Theoretical Computer Science 551 (2014) 84–101

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Recent advances in unfolding technique

Blai Bonet[a], Patrik Haslum[b], Victor Khomenko[c,*], Sylvie Thiébaux[b], Walter Vogler[d]

[a] *Departamento de Computación, Universidad Simón Bolívar, Caracas, Venezuela*
[b] *The Australian National University & NICTA, Canberra, Australia*
[c] *School of Computing Science, Newcastle University, Newcastle upon Tyne, UK*
[d] *Institut für Informatik, Universität Augsburg, Augsburg, Germany*

ARTICLE INFO

Article history:
Received 10 December 2013
Received in revised form 6 May 2014
Accepted 6 July 2014
Available online 15 July 2014
Communicated by G. Rozenberg

Keywords:
Adequate order
Unfolding prefix
Petri net
Well-foundedness
Concurrency

ABSTRACT

We propose a new, and to date the most general, framework for Petri net unfolding, which broadens its applicability, makes it easier to use, and increases its efficiency. In particular: (i) we propose a user-oriented view of the unfolding technique, which simply tells which information will be preserved in the final prefix and how to declare an event a cut-off in the algorithm, while hiding the technical parameters like the adequate order; (ii) the notion of the adequate order is generalised to a well-founded relation, and the requirement that it must refine $\subset$ is replaced by a weaker one; and (iii) the order in which the unfolding algorithm selects the possible extensions of the prefix is entirely disentangled from the cut-off condition. We demonstrate the usefulness of the developed theory on some case studies.

1. Introduction

A distinctive characteristic of reactive concurrent systems is that their sets of local states have descriptions which are both short and manageable, and the complexity of their behaviour comes from highly complicated interactions with the external environment rather than from complicated data structures and manipulations thereon. One way of coping with this complexity problem is to use formal methods and, especially, computer aided verification tools implementing model checking (see, e.g. [4]) — a technique in which the verification of a system is carried out using a finite representation of its state space.

The main drawback of model checking is that it suffers from the *state space explosion* problem. That is, even a relatively small system specification can (and often does) yield a very large state space. To alleviate this problem, a number of techniques have been proposed. Among them, a prominent technique is McMillan's (finite prefixes of) Petri net unfoldings (see, e.g. [12,26]). It relies on the partial order view of concurrent computation, and represents system states implicitly, using an acyclic net. More precisely, given a Petri net Σ, the unfolding technique aims at building a labelled acyclic net, called *(unfolding) prefix*, satisfying two key properties:

* Corresponding author.
E-mail addresses: bonet@ldc.usb.ve (B. Bonet), patrik.haslum@anu.edu.au (P. Haslum), Victor.Khomenko@ncl.ac.uk (V. Khomenko), sylvie.thiebaux@anu.edu.au (S. Thiébaux), vogler@informatik.uni-augsburg.de (W. Vogler).

http://dx.doi.org/10.1016/j.tcs.2014.07.003
0304-3975/© 2014 Elsevier B.V. All rights reserved.

– *Completeness:* each reachable marking of Σ is represented by at least one 'witness', i.e. a marking of the prefix reachable from its initial marking; similarly, for each possible firing of a transition at any reachable state of Σ there is a suitable 'witness' event in the prefix.
– *Finiteness:* the prefix is finite and thus can be used as an input to model checking algorithms, e.g. those searching for deadlocks. The finiteness is commonly achieved by identifying a set of *cut-off* events beyond which the unfolding is not generated.

A prefix satisfying these two properties can be used for model checking as a condensed representation of the state space of a system.

Practical algorithms exist for building complete prefixes [12,19], which ensure that the number of (non-cut-off) events in such a prefix can never exceed the number of reachable states of the Petri net. However, complete prefixes are often much smaller than the corresponding state graphs, especially for highly concurrent Petri nets, because they represent concurrency directly rather than by multidimensional 'diamonds' as it is done in state graphs. For example, if the original Petri net consists of 100 transitions which can fire once in parallel, the state graph will be a 100-dimensional hypercube with 2^{100} vertices, whereas the complete prefix will coincide with the net itself. In many applications, e.g. in asynchronous circuit design, the Petri net models usually exhibit a lot of concurrency, but have rather few choice points, and so their unfolding prefixes are often exponentially smaller than the corresponding state graphs; in fact, in many of the experiments conducted in [19] they are just slightly bigger then the original Petri nets themselves. Therefore, unfolding prefixes are well-suited for alleviating the state space explosion problem.

In this paper, we investigate a number of important generalisations of unfolding, which broaden its applicability and increase its efficiency. We generalise the notion of a *cutting context* [22] (a parametric framework for identifying cut-off events) in the following directions:

– The *adequate order* $\lhd$, which is commonly used to define cut-off events of the prefix, does not have to be part of the cutting context — merely the *existence* of a suitable relation $\lhd$ is sufficient for ensuring that the prefix is complete. This provides a *user-oriented view* of the unfolding technique, which simply tells which information will be preserved in the final prefix and how to declare an event a cut-off in the algorithm, while hiding the technical parameters like the adequate order.
– The adequate order is replaced by a well-founded relation. Moreover, the requirement that it must refine $\subset$ is replaced by a weaker one; cf. the definition of semi-adequate orders in [2].
– The order in which the unfolding algorithm selects the possible extensions of the prefix is entirely disentangled from the cut-off condition.

The latter generalisation in particular has important applications, see Section 8. For instance, it enables efficient search strategies to be used to construct the unfolding or perform reachability analysis, without sacrificing completeness. Without disentanglement, even a simple depth-first search strategy leads to incompleteness [11]. Disentanglement also enables the unfolding technique to find *optimal* solutions to reachability problems, even for notions of optimality which, such as makespan, cannot be captured by adequate (or semi-adequate) orderings.

The paper is organised as follows. Section 2 explains basic notions related to Petri nets and unfolding prefixes. In Section 3 we present a new unfolding algorithm, where the selection of a possible extension is entirely disentangled from the cut-off condition. In Section 4, we define our new notion of a cutting context, which is central to this paper. Section 5 proves the correctness of the proposed unfolding algorithm. Section 6 gives an overview of related work in the area of unfolding prefixes and argues in some detail that the newly proposed theory and unfolding algorithm generalise previous approaches; as a first application, we show that in our setting also depth-first search can be applied, which is somewhat in contrast to earlier results. In Section 7, we demonstrate how the canonicity result of [22] can be restored by restricting the selection strategy used by the new unfolding algorithm. Section 8 demonstrates the usefulness of the developed theory and unfolding algorithm on a number of applications: In particular, the disentanglement of the selection strategy from the cut-off condition allows one to use some extra information (in the form of a heuristic function) to guide the unfolding algorithm towards a target configuration, to prune 'dead ends' (configurations that cannot be extended to a target configuration), and to find an optimal target configuration (according to various measures of cost). Section 9 concludes the paper and gives some possible directions for future work.

2. Basic notions

In this section, we first present basic definitions concerning Petri nets, and then recall (see also [9,12]) notions related to net unfoldings.

2.1. Petri nets

A *net* is a triple $N \overset{\mathrm{df}}{=} (P, T, F)$ such that P and T are disjoint sets of respectively *places* and *transitions*, and $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*. A *marking* of N is a multiset M of places, i.e. $M : P \to \mathbb{N} = \{0, 1, 2, \ldots\}$. We adopt the standard rules about drawing nets, viz. places are represented as circles, transitions as boxes, the flow relation by arcs, and markings are shown by placing tokens within circles. As usual, ${}^\bullet z \overset{\text{df}}{=} \{y \mid (y,z) \in F\}$ and $z^\bullet \overset{\text{df}}{=} \{y \mid (z,y) \in F\}$ denote the *pre-* and *postset* of $z \in P \cup T$, and we define ${}^\bullet Z \overset{\text{df}}{=} \bigcup_{z \in Z} {}^\bullet z$ and $Z^\bullet \overset{\text{df}}{=} \bigcup_{z \in Z} z^\bullet$, for all $Z \subseteq P \cup T$. We will assume that ${}^\bullet t \neq \emptyset$, for every $t \in T$. N is *finite* if $P \cup T$ is finite, and *infinite* otherwise.

A *Petri net (PN)* is a pair $\Sigma \overset{\text{df}}{=} (N, M_0)$ comprising a finite net $N = (P, T, F)$ and an *initial* marking M_0. A transition $t \in T$ is *enabled* at a marking M if, for every $p \in {}^\bullet t$, $M(p) \geq 1$. Such a transition can be *executed*, leading to a marking $M' \overset{\text{df}}{=} M - {}^\bullet t + t^\bullet$, where '$-$' and '$+$' stand for the multiset difference and sum respectively. We denote this by $M[t\rangle M'$. The set of *reachable* markings of Σ is the least (w.r.t. $\subseteq$) set $\mathcal{M}(\Sigma)$ containing M_0 and such that if $M \in \mathcal{M}(\Sigma)$ and $M[t\rangle M'$ (for some $t \in T$) then $M' \in \mathcal{M}(\Sigma)$.

A PN Σ is *k-bounded* if, for every reachable marking M and every place $p \in P$, $M(p) \leq k$, *safe* if it is 1-bounded, and *bounded* if it is k-bounded for some $k \in \mathbb{N}$. The set $\mathcal{M}(\Sigma)$ is finite iff Σ is bounded.

2.2. Branching processes

Two nodes (places or transitions), y and y', of a net $N = (P, T, F)$ are *in conflict*, denoted by $y \# y'$, if there are distinct transitions $t, t' \in T$ such that ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$ and (t, y) and (t', y') are in the reflexive transitive closure of the flow relation F, denoted by $\preceq$. A node y is in *self-conflict* if $y \# y$.

An *occurrence net* is a net $ON \overset{\text{df}}{=} (B, E, G)$, where B is the set of *conditions* (places) and E is the set of *events* (transitions), satisfying the following: ON is acyclic (i.e. $\preceq$ is a partial order); for every $b \in B$, $|{}^\bullet b| \leq 1$; for every $y \in B \cup E$, $\neg(y \# y)$ and there are finitely many y' such that $y' \prec y$, where $\prec$ denotes the transitive closure of G. $Min(ON)$ will denote the set of minimal (w.r.t. $\prec$) elements of $B \cup E$. The relation $\prec$ is the *causality relation*, and for events $f \prec e$, f is called a *causal predecessor* of e. A *$\prec$-chain* of events is a finite or infinite set of events totally ordered by $\prec$. A $\prec$-chain is called *maximal* if it is maximal w.r.t. $\subset$. Two nodes are *concurrent*, denoted $y \| y'$, if neither $y \# y'$ nor $y \preceq y'$ nor $y' \preceq y$.

A *homomorphism* from an occurrence net ON to a PN Σ is a mapping $\varphi : B \cup E \to P \cup T$ such that: $\varphi(B) \subseteq P$ and $\varphi(E) \subseteq T$ (conditions are mapped to places, and events to transitions); for all $e \in E$, the restriction of φ to ${}^\bullet e$ is a bijection between ${}^\bullet e$ and ${}^\bullet \varphi(e)$ and the restriction of φ to $e^\bullet$ is a bijection between $e^\bullet$ and $\varphi(e)^\bullet$ (transition environments are preserved); $\varphi(Min(ON)) = M_0$ with φ extended to multisets in the natural way (minimal conditions correspond to the initial marking); and for all $e, f \in E$, if ${}^\bullet e = {}^\bullet f$ and $\varphi(e) = \varphi(f)$ then $e = f$ (there is no redundancy). A *branching process* of Σ is a pair $\pi \overset{\text{df}}{=} (ON, \varphi)$ such that ON is an occurrence net and φ is a homomorphism from ON to Σ. If an event e is such that $\varphi(e) = t$, then we will often refer to it as being *t-labelled*.

The following analog of König's lemma [23] holds for branching processes.

Proposition 1. *(See [22, Prop. 1].) A branching process is infinite iff it contains an infinite $\prec$-chain of events.*

A branching process $\pi' = (B', E', G', \varphi')$ of Σ is a *prefix* of a branching process $\pi = (B, E, G, \varphi)$, denoted by $\pi' \sqsubseteq \pi$, if (B', E', G') is a subnet of (B, E, G) containing all minimal elements and such that: if $e \in E'$ and $(b, e) \in G$ or $(e, b) \in G$ then $b \in B'$; if $b \in B'$ and $(e, b) \in G$ then $e \in E'$; and φ' is the restriction of φ to $B' \cup E'$. For each Σ there exists a unique (up to isomorphism) maximal (w.r.t. $\sqsubseteq$) branching process Unf_Σ^{max}, called the *unfolding* of Σ [9].

An example of a safe PN and two of its branching processes is shown in Fig. 1, where the homomorphism φ is indicated by the labels of the nodes. The process in Fig. 1(b) is a prefix of that in Fig. 1(c).

2.3. Configurations and cuts

A *configuration* of an occurrence net ON is a finite set of events C that is causally closed (for every $e \in C$, $f \prec e$ implies $f \in C$) and free of conflicts (for all $e, f \in C$, $\neg(e \# f)$). For every event $e \in E$, the configuration $[e] \overset{\text{df}}{=} \{f \mid f \preceq e\}$ is called the *local configuration* of e. For example, in the prefix shown in Fig. 1(b), $\{e_1, e_3, e_4\}$ is a configuration, whereas $\{e_1, e_3, e_7\}$ and $\{e_1, e_2, e_3, e_5\}$ are not (the former does not include $e_4 \prec e_7$, while the latter contains conflicting events $e_1 \# e_2$); furthermore, $\{e_1, e_3, e_4\}$ is a local configuration of e_4, whereas the configuration $\{e_1, e_3\}$ is not local.

A set of events E' is a *suffix* of a configuration C if $C \cap E' = \emptyset$ and $C \cup E'$ is a configuration; in such a case the configuration $C \oplus E' \overset{\text{df}}{=} C \cup E'$ is called an *extension* of C. Note that, according to this definition, suffixes and extensions are always finite. For example, in the prefix shown in Fig. 1(b), the configuration $C = \{e_1, e_3, e_4\}$ has a suffix $E' = \{e_6, e_7\}$, with the corresponding extension $C \oplus E' = \{e_1, e_3, e_4, e_6, e_7\}$, whereas the sets $\{e_2\}$ and $\{e_1, e_6\}$ are not suffixes of C (the former would not yield a configuration when united with C due to $e_1 \# e_2$, while the intersection of the latter with C is non-empty), and so $\oplus$ is not defined for them.

The set of all (resp. local) configurations of a branching process π will be denoted by $\mathcal{C}_{all}^{\pi}$ (resp. $\mathcal{C}_{loc}^{\pi}$), and we will drop the superscript π in the case $\pi = Unf_\Sigma^{max}$.

A set of conditions B' such that for all distinct $b, b' \in B'$, $b \| b'$, is called a *co-set*. A *cut* is a maximal (w.r.t. $\subset$) co-set. Every marking reachable from $Min(ON)$ is a cut.

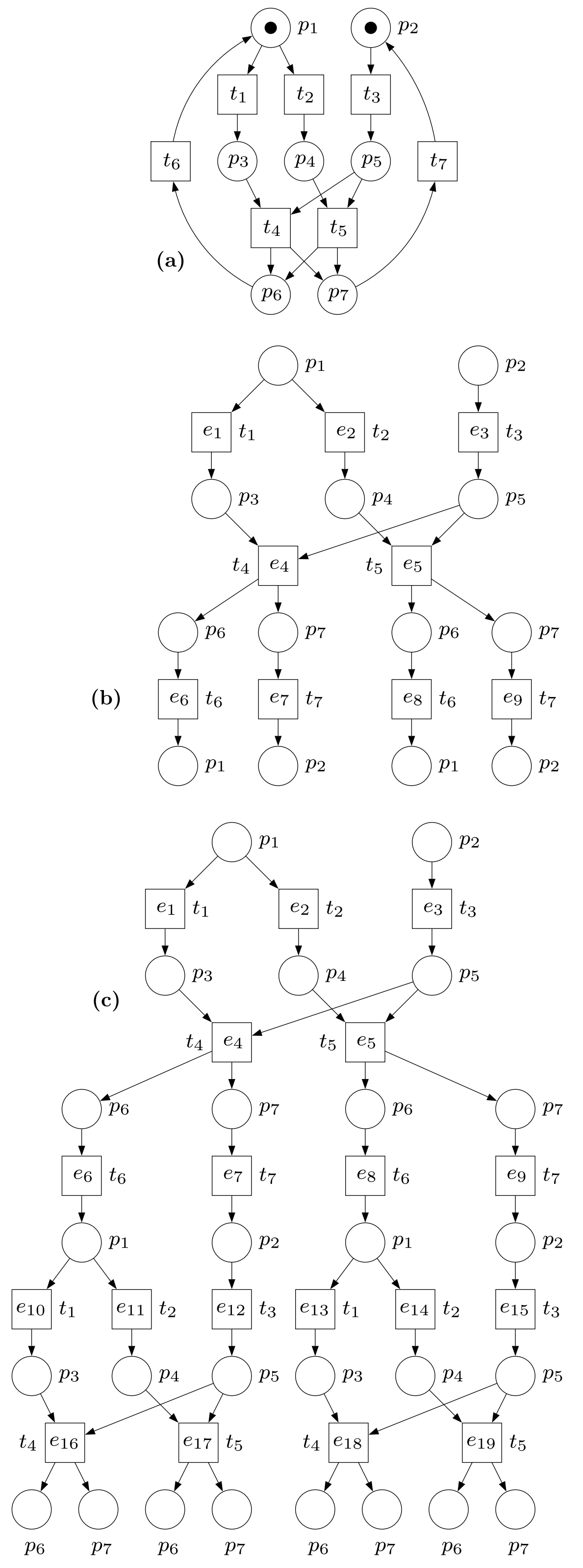

Fig. 1. A PN (**a**) and two of its branching processes (**b**, **c**).

```
input : $\Sigma = (N, M_0)$ — a Petri net
output : $\pi$ — a complete prefix of $Unf_\Sigma^{max}$
$\pi$ := the empty branching process
add instances of the places from $M_0$ to $\pi$
$cut\_off := \emptyset$
$pe := \textsc{PotExt}(\pi, cut\_off)$
while $pe \neq \emptyset$ do
  select $e \in pe$
  add $e$ and new instances of places from $\varphi(e)^\bullet$ to $\pi$
  if $\mathbb{C}(e, \pi, cut\_off)$ then $cut\_off := cut\_off \cup \{e\}$
  $pe := \textsc{PotExt}(\pi, cut\_off)$
```

Fig. 2. The unfolding algorithm.

Let C be a configuration of a branching process π. Then the set

$$Cut(C) \overset{\text{df}}{=} (Min(ON) \cup C^\bullet) \setminus {}^\bullet C$$

is a cut, and the multiset[1] of places $Mark(C) \overset{\text{df}}{=} \varphi(Cut(C))$ is a reachable marking of Σ, called the *final marking* of C. For example, in Fig. 1(b) $Mark(\{e_1, e_3, e_4\}) = \{p_6, p_7\}$.

A marking M of Σ is *represented* in π if there is $C \in \mathcal{C}_{all}^{\pi}$ such that $M = Mark(C)$. Every marking represented in π is reachable in the original PN Σ, and every reachable marking of Σ is represented in Unf_Σ^{max} [9].

In the rest of this paper, we assume that Σ is a fixed, though not necessarily bounded, PN, and that $Unf_\Sigma^{max} = (B, E, G, \varphi)$ is its unfolding.

3. Unfolding algorithm

In Fig. 2, we present a generalised version of the unfolding algorithm. It iteratively generates a prefix π of Unf_Σ^{max} together with a set *cut_off* containing some of π's events; the events in *cut_off* are always 'maximal', i.e. π without $cut\_off \cup cut\_off^\bullet$ is a prefix of π.

The function $\textsc{PotExt}(\pi, cut\_off)$ finds the set of *possible extensions* of π: a possible extension is a new event e labelled with some transition t and associated with a co-set D of π such that ${}^\bullet D \cap cut\_off = \emptyset$, $|D| = |{}^\bullet t|$, $\varphi(D) = {}^\bullet t$ and π contains no t-labelled event with preset D. If e is added to π, ${}^\bullet e$ is set to D. (In practice, instead of recomputing the set of possible extensions on each iteration of the main loop, it is sufficient to update the set left from the previous iteration by removing the selected event e from it and adding some of the direct causal successors of e; see [19, Chapt. 4] for an efficient algorithm.)

$\textsc{PotExt}(\pi, cut\_off)$ is always a finite set, since at each step of the algorithm π is finite and thus has only finitely many co-sets. Furthermore, $\textsc{PotExt}(\pi, cut\_off)$ cannot contain a pair of events e, f such that $e \prec f$, as f would not be a possible extension of π in such a case.

The structure of the algorithm in Fig. 2 is similar to that in [12], but selection of the next event from the set of possible extensions and the cut-off condition have been disentangled from each other. In fact, the former is now fully non-deterministic, while the latter is given by the predicate $\mathbb{C}$ explained below.

The selection of the next event from the set of possible extensions is modelled by the non-deterministic **select** operator, that, given a non-empty set of possible extensions, chooses an event from it. (Note that, as demonstrated above, *pe* is always finite, and the loop condition guarantees that $pe \neq \emptyset$.) In the actual implementation this non-determinism can be restricted or altogether eliminated in order to improve the efficiency and/or reduce the size of the constructed prefix; this will be referred to as *selection strategy* and illustrated by examples.

The *cut-off condition* $\mathbb{C}(e, \pi, cut\_off)$, where π is a branching process, e is an event of π, and *cut_off* is a set of events of π, is a predicate defined as follows. It depends on a decidable *cutting relation* ✂ defined in Definition 3 below, which is a parameter of the unfolding algorithm. $\mathbb{C}(e, \pi, cut\_off)$ is true iff π contains a configuration C, such that $C \cap cut\_off = \emptyset$ and C ✂ e. A configuration C satisfying these conditions is called a cutter of e. Note that condition $\mathbb{C}(e, \pi, cut\_off)$ is decidable, as π is finite (and so contains only finitely many configurations) and ✂ is required to be decidable.

At the time the unfolding algorithm adds an event e to the prefix, it designates e as a *cut-off event* (by putting it into the set *cut_off*) iff $\mathbb{C}(e, \pi, cut\_off)$ holds. Once in *cut_off*, the event stays there until the algorithm terminates; and vice versa, if the event is not put into *cut_off* at the time of addition, it cannot subsequently be put there.

[1] We extend φ to multisets in the natural way.

Consider an event e that is added to π, and let $\pi' \sqsupset \pi$ be the prefix at a later stage of the algorithm. Let also cut_off' and cut_off denote the respective sets of cut-off events. Then $\mathbb{C}(e, \pi, cut\_off)$ implies $\mathbb{C}(e, \pi', cut\_off')$, but not vice versa, because π' can contain a cutter C of e, some events of which are not in π.

4. Completeness and cutting context

In this section, we will generalise the theory developed in [22]. There, a parametric setup was presented — called *cutting context* — in which questions concerning completeness and cut-off events could be discussed in a uniform and general way.

The first parameter in a cutting context captures the information that is to be retained in a complete prefix. Crucially, the emphasis is shifted from reachable markings to the configurations of Unf_{Σ}^{max}, and this parameter, a suitably defined equivalence relation $\approx$, specifies which configurations can be regarded as equivalent. Intuitively, one has to retain at least one representative configuration from each equivalence class of $\approx$.

The standard setting when the set of reachable markings should be retained is then covered by regarding two configurations as equivalent iff they reach the same marking; this corresponds to the equivalence $\approx_{mar}$ defined as $C \approx_{mar} C' \iff Mark(C) = Mark(C')$.

The motivation for a more general $\approx$ is that in some circumstances the set of reachable markings provides either insufficient or redundant information. For example, in the Signal Transition Graphs formalism [27,3] used for specification of asynchronous circuits, the state of the system is comprised of the marking and the values of binary variables called *signals;* hence the knowledge of reachable markings only is insufficient for model checking [28]. Alternatively, if the Petri net has symmetries, it is usually enough to preserve one marking from each symmetry class, and thus building a prefix in which all reachable markings are represented is inefficient [6].

For a given equivalence relation $\approx$, one then can define what it means for a prefix to be complete. In essence, all equivalence classes of $\approx$ must be represented, and any configuration involving no cut-off events can be extended (in a single step, perhaps using a cut-off event) in exactly the same way as in the full unfolding. (In the standard setting with the equivalence being $\approx_{mar}$, the completeness is achieved in the sense that each reachable marking is represented in the prefix.)

Definition 2 *(Completeness).* (See [22, Def. 4].) A branching process π is *complete w.r.t. a set* cut_off of its events if the following hold:

- For every $C \in \mathcal{C}_{all}$ there exists a $C' \in \mathcal{C}_{all}^{\pi}$ such that $C' \cap cut\_off = \emptyset$ and $C \approx C'$.
- For every $C \in \mathcal{C}_{all}^{\pi}$ such that $C \cap cut\_off = \emptyset$, and every event e of Unf_{Σ}^{max} such that $C \oplus \{e\} \in \mathcal{C}_{all}$, we have $C \oplus \{e\} \in \mathcal{C}_{all}^{\pi}$.

In this paper, we follow this idea and retain $\approx$ as a parameter of the proposed setup, in essentially the same way as in [22], as well as the above notion of completeness.

The other parameter of the proposed setup specifies the circumstances under which an event can be designated as a cut-off. This is different from [22], where the cutting context has three parameters.

In the following, we call an arbitrary binary relation R *well-founded* if there is no infinite 'decreasing' chain $x_1\, R^{-1}\, x_2\, R^{-1}\, x_3\, R^{-1} \ldots$; note that any non-empty set has a minimal element w.r.t. a well-founded relation, i.e. some y such that there is no x in the set with xRy.

Definition 3 *(Cutting context).* A pair $\Theta \overset{\text{df}}{=} (\approx, ✂)$ is a *cutting context* if $\approx$ is an equivalence relation on $\mathcal{C}_{all}$ and $✂$, called a *cutting relation*, is a decidable relation between configurations and events of Unf_{Σ}^{max} such that there is a well-founded relation $\ll$ on $\mathcal{C}_{all}$ satisfying:

CC:PE $✂$ *has related extensions*, i.e. for every event e, every configuration C such that $C ✂ e$, and every suffix E of $[e]$, there exists a suffix E' of C such that
i) $C \oplus E' \approx [e] \oplus E$;
ii) $C \oplus E' \ll [e] \oplus E$.
If $C ✂ e$, we call C a *cutter* of e.

We call Θ a *finitary* cutting context if $\approx$ has a finite index and

CC:CHAIN for every infinite $\prec$-chain X of events whose local configurations are equivalent w.r.t. $\approx$, there exist events $f \in X$ and $e \prec f$ (e is not necessarily in X) and a configuration $C \subset [e]$ such that $C ✂ e$.

The first parameter of the cutting context, viz. the equivalence relation $\approx$ was already explained above; note that the notion of completeness of a branching process (Definition 2) depends only on $\approx$, but not on $✂$.

The second parameter $\blacktriangleright\!\!<$ specifies under which circumstances an event can be declared a cut-off. Note that the user-oriented view of the unfolding technique is provided in the following sense. The user who only wants to use a ready prefix needs to know only w.r.t. which equivalence relation $\approx$ the prefix is complete, and can ignore $\blacktriangleright\!\!<$. On the other hand, the developer of the unfolding algorithm needs to know only the decidable $\blacktriangleright\!\!<$, as it provides all the information needed for declaring an event a cut-off. The relation $\lll$ (which plays the same role as the (semi-)adequate order in [2,12,22]) is only needed for correctness purposes — it is used in the correctness proof, but not in the unfolding algorithm or subsequent prefix-based model checking; hence, it does not appear as a parameter of the cutting context, but rather its existence is postulated (it can even be undecidable). It will turn out that for *all* suitable $\lll$, all $\lll$-minimal configurations in each equivalence class of $\approx$ are preserved, though this is only used for the correctness proof.

The *CC:PE* condition is needed to ensure the completeness of the prefix; it is a generalised version of the preservation by extensions property of [22] (as it considers only the suffix E of a local (rather than an arbitrary) configuration, and a somewhat weaker precondition $C \blacktriangleright\!\!< e$ is used instead of $C \lll [e]$).

In fact, in the setup of [2,12,22], configuration C could only be used for cutting off event e if $C \lhd [e]$ (where $\lhd$ is a (semi-)adequate order), so $\blacktriangleright\!\!<$ would be, in some sense, a sub-relation of $\lll$. We do not require this here explicitly, which makes the setting and subsequent proofs simpler. But the requirement is almost implied by *CC:PE*: if $E = \emptyset$, then we would presumably have $E' = \emptyset$ in many settings.

Similarly, in the usual settings, C could only be used for cutting off event e if $C \approx [e]$ — making $\blacktriangleright\!\!<$, in some sense, a sub-relation of $\approx$. Also this is not required here, but almost implied as well.

The *CC:CHAIN* condition is needed to guarantee the finiteness of the prefix in the case when $\approx$ has a finite index. Note that this condition is much weaker than the corresponding conditions in [2,12,22]: in all those papers the finiteness of the prefix was ensured by having a cut-off event in each suitable infinite $\prec$-chain, whereas this is not required by *CC:CHAIN* as e is not required to be in X.

In Section 6 we show how to embed other existing setups, including those in [2,12,22], into the proposed one.

5. Arbitrary event selection

Different runs of the (non-deterministic) unfolding algorithm in Fig. 2 can declare different events cut-offs, and hence produce different prefixes. This is quite unlike the theory of canonical prefixes developed in [22], where the set of cut-off events (and thus the prefix itself) is completely determined by the cutting context.

In contrast to all previous approaches,[2] in our algorithm *we do not pose any restrictions on the* **select** *operator* in the unfolding algorithm in Fig. 2. Note also that the selection of the next event to be added to the prefix may actually depend on the prefix generated so far, and even on the history of generation (e.g. a depth-first strategy of prefix generation can be imposed).

Should the unfolding algorithm not terminate (which is possible if $\approx$ has an infinite index, in particular if a cutting context based on $\approx_{mar}$ is used to unfold an unbounded Petri net), we regard as its result the least upper bound (w.r.t. $\sqsubset$) of all the intermediate values of π (similarly for *cut_off* w.r.t. $\subset$); for this case, we make the fairness assumption that any event in $\textsc{PotExt}(\pi, \mathit{cut\_off})$ at some stage is eventually selected (which is always guaranteed for terminating runs). Observe that an event remains in $\textsc{PotExt}(\pi, \mathit{cut\_off})$ if it is not added to the prefix at the current step.

It is easy to see that: (i) any result of the unfolding algorithm in Fig. 2 is indeed a branching process; and (ii) whenever e is selected, all causal predecessors of e are non-cut-off events of π at that stage. However, one should be very careful when proving the correctness (i.e. the finiteness and completeness of the resulting prefix), as Definition 3 introduces a number of subtle changes to the traditional framework. Hence, though our proof of correctness is similar in structure to the corresponding proofs for traditional unfolding algorithms, it is conducted with extra care, in particular some algorithm invariants are described below.

Lemma 4 *(Algorithm invariants). The following invariants hold for the algorithm in Fig. 2:*

Inv1 If an event is added to the prefix, then it will remain in the prefix and keep its cut-off status (that is determined at the time of adding) forever.
Inv2 If an event is in the prefix at some stage, then all its proper causal predecessors are in that prefix and not cut-offs.

Proof. Inv1: Trivial.

Inv2: Induction on the number of times the main loop is executed: when an event e is added, the claim is true for the previously added events by induction, and remains true due to Inv1. As e is a possible extension, all its direct causal predecessors are in the prefix and not cut-offs; as we have seen above, this carries over to all causal predecessors of e. □

[2] The idea of the possibility of arbitrary selection (Theorems 6 and 7 in this section) was already mentioned in the book [10, Section 4.6] by Esparza and Heljanko with a forward reference to the present paper as the source of this idea. Furthermore, the proposed notion of the cutting context is more general than the approach in [10].

Lemma 5 *(Eventuality conditions). The following eventuality conditions hold for the algorithm in Fig. 2:*

Inv1 If at some stage all proper causal predecessors of an event e are in the prefix and not cut-offs, then e will be eventually added.
Inv2 For each configuration C of the unfolding, eventually either some event $e \in C$ is declared a cut-off or all events of C are added to the prefix.

Proof. Ev1: e is a possible extension at that stage and will remain so due to Inv1 until it is added. In the case of an infinite run, the latter is guaranteed by the fairness assumption.

Ev2: If not, take a causally minimal event e of C that is not in the final prefix. By the choice of e, all proper causal predecessors are in the final prefix and they are not cut-offs by assumption. This contradicts Ev1. □

With the above invariants, it is easy to prove the correctness of the unfolding algorithm. First, we prove the completeness of any prefix π it can generate. In fact, a stronger condition implying completeness is proved; it states that every $\lll$-minimal configuration in every equivalence class of $\approx$ is preserved in π. This stronger condition is very useful in practice, see Section 8.

Theorem 6 *(Completeness). Let Θ be a cutting context, $\lll$ be any relation satisfying the conditions of Definition 3, and π be any result of the unfolding algorithm in Fig. 2. Moreover, let C be any configuration of the full unfolding which is $\lll$-minimal in its equivalence class of $\approx$. Then all the events of C are in π and are not in cut_off; in particular, π is complete.*

Proof. By Ev2 of Lemma 5, if C is not in π, then it contains a cut-off event e. When e was added to π, there was some cutter C'' of e, i.e. the events of C'' belonged to the prefix but not to *cut_off*, and $C'' \,✂\, e$. Since C is an extension of $[e]$, we have by *CC:PE* an extension C' of C'' such that $C' \approx C$ and $C' \lll C$, which contradicts the $\lll$-minimality of C in its equivalence class of $\approx$.

Due to well-foundedness of $\lll$, each equivalence class of $\approx$ has a $\lll$-minimal configuration. As this configuration is in π and contains no cut-off events, π satisfies the first part of Definition 2. By Ev1 of Lemma 5, all the events $e \notin C$ such that $C \oplus \{e\} \in \mathcal{C}_{all}$ have been added to π, and so the second part of Definition 2 is satisfied, i.e. π is complete. □

We now prove the second correctness property of the unfolding algorithm in Fig. 2: Provided the cutting context is finitary, any prefix it can generate is finite. This implies the termination of the algorithm, as every iteration of its main loop adds a new event to the prefix being built, which can happen only a finite number of times for a finite prefix.

Theorem 7 *(Finiteness). Let Θ be a finitary cutting context. Then any result π of the unfolding algorithm in Fig. 2 is finite.*

Proof. Due to Proposition 1 it suffices to show that there is no infinite $\prec$-chain X of events in π; suppose otherwise. Since $\approx$ has a finite index, $\approx$ restricted to X must contain an infinite subchain X' where all elements have equivalent local configurations; so by *CC:CHAIN* there exist events $f \in X' \subseteq X$ and $e \prec f$ and a configuration $C \subset [e]$ such that $C \,✂\, e$. If e is added to the prefix, all the events of C (which are proper causal predecessors of e due to $C \subset [e]$) are already present by Inv2 of Lemma 4. Hence, e is a cut-off event of π or never added to π, and so f is not in π due to $e \prec f$ and Inv2, contradicting the assumption that X is in π. □

5.1. Remark on the chain condition

In an earlier version of the finiteness proof above, we showed that for every infinite $\prec$-chain X of events whose local configurations are equivalent w.r.t. $\approx$, there must be some $e, f \in X$ with $e \prec f$ such that $[e] \lll [f]$. We wanted to show an even stronger result that there is an infinite subchain $X' \subseteq X$ such that for all $e, f \in X'$ with $e \prec f$, $[e] \lll [f]$. For this, we proved the following generic result, which, though it eventually turned out to be unnecessary for our finiteness proof, might be quite useful for the unfolding theory. (It might be known but since we did not find a reference, we include it here.)

Lemma 8. *Let $\lessdot$ be some strict partial order, and R some other binary relation on the same set. Assume that for every infinite chain $e_1 \lessdot e_2 \lessdot \cdots$ there are $i < j$ with $e_i \; R \; e_j$. Then for every infinite chain $e_1 \lessdot e_2 \lessdot \cdots$ there is an infinite subchain $e'_1 \lessdot e'_2 \lessdot \cdots$ such that $e'_i \; R \; e'_j$ for all $i < j$.*

Proof. Consider the infinite complete graph on the given e_i; colour an edge $e_i e_j$ with $i < j$ white if $e_i \; R \; e_j$ and black otherwise. A theorem of Ramsey theory (see e.g. [8, Thm. 9.1.2 for $k = c = 2$]) ensures the existence of an infinite complete monochromatic subgraph, which yields an infinite subchain $e'_1 \lessdot e'_2 \lessdot \cdots$. By assumption, $e'_i \; R \; e'_j$ for some $i < j$, i.e. the edge $e'_i e'_j$ is white. Thus, all edges of this subgraph are white. □

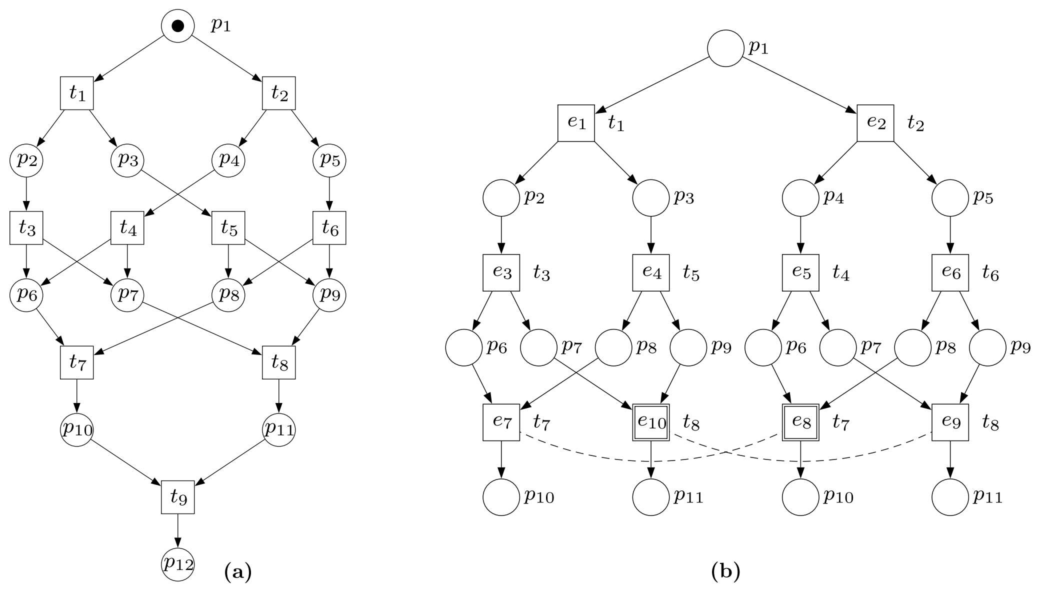

Fig. 3. A safe Petri net (**a**) and a prefix of its unfolding (**b**).

6. Related work and first applications

The development of the unfolding technique, starting from the landmark paper [26], has been a series of generalisations. We now show how the approaches proposed earlier fit into the setup of this paper.

The unfolding algorithm proposed in [26] can be obtained from that in Fig. 2 by choosing the cutting relation as

$$C \mathrel{\text{✂}} e \iff C \in \mathcal{C}_{loc} \wedge C \approx_{mar} [e] \wedge |C| < \big|[e]\big|,$$

and using a selection strategy that chooses an event with minimal local configuration size.

In [12] a general notion of an *adequate order* $\lhd$ was proposed, i.e. a strict well-founded partial order on configurations of the unfolding, that refines $\subset$ and is preserved by extensions, i.e. if $C' \approx_{mar} C''$ and $C' \lhd C''$ then for any suffix E'' of C'' there exists an isomorphic suffix E' of C' such that $C' \oplus E' \approx_{mar} C'' \oplus E''$ and $C' \oplus E' \lhd C'' \oplus E''$. The unfolding algorithm of [12] can be obtained from that in Fig. 2 by choosing the cutting relation as

$$C \mathrel{\text{✂}} e \iff C \in \mathcal{C}_{loc} \wedge C \approx_{mar} [e] \wedge C \lhd [e],$$

and using a selection strategy that chooses an event from $\min_{\lhd} pe$ rather than the whole pe. One can see that the technique of [26] is a special case of the setup of [12], with the adequate order defined as $C' \lhd_m C'' \iff |C'| < |C''|$.

The importance of the adequate order is illustrated by the example in Fig. 3, which is taken from [12]. If one defines ✂ as

$$C \mathrel{\text{✂}} e \iff C \in \mathcal{C}_{loc} \wedge C \approx_{mar} [e] \wedge |C| \le \big|[e]\big| \wedge C \ne [e]$$

then one can generate the prefix shown in Fig. 3(b), where the events are numbered according to the order in which they were added to the prefix. The events e_8 and e_{10} are marked as cut-offs, because the final markings of $[e_8]$ and $[e_{10}]$ are $\{p_7, p_9, p_{10}\}$ and $\{p_6, p_8, p_{11}\}$, which are also the final markings of the local cutters $[e_7]$ and $[e_9]$, respectively. Although no events can now be added, the prefix is not complete, because the reachable marking $\{p_{12}\}$ of the Petri net in Fig. 3(a) is not represented in it. This demonstrates that using an adequate order is essential for the completeness of the prefix.

Moreover, a total adequate order $\lhd_{erv}$ was proposed in [12], which allowed to significantly reduce the size of the prefix in practice and guarantee the following theoretical upper bound on its size: the number of non-cut-off events in such a prefix can never exceed the number of reachable markings. Furthermore, the unfolding algorithm becomes deterministic if $\lhd$ is a total order, as in this case $\min_{\lhd} pe$ is a singleton and so the **select** operator ceases to be non-deterministic.

An important observation is that *the adequate order is used by the unfolding algorithm of [12] (and, as a special case thereof, [26]) both for checking whether an event newly added to the prefix is a cut-off and for deciding which possible extension of the prefix already built should be added to it (an event minimal w.r.t. $\lhd$ is selected)*. This guarantees the following important invariant of the algorithm:

If events e and f are added to the prefix and $[e] \lhd [f]$, then e is added earlier than f. (*)

However, applications required to push the boundaries of the unfolding technique beyond [12]. In particular:

- Unfolding PNs where reachable markings are not an adequate representation of the PN states, e.g. unfolding of Signal Transition Graphs [28] or Petri nets with symmetries.

– Allowing not just local, but arbitrary configurations as cutters [15].
– The parallel unfolding algorithm of [16], where the structure of the traditional algorithm was changed in an essential way, in order to inject more non-determinism — by introducing the techniques of slicing the set of possible extensions and producing cut-offs 'in advance', i.e. declaring an event a cut-off when a part of its cutter is not in the prefix yet.

These generalisations eventually led to the notion of a *canonical prefix* [22]. There, the cutting context had three parameters:

– The equivalence relation $\approx$, that was retained in this paper;
– The adequate order $\lhd$ similar to that in [12], but with the following essential change. As $\approx$ is now used in place of marking equivalence, the preservation by extensions condition in the definition of $\lhd$ changes as follows: if $C' \approx C''$ and $C' \lhd C''$ then for any suffix E'' of C'' there exists a suffix E' of C' such that $C' \oplus E' \approx C'' \oplus E''$ and $C' \oplus E' \lhd C'' \oplus E''$. Note that it is no longer required that these suffixes are isomorphic.
– The set of configurations $\mathcal{C}_e$ for each event e of Unf_{Σ}^{max}. The configurations in $\mathcal{C}_e$ are the only ones that can be considered as cutters of e, e.g. for efficiency reasons, many existing unfolding algorithms consider only local configurations when deciding whether an event should be designated as a cut-off event, and so $\mathcal{C}_e \overset{\text{df}}{=} \mathcal{C}_{loc}$. But one can also consider arbitrary configurations for such a purpose if the size of the resulting prefix is of paramount importance [15], in which case $\mathcal{C}_e \overset{\text{df}}{=} \mathcal{C}_{all}$. Other choices are possible and turn out to be useful in practice, see below.

The above form of a cutting context leads to the central result of [22], namely the *algorithm-independent* notion of a cut-off event and the related unique *canonical* prefix; the latter is shown to be complete, and some practical sufficient conditions which guarantee its finiteness are proposed; in particular, the finiteness is guaranteed when $\approx$ has a finite index and the cutting context is *dense*, i.e. $\mathcal{C}_{loc} \subseteq \mathcal{C}_e$ for each event e. Moreover, [22] shows that many existing unfolding algorithms [12,16,26], in spite of being non-deterministic, generate precisely this canonical prefix. The canonicity is related to the fact that these algorithms preserve the invariant (*) (or a similar one).

In this paper we generalise the setup of [22] in several important ways. Intuitively, the objective is to provide a *user-oriented view* of the unfolding technique, which simply tells which information will be preserved in the final prefix and how to declare an event a cut-off in the algorithm, while hiding the technical parameters like the adequate order. Furthermore, the extensions described above are naturally included in the proposed framework, whose main differences are summarised below:

– The cutting context is now a pair $(\approx, ✂)$ rather than a triple as in [22]. The adequate order $\lhd$ is no longer a part of the cutting context; instead, one merely postulates the existence of some well-founded relation $\ll$, which plays a role similar to that of $\lhd$, but is not necessarily an order, nor does it have to refine $\subset$. This is motivated by the fact that $\approx$ alone is sufficient to define the completeness of the prefix (see Definition 2), and so for guaranteeing completeness it is sufficient to know that an appropriate $\ll$ exists, without knowing it precisely.
– The order in which the unfolding algorithm processes the events in the set of possible extensions is disentangled from the adequate order used for declaring an event a cut-off. Hence the unfolding algorithm no longer preserves the invariant (*), and thus the canonicity result of [22] no longer holds in this framework (though it can be restored by restricting the selection strategy accordingly); nevertheless, the completeness of the resulting prefix and — subject to some natural conditions — its finiteness are still guaranteed, which is sufficient for many applications.

The theory of [22] can be embedded into our settings by choosing the cutting relation as

$$C ✂ e \iff C \in \mathcal{C}_e \wedge C \approx [e] \wedge C \lhd [e],$$

i.e. all three conditions required in [22] to declare an event a cut-off are simply included in ✂. Furthermore, the non-determinism in the **select** operator should be restricted by making it to choose an event from $\min_{\lhd} pe$ rather than the whole pe. Additionally, $\lhd$ is a suitable $\ll$ required by Definition 3: *CC:PE* follows from the preservation by extensions condition in [22], since $C ✂ e$ implies $C \approx [e]$ and $C \lhd [e]$. Hence, the completeness of the resulting prefix follows directly from Theorem 6.

For the finiteness of the prefix, a finite index and a dense context are assumed in [22]. Then the cutting context above is even finitary: *CC:CHAIN* is a relaxation of the requirement that $\lhd$ should contain $\subset$; we show this with a series of relaxations. The setting [2] relaxes this requirement with the following weaker one: in every infinite chain of configurations $C_1 \subset C_2 \subset C_3 \subset \cdots$ which are equivalent according to $\approx$, there are indices $i < j$ such that $C_i \lhd C_j$, which led to the notion of semi-adequate orders. Keijo Heljanko [priv. comm.] observed that this can be further relaxed to local configurations $[e_i]$, i.e.: for every infinite $\prec$-chain $e_1 \prec e_2 \prec e_3 \prec \cdots$ such that all the respective local configurations are related by $\approx$, there must be some $e_i \prec e_j$ (hence, $[e_i] \subset [e_j]$ and $[e_i] \in \mathcal{C}_{e_j}$) with $[e_i] \lhd [e_j]$; with our definition, we can write this as: there must be some $e_i \prec e_j$ with $[e_i] ✂ e_j$. This in turn can be relaxed to require that $C ✂ e_j$ for some e_j and $C \subset [e_j]$. The final relaxation is: there exist events e_k and $e \prec e_k$ and a configuration $C \subset [e]$ such that $C ✂ e$ (to see that this is a relaxation, choose $e_k = e_{j+1}$ and $e = e_j$).

Thus, there is a series of variations that all fit into our new setting. Some of these variations may be more useful in some cases than others, but our general finiteness and completeness theorems (Theorems 6 and 7) apply to all of them.

Additionally, in our setting $\ll$ is not required to be a partial order. However, this is not so important: the transitive closure of $\ll$ is irreflexive, since $\ll$ is asymmetric due to well-foundedness, and of course transitive, i.e. it is a strict partial order that still satisfies the conditions of Definition 3. Nevertheless, it might be convenient in practice that the transitivity of $\ll$ need not be proved.

The canonical prefixes framework of [22] turned out to be useful in practice. For example, [20,21] solves the problem of constructing the canonical unfolding prefix of a Petri net Σ' obtained by insertion of a new transition into a Petri net Σ whose canonical prefix is given. It turns out that the canonical unfolding prefix of Σ' can be constructed by applying local modifications to the existing prefix, avoiding thus expensive re-unfolding; however, the resulting prefix turns out to be canonical w.r.t. a different cutting context, and, moreover, re-unfolding Σ' with the original cutting context can yield a very different prefix. Due to these complications, it would be very difficult to formally justify the approach of [20,21] without the theory of canonical prefixes.

However, some applications of the unfolding technique, though fitting into the canonical prefixes framework, required some additional proofs and/or complicated definitions. In particular:

- Some applications use the equivalence $\approx_{mar}$ as the first parameter and $\subset$ instead of the adequate order, which helps in detecting cyclic runs: The local configuration $[e]$ of every cut-off event e is an extension of its cutter $C \subset [e]$, and since $C \approx_{mar} [e]$, $[e] \setminus C$ forms a cycle. However, $\subset$ is not preserved by extensions and so is not an adequate order. Still, it is possible to fit it into the framework of [22] by choosing $\lhd_m$ as the adequate order and adjusting the third component of the cutting context as $\mathcal{C}'_e \overset{\text{df}}{=} \{C \in \mathcal{C}_e \mid C \subset [e]\}$ for all events e, so that the adequate order is in effect 'trimmed' down to $\subset$. This way, the canonicity and completeness of such a prefix are guaranteed by the framework of [22], but its finiteness has to be proved separately, as the cutting context ceases to be dense.
  To fit this into our framework, one can simply choose the cutting relation as $C \,✂\, e \iff C \approx_{mar} [e] \wedge C \subset [e] \wedge C \in \mathcal{C}_e$ (then $\lhd_m$ can be used as an appropriate $\ll$). This way, both completeness and (in the case of a bounded PN) finiteness immediately follow from the theory developed in Section 5. Note that there is no need to impose any selection strategy, and the resulting prefix is canonical due to Theorem 9 below (its precondition holds as no two possible extensions are ordered by $\prec$).
- The modular prefix construction of [24] used $\subset$ instead of the adequate order, with some further restrictions; again, finiteness had to be proven separately. As above, this easily fits into the proposed framework, enabling thus the utilisation of the completeness and finiteness results of Section 5.
- In [2] the reachability problem was solved using AI techniques to direct the search; in effect, an adequate order was still used for identifying cut-offs, but an AI heuristic was used to select an event from the set of possible extensions. Complete disentangling of the cut-off criterion from the selection strategy offered by the theory developed in this paper is particularly useful in this situation, and in Section 8 we discuss how the setting of [2] fits into our framework.

6.1. DFS unfolding

Often, state exploration using depth-first search (DFS) is preferred for verification, in particular when the traces showing the violation of the property being checked are long; cf. also Section 8. Moreover, DFS is used for solving the emptiness problem of Büchi automata in LTL model checking, and, more generally, in applications requiring detection of cycles in the reachability graph. Hence, it is an important feature of the unfolding algorithm in Fig. 2 that the selection strategy can be depth-first. In such a case, the set of possible extensions *pe* can be viewed as a stack, with new possible extensions being pushed onto it, and then the one at the top of the stack being selected. (This only works for terminating runs of the algorithm, as otherwise the fairness assumption on the selection strategy is not satisfied.)

The possibility of the DFS selection strategy does not contradict the claim in [11] that DFS in general leads to incomplete prefixes: There, an event is declared a cut-off whenever some earlier added event has the same final marking, i.e. nothing like an adequate order is used. (Note that with such a cut-off criterion the breadth-first-like search illustrated in Fig. 3 also goes wrong.)

In Fig. 4, we show an example Petri net with the prefix according to the standard approach in [12] alongside with the DFS prefix. Observe that the two prefixes are not comparable w.r.t. the prefix relation, and that the DFS prefix has fewer events in this example. In general, however, there is no guarantee that the number of non-cut-off events in the DFS prefix never exceeds the number of reachable markings of the Petri net, as in the approach of [12]; in fact, the prefix of [12] is likely to be smaller than the DFS one in practice, as all the potential cutters of an event are added to the prefix before this event (unless preceded by a cut-off). On the other hand, updating the set of possible extensions (the most expensive part of the unfolding algorithm) is easier for the DFS strategy, as the top of the stack *pe* is likely to contain the last computed possible extension e, and so the context (e.g. the set of causal predecessors of e and the events which are in conflict with e) in which it was computed still persists and can be re-used for computing further possible extensions that are causal successors of e (they inherit both the causal predecessors and conflicts of e). Hence, practical evaluation of these two strategies of prefix generation is a promising topic for future research.

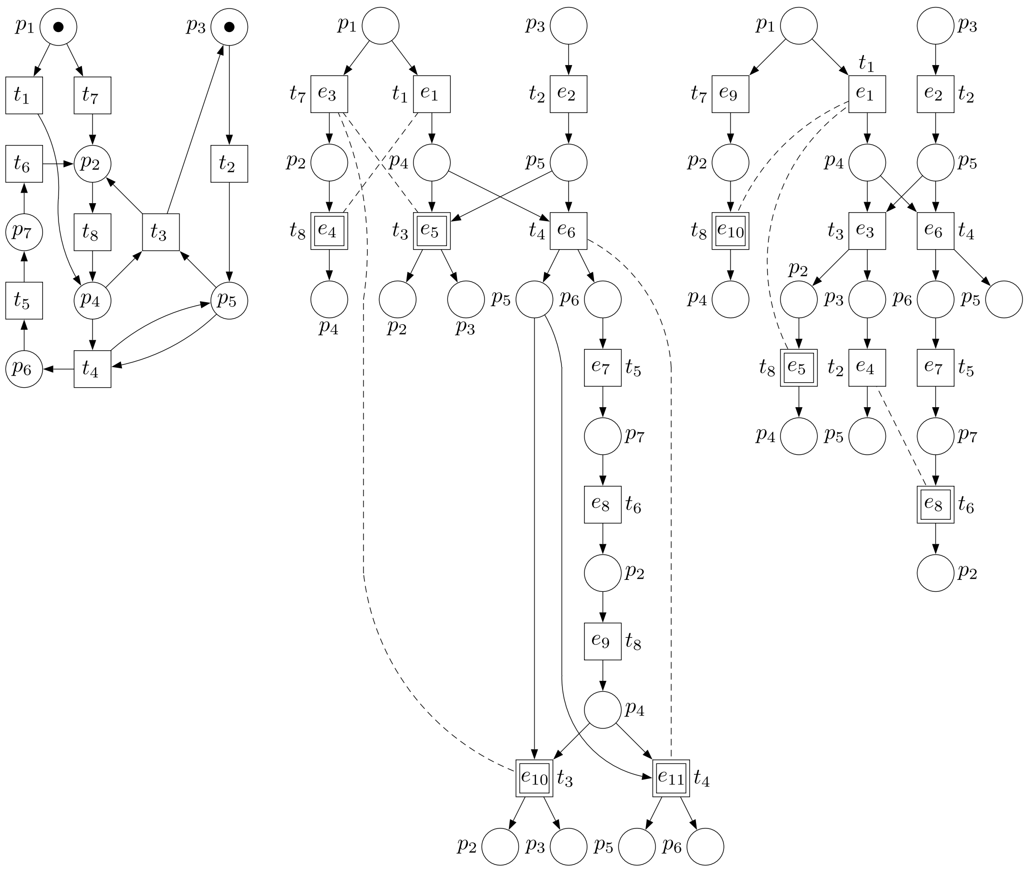

Fig. 4. A Petri net (left) and two complete prefixes of its unfolding, generated using the same cutting context based on $\approx_{mar}$, $\lhd_{erv}$ and $\mathcal{C}_{loc}$, but different selection strategies: the one based on selecting the minimal w.r.t. $\lhd_{erv}$ event (middle), and the DFS strategy (right). The events are numbered according to the order in which they have been added to the prefix, and if an event e was made cut-off due to some local configuration $[f]$ then e and f are connected by a dashed line.

7. Canonicity

The proposed setup, in contrast to that in [22], does not automatically ensure that the prefix generated by the unfolding algorithm is always the same, i.e. the canonicity is lost. This happens due to the total freedom on the order the algorithm pulls out events from the set of possible extensions. For example, suppose that at some stage $e, f \in pe$ with $[f] \mathrel{✂} e$ and the prefix does not contain any other cutters of e; then e can be either declared a cut-off (if the algorithm chooses to extract f first) or not (if e is extracted first). However, by restricting this freedom, i.e. imposing an appropriate selection strategy, one can restore the canonicity property.

One possibility is to require that all the events f of potential cutters C of e are in the prefix by the time e is pulled from the set of possible extensions (and thus, no new cutters of e can be found later). This can be ensured using the following invariant similar to (*):

The **select** statement does not select a possible extension f whenever there exist a possible extension e and a configuration C of Unf_{Σ}^{max} such that $e \in C \mathrel{✂} f$. (*′)

Theorem 9 *(Canonicity). If (*′) is an invariant of the unfolding algorithm in Fig. 2 then the resulting prefix is canonical, i.e. does not depend on the non-deterministic choices performed by the algorithm.*

Proof. For the sake of contradiction, suppose that two runs of the algorithm produced two different prefixes, π and π', or the same prefixes but different sets of cut-off events, *cut_off* and *cut_off*′. Hence, both prefixes contain an event f, which is a cut-off event in, w.l.o.g., π, and a non-cut-off in π'. Assume that f was added to π as the nth event and to π' as the n'th one, and that we have chosen the counterexample such that $n + n'$ is minimal.

When f was added to π, it was made a cut-off due to some C. When f was added to π', either some $e \in C$ was present and a cut-off or this was not the case but some $e \in C$ was missing. The former contradicts the choice of f, as e was added before f for both prefixes. The latter is impossible: initially, the minimal events of C were in pe; in case those were added to the prefix before f, some of their successors in C became possible extensions, etc. Thus, when f was added, the **select** statement preferred f to some possible extension $e \in C$, contradicting ($*'$). $\square$

Besides the loss of general canonicity, we do not provide any general bound on the size of the prefix, such as the number of reachable markings in [12]. Arbitrary selection can give good results in practice (see Section 8), but can in principle lead to very large prefixes; it seems that no interesting bound can be given without further assumptions in such a general framework as ours.

8. Directed unfolding

For some uses of unfolding, one is not interested in building a complete prefix but only in deciding whether or not the complete prefix includes a configuration with some specific property of interest, such as containing an event corresponding to a certain transition (to decide executability) or marking certain places (to decide coverability). This is the case with applications such as checking safety properties, AI planning [13], or diagnosing discrete event systems [5,25].

In this setting, it is common to use so called *on-the-fly* unfolding; this means that, instead of performing *standard unfolding* (i.e. the algorithm in Fig. 2) to the end, unfolding is terminated immediately when an event that completes a target configuration is selected from the queue of possible extensions. Traditionally, on-the-fly unfolding is done in breadth-first order, i.e. using the cardinality order or a refinement thereof [12]. However, one can use information about the set of target configurations to greatly enhance the efficiency of on-the-fly unfolding in deciding the property of interest. In heuristic state space search, this information typically comes in the form of a *heuristic function,* which provides a ranking of states indicating how "promising" they are, in terms of reaching a target state quickly or at low cost. In the unfolding setting, the heuristic is a function of configurations [2]. Heuristic functions can be obtained in a domain-independent fashion, from analysing the problem representation. This has been pioneered in the use of heuristic search for AI planning, e.g. [1,14,18], and methods of deriving domain-independent heuristics from planning problem representations easily translate to the Petri nets formalism [2]. The term *directed unfolding* refers to the on-the-fly unfolding guided by a heuristic function.

In this section, we discuss three possible uses for heuristic information:

– guiding the unfolding towards a target configuration;
– pruning dead ends (configurations that cannot be extended to a target configuration); and
– finding a target configuration that is guaranteed to be optimal, according to various measures of cost.

We will show how the theory developed in this paper simplifies the implementation of all three.

Let $\mathcal{C}_{target}$ denote the set of target configurations in the full unfolding. *We require* that $\mathcal{C}_{target}$ is *closed under* $\approx$, i.e. if $C \approx C'$ then either both are in $\mathcal{C}_{target}$ or both are not. This is essentially a restriction on the equivalence relation: we cannot reasonably consider two configurations to be equivalent if the question we are interested in is whether we can reach one of them but not the other.

A *heuristic* is a function h from configurations to some domain of non-negative numbers together with the special value ∞, e.g. $h : \mathcal{C}_{all} \to \mathbb{R}^{\geq 0} \cup \{\infty\}$. The interpretation of the number $h(C)$ that the heuristic assigns to a configuration depends on the purpose: If the aim is simply to guide the unfolding process to a target configuration as quickly as possible, it is an estimate of the number of events that need to be added to C to extend it into a target configuration. To make use of this information, the unfolding algorithm simply orders the set of possible extensions by their heuristic values and always selects an event e with smallest $h([e])$. Since this changes only the selection strategy, Theorems 6 and 7 ensure the completeness and termination of the algorithm with any suitable cutting context. If $\mathcal{C}_{target}$ is defined as the set of local configurations of events labelled with a distinguished target transition, then, under certain assumptions on the heuristic, it can be shown that a strictly better informed heuristic will never produce a larger prefix [2]. In particular, since breadth-first unfolding corresponds to a maximally uninformed heuristic, which ranks events only by their history without considering their distance to the goal, heuristically guided unfolding will never perform worse than blind unfolding.

The impact of heuristic guidance in practice, however, is much greater than what is ensured by this worst-case theoretical result. As an example, Fig. 5 (from [2]) shows how many queries were answered with a given maximal size of the prefix generated by breadth-first unfolding (solid line) and using two different heuristics (other lines) on a collection of random transition executability problems, including both solvable (upper graph) and unsolvable (lower graph). The use of heuristic information drastically reduces the number of events that need to be processed to answer the executability question in both cases: directed unfolding solves all problems, using prefixes two orders of magnitude smaller than breadth-first unfolding, which also runs out of memory on the hardest problems.

8.1. Dead end pruning

The reason why heuristic guidance improves unfolding also on unsolvable problems is that the heuristic can identify *dead ends,* that is, configurations which cannot be extended into any target configuration. Note that a dead end is not the same as a deadlock: possibly, a dead end can be extended, even arbitrarily many times; just none of these extensions is in $\mathcal{C}_{target}$. Let $\mathcal{C}_{dead}$ be the set of dead end configurations. It follows directly from the definition that $\mathcal{C}_{dead}$ is closed w.r.t. extensions.

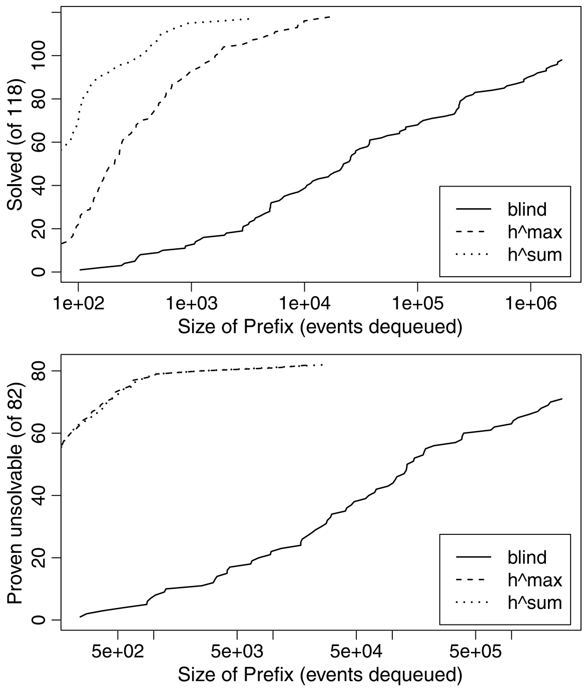

Fig. 5. The impact of heuristic guidance on the size of the prefix generated by on-the-fly unfolding.

As above for $\mathcal{C}_{target}$, we require that $\mathcal{C}_{dead}$ is closed under the equivalence relation $\approx$, i.e. if $C \approx C'$ then either both C and C' belong to $\mathcal{C}_{dead}$ or both do not. Note that this requirement automatically holds for $\approx_{mar}$: Consider two configurations C and C' such that $C \notin \mathcal{C}_{dead}$ and $C \approx_{mar} C'$; then $C \oplus E \in \mathcal{C}_{target}$ for some suffix E of C. Further, there is a suffix E' of C' that is isomorphic to E, and so $C \oplus E \approx_{mar} C' \oplus E'$. Since $\mathcal{C}_{target}$ is closed w.r.t. $\approx_{mar}$, this implies $C' \oplus E' \in \mathcal{C}_{target}$ and $C' \notin \mathcal{C}_{dead}$.

We say that a heuristic is *safely pruning* if $h(C) = \infty$ implies that $C \in \mathcal{C}_{dead}$. Most planning heuristics are safely pruning, including the two used in the experiments shown in Fig. 5, so *we assume* that we are given such an h. Note that there is no requirement that the heuristic identifies all dead ends; it is sufficient for it to be sound w.r.t. dead end detection.

If $[e] \in \mathcal{C}_{dead}$ then not adding e to the prefix does not change whether a target configuration can be reached or not, i.e. this preserves completeness w.r.t. $\mathcal{C}_{target}$. This is naturally implemented by a simple change to the unfolding algorithm: Whenever an event e with $h([e]) = \infty$ is selected from the set of possible extensions, it is simply discarded instead of being added to the prefix, and so no causal successors of e are generated. This has the same effect as declaring, ad hoc, e to be a cut-off event. If the set pe in the unfolding algorithm in Fig. 2 is maintained as a priority queue with the heuristic values as priorities, all dead end events will be at the end of the queue. Thus, when the dequeued event is a dead end, so are all the remaining events in pe, and the unfolding algorithm can terminate right away.

To demonstrate that this modification preserves completeness w.r.t. $\mathcal{C}_{target}$ and finiteness, we will show that it can be simulated by running our standard unfolding algorithm with a transformed net and cutting context, and that this new cutting context has the required properties. This also demonstrates the flexibility of our approach.

Assume that a net N and a suitable cutting context Θ are given. Now we transform the net as follows: We empty all places and add a new marked place p_{init}, followed by the new transition lo (“lights-on”) that produces the initial marking of the original net. (For simplicity, we assume here that the initial marking is safe; the treatment of the general case is similar, just using several lo transitions.) The idea is that the configuration $\emptyset$, which marks only (and is the only one to mark) p_{init}, represents all dead end configurations (although it is not itself a dead end); an important aspect is that $\emptyset$ will be present in all prefixes.

We transform Θ in two steps: In the first, we add lo to all configurations, e.g. if we had $C \text{✂} e$ before, then we have $C \cup \{lo\} \text{✂} e$ now — and $\emptyset$ is never a cutter; additionally, $\emptyset$ forms an $\approx$-class of its own. The new Θ can easily be seen to be a cutting context (observe that lo is added to any $[e]$ automatically), and all relevant features remain: the unfolding algorithm (with or without dead end detection) still works in the same way; the assumptions regarding $\approx$ still hold, e.g. the new Θ is finitary iff the old Θ was, etc. We now regard this as the original setting and denote it with $\Theta \overset{\text{df}}{=} (\approx, \text{✂})$.

In the second step, the modified cutting context Θ' is obtained as follows:

- $\ll' \overset{\text{df}}{=} \ll \cup \{(\emptyset, C) \mid C \neq \emptyset\}$ (clearly, $\ll'$ is still well-founded);
- $C \text{✂}' e$ if either $C = \emptyset$ and $h([e]) = \infty$, or $C \neq \emptyset$, $h([e]) \neq \infty$ and $C \text{✂} e$ (this is decidable);

– $C \approx' C'$ if either C and C' are both in $\mathcal{C}_{dead} \cup \{\emptyset\}$, or $C \approx C'$ and they are both not in $\mathcal{C}_{dead} \cup \{\emptyset\}$. Clearly, $\approx'$ is an equivalence: we have simply collected all configurations in $\mathcal{C}_{dead} \cup \{\emptyset\}$ into one equivalence class of $\approx'$, which is the union of some equivalence classes of $\approx$ by the assumption that $\mathcal{C}_{dead}$ is closed w.r.t. $\approx$. Note that we have only enlarged $\approx$ to $\approx'$.

The standard unfolding algorithm (without dead end detection) now declares events to be cut-offs if they are detected dead ends, so it simulates the algorithm with dead end pruning. The equivalence classes of configurations outside $\mathcal{C}_{dead}$, and hence of those in $\mathcal{C}_{target}$, remain unchanged. Furthermore, if $\approx$ has a finite index, then $\approx'$ has a finite index, too.

The following theorem implies that any prefix generated using Θ' is complete and, in case of a finitary Θ, also finite. Thus, any prefix generated with Θ' (or, equivalently, with dead end pruning) contains an $\approx$-representative for each target configuration.

Theorem 10. *Θ' is a cutting context and finitary if Θ is.*

Proof. To show *CC:PE* for Θ', suppose suitable C, e and E are given in the new setting.

By definition of $\between'$, if $C = \emptyset$ then $h([e]) = \infty$, i.e. $[e]$ is a detected dead end. Thus, $[e] \oplus E \in \mathcal{C}_{dead}$ as $\mathcal{C}_{dead}$ is closed w.r.t. extensions. We take $E' = \emptyset$, and have $C \oplus E' = \emptyset \approx' [e] \oplus E$ and $C \oplus E' = \emptyset \ll' [e] \oplus E$.

If $C \neq \emptyset$ then $h([e]) \neq \infty$ and $C \between e$ by definition of $\between'$. Consider E' according to Θ. Now $C \oplus E' \ll' [e] \oplus E$ and $C \oplus E' \approx' [e] \oplus E$ hold, since $\ll$ and $\approx$ are subrelations of $\ll'$ and $\approx'$, respectively.

If Θ is finitary, then also $\approx'$ has a finite index as noted above. Hence it remains to show the *CC:CHAIN* condition for Θ'.

Let X be an infinite $\prec$-chain of events whose local configurations are equivalent w.r.t. $\approx'$; if X contains *lo*, remove it from X. Since $\approx$ has a finite index, there exists an infinite subchain $X' \subseteq X$ in which all local configurations are related by $\approx$, and we can consider C, e and f such that $C \between e \prec f \in X'$ and $C \subset [e]$ according to *CC:CHAIN* condition for Θ. Note that $C \neq \emptyset$ since $\emptyset$ is never a cutter according to Θ by its construction.

If $h(e) \neq \infty$ then $\emptyset \neq C \subset [e]$ and $C \between' e \prec f \in X' \subseteq X$. If $h(e) = \infty$ then $\emptyset \subset [e]$ and $\emptyset \between' e \prec f \in X' \subseteq X$. In either case, *CC:CHAIN* holds for Θ'. □

8.2. Optimal unfolding

The final use of heuristic information we consider is searching for a target configuration with guaranteed minimal cost. This problem arises when we seek, for example, the shortest counter-example to a violated safety property, or the cheapest or quickest plan. We consider a *monotonic* cost function $cost : \mathcal{C}_{all} \to \mathbb{R}^{\geq 0}$, i.e. $C \subseteq C' \Rightarrow cost(C) \leq cost(C')$.

In this section, *we assume* that the *equivalence relation* is $\approx_{mar}$, i.e. marking equivalence, and that the property we are interested in is coverability, i.e. finding an optimal configuration whose marking covers a given target marking M_{target}. We apply the standard encoding trick of adding a designated target transition that consumes M_{target}; here, the target transition additionally consumes the unique token of an additional place p_{target}. We call an event labelled by this transition a *target event*. Then, we can define $\mathcal{C}_{target}$ as the configurations that contain a target event. They are characterised by the fact that p_{target} is empty under their final markings; thus, $\mathcal{C}_{target}$ is closed under $\approx_{mar}$, satisfying our general assumption.

We extend the cost function to configurations $C \oplus \{e\}$ with e being a target event by setting $cost(C \oplus \{e\}) \overset{\mathrm{df}}{=} cost(C)$; think of the target transition as having zero cost. This way, minimal cost target configurations correspond bijectively to minimal cost configurations whose marking covers M_{target}; moreover, by monotonicity of *cost*, we can restrict attention to local configurations of target events. Completeness ensures that our unfolding algorithm generates a target configuration whenever M_{target} is coverable.

The following lemma (a restated version of [17, Theorem 4.2.2]) states the conditions on the cutting relation and the event selection strategy that ensure on-the-fly unfolding returns an optimal configuration.

Lemma 11. *Consider a cutting context and a selection strategy. Assume that, in each run of the standard unfolding algorithm, (i) at least one minimum cost target configuration is generated, and (ii) no non-optimal target event is selected before an event that is part of some optimal target configuration. Then the configuration found by the respective on-the-fly unfolding algorithm has minimum cost.*

Proof. Consider a run and let $C^\star$ be an optimal target configuration generated in this run. Initially, at every iteration of the unfolding algorithm, the set of possible extensions contains all minimal events among the events of $C^\star$ not generated yet. This invariant could only be wrong if some causal predecessor of such a minimal event were a cut-off; but in this case, $C^\star$ would never be generated. The invariant remains true in the standard run until the target event in $C^\star$ is selected; at this stage, at the latest, the on-the-fly unfolding algorithm stops. By condition (ii), no non-optimal target event is selected before. Hence, the on-the-fly unfolding algorithm stops when it selects an optimal target event. □

From Theorem 6 we know that unfolding with a cutting context $\Theta = (\approx, \between)$ generates a prefix that contains a $\ll$-minimal configuration from each equivalence class of $\approx$, where $\ll$ is a well-founded relation whose existence is postulated by Definition 3. In particular, since $\mathcal{C}_{target}$ is closed under $\approx$, if Unf_{Σ}^{max} contains any target configuration then the generated prefix contains a $\ll$-minimal target configuration. Thus, if we can choose $\ll$ such that

$$C \ll C' \Rightarrow cost(C) \le cost(C') \qquad (^{**})$$

then condition (i) of Lemma 11 is ensured. Furthermore, since Theorem 6 holds for any selection strategy, we are free to choose one that ensures condition (ii). For example, one can always select an event from *pe* whose local configuration has minimum cost: If there is a possible extension e belonging to some optimal target configuration $C^\star$ and another possible extension e' being a non-optimal target event, then by monotonicity $cost([e]) \le cost(C^\star) < cost([e'])$.

This selection strategy can be improved using a heuristic. In this setting, the heuristic value $h(C)$ is an estimate of the least cost of any extension of C that belongs to $\mathcal{C}_{target}$. W.l.o.g., *we assume* $h(C) \ge cost(C)$, since any extension of C must cost at least as much as C. (Readers familiar with heuristic search should think of h here as the f-value, i.e. the combination of accumulated cost and estimated cost-to-go.)

A heuristic h is said to be *admissible* iff $h(C)$ is at most the minimum possible cost of any extension $C' \in \mathcal{C}_{target}$ of C. Consequently, $h(C) = \infty$ means that there is no such extension, hence an admissible heuristic is by definition safely pruning. Given such a heuristic, one can always select a possible extension whose estimate is minimal. This also ensures Lemma 11(ii), since, for e and e' as above, the admissibility implies $h([e]) \le cost(C^\star) < cost([e']) \le h([e'])$.

This is an example of where the flexibility provided by the disentanglement of the selection strategy from the cut-off criterion in our framework offers a clear advantage over previous unfolding techniques.

The remaining question is how to define a relation $\ll$ and a matching cutting context so that (**) is satisfied. We will consider two different cost functions; in both cases, we associate to each transition t a fixed value $cost(t) \in \mathbb{R}^{\ge 0}$, which is 0 for the target transition. The first is *additive cost,* where the cost of a configuration C is $cost(C) \overset{\text{df}}{=} \sum_{e\in C} cost(\varphi(e))$. Note that cardinality is a special case of additive cost, with $cost(t) \overset{\text{df}}{=} 1$ for each transition t except the (artificial) target one. Additive cost also models many other situations, such as minimising the size of counterexamples in model checking or minimising the number of faults in discrete event diagnosis. The second cost function is execution time, also known as “makespan”; for this, we interpret $cost(t)$ as the duration of t. Since configurations are acyclic and conflict-free, the optimal execution time is given by summing up the durations along each causal chain in the configuration and taking the maximum of all such sums. Both cost measures occur frequently in AI planning problems, where optimising the plan quality is important; they both satisfy $cost(C \oplus \{e\}) = cost(C)$ for each configuration C with e being a target event.

The additive-cost case looks simple. We define $C \ll C'$ iff

$$cost(C)<cost(C') \vee (cost(C)=cost(C') \wedge |C|<|C'|).$$

This relation clearly satisfies (**), but well-foundedness is not so obvious: the values $cost(C)$ are in $\mathbb{R}^{\ge 0}$, which is not well-founded w.r.t. $<$. One key observation is that these values are sums built from the finitely many $cost(t)$, $t \in T$. If these were rational numbers, we could write them with the same denominator and then we would essentially be in the natural numbers domain. This can still be achieved in our more complicated setting as follows.

Each value $cost(C)$ is some $\sum_{t\in T'} x_t cost(t)$, where T' is the set of transitions with positive cost and $x_t \in \mathbb{N}$. Consider some $r \in \mathbb{R}^{\ge 0}$. Then, for each $t \in T'$, there exists $n_t \in \mathbb{N}$ such that $n_t cost(t) > r$; hence, $x_t < n_t$ in any sum whose value does not exceed r. Since the set of vectors $\{(x_t) \mid t \in T' \wedge x_t < n_t\} \in \mathbb{N}^{|T'|}$ is finite, there are only finitely many sum values not exceeding r. This means that there exists an order-preserving bijection from $\mathbb{N}$ to the set of possible sum values, i.e. the relation $<$ is well-founded on the latter set, and hence on the set of possible configuration costs.

Now suppose $C_1, C_2, \ldots$ is an infinite sequence decreasing w.r.t. $\ll$. Since $<$ is well-founded on the set of configuration costs, the sequence $cost(C_i)$ stabilises. Then, the cardinality of the configurations has to decrease indefinitely, which is impossible. Hence $\ll$ is well-founded.

Next, we extend $\approx_{mar}$ to a cutting context by defining $C \,✂\, e$ if $C \approx_{mar} [e] \wedge C \ll [e]$; this is decidable. For *CC:PE* (see Definition 3), we choose E' as in the standard unfolding with $\approx_{mar}$; so i) holds, and ii) does as well since both the costs and the number of events grow by the same value when adding the suffixes. For bounded nets, $\approx_{mar}$ has a finite index, and we also have *CC:CHAIN*: For any two events $e \prec f$ in an infinite $\prec$-chain of events whose local configurations are equivalent w.r.t. $\approx_{mar}$, we have $[e] \ll [f]$ (observe that the costs of the local configurations could be equal due to events with cost 0), and thus $[e] \,✂\, f$. Note that essentially the same reasoning works if the cutting relation is defined as $C \,✂\, e$ if $C \in \mathcal{C}_{loc} \wedge C \approx_{mar} [e] \wedge C \ll [e]$.

Optimising makespan is more complicated, as *CC:PE* does not hold for the relation $\ll$ defined as above [17]. Instead, we define this relation as follows. For each condition $c \in Cut(C)$, let $cost(C, c)$ be the maximal sum of transition durations along causal chains ending in c. We then define a *point-wise* makespan comparison on configurations as $C \le_{pw} C'$ iff $C \approx_{mar} C'$ and there exists a label-preserving bijection $\psi : Cut(C) \to Cut(C')$ such that $cost(C, c) \le cost(C', \psi(c))$ for all $c \in Cut(C)$. Note that $\le_{pw}$ is only defined for configurations with equal final markings, and that $C \le_{pw} C'$ implies $cost(C) \le cost(C')$ since $cost(C)$ is the maximum among the $cost(C, c)$. Finally, we define $\ll$ as follows: $C \ll C'$ iff

– $C \le_{pw} C'$ and $C' \not\le_{pw} C$; or
– $C \le_{pw} C'$ and $C' \le_{pw} C$ and $|C| < |C'|$.

This relation satisfies (**) as just noted. Observe that, in case $C \le_{pw} C'$ and $C' \le_{pw} C$, the two bijections on conditions can be chosen such that one is the inverse of the other. To show well-foundedness of $\ll$, assume there exists an infinite sequence $C_1, C_2, \ldots$ decreasing w.r.t. $\ll$. For each condition $c \in Cut(C_1)$, the bijections according to $\le_{pw}$ give a weakly decreasing chain of values $cost(C_i, c)$. These values are sums as considered in the additive-cost case, so each of these chains stabilises on some value. Once all chains have stabilised, the cardinalities of the configurations have to decrease indefinitely, which is impossible.

In the same way as above, but with the different $\ll$, we extend $\approx_{mar}$ to a cutting context by defining $C ✄ e$ if $C \approx_{mar} [e] \wedge C \ll [e]$ (again, the reasoning below will also work if the definition of ✄ is modified by additionally requiring that $C \in \mathcal{C}_{loc}$); this is decidable. For *CC:PE* (see Definition 3), we again choose E' as in the standard unfolding with $\approx_{mar}$, and so i) holds. To show ii), consider the addition of one event of E after the other. Hence, we have some C and some D (initially $[e]$) such that $C \le_{pw} D$ (and possibly $D \le_{pw} C$ with the inverse bijection ψ^{-1}) and we add a new event f to D; then we can add an event f' to C such that $\varphi(f) = \varphi(f')$ and ${}^\bullet f = \psi({}^\bullet f')$. We modify ψ by removing the correspondences between ${}^\bullet f$ and ${}^\bullet f'$ and adding the correspondences between $f^\bullet$ and $f'^\bullet$. This also modifies ψ^{-1} in a suitable way, and the cardinalities of the two configurations have grown by 1. Thus, by induction on $|E|$, *CC:PE* holds.

For bounded nets $\approx_{mar}$ has a finite index, and we also have *CC:CHAIN*. But this is not as easy to see as in the additive-cost case since $e \prec f$ does not imply $[e] \le_{pw} [f]$ even if $[e] \approx_{mar} [f]$: Indeed, suppose there is a unique condition $c \in Cut([e])$ labelled with some place p. When extending $[e]$ to $[f]$, it might well happen that one adds a concurrent condition c' also labelled with p such that $cost([f], c') < cost([e], c)$ and then 'consumes' c with some causal successor. We proceed as follows.

Consider an infinite $\prec$-chain X of events whose local configurations are equivalent w.r.t. $\approx_{mar}$, and let M be the common final marking of these configurations. For each $e \in X$ we form a vector $v_e \overset{\text{df}}{=} (cost([e], c))|_{c \in Cut([e])}$, whose elements are ordered by $\varphi(c)$, according to some fixed ordering of the places occurring in M (it does not matter how the $cost([e], c)$ for equally labelled c are arranged). Then we apply Dickson's Lemma [7] to this infinite set of vectors, which yields two events $e \prec f$ such that $v_e \le v_f$; this implies $[e] \le_{pw} [f]$ and, since configuration size increases, also $[e] \ll [f]$. Finally, we have $[e] ✄ f$.

To summarise, we have shown how our new approach to unfolding can be used to find minimum cost target configurations for two types of configuration costs: in both cases, we have demonstrated how to extend $\approx_{mar}$ to a suitable cutting context. Furthermore, since the selection strategy is disentangled from the cut-off condition in our approach, efficiency can be improved by using admissible heuristics.

9. Conclusions and future work

We have proposed a new Petri net unfolding framework, which is the most general to date. The generalisations include the user-oriented view of the unfolding technique, the relaxation of the notions of adequate order and cutting context, and the disentanglement of the selection strategy from the cut-off condition. The completeness, finiteness, and (subject to an extra condition) canonicity of the resulting prefix are proved in this general setting. The developed theory has a number of important applications, in particular, it naturally enables the DFS selection strategy and heuristically-guided unfolding.

In future research, it would be interesting to compare experimentally the efficiency of DFS prefix generation with that of the standard approach of [12]. Moreover, the disentanglement of the selection strategy from the cut-off condition is likely to streamline the correctness proofs in the unfolding-based LTL-X model checking approach of [10], where a complicated adequate order had to be defined to impose a particular selection strategy. Furthermore, the extra flexibility offered by the proposed framework may lead to improvements in that approach.

Acknowledgements

The authors would like to thank Javier Esparza and Keijo Heljanko for helpful comments.

This research was supported by the Royal Academy of Engineering/EPSRC post-doctoral research fellowship EP/C53400X/1 (DAVAC), and EPSRC grants EP/G037809/1 (VERDAD) and EP/K001698/1 (UNCOVER). NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

[1] B. Bonet, H. Geffner, Planning as heuristic search, Artificial Intelligence 129 (1–2) (2001) 5–33.
[2] B. Bonet, P. Haslum, S. Hickmott, S. Thiébaux, Directed unfolding of Petri nets, in: Transactions on Petri Nets and Other Models of Concurrency (ToPNoC), 2008, pp. 172–198.
[3] T.-A. Chu, Synthesis of self-timed VLSI circuits from graph-theoretic specifications, Ph.D. thesis, Lab. for Comp. Sci., MIT, 1987.
[4] E. Clarke, O. Grumberg, D. Peled, Model Checking, MIT Press, 1999.
[5] M. Cordier, S. Thiébaux, Event-based diagnosis for evolutive systems, in: Proc. DX, 1994, pp. 64–69.

[6] J.-M. Couvreur, S. Grivet, D. Poitrenaud, Unfolding of products of symmetrical Petri nets, in: Proc. Petri Nets, in: LNCS, vol. 2075, Springer-Verlag, 2001, pp. 121–143.
[7] L. Dickson, Finiteness of the odd perfect and primitive abundant numbers with *n* distinct prime factors, Amer. J. Math. 35 (4) (1913) 413–422.
[8] R. Diestel, Graph Theory, Springer-Verlag, 2005.
[9] J. Engelfriet, Branching processes of Petri nets, Acta Inform. 28 (6) (1991) 575–591.
[10] J. Esparza, K. Heljanko, Unfoldings: A Partial-Order Approach to Model Checking, EATCS Series: Monographs in Theoretical Computer Science, Springer-Verlag, 2008.
[11] J. Esparza, P. Kanade, S. Schwoon, A negative result on depth-first net unfoldings, Int. J. Softw. Tools Technol. Transf. 10 (2) (2007) 161–166.
[12] J. Esparza, S. Römer, W. Vogler, An improvement of McMillan's unfolding algorithm, Form. Methods Syst. Des. 20 (3) (2002) 285–310.
[13] M. Ghallab, D. Nau, P. Traverso, Automated Planning: Theory and Practice, Morgan Kaufmann Publishers, ISBN 1-55860-856-7, 2004.
[14] P. Haslum, H. Geffner, Admissible heuristics for optimal planning, in: Proc. AIPS, AAAI Press, 2000, pp. 140–149.
[15] K. Heljanko, Minimizing finite complete prefixes, in: CS&P, 1999, pp. 83–95.
[16] K. Heljanko, V. Khomenko, M. Koutny, Parallelisation of the Petri net unfolding algorithm, in: Proc. TACAS, in: LNCS, vol. 2280, Springer-Verlag, 2002, pp. 371–385.
[17] S. Hickmott, Directed unfolding: reachability analysis of concurrent systems & applications to automated planning, Ph.D. thesis, University of Adelaide, Australia, 2008.
[18] J. Hoffmann, B. Nebel, The FF planning system: fast plan generation through heuristic search, J. Artificial Intelligence Res. 14 (2001) 253–302.
[19] V. Khomenko, Model checking based on prefixes of Petri net unfoldings, Ph.D. thesis, School of Computing Science, Newcastle University, 2003.
[20] V. Khomenko, Behaviour-preserving transition insertions in unfolding prefixes, in: Proc. Petri Nets, in: LNCS, vol. 4546, Springer-Verlag, 2007, pp. 204–222.
[21] V. Khomenko, A new type of behaviour-preserving transition insertions in unfolding prefixes, in: Proc. ICGT, in: LNCS, vol. 6372, Springer-Verlag, 2010, pp. 75–90.
[22] V. Khomenko, M. Koutny, W. Vogler, Canonical prefixes of Petri net unfoldings, Acta Inform. 40 (2) (2003) 95–118.
[23] D. König, Über eine Schlußweise aus dem Endlichen ins Unendliche, Acta Sci. Math. (Szeged) 3 (1927) 121–130, Bibliography in: Theorie der endlichen und unendlichen Graphen. Teubner, Leipzig, 1936 (reprinted 1986).
[24] A. Madalinski, E. Fabre, Modular construction of finite and complete prefixes of Petri net unfoldings, in: Proc. ACSD, IEEE Comp. Soc. Press, 2008, pp. 68–79.
[25] S. McIlraith, Toward a theory of diagnosis, testing and repair, in: Proc. DX, 1994, pp. 185–192.
[26] K. McMillan, Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits, in: Proc. CAV, in: LNCS, vol. 663, Springer-Verlag, 1992, pp. 164–177.
[27] L. Rosenblum, A. Yakovlev, Signal graphs: from self-timed to timed ones, in: Proc. Workshop on Timed Petri Nets, IEEE Comp. Soc. Press, 1985, pp. 199–206.
[28] A. Semenov, Verification and synthesis of asynchronous control circuits using Petri net unfolding, Ph.D. thesis, Department of Computing Science, Newcastle University, 1997.