# An Asynchronous Game on Distributed Petri Nets

Federica Adobbati, Luca Bernardinello, and Lucia Pomello

Dipartimento di informatica, sistemistica e comunicazione
Università degli studi di Milano - Bicocca
viale Sarca 336 U14
Milano, Italia

**Abstract.** A Petri net is distributed if its elements can be assigned to a set of locations so that each element belongs to exactly one location, and each transition belongs to the same location as its input places.
We define an asynchronous game played on the unfolding of a distributed net with two locations, the 'User' and the 'Environment'. The user can control the transitions in its location. A play in the game is a run in the unfolding, together with a sequence of cuts in that run. The rules of the game require that the environment satisfies a progress constraint: no transition in its location can be indefinitely postponed. In the general case, the game can be defined so that the user can observe only some transitions. In this paper, we only consider the case in which all transitions are observable, and study a reachability problem, in which the user tries to fire a target transition. We propose an algorithm which decides if the user has a winning strategy and, if so, computes a winning strategy.

## 1 Introduction

The ideas behind this paper were conceived while studying the problem of *weak observable liveness* ([4] and [1]), where we suppose that a Petri net models a system comprising a user and an environment; the user controls a subset of transitions, and observes a subset of transitions. The aim of the user is to force liveness of a special transition (the *target*), whatever the behaviour of the environment. The environment is supposed to guarantee progress of uncontrollable transitions.

The problem can be stated as deciding whether the user has a strategy allowing him to achieve his aim, irrespective of the choices of the environment. The strategy is formalized as a *response function*, mapping observations (sequences of observable transitions) to sets of controllable transitions.

In a first attempt to develop an algorithm for finding a strategy, the problem was translated into an infinite game on a finite graph, where the finite graph is derived from the marking graph of the net ([1]). Besides the usual problem of state explosion, this approach hides the potential concurrency in the net, by using an interleaving semantics.

Hence, the authors started to explore the idea of defining an asynchronous game, to be played on the unfoldings of Petri nets, in which to encode the weak observable liveness problem, but also several other problems, formalized by defining a suitable aim for the user. Such a game was proposed in [2], where its application to weak observable liveness was studied.

Other possible applications of such a game could be in the general frame of verification, adaptation and control of distributed systems; so that, in the case of a winning strategy for the user with respect to a specific behavioral property, the system model could be adapted imposing that specific property, for example by adding an interacting component implementing the user behavior.

In this paper, we consider *distributed* net systems, in which all choices are local to one component, restrict to the case of exactly two components (user and environment), and study a reachability problem, in which the user tries to fire a target transition, under the hypothesis of full observability, and propose an algorithm which decides if the user has a winning strategy and, if so, computes a winning strategy.

In the next section, we recall the needed notions about Petri nets, distributed Petri nets, and unfoldings, In Sect. 3 we define the general game, and the notions of strategy and winning strategy. The problem of controlled reachability is introduced in Sect. 4, together with the algorithm looking for a winning strategy.

Several approaches to notions of asynchronous games are briefly discussed in Sect. 5, while prospects for future developments are presented in the final section.

## 2 Petri nets

A Petri net models a concurrent system. The basic elements of a net are local states (places) and local transitions. The global state of a net is distributed among its local states. Several types of nets have been defined and studied. Here, we use *elementary* nets, where local states can be seen as Boolean variables, or propositions. When a transition occurs, it changes the value of local states in its neighbourhood.

**Definition 1.** *A net is a triple $N = (P, T, F)$, where $P$ and $T$ are disjoint sets. The elements of $P$ are called* places *and represented by circles, the elements of $T$ are called* transitions *and represented by squares. $F$ is called* flow relation, *with $F \subseteq (P \times T) \cup (T \times P)$, and is represented by arcs.*

By $F^*$ we denote the reflexive and transitive closure of $F$.

Let $x \in P \cup T$ be an element of the net, the pre-set of $x$ is the set ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$, the post-set of $x$ is the set $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$.

A place $p$ is a *side condition* if ${}^\bullet p \cap p^\bullet \neq \emptyset$. We assume that there are no side conditions, and that any transition has non-empty pre-set and post-set: $\forall t \in T$, ${}^\bullet t \neq \emptyset$ and $t^\bullet \neq \emptyset$.

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

If places and transitions are enumerated, a net can be also represented by a $|P| \times |T|$ matrix, called *incidence matrix*, where the value in the ith row and jth column is equal to $-1$ if $p_i \in^\bullet t_j$, 1 if $p_i \in t_j^\bullet$, and equal to 0 otherwise.

Two elements $x, y \in P \cup T$ are said to be in *conflict*, denoted $x\#y$, iff there exist $t_1, t_2 \in T : t_1 \neq t_2, t_1 F^* x, t_2 F^* y \wedge \exists p \in {}^\bullet t_1 \cap {}^\bullet t_2$.

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.

A net $N' = (P', T', F')$ is a *subnet* of $N = (P, T, F)$ if $P' \subseteq P$, $T', \subseteq T$, and $F'$ is $F$ restricted to the elements in $N'$.

**Definition 2.** *An* elementary net system *is a quadruple $\Sigma = (P, T, F, m_0)$ consisting of a finite net $N = (P, T, F)$ and an* initial marking *$m_0 \subseteq P$. A* marking *is a subset of $P$ and it represents a global state.*

A transition $t$ is *enabled* at a marking $m$, denoted $m[t\rangle$, if ${}^\bullet t \subseteq m \ \wedge \ t^\bullet \cap m = \emptyset$. A transition $t$, enabled at $m$, can *occur* (or *fire*) producing a new marking $m' = t^\bullet \cup m \setminus {}^\bullet t$, denoted $m[t\rangle m'$. A marking $m'$ is reachable from an other marking $m$, if there is a sequence $t_1 t_2 \ldots t_n$ such that $m[t_1\rangle m_1[t_2\rangle \ldots m_{n-1}[t_n\rangle m'$; in this case, we write $m' \in [m\rangle$. The set of reachable markings is the set of markings reachable from the initial marking $m_0$, denoted $[m_0\rangle$.

Two transitions, $t_1$ and $t_2$, are *concurrent* at a marking $m$ if they are independent and both enabled at $m$.

A marking $m$ is a *contact* for a transition $t$ if ${}^\bullet t \subseteq m$ and $t^\bullet \cap m \neq \emptyset$. A net system is *contact-free* if no reachable marking is a contact.

In this paper, we consider only contact-free elementary net systems.

The non sequential behaviour of elementary net systems can be recorded by occurrence nets, which are used to represent by a single object the set of potential histories of an elementary net system.

**Definition 3.** *A net $N = (B, E, F)$ is an* occurrence net *if*

- *for all $b \in B$, $|{}^\bullet b| \leq 1$*
- *$F^*$ is a partial order on $B \cup E$*
- *for all $x \in B \cup E$, the set $\{y \in B \cup E \mid yF^*x\}$ is finite*
- *for all $x \in B \cup E$, $x\#x$ does not hold*

We will say that two elements $x$ and $y$, $x \neq y$, of $N$ are *concurrent*, and write $x$ **co** $y$, if they are not ordered by $F^*$, and they are not in conflict.

By $\min(N)$ we will denote the set of minimal elements with respect to the partial order induced by $F^*$.

A *B-cut* of $N$ is a maximal set of pairwise concurrent elements of $B$. B-cuts represent potential global states through which a process can go in a history of the system. 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 ${}^\bullet e \subseteq \gamma$. We will denote by $\gamma + e$ the B-cut $(\gamma \setminus {}^\bullet e) \cup e^\bullet$. A B-cut is a *deadlock cut* if no event is enabled at it.

Let $\Gamma$ be the set of B-cuts of $N$. A partial order on $\Gamma$ can be defined as follows: let $\gamma_1, \gamma_2$ be two B-cuts. We say $\gamma_1 < \gamma_2$ iff

1. $\forall y \in \gamma_2 \exists x \in \gamma_1 : xF^* y$
2. $\forall x \in \gamma_1 \exists y \in \gamma_2 : xF^* y$
3. $\exists x \in \gamma_1, \exists y \in \gamma_2 : xF^+ y$

In words, $\gamma_1 < \gamma_2$ if any condition in the second B-cut is or follows a condition of the first B-cut and any condition in the first B-cut is or comes before a condition of the second B-cut (and there exists at least one condition coming before).

A sequence of B-cuts, $\gamma_0 \gamma_1 \dots \gamma_i \dots$ is *increasing* if $\gamma_i < \gamma_{i+1}$ for all $i \geq 0$.

We will say that an event $e \in E$ precedes a B-cut $\gamma$, and write $e < \gamma$, iff there is $y \in \gamma$ such that $eF^+ y$. In this case, each element of $\gamma$ either follows $e$ or is concurrent with $e$ in the partial order induced by the occurence net.

**Definition 4.** *A* branching process *of $\Sigma = (P, T, F, m_0)$ is an occurrence net $N = (B, E, F)$, together with a labelling function $\mu : B \cup E \to P \cup T$, such that*

- *$\mu(B) \subseteq P$ and $\mu(E) \subseteq T$*
- *for all $e \in E$, the restriction of $\mu$ to ${}^\bullet e$ is a bijection between ${}^\bullet e$ and ${}^\bullet \mu(e)$; the same holds for $e^\bullet$*
- *the restriction of $\mu$ to $\min(N)$ is a bijection between $\min(N)$ and $m_0$*
- *for all $e_1, e_2 \in E$, if ${}^\bullet e_1 = {}^\bullet e_2$ and $\mu(e_1) = \mu(e_2)$, then $e_1 = e_2$*

For $\gamma$ a B-cut of $N$, the set $\{\mu(b) \mid b \in \gamma\}$ is a reachable marking of $\Sigma$, and we refer to it as the marking corresponding to $\gamma$.

Let $(N_1, \mu_1)$ and $(N_2, \mu_2)$ be two branching processes of $\Sigma$. We say that $(N_1, \mu_1)$ is a *prefix* of $(N_2, \mu_2)$ if $N_1$ is a subnet of $N_2$, and

- $\min(N_1) = \min(N_2)$
- if $b \in B_1$ and $(e, b) \in F_2$, then $e \in E_1$
- if $e \in E_1$, and $b$ is either a precondition or a postcondition of $e$ in $N_2$, then $b \in B_1$

For any elementary 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 $\textsc{unf}(\Sigma)$ (see [5]).

A *run* of $\Sigma$ is a branching process $(N, \mu)$ describing a particular history, in which conflicts have been solved, i.e.: such that the conflict relation $\#$ is empty on its set of elements. Any run of $\Sigma$ is a prefix of the unfolding $\textsc{unf}(\Sigma)$, we say it is a run on $\textsc{unf}(\Sigma)$.

We suppose that $\Sigma$ models an *Environment* interacting with a *User*. Direct interactions happen by means of a subset of *controllable* transitions; whenever such a transition is enabled, the User can decide to fire it or not. In the general setting, we assume that some uncontrollable events are not observable by the User.

**Definition 5.** *A controlled net system is an elementary net system $\Sigma = (P, T, F, m_0, K, V)$, where $K \subseteq V \subseteq T$. The transitions in $V$ are* observable*, those in $K$ are* controllable*; the set $NK = T \setminus K$ is the set of* uncontrollable *transitions.*

Let $\text{UNF}(\Sigma) = (B, E, F, \mu)$ be the unfolding of a controlled system $\Sigma$, then $E_c = \{e \in E \mid \mu(e) \in K\}$ is the set of occurrences of controllable transitions, called *controllable events*, and $E_{nc} = E \setminus E_c$ is the set of occurrences of uncontrollable transitions, called *uncontrollable events*.

In the graphical representation, controllable transitions and events will be represented by black squares.

We assume that choices among transitions are local either to the Environment or to the User, and moreover, that both Environment and User have a sequential behaviour. We are therefore in the setting of distributed net system, as introduced and studied in [3] and in [8].

**Definition 6.** *A distributed net system over a set $L$ of locations is a system $\Sigma = (P, T, F, m_0)$ together with a map*

$$\lambda : (P \cup T) \to L$$

*such that*

1. *For every $p \in P$, $t \in T$, if $p \in {}^\bullet t$, then $\lambda(p) = \lambda(t)$;*
2. *For every pair $t, u \in T$, if $t$ and $u$ are concurrent at some reachable marking $m$, then $\lambda(t) \neq \lambda(u)$.*

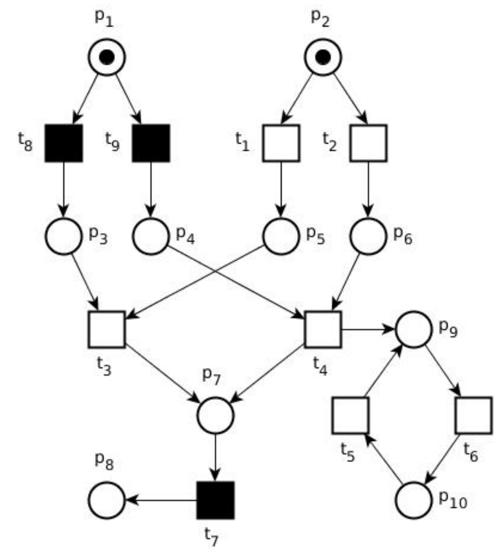

**Fig. 1.** A distributed controlled net system with two locations

We are interested in the special case of distributed controlled net systems $\langle \Sigma, \lambda \rangle$ such that $L = \{A, G\}$, i.e. in distributed net systems with only two components, representing the Environment and the User, respectively; and such that

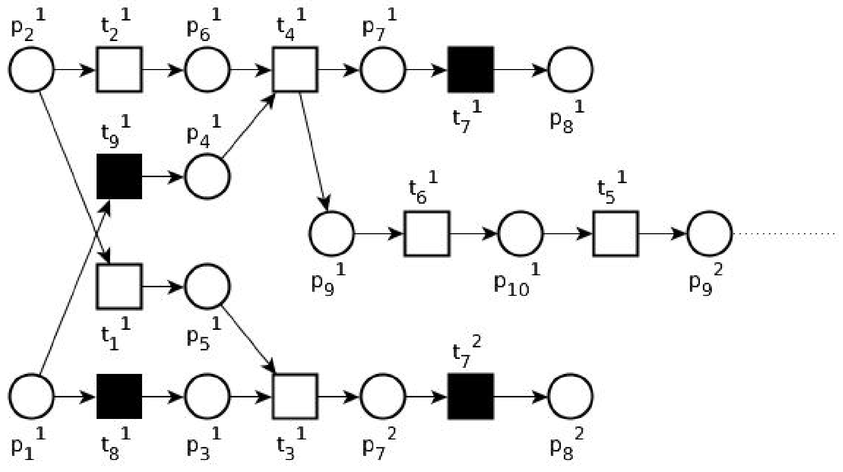

**Fig. 2.** The unfolding of a distributed controlled net system with two locations

all and only the transitions belonging to $G$ are controllable; whereas transitions belonging to $A$ are not controllable, but can be observable.

In this kind of nets, when a transition is enabled, it can never be disabled by the occurrence of transitions belonging to different componentsů In the case of a cycle this observation justifies the following lemma.

**Lemma 1.** *Let $(\Sigma, \lambda)$ be a distributed controlled net system with two locations, $A$ and $G$. Let $m$ be a marking, and*

$$m_1[t_1\rangle m_2[t_2\rangle m_3[...\rangle m_1$$

*be a firing sequence with $t_i \in A$ for each $i$. Then, if $t \in G$ is enabled at $m_i$ for some $i$ between the two repetitions of $m_1$, then $t$ is enabled at $m_j$ for each $m_j$ in the cycle.*

*Example 1.* Figure 1 shows a distributed controllable net system with two locations. Places are not explicitly divided into the two components, because their partition can be inferred by their post-transitions, moreover they are immaterial in order to determine a strategy. A prefix of the unfolding of the system is shown in Fig. 2. Each element of the unfolding is decorated with the label of an element in the net, with an exponent which distinguishes different occurrences of the same element. The dotted line suggests that the unfolding goes on by repeating occurrences of transitions $t_6$ and $t_5$, and of their neighbouring places.

## 3 An asynchronous game on the unfolding

We define a game on $\textsc{unf}(\Sigma)$ in the special case in which $T = V$, that is with full observability of the system. Definitions in this section are adapted from [2].

**Definition 7.** *Let $\rho = (B_\rho, E_\rho, F_\rho, \mu_\rho)$ be a run on* $\textsc{unf}(\Sigma)$ *and $\pi = \gamma_0, \gamma_1, \cdots,$ $\gamma_i, \cdots$ an increasing sequence of B-cuts. The pair $(\rho, \pi)$ is said to be a* play *if:*

– *$\forall e \in E_{nc} \backslash E_\rho$, the net obtained by adding $e$ and its postconditions to $\rho$ is not a run of* $\textsc{unf}(\Sigma)$*;*
– *$\forall e \in E_\rho$ there is a B-cut $\gamma_i \in \pi$ such that $e < \gamma_i$.*

In general, the winning condition for the User is defined by a set of plays. The significant cases to analyze are the ones in which the winning set of plays is determined by a property that we are interested in investigating on the model.

For example, let us suppose that we are interested in knowing if a user is able to force the occurrence of a target transition once. We can model this problem as a game in which the User wins if there is the target in the run of the play.

Another possible goal of a play, as analyzed in [7], is to verify if it is always possible to avoid a certain marking in a controllable system. In this case the User wins those plays in which there are not cuts associated with that marking. Whatever the goal of the game is, a strategy is a function formalizing the behaviour of the User during a play.

When all transitions are observable, the User can determine the current cut in the unfolding on the basis of the transition occurrences observed so far; hence, a strategy can be defined as a map from B-cuts to sets of controllable transitions.

**Definition 8.** *Let $\Gamma$ be the set of B-cuts in* $\textsc{unf}(\Sigma)$*. A* strategy *is a function $\alpha : \Gamma \to 2^{E_c}$ such that for every $\gamma \in \Gamma$ and for every $e \in E_c$, if $e \in \alpha(\gamma)$, then $e$ is enabled in $\gamma$.*

A play is weakly fair with respect to an event $e$ if $e$ is not finally postponed.

**Definition 9.** *Let $(\rho, \pi)$ be a play. An event $e$ is* finally postponed *in $(\rho, \pi)$ iff there is a cut $\gamma_i \in \pi$ in which $e$ is enabled and such that $\forall k \geq i$, $\gamma_k[e\rangle$.*

**Definition 10.** *Let $(\rho, \pi)$ be a play and $\alpha$ be a strategy. An event $e$ is* finally eligible *in $(\rho, \pi)$ by $\alpha$ iff there is a cut $\gamma_i \in \pi$ such that $e \in \alpha(\gamma_i)$ and $\forall k \geq i$, $e \in \alpha(\gamma_k)$*

A play complies with a strategy if all controllable events in the play have been chosen according to the strategy, and no controllable transition is finally postponed and eligible.

**Definition 11.** *Let $\rho = (B_\rho, E_\rho, F_\rho, \mu_\rho)$ be a run in* $\textsc{unf}(\Sigma)$*, $\pi = \gamma_1 \gamma_2, ..., \gamma_i$ be a strictly increasing sequence of B-cuts and $\alpha$ be a strategy. The pair $(\rho, \pi)$ is an* $\alpha-$play *iff:*

1. *$(\rho, \pi)$ is a play;*
2. *For every controllable event $e$ belonging to the $E_\rho$, there must be a B-cut $\gamma_i \in \pi$ such that $e \in \alpha(\gamma_i)$ and $e < \gamma_{i+1} = (\gamma_i \backslash {}^\bullet e) \cup e^\bullet$.*
3. *If $|E_\rho \cap E_c| < \infty$, there is not an event $e \in E_c \cap E_\rho$ finally eligible by $\alpha$ and finally postponed in the play.*

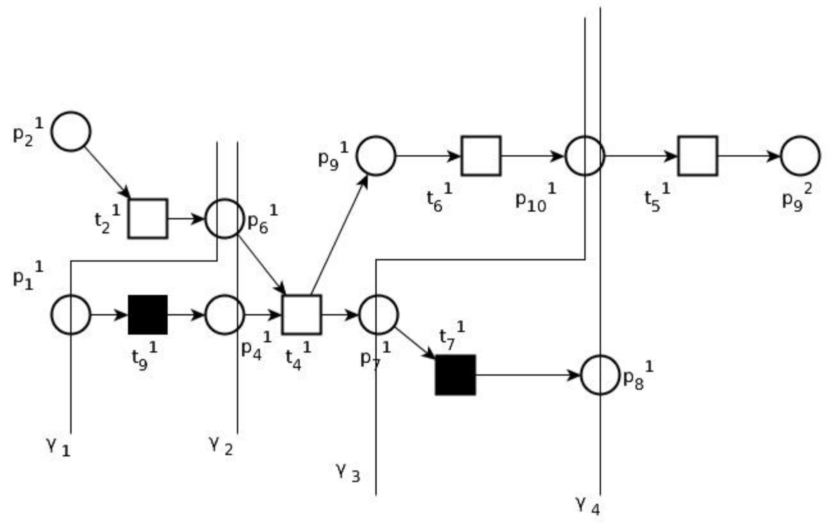

**Fig. 3.** An $\alpha$-play

A strategy $\alpha : \Gamma \rightarrow 2^{E_c}$ is winning iff the User wins all the $\alpha$-plays. In general, if there is a winning strategy, it is not unique.

*Example 2.* The net system shown in Fig. 1 is distributed and controlled, with two locations. Suppose that all transitions are observable. Define a game on its unfolding, shown in Fig. 2, so that the User wins a play if the play contains an occurrence of $t_7$.

By inspecting the net, it is clear that a winning strategy for the User consists in waiting for the Environment to choose between $t_1$ and $t_2$, and then fire, respectively, either $t_8$ or $t_9$. Since the Environment can not postpone its choice forever, and will be forced to eventually fire either $t_3$ or $t_4$, the User will be able to fire $t_7$, and win the game. Formally, the winning strategy can be defined as follows: $\alpha(\{p_1^1, p_6^1\}) = \{t_9^1\}$, $\alpha(\{p_1^1, p_5^1\}) = \{t_8^1\}$, $\alpha(\{p_7^2\}) = \{t_7^2\}$, $\alpha(\{p_7^1, p\}) = \{t_7^1\}$, where $p$ is any occurrence of either $p_9$ or $p_{10}$, $\alpha(\gamma) = \emptyset$ for any other B-cut $\gamma$. In particular, $\alpha(\{p_1^1, p_2^1\}) = \emptyset$, to encode the decision to wait, in the initial cut, for the Environment to choose its first move. Figure 3 shows an $\alpha$-play.

## 4 Controlled reachability

Let $\Sigma = (P, T, F, m_0, K, V)$ be a controlled distributed net system, with $V = T$. The problem of *controlled reachability* consists in determining if the User is able to lead the system to fire a certain transition once, despite the Environment behaviour, starting from the initial marking. This can be analysed through a game on the unfolding: let $t$ be the target transition. We define as winning condition for the User the set of plays $(\rho, \pi)$ in which there is an event $e \in E_\rho$ labelled with $t$. A target transition $t$ is controllably reachable in $\Sigma$ if, and only if, there is a strategy $\alpha$ on UNF($\Sigma$) such that the User wins every $\alpha$-play.

*Example 3.* Example 2 can be seen as a game of controlled reachability. The strategy discussed in the example is a winning strategy for this game.

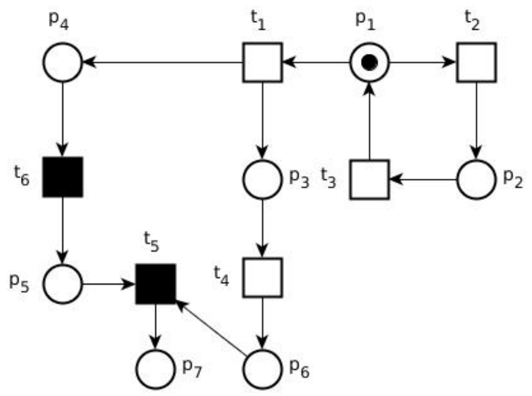

**Fig. 4.** A distributed controlled net system

*Example 4.* The net shown in Fig. 4 is distributed and controlled, with two locations. Consider the game of controlled reachability played on its unfolding, shown in Fig. 5, where the target transition is $t_5$. If the Environment cooperates with the User by eventually choosing $t_1$, then the target is reached. However, the Environment can choose $t_2$ at every cut consisting in an occurrence of $p_1$. The Environment is subject to a weak fairness constraint, but not to a strong fairness constraint. Hence, irrespective of the strategy chosen by the User, an infinite play made of repeated occurrences of the cycle $p_1$, $t_2$, $p_2$, $t_3$, $p_1$ is admissible.

In a general case, given a strategy $\alpha$, there are infinitely many $\alpha-$plays in UNF($\Sigma$), hence the exhaustive exploration of them would take infinite time. We propose an algorithm that, given a controlled distributed net system and a target transition, establishes if there is a winning strategy for the controlled reachability of the target and, if so, computes a winning strategy. The input data are the following.

- A net in which the transitions are enumerated so that all the uncontrollable transitions precede all the controllable ones. If the target is a controllable transition, it must be the first of the controllable transitions.
- The position of the first controllable transition.
- The initial marking $m_0$ of the system.

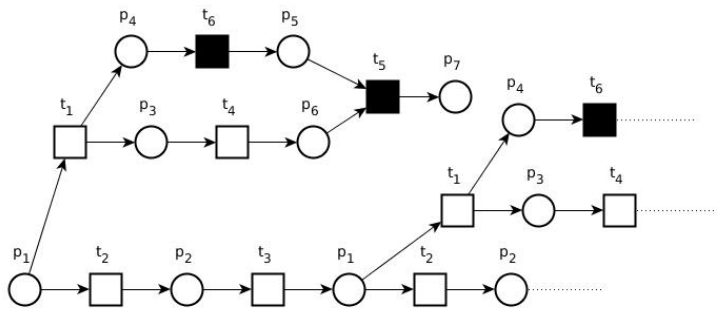

**Fig. 5.** The (prefix of the) unfolding of the net system shown in Fig. 4

**Algorithm 1** Tree exploration

```
function TREE_EXPLORATION($\gamma$, $M$, $e$, $str$, $tree$)
    if $e == target$ then return (true, $str$, $tree$)
    else if $\gamma$ is a deadlock then return (false, $str$, $tree$)
    else if $\mu(\gamma) \in M$ then return EXPLORE_CUT_C($\gamma, M$)
    else if ENAB_N($\gamma$) $\neq \emptyset$ then
        $E$ =ENAB_N($\gamma$)
        repeat
            $e_0$ = EXTRACT($E$)
            $v$, str, tree = TREE_EXPLORATION($\gamma + e_0, [M, \mu(\gamma)], e_0, str, tree$)
            if $v$ == true then
                tree = tree $\cup[\gamma, e_0, \gamma + e_0]$
            end if
        until $E == \emptyset \vee v ==$ false
        return ($v$, $str$, $tree$)
    else
        $E$ =ENAB_C($\gamma$)
  ▷ $E$ cannot be empty, because the case in which $\gamma$ is a deadlock or ENAB_N($\gamma$) = $\emptyset$
  are alternative to this else.
        repeat
            $e_0$ = EXTRACT($E$)
            v, str, tree = TREE_EXPLORATION($\gamma + e_0, [M, \mu(\gamma)], e_0, str, tree$)
            if $v$ == true then
                tree = tree $\cup[\gamma, e_0, \gamma + e_0]$
                choice = $e_0$
            end if
        until $E == \emptyset \vee v ==$ true
        if $v$ ==true then return ($v$, [str, [$\gamma$, choice]], tree)
        else return ($v$, $str$, $tree$)
        end if
    end if
end function
```

– The target transition.

The core of the algorithm is the recursive function TREE_EXPLORATION, which unfolds the net by exploring reachable cuts, and constructs at the same time a tree, representing a part of the unfolding, and a strategy. Each node in the tree consists in a cut of the unfolding, while events label the edges between them. The function takes five input arguments:

1. $\gamma$: the cut that must be analysed;
2. $M$: the list of markings associated to the cuts already analysed in the current run, in the same order in which they have been analysed;
3. $e$: the last event added to the current run, leading to $\gamma$;
4. *str*: the strategy computed so far;
5. *tree*: the tree constructed so far.

It returns three elements:

1. a Boolean value $v$ that is True if there is a winning strategy starting from the cut input cut $\gamma$, False otherwise;
2. the strategy already calculated;
3. the part of the tree already calculated.

The function starts constructing a run by adding uncontrollable events until one of the following cases occurs: (1) a deadlock cut is reached; (2) a cut is reached in which no uncontrollable event is enabled, and some controllable events are enabled; (3) a cut is reached corresponding to a marking which has been already visited in the current run.

In cases (2) and (3), a controllable event is added, and the exploration restarts from the new cut. In case (1), the current run corresponds to a play won by the Environment; hence, the function tries to backtrack along choices among controllable events, if possible.

Consider the net system shown in Fig. 1, and its unfolding (Fig. 2), where the ordering on the set of transitions is given by their indices. Starting from the initial B-cut, the algorithm adds the event $t_1^1$, reaching a cut in which only controllable transitions are enabled. It then adds $t_8^1$, reaching the cut $\{p_3^1, p_5^1\}$, and starts again adding uncontrollable transitions. This run will lead to the target event $t_7^2$, hence it is not necessary to backtrack on controllable events.

The next backtracking step goes back to the initial cut, and starts exploring a new run by adding $t_2^1$; From $\{p_1^1, p_6^1\}$, $t_8^1$ is added, leading to a deadlock. The algorithm backtracks and tries $t_9^1$, and so on.

The rest of this section describes TREE_EXPLORATION in detail.

At every step, given a cut $\gamma$ that must be analysed, the algorithm has three possible behaviours.

1. It can consider $\gamma$ as a leaf of the tree and stop the in depth exploration. The cases in which this happens are discussed later.

2. If there are $k$ uncontrollable events $e_1, ..., e_k$ enabled in $\gamma$ and $\mu(\gamma)$ is unique in the part of the play that has already been generated, then the algorithm extends the play in $k$ ways, each of them adding to the play one of the $k$ events $e_1, ..., e_k$ with the respective following cuts $\gamma_1 = (\gamma \backslash {}^\bullet e_1) \cup e_1^\bullet, ..., \gamma_k = (\gamma \backslash {}^\bullet e_k) \cup e_k^\bullet$. Since there is no concurrency in the same component, the $k$ plays corresponds to different runs. In this case we will call $\gamma$ *uncontrollable cut* or *uncontrollable node.*
3. If in $\gamma$ there are only controllable enabled events, or if there is a $\gamma' < \gamma$ such that $\mu(\gamma) = \mu(\gamma')$ and all the events $e_i \in \{e \in E : \gamma' < e < \gamma\}$ are uncontrollable, then the controllable enabled events are analysed, in the same order in which the respective transitions are enumerated. Such a $\gamma$ is called *controllable cut*, or *controllable node* referring to the tree.

At the first call, the input is:

1. the initial cut $\gamma_0$;
2. an empty list;
3. a fictitious event $i$;
4. an empty list for the strategy;
5. an empty set for the tree.

The final output is the following:

1. a Boolean value that is True if there is a winning strategy, False otherwise.
2. a strategy
3. a set of triples describing part of the plays consistent with this strategy.

To find a winning strategy, we observe that, if at least an uncontrollable event is enabled in the current cut, the User cannot prevent the Environment from firing it and cannot affect the Environment choices. Hence, the User should wait to observe the behaviour of the Environment as much as possible and then, using the information gained through the observations, decide what to do. On the other hand, if cyclic behaviours with only uncontrollable transitions are admitted in the system, if there are some controllable enabled events, the User can win only by using the fact that there is no $\alpha-$play with a finally eligible and finally enabled event. This can be done by letting the enabled controllable events to be chosen by the strategy in all the markings in which they are enabled and that are part of the uncontrollable cyclic behaviour. The moment in which the controllable event will fire is unknown, but we are sure that, if it is finally enabled and eligible, at some point it will.

Following this idea, the algorithm computes a strategy $\alpha$ and some prefixes of $\alpha-$plays in the unfolding: given a cut $\gamma$, if $\gamma$ is uncontrollable, then all the enabled uncontrollable events are considered. This is necessary, because if all of them lead to subtrees in which there is a winning stretegy, then there is also a winning strategy from $\gamma$. If at least one of the subtrees does not necessarily lead to the victory, then there cannot be a winning strategy from $\gamma$, since the User cannot prevent the Environment to fire that event. If $\gamma$ is a controllable cut, then, if there is at least a transition leading to a cut that is root of a subtree with a winning strategy, there is a winning strategy also starting from $\gamma$. Hence, once that such a controllable event has been found, continuing to explore the cut is not necessary.

When the algorithm ends, we obtain a list of prefixes of plays. Each of them is the finest prefix of all the plays starting with it, since for every event there is the cut immediately preceding it and the one immediately following it.

Since we are interested in finding a strategy in a finite amount of time, we need to define some *ending criteria*:

1. the target has been reached in the play;
2. the play has reached a deadlock;
3. there is a cut $\gamma$ in the play such that there is another cut $\gamma'$ such that $\mu(\gamma) = \mu(\gamma')$ and $\gamma' < \gamma$ and $\gamma'$ is the first occurrence of $\mu(\gamma)$ in the same play and there is no event $e \in E_c$ enabled in all the $\gamma_i : \gamma' < \gamma_i < \gamma$.

In this way, the process of generation of the tree is finite. This follows from the finite number of reachable markings and from the way of working of the algorithm. In fact, if a cut corresponding to the same marking of a previous cut in the same play is reached, then, if possible, at the next step, the algorithm adds a controllable event. If a cut corresponding to the same marking as before is reached again in the play and the first two criteria have not been satisfied, the algorithm stops due to the third criterion: a controllable event occurred between the first cut corresponding to the marking and the last one, and since there is not concurrency between controllable events, there cannot be another controllable event enabled in all the cuts between these two.

If we can find a strategy on the prefix, we can extend it on the whole unfolding by redefining the strategy as a function $\alpha : M \to 2^K$, where $M$ is the set of reachable markings in the system. With such a definition, in the unfolding, the strategy is the same in the cuts corresponding to the same marking. This is reasonable, because with full observability, given a cut $\gamma$ and a transition $t$ enabled in $\mu(\gamma)$, there is always only an event $e$ enabled in $\gamma$ and such that $\mu(e) = t$. Moreover, if a transition is controllably reachable starting from a cut corresponding to a certain marking, then it is controllably reachable from all the cuts corresponding to that marking, since the parts of the unfolding starting with two cuts like these are isomorphic.

**Lemma 2.** *Let $\alpha$ be a strategy defined on the markings. Considering a prefix of an $\alpha-$play determined with one of the ending criteria, we can decide if User wins all the $\alpha-$plays starting with such a prefix.*

*Proof.*
1. If the target is reached in the prefix, in all $\alpha-$plays extending it, the target occurs at least once. The User wins all $\alpha-$plays with such a prefix. Of course, this is not enough to state that $\alpha$ is a winning strategy.
2. If there is a deadlock, the run is maximal in the unfolding, hence the prefix is also an $\alpha-$play without the target, hence lost by the User.
3. In the third case, if the prefix follows the strategy $\alpha$, then the play repeating the behaviour of the prefix infinitely is an $\alpha-$play and the target never

occurs. We cannot guarantee that the User will lose all the $\alpha$-plays with such a prefix, but the fact that there is at least one is enough to state that $\alpha$ is not a winning strategy for the User.

**Lemma 3.** *Let $\gamma_1$ and $\gamma_2$ be two cuts in the unfolding such that $\mu(\gamma_1) = \mu(\gamma_2)$ and let $\alpha$ be a strategy computed by Algorithm 1. There are two cases:*

1. $\alpha(\gamma_1) = \alpha(\gamma_2)$;
2. $\alpha(\gamma_1) = e \in E_c \wedge \alpha(\gamma_2) = \emptyset$.

*In the second case, if $\alpha$ is a winning strategy, the strategy associating to $\gamma_2$ the same choices as in $\gamma_1$ is also winning.*

If the algorithm states the existence of a winning strategy, the strategy $\alpha$ exhibited by the algorithm is singular. In fact, the strategy is updated if all the leaves of the subtree having the cut below a controllable node as root are cuts following the target. When the algorithm finds such a subtree, it adds to the strategy the controllable cut associated with the event labelling the connection between it and the root of the subtree and ends the exploration of the subtrees of the children of the controllable cuts. Given a controllable cut $\gamma$ in the prefix, the enabled events are examined in the same order in which their corresponding transitions are enumerated. Since for all the controllable cuts $\gamma'$ such that $\mu(\gamma) = \mu(\gamma')$ in the prefix, the unfolding starting from $\gamma'$ is the same as the unfolding starting from $\gamma$, if the choice of $e$ is winning starting from $\gamma$, the choice of $e' : \mu(e') = \mu(e)$ must be winning also starting from $\gamma'$. If $e$ is the first winning choice analysed, also $e'$ must be the first. If there was another winning choice $e''$ from $\gamma'$ such that $\mu(e'')$ precedes $\mu(e')$ in the incidence matrix, then also $e^1 : \gamma[e^1\rangle \wedge \mu(e^1) = \mu(e'')$ should be a winning choice in $\gamma$ and should have been analysed before $e$. Hence, fixed every reachable marking $m$, for every $\gamma_i \in \{\gamma : \mu(\gamma) = m\}$ controllable in the prefix, an event corresponding to the same transition is chosen.

It is possible that both a controllable cut and an uncontrollable one correspond to the same marking due to the uncontrollable cyclic behaviour. In this case, if the strategy is winning, then it is updated in this way: starting from the first cut corresponding to the same marking as the controllable one and for all the cuts between them, the strategy associates the same choice made for the controllable cut. This does not prevent a winning strategy to be singular: let us assume to have four cuts $\gamma_1, \gamma_1', \gamma_2, \gamma_2'$ such that $\mu(\gamma_1) = \mu(\gamma_1') \wedge \gamma_1 < \gamma_1'$ and $\mu(\gamma_2) = \mu(\gamma_2') \wedge \gamma_2 < \gamma_2'$, such that all the events $e \in E : \gamma_1 < e < \gamma_1' \wedge \gamma_2 < e < \gamma_2'$ are uncontrollable and are not the target. Let us also assume that there are two cuts $\gamma_3, \gamma_4$ such that $\mu(\gamma_3) = \mu(\gamma_4)$ and $\gamma_1 < \gamma_3 < \gamma_1'$ and $\gamma_2 < \gamma_4 < \gamma_2'$. If the algorithm states that there is a winning strategy, the construction of it impose that $\alpha(\mu(\gamma_1)) \subseteq \alpha(\mu(\gamma_3))$ and $\alpha(\mu(\gamma_2)) \subseteq \alpha(\mu(\gamma_4))$. Specifically $\alpha(\mu(\gamma_4)) = \alpha(\mu(\gamma_3)) = \alpha(\mu(\gamma_1)) \cup \alpha(\mu(\gamma_2))$. If the hypothesis is false and the strategy is not singular, it must be $t_1 = \alpha(\mu(\gamma_1)) \neq \alpha(\mu(\gamma_2)) = t_2$. Without loss of generality we can assume $t_1 < t_2$. Since the algorithm chooses the first transition that leads to the victory of the User, this means that $t_1$ is not winning in $\mu(\gamma_2)$, but the User cannot guarantee that in $\mu(\gamma_3)$ the system will not arrive in $\mu(\gamma_2)$, since $t_1$ is concurrent with the uncontrollable cycle. In such a situation the strategy would not be winning and the algorithm would recognize it, because there would be a path leading to $\gamma_2'$ from $\gamma_4$.

**Algorithm 2** Full strategy

```
v, str, tree = TREE_EXPLORATION($\gamma_0$, [], $i$, [], [])
if v == True then
    str = CUTS_TO_MARKINGS(str)
    str = COMPLETE_STRATEGY(str, tree)
end if
```

In Algorithm 2 there are calls to some auxiliary functions:

- CUTS_TO_MARKINGS(str) takes the strategy defined on the cuts as input and returns a strategy defined on the markings. The Lemma 3 guarantees that this will not change the value of $v$.
- COMPLETE_STRATEGY(str, tree) considers all the runs in the generated tree starting from the root of the prefix and checks if there are cuts corresponding to the same marking in the same run such that all the events between the two of them are uncontrollable. If there are, the strategy is completed adding the choice made in the repeated marking to all the marking associated with cuts in between. This is crucial in order to have a winning strategy, because if this happens, it means that the User can win only if the play with only uncontrollable events between cuts corresponding to the same marking is not admitted, and this happens if a controllable event is finally enabled and eligible, hence this event must be chosen in all the markings of the uncontrollable cycle.

In Algorithm 1 the functions that are used, but not described are:

- ENAB_N($\gamma$) is a function that given a cut $\gamma$, returns the uncontrollable events enabled in that cut;
- analogously, ENAB_C($\gamma$) returns the list of controllable events enabled in $\gamma$.
- Given the list of events (controllable or not) that we are interested to explore, EXTRACT(E) returns the first one and deletes that event from the list $E$.

We wish to show that the algorithm provides a sufficient condition to find a winning strategy. In order to do this, we need to show that if the algorithm states that there is a winning strategy, then there is one, and the strategy proposed by the algorithm is winning.

Let us observe that if the algorithm finds a winning strategy, all the prefixes of plays in the prefix reach the target and are consistent with the strategy. All the plays in the list are consistent with the strategy, because every time that the algorithm analyzes a controllable node, it explores the subtrees following each of the controllable enabled transitions, stopping when it finds a transition leading to a subtree in which there is a winning strategy. When this happens,

**Algorithm 3** Controllable cuts due to a repeated marking

**Input:** the cut $\gamma$ that must be analyzed and an ordered list $M$ of the markings corresponding to the cuts visited in the run before $\gamma$.

```
function EXPLORE__CUT__C($\gamma, M$)
    if F($\gamma, M$))= $\emptyset$ then return (false, str, tree)
    else
        $E$ = F($\gamma, M$)
        repeat
            $e_0$ = EXTRACT($E$)
            v, str, tree = TREE__EXPLORATION($\gamma + e_0, [M, \mu(\gamma)], e_0$)
            if $v$ == true then
                tree = tree $\cup[\gamma, e_0, \gamma + e_0]$
                choice = $e_0$
            end if
        until $E == \emptyset \vee v ==$ true
        if $v$ ==true then return ($v$, [str, [$\gamma$, choice]], tree)
        else return ($v$, $str$, $tree$)
        end if
    end if
end function
```

**Algorithm 4** Controllable events that can be forced to fire

**Input:** the cut $\gamma$ that must be analyzed and an ordered list $M$ of the markings corresponding to the cuts visited in the run before $\gamma$.

**Output:** list of events that have been enabled from the cut associated with the first repetition of the marking $\mu(\gamma)$ to $\gamma$.

```
function F($\gamma, M$)
    i = 0
    while $M[i] \neq m(\gamma)$ do
        i = i+1
    end while
    $E = []$
    for all $e \in$ENAB__C($\gamma$) do
        if $\mu(e)$ enabled in $m$ $\forall m \in M[i : len(M)]$ then
            $E = [E, e]$
        end if
    end for
return $E$
end function
```

the subtree is connected with the controllable node through the controllable transition and the strategy is updated choosing that controllable transition in the cut corresponding to the controllable node. In this way, at every step, all the parts of runs in the prefix constructed until that moment are consistent with the strategy updated until that moment. Given a controllable node that has been analyzed, it can happen that there are no controllable enabled transitions leading to a subtree in which there is a winning strategy; in this case the part of the prefix already generated cannot be connected to the initial cut in the unfolding. Also the part of strategy already computed is not deleted, but, if there is a winning strategy, it cannot depend on the strategy calculated on the disconnected parts of the tree. If the algorithm finds a winning strategy and a disconnected part was found, since there are controllable nodes only if taking a controllable transition is needed to win, then there is another controllable node in the prefix, closer to the root from which the subtree has a winning strategy.

All maximal runs in the prefix contain the target. If a run ends without the target, then the strategy allowing that run is not winning and must be changed. If it cannot be changed, then the algorithm will not state that there is a winning strategy, hence there must be a controllable node in which the decision previously taken can be changed. When another possible choice is analysed, all parts of runs depending on the previous one are deleted. Hence all the remained runs contain the target.

If the algorithm finds a winning strategy, all the plays in the unfolding consistent with this strategy can be considered as equivalent to an extension of a play in the prefix. Let us first consider the case without uncontrollable cyclic behaviours of the system. In every run, the uncontrollable transitions can be ordered in just one way, since there is no concurrency inside a component. The strategy $\alpha$ constructed by the algorithm chooses a controllable transition only if there are not uncontrollable enabled ones, hence the order in which a given sequence of transitions $t_1, ..., t_n$ will fire in an $\alpha-$play is unique and completely determined. Given an $\alpha-$play, there must be a prefix of its run in the unfolding, because all the uncontrollable transitions are considered in all the uncontrollable cuts of the prefix and the strategy is singular, hence the controllable choices must be the same of the ones considered in the prefix. This is enough to state that the play is won by the User, because in the common prefix of the run there is the target transition. If there are uncontrollable cyclic behaviours, such that there is a controllable enabled transition leading to the target, and such that it is concurrent with all the ones in the uncontrollable cycle, then there is more variety in the possible $\alpha-$plays, because the strategy is defined on markings in which uncontrollable cuts are enabled. Anyway, if an $\alpha$-play has a prefix with the same events of one of the prefixes produced by the algorithm, then it is won by the User, regardless of the order of the cuts in the play. Some of the $\alpha-$plays have a longer uncontrollable part, because if a transition is always enabled and eligible, there must be a certain point in which it will fire, but the precise point is unknown. However, since we complete every cycle once and from every uncontrollable cut all the possible uncontrollable extensions are explored, and since the part of the unfolding starting from a given cut is isomorphic to the part of the unfolding starting from every cut corresponding to the same marking, the uncontrollable sequence of the $\alpha-$play can be divided in parts such that an isomorphic one has been considered by the algorithm.

Based on the previous observations, if the algorithm finds a winning strategy, the proposed strategy is winning in the unfolding.

## 5 Other approaches to asynchronous games

The general notion of asynchronous game presented in this paper was defined in [2], where it was applied to a problem of controlled liveness, under the hypothesis of full observability.

An asynchronous game on Petri nets was also defined by Finkbeiner and Olderog in [7]. This game is developed for Place/Transition nets, and is played on their unfoldings. The players are represented by tokens, moving on the places of the unfolding, divided into two teams: system and environment. System players have an equivalent function as the User in the game defined by [2] and used in this paper. Their objective is to guarantee a safety property. For example, the aim might be to avoid reaching a certain place. The places are divided into system places, where system players can move, and environment places, reserved to environment players. The strategy is defined on each place and states which is the next place where a token has to move. Places are the central elements in this game, in contrast to the game in [2] where the focus is on transitions.

The information available to the players is another difference. In [2], and in our approach, this information consists in observed transitions. If a transition is observable, then the User knows whether the transition occurred or not. If a transition is unobservable, then there is no way for the User to know whether it occurred, unless he can infer this from observations. In the game described by Finkbeiner and Olderog, the players communicate by means of synchronizations. Participating in the same transition, they acquire the knowledge of the past of the players that take part to the synchronization. One or the other approach may be more convenient for the User/System depending on the structure of the system and on the property that has to be verified.

In [7] a strategy for the User is defined on the unfolding of the net system, and must be fair, i.e. if a System player can move, then it must do it. This requirement avoids the trivial case in which safety is verified just because the players refuse to move. In the game in [2] for a similar reason, progress is granted by the environment. In that case the User wishes to force a transition to occur infinitely often. In almost every case this goal would be impossible to reach if the environment does not fire any of its transitions.

Under the restricted hypothesis of just one environment player (and an arbitrary number of system players), and complete information, Bernd Finkbeiner, Manuel Gieseking and Ernst-Rüdiger Olderog developed a tool, presented in [6], finding a strategy for the game as defined in [7]. The tool translates the game to a standard two-players game over finite graphs.

A different approach for the verification of properties through asynchronous games was developed by several authors, among which Glynn Winskel ([10]) and Julian Gutierrez ([9]). The game is defined on event structures. An event structure is a set of events in which a partial order and a conflict relation are defined. Event structures are in relation with Petri nets used in this paper: given an occurrence net, there is always an event structure with the same partial order and the same conflict relations of the events in the occurrence net. The opposite is also true: constructing an occurrence net in which the partial order between events is the same as in a event structure is always possible. However, this occurrence net is not always equivalent to the unfolding of a Petri net. As in the game in [7], the two players have limited knowledge of what happens in the system. When two or more events cause the occurrence of another one, there is an exchange of information that can be used by the strategy. Gutierrez shows that the game can be applied to the bisimulation problem and model-checking.

## 6 Conclusions

In this work we have presented an algorithm for the computation of a strategy for a reachability problem in a distributed net system with full observability. The algorithm has been implemented in Python. The next step consists in studying the complexity of the algorithm and testing it on different nets.

We plan to apply the general idea of the game to different problems and to define proper algorithms to find winning strategies in each case.

On the theoretical side, we will consider the case of partial observability. In this extended case the definition of a strategy needs to be redefined, because in general, if only some transitions are observable, the current marking of the system, and the current cut on the unfolding, are unknown. Moreover, while with full observability the information given by the observations on the system or on the unfolding is the same, with partial observability a strategy on the unfolding may be able to distinguish two different evolutions of the system, even if the observed transitions are the same. This happens because in the structure of the unfolding there is a track of the different stories of the system, hence being able to distinguish two events corresponding to the same transition would mean being able to reconstruct also the unobservable story of the system up to every observed event.

Another future generalization is increasing the number of players. It would be interesting to analyse a game in which more than two players try to reach a goal, eventually in a cooperative or in a competitive way, and considering a game in which concurrency within a component is allowed.

## Acknowledgments

This work has been partially supported by MIUR. The authors thank the anonymous referees for their useful comments.

## References

1. Bernardinello, L., Kilinç, G., Pomello, L.: Weak observable liveness and infinite games on finite graphs. In: van der Aalst, W.M.P., Best, E. (eds.) Application and Theory of Petri Nets and Concurrency - 38th International Conference, PETRI NETS 2017, Zaragoza, Spain, June 25-30, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10258, pp. 181–199. Springer (2017). https://doi.org/10.1007/978-3-319-57861-3, https://doi.org/10.1007/978-3-319-57861-3
2. Bernardinello, L., Pomello, L., Puerto Aubel, A., Villa, A.: Checking weak observable liveness on unfoldings through asynchronous games. In: Moldt, D., Kindler, E., Rölke, H. (eds.) Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'18), Bratislava, Slovakia, June 24-29, 2018. CEUR Workshop Proceedings, vol. 2138, pp. 15–34. CEUR-WS.org (2018), http://ceur-ws.org/Vol-2138/paper1.pdf
3. Best, E., Darondeau, P.: Petri net distributability. In: Clarke, E.M., Virbitskaite, I., Voronkov, A. (eds.) Perspectives of Systems Informatics - 8th International Andrei Ershov Memorial Conference, PSI 2011, Novosibirsk, Russia, June 27-July 1, 2011, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7162, pp. 1–18. Springer (2011). https://doi.org/10.1007/978-3-642-29709-0, https://doi.org/10.1007/978-3-642-29709-0
4. Desel, J., Kilinç, G.: Observable liveness of Petri nets. Acta Inf. **52**(2-3), 153–174 (2015). https://doi.org/10.1007/s00236-015-0218-1, https://doi.org/10.1007/s00236-015-0218-1
5. Engelfriet, J.: Branching processes of Petri nets. Acta Inf. **28**(6), 575–591 (1991). https://doi.org/10.1007/BF01463946, https://doi.org/10.1007/BF01463946
6. Finkbeiner, B., Gieseking, M., Olderog, E.: Adam: Causality-based synthesis of distributed systems. In: Kroening, D., Pasareanu, C.S. (eds.) Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9206, pp. 433–439. Springer (2015). https://doi.org/10.1007/978-3-319-21690-4_25, https://doi.org/10.1007/978-3-319-21690-4_25
7. Finkbeiner, B., Olderog, E.: Petri games: Synthesis of distributed systems with causal memory. Inf. Comput. **253**, 181–203 (2017). https://doi.org/10.1016/j.ic.2016.07.006, https://doi.org/10.1016/j.ic.2016.07.006
8. van Glabbeek, R.J., Goltz, U., Schicke-Uffmann, J.: On characterising distributability. Logical Methods in Computer Science **9**(3) (2013). https://doi.org/10.2168/LMCS-9(3:17)2013, https://doi.org/10.2168/LMCS-9(3:17)2013
9. Gutierrez, J.: Concurrent logic games on partial orders. In: Beklemishev, L.D., de Queiroz, R.J.G.B. (eds.) Logic, Language, Information and Computation - 18th International Workshop, WoLLIC 2011, Philadelphia, PA, USA, May 18-20, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6642, pp. 146–160. Springer (2011). https://doi.org/10.1007/978-3-642-20920-8, https://doi.org/10.1007/978-3-642-20920-8
10. Winskel, G.: Distributed games and strategies. CoRR **abs/1607.03760** (2016), http://arxiv.org/abs/1607.03760