HAL
open science

Goal-Driven Unfolding of Petri Nets

Thomas Chatain, Loïc Paulevé

To cite this version:

Thomas Chatain, Loïc Paulevé. Goal-Driven Unfolding of Petri Nets. 28th International Conference on Concurrency Theory (CONCUR 2017), Sep 2017, Berlin, Germany. 10.4230/LIPIcs.CONCUR.2017.14 . hal-01392203v1

HAL Id: hal-01392203

https://hal.science/hal-01392203v1

Submitted on 29 Jun 2017 (v1), last revised 4 Jul 2017 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Goal-Driven Unfolding of Petri Nets

Thomas Chatain[1] and Loïc Paulevé[2]

[1] LSV, ENS Cachan, INRIA, CNRS, Université Paris-Saclay, France
[2] LRI UMR 8623, Univ. Paris-Sud, CNRS, Université Paris-Saclay, France

Abstract. Unfoldings provide an efficient way to avoid the state-space explosion due to interleavings of concurrent transitions when exploring the runs of a Petri net. The theory of adequate orders allows one to define finite prefixes of unfoldings which contain all the reachable markings. In this paper we are interested in reachability of a single given marking, called the goal. We propose an algorithm for computing a finite prefix of the unfolding of a 1-safe Petri net that preserves all minimal configurations reaching this goal. Our algorithm combines the unfolding technique with on-the-fly model reduction by static analysis aiming at avoiding the exploration of branches which are not needed for reaching the goal. We present some experimental results.

1 Introduction

Analysing the possible dynamics of a concurrent system expressed as Petri nets can be eased by means of unfoldings and their prefixes which avoid exploring redundant interleaving of transitions.

In this paper, we propose a method which combines the unfolding technique with model reduction in order to explore efficiently and completely the minimal configurations (partially ordered occurrences of transitions) which lead to a given goal marking/marked place. In particular, we aim at ignoring configurations that cannot reach the goal, but also configurations containing transient cycles.

The goal-driven unfolding relies on calling, on the fly, an external model reduction procedure which identifies transitions not part of any minimal configuration for the goal reachability from the current marking. Those useless transitions are then skipped by the unfolding.

We show how model reduction can be applied to the unfolding of a safe Petri net $\mathcal{N}$ in such a way that it preserves minimal configurations. Then we present an algorithm to construct a corresponding goal-driven finite prefix.

We illustrate this procedure on the Petri net of Figure 1. The goal is $\{p'_3, p_5\}$. Notice that only one occurrence of t_3 is needed to reach the goal. So, after the corresponding event, t_3 can be declared useless. Also, after firing t_1, t'_2 is fireable but firing it makes the goal unreachable. Therefore, a reduction procedure may declare that t'_2 is useless once t_1 has occurred, allowing one to avoid exploring this branch. Symmetrically, t'_1 is useless once t_2 has occurred. It is easy to imagine a larger model where a large piece of behaviour would be reachable from $\{p_1, p'_2, p_3\}$ (but would not allow to reach the goal); or from $\{p_3, p_4\}$ (but

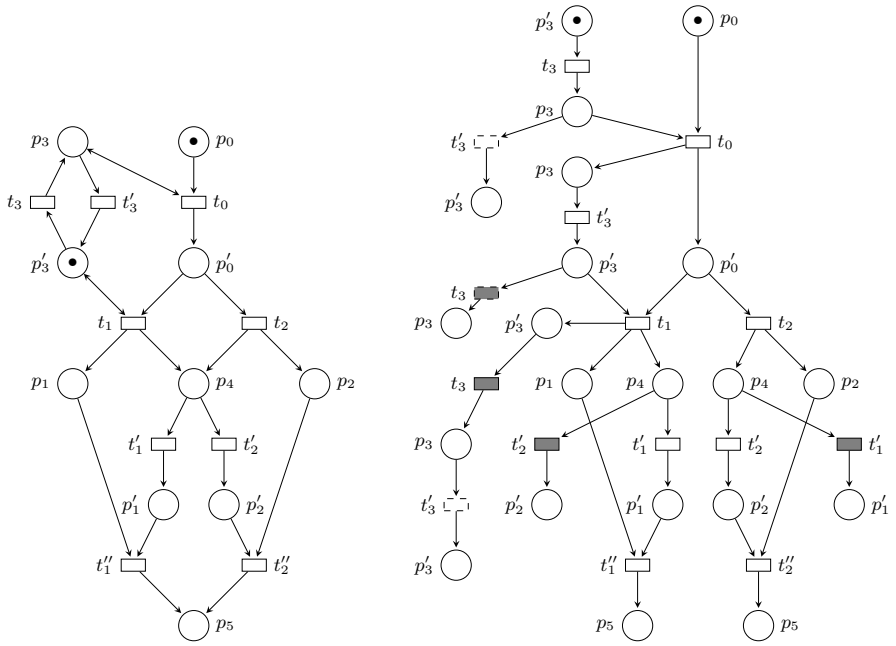

Fig. 1. A safe Petri net (left) and a finite complete prefix (right) of its unfolding. Dashed events are flagged as *cut-offs*: the unfolding procedure does not continue beyond them. Events in gray can be declared as useless by the reduction procedure for $\{p'_3, p_5\}$ reachability, and can be skipped during the goal-driven prefix computation.

would involve transient cycles): the usual complete finite prefix would explore such configurations, while our model reduction can avoid their computation.

The design of the model reduction procedure which identifies useless transitions is out of the scope of the paper. Instead, we consider it as a blackbox, and design our approach assuming the reduction preserves all the minimal (acyclic) sequences of transitions leading to the goal. Moreover, to be of practical interest, the reduction should show a complexity lower than the reachability problem (PSPACE-complete [7]).

As detailed in Section 4.2, skipping transitions declared useless by a reduction procedure involves non-trivial modifications to the algorithm for computing the prefix of the unfolding. Indeed, a particular treatment of cut-offs has to be introduced in order to ensure that the resulting goal-driven prefix includes all the minimal sequences of transitions.

The goal-driven unfolding has practical applications in systems biology [19]. Indeed, numerous dynamical properties relevant for biological networks focus on the reachability of the activity of a particular node in the network, typically a transcription factor known to control a given cellular phenotype. In this perspective, having computational methods that can be tailored for such narrow reachability properties is of practical interest. The *completeness of the minimal*

sequences of transitions for the goal reachability is *critical* for several analyses of biological system dynamics. An example is the identification of parts of the network that play a central role to activate a node of interest. By altering such parts (e.g., with mutations) one can expect prevent such an activation [18]. If the analysis considers only a partial set of minimal sequences, there is no guarantee that the predicted mutations are sufficient to prevent the goal reachability.

Related Work. Numerous work address the computation of reachable states in concurrent systems using unfoldings. [12] compares several algorithms for checking reachability based on a previously computed finite complete prefix of a Petri net. [2] defines over-approximations of the unfolding (i.e., which contains all the reachable markings, but potentially more) for graph transformation systems.

Despite the negative result [9] which states that depth-first-search strategies are not correct for classical unfolding algorithms, [3] defines *directed unfolding* of Petri nets, which is closely related to our goal-driven unfolding. They rely on a heuristic function (on configurations) to generate an ordering of the events for making a given transition appear as soon as possible during the unfolding. In addition, they can consider heuristic functions to detect configuration from which the goal transition is not reachable. In such a case, no extension will be made to that configuration, which may significantly prune the computed prefix. The major difference with the work presented in the paper is that directed unfolding does not prune transitions leading to spurious transient cycles on the way to the goal. Actually, in their terms, our reduction procedure would not be considered *safely pruning* because we discard (non-minimal) configurations reaching the goal. In a sense, the reduction they achieve on the prefix size corresponds to the extreme case when our external reduction procedure returns the full model if the goal is reachable, and the empty model if not. Indeed, except for the case when the goal is detected as non-reachable, all the other configurations are kept in the directed unfolding, whereas our approach can potentially output a prefix containing only, but all, minimal configurations for the goal reachability.

Less related to our work, static analysis techniques were also used in combination with partial order reductions. [13,21] rely on an on-the-fly detection of independence relations by static analysis, to improve partial order reductions.

Outline. Section 2 gives the basics of Petri net unfoldings and of their complete finite prefixes. The concepts of minimal configuration and model reduction are introduced in Section 3, and Section 4 details the goal-driven unfolding and prefix with proofs of completeness. Finally, Section 5 applies the goal-driven prefix to actual biological models, and Section 6 concludes this paper.

2 Unfoldings of Petri nets

In this section, we explain the basics of Petri net unfoldings. A more extensive treatment of the theory explained here can be found, e.g., in [8]. Roughly speaking, the unfolding of a Petri net $\mathcal{N}$ is an "acyclic" Petri net $\mathcal{U}$ that has

the same behaviours as $\mathcal{N}$ (modulo homomorphism). In general, $\mathcal{U}$ is an infinite net, but if $\mathcal{N}$ is safe, then it is possible [16] to compute a finite prefix $\mathcal{P}$ of $\mathcal{U}$ that is "complete" in the sense that every reachable marking of $\mathcal{N}$ has a reachable counterpart in $\mathcal{P}$. Thus, $\mathcal{P}$ represents the set of reachable markings of $\mathcal{N}$. Figure 1 shows a Petri net and a finite complete prefix of its unfolding.

We now give some technical definitions to introduce unfoldings formally.

Definition 1 ((Safe) Petri Net). *A* (safe) Petri net *is a tuple* $\mathcal{N} = \langle P, T, F, M_0 \rangle$ *where* P *and* T *are sets of* nodes *(called* places *and* transitions *respectively), and* $F \subseteq (P \times T) \cup (T \times P)$ *is a* flow relation *(whose elements are called* arcs*). A subset* $M \subseteq P$ *of the places is called a marking, and* M_0 *is a distinguished* initial marking.

For any node $x \in P \cup T$, we call *pre-set* of x the set ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$ and *post-set* of x the set $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$. These notations are extended to sets $Y \subseteq P \cup T$, with ${}^\bullet Y = \cup_{x \in Y} {}^\bullet x$ and $Y^\bullet = \cup_{x \in Y} x^\bullet$.

A transition $t \in T$ is *enabled* at a marking M if and only if ${}^\bullet t \subseteq M$. Then t can *fire*, leading to the new marking $M' = (M \setminus {}^\bullet t) \cup t^\bullet$. We write $M \xrightarrow{t} M'$. A *firing sequence* is a (finite or infinite) word $w = t_1 t_2 \ldots$ over T such that there exist markings $M_1, M_2, \ldots$ such that $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \ldots$ For any such firing sequence w, the markings $M_1, M_2, \ldots$ are called *reachable markings*.

The Petri nets we consider are said to be *safe* because we will assume that any reachable marking M is such that for any $t \in T$ that can fire from M leading to M', the following property holds: $\forall p \in M \cap M'$, $p \in {}^\bullet t \cap t^\bullet \vee p \notin {}^\bullet t \cup t^\bullet$.

Figure 1 (left) shows an example of a safe Petri net. The places are represented by circles and the transitions by rectangles (each one with a label identifying it). The arrows represent the arcs. The initial marking is represented by dots (or tokens) in the marked places.

Definition 2 (Causality, conflict, concurrency). *Let* $\mathcal{N} = \langle P, T, F, M_0 \rangle$ *be a net and* $t, t' \in T$ *two transitions of* $\mathcal{N}$*. We say that* t *is a* causal predecessor *of* t'*, noted* $t < t'$*, if there exists a non-empty path of arcs from* t *to* t'*. We note* $t \leq t'$ *if* $t < t'$ *or* $t = t'$*. If* $t \leq t'$ *or* $t' \leq t$*, then* t *and* t' *are said to be* causally related*. The set of causal predecessors of* t *is denoted* $\lfloor t \rfloor$*. We write* $\lceil t \rceil$ *for* $\lfloor t \rfloor \cup \{t\}$*, which we call the* causal past *of* t*. Transitions* t *and* t' *are* in conflict*, noted* $t \# t'$*, if there exist* $u, v \in T$ *such that* $u \neq v$*,* $u \leq t$*,* $v \leq t'$ *and* ${}^\bullet u \cap {}^\bullet v \neq \emptyset$*. We call* t *and* t' concurrent*, noted* $t \; co \; t'$*, if they are neither causally related nor in conflict.*

As we said before, an unfolding is an "acyclic" net. This notion of acyclicity is captured by Definition 3. As is convention in the unfolding literature, we shall refer to the places of an occurrence net as *conditions* and to its transitions as *events*. Due to the structural constraints, the firing sequences of occurrence nets have special properties: if some condition c is marked during a run, then the token on c was either present initially or produced by one particular event (the single event in ${}^\bullet c$); moreover, once the token on c is consumed, it can never be replaced by another token, due to the acyclicity constraint on $<$.

Definition 3 (Occurrence net). *An* occurrence net $\mathcal{O} = \langle P, T, G, M_0 \rangle$ *is a Petri net* $\langle P, T, F, M_0 \rangle$ *with* $P = C$, $T = E$, $F = G$, $M_0 = C_0$ *for which:*

1. *The causality relation* $<$ *is acyclic;*
2. $|{}^\bullet p| \leq 1$ *for all places* p*, and* $p \in M_0$ *iff* $|{}^\bullet p| = 0$*;*
3. *for every transition* t*,* $t \# t$ *does not hold, and* $\{x \mid x \leq t\}$ *is finite.*

Definition 4 (Configuration, cut). *Let* $\mathcal{O} = \langle C, E, G, C_0 \rangle$ *be an occurrence net. A set* $\mathcal{C} \subseteq E$ *is called* configuration *(or* process*) of* $\mathcal{O}$ *if (i)* $\mathcal{C}$ *is* causally closed*, i.e. for all* $e, e' \in E$ *with* $e' < e$*, if* $e \in \mathcal{C}$ *then* $e' \in \mathcal{C}$*; and (ii)* $\mathcal{C}$ *is conflict-free, i.e. if* $e, e' \in \mathcal{C}$*, then* $\neg(e \# e')$*. The* cut *of* $\mathcal{C}$*, denoted* $Cut(\mathcal{C})$*, is the set of conditions* $(C_0 \cup \mathcal{C}^\bullet) \setminus {}^\bullet\mathcal{C}$*.*

An occurrence net $\mathcal{O}$ with a net homomorphism h mapping its conditions and events to places and transitions of a net $\mathcal{N}$ is called a *branching process* of $\mathcal{N}$. Intuitively, a configuration of $\mathcal{O}$ is a set of events that can fire during a firing sequence of $\mathcal{N}$, and its cut is the set of conditions marked after that sequence.

Unfolding. Let $\mathcal{N} = \langle P, T, F, M_0 \rangle$ be a safe Petri net. The unfolding $\mathcal{U} = \langle C, E, G, C_0 \rangle$ of $\mathcal{N}$ is the unique (up to isomorphism) maximal branching process such that the firing sequences and reachable markings of $\mathcal{U}$ represent exactly the firing sequences and reachable markings of $\mathcal{N}$ (modulo h). $\mathcal{U}$ is generally infinite but its conditions and events can be inductively constructed as follows:

1. The conditions C are a subset of $(E \cup \{\bot\}) \times P$. For a condition $c = \langle x, p \rangle$, we will have $x = \bot$ iff $c \in C_0$; otherwise x is the singleton event in ${}^\bullet c$. Moreover, $h(c) = p$. The initial marking C_0 contains one condition $\langle \bot, p \rangle$ per initially marked place p of $\mathcal{N}$.
2. The events E are a subset of $2^C \times T$. More precisely, we have an event $e = \langle C', t \rangle$ for every set $C' \subseteq C$ such that $c \; co \; c'$ holds for all $c, c' \in C'$ and $\{\, h(c) \mid c \in C' \,\} = {}^\bullet t$. In this case, we add edges $\langle c, e \rangle$ for each $c \in C'$ (i.e. ${}^\bullet e = C'$), we set $h(e) = t$, and for each $p \in t^\bullet$, we add to C a condition $c = \langle e, p \rangle$, connected by an edge $\langle e, c \rangle$.

Intuitively, a condition $\langle x, p \rangle$ represents the possibility of putting a token onto place p through a particular firing sequence, while an event $\langle C', t \rangle$ represents a possibility of firing transition t in a particular context.

Every firing sequence σ is represented by a configuration of $\mathcal{U}$; we denote this configuration $\mathcal{K}(\sigma)$. Conversely, every configuration $\mathcal{C}$ of $\mathcal{U}$ represents one or several firing sequences ($\mathcal{K}$ is not injective in general); these firing sequences are equivalent up to permutation of concurrent transitions. Their (common) resulting marking corresponds, due to the construction of $\mathcal{U}$, to a reachable marking of $\mathcal{N}$. This marking is defined as $Mark(\mathcal{C}) := \{\, h(c) \mid c \in Cut(\mathcal{C}) \,\}$.

Finite Complete Prefix. The unfolding $\mathcal{U}$ of a finite safe Petri net $\mathcal{N}$ is infinite in general, but it shows some regularity because $\mathcal{N}$ has finitely many markings and two events e and e' having $Mark(\lceil e \rceil) = Mark(\lceil e' \rceil)$ have isomorphic extensions.

It is known [16,11] that one can construct a *finite complete prefix* $\mathcal{P}$ of $\mathcal{U}$, i.e. a causally closed set E' of events of $\mathcal{U}$ which is sufficiently large for satisfying the following: for every reachable marking M of $\mathcal{N}$ there exists a configuration $\mathcal{C}$ of $\mathcal{P}$ such that $Mark(\mathcal{C}) = M$. One can even require that for each transition t of $\mathcal{N}$ enabled in M, there is an event $\langle C, t\rangle \in E'$ enabled in $Cut(\mathcal{C})$.

The idea of the construction is to explore the future of only one among the events e having equal $Mark(\lceil e \rceil)$. The selected event is the one having minimal $\lceil e \rceil$ w.r.t. a so-called *adequate order* on the finite configurations of $\mathcal{U}$. The others are flagged as *cut-offs*; they do not "contribute any new reachable markings". These events are represented by dashed lines in Figure 1.

Definition 5 (Adequate orders). *A strict partial order $\lhd$ on the finite configurations of the unfolding of a safe Petri net $\mathcal{N}$ is called* adequate *if:*

- *it refines (strict) set inclusion $\subsetneq$, i.e. $C \subsetneq C'$ implies $C \lhd C'$, and*
- *it is preserved by finite extensions, i.e. for every pair of configurations C, C' such that $Mark(C) = Mark(C')$ and $C \lhd C'$, and for every finite extension D of C, the finite extension D' of C' which is isomorphic to D satisfies $C \uplus D \lhd C' \uplus D'$.*

The initial definition of adequate orders [11] also requires that $\lhd$ is well founded, but [6] showed that, for unfoldings of safe Petri nets, well-foundedness is a consequence of the other requirements.

Efficient tools [20,14] exist for computing finite complete prefixes.

3 Goal-Oriented Model Reduction

The goal-driven unfolding relies on model reduction procedures which preserve minimal firing sequence to reach a given goal g. These reductions aim at removing as many transitions as possible among those that do not participate in any minimal firing sequence. This section details the properties required by our method and introduce several notations used in the rest of the paper.

Definition 6 (Minimal firing sequence). *A firing sequence $t_1 \dots t_n$ of a Petri net $\mathcal{N} = \langle P, T, F, M_0\rangle$ visiting markings $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \dots \xrightarrow{t_n} M_n$ is said* cycling *if it visits twice the same marking, i.e. $M_i = M_j$ for some $0 \leq i < j \leq n$. A* minimal firing sequence *of $\mathcal{N}$ to a goal g is a firing sequence $t_1 \dots t_n$ leading to g which has no feasible permutation*[3] *being a cycling firing sequence of $\mathcal{N}$.*

For example with Petri net of Fig. 1 and considering the goal $\{p'_3, p_5\}$, $t_3t_0t'_3t_2t_3t'_2t''_2t'_3$ is not minimal because its permutation $t_3t_0t'_3t_3t_2t'_2t''_2t'_3$ is also feasible and visits the marking $\{p_3, p'_0\}$ twice. Intuitively, the cycle t'_3t_3 can be removed. The minimal firing sequences of $\mathcal{N}$ to the goal are $t_3t_0t'_3t_1t'_1t''_1$, $t_3t_0t_2t'_3t'_2t''_2$ and their feasible permutations, for instance $t_3t_0t_2t'_2t''_2t'_3$.

[3] Contrary to what is common in concurrency theory, we do not necessarily restrict to permutations of independent transitions w.r.t. an independence relation.

Remark 1. Alternatively, the goal can be seen not as a marking but simply as a set of places to be marked together, possibly with others. Then, one is looking for sequences reaching any marking M with $g \subseteq M$. For minimality, we would then require additionally that no intermediate marking reached before the end of the sequence marks the places in g (and the same for its permutations).

Definition 7 (Minimal configuration). *A* minimal configuration *of a Petri net $\mathcal{N}$ to a goal g is a configuration $E = \mathcal{K}(\sigma)$ for some minimal firing sequence σ of $\mathcal{N}$ to g. Notice that, since all the other σ' such that $E = \mathcal{K}(\sigma')$ are permutations of σ, they are all minimal.*

Lemma 1. *The goal g is reachable iff it is reachable by a minimal firing sequence (and, consequently, by a minimal configuration).*

Proof. Assume that g is reachable by a non-minimal firing sequence σ. This means that σ has a permutation $t_1 \dots t_n$ which visits the same marking twice, i.e. $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \dots \xrightarrow{t_i} M_i \dots \xrightarrow{t_j} M_j \dots \xrightarrow{t_n} M_n = g$ with $M_i = M_j$ and $i < j$. Then g is also reachable by the strictly shorter sequence $t_1 \dots t_i t_{j+1} \dots t_n$. This operation can be iterated if needed; it always terminates and gives a minimal firing sequence which reaches the goal g.

Definition 8 (Reduction procedure, useless transitions). *A* reduction procedure *`useless-trs` is a function which outputs, for a safe Petri net $\mathcal{N}$ and a goal $g \subseteq P$, a set `useless-trs`$(\mathcal{N}, g) \subseteq T$ of transitions of $\mathcal{N}$ which do not occur in any* minimal *firing sequence of $\mathcal{N}$ to goal g: for every minimal firing sequence $t_1 \dots t_n$ to goal g, `useless-trs`$(\mathcal{N}, g) \cap \{t_1, \dots, t_n\} = \emptyset$.*

For example, let $\mathcal{N} = \langle P, T, F, M_0 \rangle$ be the Petri net of Fig. 1. All the transitions occur in at least one minimal firing sequence to the goal $g = \{p'_3, p_5\}$, so every reduction procedure outputs `useless-trs`$(\mathcal{N}, g) = \emptyset$. After firing $t_3 t_0 t'_3$, one reaches marking $\{p'_3, p'_0\}$ from which the only minimal firing sequences to g are $t_1 t'_1 t''_1$ and $t_2 t'_2 t''_2$. Hence, a reduction procedure called as `useless-trs` $(\langle P, T, F, \{p'_3, p'_0\}\rangle, g)$ may declare t_0, t_3 and t'_3 useless, or any subset of those.

Given a Petri net $\mathcal{N} = \langle P, T, F, M_0 \rangle$, $\mathcal{N} \setminus$ `useless-trs`$(\mathcal{N}, g)$ denotes the reduced model $\langle P, T', F', M_0 \rangle$ where $T' = T \setminus$ `useless-trs`$(\mathcal{N}, g)$ and $F' = F \cap ((P \times T') \cup (T' \times P))$. Property 1 derives from Def. 8 and Lemma 1.

Property 1. Every reduction procedure preserves reachability of the goal: g is reachable in $\mathcal{N}$ iff it is reachable in $\mathcal{N} \setminus$ `useless-trs`$(\mathcal{N}, g)$.

In the sequel, we aim at iterating the reduction procedures: starting from a model $\mathcal{N} = \langle P, T, F, M_0 \rangle$ and a goal g, we will apply the reduction to $\mathcal{N}$, then explore the reduced net $\mathcal{N} \setminus$ `useless-trs`$(\mathcal{N}, g)$; later on, we will apply again the reduction from a reached state M and compute `useless-trs`$(\mathcal{N}', g)$ with $\mathcal{N}' = \langle P, T, F, M \rangle \setminus$ `useless-trs`$(\mathcal{N}, g)$ allowing to explore a further reduced net $\mathcal{N}' \setminus$ `useless-trs`$(\mathcal{N}', g)$ from M. These iterated calls to the reduction procedure are justified by the following lemma.

Lemma 2. *Any minimal sequence in $\mathcal{N} \setminus$ `useless-trs`$(\mathcal{N}, g)$ is minimal in $\mathcal{N}$.*

Proof. Any firing sequence of $\mathcal{N} \setminus \texttt{useless-trs}(\mathcal{N}, g)$ is a firing sequence of $\mathcal{N}$, and the minimality criterion does not depend on the set of transitions in $\mathcal{N}$.

In the remainder of the paper, for a Petri net $\mathcal{N} = \langle P, T, F, M_0 \rangle$ and any set $I \subseteq T$ and reachable marking M, we write $\texttt{useless-trs}\,(\mathcal{N}, g, M, I)$ for $\texttt{useless-trs}(\langle P, T, F, M \rangle \setminus I, g) \cup I$.

4 Goal-Driven Unfolding

In this section, we first show that model reduction can be performed during the unfolding of a safe Petri net $\mathcal{N}$ while preserving the minimal configurations to the goal. Next we present an algorithm to construct a finite goal-driven prefix which preserves the reachable markings of the goal-driven unfolding.

4.1 Guiding the Unfolding by a Model Reduction Procedure

The principle of the goal-driven unfolding is that, for some events e in the unfolding (at discretion), a model reduction procedure `useless-trs` is called and the transitions declared useless will not be considered in the future of e. More precisely, the reduction procedure is called on the marking $\mathit{Mark}(\lceil e \rceil)$ of the causal past of e.

Notice that the reduction procedure may already have been used on some events in the causal past of e. Then,

- even if `useless-trs` is not called on e, information about useless transitions inherited from the causal predecessors of e can be used (without calling the model reduction procedure), and this will already prune some branches in the future of e;
- if the reduction procedure is called on $\mathit{Mark}(\lceil e \rceil)$, it can take as input the model already reduced by the transitions declared useless after some event in the causal past of e.

Let $\mathcal{U} = \langle C, E, G, C_0 \rangle$ be the full unfolding of a safe Petri net $\mathcal{N}$. Denote E' the set of events on which the reduction procedure is called. The set E' and the reduction procedure define the set of transitions $\mathit{Useless}(e)$ to be ignored in the future of an event $e \in E$. We define $\mathit{Useless}$ inductively as:

$$\mathit{Useless}(e) \stackrel{\text{def}}{=} \begin{cases} \bigcup_{e' \in \lfloor e \rfloor} \mathit{Useless}(e') & \text{if } e \notin E' \\ \texttt{useless-trs}\left(\mathcal{N}, g, \mathit{Mark}(\lceil e \rceil), \bigcup_{e' \in \lfloor e \rfloor} \mathit{Useless}(e')\right) & \text{if } e \in E'. \end{cases}$$

Thus, every event $e = \langle C, t \rangle \in E$ such that $t \in \mathit{Useless}(e')$ for some $e' \in \lfloor e \rfloor$, is discarded from the goal-driven unfolding. Denote E_{Ignored} the set of such events.

It remains to define the goal-driven unfolding as the maximal prefix of the full unfolding $\mathcal{U}$ having no event in E_{Ignored}. Since every discarded event automatically discards all its causal successors, the set of events remaining in the goal-driven unfolding $\mathcal{U}_{\text{gd}}$ is

$$E_{\text{gd}} \stackrel{\text{def}}{=} \{e \in E \mid \lceil e \rceil \cap E_{\mathit{Ignored}} = \emptyset\}\ .$$

Notice that the events and conditions of the goal-driven unfolding as defined above can be constructed inductively following the procedure described in Section 2, enriched so that it attaches the set $\mathit{Useless}(e)$ to every new event e.

Theorem 1. *(proof in Appendix A) The goal-driven unfolding preserves all minimal configurations from M_0 to the goal.*

A direct corollary is that the goal is reachable in $\mathcal{N}$ iff the goal-driven unfolding contains a configuration which reaches it.

Notice that the precise definition of minimal sequences/configurations is crucial here, and especially the fact that the reduction procedure preserves *all* minimal sequences/configurations. Indeed, imagine a situation where the minimal firing sequences to the goal fire two concurrent transitions t_1 and t_2 and then one out of two possible transitions t_3 and t_4. A reduction procedure which would guarantee only the preservation of *some* minimal firing sequence to the goal could declare t_3 useless when called after the event corresponding to t_1, and declare t_4 useless when called after t_2, thus preventing to reach the goal.

4.2 Goal-Driven Prefix

We now define a finite goal-driven prefix. Our Algorithm 1 relies on the theory of adequate orders [11] developed for unfoldings. Any adequate order on the configurations of the full unfolding can be used, but, since our goal-driven unfolding prunes some branches of the unfolding, we have to adapt the construction.

A prefix $\mathcal{P}$ has the same structure as an unfolding, with an additional field *coff* for the set of cut-off events. As usual, the procedure Putative-GD-Prefix extends iteratively the prefix $\mathcal{P} = \langle C, E, G, C_0, \mathit{coff}\rangle$. An extension is an event $e = \langle C', t\rangle$ with $C' \subseteq C$ s.t. $\forall c, c' \in C',\ c \ \mathit{co}\ c',\ \{h(c) \mid c \in C'\} = {}^\bullet t$, and $\forall \langle e', p\rangle \in C',\ e' \notin \mathit{coff}$. Here the procedure maintains a map Δ of transitions that can be ignored, and considers an extension $e = \langle C', t\rangle$ only if the transition t is not declared useless, i.e., t is absent from $\Delta(c')$ for all pre-condition $c' \in C'$.

The difficult part is that, when an event e is declared cut-off because $\mathit{Mark}(\lceil e\rceil) = \mathit{Mark}(\lceil e'\rceil)$ for an event $e' \lhd e$, nothing guarantees that the transitions allowed after e are also allowed after e'. Then, e and e' have the same future in the full unfolding, but not necessarily in the goal-driven unfolding.

Fig. 2 illustrates this situation. Let the goal be $g = \{p_4, p_3\}$. It can be reached by the firing sequences $a(bb')^*c(bb')^*b$ or $a'b'(bb')^*c(bb')^*b$. Only those who do not take the cycle bb' are minimal, namely acb and $a'b'cb$. Notice that all the transitions participate in at least one minimal firing sequence, so the model $\mathcal{N}$ cannot be reduced from the initial marking (every reduction procedure will output $\texttt{useless-trs}(\mathcal{N}, g) = \emptyset$). On the other hand, if transition a is fired, we reach marking $\{p_1, p_2\}$ from which b', a and a' become useless.

Now, observe the branching process on the right of Fig. 2 (it is a prefix of the unfolding $\mathcal{U}$ of $\mathcal{N}$). Notice that the causal past $\lceil e\rceil$ of the event labeled a' and the causal past of the event e' labeled b reach the same marking

Algorithm 1 Algorithm for goal-driven prefix computation.

procedure PUTATIVE-GD-PREFIX($\mathcal{N}, \Delta$) with $\mathcal{N} = \langle P, T, F, M_0 \rangle$
  $\mathcal{P} \leftarrow \langle C \leftarrow \{\langle \bot, p \rangle \mid p \in M_0\}, E \leftarrow \emptyset, G \leftarrow \emptyset, C_0 \leftarrow \{\langle \bot, p \rangle \mid p \in M_0\}, \mathit{coff} \leftarrow \emptyset \rangle$
  repeat
    Let $e = \langle C', t \rangle$ be a $\lhd$-minimal extension of $\mathcal{P}$ s.t. $t \notin \bigcup_{c' \in C'} \Delta(c')$.
    $E \leftarrow E \cup \{e\}$
    $C \leftarrow C \cup \{\langle e, p \rangle \mid p \in t^\bullet\}$
    $G \leftarrow G \cup \{\langle c', e \rangle \mid c' \in C'\} \cup \{\langle e, \langle e, p \rangle \rangle \mid p \in t^\bullet\}$
    if $\exists e' \in E$ s.t. $\mathit{Mark}(\lceil e \rceil) = \mathit{Mark}(\lceil e' \rceil)$ **then**
      $\mathit{coff} \leftarrow \mathit{coff} \cup \{e\}$ */e is a cut-off event/*
    end if
    for all $c \in \{\langle e, p \rangle \mid p \in t^\bullet\}$ s.t. $c \notin \Delta$ **do** */extend Δ with new cond./*
      $\Delta(c) \leftarrow \mathit{Useless}(c, \Delta, \mathcal{P})$
    end for
  until no extension exists
end procedure

procedure POST-Δ($\Delta, \mathcal{P}$) with $\mathcal{P} = \langle C, E, G, C_0, \mathit{coff} \rangle$
  $\Delta' \leftarrow \Delta$ */copy map Δ/*
  for $e \in E$ following $\lhd$ order **do**
    for all $c \in e^\bullet$ **do**
      $\Delta'(c) \leftarrow \Delta'(c) \cap \mathit{Useless}(c, \Delta, \mathcal{P})$
    end for
    if $\exists e' \in E \setminus \mathit{coff}$ s.t. $\mathit{Mark}(\lceil e \rceil) = \mathit{Mark}(\lceil e' \rceil)$ **then**
      for all $c' \in \mathit{Cut}(\lceil e' \rceil)$ with $c \in \mathit{Cut}(\lceil e \rceil)$ and $h(c) = h(c')$ **do**
        $\Delta'(c') \leftarrow \Delta'(c') \cap \Delta'(c)$
      end for
    end if
  end for
end procedure

procedure GD-PREFIX($\mathcal{N}$) with $\mathcal{N} = \langle P, T, F, M_0 \rangle$
  $\Delta' \leftarrow \{\langle \bot, p \rangle \mapsto \emptyset \mid p \in M_0\}$
  repeat
    $\Delta \leftarrow \Delta'$ */copy map Δ'/*
    $\mathcal{P} \leftarrow$ PUTATIVE-GD-PREFIX($\mathcal{N}, \Delta$) */can add new entries in Δ/*
    $\Delta' \leftarrow$POST-Δ($\Delta, \mathcal{P}$)
  until $\Delta' = \Delta$
end procedure

$\mathit{Mark}(\lceil e \rceil) = \mathit{Mark}(\lceil e' \rceil) = \{p_1, p_3\}$. Moreover, an adequate order on the configurations of $\mathcal{U}$ may order them as $\lceil e' \rceil \lhd \lceil e \rceil$. Consequently, e is a cut-off and the minimal configuration $\mathcal{K}(a'b'cb)$ is not represented in the finite prefix. Following the idea of the proof of completeness of finite prefixes based on adequate orders, we can indeed shift the extension $b'cb$ of $\lceil e \rceil$ (in gray on the right of Fig. 2) to the isomorphic extension of $\lceil e' \rceil$ (also in gray on the figure). We get the configuration $\mathcal{K}(abb'cb)$, which reaches the goal as well. But this configuration is *not*

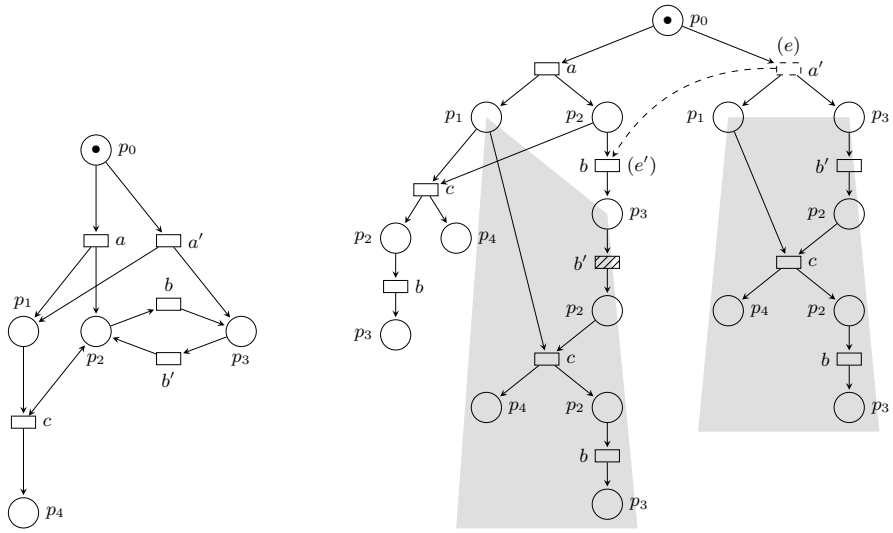

Fig. 2. A safe Petri net (left) and one of its branching processes (right). Configurations $\lceil e \rceil$ and $\lceil e' \rceil$ lead to the same marking $\{p_1, p_3\}$ and have isomorphic extensions (in gray). The dashed arrow represents the fact that $\lceil e' \rceil \lhd \lceil e \rceil$. Consequently e is a cut-off.

minimal any more because it executes the cycle bb': the marking reached after a is the same as the marking reached after abb'. Actually, the model reduction procedure called from the event labeled a may very well have declared b' useless. Consequently, $\mathcal{K}(abb'cb)$ would not be represented in the prefix. We correct this by allowing after $\lceil e' \rceil$ all the transitions that were allowed after $\lceil e \rceil$.

The difficulty in the definition and in the computation of a finite prefix $\mathcal{P}_{\text{gd}}$ of $\mathcal{U}$ which preserves the markings reachable in $\mathcal{U}_{\text{gd}}$ is to allow in the future of an event e' all the transitions that are useful for at least one of all the configurations which are shifted to $\lceil e' \rceil$ by the mechanics described above. The first answer to this problem is to allow after $\lceil e' \rceil$ all the transitions that were allowed after $\lceil e \rceil$. This solves the problem of an event consuming only post-conditions of e', like the occurrence of b' after e in our example of Fig. 2: its corresponding event after e' is now allowed. However, this is not sufficient in general: an event f consuming a post-condition of e may also consume other conditions which are created by events concurrent to $\lceil e \rceil$. Such event f has a corresponding f' in the future of e', consuming conditions which are available after firing a configuration of the form $\lceil e' \rceil \cup \mathcal{C}'$ for some $\mathcal{C}'$ concurrent to $\lceil e' \rceil$. We need transition $t = h(e) = h(e')$ to be allowed after all the conditions consumed by f'. In the case of a condition $c' \in {}^\bullet f' \setminus \mathit{Cut}(\lceil e' \rceil)$, our procedure ensures this as follows: if it calls the model reduction procedure after the event ${}^\bullet c'$, it also calls it on the marking $\mathit{Mark}(\lceil e' \rceil \cup \lceil {}^\bullet c' \rceil)$ which equals $\mathit{Mark}(\lceil e \rceil \cup \lceil {}^\bullet c \rceil)$. Hence, if t is needed after $\lceil e \rceil \cup \lceil {}^\bullet c \rceil$, it will also be allowed after c'. In the end, when applying the

reduction procedure after a configuration $\mathcal{C}$, we also take into account a set $Alt(\mathcal{C})$ of alternating configurations defined inductively as:

- $\mathcal{C} \in Alt(\mathcal{C})$
- $\forall \mathcal{C}' \in Alt(\mathcal{C})$, $\forall e, e' \in E$ such that $\lceil e \rceil \rhd \lceil e' \rceil$ and $Mark(\lceil e \rceil) = Mark(\lceil e' \rceil)$, if $Cut(\lceil e' \rceil) \cap {\mathcal{C}'}^{\bullet} \neq \emptyset$ and $\lceil e' \rceil \cup \mathcal{C}'$ is conflict free, then $\lceil e' \rceil \cup \mathcal{C}' \in Alt(\mathcal{C})$.

However, in practice, during the computation of the goal-driven prefix, $Alt(\mathcal{C})$ will be computed on the events and configurations derived so far, hence ignoring events later added in the prefix. Also, as explained above, when an event e is stated cut-off because of a $\lhd$-smaller event e', we allow after e' all the transitions allowed after e; but this implies reconsidering some new extensions of e'.

For these reasons, the procedure GD-Prefix($\mathcal{N}$) presented in Algorithm 1 iterates the computation of a putative prefix, progressively refining an over-approximation of transitions to ignore (map Δ), by identifying *a posteriori* the transitions that should *not* have been ignored.

At each iteration, the procedure Putative-GD-Prefix($\mathcal{N}, \Delta$) computes a putative prefix, relying on the previous value of the map Δ of transitions that can be ignored. Essentially, the prefix $\mathcal{P}$ obtained at the first iteration is the naive prefix of $\mathcal{U}_{\mathrm{gd}}$ (prefix without the gray parts on the example of Fig. 2).

Once a putative prefix has been computed, we verify *a posteriori* if its related map Δ is correct. This is done by re-computing Δ using the procedure Post-$\Delta(\Delta, \mathcal{P})$, this time taking into account all the events in $\mathcal{P}$ (line 34). By construction, the resulting Δ' can only allow more transitions than Δ. If Δ' differs from Δ, a new putative prefix is computed according to the corrected Δ'.

The procedure Post-$\Delta(\Delta, \mathcal{P})$ takes the $Alt(\lceil e \rceil)$ into account by the way of a modified version of *Useless*(), now defined on conditions rather than on events. Given a condition $c \in e^{\bullet}$ in a prefix $\mathcal{P}$,

$$Useless(c, \Delta, \mathcal{P}) \stackrel{\text{def}}{=} \begin{cases} \bigcup_{c' \in {}^{\bullet}e} \Delta(c') & \text{if } e \notin E' \\ \bigcap_{C^* \in Alt(\lceil e \rceil)} \texttt{useless-trs}\left(\mathcal{N}, g, Mark(C^*), \bigcup_{c' \in {}^{\bullet}e} \Delta(c')\right) & \text{if } e \in E', \end{cases}$$

where E' is the set of events triggering an explicit reduction (Section 4.1).

This iterative construction necessarily terminates (Lemma 3, proof in Appendix B) and converges to a unique finite prefix $\mathcal{P}_{\mathrm{gd}}$. Regarding complexity, putting aside the call to model reduction, whereas all the structures are finite, $Alt(\mathcal{C})$ can have an exponential numbers of configurations due to multiple combinations of configurations sharing an intersection.

Lemma 3. *The procedure* GD-Prefix*($\mathcal{N}$) terminates.*

Notice that $\mathcal{P}_{\mathrm{gd}}$ may contain events that are not in E_{gd}. Hence, goal-driven prefix is a prefix of $\mathcal{U}$, but not necessarily a prefix of $\mathcal{U}_{\mathrm{gd}}$. This is the case of the event labeled b' after e', as we discussed above for the example in Fig. 2.

Theorem 2 (proof in Appendix C) states completeness of $\mathcal{P}_{\mathrm{gd}}$ w.r.t. minimal configurations. Thus, the goal-driven prefix preserves the reachability of the goal.

One can finally remark that, by construction, $\mathcal{P}_{\mathrm{gd}}$ contains at most one non-cutoff event per reachable marking, assuming the adequate order $\lhd$ is total.

Theorem 2. *For every configuration $\mathcal{C}$ of $\mathcal{U}_{\mathrm{gd}}$ and for every single-event extension $\{f\}$ of $\mathcal{C}$ such that $\mathcal{C} \cup \{f\}$ is a prefix of a minimal configuration to the goal, there exists a configuration $\mathcal{C}'$ in the goal-driven prefix and a single-event extension $\{f'\}$ of $\mathcal{C}'$ with $Mark(\mathcal{C}) = Mark(\mathcal{C}')$ and $h(f) = h(f')$.*

Example. Let us consider the Petri net of Fig. 2(left) with the goal $\{p_4, p_3\}$.

The goal-driven unfolding can lead to the branching process of Fig. 2(right) where the dashed transition b' has been removed. Indeed, after transition a, transition b' is declared useless as it is not part of any minimal configuration extending $\mathcal{K}(a)$. Therefore 3 maximal configurations are remaining in the goal-driven unfolding: the two minimal configurations $\mathcal{K}(acb)$ and $\mathcal{K}(a'b'cb)$, and the configuration $\mathcal{K}(ab)$ which does not reach the goal.

The goal-driven prefix can lead to the branching process of Fig. 2(right) where the event e is cut-off (because of e'), and therefore its future events are ignored, and where one of the two remaining events firing transition c is declared cut-off (because of the other one). Although the b' transition can be declared useless after $\mathcal{K}(a)$ (and hence $\mathcal{K}(ab)$), the cut-off of e will remove b' from the set of ignored transitions of the conditions matching with p_1 and p_3 on the cut of $\mathcal{K}(ab)$. Therefore, the events and conditions in the left gray area will be added to the prefix, from which all the minimal configurations can be identified.

5 Experiments

In this section, we compare the size of the complete prefix with the goal-driven prefix on different Petri net models of biological signalling and gene regulatory networks. In general, such networks gather dozens to thousands nodes having sparse interactions (each node is directly influenced by a few other nodes), which call for concurrency-aware approaches to cope with the state space explosion. We took the networks from systems biology literature, specified as Boolean or automata networks: each node is modelled by an automaton, where states model its activity level, most often being binary (active or inactive). The Petri nets are encodings of these automata networks which ensure bisimilarity [5].

Implementation. In practice, instead of computing putative prefixes from scratch as it is described in Algorithm 1, our implementation for the goal-driven prefix[4] iteratively corrects the putative prefix by propagating transitions missed in the previous iteration. At this stage, it does not use any particular optimization [1], our primary objective being to compare the size of the resulting prefixes. In order to obtain a proper comparison [15], our implementation uses the same arbitrarily-fixed ordering for the complete and goal-driven prefixes extensions.

[4] Code and models available at `http://loicpauleve.name/godunf.tbz2`

The computation of `useless-trs` $(\mathcal{N}, g, M, I)$ relies on the goal-oriented reduction of asynchronous automata networks introduced in [17]. This method is based on a static analysis of causal dependencies of transitions and an abstract interpretation of traces which allow to collect all the transitions involved in the minimal configurations to the goal: non-collected transitions can then be ignored. The complexity of the reduction is polynomial with the number of automata and transitions, and exponential with the number of states in individual automata (i.e., number of qualitative states of nodes). As shown in [17], the method can lead to drastic model reductions and can be executed in a few hundredths of a second on networks with several hundreds of nodes.

We applied the goal-driven unfolding to 1-safe Petri net encodings of the automata networks, where there is one place for each local state of each automaton, and a one-to-one relationship between transitions. The places corresponding to states of a same automaton are mutually exclusive by construction. Future work may consider goal-driven unfolding of products of transition systems [10].

The goal-driven prefix we define in this paper supports calling the model reduction procedure at discretion: even if it has a low computational cost, performing the model reduction after each event may turn out to be very time consuming. Our prototype implements simple strategies to decide when the call to the model reduction should be performed: after each event; only for the first n events; and only for events up to a given level in the unfolding.

Benchmarks. Given a Petri net with an initial marking M_0 and a goal g, we first compute the goal-oriented model reduction from initial marking (`useless-trs` $(\mathcal{N}, g, M_0, \emptyset)$). The resulting net is then given as input to the unfolding, either with the complete finite prefix computation, or with the goal-driven. Therefore, the difference in the size of the prefixes obtained is due only to transition exclusions after at least one event.

Table 1 summarizes the benchmarks between complete and goal-driven prefix on different models of biological networks. The size of a prefix is the number of its non-cutoff events. "RB/E2F" is a model of the cell cycle [4]; "T-LGL" is a model of survival signaling in large granular lymphocyte leukemia [23]; and "VPC" is a model for the specification of vulval precursor cells and cell fusion control in Caenorhabditis elegans [22]. All those models have very different network topology and dynamical features. For each model, the initial marking and goal correspond to biological states of interest (checkpoints or differentiated states).

On these models, the goal-driven prefix shows a significant size reduction, while containing all the minimal configurations. The number of reductions can be larger than the size of the prefix as it accounts for the intermediate putative prefixes (as explained in Section 4.2). For the "VPC" model, we applied several strategies for deciding when the model reduction should be called. In this case, the systematic model reduction led to some re-ordering of the extensions and cut-offs declaration, which required numerous additional calls to the model reduction procedure. This motivates the design of heuristics to estimate when a model reduction should be performed. For the "T-LGL" model, it was impossible to compute the complete finite prefix, whereas the goal-driven cuts

Model	Prefix	Strategy	Prefix size	Time	Nb reductions
RB/E2F	complete	N/A	15,210	24s	N/A
$\|P\| = 80 \quad \|T\| = 54$	goal-driven	always	112	0.5s	136
T-LGL	complete	N/A	>1,900,000*	OT*	N/A
$\|P\| = 98 \quad \|T\| = 159$	goal-driven	always	17	0.3s	17
VPC	complete	N/A	44,500	176s	N/A
$\|P\| = 135 \quad \|T\| = 216$	goal-driven	always	1,827	2h	16,009
		first $1,000$	2,036	60s	1,000
		level ≤ 2	2,400	7s	38

Table 1. Benchmarks of the goal-driven w.r.t. complete prefix of 1-safe Petri nets. For each model, the number of places $|P|$ and transitions $|T|$ is given. The strategy decides when the model reduction should be performed; the number of calls to the reduction procedure is indicated in the column "Nb reductions". Computation times were obtained on an Intel® Core™ i7 3.4GHz CPU with 16GB RAM. N/A: Non Applicable; *: out-of-memory computation (with mole [20], with the same ordering for extensions as our implementation), the indicated prefix size is only a lower bound.

most of the configurations and produces a very concise prefix. This behaviour can be explained by large transient cycles prior to the goal reachability, which are avoided by the use of model reduction during the prefix computation.

6 Conclusion

We introduced the goal-driven unfolding of safe Petri nets for identifying efficiently *all* the minimal configurations that lead to a given goal. The goal can be a marking of the net, or any partially specified marking, and notably a single marked place. The goal-driven unfolding relies on an external reduction method which identifies transitions that are not part of minimal configuration for the goal reachability. Such useless transitions are then skipped by the unfolding. The computation of a goal-driven prefix requires a particular treatment of cut-offs to ensure that all the markings reachable in the goal-driven unfolding are preserved.

We applied our approach to different models of biological systems which show a significant reduction of the prefix when driven by the goal. In our framework, the reduction procedure can be applied at discretion, and many possible heuristics could be embedded to decide when the reduction is timely, which impacts both the execution time and the size of the prefix. The resulting goal-driven prefix contains fewer events prefix than reachable markings, due to the total adequate order, as well as for classical finite complete prefix.

Future work will explore the combination with the semi-adequate ordering of configurations of directed unfolding [3] as it may reduce the need for propagating transitions allowed by a cut-off event. Although our approach considers the model reduction procedure as a blackbox, on-going work is currently generalizing the one used in the experimentations to any safe Petri net. Finally, we are considering implementing the goal-driven unfolding within Mole [20].

References

1. P. Baldan, A. Bruni, A. Corradini, B. König, C. Rodríguez, and S. Schwoon. Efficient unfolding of contextual Petri nets. *TCS*, 449:2–22, 2012.
2. P. Baldan, A. Corradini, and B. König. A static analysis technique for graph transformation systems. In *CONCUR 2001*, volume 2154 of *LNCS*, pages 381–395. Springer, 2001.
3. B. Bonet, P. Haslum, S. L. Hickmott, and S. Thiébaux. Directed unfolding of Petri nets. *Trans. Petri Nets and Other Models of Concurrency*, 1:172–198, 2008.
4. L. Calzone, A. Gelay, A. Zinovyev, F. Radvanyi, and E. Barillot. A comprehensive modular map of molecular interactions in RB/E2F pathway. *Molecular Systems Biology*, 4(1), 2008.
5. T. Chatain, S. Haar, L. Jezequel, L. Paulevé, and S. Schwoon. Characterization of reachable attractors using Petri net unfoldings. In *Computational Methods in Systems Biology*, volume 8859 of *LNCS*, pages 129–142. Springer, 2014.
6. T. Chatain and V. Khomenko. On the well-foundedness of adequate orders used for construction of complete unfolding prefixes. *Inf. Process. Lett.*, 104(4):129–136, 2007.
7. A. Cheng, J. Esparza, and J. Palsberg. Complexity results for 1-safe nets. *Theor. Comput. Sci.*, 147(1&2):117–136, 1995.
8. J. Esparza and K. Heljanko. *Unfoldings – A Partial-Order Approach to Model Checking*. Springer, 2008.
9. J. Esparza, P. Kanade, and S. Schwoon. A negative result on depth-first net unfoldings. *STTT*, 10(2):161–166, 2008.
10. J. Esparza and S. Römer. An unfolding algorithm for synchronous products of transition systems. In *CONCUR*, volume 1664 of *LNCS*, pages 2–20. Springer, 1999.
11. J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. *FMSD*, 20:285–310, 2002.
12. J. Esparza and C. Schröter. Unfolding based algorithms for the reachability problem. *Fundam. Inform.*, 47(3-4):231–245, 2001.
13. C. Flanagan and P. Godefroid. Dynamic partial-order reduction for model checking software. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2005*, pages 110–121. ACM, 2005.
14. V. Khomenko. Punf. `http://homepages.cs.ncl.ac.uk/victor.khomenko/tools/punf/`.
15. V. Khomenko, M. Koutny, and W. Vogler. Canonical prefixes of Petri net unfoldings. *Acta Inf.*, 40(2):95–118, 2003.
16. K. L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *CAV*, pages 164–177, 1992.
17. L. Paulevé. Goal-Oriented Reduction of Automata Networks. In *CMSB 2016 - 14th conference on Computational Methods for Systems Biology*, volume 9859 of *Lecture Notes in Bioinformatics*. Springer, 2016.
18. L. Paulevé, G. Andrieux, and H. Koeppl. Under-approximating cut sets for reachability in large scale automata networks. In *CAV*, volume 8044 of *LNCS*, pages 69–84. Springer, 2013.
19. R. Samaga, A. Von Kamp, and S. Klamt. Computing combinatorial intervention strategies and failure modes in signaling networks. *J. of Computational Biology*, 17(1):39–53, 2010.
20. S. Schwoon. Mole. `http://www.lsv.ens-cachan.fr/~schwoon/tools/mole/`.

21. C. Wang, Z. Yang, V. Kahlon, and A. Gupta. Peephole partial order reduction. In *TACAS 2008*, volume 4963 of *LNCS*, pages 382–396. Springer, 2008.
22. N. Weinstein and L. Mendoza. A network model for the specification of vulval precursor cells and cell fusion control in caenorhabditis elegans. *Frontiers in Genetics*, 4(112), 2013.
23. R. Zhang, M. V. Shah, J. Yang, S. B. Nyland, X. Liu, J. K. Yun, R. Albert, and T. P. Loughran. Network model of survival signaling in large granular lymphocyte leukemia. *PNAS*, 105:16308–13, 2008.

A Proof of Theorem 1

Let E be a minimal configuration of $\mathcal{N}$ to the goal. We want to prove that $E \subseteq E_{\mathrm{gd}}$. Given the definition of E_{gd} and given that E is causally closed, this is equivalent to proving that $E \cap E_{Ignored} = \emptyset\}$: trivially, if some $e \in E$ is also in $E_{Ignored}$, then $\lceil e \rceil \cap E_{Ignored}$ contains at least e, so it is nonempty, and by definition $e \notin E_{\mathrm{gd}}$; conversely, if some $e \in E$ is not in E_{gd}, this is because $\lceil e \rceil \cap E_{Ignored} \neq \emptyset$, and since E is causally closed, $\lceil e \rceil \subseteq E$, which implies that $E \cap E_{Ignored} \neq \emptyset$.

Then, it remains to show that $E \cap E_{Ignored} = \emptyset$. Let $e = \langle C, t \rangle \in E$; we have to show that, for every $e' \in \lfloor e \rfloor$, $t \notin \mathit{Useless}(e')$. If $e' \notin E'$, then, by definition, $\mathit{Useless}(e') \stackrel{\text{def}}{=} \bigcup_{e'' \in \lfloor e' \rfloor} \mathit{Useless}(e'')$, which means that e' has a causal predecessor e'' which also satisfies $t \notin \mathit{Useless}(e'')$.

Select now an e' which is minimal w.r.t. causality. This eliminates the previous case, so we have $e' \in E'$ and $t \in \bigcup_{e'' \in \lfloor e' \rfloor} \mathit{Useless}(e'')$ and $t \notin \texttt{useless-trs}\Big(\mathcal{N}, g, \mathit{Mark}(\lceil e' \rceil), \bigcup_{e'' \in \lfloor e' \rfloor} \mathit{Useless}(e'')\Big)$. Assuming that `useless-trs` is a reduction procedure satisfying Definition 8, this implies that no minimal firing sequence from $\mathit{Mark}(\lceil e' \rceil)$ to the goal uses t, which contradicts the fact that E is a minimal configuration to the goal: Indeed, since $e' \in \lfloor e \rfloor$, there exists a linearization[5] $e_1, \ldots, e_{|E|}$ of E in which the events in $\lceil e' \rceil$ occur before the others, i.e. $\{e_1, \ldots, e_{|\lceil e' \rceil|-1}\} = \lfloor e' \rfloor$, $e_{|\lceil e' \rceil|} = e'$ and $\{e_{|\lceil e' \rceil|+1}, \ldots, e_{|E|}\} = E \setminus \lceil e' \rceil$; then $t_{|\lceil e' \rceil|+1} \ldots t_{|E|}$ (with $t_i \stackrel{\text{def}}{=} h(e_i)$ the transition corresponding to e_i) is a firing sequence from $\mathit{Mark}(\lceil e \rceil')$ to the goal and it uses t. If it is not minimal, then because it has a feasible cycling permutation σ, then $t_1, \ldots, t_{|\lceil e' \rceil|} \cdot \sigma$ is a feasible cycling permutation from M_0 to the goal, which contradicts the fact that E is a minimal configuration to the goal. □

B Proof of Lemma 3

Because the set of markings is finite and because $\mathit{Alt}(\mathcal{C})$ is also finite (computed on a finite prefix), procedure Putative-GD-Prefix($\mathcal{N}, \Delta$) always terminates; moreover all the iterations in procedure Post-$\Delta(\Delta, \mathcal{P})$ are over finite sets. Finally, we prove that procedure GD-Prefix($\mathcal{N}$) terminates, i.e., after a finite number of iterations, Post-$\Delta(\Delta, \mathcal{P}) = \Delta$. First, by construction, $\forall c \in \Delta$, $c \in \Delta'$ and $\Delta'(c) \subseteq \Delta(c)$, with $\Delta' =$ Post-$\Delta(\Delta, \mathcal{P})$. Then, remark that, due to the cut-off treatment, any event of any putative prefix has a bounded number of event ancestors (causal past): the number of reachable markings. Finally, because the branching up to a given depth is finite, only a finite number of events can be considered in any iteration of the putative prefix; hence the number of events registered in Δ is finite. Therefore, due to the monotonicity of Δ modifications, the iterative procedure necessarily converges towards a unique finite prefix in a finite number of steps. □

[5] A linearization of E is a total ordering of $e_1, \ldots, e_{|E|}$ of the events in E such that for $e_i < e_j \implies i < j$.

C Proof of Theorem 2

We first show that for every configuration $\mathcal{C}$ that can be extended to a minimal configuration to the goal, there exists a configuration in the goal-driven prefix which contains no cut-off event and reaches $Mark(\mathcal{C})$. The principle is the one used for completeness of classical finite prefixes defined using adequate orders: if $\mathcal{C}$ contains no cut-off event, it is in the goal-driven prefix, since the construction of the *Useless* is more permissive in the goal-driven prefix (with the use of $Alt()$) than in the goal-driven unfolding. Now, if $\mathcal{C}$ contains a cut-off event e (w.r.t. an event e' such that $\lceil e' \rceil \lhd \lceil e \rceil$ and $Mark(\lceil e' \rceil) = Mark(\lceil e \rceil)$), then $\mathcal{C}$ can be decomposed as $\lceil e \rceil \uplus D$ and e' has an extension D' isomorphic to D. Then $\lceil e' \rceil \uplus D'$ is smaller than $\mathcal{C}$ w.r.t. $\lhd$ and reaches the same marking. This operation can be iterated if needed; it terminates because $\lhd$ is well founded, and gives a configuration $\mathcal{C}'$ without cut-offs which reaches the same marking as $\mathcal{C}$. If $\mathcal{C}$ can be extended with an event f, then so can $\mathcal{C}'$ with an event f' corresponding to the same transition $h(f') = h(f)$. The event f' is in the prefix but may be a cut-off.

It remains to make sure that the transitions of $\mathcal{C}$ (plus $h(f)$) are not considered useless. For this, focus on $\lceil e \rceil \uplus D$ mapped to $\lceil e' \rceil \uplus D'$. For every event $d \in D$, let d' be the corresponding event in D'. We have $\lceil d \rceil \cup \lceil e \rceil \in Alt(\lceil d' \rceil)$ because the causal past of d' uses at least one condition from the cut of $\lceil e' \rceil$. This ensures that the transitions fired in D after $\lceil d \rceil \cup \lceil e \rceil$ are taken into account in the computation of the transitions allowed after d' (if d' is itself in the prefix, otherwise apply this inductively), ensuring that in the end $\mathcal{C}' \cup \{f'\}$ is in the goal-driven prefix. $\square$