Asynchronous games on Petri nets

Federica Adobbati

DISCo, Università degli studi di Milano-Bicocca
f.adobbati@campus.unimib.it

1 Introduction

I discuss a notion of game on Petri nets. Games allow us to verify properties on concurrent systems partially controlled by different autonomous agents. Through the game, we can analyze the interaction of a group of *users* with a shared goal on the net system. Every user controls some of the transitions in the net and observes some of its states. If a user controls a transition that is enabled in a given marking, he can decide whether to fire it or not. We can interpret users as *players* of the game. The game is played on the unfolding of the net, a *play* is a run on the unfolding. The behavior of each user is determined by a *strategy*, a function that determines the choices of the users based on their observations. Some of the transitions are not controllable by any user; we say that they belong to the *environment*. The environment does not have any goal, but must guarantee the progress of the system. The game is *asynchronous*, i.e. there is not an established order to move for the players, but at every moment every enabled transition can occur. The users win a play if the behavior of the system satisfies their goal in that play. If the users can win every play, regardless of the behavior of the environment, we say that they have at least one winning strategy. In the research of a winning strategy we can interpret the environment as the *opponent* of the users.

In the last decades, games have often been used to model the interaction between autonomous agents. We briefly mention some approaches, in which authors study notions of games on concurrent systems.

In 2002, Alur, Henzinger and Kupferman developed a game-based temporal logic [3] interpreted over *concurrent game structures*. Using this logic, they characterized properties that can be forced by a subset of players, independently from the behavior of the others.

In 1999, Abramsky and Melliès defined concurrent games on partial orders [1]. In their model, a strategy is a function on domains of positions and it is not necessarily defined in all the positions of the domain, since some positions may never be reached by following that strategy. In the next years other authors generalized their results by developing notions of games on partial orders and event structures [7, 4].

Finally, Finkbeiner and Olderog in [6] developed *Petri games* for the verification of a safety property. In their model players are interpreted as the tokens on the net, and they exchange information through synchronizations.

2 Controlled reachability

My recent work focused on a restricted case of the general game model described above. The system is modelled by a 1−safe Petri net on which only two players interact, a user and the environment. The user controls a sequential component of the net system, there is no conflict between the transitions of the user and of the environment, but there are conflicts inside each component. The user wins a play if a *target transition* occurs during the play. Hence, the user has a winning strategy, if he can force the occurrence of the target in every play.

As an example, consider the net in Figure 1: the user controls the black transitions, while the white ones belong to the environment. The target is the transition t_7. The initial marking enables two transitions belonging to the environment and two belonging to the user, hence both players could move. The user loses if the net system reaches the markings $\{p_3, p_6, p_{10}\}$ or

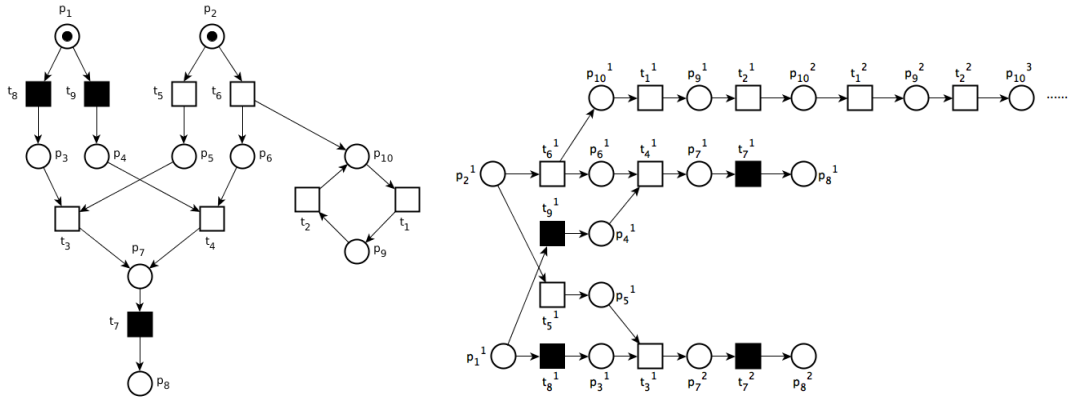

Figure 1: A net system and its unfolding

$\{p_4, p_5\}$, since the user cannot reach t_7 from them. The user wins if the system reaches $\{p_3, p_5\}$ or $\{p_4, p_6, p_{10}\}$, since the environment has to guarantee progress and cannot prevent the user from firing t_7 once the transition has been finally enabled. Hence, if the user knows whether t_5 or t_6 occurred, he is always able to win. The winning strategy chooses $\{t_8\}$ when the system is in $\{p_1, p_5\}$, $\{t_9\}$ in $\{p_1, p_6, p_{10}\}$ and $\{p_1, p_6, p_9\}$, and finally $\{t_7\}$ in $\{p_7, p_{10}\}$, $\{p_7, p_9\}$ and $\{p_7\}$. If the user does not know whether t_5 or t_6 occurred, he may win some plays by chance, but he does not have a winning strategy.

Together with my supervisors, in [2] I developed an algorithm to find a winning strategy, if one exists, in case of complete information for the user. The algorithm recursively constructs a prefix of the unfolding while looking for the strategy. For every cut that is reached in the prefix, the algorithm explores all the possible plays obtained by adding an uncontrollable transition to the cut. It inserts a controllable transition in the strategy when, starting from the cut following the transition, every play consistent with the strategy that has already been constructed reaches the target.

We are currently working to generalize the algorithm to the case of partial observability for the user. Given a subset of observable elements of the net, we believe that it is possible to use the winning strategy for full observability to explore the existence of a strategy in the case of partial observability. The idea seems reasonable because we assume that the user knows the structure of the net, even if he cannot observe some of its elements, therefore he is always able to compute a strategy in the ideal case of complete information. Hence, the user can reconstruct all the possible scenarios that lead to his observations and choose controllable transitions that let him win in all of them, if some exist. In the example in Figure 1, if we suppose that the user does not observe the states p_5, p_6, p_{10} and p_9, then the user does not have a winning strategy, since he cannot be sure to avoid the deadlock. If the user observes all but p_2, p_6 and p_{10}, he has a winning strategy, because observing p_5 and p_9 is enough to determine which transition occurred between t_5 and t_6.

3 Ongoing works

In addition to partial observability, we are currently working to extend the research of a strategy to other goals for the user. Moreover, we would like to express the goals with temporal logic formulae. To this aim, we are exploring the possibility to adapt Alternating-time Temporal Logic (ATL) to Petri nets and their unfoldings. In its original formulation [3], ATL considers infinite paths on game structures with a finite number of states and a finite number of transitions available in every state for every player. At every step, each player chooses his move among the ones available in the current state of the system. The next state is determined by the combination of the moves of all the players. Because our game is not divided into steps, every player can fire every enabled transition at any time without waiting for other players to move.

This makes the projection of the Petri net game on a game structure nontrivial, but succeeding in it would allow to interpret ATL formulae on the net and enable the use of the model checking techniques defined on game structures.

Another topic that we want to investigate is the complexity of the algorithms to find strategies. Developing the game on the unfolding rather than on the case graph of the net will hopefully lead to more efficient algorithms. Some authors, like Esparza, Römer and Vogler [5], showed how to construct a prefix of the unfolding that contains enough information to verify a certain property on the system. The dimension of these prefixes is usually smaller than the case graph of the net, because the unfolding can represent concurrency, while the dimension of the reachability graph grows exponentially with the level of concurrency of the net. Considering these results, one of our future goals is the definition of a prefix of the unfolding in which there is enough information to decide if the users have a winning strategy.

Finally, applications of such a game are in the general frame of verification, adaptation and control of distributed systems [8]; in this context, we want to study if and when it is possible to implement a strategy on the net systems by adding causal constraints between the transitions of the net. To clarify this point, consider as example the net in Figure 1: the transition t_8 is independent from t_5, but the strategy asks the user to wait for the occurrence of t_5 before firing t_8. In other words, when the user follows the strategy, he introduces a causal dependency between t_5 and t_8 that is not represented in the net. We can impose this behavior to the net by adding a place that is a post-condition of t_5 and a pre-condition of t_8. Analogously, we could add a post-condition of t_6 that is also a pre-condition of t_9. In the modified net, every run respects the strategy. Hence, if the users have a winning strategy in the original system and a such a correction is possible, we could modify the net such that the system satisfies the winning condition independently from the users behavior.

References

[1] Samson Abramsky and Paul-André Melliès. Concurrent games and full completeness. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 431–442. IEEE Computer Society, 1999.

[2] Federica Adobbati, Luca Bernardinello, and Lucia Pomello. An asynchronous game on distributed Petri nets. In Daniel Moldt, Ekkart Kindler, and Manuel Wimmer, editors, *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE 2019), Aachen, Germany, June 23-28, 2019*, volume 2424 of *CEUR Workshop Proceedings*, pages 17–36. CEUR-WS.org, 2019.

[3] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[4] Simon Castellan, Pierre Clairambault, Silvain Rideau, and Glynn Winskel. Games and strategies as event structures. *Logical Methods in Computer Science*, 13(3), 2017.

[5] Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods Syst. Des.*, 20(3):285–310, 2002.

[6] Bernd Finkbeiner and Ernst-Rüdiger Olderog. Petri games: Synthesis of distributed systems with causal memory. *Inf. Comput.*, 253:181–203, 2017.

[7] Julian Gutierrez. Concurrent logic games on partial orders. In Lev D. Beklemishev and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation - 18th International Workshop, WoLLIC 2011, Philadelphia, PA, USA, May 18-20, 2011. Proceedings*, volume 6642 of *Lecture Notes in Computer Science*, pages 146–160. Springer, 2011.

[8] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81, 1989.