A Two Player Asynchronous Game with Privacy Constraints on Petri Nets (short paper)

Federica Adobbati[1,2,*,†], Luca Bernardinello[2,*,†] and Lucia Pomello[2,*,†]

[1]National Institute of Oceanography and Applied Geophysics - OGS, Trieste, Italia
[2]Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca

Abstract

In this short presentation we propose a formal framework based on a two-player game on Petri nets. We consider multi-agent systems in which a component, the controller, needs to guarantee a certain liveness property on the system behaviour, while keeping secret some of its actions to the other system components, and we discuss the basis for a uniform solution of this problem.

Keywords

Two-player game; non-interference; reveals relations; Petri nets.

1. Introduction

We propose a framework for the formal study of concurrent multi-agent systems, in which a component (the *controller*) needs to keep certain services active, and at the same time needs to keep secret some of its actions.

These two problems are often studied separately; a common way to check whether the controller is able to guarantee a certain property consists in modelling the problem as a two-player game, and verifying the existence of a winning strategy for the controller, namely a selection of moves such that the property is always guaranteed in the system, regardless of the behavior of the remaining part of the system. Typical goals for the controller consist in avoiding a set of states [1], performing a given action [2], or satisfying a given temporal logic formula [3].

To check whether a system satisfies some privacy requirements, several notions of non-interference and opacity have been introduced [4, 5, 6], aiming to verify that an intruder cannot be able to recover secret information by just observing the visible behaviour of the system.

Our goal is to put the basis for a unified analysis of these two problems. We define a multi-agent system as a 1-safe Petri net, where transitions are partitioned into controllable and uncontrollable, and a subset of controllable transitions needs to remain secret. We define the privacy requirement through the *reveals* relation [7], that was used in [6] to provide a non-interference notion, and we model the interaction of the controller with the rest of the system,

ICTCS'24: Italian Conference on Theoretical Computer Science, September 11–13, 2024, Torino, Italy
*Corresponding authors.
†These authors contributed equally.
fe.adobbati@gmail.com (F. Adobbati); luca.bernaridinello@unimib.it (L. Bernardinello); lucia.pomello@unimib.it (L. Pomello)

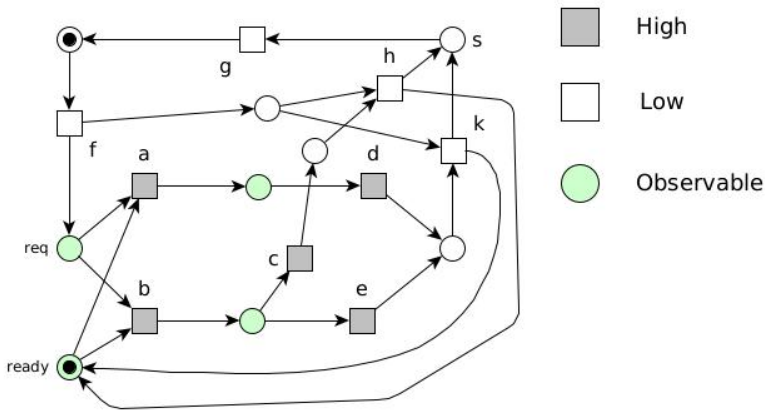

Figure 1: In this Petri net, the set of controllable transitions coincides with the set of high transitions (in grey). The goal of the controller is to reach place s in case a request is done (place *req*), while keeping secret the high transitions.

that we will call *environment*, as a two player game, where players can move asynchronously. We assume that the controller can base its strategy on the partial observations of the current state of the system, hence it observes a subset of places, whereas the environment can deduce information by observing the occurrences of some transitions. We discuss the interactions between the liveness and the non-interference goals, and different notions of winning strategies based on different ability assumptions for the environment.

This work builds on [8], where we define an algorithm to check the existence of a winning strategy when the goal of the controller is a LTL formula and the system is modelled as a 1-safe net, and on [9], where we propose an algorithm to verify the existence of *reveals* relations for a subclass of bounded P/T systems.

The main ideas of our approach are illustrated in the following, extremely simplified, example. Consider the Petri net in Fig. 1, where transitions in grey belong to the controller. Transition f models the request of a service, which can be supplied in two distinct ways (either a or b). The controller must guarantee that every request is sooner or later satisfied, so that the environment will reach place s; at the same time, the environment should not be able to deduce whether a or b was chosen. Suppose that the environment does not directly observe the grey transitions. If transition c fires, then the environment can deduce that b was chosen. Hence, a winning strategy for the controller consists in never firing c.

2. Basic definitions

In the following we assume the basic definitions of Petri net theory such as 1-safe net systems and their unfoldings ([10], [11]).

Let $\Sigma = (P, T, F, m_{in})$ be a 1-safe net system and $\textsc{unf}(\Sigma) = (B, E, F, \mu)$ be its unfolding.

The set T is partitioned into two disjoint subsets, T_c, the set of controllable transitions, and T_{env}, the set of uncontrollable transitions which belong to the environment.

In addition, T can be partitioned into *high* and *low* transitions, denoted T_h and T_l, respectively. We assume $T_h \subseteq T_c$, being T_h the set of controllable transitions which must remain secret, i.e.: the set of transitions whose occurrence should not be deduced by the environment. By consequence, the events in the unfolding will be partitioned into E_c, controllable, and E_{env}, uncontrollable events, according to their labels; moreover, E_c will contain the set E_h of occurrence of secret transitions, the high events.

A *run* is a prefix $\rho = (B_\rho, E_\rho, F_\rho, \mu_\rho)$ of the unfolding of Σ, describing a particular history in which conflicts have been solved, i.e.: the underlying net (B_ρ, E_ρ, F_ρ) is conflict-free. A *cut* in ρ is a maximal set of pairwise concurrent elements of B_ρ.

Let R be a set of runs in $\text{UNF}(\Sigma)$ and $t_1, t_2 \in T$; we say t_1 *reveals* t_2 in R, denoted $t_1 \triangleright_R t_2$, if each run in R which contains an occurrence of t_1 also contains at least one occurrence of t_2.

Formally, $t_1 \triangleright_R t_2$ iff 1) $\forall \rho \in R\ t_1 \in \mu_\rho(E_\rho) \Rightarrow t_2 \in \mu_\rho(E_\rho)$, and 2) there is a run $\rho \in R$ such that: $t_1 \in \mu_\rho(E_\rho)$.

We will say that the set of runs R satisfies the non-interference property if and only if there is no pair of transitions $t_l \in T_l$, $t_h \in T_h$ such that t_l reveals t_h in R.

3. The game

In this section we define a two-player asynchronous game.

Let $\Sigma = (P, T, F, m_{in})$ be a 1-safe net, T_c the set of controllable transitions, $T_{env} = T \setminus T_c$; $\text{UNF}(\Sigma) = (B, E, F, \mu)$ its unfolding, with $E_c \subset E$ the set of controllable events. A *strategy* is a function that returns a subset of controllable transitions, based on the current controller observation. In this work we assume that the controller can observe a subset of places in the system, and that has no memory, therefore its observations are subsets of markings.

The game we are defining is asynchronous, in the sense that players do not take turns; any player can decide to fire one of its transitions, when the transition is enabled, and we assume that the system is distributed in space. This implies that even if a place is observable by the controller, in general the controller cannot base its strategy on the information that the place is marked or unmarked, if a non controllable transition is enabled, and might change the marking of the place. Hence we introduce the concept of *stable part* of a marking. Let m be a reachable marking. Its stable part is defined as the set of places in m whose tokens cannot be consumed by any sequence of uncontrollable transitions enabled in m. We assume that the controller can base his decisions only upon the observable and stable parts of markings. This is motivated by the fact that even if the value of a certain local state is not hidden to the controller, its information may arrive to the controller with a certain delay. By definition, if a place is not in the stable part of a marking, its value may change due to a transition that is out of the control of the controller, therefore the information about it may arrive outdated to the controller, that cannot count on it. If we denote with M the set of reachable markings and with OBS the function that for each marking associates its stable and observable part, we can formally define a strategy (abusing the notation) as $\alpha : \text{OBS}(M) \to 2^{T_c}$.

We assume that the set of the environment transitions $T_{env} \subseteq T$ has a progress constraint, that is, if a transition $t \in T_{env}$ is constantly enabled, it will eventually fire. Let $E_{env} \subseteq E$ be

the set of events corresponding to the occurrence of transitions in T_{env}. A *play* is a run ρ of the unfolding, maximal with respect to E_{env}, namely ρ' obtained by extending ρ with an $e \in E_{env}$ cannot be a run.

A play ρ is *consistent with a strategy* α iff (1) for each $e \in \rho \cap E_c$ there is a cut γ such that $\mu(\gamma) = m$ and $e \in \alpha(\text{OBS}(m))$, i.e. each controllable event was allowed by α in the play; (2) no event belonging to the controller is finally postponed, namely there is no event $e \in E_c$ enabled in ρ but not present in ρ, which is constantly selected by α.

Given a play, to decide whether the controller wins it, we need to formally define its goal. As stated in the introduction, we consider goals composed by two parts that need to be verified at the same time. The first is the ability of the controller to eventually guarantee a certain service, whenever it is required. Let *req* be the place denoting that the service has been requested and *s* the place denoting the service. We express this part of the goal with an LTL formula, specifically $G(req \rightarrow Fs)$, where G and F are, respectively, the *globally* and the *eventually* temporal operators. In words, whenever a request is made, it will eventually be satisfied. The second is the secrecy property, namely let $T_h \subseteq T_c$ be the subset of high transitions and $T_l = T \setminus T_h$ the subset of low transitions. The environment should not be able to discover the occurrence of a high transition through the observation of low transitions. We model this property through *reveals*, namely we require that there is no pair t_h, t_l, in the considered set of runs R, such that $t_l \triangleright_R t_h$. In the literature [7, 9], reveals is defined with respect to the maximal runs of the unfolding; however, in our context this seems not the best option, since the set of plays in the game (hence the runs which can actually occur) do not necessarily coincide with the set of maximal runs. Here we propose some alternative definitions of set of plays R on which computing reveals can be significant in our context, based on different assumptions on the knowledge and the ability of the environment. These options are ordered from the larger to more restrictive R.

1. R could be the set of all the runs on the system maximal with respect to the set E_{env}. This is the set of all the runs allowed by the game. This option works if the environment is not aware of the goal of the controller; if this is the case, it cannot restrict the behaviours of the system by computing a set of possible strategies, but has to consider that any possible run allowed in the game can occur.
2. R is the set of all the runs on the system, maximal with respect to the set E_{env} and where the LTL formula is satisfied. In this case we assume that the environment is aware of the first goal of the controller, but it is still not able to compute a strategy that the controller could take.
3. R is a set of runs such that for each $\rho \in R$ there is a strategy α such that α is winning with respect to the LTL goal, and ρ is a play consistent with α. In this case we consider a partially rational environment, that is able to restrict the behaviours of the controller by knowing its goal and by computing which behaviours the controller could have in order to be sure to reach it.
4. R is a set of runs such that for each $\rho \in R$ there is a strategy α as follows:
    - α is winning for the LTL goal;

- for each $t_l \in T_l$ that can fire by following α, there is no $t_h \in T_h$ that fires by following α such that $t_l \triangleright_R t_h$ in the runs allowed by α.

This is the case in which the environment is not only able to compute a strategy for liveness, but also to compute which strategies should be avoided in order to let it deduce information about occurrences of high transitions.

When we pursue the last option, we only need to check whether R is empty; if it is not, the controller has a winning strategy. In all the other cases we need to compute the reveals relation on the set R.

4. Conclusion and prospects

We have described a framework, based on Petri nets, in which it is possible to combine two types of problems, that we tackled in recent years: exploiting a kind of asynchronous game in order to prove certain liveness properties, and studying a relation between events, useful in formalizing non-interference in distributed systems. The next steps we plan to carry on relate to the search for algorithms to decide whether a winning strategy for the controller exists. To this aim, we plan to start from the algorithms that we developed for solving each of these problems [9, 8], by adapting them to work in the combined case. This step is not trivial, since a strategy tuned for one goal among liveness and non-interference may not satisfy the other. Longer term goals include considering generalizations of the $reveals$ relation, using an $excludes$ relation [6] in order to define more flexible notions of non-interference, and generalizations of the overall scheme, by allowing more than two agents.

References

[1] B. Finkbeiner, E.-R. Olderog, Petri games: Synthesis of distributed systems with causal memory, Information and Computation 253 (2017) 181–203.

[2] L. Bernardinello, G. Kılınç, L. Pomello, Weak observable liveness and infinite games on finite graphs, in: Application and Theory of Petri Nets and Concurrency: 38th International Conference, PETRI NETS 2017, Zaragoza, Spain, June 25–30, 2017, Proceedings 38, Springer, 2017, pp. 181–199.

[3] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, Journal of the ACM (JACM) 49 (2002) 672–713.

[4] N. Busi, R. Gorrieri, A survey on non-interference with Petri nets, Lectures on Concurrency and Petri Nets: Advances in Petri Nets 4 (2004) 328–344.

[5] J. W. Bryans, M. Koutny, P. Y. Ryan, Modelling opacity using Petri nets, Electronic Notes in Theoretical Computer Science 121 (2005) 101–115.

[6] L. Bernardinello, G. Kılınç, L. Pomello, Non-interference notions based on reveals and excludes relations for Petri nets, Transactions on Petri Nets and Other Models of Concurrency XI (2016) 49–70.

[7] S. Haar, Unfold and cover: Qualitative diagnosability for Petri nets, in: 2007 46th IEEE Conference on Decision and Control, IEEE, 2007, pp. 1886–1891.

[8] F. Adobbati, L. Bernardinello, L. Pomello, Looking for winning strategies in two-player games on Petri nets with partial observability, arXiv preprint arXiv:2204.01603 (2022).
[9] F. Adobbati, L. Bernardinello, G. Kılınç Soylu, L. Pomello, Computing a parametric reveals relation for bounded equal-conflict Petri nets, in: Transactions on Petri Nets and Other Models of Concurrency XVII, Springer, 2023, pp. 54–83.
[10] T. Murata, Petri nets: Properties, analysis and applications, Proceedings of the IEEE 77 (1989) 541–580. doi:10.1109/5.24143.
[11] J. Engelfriet, Branching processes of Petri nets, Acta Inf. 28 (1991) 575–591. URL: https://doi.org/10.1007/BF01463946. doi:10.1007/BF01463946.