Implementable strategies for a two-player asynchronous game on Petri nets

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

[1]*Università degli studi di Milano–Bicocca, DISCo*

Abstract

We consider a two-player game on Petri nets, in which each player controls a subset of transitions. The players are called 'user' and 'environment'; we assume that the environment must guarantee progress on its transitions. A play of this game is a run in the unfolding of the net, satisfying the progress assumption. In general, we define a strategy for the user as a map from the set of 'observations' to subsets of transitions owned by the user. Different restrictions on strategies can be used to encode observability assumptions. We say that a given strategy is implementable if the net can be endowed with new places so that the runs of the new net coincide with the plays of the original net, complying with the strategy. We propose an algorithm based on the search of regions to decide whether a strategy is implementable.

1. The game

We recall the definition of a two-player, asynchronous game on 1-safe Petri nets (see [1]), define a notion of implementable strategy for one of the players, and give a simple algorithm to decide whether a given, general, strategy is implementable.

Let us introduce the main notions related to the game through an example. Consider the net in Fig. 1. Two players interact on it, the *user*, by controlling the light grey transitions, and the *environment*, by controlling the white ones. The game is asynchronous: the players can concurrently fire their transitions. The game is asymmetric: the user has the right to keep his transitions blocked when they are enabled, whereas the environment must guarantee progress of its transitions. In this example, we suppose that the user has full knowledge of the current marking, and that he has the goal of marking place q infinitely often. In order to win, the user must wait for the environment to choose between firing t_1 or t_2. He can do it, since the environment cannot delay this choice forever. In the former case, the user chooses u_1, otherwise u_2. The environment is then forced to fire either v_1 or v_2, with the effect of marking q, and then to fire z, reproducing the initial marking.

Formally, a Petri net is a tuple $\Sigma = (P, T, F, m_0)$, where P is the set of places, T the set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ the flow relation, $m_0 : P \to \mathbf{N}$ the initial marking. A marking is a map $m : P \to \mathbf{N}$. We suppose the reader knows the definition of the firing rule, denoted $m[t\rangle m'$, and how to compute the set of reachable markings. Let M be the set of reachable markings in Σ; then Σ is 1-safe iff $\forall m \in M$, $\forall p \in P$, $m(p) \leq 1$. We assume that the game is played on a 1-safe Petri net. In a 1-safe Petri net a marking can be interpreted as the characteristic function of a subset of places. Hence, in the rest of the paper we will denote

ATAED'22, June 20-21, 2022, Bergen, Norway

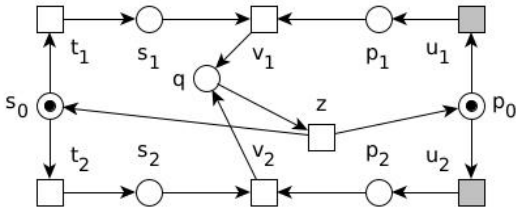

Figure 1: A game net

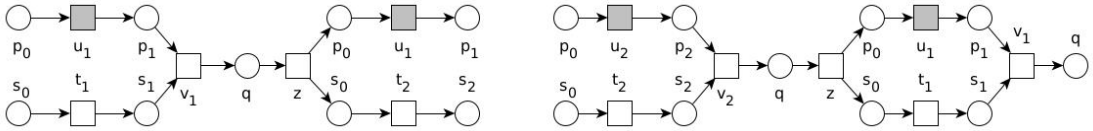

Figure 2: A full play (left) and a partial play (right)

markings as sets of places. We denote with $\mathsf{MG}(\Sigma) = (M, T, A, m_0)$ the *sequential marking graph* of Σ, where $A = \{(m, t, m') : m, m' \in M, t \in T and\ m[t\rangle m'\}$ is the set of labelled arcs. We denote with T_u the set of transitions controlled by the user, and with T_e the set of transitions uncontrollable for him. We assume that $T_e \cup T_u = T$, $T_e \cap T_u = \emptyset$.

We will also refer to the *unfolding* of the net (see [2] for the formal definitions). The unfolding is an acyclic, possibly infinite, net representing all the possible histories of the executions of a net system. In the unfolding of a net system, only forward conflicts are allowed: two events can share a precondition, but no postcondition. Since the unfolding is acyclic, the reflexive and transitive closure of the flow relation is a partial order. If two events e_1 and e_2 are in conflict, then we say that every descendant of e_1 is in conflict with every descendant of e_2. Two elements are *concurrent* if they are neither ordered nor in conflict. The events of the unfolding are partitioned into *controllable* and *uncontrollable* events depending on their correspondence to occurrences of controllable or uncontrollable transitions, respectively.

A *run* is a subnet of the unfolding representing an execution, i.e. it is a net without conflicts and close with respect to the past of its elements. A run is *maximal* if no event can be added without creating a conflict.

A *play* in the game is formally defined as a run in the unfolding of the net, maximal with respect to uncontrollable transitions. The winning condition for the user is a set of plays. The user wins a play if the play belongs to the winning condition. In the example discussed above, the winning condition is formed by all the maximal runs with an infinite number of occurrences of the place q. Figure 2 shows, on the left, a play ending in a deadlock (the environment wins) and, on the right, the initial segment of an infinite play (the user wins).

2. Strategies

In order to reach his goal, the user can apply a strategy. We assume that the user has no memory and only a partial knowledge of the current state of the net. This is formalized by assuming an equivalence relation, denoted by $\equiv$, on the set of reachable markings. The equivalence classes of this relation will be called *observations*, and a strategy is defined as a map from observations to sets of controllable transitions. For example, if we assume the user may observe only some places, two markings are considered equivalent if they share the same observable places. An example of equivalence relation between partially observable markings has been considered in the game presented in [3].

Let Obs be the set of observations. Then a strategy is a map $\alpha : \mathsf{Obs} \to 2^{T_u}$. The notion of observation can be extended to unfoldings: a maximal set of pairwise concurrent places in the unfolding is called a *B-cut*. Every B-cut corresponds to a reachable marking. We say that two B-cuts are equivalent if their corresponding markings are equivalent.

Two B-cuts of a run π, γ and γ', are ordered, denoted $\gamma < \gamma'$, iff $\gamma \neq \gamma'$ and $\forall b \in \gamma, \exists b' \in \gamma'$ such that: $b \leq b'$. A sequence $\gamma_1\gamma_2...\gamma_n...$ of B-cuts is *increasing* if $\gamma_i < \gamma_j$ for each $i, j \in \mathbf{N}$ with $i < j$.

A play π *complies* with a strategy α if (1) there is a sequence μ of observations which is the projection of an increasing sequence of B-cuts in π, and (2) every controllable event in π is chosen by the strategy in one of the observations in μ.

Given a strategy α, we can construct the reduced version of $\mathsf{MG}(\Sigma)$, by removing arcs corresponding to controllable transitions not chosen by the strategy, and then removing all states and transitions unreachable from the initial marking. The reduced version will be denoted by $\mathsf{MG}_\alpha(\Sigma)$, and the operation will be referred to as the α-reduction of $\mathsf{MG}(\Sigma)$. Fig. 3 shows the reduced version of the marking graph of the net in Fig. 1, when α is the strategy described above.

2.1. Implementable strategies

As defined above, the notion of strategy is not encoded in the net system itself. We will now investigate the possibility of defining a strategy within the net.

The idea we pursue is this: a strategy is encodable within a net if its decisions can be represented as extra places, corresponding to causal relations from uncontrollable transitions to controllable transitions. We will then say that a strategy is implementable if we can add new places to the given net, so that the runs of the augmented nets are exactly the plays complying with the strategy in the original net. Formally, we need a notion of "augmented" net system, in which we add places that restrict the possible behaviours.

For $\Sigma_i = (P_i, T_i, F_i, m_{0i})$, with $i = 1, 2$, we write $\Sigma_1 \sqsubseteq \Sigma_2$ if Σ_2 is obtained by adding new places to Σ_1, hence $P_1 \subseteq P_2$, $T_1 = T_2$, $F_1 \subseteq F_2$ and new arcs in F_2 have one end in $P_2 \setminus P_1$, $m_{01} \subseteq m_{02}$ and $m_{02} \setminus m_{01} \subseteq P_2 \setminus P_1$.

A given strategy α might prohibit any occurrence of a given controllable transition, and might make some uncontrollable transition unreachable in complying plays. Let T_α be the set of reachable transitions in complying plays, and let Σ_α be the net system whose underlying net is the subnet generated by T_α, i.e.: $\Sigma_\alpha = (P_\alpha, T_\alpha, F_\alpha, m_{0\alpha})$, where $P_\alpha = {}^\bullet T_\alpha \cup T_\alpha^\bullet$, F_α is F

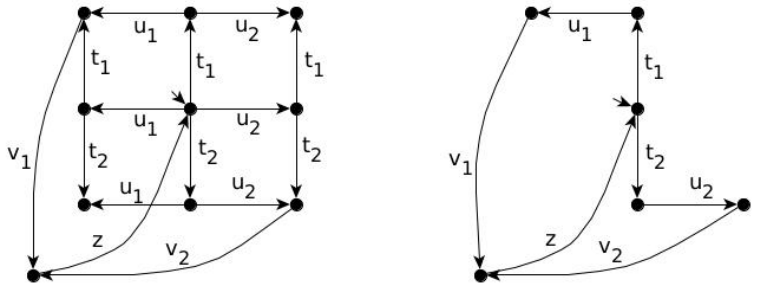

Figure 3: The marking graph of the net in Fig. 1 and its reduced version $\mathsf{MG}_\alpha(\Sigma)$

restricted to $(P_\alpha \times T_\alpha) \cup (T_\alpha \times P_\alpha)$ and $m_{0\alpha} = m_0 \cap P_\alpha$.

Definition 1. *A strategy α for a game on Σ is* implementable *if there exists a 1-safe net system $\Sigma' = (P', T_\alpha, F', m'_0)$ such that (1) $\Sigma_\alpha \sqsubseteq \Sigma'$; (2) the marking graph of Σ' is isomorphic to $\mathsf{MG}_\alpha(\Sigma)$.*

Consider again the net system in Fig. 1. Suppose that the user can observe all the places in the net, so that it has full knowledge of the current marking. If his goal is to reach infinitely often the place q, then a winning strategy is defined by the following clauses: (1) $\alpha(\{p_0, s_0\}) = \emptyset$; (2) $\alpha(\{p_0, s_1\}) = \{u_1\}$; (3) $\alpha(\{p_0, s_2\}) = \{u_2\}$. In this case, the strategy α does not prevent the occurrence of any controllable transition, it only selects which one between u_1 and u_2 to choose, depending on the observation of $\{p_0, s_1\}$ or $\{p_0, s_2\}$; therefore the set of reachable transitions in plays complying with α, T_α, is T itself, and then $\Sigma_\alpha = \Sigma$.

The marking graph of Σ, $\mathsf{MG}(\Sigma)$, is shown on the left side of Fig. 3, whereas the reduced version $\mathsf{MG}_\alpha(\Sigma)$ is on the right side of the same figure. The information needed to take the right decision in accordance with α, in this simple case, can be encoded by adding to Σ a couple of places, one going from t_1 to u_1, the other from t_2 to u_2, as shown in Fig. 4, on the right, where the new places are drawn in dark grey. It is easy to see that the obtained system has a marking graph isomorphic to $\mathsf{MG}_\alpha(\Sigma)$.

3. An algorithm to decide if a strategy is implementable

In this section, we propose a simple algorithm to decide whether a given strategy is implementable. The algorithm checks whether the reduced marking graph is isomorphic to the marking graph of a 1-safe Petri net, and relies on the theory of regions [4]. We first recall some basic notions related to regions of transition systems.

Region theory and separation problems As remarked above, each node of the marking graph of a 1-safe Petri net is a set of places. Hence, we can associate to each place its *extension*, namely the set of reachable markings to which it belongs: $\forall p \in P \quad r(p) = \{m \in M \mid p \in m\}$. The extension of a place satisfies the *uniform crossing* property: for any given transition t, if

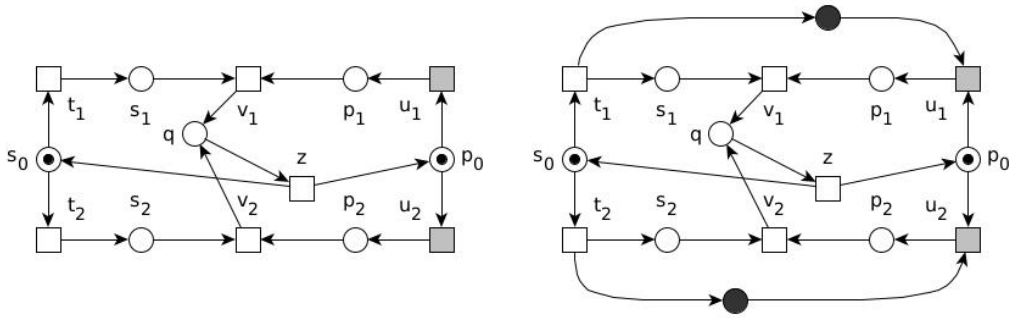

Figure 4: A game net with an implemented strategy

one occurrence of t in the marking graph enters $r(p)$, then all occurrences of t do the same, and analogously when an occurrence leaves $r(p)$. This property can be defined more abstractly for general labelled transition systems, giving the notion of a region of a transition system.

The synthesis problem consists in deciding if a given labelled transition system is isomorphic to the marking graph of a net system, and, if so, in constructing such a net. The problem admits a solution if the set of regions satisfies the so-called state separation problems and the event-state separation problems. In this case, we say that the transition system is separated.

Consider now a marking graph $\mathsf{MG}(\Sigma)$, a strategy α, and the reduced transition system $\mathsf{MG}_\alpha(\Sigma) = (Q, T_\alpha, A, q_0)$. Then, it is easy to prove that, for each region r of $\mathsf{MG}(\Sigma)$, the set $r \cap Q$ is a region of $\mathsf{MG}_\alpha(\Sigma)$. This implies that all state separation problems of $\mathsf{MG}_\alpha(\Sigma)$ are solved by those regions, since $\mathsf{MG}(\Sigma)$ is separated by definition. This is not the case for the event-state separation problems: for each edge labelled with t removed from $\mathsf{MG}(\Sigma)$ in the state s, we need to check if there is a region without state s from which t leaves.

The algorithm Given a strategy α for a game on Σ, we can now describe an algorithm to decide whether α is implementable. The algorithm first computes $\mathsf{MG}_\alpha(\Sigma)$, and then checks if it is separated by trying to solve each new event-state separation problem. The new regions solving those problems, if any, encode the flow of information from observations to the controllable part of the system.

4. Conclusion

In this work we presented a notion of implementable strategy on 1-safe nets, and we provided an algorithm to decide whether the strategy is implementable. In future works, we plan to broad the definition of implementable strategy by allowing for strategies modelled by adding general bounded places. As an example, consider the net on the left of Fig. 5 where the user can observe the places $\{3, 4, 5, 6, 10\}$, and his goal is to reach place 12. A winning strategy is: $\alpha(\{4, 6, 10\})$ =H, $\alpha(\{3, 5, 10\}) = \alpha(\{3, 6, 10\}) = \alpha(\{4, 5, 10\}) =$ I. The choice of H can be represented by adding a place from B to H, and a place from D to H. Instead, the choice of I

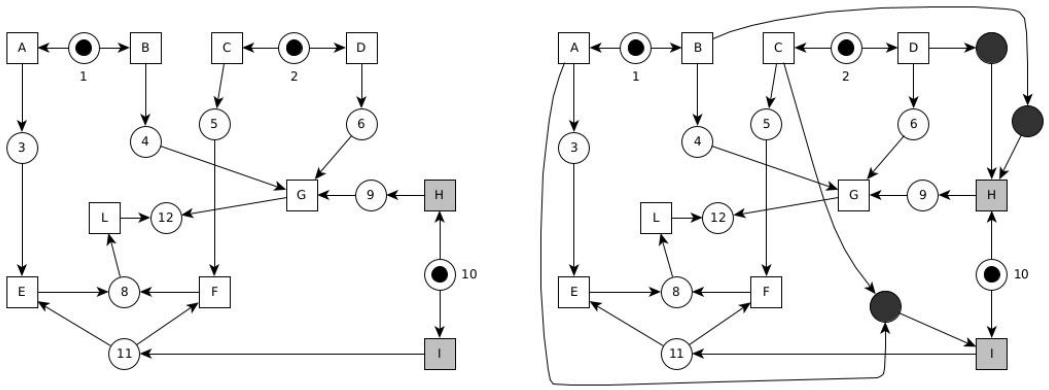

Figure 5: An example of non-implementable strategy

depends on the occurrence of either A or C, hence it may be represented with a place allowing for two tokens, making the net not 1-safe. Although the strategy is not implementable according to the given definition, the net on the right of Fig. 5 realizes the strategy.

Furthermore, we plan to relax the definition of implementable strategy, by allowing for implementations in which only parts of the behaviours in the reduced marking graph are reproducible on the net. Specifically, a strategy is implementable if we can add places to the initial net so that there is at least a maximal run in the unfolding of the augmented net, and all its maximal runs are associated to maximal paths of the reduced marking graph.

A similar use of regions can be found in applications to problems of control: in [5] and [6] regions are used to synthesize maximally permissive controllers avoiding unwanted states; [7] proposes an implementation based on region theory of a controller, applied to flexible manufacturing systems.

A general overview of works and approaches to automatic control using Petri nets can be found in [8].

Acknowledgments

This work is supported by the Italian MUR.

References

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

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

[3] F. Adobbati, L. Bernardinello, L. Pomello, Looking for winning strategies in two-player

games on Petri nets with partial observability, 2022. URL: https://arxiv.org/abs/2204.01603. doi:10.48550/ARXIV.2204.01603.

[4] E. Badouel, L. Bernardinello, P. Darondeau, Petri Net Synthesis, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2015. URL: http://dx.doi.org/10.1007/978-3-662-47967-4. doi:10.1007/978-3-662-47967-4.

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

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

[7] S. Rezig, C. Ghorbel, Z. Achour, N. Rezg, PLC-based implementation of supervisory control for flexible manufacturing systems using theory of regions, International Journal of Automation and Control 13 (2019) 619–640.

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