Canonical Prefixes of Petri Net Unfoldings

Victor Khomenko[1], Maciej Koutny[1], and Walter Vogler[2]

[1] Department of Computing Science, University of Newcastle
Newcastle upon Tyne NE1 7RU, U.K.
{Victor.Khomenko, Maciej.Koutny}@ncl.ac.uk

[2] Institut für Informatik, Universität Augsburg
D-86135 Augsburg, Germany
Walter.Vogler@informatik.uni-augsburg.de

Abstract. In this paper, we develop a general technique for truncating Petri net unfoldings, parameterized according to the level of information about the original unfolding one wants to preserve. Moreover, we propose a new notion of completeness of a truncated unfolding, which with the standard parameters is strictly stronger than that given in [5, 6], and is more appropriate for the existing model checking algorithms.
A key aspect of our approach is an *algorithm-independent* notion of cut-off events, used to truncate a Petri net unfolding in the existing model checking systems. Such a notion is based on a *cutting context* and results in the unique *canonical* prefix of the unfolding. We show that the canonical prefix is complete in the new, stronger sense, and provide necessary and sufficient conditions for its finiteness, as well as upper bounds on its size in certain cases. We then consider the unfolding algorithm presented in [5, 6], and the parallel unfolding algorithm proposed in [10]. We prove that both algorithms, despite being non-deterministic, generate the canonical prefix. This gives an alternative correctness proof for the former algorithm, and a new (much simpler) proof for the latter one.
Keywords: Model checking, Petri nets, unfolding, canonical prefix.

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., [1, 2]) — 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 help in coping with this, a number of techniques have been proposed, which can roughly be classified as aiming at a compact representation of the full state space of a reactive concurrent system, or at generation of its reduced (though sufficient for a given verification task) representation. McMillan's (finite prefixes of) Petri Net unfoldings (see, e.g., [5, 15]) rely on the partial order view of concurrent computation, and represent system's actions and local states implicitly, using an acyclic net. In this paper, we show that this technique can be combined with the reduced representation of the state space, if one specifies which information must be preserved for a particular model checking task.

The essential feature of the existing unfolding algorithms (see, e.g., [5, 6, 10, 15]) is the use of cut-off events beyond which the unfolding starts to repeat itself, and so can be truncated without loss of information. So far, cut-off events were considered as an algorithm-specific issue, and were defined w.r.t. the part of the prefix already built by an unfolding algorithm. A first main contribution of this paper is to define an *algorithm-independent* notion of a cut-off

event and the related *canonical* prefix of a Petri net unfolding. We present conditions which guarantee the finiteness of such a prefix and, in doing so, prove a version of König's Lemma for branching processes of (possibly unbounded) Petri nets.

When constructing a prefix, one wants to ensure that it is in some sense complete, i.e., that it preserves enough information to efficiently decide certain properties. Completeness of a prefix is usually defined w.r.t. the set of reachable markings, i.e., a complete prefix comprises configurations representing all the reachable markings of the original Petri net. However, there are alternative definitions. For example, one might be interested not only in markings, but also in the ways in which they have been reached (see, e.g., [13, 18]), or exploit symmetries present in the original net system in order to reduce the size of the prefix (see, e.g., [3]). Moreover, for efficiency reasons, the usual versions of the unfolding algorithm only consider local configurations when deciding whether an event should be designated as a cut-off event; but one can also consider arbitrary finite configurations for such a purpose (see, e.g., [7]). A second main contribution of this paper is that we generalise the entire set-up and define cut-offs and completeness in an abstract parametric setting: one parameter captures the information we intend to retain in a complete prefix, while the other specify under which circumstances an event can be designated as a cut-off event. This approach results both in a more elegant presentation, and a powerful and flexible tool to deal with different variants of the unfolding technique.

The notion of a complete prefix (i.e., one which contains enough information to re-construct the entire unfolding) given in [5, 6] did not mention cut-off events at all. But, with the development of model-checking algorithms based on unfoldings, it appeared that cut-off events are heavily employed by almost all of them. Indeed, the deadlock detection algorithm presented in [15] is based on the fact that a Petri net is deadlock-free iff each configuration of its finite and complete prefix can be extended to one containing a cut-off event, i.e., a Petri net has a deadlock iff there is a configuration which is in conflict with all cut-off events. The algorithms presented in [8, 9, 11, 12, 17] use the fact that there is a certain correspondence between the deadlocked markings of the original net and the deadlocked markings of a finite and complete prefix, and cut-off events are needed to distinguish the 'real' deadlocks from the 'fake' ones, introduced by truncating the unfolding. Moreover, those algorithms need a stronger notion of completeness than the one presented in [5, 6], in order to guarantee that deadlocks in the prefix do correspond to deadlocks in the original Petri net.[1] Since all these algorithms make certain assumptions about the properties of a prefix with cut-off events, there is a clear need to formally define cut-off events and complete prefixes containing them, in order to close this rather uncomfortable gap between theory and practice.

The above two aspects of this paper are tied together by showing that the canonical prefix is 'special' in the sense that, for all parameter values, it is complete in a stronger sense than the notion of completeness used in [5, 6], and is generated by an arbitrary run of the (non-deterministic) unfolding algorithm presented in [5, 6] as well as an arbitrary run of the (even more non-deterministic) parallel algorithm from [10]. This results in an alternative and, in the latter case, much simplified correctness proof.

2 Basic Notions

In this section, we first present basic definitions concerning Petri nets, and then recall (see also [4–6]) notions related to net unfoldings.

[1] According to the notion of completeness presented in [5, 6], a marking M enabling a transition t may be represented by a deadlocked configuration C in a complete prefix, as long as there is another configuration C' representing this marking and enabling an instance of t. This means that the prefix may contain a deadlock, which does not correspond to any deadlock in the original net system (see Figure 2).

Petri nets A *net* is a triple $N \stackrel{\text{df}}{=} (P, T, F)$ 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, ${}^\bullet z \stackrel{\text{df}}{=} \{y \mid (y, z) \in F\}$ and $z^\bullet \stackrel{\text{df}}{=} \{y \mid (z, y) \in F\}$ denote the *pre-* and *postset* of $z \in P \cup T$, and we define ${}^\bullet Z \stackrel{\text{df}}{=} \bigcup_{z \in Z} {}^\bullet z$ and $Z^\bullet \stackrel{\text{df}}{=} \bigcup_{z \in Z} z^\bullet$, for all $Z \subseteq P \cup T$. We will assume that ${}^\bullet t \neq \emptyset \neq t^\bullet$, for every $t \in T$.

A *net system* is a pair $\Sigma \stackrel{\text{df}}{=} (N, M_0)$ comprising a finite net $N = (P, T, F)$ and a finite *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 $M' \stackrel{\text{df}}{=} M - {}^\bullet t + t^\bullet$, 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 smallest (w.r.t. set inclusion) 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 net system Σ is *k-bounded* if, for every reachable marking M and every place $p \in P$, $M(p) \leq k$, and *safe* if it is 1-bounded. Moreover, Σ is *bounded* if it is k-bounded for some $k \in \mathbb{N}$. The set $\mathcal{M}(\Sigma)$ is finite iff Σ is bounded.

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 $ON \stackrel{\text{df}}{=} (B, E, G)$, 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*. A *$\prec$-chain* of events is a finite or infinite sequence of events such that for each two consecutive events, e and f, it is the case that $e \prec f$. Two nodes are *concurrent*, denoted $y \; co \; y'$, if neither $y \# y'$ nor $y \preceq y'$ nor $y' \preceq y$.

A *homomorphism* from an occurrence net ON to a net system Σ is a mapping $h : B \cup E \to P \cup T$ such that: $h(B) \subseteq P$ and $h(E) \subseteq T$; for all $e \in E$, the restriction of h to ${}^\bullet e$ is a bijection between ${}^\bullet e$ and ${}^\bullet h(e)$; the restriction of h to $e^\bullet$ is a bijection between $e^\bullet$ and $h(e)^\bullet$; the restriction of h to $Min(ON)$ is a bijection between the multisets $Min(ON)$ and M_0; and for all $e, f \in E$, if ${}^\bullet e = {}^\bullet f$ and $h(e) = h(f)$ then $e = f$. If an event e is such that $h(e) = t$, then we will often refer to it as being *t-labelled*.

A *branching process* of Σ (see [4]) is a quadruple $\pi \stackrel{\text{df}}{=} (B, E, G, h)$ such that (B, E, G) is an occurrence net and h is a homomorphism from ON to Σ. A branching process $\pi' = (B', E', G', h')$ of Σ is a *prefix* of a branching process $\pi = (B, E, G, h)$, 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 h' is the restriction of h 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.

Sometimes it is convenient to start a branching process with a (virtual) initial event $\perp$, which has the postset $Min(ON)$, empty preset, and no label; we will henceforth use such an event, without drawing it in figures or treating it explicitly in algorithms.

An example of a safe net system and two of its branching prefixes is shown in Figure 1, where the homomorphism h is indicated by the labels of the nodes. The process in Figure 1(b) is a prefix of that in Figure 1(c).

Configurations and cuts A *configuration* of an occurrence net ON is a set of events C such that for all $e, f \in C$, $\neg(e \# f)$ and, for every $e \in C$, $f \prec e$ implies $f \in C$; since we assume the

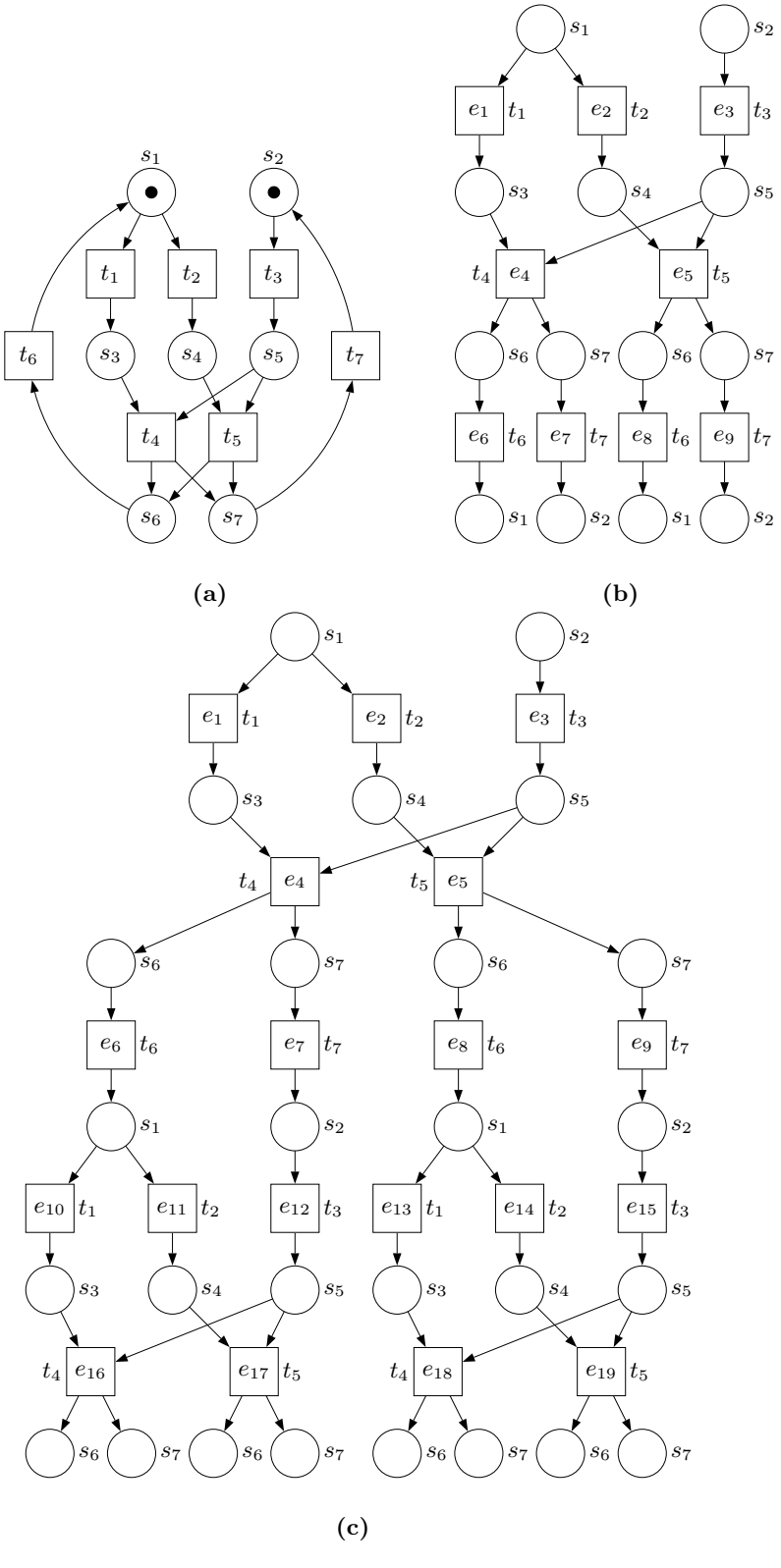

Fig. 1. A net system **(a)** and two of its branching processes **(b,c)**.

initial event $\bot$, we additionally require that $\bot \in C$. For every event $e \in E$, the configuration $[e] \stackrel{\text{df}}{=} \{f \mid f \preceq e\}$ is called the *local configuration* of e, and $\langle e \rangle \stackrel{\text{df}}{=} [e] \setminus \{e\}$ denotes the set of *causal predecessors* of e. Moreover, for a set of events E' we denote by $C \oplus E'$ the fact that $C \cup E'$ is a configuration and $C \cap E' = \emptyset$. Such an E' is a *suffix* of C, and $C \oplus E'$ is an *extension* of C.

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

A set of events E' is *downward-closed* if all causal predecessors of the events in E' also belong to E'. Such a set *induces* a unique branching process π whose events are exactly the events in E', and whose conditions are the conditions adjacent to the events in E' (including $\bot$).

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

Let C be a finite configuration of a branching process π. Then $Cut(C) \stackrel{\text{df}}{=} (Min(ON) \cup C^\bullet) \setminus {}^\bullet C$ is a cut; moreover, the multiset of places $h(Cut(C))$ is a reachable marking of Σ, denoted $Mark(C)$. A marking M of Σ is *represented* in π if there is $C \in \mathcal{C}^\pi_{fin}$ such that $M = Mark(C)$. Every marking represented in π is reachable in the original net system Σ, and every reachable marking of Σ is represented in the unfolding of Σ.

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

König's Lemma for branching processes König's Lemma (see [14]) states that a finitely branching, rooted, directed acyclic graph with infinitely many nodes reachable from the root has an infinite path.[2] We end this section proving what can be regarded as a version of such a result for branching processes of Petri nets.

Proposition 1. *A branching process π is infinite iff it contains an infinite $\prec$-chain of events.*

Proof. The 'if' part of the statement is trivial. To prove the 'only if' part, we observe that if π is infinite then, by the the fact that branching on each event (including $\bot$) is finite, π comprises infinitely many events.

For every event e of π, let $d(e)$ be the length of the longest $\prec$-chain of events from $\bot$ to e. It is easily seen that $d(e)$ is a well-defined integer, and that, for every k, there are finitely many events e of π such that $d(e) \leq k$ (comp. the proof of Proposition 4.8 in [6]).

Now, consider a graph $\mathcal{G}$ on the set of events of π, such that there is an arc from e to e' iff $e \prec e'$ and $d(e') = d(e) + 1$. Clearly, $\mathcal{G}$ is an infinite, rooted, directed acyclic graph with all its nodes being reachable from the root $\bot$. By the observation made in the previous paragraph, $\mathcal{G}$ is finitely branching, and so, by König's Lemma, there is an infinite path in $\mathcal{G}$. Clearly, such a path determines an infinite $\prec$-chain in π. □

It is worth noting that the above result does not follow directly from the original König's Lemma, since conditions of a branching process can have infinitely many outgoing arcs.

3 Complete prefixes of Petri net unfoldings

As explained in the introduction, there exist different methods of truncating Petri net unfoldings. The differences are related to the kind of information about the original unfolding one wants to preserve in the prefix, as well as to the choice between using either only local configurations (which can improve the running time of an algorithm), or all finite configurations (which can result in a smaller prefix). Also, we need a more general notion of completeness for branching processes. In this section, we generalise the entire set-up so that it will be applicable to different methods of truncating unfoldings and, at the same time, allow one to express the completeness w.r.t. properties other than marking reachability.

[2] The graph may be such that, for every $n \in \mathbb{N}$, there is a node with more than n branches, but no node can have an infinite number of branches.

Cutting contexts In order to cope with different variants of the technique for truncating unfoldings, we generalise the whole setup, using an abstract parametric model. The first parameter will determine the information we intend to preserve in a complete prefix (in the standard case, this is the set of reachable markings). The main idea behind it is to speak about finite configurations of Unf_Σ^{max} rather than reachable markings of Σ. Formally, the information to be preserved corresponds to the equivalence classes of some equivalence relation $\approx$ on $\mathcal{C}_{fin}$. The other parameters are more technical: they specify the circumstances under which an event can be designated as a cut-off event.

Definition 2. A *cutting context* is a triple

$$\Theta \stackrel{\text{df}}{=} \left(\approx, \lhd, \{\mathcal{C}_e\}_{e\in E}\right) ,$$

where:

1. $\approx$ is an equivalence relation on $\mathcal{C}_{fin}$.
2. $\lhd$, called an *adequate* order (comp. [6, Definition 4.5]), is a strict well-founded partial order on $\mathcal{C}_{fin}$ refining $\subset$, i.e., $C' \subset C''$ implies $C' \lhd C''$.
3. $\approx$ and $\lhd$ are *preserved by finite extensions*, i.e., for every pair of configurations $C' \approx C''$, and for every suffix E' of C', there exists[3] a finite suffix E'' of C'' such that
   (a) $C'' \oplus E'' \approx C' \oplus E'$, and
   (b) if $C'' \lhd C'$ then $C'' \oplus E'' \lhd C' \oplus E'$.
4. $\{\mathcal{C}_e\}_{e\in E}$ is a family of subsets of $\mathcal{C}_{fin}$. ◇

The main idea behind the adequate order is to specify which configurations will be preserved in the complete prefix; it turns out that all $\lhd$-minimal configurations in each equivalence class of $\approx$ will be preserved. The last parameter is needed to specify the set of configurations used later to decide whether an event can be designated as a cut-off event. For example, $\mathcal{C}_e$ may contain all finite configurations of Unf_Σ^{max}, or, as it is usually the case in practice, only the local ones. We will say that a cutting context Θ is *dense* (*saturated*) if $\mathcal{C}_e \supseteq \mathcal{C}_{loc}$ (resp. $\mathcal{C}_e = \mathcal{C}_{fin}$), for all $e \in E$.

In practice, Θ is usually dense (or even saturated, see [7]), and at least the following cases of the equivalence $\approx$ have been shown to be of interest:

- $C' \approx_{mar} C''$ if $Mark(C') = Mark(C'')$. This is the most widely used equivalence (see [5–7, 10, 15]). Note that the equivalence classes of $\approx_{mar}$ correspond to the reachable markings of Σ.
- $C' \approx_{code} C''$ if $Mark(C') = Mark(C'')$ and $Code(C') = Code(C'')$, where $Code(C)$ is the signal coding function. Such an equivalence is used in [18] for unfolding Signal Transition Graphs (STGs) specifying asynchronous circuits. Briefly, an STG is a Petri net together with a set of binary signals, which can be set or reset by transition firings. A transition can either change the value of one specific signal, or not affect any signal at all. As a result, the current values of the signals depend not on the current marking, but rather on the sequence of transition firings from the initial marking. And, in effect, one is interested in a 'combined' system state which includes both the current marking and the current values of the binary signals.
- $C' \approx_{sym} C''$ if $Mark(C')$ and $Mark(C'')$ are symmetric (equivalent) markings. This equivalence is the basis of the approach exploiting symmetries to reduce the size of the prefix, described in [3].

[3] Note that unlike [5, 6], we do not require that $E'' = I_1^2(E')$, where I_1^2 is the 'natural' isomorphism between the finite extensions of C' and C''. That isomorphism is defined only if $Mark(C') = Mark(C'')$, and thus cannot be used in our generalised settings.

For an equivalence relation $\approx$, we denote by $\mathfrak{R}_{\approx}^{fin} \stackrel{\text{df}}{=} \mathcal{C}_{fin}/_{\approx}$ the set of its equivalence classes, and by $\mathfrak{R}_{\approx}^{loc} \stackrel{\text{df}}{=} \mathcal{C}_{loc}/_{\approx}$ the set of its equivalence classes on the local configurations. We will also denote by Θ_{ERV} the cutting context corresponding to the framework used in [5, 6], i.e., such that $\approx$ is equal to $\approx_{mar}$, $\lhd$ is the total adequate order for safe net systems proposed there, and $\mathcal{C}_e = \mathcal{C}_{loc}$, for all $e \in E$.

We will write $e \lhd f$ whenever $[e] \lhd [f]$. Clearly, $\lhd$ is a well-founded partial order on the set of events. Hence, we can use Noetherian induction for definitions and proofs, i.e., it suffices to define or prove something for an event under the assumption that it has already been defined or proven for all its $\lhd$-predecessors.

Proposition 3. *Let e and f be two events, and C be a finite configuration.*

1. *If $f \prec e$ then $f \lhd e$.*
2. *If $f \in C \lhd [e]$ then $f \lhd e$.*

Proof. The first part follows from the fact that $f \prec e$ implies $[f] \subset [e]$, and so, by Definition 2(2), $[f] \lhd [e]$. To show the second part, we observe that if $[f] \subset C$ then, by Definition 2(2), $[f] \lhd C$. Hence $[f] \lhd [e]$, by the transitivity of $\lhd$. □

In the rest of this paper, we assume that the cutting context Θ is fixed.

Completeness of branching processes We now introduce a new notion of completeness for branching processes.

Definition 4. A branching process π is *complete w.r.t. a set E_{cut}* of events of Unf_{Σ}^{max} if the following hold:

1. If $C \in \mathcal{C}_{fin}$, then there is $C' \in \mathcal{C}_{fin}^{\pi}$ such that $C' \cap E_{cut} = \emptyset$ and $C \approx C'$.
2. If $C \in \mathcal{C}_{fin}^{\pi}$ is such that $C \cap E_{cut} = \emptyset$, and e is an event such that $C \oplus \{e\} \in \mathcal{C}_{fin}$, then $C \oplus \{e\} \in \mathcal{C}_{fin}^{\pi}$.

A branching process π is *complete* if it is complete w.r.t. some set E_{cut}. ◇

Note that, in general, π remains complete after removing all events e for which $\langle e \rangle \cap E_{cut} \neq \emptyset$; i.e., without affecting the completeness, one can truncate a complete prefix so that the events from E_{cut} (usually referred to as *cut-off* events) will be either maximal events of the prefix or not in the prefix at all. Note also that the last definition depends only on the equivalence $\approx$, and not on the other components of the cutting context.

For the relation $\approx_{mar}$, each reachable marking is represented by a configuration in $\mathcal{C}_{fin}$ and, hence, also by a configuration in $\mathcal{C}_{fin}^{\pi}$, provided that π is complete (see Definition 4). This is what is usually expected from a correct prefix. But even in this special case, our notion of completeness differs from that presented in [5, 6], since it requires *all* configurations in $\mathcal{C}_{fin}^{\pi}$ containing no events from E_{cut} to preserve all transition firings, rather than the *existence* of a configuration preserving all firings (see Figure 2). The justification why such a stronger property is desirable was given in the introduction. One can easily prove that our notion is strictly stronger than the one considered in [5, 6], i.e., that the completeness in the sense of Definition 4 implies the completeness in the sense of [5, 6], but not vice versa. However, it should be noted that the proof of completeness in [5, 6] almost gives the stronger notion; we have adopted it (see Proposition 8) with relatively few modifications.

As an example, consider the net system in Figure 1. If $\approx$ is equal to $\approx_{mar}$, then the prefix in Figure 1(b) is not complete w.r.t. any set E_{cut} for the following reason. The reachable marking $\{s_1, s_7\}$ is only represented by the local configurations of e_6 and e_8; for completeness, at least one of these events would not be in E_{cut}, but then its local configuration would not have an extension with a t_1-labelled event. In contrast, the prefix in Figure 1(c) is complete w.r.t. the set $E_{cut} = \{e_5, e_{16}, e_{17}\}$.[4] Notice that the events e_8, e_9, e_{13}–e_{15}, e_{18}, and e_{19} can be removed from the prefix without affecting its completeness.

[4] This choice of E_{cut} is not unique: one could have chosen, e.g., $E_{cut} = \{e_4, e_{18}, e_{19}\}$.

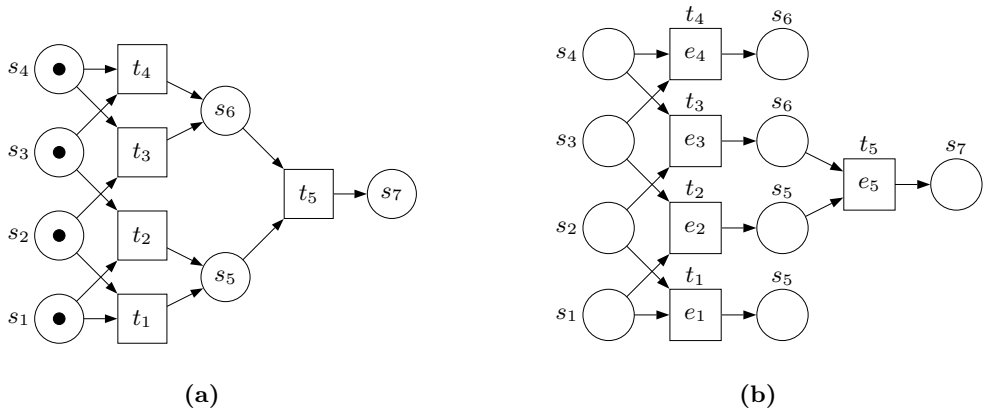

Fig. 2. A Petri net **(a)** and one of its branching processes **(b)**, which is complete w.r.t. the definition used in [5, 6], but not w.r.t. Definition 4. Note that the configuration $\{e_1, e_4\}$ does not preserve firings and introduces a fake deadlock. In order to make this prefix complete w.r.t. Definition 4 one has to add another instance of t_5, consuming the conditions produced by e_1 and e_4.

4 Canonical prefix

In this section, we develop the central results of this paper. First, we show that cut-off events can be defined without resorting to any algorithmic argument. This yields a definition of the canonical prefix, and we then prove several of its relevant properties.

Static cut-off events In [5, 6], the notion of a cut-off event was considered as algorithm-specific, and was given w.r.t. the already built part of a prefix. Now we define cut-off events w.r.t. the whole unfolding instead, so that it will be independent of an algorithm (hence the term 'static'), together with *feasible* events, which are precisely those events whose causal predecessors are not cut-off events, and as such must be included in the prefix determined by the static cut-off events.

Definition 5. The set of *feasible* events, denoted by fsble_Θ, and the set of *static cut-off* events, denoted by cut_Θ, are two sets of events of $\mathit{Unf}_\Sigma^{max}$ defined inductively, in the following way:

1. An event e is a feasible event if $\langle e \rangle \cap \mathit{cut}_\Theta = \emptyset$.
2. An event e is a static cut-off event if it is feasible, and there is a configuration $C \in \mathcal{C}_e$ such that $C \subseteq \mathit{fsble}_\Theta \setminus \mathit{cut}_\Theta$, $C \approx [e]$, and $C \lhd [e]$. In what follows, any C satisfying these conditions will be called a *corresponding* configuration of e. ◇

Note that fsble_Θ and cut_Θ are well-defined sets. Indeed, when considering an event e, by the well-foundedness of $\lhd$ and Proposition 3(1), one can assume that for the events in $\langle e \rangle$ it has already been decided whether they are in fsble_Θ or in cut_Θ. And, by Proposition 3(2), the same holds for the events in any configuration C satisfying $C \lhd [e]$.

The above definition implies that $\bot \in \mathit{fsble}_\Theta$ since $\langle \bot \rangle = \emptyset$. Furthermore, $\bot \notin \mathit{cut}_\Theta$, since $\bot$ cannot have a corresponding configuration. Indeed, $[\bot] = \{\bot\}$ is the smallest (w.r.t. set inclusion) configuration, and so, by Definition 2(2), it is also $\lhd$-minimal.

Remark 1. A naïve attempt to define an algorithm-independent notion of a cut-off event as an event e for which there is a configuration $C \in \mathcal{C}_e$ such that $C \approx [e]$ and $C \lhd [e]$ fails. Indeed, suppose that $\Theta = \Theta_{ERV}$, as it is often the case in practice. It may happen that a corresponding local configuration C of a cut-off event e defined in this way contains another cut-off event.

Though in this case Unf_Σ^{max} contains another corresponding configuration $C' \lhd C$ with no cut-off events and the same final marking, such a configuration is not necessarily local.

The approach proposed in this paper, though slightly more complicated, allows to deal uniformly with arbitrary cutting contexts. Moreover, it coincides with the described naïve approach when Θ is saturated. ◇

Proposition 6. *Let e be an event of Unf_Σ^{max}.*

1. *$e \in fsble_\Theta$ iff $\langle e\rangle \subseteq fsble_\Theta \setminus cut_\Theta$.*
2. *$e \in cut_\Theta$ implies $e \in fsble_\Theta$.*

Proof. The 'if' part of the first clause follows directly from Definition 5(1). To prove the 'only if' part, suppose that $g \notin fsble_\Theta \setminus cut_\Theta$ for some $g \prec e$. Since $e \in fsble_\Theta$ and thus $\langle e\rangle \cap cut_\Theta = \emptyset$ by Definition 5(1), $g \notin fsble_\Theta$. Therefore, by Definition 5(1), $\langle g\rangle \cap cut_\Theta \neq \emptyset$, and so $\langle e\rangle \cap cut_\Theta \neq \emptyset$, a contradiction. Hence $\langle e\rangle \subseteq fsble_\Theta \setminus cut_\Theta$.

The second clause follows directly from Definition 5(2). □

Canonical prefix and its properties Once we have defined the feasible events, the notion of the canonical prefix arises quite naturally, after observing that the 'only if' part of Proposition 6(1) implies that Unf_Σ^Θ is a downward-closed set of events.

Definition 7. The branching process Unf_Σ^Θ induced by the set of events $fsble_\Theta$ is called the *canonical* prefix of Unf_Σ^{max}. ◇

Note that Unf_Σ^Θ is uniquely determined by the cutting context Θ.

In what follows, we prove several fundamental properties of Unf_Σ^Θ. Note that, unlike those given in [5, 6], our proofs are not algorithm-specific.

Proposition 8 (completeness). *Unf_Σ^Θ is complete w.r.t. $E_{cut} = cut_\Theta$.*

Proof. (Comp. the proof of Proposition 4.9 in [6]).

Let $\pi \stackrel{\text{df}}{=} Unf_\Sigma^\Theta$. To show Definition 4(1), suppose that $C \in \mathcal{C}_{fin}$. Let $C' \in \mathcal{C}_{fin}$ be $\lhd$-minimal among the configurations satisfying $C' \approx C$. Note that C' exists since, by Definition 2(2), $\lhd$ is well-founded. Suppose that $C' \cap cut_\Theta \neq \emptyset$, i.e., there is an event $e \in C' \cap cut_\Theta$. Then $C' = [e] \oplus E'$, for some finite suffix E' of $[e]$. Let C_e be a corresponding configuration of e. Since $C_e \approx [e]$ and $C_e \lhd [e]$, by Definition 2(3), there exists a suffix E'' of C_e such that $C_e \oplus E'' \approx [e] \oplus E'$ and $C_e \oplus E'' \lhd [e] \oplus E'$, i.e., $C_e \oplus E'' \approx C' \approx C$ and $C_e \oplus E'' \lhd C'$. Since $\lhd$ is strict, this contradicts the choice of C', and so $C' \cap cut_\Theta = \emptyset$. Therefore, by Definition 5(1), each event of C' is feasible, and so C' is a configuration of Unf_Σ^Θ containing no events from E_{cut}.

To show Definition 4(it-compl-firings), suppose that $C \in \mathcal{C}_{fin}^\pi$, $C \cap E_{cut} = \emptyset$, and $C \oplus \{f\} \in \mathcal{C}_{fin}$. Then, by Proposition 6(1), $f \in fsble_\Theta$, and so $C \oplus \{f\} \in \mathcal{C}_{fin}^\pi$. □

Having proved that the canonical prefix is always complete, we now set out to analyse its finiteness. This property is, clearly, crucial if one intends to use such a prefix for model checking.

Proposition 9 (finiteness I). *Unf_Σ^Θ is finite iff there is no infinite $\prec$-chain of feasible events.*

Proof. Follows directly from Definition 7 and Proposition 1. □

Thus, in order to guarantee that the canonical prefix is finite, one should choose the cutting context so that the $\mathcal{C}_e$'s contain enough configurations, and $\approx$ is coarse enough, to cut each infinite $\prec$-chain. It is interesting to observe that certain cutting contexts sometimes allow one to produce finite canonical prefixes even for unbounded net systems. Figure 3(a) shows a net modelling a loop, where place p_2, used for counting the number of iterations, is unbounded.

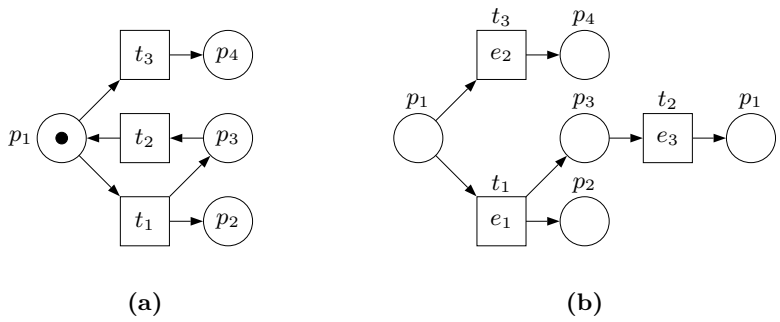

Fig. 3. An unbounded net system **(a)** and its canonical prefix **(b)**. The cutting context is such that $C' \approx C'' \Leftrightarrow Mark(C') \cap \{p_1, p_3, p_4\} = Mark(C'') \cap \{p_1, p_3, p_4\}$ and $\{\bot\} \in \mathcal{C}_{e_3}$, and so e_3 is a static cut-off event.

If $\approx$ ignores the value of this counter, it is possible to build a finite and complete canonical prefix, shown in Figure 3(b).

A necessary condition for the finiteness of the canonical prefix is the finiteness of the set of the equivalence classes of $\approx$. Moreover, this becomes also a sufficient condition if Θ is dense. The following result provides quite a tight and practical indication for deciding whether Unf_Σ^Θ is finite or not.

Proposition 10 (finiteness II).

1. *If $\approx$ has finitely many equivalence classes and Θ is dense, then Unf_Σ^Θ is finite.*
2. *If $\approx$ has infinitely many equivalence classes, then Unf_Σ^Θ is infinite.*

Proof. (1) (Comp. the proof of Proposition 4.8 in [6].)

By Proposition 9, it is enough to show that there is no infinite $\prec$-chain of feasible events. To the contrary, suppose that such a chain $e_1 \prec e_2 \prec \cdots$ does exist. Since the number of equivalence classes of $\approx$ is finite, there exist $i, j \in \mathbb{N}$ such that $i < j$ and $[e_i] \approx [e_j]$. Since $e_i \prec e_j$, we have $[e_i] \subset [e_j]$, and so $e_i \lhd e_j$ by Definition 2(2). Since Θ is dense, $[e_i] \in \mathcal{C}_{e_j}$, and since e_j is feasible, no event in $[e_i]$ belongs to cut_Θ. Thus, $e_j \in cut_\Theta$ and has no feasible causal successors, a contradiction.

(2) By Proposition 8, Unf_Σ^Θ is complete. Therefore, it contains at least one configuration from every equivalence class of $\approx$. Since a finite branching process contains only a finite number of configurations, Unf_Σ^Θ is infinite. □

Corollary 11 (finiteness III). *Let $\approx$ be either of $\approx_{mar}$, $\approx_{code}$, $\approx_{sym}$.*

1. *If Σ is bounded and Θ is dense, then Unf_Σ^Θ is finite.*
2. *If Σ is unbounded, then Unf_Σ^Θ is infinite.*

Proof. Follows from Proposition 10 and the fact that each of the three equivalences has a finite number of equivalence classes iff Σ is bounded. Indeed, since there are only finitely many signals, the set $Code(\mathcal{C}_{fin})$ is finite, and so the set of combined states $\{(Mark(C), Code(C)) \mid C \in \mathcal{C}_{fin}\}$, which is isomorphic to $\mathfrak{R}^{fin}_{\approx_{code}}$, is finite iff the set $\mathcal{M}(\Sigma)$ of reachable markings is finite. For $\approx_{sym}$, $|\mathfrak{R}^{fin}_{\approx_{sym}}| \leq |\mathfrak{R}^{fin}_{\approx_{mar}}|$, and so $\mathfrak{R}^{fin}_{\approx_{sym}}$ is finite if so is $\mathfrak{R}^{fin}_{\approx_{mar}}$. Moreover, since $C \approx_{sym} C'$ implies $|Mark(C)| = |Mark(C')|$, $\mathfrak{R}^{fin}_{\approx_{sym}}$ is infinite if $\mathfrak{R}^{fin}_{\approx_{sym}}$ is infinite. □

In the important special case of a total adequate order, one can also derive an upper bound on the number of non-cut-off events in Unf_Σ^Θ. A specialised version of the next result (for $\Theta = \Theta_{ERV}$) was proven in [5, 6] for the prefix generated by the unfolding algorithm presented there.

Proposition 12 (upper bound). *Suppose that $\lhd$ is total and:*

- *The set $\mathfrak{R}^{loc}_{\approx}$ is finite.*
- *For every $\mathcal{R} \in \mathfrak{R}^{loc}_{\approx}$, there is an integer $\gamma_{\mathcal{R}} > 0$ such that, for every chain $e_1 \lhd e_2 \lhd \cdots \lhd e_{\gamma_{\mathcal{R}}}$ of feasible events whose local configurations belong to $\mathcal{R}$, there is at least one $i \leq \gamma_{\mathcal{R}}$ such that $[e_i] \in \bigcap_{[e] \in \mathcal{R}} \mathcal{C}_e$.*

Then

$$\left| \mathit{fsble}_\Theta \setminus \mathit{cut}_\Theta \right| \leq \sum_{\mathcal{R} \in \mathfrak{R}^{loc}_{\approx}} \gamma_{\mathcal{R}} \ . \tag{1}$$

Proof. Suppose that $e_1, \ldots, e_{\gamma_{\mathcal{R}}+1} \in \mathit{fsble}_\Theta \setminus \mathit{cut}_\Theta$ are distinct events whose local configurations belong to the same equivalence class $\mathcal{R}$ of $\approx$. Since $\lhd$ is total, we may assume, without loss of generality, that $e_1 \lhd \ldots \lhd e_{\gamma_{\mathcal{R}}+1}$. Hence, for at least one $i \leq \gamma_{\mathcal{R}}$, $[e_i] \in \bigcap_{[e] \in \mathcal{R}} \mathcal{C}_e \subseteq \mathcal{C}_{e_{\gamma_{\mathcal{R}}+1}}$, i.e., $e_{\gamma_{\mathcal{R}}+1} \in \mathit{cut}_\Theta$, a contradiction. Thus (1) holds. □

Note that if Θ is dense, then $\gamma_{\mathcal{R}} = 1$ for every $\mathcal{R} \in \mathfrak{R}^{loc}_{\approx}$, and

$$\sum_{\mathcal{R} \in \mathfrak{R}^{loc}_{\approx}} \gamma_{\mathcal{R}} = |\mathfrak{R}^{loc}_{\approx}| \leq |\mathfrak{R}^{fin}_{\approx}| \ .$$

The standard result of [6] is then obtained by taking $\Theta = \Theta_{ERV}$. Indeed, since the reachable markings of Σ correspond to the equivalence classes of $\approx_{mar}$, the upper bound on the number of non-cut-off events in $\mathit{Unf}^{\Theta}_{\Sigma}$ in this case is equal to $|\mathcal{M}(\Sigma)|$. Using the above proposition, one can easily derive the following upper bounds for the remaining two equivalences considered in this paper (in each case, we assume that Θ is dense):

- $\left|\mathfrak{R}^{fin}_{\approx_{code}}\right| = \left|\{(\mathit{Mark}(C), \mathit{Code}(C))\}_{C \in \mathcal{C}_{fin}}\right| \leq \left|\mathcal{M}(\Sigma)\right| \cdot \left|\mathit{Code}(\mathcal{C}_{fin})\right| \leq \left|\mathcal{M}(\Sigma)\right| \cdot 2^n$, where n is the number of signals.
- $|\mathfrak{R}^{fin}_{\approx_{sym}}| \leq |\mathfrak{R}^{fin}_{\approx_{mar}}| = |\mathcal{M}(\Sigma)|$.

Note that these upper bounds are rather pessimistic, particulary because we bound $|\mathfrak{R}^{loc}_{\approx}|$ by $|\mathfrak{R}^{fin}_{\approx}|$. In practice, the set $\mathfrak{R}^{fin}_{\approx}$ is usually exponentially larger than $\mathfrak{R}^{loc}_{\approx}$, and so prefixes are often exponentially smaller than reachability graphs.

5 Unfolding algorithms

We now show that the unfolding algorithm presented in [5, 6] and the parallel unfolding algorithm proposed in [10] generate the canonical prefix defined in the previous section.

ERV unfolding algorithm The unfolding algorithm presented in [5, 6] can be expressed as shown in Figure 4. We will call it the *basic* algorithm. In the present paper, it is parameterized by a cutting context Θ. It is assumed that the function PotExt(Pref_Σ) finds the set of *possible extensions* of a branching process Pref_Σ, according to the following definition.

Definition 13. Let π be a branching process of Σ. A *possible extension* of π is a pair (t, D), where D is a co-set in π and t is a transition of Σ, such that $h(D) = {}^\bullet t$ and π contains no t-labelled event with preset D.[5]
The pair (t, D) is an event used to extend π, and we will take it as being t-labelled and having D as its preset. ◇

When $\lhd$ is a total order, the algorithm in Figure 4 is deterministic, and thus always yields the same result for a given net system. A rather surprising fact is that this is also the case for an arbitrary adequate order for which the algorithm is, in general, non-deterministic. This fact was proven in [10] in a rather complicated way, by comparing two runs of the algorithm. Below, we provide a more elegant proof, by showing that the algorithm always generates the canonical prefix.

[5] Note that the π constructed at some stage can only contain a t-labelled event with preset D, if it contains (t, D) already.

input : $\Sigma = (N, M_0)$ — a net system
output : $Pref_\Sigma$ — the canonical prefix of Σ's unfolding (if it is finite)

$Pref_\Sigma \leftarrow$ the empty branching process
add instances of the places from M_0 to $Pref_\Sigma$
$pe \leftarrow \textsc{PotExt}(Pref_\Sigma)$
$cut\_off \leftarrow \emptyset$
while $pe \neq \emptyset$ **do**
  choose $e \in \min_\lhd pe$
  if $[e] \cap cut\_off = \emptyset$
  then
    add e and new instances of the places from $h(e)^\bullet$ to $Pref_\Sigma$
    $pe \leftarrow \textsc{PotExt}(Pref_\Sigma)$
    if e is a cut-off event of $Pref_\Sigma$ **then** $cut\_off \leftarrow cut\_off \cup \{e\}$
  else $pe \leftarrow pe \setminus \{e\}$

Note: e is a cut-off event of $Pref_\Sigma$ if there is $C \in \mathcal{C}_e$ such that
the events of C belong to $Pref_\Sigma$ but not to cut_off, $C \approx [e]$, and $C \lhd [e]$.

Fig. 4. The unfolding algorithm presented in [5].

Unfolding with slices An unfolding algorithm which admits efficient parallelization (proposed in [10]) is shown in Figure 5. We will call it the *slicing* algorithm. When compared to the basic algorithm, it has the following modifications in its main loop. A set of events $Sl \in \textsc{Slices}(pe)$, called a *slice* of the current set of possible extensions, is chosen on each iteration and processed as a whole, without taking or adding any other events from or to pe. A slice must satisfy the following:

- Sl is a non-empty subset of pe; and
- for every event $e \in Sl$ and every event $f \lhd e$ of Unf_Σ^{max}, $f \notin pe \setminus Sl$ and $pe \cap \langle f \rangle = \emptyset$. (*)

In particular, if $f \in pe$ and $f \lhd e$ for some $e \in Sl$, then $f \in Sl$. The set $\textsc{Slices}(pe)$ is chosen so that it is non-empty whenever pe is non-empty. Note that this algorithm, in general, exhibits more non-determinism than the basic one (it may be non-deterministic even if the order $\lhd$ is total).

It was proven (in a very complicated way) in [10] that the unfolding algorithms shown in Figures 4 and 5 are equivalent, in the sense that prefixes produced by arbitrary runs of these algorithms are isomorphic. In this paper, we prove this result by showing that arbitrary runs of these algorithms generate the canonical prefix.

Since the basic algorithm can be obtained as a special case of the slicing one, by setting $\textsc{Slices}(pe) \stackrel{\text{df}}{=} \{\{e\} \mid e \in \min_\lhd pe\}$ (comp. [10]), the proofs below are given only for the slicing algorithm.

Lemma 14. *Consider the state of the algorithm in Figure 5 before adding an event e to $Pref_\Sigma$. If $cut\_off \subseteq cut_\Theta$, $g \in fsble_\Theta$ and $g \lhd e$, then g is in $Pref_\Sigma$.*

Proof. Let $Pref'_\Sigma$ be the state of the variable $Pref_\Sigma$ at the moment when a slice Sl was chosen in the current iteration of the main loop of the algorithm.

Suppose that $g \in fsble_\Theta$ is such that $g \lhd e$ and g is not in $Pref_\Sigma$. Consider the set $G \stackrel{\text{df}}{=} \{h \in [g] \mid h \text{ is not in } Pref'_\Sigma\}$. Clearly, $G \neq \emptyset$ since $g \in G$ (as g is not in $Pref_\Sigma$ and the algorithm never removes events from the prefix being constructed). Thus there is $f \in \min_\prec G$. By Proposition 3(2), $f \lhd e$. Moreover, $f \in pe$ because all its causal predecessors are in $Pref'_\Sigma$ by the choice of f.

input : $\Sigma = (N, M_0)$ — a net system
output : $Pref_\Sigma$ — the canonical prefix of Σ's unfolding (if it is finite)

$Pref_\Sigma \leftarrow$ the empty branching process
add instances of the places from M_0 to $Pref_\Sigma$
$pe \leftarrow$ PotExt($Pref_\Sigma$)
$cut\_off \leftarrow \emptyset$
while $pe \neq \emptyset$ **do**
  choose $Sl \in$ Slices(pe)
  if $\exists e \in Sl : \ [e] \cap cut\_off = \emptyset$
  then
    for all $e \in Sl$ in any order refining $\lhd$ **do**
      if $[e] \cap cut\_off = \emptyset$
      then
        add e and new instances of the places from $h(e)^\bullet$ to $Pref_\Sigma$
        if e is a cut-off event of $Pref_\Sigma$ **then** $cut\_off \leftarrow cut\_off \cup \{e\}$
    $pe \leftarrow$ PotExt($Pref_\Sigma$)
  else $pe \leftarrow pe \setminus Sl$

Note: e is a cut-off event of $Pref_\Sigma$ if there is $C \in \mathcal{C}_e$ such that
the events of C belong to $Pref_\Sigma$ but not to cut_off, $C \approx [e]$, and $C \lhd [e]$.

Fig. 5. Unfolding algorithm with slices.

By the first part of (*), $e \in Sl$, $f \lhd e$, and $f \in pe$, we have that $f \in Sl$. Moreover, if $f \neq g$ then $f \prec g$, contradicting the second part of (*), since $e \in Sl$, $g \lhd e$, and $pe \cap \langle g \rangle \neq \emptyset$ (because $f \in pe \cap \langle g \rangle$). Therefore, $g = f \in Sl$. Since $g \lhd e$, it was processed before e in the **for all** loop of the algorithm. By $g \in fsble_\Theta$, $cut\_off \subseteq cut_\Theta$, and Proposition 6(1), we obtain that $\langle g \rangle \cap cut\_off \subseteq \langle g \rangle \cap cut_\Theta = \emptyset$. Moreover, $g \notin cut\_off$ when g is being processed. Therefore, g has been added to $Pref_\Sigma$ before the processing of e, a contradiction. □

Lemma 15 (soundness). *If the algorithm in Figure 5 adds an event e to $Pref_\Sigma$, then $e \in fsble_\Theta$. Moreover, such an event e is added to cut_off iff $e \in cut_\Theta$.*

Proof. By induction on the number of events added before e. When e is being added to $Pref_\Sigma$, the condition $[e] \cap cut\_off = \emptyset$ is satisfied. Since the events in $\langle e \rangle$ have been added before, $\langle e \rangle \cap cut_\Theta = \emptyset$ by induction, and so $e \in fsble_\Theta$.

If e is then added to cut_off due to some corresponding configuration $C \lhd [e]$, then the events of C belong to $Pref_\Sigma$ but not to cut_Θ. Hence, by induction, $C \subseteq fsble_\Theta \setminus cut_\Theta$. Thus, $e \in cut_\Theta$.

Now assume an event e added to $Pref_\Sigma$ is in cut_Θ with a corresponding configuration C. Since, by Proposition 3(2), $g \in C \lhd [e]$ implies $g \lhd e$, and by the fact that $cut\_off \subseteq cut_\Theta$ before e was added to $Pref_\Sigma$ (by induction), each $g \in C$ has already been added to $Pref_\Sigma$ by Lemma 14. Furthermore, $g \notin cut\_off$, by $C \cap cut_\Theta = \emptyset$ and the induction hypothesis. Therefore, the algorithm will add e to cut_off. □

Lemma 16 (termination). *If Unf_Σ^Θ is finite, then the algorithm in Figure 5 terminates in a finite number of steps.*

Proof. The only time when events are added to pe is the call to PotExt. Such a call returns only a finite set of possible extensions, and so pe is always finite. In the body of the **while** loop, non-empty slices are removed from pe. This is repeated until a new event is added to $Pref_\Sigma$, which, by the assumption and Lemma 15, can happen only finitely many times, or until pe becomes empty and the algorithm terminates. □

Lemma 17 (completeness). *Let $e \in fsble_\Theta$. If the algorithm in Figure 5 terminates yielding a prefix $Pref_\Sigma$, then e is an event of $Pref_\Sigma$.*

Proof. Suppose that e is a $\lhd$-minimal event of $fsble_\Theta$ which is not in $Pref_\Sigma$ at termination. All causal predecessors of e are in $Pref_\Sigma$, but, by Lemma 15, not in cut_off. Thus, e was in pe after possible extensions where computed for the last time; the condition $[e] \cap cut\_off = \emptyset$ holds at termination, and thus has been holding before. Therefore, since pe is empty at termination, e was added to $Pref_\Sigma$, a contradiction. □

Theorem 18 (correctness). *If Unf_Σ^Θ is finite, then the algorithms in Figures 4 and 5 generate Unf_Σ^Θ in a finite number of steps.*

Proof. Follows directly from Lemmata 15, 16, and 17, and the fact that the basic algorithm is a special case of the slicing one. □

In particular, this result implies the completeness of prefixes produced by the basic and slicing algorithms w.r.t. our stronger notion of completeness (when compared to that used in [5, 6]), and the fact that arbitrary runs of these non-deterministic algorithms always yield the same result.

6 Conclusions

In this paper, we presented a general framework for truncating Petri net unfoldings. It provides a powerful tool for dealing with different variants of the unfolding technique, in a flexible and uniform way. In particular, by finely tuning the cutting contexts, one can build prefixes which better suit a particular model checking problem. A fundamental result is that, for an arbitrary Petri net and a cutting context, there exists a 'special' canonical prefix of its unfolding, which can be defined without resorting to any algorithmic argument.

We introduced a new, stronger notion of completeness of a branching process, which was implicitly assumed by many existing model checking algorithms employing unfoldings (see the introduction). We have shown that the canonical prefix is complete w.r.t. this notion, and that it is exactly the prefix generated by arbitrary runs of the non-deterministic unfolding algorithms presented in [5, 6, 10]. This gives a new correctness proof for the unfolding algorithms presented there, which is much simpler in the case of the slicing algorithm developed in [10]. As a result, relevant model checking tools can now make stronger assumptions about the properties of the prefixes they use. In particular, they can safely assume that for each configuration containing no cut-off events, all firings are preserved.

Finally, we proposed conditions for the finiteness of the canonical prefix, and presented criteria allowing placing bounds on its size, which should help in choosing problem-specific cutting contexts. It is worth noting that to deal with the finiteness problem we proved a version of König's Lemma for branching processes of (possibly unbounded) Petri nets.

We believe that the results contained in this paper, on the one hand, will help to better understand the issues relating to prefixes of Petri net unfoldings, and, on the other hand, will facilitate the design of efficient model checking tools.

Acknowledgements

This research was supported by an Ors Awards Scheme grant ORS/C20/4, and by Epsrc grants GR/M99293 and GR/M94366 (Movie).

References

1. E. M. Clarke, E. A. Emerson and A. P. Sistla: Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications. *ACM TOPLAS* 8 (1986) 244–263.

2. E. M. Clarke, O. Grumberg, and D. Peled: *Model Checking*. MIT Press (1999).
3. J. -M. Couvreur, S. Grivet, and Denis Poitrenaud: Unfolding of Products of Symmetrical Petri Nets. Proc. of *ICATPN'2001*, J. -M. Colom and M. Koutny (Eds.). Springer-Verlag, Lecture Notes in Computer Science 2075 (2001) 121–143.
4. J. Engelfriet: Branching processes of Petri Nets. *Acta Informatica* 28 (1991) 575–591.
5. J. Esparza, S. Römer and W. Vogler: An Improvement of McMillan's Unfolding Algorithm. Proc. of *TACAS'96*, Margaria T., Steffen B. (Eds.). Springer-Verlag, Lecture Notes in Computer Science 1055 (1996) 87–106.
6. J. Esparza, S. Römer and W. Vogler: An Improvement of McMillan's Unfolding Algorithm. *Formal Methods in System Design* (2001) to appear.
7. K. Heljanko: Minimizing Finite Complete Prefixes. Proc. of *CS&P'99*, Workshop Concurrency, Specification and Programming (1999) 83–95.
8. K. Heljanko: Deadlock and Reachability Checking with Finite Complete Prefixes. Technical Report A56, Laboratory for Theoretical Computer Science, HUT, Espoo, Finland (1999).
9. K. Heljanko: Using Logic Programs with Stable Model Semantics to Solve Deadlock and Reachability Problems for 1-Safe Petri Nets. *Fundamentae Informaticae* 37 (1999) 247–268.
10. K. Heljanko, V. Khomenko and M. Koutny: Parallelisation of the Petri Net Unfolding Algorithm. Technical Report CS-TR-733, Department of Computing Science, University of Newcastle (2001).
11. V. Khomenko and M. Koutny: Verification of Bounded Petri Nets Using Integer Programming. Technical Report CS-TR-711, Department of Computing Science, University of Newcastle (2000).
12. V. Khomenko and M. Koutny: LP Deadlock Checking Using Partial Order Dependencies. Proc. of *CONCUR'2000*, Palamidessi C. (Ed.). Springer-Verlag, Lecture Notes in Computer Science 1877 (2000) 410–425.
13. V. Khomenko, M. Koutny and A. Yakovlev: Detecting State Coding Conflicts in STGs Using Integer Programming. Technical Report CS-TR-736, Department of Computing Science, University of Newcastle (2001).
14. D. König: Über eine Schlußweise aus dem Endlichen ins Unendliche. *Acta Litt. ac. sci. Szeged* 3 (1927) 121–130. Bibliography in: Theorie der endlichen und unendlichen Graphen. Teubner, Leipzig (1936, reprinted 1986)
15. K. L. McMillan: Using Unfoldings to Avoid State Explosion Problem in the Verification of Asynchronous Circuits. Proc. of *4th CAV*, G. von Bochmann and D. K. Probst (Eds.). Springer-Verlag, Lecture Notes in Computer Science 663 (1992) 164–174.
16. K. L. McMillan: *Symbolic Model Checking*. PhD thesis, CMU-CS-92-131 (1992).
17. S. Melzer and S. Römer: Deadlock Checking Using Net Unfoldings. Proc. of *Computer Aided Verification (CAV'97)*, O. Grumberg (Ed.). Springer-Verlag, Lecture Notes in Computer Science 1254 (1997) 352–363.
18. A. Semenov: *Verification and Synthesis of Asynchronous Control Circuits Using Petri Net Unfolding*. PhD Thesis, University of Newcastle upon Tyne (1997).