Asynchronous Games on Petri Nets and Partial Orders

Communication

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

1 Introduction

We discuss an asynchronous game on Petri nets, modelling the interaction between one or more users and an environment in a partially controllable and partially observable distributed system. By *partially controllable* we mean that each user controls a subset of the transitions, while some transitions are part of the environment, and their occurrences can not be directly prevented by the users; by *partially observable* we mean that each user can observe directly only some elements of the system. We assume that each choice in the behaviour is local, namely under the control of either one user or the environment.

Through the game, we investigate properties of the system, checking if the user, controlling just a part of it, is able to impose a specified behavior on the system. The decisions of the user are formalized with a *strategy*, a function associating a subset of controllable choices to the system's configurations.

We introduce the game in a very general setting, so that, defining proper strategies, we can adapt the same model for the verification of different properties by imposing different conditions on the systems.

The idea behind this definition was firstly addressed to solve the problem of *weak observable liveness*, as defined in [5]. The first solution proposed was a synchronous game on the case graph ([3]). This has disadvantages: it hides the concurrency of the system and suffers from state explosion. To overcome these issues, an asynchronous game played on the unfolding was developed in [4]. This game was used to check properties of systems respecting some specified restrictions in [4] and [2].

Games are well suitable models to analyse interactions between autonomuos agents. For this reason, in the last decades, many different models have been developed. We provide here a non-exhaustive historical overview of the works dealing with models of distributed systems and asynchronous games.

In 1999, Samson Abramsky and Paul-André Melliès in [1] defined a game that they called *concurrent*. In their work, the game is a domain of positions on partial orders, a strategy is a closure operator on it, and the idea is applied

* Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

to semantics of Linear Logic. In 2007, Melliès and Samuel Mimram developed this idea in [9], defining an asynchronous game played on a partial order of events, where strategies are sets of plays. In the last years other authors, among which Julian Gutierrez ([8]) and Glynn Winskel ([11]) generalized these previous contributions defining asynchronous games on event structures.

Closer to the game we are going to present, Bernd Finkbeiner and Ernst-Rüdiger Olderog developed in [7] a game on Petri nets, where plays are runs on the unfolding, players are tokens on the net and exchange information through synchronizations. The aim of their game is the verification of a safety property; specifically, they aim to verify if the so called system players are always able to avoid a target place to be marked in the system. The information is carried by tokens; every token knows only his past until it exchanges information with other tokens through synchronizations.

2 General Definitions

In this section we describe in some detail the general idea of the game, and suggest some specific cases of it. In the following we assume the basic definitions of Petri net theory such as elementary net systems and their unfoldings ([10], [6]).

Let $\Sigma = (P, T, F, m_0)$ be an elementary net system and $\textsc{unf}(\Sigma) = (B, E, F, \mu)$ its unfolding. Let $K \subseteq T$ be the set of controllable transitions, and $E_k \subseteq E$ the set of controllable events in the unfolding (occurrences of controllable transitions).

A *run* is a prefix of $\textsc{unf}(\Sigma)$ $\rho = (B_\rho, E_\rho, F_\rho, \mu_\rho)$ describing a particular history, in which conflicts have been solved. A *configuration* is a set of elements of $\textsc{unf}(\Sigma)$, down-closed (if x belongs to a configuration, then all the elements in its past are contained in the configuration), and conflict-free. By $\mathrm{FRuns}(\Sigma)$ we mean the set of runs which are weakly fair with respect to uncontrollable events.

A play in the game is essentially a run in $\mathrm{FRuns}(\Sigma)$. The run is paired by a sequence of configurations representing global states, as might be observed by some external observer. From one configuration to the next in the sequence, several events occur, corresponding to concurrent or causally dependent moves of the players and of the environment.

Definition 1. *A* play *is a pair (ρ, Π), where ρ is a run in* $\mathrm{FRuns}(\Sigma)$ *and Π is a sequence of configurations $\delta_1 \subset ... \subset \delta_n \subset ...$ with $\delta_i \subset B_\rho \cup E_\rho$ for all i.*

Definition 2. *A* winning condition *is a subset $W \subset \mathrm{FRuns}(\Sigma)$,*

In order to deal with the notion of partial observability, we suppose that a set of *observable configurations* is defined. In a simple case, an observable configuration can be the projection of a configuration onto a given set of observable events, or of observable places, but other notions are possible.

A *strategy* is then defined as a map from the set of observable configurations to the set of controllable events. More precisely, we allow for an equivalence relation on observable configurations, to deal with cases in which two different observable configurations might be indistinguishable by a player, and define a strategy as a map on equivalence classes of it.

The set of all observable configurations on $\text{UNF}(\Sigma)$ is denoted by $OC(\Sigma)$.

Definition 3. *Let $\sim \subseteq OC(\Sigma) \times OC(\Sigma)$ be an equivalence relation. A strategy is a function $\alpha : OC(\Sigma)/\sim \to 2^K$.*

A play complies with a strategy if every controllable event occurring in the play is the result of applying the strategy to an observable configuration visited during the play, and no enabled choice of the strategy is indefinitely postponed.

Definition 4. *A play (ρ, Π) complies with α if*

- *for all $e \in E_K \cap \rho$ there is $\delta_i \in \Pi$ such that $\lambda(e) \in \alpha(\text{OBS}(\delta_i))$ and $e \in \delta_{i+1}$*
- *for all $e \in E_K$, e is not finally eligible and finally postponed in (ρ, Π).*

A strategy is winning if all plays complying with it are elements of the winning condition.

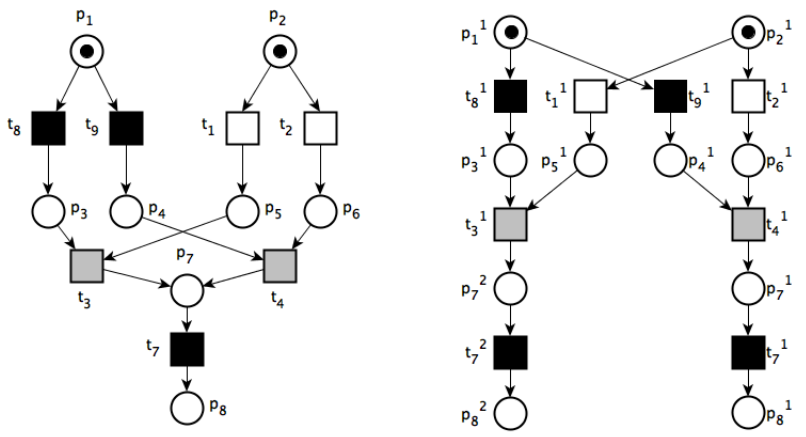

Fig. 1. Black transitions are controllable by the user, white ones are observable, and grey ones unobservable. The transitions' observability on the system (on the left) is inherited by the unfolding (on the right).

Example 1. Suppose that the goal of the user is to fire once a target transition. Coherently, we define the winning condition for the user as the set of runs containing the target. We assume that the user observes a subset of transitions of the system, hence, letting ρ be a run on $\text{UNF}(\Sigma)$ and δ one of its prefixes, $\text{OBS}(\delta)$ is the set of events in δ such that the associated transition is observable in the system. Finally, we define the relation $\sim$ as the identity relation.

Let us consider the system in Figure 1, and let us assume that t_7 is the target transition. A winning strategy for the user is: $\alpha(\{t_1^1\}) = \{t_8\}$, $\alpha(\{t_2^1\}) = \{t_9\}$, $\alpha(\{t_1^1, t_8^1\}) = \alpha(\{t_2^1, t_9^1\}) = \{t_7\}$. For all other observable configurations, the strategy should decide for the empty set. Notice that before every occurrence of t_7 there must be an occurrence of t_3 or t_4, but they are not in any observable configuration because they are unobservable by the user. In this situation, observability does not change the user's chances of winning, but this is not always the case. For example, if in the system in Figure 1 the transitions t_1 and t_2 had been unobservable, the user would not have had a winning strategy. Actually, the user could not distinguish between the situations in which t_1 or t_2 already fired and the one in which nothing happened, but this is crucial in order to take the winning decisions, since the two reachable markings $\{p_3, p_6\}$ and $\{p_4, p_5\}$ are deadlocks for this system.

3 Contributions

In [4] and [2], the game described in Section 2, where some constraints are imposed, is used for the analysis of two different properties. In [4] the authors analyse the *weak observable liveness* of a target transition in systems with full observability. They prove the equivalence between weak observable liveness of a target transition on the sequences of the system, and the existence of a winning strategy in a game on the unfolding, where the goal of the user is having for every play infinite occurrences of the target. Studying the game on the unfolding rather than analysing the property directly on the system can be useful in developing methods and techniques for verification. Keeping track of the different evolutions of the system can help in understanding when all the different behaviours have been explored. In addition, the concurrent structure is preserved, hopefully reducing the complexity of the method; indeed, complexity is the main drawback of algorithms studying infinite games on finite graphs representing concurrent structures, as the one introduced in [3].

In [2] an algorithm is proposed for determining if there is a winning strategy for controlled reachability of a target transition. The algorithm works on distributed net systems with full observability, i.e.: on elementary net systems where concurrency is present only between transitions of the user and of the environment, and where all choices are local either to the user or to the environment. The idea behind the algorithm is that the user increases the chances of winning by observing as much as possible the decisions of the environment. Determining the complexity of the algorithm is still an open problem.

4 Ongoing work and future developments

We are generalizing the results mentioned in the previous section mainly in two directions: on one side we consider systems with partial observability, i.e.: systems in which some transitions are not observable, and where there is no restriction on concurrency between transitions; on the other side we intend to study when strategies can be implemented by adding new elements to the system model, in such a way that the desired property is satisfied.

For what concerns the first direction, the following example can give the idea of how the definition of a game on unfolding may allow to define a winning strategy for the user, whereas this would be not possible by considering a game based on interleaving semantics.

Example 2. Let us consider the net system given in the upper part of Figure 2, and let us assume that t is the target transition, and that the game is played on the system unfolding as given in the lower part of the same figure.

A winning strategy for the user is: $\alpha(\{a, b_1\}) = \{x_1\}$, $\alpha(\{a, b_2\}) = \{y_2\}$. For all other observable configurations, the strategy should decide for the empty set.

More explicitly, if the user observes with the event corresponding to transition a the event b_1, corresponding to transition b, then, in order not to reach a deadlock, he has to choose the event corresponding to transition x, whereas if he observes the event b_2, he has to choose the event corresponding to transition y. This means that, by observing the events on the unfolding, the user is able to infer which unobservable transition occurred before b. This inference would have not be possible in the case of a game based on interleaving; in fact, just by observing sequences, a user would not have a winning strategy: after observing the sequence ab, he would have no strategy to choose between x and y.

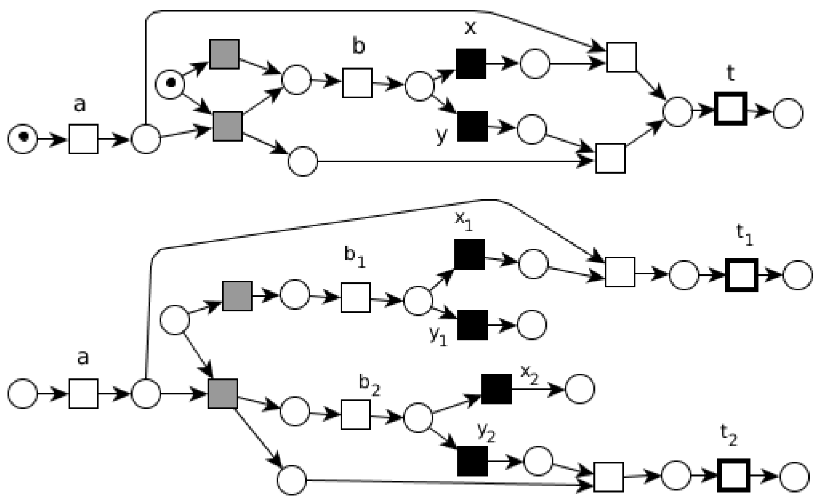

Fig. 2. A net system (above) and its unfolding (below).

This work has been partially supported by MIUR.

References

1. Abramsky, S., Melliès, P.: Concurrent games and full completeness. In: 14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999. pp. 431–442. IEEE Computer Society (1999). https://doi.org/10.1109/LICS.1999.782638, https://doi.org/10.1109/LICS.1999.782638
2. Adobbati, F., Bernardinello, L., Pomello, L.: An asynchronous game on distributed Petri nets. In: Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'19), 2019, Aachen, Germany, June 24, 2019. (2019)
3. 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
4. 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), 2018, 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
5. 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
6. 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
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. 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
9. Melliès, P., Mimram, S.: Asynchronous games: Innocence without alternation. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Lisbon, Portugal, September 3-8, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4703, pp. 395–411. Springer (2007). https://doi.org/10.1007/978-3-540-74407-8, https://doi.org/10.1007/978-3-540-74407-8
10. Rozenberg, G., Engelfriet, J.: Elementary net systems. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996. Lecture Notes in Computer Science, vol. 1491, pp. 12–121. Springer (1996). https://doi.org/10.1007/3-540-65306-6, https://doi.org/10.1007/3-540-65306-6
11. Winskel, G.: Distributed games and strategies. CoRR **abs/1607.03760** (2016), http://arxiv.org/abs/1607.03760