# Solving a Safety Game on the Unfolding of Safe Petri Nets

Federica Adobbati[1], Luca Bernardinello[1] and Lucia Pomello[1]

[1]*Dipartimento di informatica, sistemistica e comunicazione*
*Università degli Studi di Milano- Bicocca*
*viale Sarca 336 U14, Milano, Italia*

**Abstract**

We consider a two-player game on 1-safe Petri nets, in which each player controls a subset of transitions. The players are called 'user' and 'environment'; we assume that the user must guarantee progress on its transitions, and has a safety goal on the system. A play of this game is a run in the unfolding of the net; this is a partial order structure describing all the possible executions of the net. In general, we define a strategy for the user as a map from equivalence classes of markings to subsets of transitions owned by the user. We propose an algorithm to check whether the user has a winning strategy on a finite prefix of the unfolding.

**Keywords**

Petri nets, Asynchronous games, safety goal, prefix of the unfolding

## 1. Introduction

Several real world situations can be abstractly described as follows: an agent, which we will call *user*, interacts with an environment trying to achieve a goal. The goal can consist, for example, in reaching a given state, or avoiding a set of states, even if the environment is hostile, or non cooperating. This abstract model can then be seen as a game, and one can investigate whether the user has a winning strategy assuring him to reach the goal for any possible behaviour of the environment.

In this paper, we use 1-safe Petri nets as a formal model, on which one can define such a game. A Petri net represents a system in which global states are explicitly distributed in sets of "local states" (*places*), and the occurrence (or *firing*) of a transition can change a subset of local states. This allows for a clear representation of concurrency and conflicts between events. Global states are usually called *markings*. In 1-safe nets, a marking is a set of places, and transitions are characterized by their pre- and post-conditions. Places belonging to a marking are said to be marked in that marking.

Different semantics can be defined for Petri nets. An interleaving semantics associates, to a given Petri net, a labelled transition system, in which states correspond to reachable markings of the net, and arcs are labelled by transitions. A 'true concurrency' semantics can be defined, based on the idea of "unfolding" a net, by recording occurrences of local states and of transitions; in this way, the set of all possible behaviours of the net is described by another net, possibly

*PNSE'23, International Workshop on Petri Nets and Software Engineering, 2023*

infinite. The unfolding of a finite 1-safe net, although infinite, is built on a finite set of repeated substructures, made of a transition and its pre- and post-conditions. This allows for constructing a finite prefix of the whole unfolding, containing enough information to reconstruct, by glueing pieces, the full unfolding; such a prefix is said to be complete.

The basic definitions on Petri nets and unfoldings are recalled in Sect. 2.

Using 1-safe Petri nets, we study a game in which the user aims at avoiding markings in which any place in a given subset $S$ is marked. We assume that the transitions of the net are partitioned into controllable transitions (under the control of the user) and uncontrollable transitions. We further assume that the user has a "progress" obligation: if a controllable transition is enabled, then it must eventually happen, or be disabled.

A play in the game is an execution of the net, in which user and environment can move concurrently. Formally, a play is a *run* in the unfolding of the net, namely the recording of occurrences of places and transitions, where all conflicts have been solved.

We will assume that the user knows the structure of the system, and has full knowledge of the current marking, but cannot prevent any uncontrollable transition. A strategy for the user is then defined as a map from markings to controllable transitions, and guides the user in choosing which controllable transitions to fire. The game is formally defined in Sect. 3.

A strategy is winning if any play in which the user complies with it satisfies the user's aim: in all reached markings in the play, no place in $S$ is marked.

The main problem we address is to decide whether the user has a winning strategy. To that aim, we work on a complete prefix of the unfolding of the net, and give an algorithm which solves the problem by discovering bad choices that would lead the user to lose a play. Strictly speaking, the algorithm solves the decision problem; however, by inspecting the set of bad choices found, one can construct a strategy.

The algorithm is defined and discussed in Sect. 4. Sec. 5 concludes the paper discussing some related and future works.

## 2. Petri nets

In this section we recall basic definitions concerning Petri nets, net unfoldings and their prefixes, that will be useful in the rest of the paper, see also [1, 2, 3]. Among the several classes of nets defined and studied in the literature, in this paper we use the class of *1-safe* Petri nets.

A *net* is a triple $N = (P, T, F)$, where $P$ and $T$ are disjoint sets, the elements of $P$ are called *places* and are represented by circles, the elements of $T$ are called *transitions* and are represented by squares, $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*, represented by directed arcs. The *pre-set* of an element $x \in P \cup T$ is the set ${}^\bullet x = \{y \in P \cup T : (y, x) \in F\}$; the *post-set* of $x$ is the set $x^\bullet = \{y \in P \cup T : (x, y) \in F\}$. Let $X \subseteq P \cup T$ be a subset of elements, its pre-set is defined as ${}^\bullet X = \{y \in P \cup T : \exists x \in X : (y, x) \in F\}$, and its post-set as $X^\bullet = \{y \in P \cup T : \exists x \in X : (x, y) \in F\}$. We assume that each transition has non-empty pre-set and post-set.

Two transitions, $t_1$ and $t_2$, are *independent* if $({}^\bullet t_1 \cup t_1^\bullet)$ and $({}^\bullet t_2 \cup t_2^\bullet)$ are disjoint. They are structurally in *conflict* if ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$.

A net is *finite* if $P \cup T$ is finite, and *infinite* otherwise.

A net $(P, T, F)$ is a *subnet* of another net $(P', T', F')$ if $P \subseteq P'$, $T \subseteq T'$ and $F$ is the restriction of $F'$ to $(P \times T) \cup (T \times P)$.

A net $(P, T, F)$ is the *subnet* of $(P', T', F')$ *generated by* $T$ if $P = {}^\bullet T \cup T^\bullet$, $T \subseteq T'$ and $F$ is the restriction of $F'$ to $(P \times T) \cup (T \times P)$.

A *net system* is a quadruple $\Sigma = (P, T, F; m_{in})$ consisting of a finite net $N = (P, T, F)$ and an *initial marking* $m_{in} : P \to \mathbb{N}$, representing the initial state of the system. A *marking* is a map $m : P \to \mathbb{N}$.

A transition $t$ is *enabled* at a marking $m$, denoted $m[t\rangle$, if, for each $p \in {}^\bullet t$, $m(p) > 0$. A transition $t$, enabled at $m$, can *occur* (or *fire*) producing a new marking $m'$ (denoted $m[t\rangle m'$), where

$$m'(p) = \begin{cases} m(p) + 1 & \text{if } p \in t^\bullet \setminus {}^\bullet t \\ m(p) - 1 & \text{if } p \in {}^\bullet t \setminus t^\bullet \\ m(p) & \text{otherwise} \end{cases}$$

A marking $m'$ is reachable from another marking $m$, if there is a sequence of transitions $t_1 t_2 \dots t_n$ such that $m[t_1\rangle m_1[t_2\rangle \dots m_{n-1}[t_n\rangle m'$, this is also denoted with $m[t_1 t_2 \dots t_n\rangle m'$; $[m\rangle$ denotes the set of markings reachable from $m$. A marking $m$ is *reachable* if it is reachable from the initial marking $m_{in}$, i.e.: if $m \in [m_{in}\rangle$; $[m_{in}\rangle$ will be also denoted $M$ in the next sections.

A net system is *1-safe* if $m(p) \leq 1$, for each place $p$ and for each reachable marking $m$. Markings in 1-safe net systems can, and will, be considered as subsets of places.

In a net system, two transitions, $t_1$ and $t_2$, are *concurrent* at a marking $m$ if they are independent and both enabled at $m$. They are in *conflict at a marking* $m$ if they are both enabled at $m$, however they are not concurrently enabled at $m$: the occurrence of one of them disables the other one, this is possible if they are structurally in conflict, i.e.: they share at least a pre-place.

An example of a 1-safe net system is given on Fig. 1. Transitions structurally in conflict are for example $a$ and $n$, as well as $g$ and $h$, or $e$ and $f$; $a$ and $n$ are also in conflict at the initial marking $\{1\}$. Transitions $c$ and $f$ are independent, however they are never both enabled at a reachable marking, hence they are never concurrent; $f$ and $g$ are independent and both enabled at the reachable marking $\{3, 4\}$, they are concurrent at $\{3, 4\}$. We now introduce two technical relations that will be useful to define the partial order semantics of net systems. The $\prec$ relation on the elements of a net $N$ is the transitive closure of $F$ and $\preceq$ is the reflexive closure of $\prec$. Let $x, y \in P \cup T$; then $x \# y$ iff there exist $t_1, t_2 \in T : t_1 \neq t_2,\ t_1 \preceq x,\ t_2 \preceq y$ and there exists $p \in {}^\bullet t_1 \cap {}^\bullet t_2$.

The non sequential behaviour of net systems can be recorded by occurrence nets, which are used to represent by a single object the set of potential histories of a net system. A net $N = (B, E, F)$, possibly infinite, is an *occurrence net* if the following restrictions hold:

1. $\forall x \in B \cup E : \neg(x \prec x)$
2. $\forall x \in B \cup E : \neg(x \# x)$
3. $\forall e \in E : \{x \in B \cup E \mid x \preceq e\}$ is finite
4. $\forall b \in B : |{}^\bullet b| \leq 1$

In an occurrence net, the elements of $B$ are called *conditions* and the elements of $E$ are called *events*; the transitive and reflexive closure of $F$, $\preceq$, forms a *partial order*. The set of minimal

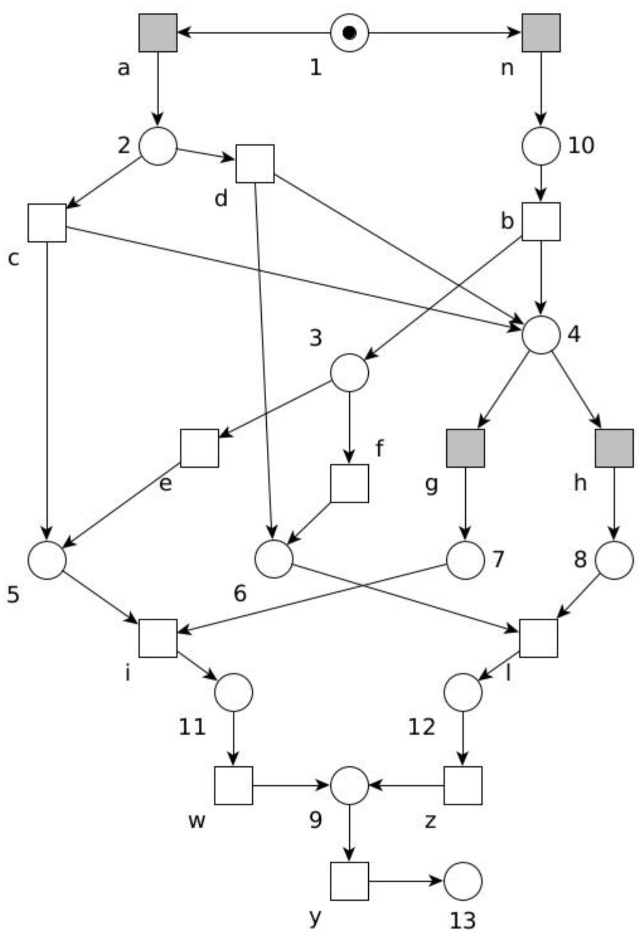

**Figure 1:** A 1-safe net system.

elements of an occurrence net $N$ with respect to $\preceq$ will be denoted by $\min(N)$. Since we only consider nets in which every transition has nonempty pre-set and post-set, the elements of $\min(N)$ are conditions.

An occurrence net represents the alternative histories of a system; therefore its underlying graph is acyclic (cond. 1), and paths branching from a condition, corresponding to a choice between alternative behaviours, never converge (cond. 2).

An example of occurrence net is given on Fig. 2. It represents possible histories of the net system given on Fig. 1, where the correspondence with the elements of the system is given by the labels, and $\min(N) = \{1_1\}$.

Given an occurrence net $N = (B, E, F)$, we define an event $e \in E$ as *terminal* if there is no event $e' \in E$ such that $e \prec e'$.

On the elements of an occurrence net the relation of concurrency, **co**, is defined as follows: let $x, y \in P \cup T$; then $x$ **co** $y$ if $\neg(x \prec y)$ and $\neg(y \prec x)$ and $\neg(x \# y)$.

A *B-cut* of $N$ is a maximal set of pairwise concurrent elements of $B$, and can be intuitively seen as a potential global state through which a process can go in a history of the system.

In Fig. 2 for example: $2_1 \# 3_1$, $2_1 \# b_1$, $h_2 \# e_1$, $4_1$ **co** $5_1$, $\{4_1, 5_1\}$ is a B-cut, $g_1$ **co** $5_1$.

By analogy with net systems, we will sometimes say that an event $e$ of an occurrence net is enabled at a B-cut $\gamma$, denoted $\gamma[e\rangle$, if ${}^\bullet e \subseteq \gamma$; for example $\{4_1, 5_1\}[g_1\rangle$.

A *configuration* of an occurrence net $N = (B, E, F)$ is a set of events $C \subseteq E$ which is causally closed (for every $e \in C$, $e' \preceq e \Rightarrow e' \in C$) and free of conflicts ($\forall e_1, e_2 \in C$, $\neg(e_1 \# e_2)$). $C$ is *maximal* if it is maximal with respect to set inclusion. Note that $C$ can be an infinite set. Intuitively, a configuration is a set of events 'firable' from $\min(N)$, the natural initial marking of $N$, i.e.: there is a firing sequence from $\min(N)$ in which each event of the configuration occurs exactly once.

The *local configuration* $[e]$ of an event $e$ of an occurrence net is the set of events $e'$ such that $e' \leq e$. Local configurations are finite; they will play an important role in the following sections.

In Fig. 2: $\{n_1, b_1, e_1, g_3\}$ is a configuration which is not a local one; whereas $[i_1] = \{a_1, c_1, g_1, i_1\}$ and $[i_2] = \{n_1, b_1, e_1, g_3, i_2\}$ are examples of local configurations, the last one is also a maximal configuration.

Finite configurations and B-cuts are tightly related: if $C \subseteq E$ is a finite configuration of an occurrence net $N$, then $\mathrm{cut}(C) = (\min(N) \cup C^\bullet) \setminus {}^\bullet C$ is a B-cut. Intuitively, $\mathrm{cut}(C)$ represents the 'marking' of $N$ reachable from $\min(N)$ after the firing of the events in $C$. In the following sections, given a local configuration $[e]$ and its cut, $\gamma = cut([e])$, $\circ e$ will denote the cut obtained from $\gamma$ by 'backward firing' the event $e$, i.e.: $\circ e = (\gamma \setminus e^\bullet) \cup {}^\bullet e$; intuitively, $\circ e$ represents the 'marking' of $N$ reachable from $\min(N)$ after the firing of the events from which the event $e$ causally depends.

In the example: $\mathrm{cut}(\{n_1, b_1, e_1, g_3\}) = \{5_2, 7_3\}$; $\mathrm{cut}([i_2]) = \{11_2\}$; $\circ i_2 = \{5_2, 7_3\}$; $[e_1] = \{n_1, b_1, e_1\}$, $\mathrm{cut}([e_1]) = \{5_2, 4_3\}$ and $\circ e_1 = \{3_1, 4_3\}$.

A *branching process* of a 1-safe net system $\Sigma = (P, T, F; m_{in})$ is an occurrence net $N = (B, E, F)$, together with a labelling function $\lambda : B \cup E \to P \cup T$, such that

- $\lambda(B) \subseteq P$ and $\lambda(E) \subseteq T$ ($\lambda$ preserves the nature of nodes)
- for all $e \in E$, the restriction of $\lambda$ to ${}^\bullet e$ is a bijection between ${}^\bullet e$ and ${}^\bullet \lambda(e)$; the same holds for $e^\bullet$ and $\lambda(e)^\bullet$ ($\lambda$ preserves the environments of transitions)
- the restriction of $\lambda$ to $\min(N)$ is a bijection between $\min(N)$ and $m_{in}$ (the process starts at $m_{in}$)
- for all $e_1, e_2 \in E$, if ${}^\bullet e_1 = {}^\bullet e_2$ and $\lambda(e_1) = \lambda(e_2)$, then $e_1 = e_2$ ($\lambda$ does not duplicate the transitions of $\Sigma$)

The labelled occurrence net on Fig. 2 is a branching process of the system given on Fig. 1. The labels of the elements specify the correspondence with the elements of the system they represent.

Given a branching process $(N, \lambda)$ of $\Sigma$ and a B-cut $\gamma$ of $N$, then the set $\lambda(\gamma) = \{\lambda(b) \mid b \in \gamma\}$ is a reachable marking of $\Sigma$ ([2]), and we refer to it as the marking corresponding to $\gamma$. For example, the B-cut $\{5_2, 7_3\}$ corresponds to the reachable marking $\{5, 7\}$.

Let $(N_1, \lambda_1)$ and $(N_2, \lambda_2)$ be two branching processes of $\Sigma$, where $N_i = (B_i, E_i, F_i)$, $i = 1, 2$. We say that $(N_1, \lambda_1)$ is a *prefix* of $(N_2, \lambda_2)$ if $N_1$ is a subnet of $N_2$, and $\lambda_2|_{B_1 \cup E_1} = \lambda_1$. For any net system $\Sigma$, there exists a unique, up to isomorphism, maximal branching process of $\Sigma$. We will call it the *unfolding* of $\Sigma$, and denote it by $\text{Unf}(\Sigma)$.

A *run* of $\Sigma$ describes a particular history of $\Sigma$, in which conflicts have been solved, it is a branching process $(N, \lambda)$ such that there is no pair of elements $x, y$ in $N$ such that $x\#y$. Any run of $\Sigma$ is a prefix of the unfolding $\text{Unf}(\Sigma)$; we also say that it is a run on $\text{Unf}(\Sigma)$. A run in Fig. 2 is for example the labelled subnet whose elements are: $\{1_1, 10_1, 3_1, 4_3, 5_2\}$ as conditions and $\{n_1, b_1, e_1\}$ as events.

The following is an important property of the unfolding [3], which will be used in the next sections. Let $\Sigma = (P, T, F; m_{in})$, and $C$ be a finite configuration of $\text{Unf}(\Sigma) = (N, \lambda)$. Let $\Uparrow C = (N_C, \lambda')$, where $N_C$ is the unique (up to isomorphism) subnet of $N$ whose set of nodes is $\{x \mid x \notin (C \cup {}^\bullet C) \wedge \forall y \in C : \neg(x\#y)\}$ and $\lambda'$ is the restriction of $\lambda$ to the nodes of $N_C$. Then $\Uparrow C$ is the unfolding of the net system $(P, T, F; \lambda(\text{cut}(C)))$, up to isomorphism. From this property it follows that if $C_1$ and $C_2$ are finite configurations leading to the same marking, i.e.: $\lambda(\text{cut}(C_1)) = \lambda(\text{cut}(C_2)) = m$, then $\Uparrow C_1$ and $\Uparrow C_2$ are isomorphic to the unfolding of $(P, T, F; m)$, and they are isomorphic to each other. For example the finite configurations $[d_1]$ and $[f_1]$ lead to the same marking $\{4, 6\} = \lambda(\text{cut}([d_1])) = \lambda(\text{cut}([f_1]))$ and $\Uparrow [d_1]$ and $\Uparrow [f_1]$ are isomorphic to the unfolding of the system $\Sigma'$ with the same net of $\Sigma$ and initial marking $\{4, 6\}$.

The unfolding of a system is usually infinite. In [3] an algorithm is proposed which constructs a *finite complete prefix*, a *final* initial part of the unfolding containing as much information as the unfolding itself, in the sense that the unfolding can be constructed from its complete prefix. This notion of prefix was then further studied and generalized in the literature and used for studying various system properties, see for example [4, 5].

A branching process $(N, \lambda)$ of a net system $\Sigma$ is *complete* if for every reachable marking $m$ there exists a configuration $C$ in $N$ such that: $\lambda(\text{cut}(C)) = m$ (i.e.: $m$ is represented in $(N, \lambda)$), and for every transition $t$ enabled at $m$ ($m[t\rangle$) there exists a configuration $C \cup \{e\}$ such that $e \notin C$ and $\lambda(e) = t$.

The unfolding of a net system is always complete. Since a 1-safe net system has only a finite number of reachable markings, its unfolding contains at least one complete finite prefix.

In this paper we will use finite complete prefixes of the unfoldings of 1-safe net systems to study if the user has a strategy to avoid occurrences of a set of places. In particular, the complete prefixes we will use are obtained by adding to the one produced by the algorithm in [3] a finite set of events in order to guarantee a certain property as explained in section 4.1.

The branching process given in Fig. 2 is a finite complete prefix of the unfolding. This branching process is an extension of the one which is obtained by the algorithm presented in [3], to which events $i_2$ and $k_2$, have been added, see also section 4.1 in the following.

The full unfolding of this net system is finite; by adding transitions that reproduce the initial marking, one easily obtains a net system with an infinite unfolding. The following discussions (see Ex. 1 and 4), and the working of the algorithm in Sect. 4 would essentially apply to this cyclic case, but we chose to keep the nets as simple as possible.

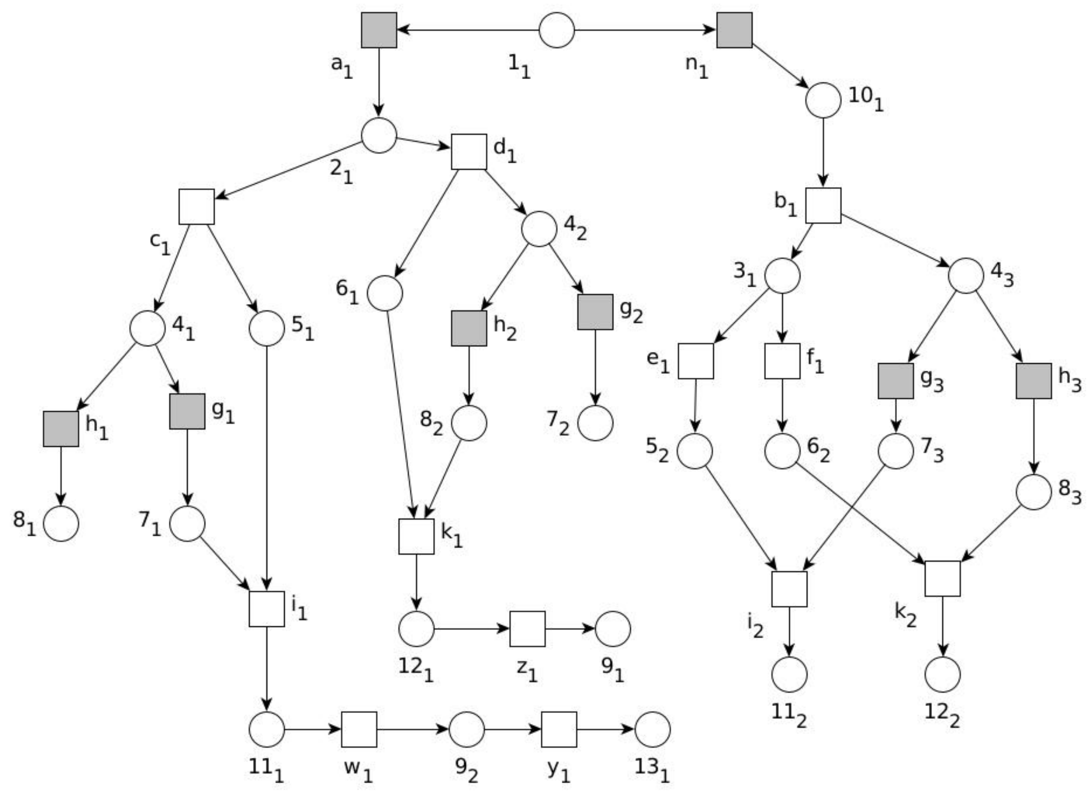

**Figure 2:** A branching process of the 1-safe net system in Fig.1.

## 3. The Game

In this section we formally define a two-player game on a 1-safe Petri net. Consider a 1-safe net system $\Sigma = (P, T, F; m_{in})$, where $T$ is partitioned into two disjoint subsets, $T_u$ and $T_{env}$. We will say that transitions in $T_u$ belong to the *user*, while transitions in $T_{env}$ belong to the *environment.* A transition is *controllable* if it belongs to $T_u$, *uncontrollable* otherwise. Events in the unfolding $\mathrm{Unf}(\Sigma)$ are partitioned into controllable events, $E_u$, and uncontrollable events, $E_{env}$, according to their labels.

We assume that $\Sigma$ satisfies the following constraints:

1. the subnet generated by $T_u$ is *extended free-choice*; by this we mean that, for each pair of transitions, $t_1$ and $t_2$, in $T_u$, if ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$, then ${}^\bullet t_1 = {}^\bullet t_2$; controllable transitions can be in conflict with uncontrollable ones without any extended free-choice constraint;
2. the user behaves sequentially; by this we mean that if $t_1$ and $t_2$ are in $T_u$, then, for each reachable marking $m$, if $m[t_1\rangle$ and $m[t_2\rangle$, then ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$, so that the two transitions are in conflict at $m$.

In this paper, we consider a game in which the user has the goal to avoid a given subset $S$ of places; we assume that the user cannot keep any of its transitions indefinitely enabled, whereas the environment can decide whether to fire or not transitions, when they are enabled; however, the progress assumption for the user does not guarantee that the solution of conflicts between a controllable transition and an uncontrollable one is in favour of the user.

A *play* in this game is then a (possibly infinite) run $\rho = (B_\rho, E_\rho, F_\rho, \lambda_\rho)$ in the unfolding of $\Sigma$, maximal with respect to controllable events. The play is won by the user if, for all $b \in B_\rho$, $b$ is not an occurrence of any place in $S$: $\forall b \in B_\rho : \lambda_\rho(b) \notin S$.

In pursuing its goal, the user can apply a strategy, based on the current state of the net. A *strategy* is a map $\alpha : [m_{in}\rangle \to 2^{T_u}$, such that

1. for each $m \in [m_{in}\rangle$ and for each $t \in \alpha(m)$, $t$ is enabled at $m$;
2. for each $m \in [m_{in}\rangle$ and $m$ reachable following $\alpha$, if $m[t\rangle$ for some $t \in T_u$, then $\alpha(m) \neq \emptyset$.

Since the environment can move concurrently with the user, a strategy is well defined if, and only if, for each pair of markings $m_1, m_2 \in [m_{in}\rangle$ such that $m_1$ and $m_2$ enable the same set of controllable transitions, and such that there is a sequence of uncontrollable transitions leading from $m_1$ to $m_2$, $\alpha(m_1) \subseteq \alpha(m_2)$ holds. This is due to the fact that the user in $m_1$ cannot be sure that while firing its transition, the system will not move to the marking $m_2$; therefore, any choice made in $m_1$ needs to be acceptable also in $m_2$. We do not require that $\alpha(m_2) = \alpha(m_1)$, because in $m_2$ the system may have evolved so that some uncontrollable alternatives are not possible anymore, therefore a higher number of transitions may be safe for the user to fire.

Let $e$ be a controllable event of a play $\rho = (B_\rho, E_\rho, F_\rho, \lambda_\rho)$, and $\alpha$ be a strategy. Then $e$ is justified by $\alpha$ if there is a cut $\gamma$ in $\rho$, containing ${}^\bullet e$ and following (or coinciding with) $\circ e$, such that $\alpha$ chooses the transition corresponding to $e$ in the marking corresponding to $\gamma$: $\lambda(e) \in \alpha(\lambda_\rho(\gamma))$.

We say that a play $\rho = (B_\rho, E_\rho, F_\rho, \lambda_\rho)$ is *consistent with a strategy* $\alpha$ iff every controllable event in the play is justified by $\alpha$.

A strategy $\alpha$ is *winning* for the user iff there is at least one play consistent with $\alpha$, and the user wins any play consistent with $\alpha$.

**Example 1.** *In the net system in Fig. 1 assume that the user controls transitions $a, n, g$ and $h$, and wants to avoid to reach place 13. A winning strategy is the function $\alpha : [\{1\}\rangle \to 2^{\{a,n,g,h\}}$ defined as follows: $\alpha(\{1\}) = \{a\}$, $\alpha(\{4,5\}) = \{h\}$, $\alpha(\{6,4\}) = \{g\}$, and $\alpha(m) = \emptyset$ for all the other $m \in [\{1\}\rangle$.*

*By following this strategy, the user satisfies the progress constraint and never reaches place 13.*

*On the prefix $(N, \lambda)$ in Fig. 2, the run whose events are $\{a_1, d_1, g_2\}$ is consistent with the strategy, since the controllable event $a_1$ follows the cut $\{1_1\}$, and $\lambda(a_1) = a \in \alpha(\lambda(\{1_1\}))$, and the controllable event $g_2$ follows the cut $\{6_1, 4_2\}$, and $g$ is chosen in the marking $\{4, 6\}$.*

*Note that the user cannot win by choosing transition $n$, since in the marking $\{3, 4\}$ it must satisfy the progress constraint: the user must select between $g$ and $h$ before knowing about the environment decision between $e$ and $f$, and this may always lead to reach place 13.*

## 4. Algorithm

The first part of this section recalls some notions used in [3] to construct a finite complete prefix, and proposes an extension of it. Then, in Sec. 4.2 we present our main contribution: an algorithm that checks on the extension of the complete prefix whether the user has a winning strategy.

## 4.1. Construction of an extended complete prefix

The structure that we will use to determine the existence of a winning strategy is an extension of the complete prefix produced by the algorithm in [3]. Other works discuss algorithms to find finite prefixes of the unfolding ([4, 5]); the construction of the prefix proposed in [3] is characterized by the definition of an order relation between configurations, and by the definition of *cut-off* event.

The order relation, denoted $\prec_{\mathrm{F}}$, is based on three elements: (1) the cardinality of configurations; (2) an arbitrarily defined total order $\ll$ on the elements of $T$, the set of transitions of the system net; (3) the Foata normal form of a configuration (see [6]). Here we aim to give an intuitive idea of how to compare two configurations according to $\prec_{\mathrm{F}}$, refer to [3] for details.

Given two configurations $C_1$ and $C_2$, if $|C_1| < |C_2|$, then $C_1 \prec_{\mathrm{F}} C_2$. If $|C_1| = |C_2|$, then we can order the elements of $C_1$ and $C_2$ according to $\ll$ forming two sequences $\sigma(C_1)$ and $\sigma(C_2)$; if $\sigma(C_1) \ll \sigma(C_2)$ in the lexicographic order, then $C_1 \prec_{\mathrm{F}} C_2$. Otherwise, if $|C_1| = |C_2|$ and $\sigma(C_1) = \sigma(C_2)$, we compare the Foata normal form of $C_1$ and $C_2$ according to the order $\ll$.

In [3] the authors prove that $\prec_{\mathrm{F}}$ is total for 1-safe nets.

For each event $e$ of a prefix $(N, \lambda)$ of $\mathrm{Unf}(\Sigma)$, in the following we denote $\mathrm{cut}(e) = \mathrm{cut}([e])$ and $\mathrm{mark}(e) = \lambda(\mathrm{cut}([e]))$.

**Definition 1.** *Let $(N, \lambda)$ be a prefix of $\mathrm{Unf}(\Sigma)$, and $e$ an event. The event $e$ is* cut-off *iff there is another event $e' \in (N, \lambda)$ such that the following conditions hold.*

- $\mathrm{mark}(e) = \mathrm{mark}(e')$*;*
- $[e'] \prec_{\mathrm{F}} [e]$.

The prefix in [3] is constructed starting from the initial cut, adding the enabled events to the prefix; when a cut-off event is reached, no event can be added in its future. The construction stops when there is no event that can be added to the prefix without being in the future of a cut-off. Since $\prec_{\mathrm{F}}$ is total in 1-safe nets, the number of non cut-off events never exceeds the number of reachable markings [3].

**Example 2.** *Consider the net system $\Sigma$ in Fig. 1 and $(N, \lambda)$ in Fig. 2. The net $(N, \lambda)$ is an extension of the complete prefix of $\Sigma$ as defined in [3]. The cut-off events are $z_1$, $e_1$, and $f_1$. The event $z_1$ is cut-off because of $w_1$, since $\mathrm{mark}[z_1] = \mathrm{mark}[w_1] = \{9\}$, and $[w_1] \prec_{\mathrm{F}} [z_1]$. The event $e_1$ is cut-off because of $c_1$, and $f_1$ because of $d_1$. The events $i_2$ and $k_2$ are not part of the prefix constructed as in [3] because they follow a cut-off event.*

As will be shown in Ex. 4, in general, a complete prefix as in [3] is not sufficient for us to check the existence of a winning strategy. For this reason, we extend it to a prefix $(N', \lambda')$ such that the following conditions (called *extention* conditions) hold.

1. Each terminal event $e$ in the prefix $(N', \lambda')$ satisfies one of the following:
    a) There is an event $e'$ in the prefix such that $e$ and $e'$ satisfy the conditions in Def. 1.

   b) There is no event $e'$ in the unfolding such that $e \prec e'$, i.e.: $e$ is terminal in the unfolding.
2. For each $e \in \mathrm{Unf}(\Sigma)$, if for each $b \in {}^\bullet e$, there exists $e_i \in (N', \lambda')$ such that $b \in e_i^\bullet$ and $e_i$ not terminal in $(N', \lambda')$, then $e \in (N', \lambda')$.

Intuitively, all the terminal events in $(N', \lambda')$ are either terminal also in $\mathrm{Unf}(\Sigma)$, or cut-off as in Def. 1 (condition 1). Moreover, any event of $\mathrm{Unf}(\Sigma)$ immediately following only non-terminal events in the prefix $(N', \lambda')$, must also belong to $(N', \lambda')$ (condition 2). Since in 1-safe systems the number of markings and the number of transitions are finite, we cannot continue adding infinitely often events always producing markings that have not been visited before, hence a finite prefix satisfying the above conditions exists.

**Example 3.** *The events $g_3$ and $h_3$ do not satisfy the conditions above, since they are neither cut-off events as in Def. 1, nor terminal on the unfolding. For this reason, we extended the prefix by adding $i_2$ and $k_2$. In the prefix in Fig. 2 all the events satisfy the extension conditions.*

## 4.2. Algorithm for the winning strategy

Let $\Sigma = (P, T, F, m_0)$ be a 1-safe net system, and $\mathrm{cp}(\Sigma) = (B, E, F, \lambda)$ an extended complete prefix of $\mathrm{Unf}(\Sigma)$, constructed as described in Sec. 4.1. We present an algorithm that checks whether the user has a winning strategy to avoid all the occurrences of bad places. Its pseudocode is in Algorithm 1. Here we provide an intuitive explanation of how it works. For technical reasons, we slightly modify the net system $\Sigma$ in input for the algorithm by adding a unique place initially marked and a controllable transition $\tau$ with this place as precondition, and the initial marking of $\Sigma$ as postconditions. Due to the progress constraint for the user, this event must fire in the initial marking, and after it, the modified net behaves as $\Sigma$, therefore the existence of a strategy is not affected. The algorithm starts from each occurrence of bad places in the prefix, and looks for the last controllable event preceding it. If this event is the only occurrence of $\tau$ in the prefix, then the user cannot prevent the system to reach a bad place, since the environment alone can reach it. Otherwise, this event is unique, since there is no marking enabling two concurrent controllable transitions, and the user must prevent its occurrence. Let $e \in E_u$ be such event; it can be avoided either by choosing another controllable event enabled in $\circ e$, or by preventing the system to arrive in $\circ e$. This is how the algorithm proceeds: first, it marks the event $e$ as bad choice (through the set bad_events), then it checks whether $\circ e$ enables a potentially good alternative. If not, the user cannot prevent the occurrence of a bad place when the system reaches $\circ e$, due to the progress constraint, therefore the user must prevent the system to reach that cut. Once again, this is checked by looking whether it is possible to change the last controllable choice preceding $\circ e$.

In a second round the algorithm checks whether there is any terminal event after which the user cannot prevent the occurrence of a bad place in the unfolding. This is done by comparing the marking of the local configuration of each terminal event with the set of bad_markings.

If the algorithm always finds an alternative to avoid all the occurrences of bad places, then it returns true, and the user has a winning strategy; otherwise it returns false.

**Example 4.** *Consider the net system in Fig. 1 and the prefix of its unfolding in Fig. 2. This prefix is extended complete. Assume that the goal of the user is to avoid place* $13$*.*

*On the prefix there is only one occurrence of* $13$*, namely* $(13_1)$*. The last controllable event preceding* $13_1$ *is* $g_1$*; in* $\circ g_1 = \{4_1, 5_1\}$*, the event* $h_1$ *is enabled, and can be an alternative to* $g_1$*, therefore the algorithm can stop the exploration of the past of* $13_1$*. The event* $g_1$ *is inserted in bad_events, all the cuts associated to local configurations following* $g_1$ *are put into bad_cuts, namely* $\{5_1, 7_1\}, \{11_1\}, \{9_2\}, \{13_1\}$*, and all the associated markings are put into bad_markings. Then, the algorithm starts to analyze the terminal event* $z_1$*. Since* $\mathrm{cut}(z_1)$ *is associated to a bad marking, the algorithm looks for the last controllable event, namely* $h_2$*. Since in* $\{4_2, 6_1\}$ *the event* $g_2$ *is enabled and it is not marked as bad, the algorithm stops the exploration of the past of* $\{9_1\}$*. Then, the events* $i_2$ *and* $k_2$ *are analysed. Since both* $\{11_2\}$ *and* $\{12_2\}$ *are associated to bad markings, in* $\{3_1, 4_3\}$ *both* $g_3$ *and* $h_3$ *are put into bad_events. Since there is no alternative to them, the algorithm must check if the cut* $\{3_1, 4_3\}$ *can be avoided, through a previous controllable choice. In this case, in the initial marking the user can avoid to fire* $n_1$*, and select its alternative* $a_1$*, therefore the user has a winning strategy, that is returned by the algorithm (also described in Ex. 1).*

*Note that if* $n$ *had been uncontrollable, we would not have had an alternative to avoid* $\{3_1, 4_3\}$*, therefore the user would not have had a winning strategy.*

*The algorithm would not return the right answer when we look for it in the smaller prefix defined in [3]: this prefix does not include the events* $i_2$ *and* $k_2$*, so the terminal events are* $e_1$*,* $f_1$*,* $g_3$*, and* $h_3$*.* $\mathrm{cut}(e_1) = \{5_2, 4_3\}$ *and* $\mathrm{cut}(f_1) = \{6_2, 4_3\}$ *are not associated to bad_markings, and this is correct since from these cuts the user is always able to avoid place* $13$*; however, also* $\mathrm{cut}(g_3) = \{3_1, 7_3\}$ *and* $\mathrm{cut}(h_3) = \{3_1, 8_3\}$ *are not associated to bad_markings, although the user cannot be sure to win by reaching them. This is due to the fact that* $g_3$ *and* $h_3$ *are not cut-off events as defined in Def. 1, and they are terminal in the prefix only because each event following them also follows* $e_1$ *or* $f_1$*, which are cut-off. This is the reason why we needed to extend the prefix by imposing a stronger condition on terminal events: by running the algorithm on the smaller prefix the strategy would allow to fire* $n$ *in the initial marking and* $g_3$ *and* $h_3$ *in* $\{3_1, 4_3\}$*, possibly leading the user to lose.*

In what follows we will prove that Algorithm 1 returns true if the user has a strategy to avoid all the occurrences of bad places in $S$, false otherwise.

Let $E$ be the set of all the events in the prefix, $E_u$ the set of the controllable ones, and $E_b$ the set of events in bad_events after the end of the execution of Algorithm 1. We denote with $E_g = E_u \setminus E_b$ the set of controllable events not classified as bad_events.

**Lemma 1.** *At the end of Algorithm 1, from all the cuts in bad_cuts the user cannot prevent the occurrence of a bad place.*

*Proof.* The first set of calls to find_alternative (line 6) analyse the past of the occurrences of bad places in the prefix. We show inductively that in each call, from all the cuts in bad_cuts the user cannot avoid to reach an occurrence of a bad place. In the first call of find_alternative, in the first execution of the while cycle, the set bad_cuts includes only cuts from which an occurrence of a bad place can be reached through uncontrollable events, of which the user cannot prevent the occurrence. The set bad_events includes the last controllable event $e_1$. By construction, if $e_1$ is

**Algorithm 1** Algorithm to find the existence of a winning strategy

```
function exStrategy($\Sigma$, cp($\Sigma$), $S$)  ▷ cp($\Sigma$) is a complete prefix of $\Sigma$, $S$ is the set of bad places
                                   ▷ Returns True if there is a winning strategy, False otherwise
    global bad_cuts $\leftarrow$ []; bad_markings $\leftarrow$ []; bad_events $\leftarrow$ []
    for $b \in B$ with $\lambda(b) \in S$ do
        $e \leftarrow {}^\bullet b$
        st $\leftarrow$ find_alternative(cut($e$))
        if st = False then
            return False
        end if
    end for
    repeat
        old_bad_cuts $\leftarrow$ bad_cuts
        for all terminal event $e$ not marked as bad do
            if mark($e$) $\in$ bad_markings then
                st $\leftarrow$ find_alternative(cut($e$))
                if st = False then
                    return False
                end if
            end if
        end for
    until old_bad_cuts = bad_cuts
    return True
end function

function find_alternative($\gamma_f$)        ▷ Looks for a controllable conflict in the past of $\gamma_f$
    alternative $\leftarrow$ False
    while alternative = False do
        $e_c \leftarrow$ last_contr_event($\gamma_f$)
        if $e_c = null$ then
            return False
        end if
        bad_events.add($e_c$)
        bad_cuts.add(cut($e_c$))
        bad_markings.add(mark($e_c$))
        for $e$ in the interval [cut($e_c$), $\gamma_f$] do
            bad_cuts.add(cut($e$))
            bad_markings.add(mark($e$))
        end for
        $\gamma = (\text{cut}(e_c) \setminus e_c^\bullet) \cup {}^\bullet e_c$
        if contr_enabled($\lambda(\gamma)$) $\cap$ $\lambda(E \setminus$ bad_events) $\neq \emptyset$ then
            alternative $\leftarrow$ True
        end if
        $\gamma_f \leftarrow \gamma$
    end while
    return True
end function
```

in bad_events, $\mathrm{cut}(e_1)$ is in bad_cuts. We assume that after $n$ iterations the property still holds. By hypothesis, at the $n+1$-th iteration, the property holds in $\mathrm{cut}(e_c) = (\gamma_f \setminus {}^\bullet e_c) \cup e_c^\bullet$ assigned during the $n$-th iteration. Since the iteration did not stop after $n$ steps, all the controllable events enabled in $\gamma_f$ have been analysed and put in bad_events.

Since the user's subnet is free-choice, for each controllable $e_i$ in conflict with $e_c$, $\gamma_f = \circ e_c = \circ e_i$. From $\gamma_f$, arriving in $\mathrm{cut}(e_c)$ or in $\mathrm{cut}(e_i)$ for some event $e_i$ in conflict with $e_c$ is unavoidable for the user due to the progress constraints, and both $\mathrm{cut}(e_c)$ and all the $\mathrm{cut}(e_i)$ have been included in bad_cuts in the previous steps. From all the cuts included in bad_cuts at the $(n+1)$-th iteration, we can reach $\gamma_f$ by firing only uncontrollable event. Therefore the user cannot prevent the system to arrive in $\gamma_f$, and from it the bad occurrence is reachable by inductive hypothesis.

The second set of calls (line 15) is used to analyse the terminal events in the prefix. From the previous point, we know that when these calls start, for all the cuts in bad_cuts the property holds. Let $e$ be a terminal event whose local configuration is associated to a bad marking $m$. Since $m$ is in bad_markings, there must be a cut $\gamma' \in$ bad_cuts such that $\lambda(\gamma') = m$. In the unfolding, the subnet starting from $\gamma'$ and the one starting from $\gamma$ are isomorphic, therefore the user cannot prevent the system to reach an occurrence of a bad place from $\gamma$. To prove the thesis, we can repeat the above reasoning about the iterations inside find_alternative. □

A consequence of Lemma 1 is that after firing an event in bad_events, the user cannot prevent the system to arrive in an occurrence of a bad place, since if $e \in$ bad_events, then cut$(e) \in$ bad_cuts.

**Lemma 2.** *At the end of Algorithm 1, if a marking $m$ belongs to the set of bad_markings, then from any cut $\gamma$ of the unfolding such that $\lambda(\gamma) = m$ the user has no winning strategy, i.e.: the user cannot prevent the occurrence of a bad place.*

*Proof.* Let $\gamma$ be a cut such that $\lambda(\gamma) = m \in$ bad_markings, we have two cases: either $\gamma$ is in bad_cuts, and then this lemma is true by Lemma1, or $\gamma$ is not in bad_cuts, in this case, since $m$ is bad, there must be another cut $\gamma'$ in bad_cuts corresponding to $m$ and such that, always by Lemma 1, the user cannot prevent the occurrence of a bad place from it.

Since $\gamma$ and $\gamma'$ correspond to the same marking and then their future in the unfolding are isomorphic, the user cannot prevent the occurrence of a bad place also from $\gamma$. □

The set of events not in bad_events can be used to construct a strategy for the user.

**Remark 1.** *It is immediate to note that: at the end of Algorithm 1, if a reachable marking $m$ does not belong to bad_markings, then for any cut $\gamma$ such that $\lambda(\gamma) = m$ it holds: $\gamma \notin$ bad_cuts.*

**Lemma 3.** *At the end of Algorithm, if a reachable marking $m$ does not belong to bad_markings and $m = \lambda(\mathrm{cut}(e_c))$ for some $e_c \in E_u$, then, starting from any $\gamma$ such that $\lambda(\gamma) = m$, the user is able to prevent the occurrence of a bad place.*

*Proof.* Since $m$ is not in bad_markings, by the previous remark, any cut $\gamma$ such that $\lambda(\gamma) = m$ does not belong to the set of bad_cuts, moreover we know that the future in the unfolding from any such $\gamma$ are isomorphic. Being $m$ not in bad_markings, and then $e_c \notin$ bad_events, we consider two cases. (1) There is in the prefix a condition $b$ such that $\lambda(b) \in S$, and such that $e_c \prec b$; then, the algorithm going backward from $b$ found a controllable event $e$ leading to b, $e_c \prec e \prec b$, and in conflict with an other controllable event not belonging to bad_events. The latter event will be suggested by the strategy. (2) If there is no such condition $b$, and there is, always in the prefix, a terminal event $e'$, $e_c \prec e'$ such that $\mathrm{mark}(e') \in$ bad_markings, then again the algorithm going backward from $cut(e')$ found a controllable event $e''$, $e'' \prec e'$, in conflict with another controllable one not belonging to the set of bad_events. Also this last one will be suggested by the strategy.

□

**Remark 2.** *A consequence of Lemma 3 is that if the algorithm returns true, the user has a winning strategy to avoid all the occurrences of bad places in the prefix, since the initial cut cannot be in bad_cuts. Furthermore, if the algorithm returns true, the user can avoid reaching all the cuts in bad_cuts, and to fire all the events in bad_events.*

**Theorem 1.** *Algorithm 1 is correct, i.e. there is a winning strategy for the user iff it returns true.*

*Proof.* We first consider the case in which the algorithm returns false. This happens if the function find_alternative returns false, which means that there is no controllable event (1) in the past of an occurrence of a bad place, (2) in the past of a cut in bad_cuts, or (3) in the past of a cut of a local configuration of a terminal event, associated to the same marking of a cut in bad_cuts.

In case (1), there is no winning strategy on the prefix, and a fortiori on the unfolding, that extends the prefix; in case (2) the non-existence of a winning strategy is a consequence of Lemma 1; in case (3) the non-existence of a winning strategy is a consequence of Lemma 2 and of the isomorphism of the subnets of the unfolding starting from two cuts associated to the same marking.

Finally, we consider the case in which the algorithm returns true. This happens when any place $b$ in the prefix such that $\lambda(b) \in S$ can be avoided by the strategy by choosing a controllable event in conflict with a controllable one leading to $b$ (see Remark 2). Let $e$ be any terminal event. If $\mathrm{cut}(e)$ is in bad_cuts, then the user will never fire it by following the strategy. The same happens if $\mathrm{cut}(e) \notin$ bad_cuts, but there is a place $b$ such that $\lambda(b) \in S$, and $b \prec e$, as discussed in Remark 2. Hence, the only terminal events that we can reach by following the strategy are those, not belonging to bad_events, such that there is no $b$ preeceding them such that $\lambda(b) \in S$. Let $e$ be one of such events. Since $e$ is terminal, there must be an other event $e'$ in the prefix such that $\mathrm{mark}(e) = \mathrm{mark}(e')$. Lemma 3 guarantees that from $\mathrm{cut}(e')$ the user can avoid all the occurrences of bad places. Since the part of the unfolding starting from $\mathrm{cut}(e)$ and from $\mathrm{cut}(e')$ are isomorphic, we can "past" the part of the unfolding following $\mathrm{cut}(e')$ to $\mathrm{cut}(e)$. We can repeat the pasting operation for all the terminal events $e''$ such that $\mathrm{cut}(e'') \notin$ bad_cuts, and $\nexists b$ bad place such that $b \prec e''$. In this larger prefix the user has a strategy to avoid all the occurrences of bad places. We can repeat this pasting operation an arbitrary number of time. For each of such prefix elongations the user has a winning strategy. □

**Theorem 2.** *Algorithm 1 terminates.*

*Proof.* This is a consequence of the finiteness of the prefix: occurrences of bad places and cut-off events are finite in the prefix, and so is their past. □

**Strategy** Algorithm 1 does not explicitly provide any strategy. However, for each event $e \in E_u$, we can easily define a strategy in $\lambda(\circ e)$ by considering the set $E_{\circ e}$ of controllable events enabled in $\circ e$ belonging to $E_g$, namely $\alpha(\lambda(\circ e)) = E_g \cap \circ e$. In general, this strategy is only partial, and not defined in some markings, but it is sufficient to compute a set of good choices for every marking that is reached in the system while following the strategy. In particular, assume that in a certain execution the user observes the general marking $m$. If there is no controllable transition enabled in $m$, then the user cannot make any choice; otherwise, we can look for an occurrence of a cut $\gamma$ associated to $m$ in the prefix, enabling a set of controllable events. Such a cut must exist, since the prefix extends the one in [3], where all the markings are represented. Let $e \in E_u$ be an event enabled in $\gamma$, and consider $\circ e$. All the winning choices in $\lambda(\circ e)$ are also winning in $m$, since all the events between $\circ e$ and $\gamma$ are uncontrollable and concurrent with $e$, and the controllable events enabled in $\circ e$ are also enabled in all the cuts $\gamma$ such that $\lambda(\gamma) = m$ since the controllable component of the user is extended free-choice. Therefore, from $\lambda(\circ e)$, the user cannot be sure that, while executing any of its controllable transitions, $m$ is not reached.

## 5. Conclusion and related works

In this work we presented a two-player game on the unfolding of a Petri net, where the goal of the user is to avoid to reach a certain set of places, and we proposed an algorithm to check whether the user has a winning strategy on a complete prefix of the unfolding. Although we assume that there is no place inaccessible for the user to observe, the strategy cannot rely on local states whose value may change due to uncontrollable transitions.

In a previous work [7], we modelled a controlled reachability problem with a game on the unfolding of 1-safe nets. We considered a progress constraint on the environment, leaving the user free to decide whether to fire its transitions or not. Although also the algorithm proposed in [7] was based on a prefix of the unfolding, with the reversed progress constraint, we could not use local configurations to decide the existence of a winning strategy. Contrary to what happens in this game, in that context, a good strategy collects as much information as possible before selecting a controllable choice. Because of this, for many nets the constructed prefix was larger than the complete prefix considered in this work.

Another notion of game on Petri nets was defined in 2014 by Finkbeiner and Olderog [8]. In these Petri games [9, 10, 11], the players, represented by tokens of the net, are divided into two teams: the environment and the system. The goal of the system is to avoid a given set of markings. In their game, the players have a causal memory of their past, and get information about the history of other players through synchronizations on the same transitions.

Another important line of works considering controlled systems was started by Ramadge and Wonham on labelled transition systems [12, 13]; starting from their theory, some authors developed techniques based on regions to implement a controller limiting the behaviours of the net, so to cut out undesired behaviour [14, 15]. In general, the body of works analysing control methods on formal models is quite large, refer to [16, 17] for an overview.

In our approach we do not need to compute the whole set of reachable markings to decide whether the user has a winning strategy, but we only consider markings associated to local configurations of the events. It is known that the complete prefix in [3] can reach the dimension of the marking graph, but in many cases it is smaller [3, 4, 5]. We plan to develop an algorithm to compute the extension proposed in this paper and to make experiments to check how its dimension changes.

In addition we plan to consider larger classes of nets, both by considering more general user's components, and by considering more than two players. On this line, we plan to relate ourselves to the works considering the analysis of multi-agent systems, such as [18, 19].

Another extension that we are planning concerns the knowledge of the user on the system. Since some places may contain private information, the user may not be able to ever observe their value. In this case, the user should make the same choice in any pair of markings that are indistinguishable from its point of view. We believe that a strategy satisfying this constraint, if one exists, can also be found on the prefix, by restricting the choices that the algorithm currently selects as good.

Finally, we plan to consider different goals for the user, for example analysing formulas expressed with the temporal logic ATL [20].

## Acknowledgments

This work has been supported by the Italian Ministero dell'Università e della Ricerca. The authors thank the anonymous reviewers.

## References

[1] T. Murata, Petri nets: Properties, analysis and applications, Proceedings of the IEEE 77 (1989) 541–580. doi:10.1109/5.24143.

[2] J. Engelfriet, Branching processes of Petri nets, Acta Inf. 28 (1991) 575–591. URL: https://doi.org/10.1007/BF01463946. doi:10.1007/BF01463946.

[3] J. Esparza, S. Römer, W. Vogler, An improvement of McMillan's unfolding algorithm, Formal Methods in System Design 20 (2002) 285–310. doi:10.1023/A:1014746130920.

[4] V. Khomenko, M. Koutny, W. Vogler, Canonical prefixes of Petri net unfoldings, Acta Informatica 40 (2003) 95–118.

[5] B. Bonet, P. Haslum, V. Khomenko, S. Thiébaux, W. Vogler, Recent advances in unfolding technique, Theoretical Computer Science 551 (2014) 84–101.

[6] V. Diekert, Combinatorics on traces, volume 454, Springer Science & Business Media, 1990.

[7] F. Adobbati, L. Bernardinello, L. Pomello, A two-player asynchronous game on fully observable Petri nets., Trans. Petri Nets Other Model. Concurr. 15 (2021) 126–149.

[8] B. Finkbeiner, E. Olderog, Petri games: Synthesis of distributed systems with causal memory, in: A. Peron, C. Piazza (Eds.), Proceedings Fifth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2014, Verona, Italy, September

10-12, 2014, volume 161 of *EPTCS*, 2014, pp. 217–230. URL: https://doi.org/10.4204/EPTCS.161.19. doi:10.4204/EPTCS.161.19.

[9] B. Finkbeiner, E. Olderog, Petri games: Synthesis of distributed systems with causal memory, Inf. Comput. 253 (2017) 181–203. URL: https://doi.org/10.1016/j.ic.2016.07.006. doi:10.1016/j.ic.2016.07.006.

[10] M. Gieseking, E. Olderog, N. Würdemann, Solving high-level Petri games, Acta Informatica 57 (2020) 591–626. URL: https://doi.org/10.1007/s00236-020-00368-5. doi:10.1007/s00236-020-00368-5.

[11] B. Finkbeiner, M. Gieseking, J. Hecking-Harbusch, E. Olderog, Global winning conditions in synthesis of distributed systems with causal memory, in: F. Manea, A. Simpson (Eds.), 30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference), volume 216 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 20:1–20:19. URL: https://doi.org/10.4230/LIPIcs.CSL.2022.20. doi:10.4230/LIPIcs.CSL.2022.20.

[12] P. J. Ramadge, W. M. Wonham, Supervisory control of a class of discrete event processes, SIAM journal on control and optimization 25 (1987) 206–230.

[13] P. J. Ramadge, W. M. Wonham, The control of discrete event systems, Proceedings of the IEEE 77 (1989) 81–98.

[14] A. Ghaffari, N. Rezg, X. Xie, Design of a live and maximally permissive Petri net controller using the theory of regions, IEEE Transactions on Robotics and Automation 19 (2003) 137–141. doi:10.1109/TRA.2002.807555.

[15] Z. Li, M. Zhou, M. Jeng, A maximally permissive deadlock prevention policy for FMS based on Petri net siphon control and the theory of regions, IEEE Transactions on Automation Science and Engineering 5 (2008) 182–188.

[16] A. Giua, M. Silva, Petri nets and automatic control: A historical perspective, Annual Reviews in Control 45 (2018) 223–239.

[17] W. Wonham, K. Cai, K. Rudie, Supervisory control of discrete-event systems: A brief history, Annual Reviews in Control 45 (2018) 250–256.

[18] W. Jamroga, W. Penczek, T. Sidoruk, P. Dembinski, A. W. Mazurkiewicz, Towards partial order reductions for strategic ability, J. Artif. Intell. Res. 68 (2020) 817–850. URL: https://doi.org/10.1613/jair.1.11936. doi:10.1613/jair.1.11936.

[19] R. Lukomski, K. Wilkosz, Modeling of multi-agent system for power system topology verification with use of Petri nets, in: 2010 Modern Electric Power Systems, IEEE, 2010, pp. 1–6.

[20] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (2002) 672–713. URL: https://doi.org/10.1145/585265.585270. doi:10.1145/585265.585270.