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, parameterised 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. A key aspect of our approach is an *algorithm-independent* notion of cut-off events, used to truncate a Petri net unfolding. Such a notion is based on a *cutting context* and results in the unique *canonical* prefix of the unfolding. Canonical prefixes are complete in the new, stronger sense, and we provide necessary and sufficient conditions for its finiteness, as well as upper bounds on its size in certain cases. A surprising result is that after suitable generalisation, the standard unfolding algorithm presented in [5], and the parallel unfolding algorithm proposed in [8], 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

Computer aided verification tools implementing model checking (see, e.g., [1]) verify a concurrent system using a finite representation of its state space, and thus may suffer from the state explosion problem. To cope with this, several techniques have been developed, which usually aim either at a compact representation of the full state space of the system, or at the generation of its reduced (though sufficient for a given verification task) state space. Among them, a prominent technique is McMillan's (finite prefixes of) Petri Net unfoldings (see, e.g., [5,13]). They rely on the partial order view of concurrent computation, and represent system states implicitly, using an acyclic net. More precisely, given a Petri net Σ, the unfolding technique aims at building a labelled acyclic net Unf_Σ (a *prefix*) satisfying two key properties:

– *Completeness.* Each reachable marking of Σ is represented by at least one 'witness', i.e., one marking of Unf_Σ reachable from its initial marking. (Similarly, for each possible firing of a transition in Σ there is a suitable 'witness' event in Unf_Σ.)

D. Brinksma and K. G. Larsen (Eds.): CAV 2002, LNCS 2404, pp. 582–595, 2002.

– *Finiteness.* The prefix is finite and thus can be used as input to model checking algorithms, e.g., those searching for deadlocks.

This paper presents a uniform treatment of both these aspects and provides a fresh impetus for further development of unfolding-based model checking techniques.

There are two fundamental issues which we wish to address here, namely the precise *semantical* meaning of completeness, and the *algorithmic* problem of generating complete prefixes.

Semantical Meaning of Completeness A direct motivation to re-examine the issue of completeness was provided by our own experience of dealing with unfoldings of Signal Transition Graphs (STGs) in [11], used to specify the behaviour of asynchronous circuits. Briefly, an STG (see [16]) is a Petri net together with a set of binary signals (variables), which can be set or reset by transition firings. A transition can either change the value of one specific signal, or affect no signals at all. Thus, the current values of the signals depend not on the current marking, but rather on the sequence of transition firings that leads to it. In effect, one is interested in a 'combined' system state which includes both the current marking and the current values of the binary signals. Therefore, if one wants to ensure that a prefix represents the entire state space, some additional information (in this case, the valuation of signal variables) must be taken into account. Clearly, the completeness as sketched above does not guarantee this.

We soon found that the situation can also be a totally opposite one, i.e., the standard notion of completeness can be unnecessarily strong. As an example, one can consider the building of a prefix when there is a suitable notion of symmetric (equivalent) markings, as described in [3]. The idea is then to ensure that each marking of Σ is represented in Unf_Σ either directly or by a symmetric marking. Such an approach may significantly reduce the size of the prefix.

Having analysed examples like these, we have concluded that the original notion of completeness, though sufficient for certain applications, may be too crude and inflexible if one wants to take into consideration more complex semantics of concurrent systems, or their inherent structural properties.

Algorithmics of Prefix Generation The essential feature of the existing unfolding algorithms (see, e.g., [5,8,13]) 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 (in other words, at run-time). Such a treatment was quite pragmatic and worked reasonably well. But, in more complicated situations, the dynamic notion of a cut-off event may hinder defining appropriate algorithms and, in particular, proving their correctness. This has become apparent when dealing with a parallel algorithm for generating prefixes in [8], where the degree of possible non-determinism brought up both these issues very clearly. To conclude, the algorithm-dependent notion of a cut-off event is increasingly difficult to manage.

There is also an important aspect linking cut-off events and completeness, which was somewhat overlooked in previous works. To start with, the notion of a complete prefix given in [5] 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 [13] 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 [7,9,15] 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], 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, it is natural to formally link cut-off events with the notion of completeness, closing up a rather uncomfortable gap between theory and practice.

The New Approach In order to address issues of semantical meaning and algorithmic pragmatics relating to the finite prefixes of Petri net unfoldings, we propose a parametric set-up in which questions concerning, e.g., completeness and cut-off events, could be discussed in a uniform and general way. One parameter captures the information we intend to retain in a complete prefix, while the other two specify under which circumstances a given event can be designated as a cut-off event. Crucially, we decided to shift the emphasis from markings to the execution histories of Σ, and the former parameter, a suitably defined equivalence relation $\approx$, specifies which executions can be regarded as equivalent. Intuitively, one has to retain at least one representative execution from each equivalence class of $\approx$. (The standard case in [5,13] is then covered by regarding two executions as equivalent iff they reach the same marking.)

For efficiency reasons, the existing unfolding algorithms usually consider only 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 if the size of the resulting prefix is of paramount importance (see, e.g., [6]). As a result, the final definition of the set-up, called here a *cutting context*, contains besides an adequate order (as in [5]) a parameter which specifies precisely those configurations which can be used to designate an event as a cut-off event. For a given equivalence relation $\approx$, we then define what it means for a

[1] According to the notion of completeness presented in [5], 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 1).

prefix to be complete. In essence, we require that all equivalence classes of $\approx$ are represented, and that any history involving no cut-off events can be extended (in a single step) in exactly the same way as in the full unfolding.

The definition of a cutting context leads to our central result, the *algorithm-independent* notion of a cut-off event and the related unique *canonical* prefix; the latter is shown to be complete w.r.t. our new notion of completeness. Though the canonical prefix is always complete, it may still be infinite, making it unusable for model checking. We therefore investigate what guarantees the finiteness of the canonical prefix and, in doing so, formulate and prove a version of König's Lemma for unfoldings of (possibly unbounded) Petri nets.

To summarise, this paper addresses both semantical and algorithmic problems using a single device, namely the canonical prefix. The theoretical notion of a complete prefix is useful as long as it can be the basis of a practical prefix-building algorithm. We show that this is indeed the case, generalising the already proposed unfolding algorithm presented in [5] as well as the parallel algorithm from [8]. We believe that the above approach results in a more elegant framework for investigating issues relating to unfolding prefixes, and provides a powerful and flexible tool to deal with different variants of the unfolding technique. All proofs can be found in the technical report [10].

2 Basic Notions

In this section, we first present basic definitions concerning Petri nets, and then recall (see also [4,5]) notions related to net unfoldings.

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., $M : P \to \{0, 1, 2, \ldots\}$. 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 $z \in P \cup T$. 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 and an *initial* marking M_0. We assume the reader is familiar with the standard notions of the theory of Petri nets, such as the *enabledness* and *firing* of a transition, marking *reachability*, and net *boundedness* and (1-)*safeness*. We will denote the set of reachable markings of Σ by $\mathcal{M}(\Sigma)$.

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) (i.e., $B' \subseteq B$, $E' \subseteq E$ and $G' = G \cap (B' \times E' \cup E' \times B')$) 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 net system Σ there exists a unique (up to isomorphism) maximal (w.r.t. $\sqsubseteq$) branching process Unf_Σ^{max}, called the *unfolding* of Σ.

For convenience, we assume a branching process to start with a (virtual) *initial event* $\perp$, which has the postset $Min(ON)$, empty preset, and no label. We do not represent $\perp$ in figures nor treat it explicitly in algorithms.

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 initial event $\perp$, we additionally require that $\perp \in C$. For $e \in E$, 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. $\mathcal{C}^\pi_{loc}$) — or simply by $\mathcal{C}_{fin}$ (resp. $\mathcal{C}_{loc}$) if $\pi = Unf_\Sigma^{max}$.

A *co-set* is a set of mutually concurrent conditions. 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 such marking is reachable in 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 $Unf_\Sigma^{max} = (B, E, G, h)$ is its unfolding.

König's Lemma for Branching Processes König's Lemma (see [12]) states that a finitely branching, rooted, directed acyclic graph with infinitely many

nodes reachable from the root has an infinite path. It turns out that a version of such a result holds for branching processes of Petri nets.

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

Note that the above result does not follow directly from the original König's Lemma [12], since the conditions of π can have infinitely many outgoing arcs.

3 Complete Prefixes of Petri Net Unfoldings

As explained in the introduction, there exist several 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. Here 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.

Cutting Contexts For flexibility, our new set-up is parametric. The first parameter determines the information to be preserved in a complete prefix (in the standard case, the set of reachable markings). The main idea here is to shift the emphasis from the reachable markings of Σ to the finite configurations of Unf_Σ^{max}. Formally, the information to be preserved in the prefix corresponds to the equivalence classes of some equivalence relation $\approx$ on $\mathcal{C}_{fin}$. The other two parameters are more technical: they specify under which circumstances 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. [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[2] 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 $\mathcal{C}_{fin}$, i.e., $\mathcal{C}_e \subseteq \mathcal{C}_{fin}$ for all $e \in E$. ◇

[2] Unlike [5], 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 may be undefined if $Mark(C') \neq Mark(C'')$, and thus cannot be used in our generalised settings.

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 [6]), and at least the following three kinds of the equivalence $\approx$ have been used:

- $C' \approx_{mar} C''$ if $Mark(C') = Mark(C'')$. This is the most widely used equivalence (see [5,6,8,13]). 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 [16] for unfolding Signal Transition Graphs (STGs) specifying asynchronous circuits.
- $C' \approx_{sym} C''$ 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].

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], 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 $[e] \lhd [f]$. Since $\lhd$ is a well-founded partial order on the set of events, we can use Noetherian induction (see [2]) 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 $f \lhd e$.

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}. $\diamond$

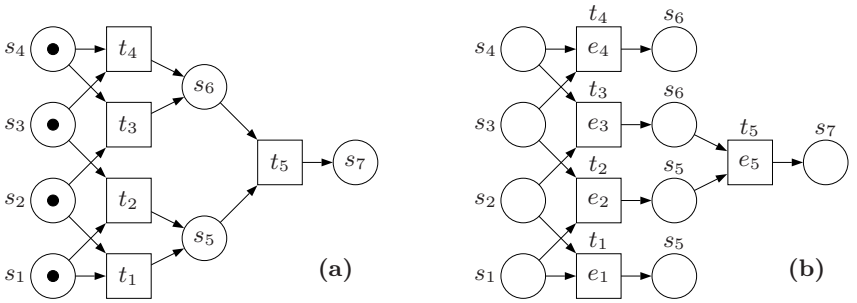

Fig. 1. A Petri net **(a)** and one of its branching processes **(b)**, which is complete according to [5], but not w.r.t. Definition 4: the configuration $\{e_1, e_4\}$ does not preserve firings and introduces a fake deadlock. To make the prefix complete w.r.t. Definition 4, one has to add an instance of t_5 consuming the conditions produced by e_1 and e_4

Note that π remains complete following the removal of all events e for which $\langle e \rangle \cap E_{cut} \neq \emptyset$, after which 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 $\mathcal{C}^{\pi}_{fin}$, provided that π is complete. 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], since it requires *all* configurations in $\mathcal{C}^{\pi}_{fin}$ containing no events from E_{cut} to preserve all transition firings, rather than the *existence* of a configuration preserving all firings. The justification why such a stronger property is desirable, e.g., for deadlock detection, was given in the introduction (see also Figure 1). Obviously, our notion is strictly stronger than the one in [5], i.e., it implies the completeness in the sense of [5], but not vice versa (see Figure 1). However, the proof of completeness in [5] almost gives the stronger notion; we have adopted it (see [10, Proposition 8]) with relatively few modifications.

4 Canonical Prefix

This and the next section develop our central results. First, we define cut-off events without resorting to any algorithmic argument. This yields a definition of the canonical prefix, and we then establish several of its relevant properties.

Static Cut-Off Events In [5], the definition of a cut-off event was algorithm-specific, and given w.r.t. the already built part of a prefix. Here 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 sets of *feasible* events, denoted by $fsble_\Theta$, and *static cut-off* events, denoted by cut_Θ, are two sets of events e of Unf_Σ^{max} defined thus:

1. An event e is a feasible event if $\langle e\rangle \cap cut_\Theta = \emptyset$.
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 $C \subseteq fsble_\Theta \setminus cut_\Theta$, $C \approx [e]$, and $C \lhd [e]$. Any C satisfying these conditions will be called a *corresponding* configuration of e. ◇

Note that $fsble_\Theta$ and cut_Θ are well-defined sets due to Noetherian induction. 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 $fsble_\Theta$ or in cut_Θ. And, by Proposition 3(2), the same holds for the events in any configuration C satisfying $C \lhd [e]$.

Since $\langle\bot\rangle = \emptyset$, $\bot \in fsble_\Theta$ by the above definition. Furthermore, $\bot \notin cut_\Theta$, since $\bot$ cannot have a corresponding configuration. Indeed, $[\bot] = \{\bot\}$ is the smallest (w.r.t. set inclusion) configuration, and so $\lhd$-minimal by Definition 2(2).

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. Then a corresponding local configuration C of a cut-off event e defined in this way may contain 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 naïve approach when Θ is saturated. ◇

Canonical Prefix and Its Properties Once we have defined feasible events, the following notion arises quite naturally. The *canonical* prefix of Unf_Σ^{max} is the unique branching process Unf_Σ^Θ, whose set of events is $fsble_\Theta$, and whose conditions are the conditions adjacent to these events. Thus Unf_Σ^Θ is uniquely determined by the cutting context Θ.

In what follows, we present several fundamental properties of Unf_Σ^Θ. We stress that, unlike those given in [5], their proofs are not algorithm-specific (see [10]).

Theorem 6. *Unf_Σ^Θ is complete w.r.t. $E_{cut} = cut_\Theta$.*

Having established that the canonical prefix is always complete, we now set out to analyse its finiteness. A necessary and sufficient condition for the latter follows directly from our version of König's Lemma for branching processes.

Theorem 7. *Unf_Σ^Θ is finite iff there is no infinite $\prec$-chain of feasible events.*

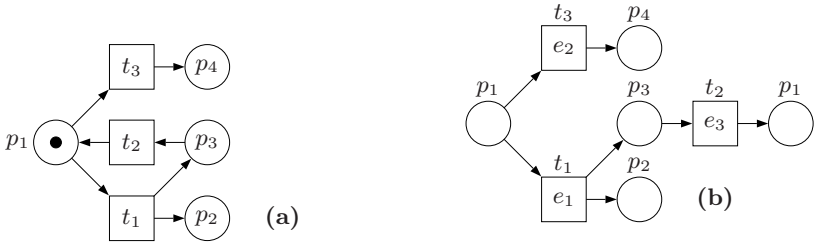

Fig. 2. An unbounded net system **(a)** and its canonical prefix **(b)**. The cutting context is such that $C' \approx C'' \Leftrightarrow Mark(C') \cap \{p_1, p_3, p_4\} = Mark(C'') \cap \{p_1, p_3, p_4\}$ and $\{\bot\} \in \mathcal{C}_{e_3}$, and so e_3 is a static cut-off event

Thus, in order to get a finite canonical prefix, one should choose a cutting context such that the $\mathcal{C}_e$'s contain enough configurations, and $\approx$ is coarse enough, to cut each infinite $\prec$-chain. Interestingly, certain cutting contexts sometimes give a finite canonical prefix even for unbounded net systems. Figure 2(a) shows a net modelling a loop, where place p_2, used for counting the number of iterations, is unbounded. If $\approx$ ignores the value of this 'counter' place, it is possible to build a finite and complete canonical prefix, shown in Figure 2(b).

The following result provides quite a tight and practical indication for deciding whether Unf^{Θ}_{Σ} is finite or not.

Proposition 8. *If $|\mathfrak{R}^{fin}_{\approx}| < \infty$ and Θ is dense, then Unf^{Θ}_{Σ} is finite. If $|\mathfrak{R}^{fin}_{\approx}| = \infty$, then Unf^{Θ}_{Σ} is infinite.*

Corollary 9. *Let $\approx$ be either of $\approx_{mar}$, $\approx_{code}$, $\approx_{sym}$. If Σ is bounded and Θ is dense, then Unf^{Θ}_{Σ} is finite. If Σ is unbounded, then Unf^{Θ}_{Σ} is infinite.*

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

Theorem 10. *Suppose that $\lhd$ is total, $|\mathfrak{R}^{loc}_{\approx}| < \infty$, and the following holds: For every $\mathcal{R} \in \mathfrak{R}^{loc}_{\approx}$, there is $\gamma_{\mathcal{R}} > 0$ 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 $|\, fsble_{\Theta} \setminus cut_{\Theta}\,| \leq \sum_{\mathcal{R} \in \mathfrak{R}^{loc}_{\approx}} \gamma_{\mathcal{R}}$.*

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 [5] 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 Unf^{Θ}_{Σ} in this case is equal to $|\mathcal{M}(\Sigma)|$. Using the above theorem, 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):

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 $Sl \in \textsc{Slices}(pe)$
  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** $cut\_off \leftarrow cut\_off \cup \{e\}$
    $pe \leftarrow \textsc{PotExt}(Pref_\Sigma)$
  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. 3. Unfolding algorithm with slices

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

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 Algorithms for Generating Canonical Prefixes

It turns out that canonical prefixes can be generated by straightforward generalisations of the existing unfolding algorithms (see, e.g., [5,8]). The *slicing* algorithm from [8], parameterised by a cutting context Θ, is shown in Figure 3. (The algorithm proposed in [5] is a special case.) It is assumed that the function $\textsc{PotExt}(Pref_\Sigma)$ finds the set of *possible extensions* of a branching process $Pref_\Sigma$, according to the following definition.

Definition 11. For a branching process π of Σ, a *possible extension* 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. We will take the pair (t, D) as a t-labelled event having D as its preset. ◇

Compared to the standard unfolding algorithm in [5], the slicing algorithm has the following modifications in its main loop. A set of events Sl, called a *slice*,

is chosen on each iteration and processed as a whole, without taking or adding any events from or to pe. A slice must satisfy the following conditions:

- Sl is a non-empty subset of the current set of possible extensions pe;
- for every $e \in Sl$ 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 one from [5]: it may be non-deterministic even if the order $\lhd$ is total. Since the events in the current slice can be processed completely independently, the slicing algorithm admits efficient parallelisation (along the lines proposed in [8]). A crucial property of the slicing unfolding algorithm is that it generates the canonical prefix.

Theorem 12. *If Unf_Σ^Θ is finite, then the slicing algorithm generates Unf_Σ^Θ in a finite number of steps.*

As far as this paper is concerned, the above theorem completes our investigation. What remains is to put this section in the context of the previous work. In the case $\Theta = \Theta_{ERV}$ the slicing algorithm is nothing more but the algorithm proposed in [8]. Moreover, by setting $\textsc{Slices}(pe) \stackrel{\text{df}}{=} \{\{e\} \mid e \in \min_\lhd pe\}$, one can obtain the unfolding algorithm of [5]. For the slicing algorithm, the correctness was proven (in a very complicated way) in [8] by showing that it is equivalent to the unfolding algorithm of [5], in the sense that prefixes produced by arbitrary runs of these algorithms are isomorphic (and then relying on the correctness results developed in [5]). The theory developed in this paper allows for a much more elegant and general proof, essentially by showing that arbitrary runs of both these algorithms generate the canonical prefix. Moreover, one should not forget that the notion of completeness developed in this paper is strictly stronger than that used in previous works; in particular, algorithms shown correct here are also correct w.r.t. the weaker notion.

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). The canonical prefix is complete w.r.t. this notion, and it is exactly the prefix generated by arbitrary runs of the non-deterministic unfolding algorithms presented in [5,8]. This gives a new

correctness proof for the unfolding algorithms presented there, which is much simpler in the case of the algorithm developed in [8]. 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 gave conditions for the finiteness of the canonical prefix and, in certain cases, the upper bounds on its size, which are helpful in choosing problem-specific cutting contexts. To deal with the finiteness problem, we developed 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, EPSRC grants GR/M99293 and GR/M94366 (MOVIE), and an ARC grant JIP.

References

1. E. M. Clarke, O. Grumberg, and D. Peled: *Model Checking*. MIT Press (1999). 582
2. P. M. Cohn: *Universal Algebra*. Reidel, 2nd edition (1981). 588
3. J.-M. Couvreur, S. Grivet and D. Poitrenaud: Unfolding of Products of Symmetrical Petri Nets. Proc. of *ICATPN'2001*. Springer LNCS 2075 (2001) 121–143. 583, 588
4. J. Engelfriet: Branching processes of Petri Nets. *Acta Inf.* 28 (1991) 575–591. 585, 586
5. J. Esparza, S. Römer and W. Vogler: An Improvement of McMillan's Unfolding Algorithm. Proc. of *TACAS'96*. Springer LNCS 1055 (1996) 87–106. 582, 583, 584, 585, 587, 588, 589, 590, 591, 592, 593
6. K. Heljanko: Minimizing Finite Complete Prefixes. Proc. of *CS&P'99*, Warsaw, Poland (1999) 83–95. 584, 588
7. K. Heljanko: Using Logic Programs with Stable Model Semantics to Solve Deadlock and Reachability Problems for 1-Safe Petri Nets. *Fund. Inf.* 37(3) (1999) 247–268. 584
8. K. Heljanko, V. Khomenko and M. Koutny: Parallelisation of the Petri Net Unfolding Algorithm. Proc. of *TACAS'02*. Springer LNCS 2280 (2002) 371–385. 582, 583, 585, 588, 592, 593, 594
9. V. Khomenko and M. Koutny: LP Deadlock Checking Using Partial Order Dependencies. Proc. of *CONCUR'2000*. Springer LNCS 1877 (2000) 410–425. 584
10. V. Khomenko, M. Koutny and W. Vogler: Canonical Prefixes of Petri Net Unfoldings. Techn. Rep. CS-TR-741, Dept. of Comp. Sci., Univ. of Newcastle (2001). 585, 589, 590
11. V. Khomenko, M. Koutny and A. Yakovlev: Detecting State Coding Conflicts in STGs Using Integer Programming. Proc. of *DATE'02*. IEEE (2002) 338–345. 583

12. 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) 586, 587
13. K. L. McMillan: Using Unfoldings to Avoid State Explosion Problem in the Verification of Asynchronous Circuits. Proc. of *CAV'92*. LNCS 663 (1992) 164–174. 582, 583, 584, 588
14. K. L. McMillan: *Symbolic Model Checking*. PhD thesis, CMU-CS-92-131 (1992).
15. S. Melzer and S. Römer: Deadlock Checking Using Net Unfoldings. Proc. of *CAV'97*. Springer LNCS 1254 (1997) 352–363. 584
16. A. Semenov: *Verification and Synthesis of Asynchronous Control Circuits Using Petri Net Unfolding*. PhD Thesis, University of Newcastle upon Tyne (1997). 583, 588