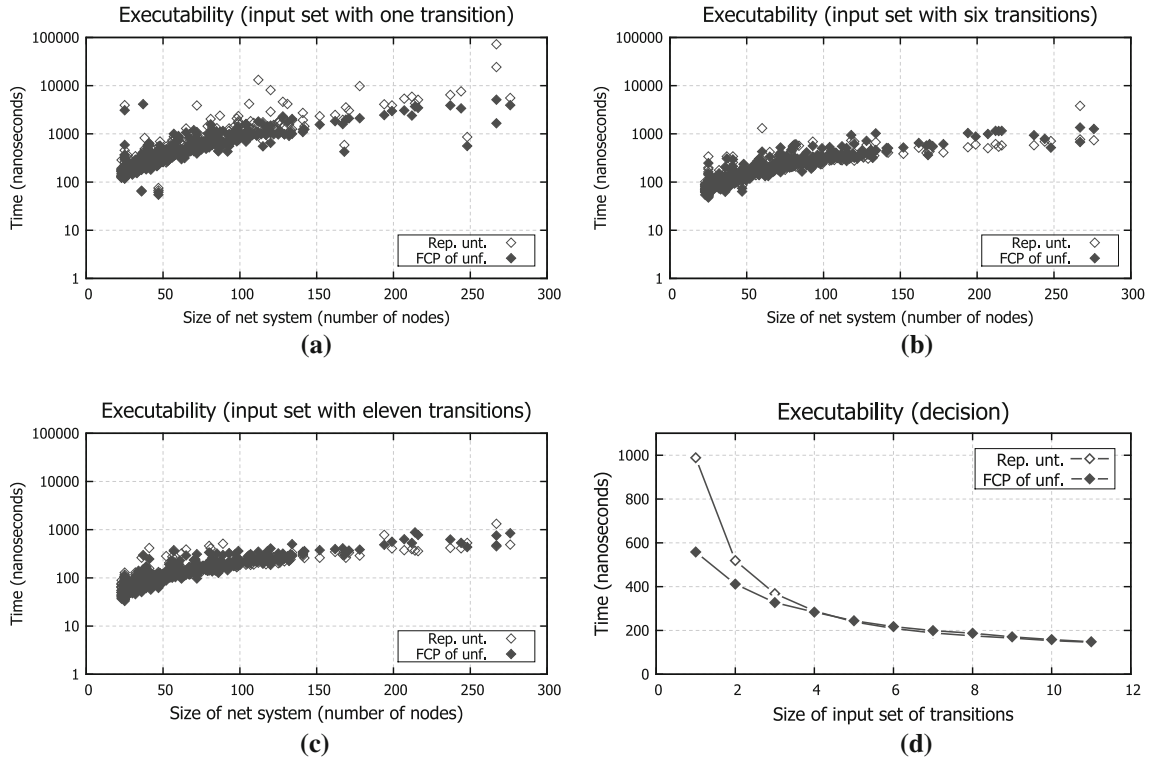

Fig. 12. Average times of deciding executability using finite complete prefixes and representative untanglings: **a** with an input set that contains 1 transition, **b** with an input set that contains 6 transitions, **c** with an input set that contains 11 transitions, and **d** over all net systems in the collection

Because solutions to the executability problem have linear time complexity, both on representative untanglings and on finite complete prefixes, and considering that representative untanglings are never smaller than finite complete prefixes (refer to Sect. 7.1), it does not come as a surprise that the times spent on deciding executability based on finite complete prefixes are often smaller than those on representative untanglings. However, the measurements also show that with the growth of the size of input sets of transitions this difference becomes negligible. We observed that certain decisions based on representative untanglings were carried out faster than those based on finite complete prefixes. These are the cases when visited parts of the untanglings turned out to be smaller than visited parts of the respective finite complete prefixes.

Given a net system S and a subset of its transitions U, one can check if S can execute some transition in U by verifying whether U overlaps with the set of all transitions that are elements of all occurrence sequences in S. Clearly, this set consists only of those transitions that are described by at least one event of a representative untangling of S or, similarly, by at least one event of a finite complete prefix of S. For the net systems under evaluation, on average such sets were constructed in 18 and 33 μs using finite complete prefixes and representative untanglings, respectively.

The performed tests hint at the fact that differences between running times of techniques for solving decision problems of the same time complexity which operate on representative untanglings and finite complete prefixes can often be negligible in practice. Further experiments are required to justify this claim.

In [PLtH14], we introduced the *total executability problem*. Given a net system S and a subset of its transitions U, the total executability problem consists of deciding whether there exists an occurrence sequence in S of which every transition in U is an element. We showed that this problem can be solved efficiently using a representative untangling of S and hinted at the fact that the proposed solution can be of help when designing automated retrieval systems that query for behavioral information in collections of process models.

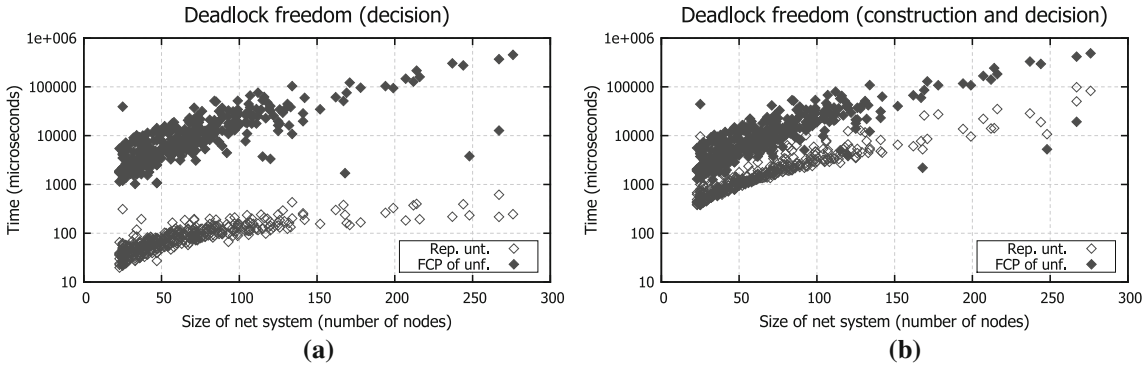

Fig. 13. Average times of deciding deadlock freedom using finite complete prefixes and representative untanglings: **a** decision times and **b** times spent to construct auxiliary artifacts plus decision times

7.3. Deadlock freedom

Another important problem in concurrency theory is the *deadlock freedom problem*. It deals with answering the question whether there exists a deadlock marking reachable from a given net system.

Definition 7.4 (*Deadlock freedom*)
A marking M of a net $N := (P, T, F)$ is a *deadlock* marking iff there exists no transition $t \in T$ that is enabled in (N, M). A net system $S := (N, M)$ is *deadlock free* iff there exists no deadlock marking of N reachable from S. ⌟

The problem of deciding deadlock freedom of a bounded net system S is known to be NP-complete on a finite complete prefix of the unfolding of S [McM95, MR97]. Alternatively, one can use a representative untangling of S to decide deadlock freedom.

Lemma 7.5 (Deadlock freedom)
Let Π *be a representative untangling of a net system* $S := (N, M)$. *S is deadlock free iff for every process* $\pi := (N_\pi, \rho)$ *in* Π *it holds that* $(C, \rho\,|_C)$, *where* $C := Max(N_\pi)$, *is not a deadlock marking of* N. ⌟

Proof We prove each direction of the statement separately.

($\Rightarrow$) If S is deadlock free then every marking that is reachable from S is not a deadlock marking of N. Because every maximal cut of every process in Π corresponds to some marking that is reachable from S, refer to Theorem 2.9 for details, it trivially holds that all maximal cuts of all processes in Π do not correspond to deadlock markings.

($\Leftarrow$) Assume that for every process $\pi := (N_\pi, \rho)$ in Π it holds that $(C, \rho\,|_C)$, where $C := Max(N_\pi)$, is not a deadlock marking of N, but S is not deadlock free. Then, there exists a deadlock marking M' of N reachable from S via some run δ in S. There also exists a process $\pi := (N_\pi, \rho)$ in Π that represents δ, $N_\pi := (B, E, G)$. Because π represents every step of δ, it holds that there exists a cut C' of N_π that corresponds to marking M'. If C' is not the maximal cut of N_π, then M' is not a deadlock marking of N (M' enables some transition $\rho(e)$, where $e \in E$ and $\bullet e \subseteq C'$). Then, it holds that C' is the maximal cut of N_π, which leads to a contradiction. ■

It is easy to see that deadlock freedom of a net system S can be decided efficiently on a representative untangling of S.

Proposition 7.6 (Deadlock freedom) *Given a representative untangling* Π *of a net system* S, *the following problem can be solved in linear time on* Π*: To decide if* S *is deadlock free.* ⌟

The proof of Proposition 7.6 is a direct consequence of Lemma 7.5. Indeed, to check deadlock freedom of S it suffices to test if the maximal cut of some process of a representative untangling of S induces a deadlock marking. This check can be accomplished during a single scan through all conditions of the representative untangling. For example, one can decide that the net system S in Fig. 1a is not deadlock free based on the maximal cut $D_3 := \{c_9\}$ of the causal net in Fig. 4a. This cut induces a deadlock marking of the net in Fig. 1a which puts one token at place p_9 and no tokens elsewhere; recall that the process in Fig. 4a is one out of the five processes shown in Figs. 2 and 4, which together constitute a representative untangling of S.

Figure 13 reports average times (on a logarithmic scale) of deciding deadlock freedom for each of the 448 net systems under evaluation based on representative untanglings (rep. unt.) and finite complete prefixes of unfoldings (FCP of unf.). The checks on finite complete prefixes follow the linear algebraic approach proposed in [MR97]. The implementation relies on LpSolve ver. 5.5.2.0 when solving systems of inequalities.[13] Figure 13a suggests the times spent to carry out the decisions considering that representative untanglings and finite complete prefixes are already constructed, whereas Fig. 13b shows overall times spent to decide deadlock freedom, i.e., the times spent on constructing auxiliary artifacts (either representative untanglings or finite complete prefixes) plus the times spent to carry out the decisions based on the subsequent checks on the respective artifacts. To eliminate load times from the measurements, each deadlock freedom problem was solved six times, and we recorded average times spent on the second to sixth problem solving exercise. On average, it took 89.99 μs and 15.66 ms to decide deadlock freedom based on representative untanglings and finite complete prefixes, respectively. When taking the times spent on constructing the auxiliary artifacts into account, the average times of checking deadlock freedom increase to 2.92 and 17.23 ms, respectively.

In most of the cases, the checks based on representative untanglings outperform by far the checks that rely on the use of finite complete prefixes. The significant difference between decision times on representative untanglings and finite complete prefixes, refer to Fig. 13a, confirms that most of the value when using representative untanglings comes from their reuse for checking multiple behavioral properties, e.g., once a representative untangling of a net system is constructed it can be used to decide deadlock freedom but also to solve an arbitrary number of instances of the executability problem.

7.4. Mutual exclusiveness

Together with the studies of fundamental problems in concurrency theory, such as the executability problem and the deadlock freedom problem, much of attention of researchers has been devoted to decidability issues for specification languages in which a large number of behavioral properties can be expressed. In particular, given a model of a system, *model checking* studies the problem of automatically checking whether the model satisfies a behavioral property specified in a language that often takes the form of a temporal logic. *Linear time logics* [Pnu77] for Petri nets form a class of temporal logics that are usually interpreted on maximal computations. One behavioral phenomenon that can be conveniently expressed in terms of a linear time logic is *mutual exclusiveness* of events, or the *conflict* relation, which is one of the fundamental behavioral relations as classified by the 4C spectrum [PWC+14]. In general, two events are mutually exclusive if they cannot occur together. For instance, two events of observing heads or tails when tossing a coin are mutually exclusive. The phenomenon of mutual exclusiveness of events can be adapted to events that describe occurrences of transitions and can consequently be characterized on transitions of Petri nets as follows.

Definition 7.7 (*Mutual exclusiveness*)
Let $S := (N, M)$, $N := (P, T, F)$, be a net system. Transitions $t_1 \in T$ and $t_2 \in T$ are *mutually exclusive* in S iff for every occurrence sequence σ in S it holds that either t_1 or t_2 is not an element of σ. ⌟

For a transition $t \in T$ it trivially holds that t is mutually exclusive with itself if and only if t is not an element of any occurrence sequence in S. Clearly, mutual exclusiveness is a symmetric relation.

One can rely on finite complete prefixes when model checking behavioral properties of corresponding net systems, e.g., mutual exclusiveness of transitions. Given a net system S and a linear temporal logic formula, a common approach is to augment S with a *Büchi tester* [EH08] (a new component added to S) in order to observe and register occurrences of certain transitions in a finite complete prefix of the unfolding of the augmented net system. These transition occurrences signify violations of the property specified in the formula. Thus, when model checking mutual exclusiveness of two transitions of a given net system a fresh finite complete prefix of an augmented version of the system must be constructed. For instance, if one wants to check mutual exclusiveness of every distinct pair of transitions in the net system in Fig. 1a, one needs to construct and analyze 36 finite complete prefixes! Additionally, one finite complete prefix must be constructed to check whether each transition of the system is related with itself via mutual exclusiveness. Note that one can check whether a transition is an element of some occurrence sequence in a net system by using the solution to the executability problem proposed in [EH08].

In contrast, one can rely on a single representative untangling of a net system to *efficiently* decide mutual exclusiveness for all pairs of transitions of this net system.

[13] http://lpsolve.sourceforge.net/.

Lemma 7.8 (Mutual exclusiveness—untanglings)
Let Π *be a representative untangling of a net system* $S := (N, M)$, $N := (P, T, F)$. *Transitions* $t_1 \in T$ *and* $t_2 \in T$ *are* mutually exclusive *in* S *iff for every process* $\pi \in \Pi$, $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, *and for every two events* $e_1 \in E$ *and* $e_2 \in E$ *it holds that either* $\rho(e_1) \neq t_1$ *or* $\rho(e_2) \neq t_2$. ⌟

Proof We prove each direction of the statement separately.

($\Rightarrow$) Assume that for every occurrence sequence σ in S it holds that either t_1 or t_2 is not an element of σ, but there exist a process $\pi \in \Pi$, $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, and events $e_1, e_2 \in E$ such that $\rho(e_1) = t_1$ and $\rho(e_2) = t_2$. Then, according to Lemma 1 in [Des00], there exists an occurrence sequence in S of which both t_1 and t_2 are elements.

($\Leftarrow$) Assume that for every process $\pi \in \Pi$, $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, and for every two events $e_1 \in E$ and $e_2 \in E$ it holds that either $\rho(e_1) \neq t_1$ or $\rho(e_2) \neq t_2$. According to Definition 4.1, every run in S is represented by some process $\pi \in \Pi$. Then, there exists no run and, thus, no occurrence sequence σ in S such that t_1 and t_2 are elements of σ. ■

Given a net system S, one can efficiently decide whether two transitions of S are mutually exclusive using a representative untangling of S.

Proposition 7.9 (Mutual exclusiveness)
Given a representative untangling Π *of a net system* $S := (N, M)$, $N := (P, T, F)$, *the following problem can be solved in linear time on* Π*: To decide if transitions* $t_1 \in T$ *and* $t_2 \in T$ *are mutually exclusive in* S. ⌟

The proof of Proposition 7.9 is a direct consequence of Lemma 7.8. Indeed, given two transitions t_1 and t_2 of a net system S, one can verify whether they are mutually exclusive in S by checking if a representative untangling of S contains a process with events that describe occurrences of t_1 and t_2. For example, one can use the processes in Figs. 2 and 4 to verify that transitions t_1 and t_4 are the only two mutually exclusive transitions in the net system in Fig. 1a. Indeed, none of the five processes in the two figures contains an event that describes an occurrence of transition t_1 and an event that describes an occurrence of transition t_4.

Next, we compare two techniques for computing the mutual exclusiveness relation on transitions of a net system, one using a representative untangling of the net system and the other relying on the study of a finite complete prefix of the net system. To have a fair comparison, instead of adopting the model checking approach proposed in [EH08], which as discussed above requires us to construct a fresh finite complete prefix for each pair of distinct transitions of the net system, we reduce the problem to the *covering problem* [Rac78], which can be solved for all pairs of transitions on a single finite complete prefix of a specially augmented version of the net system. More precisely, prior to constructing a finite complete prefix of a net system, it gets transformed via *injections of guards*. An injection of a guard for a transition of a net system results in a structural transformation of the net system. This structural transformation was used in [PWC+14] to reduce the problem of computing *can conflict* and *can co-occur* relations of the 4C spectrum to the *reachability problem* [Hac75], refer to Definition 5.1 and Proposition 5.2 in [PWC+14]. Finally, given a net system $S := (N, M_0)$, $N := (P, T, F)$, and a marking $M := (K, \mu)$, the covering problem consists of deciding whether there exists a marking $M' := (K', \mu')$ that is reachable from S and *covers* M, where M' covers M iff for every place $p \in P$ it holds that μ' assigns at least as many tokens to p as μ.

Figure 14a shows the net system S' obtained from the net system S in Fig. 1a via injections of guards for transitions t_4 and t_8. In the figure, fresh places and transitions added to S are highlighted with grey background. Transitions $\hat{t}_4$ and $\hat{t}_8$ are the *guard* transitions for t_4 and t_8, respectively. Places q and s are the *control* places for t_4 and t_8, respectively. A guard transition t' for a transition t can occur instead of t and implements the effect of an occurrence of t, i.e., the preset and postset of t' contains the preset and postset of t, respectively. For every run δ in S' it holds that t' can occur at most once in δ. This is ensured by one fresh token assigned to the special fresh input place of t', e.g., see places p and r in Fig. 14a. An occurrence of t' 'allows' subsequent occurrences of t by producing a token at the control place for t.

Clearly, different orderings of injections of guards in S for all transitions from a given set of transitions of S lead to the same resulting net system S'. Moreover, occurrence sequences in S and S' are closely related. In particular, one can construct an occurrence sequence in S from an occurrence sequence δ in S' by replacing every occurrence of every guard transition in δ with an occurrence of a transition that this guard was introduced for. Similarly, one can construct an occurrence sequence in S' from an occurrence sequence δ in S by replacing all first occurrences of transitions for which guard transitions in S were introduced with the respective guard transitions.

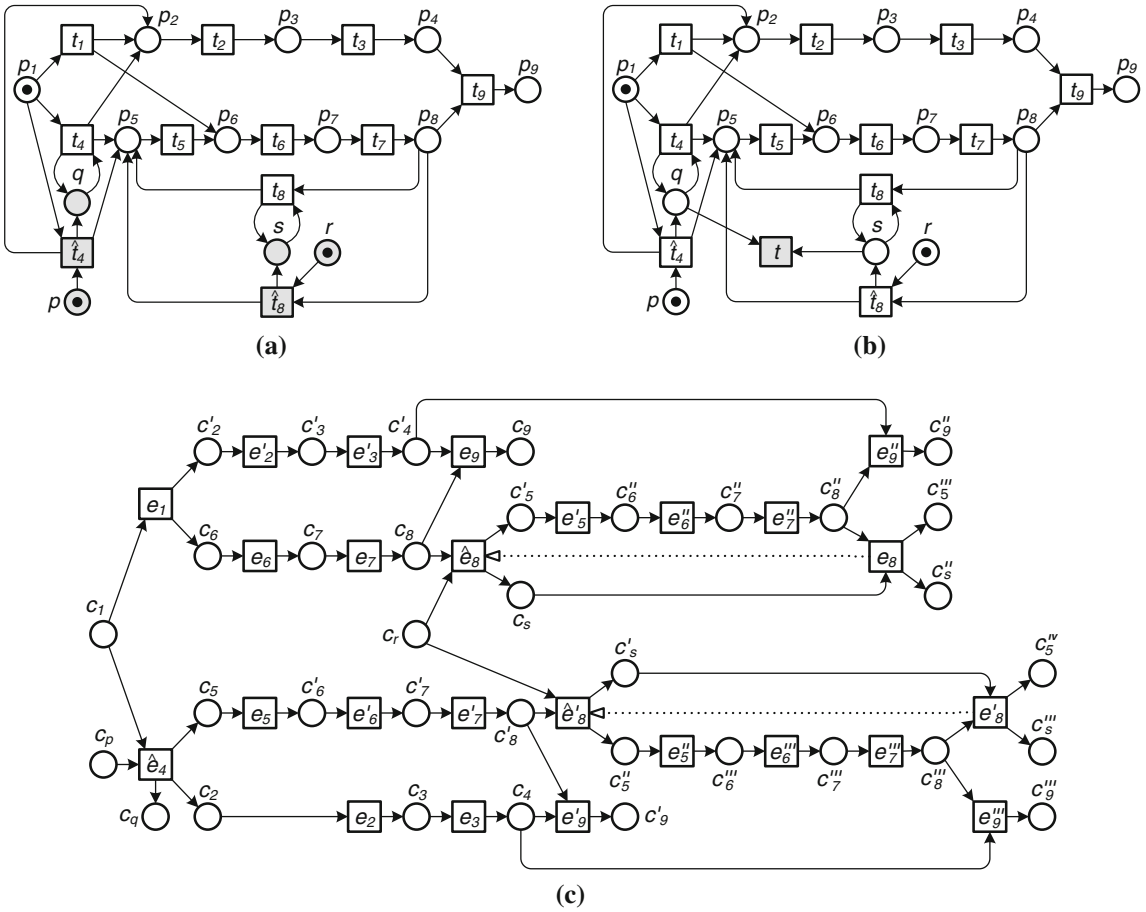

Fig. 14. **a** A net system obtained from the net system in Fig. 1a via injections of guards for transitions t_4 and t_8, **b** a net system obtained from the net system in **a** by adding control transition t for places q and s, and **c** a finite complete prefix of the net system in **a**

For example, $t_4\ t_5\ t_6\ t_7\ t_8\ t_5\ t_6\ t_7\ t_8$ and $\hat{t}_4\ t_5\ t_6\ t_7\ \hat{t}_8\ t_5\ t_6\ t_7\ t_8$ are two such related occurrence sequences in the net systems in Fig. 1a and Fig.14a, respectively.

Lemma 7.10 (Mutual exclusiveness—the covering problem) *Let* $S := (N, M)$, $N := (P, T, F)$, *be a net system and let* $S' := (N', M')$, $N' := (P', T', F')$, *be a net system obtained via injections of guards in* S, *see Definition* 5.1 *in* [PWC+14], *for all transitions in* T. *Let* $t_1 \in T$ *and* $t_2 \in T$ *be transitions of* N, *where* $p_1 \in P'$ *and* $p_2 \in P'$ *are the control places for* t_1 *and* t_2, *respectively. Transitions* t_1 *and* t_2 *are mutually exclusive in* S iff *for every marking* M'' *that is reachable from* S' *it holds that* M'' *does not cover a marking of* N' *that induces the state* $[p_1\ p_2]$. ⌟

Indeed, if t_1 and t_2 are mutually exclusive then there exists no occurrence sequence in S of which t_1 and t_2 are elements. Hence, there exists no occurrence sequence in S' of which the guard transitions for t_1 and t_2 are elements. Then, one cannot reach a marking from S' that covers a marking that puts one token at place p_1, one token at place p_2, and no tokens elsewhere. Moreover, if one cannot reach a marking from S' that assigns tokens to p_1 and p_2 then there exists no occurrence sequence in S' that contains occurrences of the respective guard transitions. Consequently, one cannot find an occurrence sequence in S' from which it is possible to construct an occurrence sequence in S of which both t_1 and t_2 are elements.

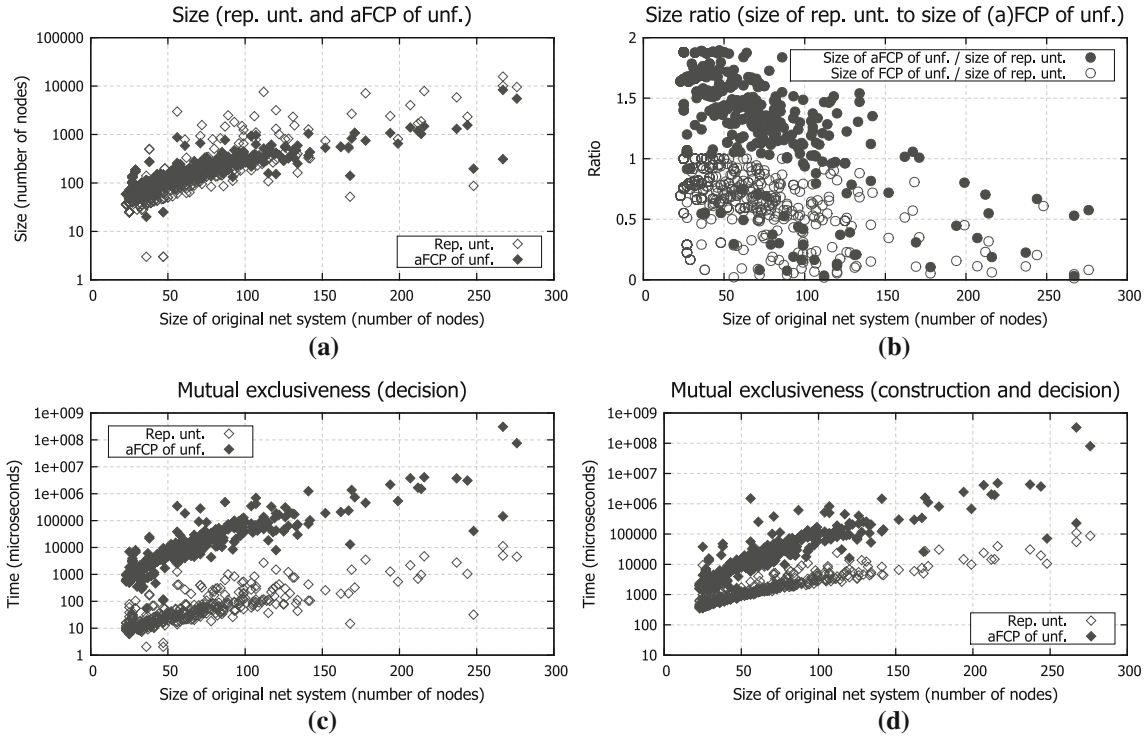

Fig. 15. a Sizes of representative untanglings of the original net systems and of finite complete prefixes of the augmented net systems obtained after injecting guards for all transitions, **b** ratios between sizes of finite complete prefixes and representative untanglings of original and augmented net systems, and average times of deciding mutual exclusiveness on all transitions of net systems using representative untanglings and finite complete prefixes: **c** decision times and **d** times spent on constructing auxiliary artifacts plus decision times

Note that the *covering problem* is decidable [Rac78]. The procedure for deciding the covering problem proposed in [Rac78] operates in space exponential in the size of the input net system. Alternatively, one can use finite complete prefixes when deciding the covering problem. The procedures for deciding the covering problem of net systems on their finite complete prefixes operate in times that are exponential in the sizes of the prefixes [Kho03]. However, these times are often acceptable in practice.

We performed a number of measurements to compare two techniques for computing the mutual exclusiveness relation on transitions of net systems in the collection under evaluation, one that uses representative untanglings (as per Lemma 7.8) and the other that relies on the solution to the covering problem on finite complete prefixes (as per Lemma 7.10), cf. [Kho03] for details.

Figure 15a reports on sizes (as measured by the number of nodes) of constructed representative untanglings (rep. unt.) of the original net systems (these sizes are the same as those reported in Fig. 11a) and finite complete prefixes of the augmented net systems obtained after injections of guards for all transitions (aFCP of unf.); again, note the use of a logarithmic scale. Observe that injections of guards preserve safeness of systems, which allows us to keep using the total adequate order for safe systems proposed in [ERV02] when constructing the prefixes. Similar to Fig. 11a, the non-shaded diamonds encode sizes of representative untanglings, whereas the shaded ones give sizes of finite complete prefixes.

Though construction of a finite complete prefix of a bounded net system is considered to be efficient in practice, it is of an exponential nature and can easily explode [KKKV06], both in terms of computation time and the size of the resulting prefix. For instance, Fig. 14c shows a finite complete prefix of the net system in Fig. 14a; again the dotted arrows encode relations between cutoffs and corresponding events of the prefix. The prefix is composed of 60 nodes (conditions and events). Note that the finite complete prefix of the original net system (before injections of guards) is shown in Fig. 10 and consists of 25 nodes.

For every net system S in the collection, Fig. 15b reports on the ratio of the size of the finite complete prefix of S to the size of the representative untangling of S. The non-shaded circles encode ratios of sizes of finite complete prefixes of original net systems (FCP of unf.) to sizes of their representative untanglings (rep. unt.). For every constructed representative untangling, it holds that it is either of the same size or is larger than the constructed finite complete prefix of the corresponding net system, i.e., the respective ratio is less than or equal to one. The situation changes once net systems get augmented via injections of guards (a guard is injected for every transition of every system in the collection). The shaded circles in Fig. 15b report on ratios of sizes of finite complete prefixes of augmented net systems (aFCP of unf.) to sizes of representative untanglings of original net systems (rep. unt.). In most of the cases finite complete prefixes turn to be larger in size than their respective representative untanglings. On average, we observed that the size of the constructed representative untangling of the original net system is 70 % of the size of the constructed finite complete prefix of the corresponding augmented net system.

Figure 15c reports average times (on a logarithmic scale) of computing the mutual exclusiveness relations on transitions for the 448 net systems under evaluation using representative untanglings (as per Lemma 7.8) and finite complete prefixes of unfoldings of augmented net systems (as per Lemma 7.10). On average, a mutual exclusiveness relation was computed in 169 μs and 931 ms using representative untanglings (rep. unt.) and finite complete prefixes (aFCP of unf.), respectively. Similarly, Fig. 15d reports overall times spent on constructing auxiliary artifacts, either representative untanglings or finite complete prefixes of augmented net systems, and subsequently computing the mutual exclusiveness relation using the respective artifacts. On average, the observed overall time of computing the mutual exclusiveness relation for a net system S, i.e., deciding mutual exclusiveness for every pair of transitions of S, is 3 and 1,027 ms using its representative untangling and the finite complete prefix of its augmented version, respectively.

In [McM92], Ken McMillan proposes an alternative method for solving the covering problem for a safe net system, which relies on the use of a finite complete prefix of the system. If one wants to verify whether a given set of places of a net system can ever be marked simultaneously at some of its reachable markings, one first needs to add a fresh control transition with the given places as inputs and without output places; note that in order to use this method we need to violate the requirement of every net being T-restricted. Consequently, if a finite complete prefix of the resulting net system contains an event that describes an occurrence of the control transition, then from the original net system one can reach a marking that covers a marking that puts one token at every place in the given set of places and no tokens at the other places. Hence, one can organize the computation of the mutual exclusiveness relation by augmenting a net system obtained after injections of guards in a given net system further by adding one fresh transition for every two distinct injected control places, such that these two places are its only inputs. Then, one can decide mutual exclusiveness for every two distinct transitions of the given net system by registering events that describe occurrences of the control transitions during a single traversal over all the events in a finite complete prefix of the resulting net system. For example, Fig. 14b shows the net system obtained from the net system in Fig. 14a by adding control transition t (highlighted with grey background) for the control places q and s. One can decide mutual exclusiveness of transitions t_4 and t_8 in the net system in Fig. 1a based on the presence of events that describe occurrences of t in a finite complete prefix of the unfolding of the net system in Fig. 14b. Though the described procedure is efficient, finite complete prefixes of net systems after injections of guards and subsequent insertions of control transitions tend to explode in size. For instance, the finite complete prefix of net system S' obtained from net system S in Fig. 1a after injecting guards for all transitions of S is composed of 90 nodes, whereas the finite complete prefix of net system S'' obtained from S' by adding control transitions for each pair of distinct control places is composed of 193 nodes. For the collection of 448 net systems under evaluation, we observed that the size of the constructed representative untangling of a net system is on average 24 % of the size of the constructed finite complete prefix of the augmented version of the net system obtained after injections of guards and control transitions.

8. Conclusion

This article contributes the theory of untanglings, which provides a novel characterization of behaviors encoded in concurrent systems. The theory revolves around the notion of a representative untangling, i.e., an alternative representation of a concurrent system, which strikes a good balance between the size of the model of a system and the time required for analysis of its behavioral properties.

The article proposes a baseline algorithm for constructing representative untanglings of concurrent systems. As the name suggests, this algorithm should be used as a reference that specifies basic principles for constructing

representative untanglings. The article also contributes a generic framework for improving the baseline algorithm based on structural transformations of input systems, and describes two reduction rules that fit this framework. It is shown that the desired properties of representative untanglings are preserved by this framework. Moreover, an experiment with a real-life collection of concurrent systems confirms the scalability of the improved algorithm both in terms of time and space. Finally, this article demonstrates the use of representative untanglings for analysis of concurrent systems. It is shown that representative untanglings offer significant benefits over the state-of-the-art when deciding behavioral properties of concurrent systems.

Future work can be performed (i) to extend the repertoire of structural transformations that fit the proposed framework for improving the baseline untangling algorithm, (ii) to improve the baseline untangling algorithm, e.g., by introducing mechanisms for pruning redundant (parts of) discovered maximal significant runs, (iii) to introduce the notion of a canonical representative untangling of a system, i.e., a representative untangling of the system with the least number of nodes, and (iv) to propose new analysis techniques that rely on the use of (representative) untanglings.

Acknowledgments

This work is partly funded by the Australian Research Council grant LP110100252. NICTA is funded by the Australian Government (Department of Broadband, Communications and the Digital Economy) and the Australian Research Council through the ICT Centre of Excellence program.

References

[BD90] Best E, Desel J (1990) Partial order behaviour and structure of Petri nets. Form Asp Comput 2(2):123–138

[DE95] Desel J, Esparza J (1995) Free choice Petri nets. Cambridge tracts in theoretical computer science, vol 40. Cambridge University Press, Cambridge

[Des00] Desel J (2000) Validation of process models by construction of process nets. In: Business process management (BPM). LNCS, vol 1806. Springer, Berlin, pp 110–128

[EH08] Esparza J, Heljanko K (2008) Unfoldings: a partial-order approach to model checking. Monographs in theoretical computer science. An EATCS series. Springer, Berlin

[ERV02] Esparza J, Römer S, Vogler W (2002) An improvement of McMillan's unfolding algorithm. Form Methods Syst Des 20(3):285–310

[Fah10] Fahland D (2010) From Scenarios to Components. PhD thesis, Humboldt-Universität zu Berlin, Berlin, Germany

[FFK+11] Fahland D, Favre C, Koehler J, Lohmann N, Völzer H, Wolf K (2011) Analysis on demand: Instantaneous soundness checking of industrial business process models. Data Knowl Eng 70(5):448–466

[GR83] Goltz U, Reisig W (1983) The non-sequential behavior of Petri nets. Inf Control (IANDC) 57(2–3):125–147

[GW93] Godefroid P, Wolper P (1993) Using partial orders for the efficient verification of deadlock freedom and safety properties. Form Methods Syst Des 2(2):149–164

[Hac75] Hack M (1975) Decidability questions for Petri nets. Outstanding dissertations in the computer sciences. Garland Publishing, New York

[Kho03] Khomenko V (2003) Model checking based on prefixes of Petri net unfoldings. PhD thesis, University of Newcastle upon Tyne, School of Computing Science, Newcastle upon Tyne, UK

[KKKV06] Khomenko V, Kondratyev A, Koutny M, Vogler W (2006) Merged processes: a new condensed representation of Petri net behaviour. Acta Informatica 43(5):307–330

[KM11] Khomenko V, Mokhov A (2011) An algorithm for direct construction of complete merged processes. In: Petri nets. LNCS, vol 6709. Springer, Berlin, pp 89–108

[LSV98] Lee EA, Sangiovanni-Vincentelli AL (1998) A framework for comparing models of computation. IEEE Trans Comput Aided Des Integr Circuits Syst 17(12):1217–1229

[McM92] McMillan KL (1992) Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In: Computer aided verification (CAV). LNCS, vol 663. Springer, Berlin, pp 164–177

[McM95] McMillan KL (1995) A technique of state space search based on unfolding. Form Methods Syst Des 6(1):45–65

[MR97] Melzer S, Römer S (1997) Deadlock checking using net unfoldings. In: Computer aided verification (CAV). LNCS, vol 1254. Springer, Berlin, pp 352–363

[Mur89] Murata T (1989) Petri nets: properties, analysis and applications. Proc IEEE 77(4):541–580

[NPW81] Nielsen M, Plotkin GD, Winskel G (1981) Petri nets, event structures and domains. Part I. Theor Comput Sci 13:85–108

[Pet77] Petri CA (1977) Non-sequential processes. Translation of a Lecture given at the IMMD Jubilee Colloquium on "Parallelism in Computer Science", Unversität Erlangen-Nürnberg. Translated by Philip Krause and John Low, Petri, CA, St. Augustin: Gesellschaft fnr Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-77-5

[PLtH14] Polyvyanyy A, La Rosa M, ter Hofstede AHM (2014) Indexing and efficient instance-based retrieval of process models using untanglings. In: Advanced information systems engineering (CAiSE). LNCS, vol 8484. Springer International Publishing, Berlin, pp 439–456

[Pnu77] Pnueli A (1977) The temporal logic of programs. In: annual symposium on foundations of computer science (FOCS). IEEE Computer Society, pp 46–57
[PW13] Polyvyanyy A, Weidlich M (2013) Towards a compendium of process technologies: the jBPT library for process model analysis. In: CAiSE forum. CEUR workshop proceedings, vol 998. CEUR-WS.org, pp 106–113
[PWC+14] Polyvyanyy A, Weidlich M, Conforti R, La Rosa M, ter Hofstede AHM (2014) The 4C spectrum of fundamental behavioral relations for concurrent systems. In: Petri nets. LNCS, vol 8489. Springer International Publishing, Berlin, pp 210–232
[Rac78] Rackoff C(1978) The covering and boundedness problems for vector addition systems. Theor Comput Sci 6(2):223–231
[Rei13] Reisig W(2013) Understanding Petri nets: modeling techniques, analysis methods, case studies. Springer, Berlin
[RSK13] Rodríguez C, Schwoon S, Khomenko V. (2013) Contextual merged processes. In: Petri nets. LNCS, vol 7927. Springer, Berlin, pp 29–48
[SNW96] Sassone V, Nielsen M, Winskel G (1996) Models for concurrency: towards a classification. Theor Comput Sci 170(1–2):297–348
[Tar97] Tarasyuk IV (1997) Equivalence notions for models of concurrent and distributed systems. PhD thesis, A.P. Ershov Institute of Informatics Systems, Siberian Division of the Russian Academy of Sciences, Novosibirsk, Russia
[vdA97] van der Aalst WMP (1997) Verification of workflow nets. In: Petri nets. LNCS, vol 1248. Springer, Berlin, pp 407–426
[vGV87] van Glabbeek RJ, Vaandrager FW (1987) Petri net models for algebraic theories of concurrency. In: Parallel architectures and languages Europe (PARLE). LNCS, vol 259. Springer, Berlin, pp 224–242

Received 13 July 2013
Revised 24 April 2014
Accepted 4 September 2014 by Jim Woodcock
Published online 13 January 2015