Acta Informatica (2020) 57:591–626
https://doi.org/10.1007/s00236-020-00368-5

ORIGINAL ARTICLE

Solving high-level Petri games

Manuel Gieseking[1] · **Ernst-Rüdiger Olderog**[1] · **Nick Würdemann**[1]

Published online: 6 May 2020

Abstract
The manual implementation of local controllers for autonomous agents in a distributed and concurrent setting is an ambitious and error-prune task. Synthesis algorithms, however, allow for the automatic generation of such controllers given a formal specification of the system's goal. Recently, high-level Petri games were introduced to allow for a concise modeling technique of distributed systems with a safety objective. One way of solving these games is by a translation to low-level Petri games and applying an existing solving algorithm. In this paper we present a new solving technique for a subclass of high-level Petri games with a single uncontrollable player, a bounded number of controllable players, and a local safety objective. The technique exploits symmetries in the high-level Petri game. We report on encouraging experimental results of a prototype implementation generating the reduced state space. The results for four existing and one new benchmark family show a state space reduction by up to three orders of magnitude.

1 Introduction

Due to the constant availability of networks and the minimization of powerful devices, modern systems are increasingly composed of a huge number of networked computers. Even if the system itself appears to be a single coherent unit, the components of such a *distributed system* act autonomously [52]. To avoid a constant communication of every single component with a central control, systems are more and more decentralized. This comes with the cost of an incomplete knowledge of the system's components about the system's environment.

This work was supported by the German Research Foundation (DFG) through the Research Training Group (DFG GRK 1765) SCARE and through Grant Petri Games (No. 392735815). Ernst-Rüdiger Olderog dedicates this article to Rob van Glabbeek in memory of many pleasant meetings starting in the 1980s in Amsterdam and in admiration for his unique talent in designing and analyzing mathematical models of concurrent processes.

✉ Nick Würdemann
wuerdemann@informatik.uni-oldenburg.de

Manuel Gieseking
gieseking@informatik.uni-oldenburg.de

Ernst-Rüdiger Olderog
olderog@informatik.uni-oldenburg.de

[1] Department of Computing Science, University of Oldenburg, Oldenburg, Germany

Especially in manufacturing there is a rising demand for the development of local controllers and their mutual communication [40,41].

The growth of these systems in size and complexity makes it even more challenging for humans to correctly implement sound controllers. *Synthesis* [11] avoids this error-prune task by automatically generating controllers from a description of all possible actions in the system and a specification of the system's common goal which should be guaranteed against all possible behavior of the system's environment.

In this paper we consider *Petri games* [25], a game-theoretic approach for the synthesis of distributed systems. The game is played between two teams: the *system players* (the controllable behavior) and the *environment players* (the uncontrollable behavior). Solving such games means finding a *strategy* for the system players that is *winning*, i.e., it satisfies a given safety objective against all the environment's decisions. A strategy takes the locally known decisions of the other players as input and produces a deterministic output in form of a decision for the next step. In Petri games the players are the tokens of the underlying place/transition Petri net (P/T Petri net). The places of the net are partitioned into system and environment places. The assignment of the team membership is done via the place the player currently resides on. Players in distant locations do not know anything about each other as long as they are not communicating, i.e., participating on the firing of a joint transition. During such communications the players exchange their knowledge about the *causal past* of the other players, i.e., the places and transitions the player previously resided on or participated in firing, as well as his or her own past.

The high-level representation of Petri games [30] allows for a concise description of these games. Rather than depending on P/T Petri nets, high-level Petri games are based on schemata of Colored Petri Nets (CPNs) [29,36]. This facilitates to have several individual and distinguishable tokens residing on a single place. So far high-level Petri games are solved via a transformation to the equivalent low-level variant of the game and by applying the solving algorithms for P/T Petri games to the result of the transformation. In practical applications, modeling with high-level Petri games often results in low-level Petri nets exhibiting a large amount of symmetric behavior. The main reason is that in many cases the individual tokens, e.g., robots, processes, work pieces, etc., do not need to behave that differently to win the game.

In this paper we present a solving algorithm for a subclass of high-level Petri games with a single environment player, a bounded number of system players, and a safety objective. This new algorithm exploits the symmetries of the system. The subclass is defined by two restrictions. First, we consider only *set-based* high-level Petri games, where the markings are sets rather than multisets of individual, colored tokens. Second, we consider only Petri games where the single environment player is *recurrently interfering*, i.e., in every infinite sequence of transitions there are infinitely many environment transitions.

The key idea of the algorithm is a combination of the reduction technique of the corresponding class of P/T Petri games described in [25] and the construction of a reduced reachability graph for CPNs presented in [8]. We introduce the *symbolic two-player game* $\mathbb{G}^H$, a two-player game over a finite graph with complete information. This game is solvable if and only if the corresponding high-level Petri game $\mathscr{H}$ is solvable. The states of $\mathbb{G}^H$ are equivalence classes of the states of the two-player game $\mathbb{G}^L$ presented in [25] of the corresponding low-level Petri game, with respect to defined symmetries. The correctness of the construction is shown via a bisimulation between these two-player games. Furthermore, we provide an algorithm to create a winning positional strategy in $\mathbb{G}^L$ from the winning positional strategy in $\mathbb{G}^H$.

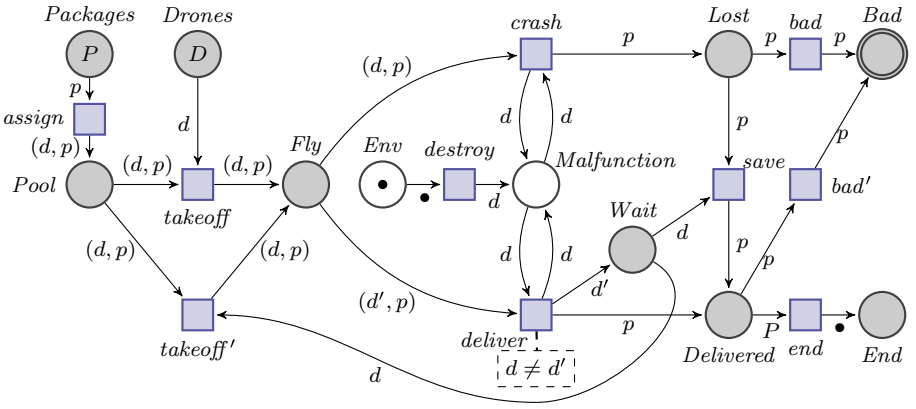

Fig. 1 A high-level Petri game representing the benchmark family *PD*. An armada of n drones $D = \{d_1, \ldots, d_n\}$ wants to deliver m packages $P = \{p_1, \ldots, p_m\}$. The hostile environment, initially residing in the place *Env*, destroys via transition *destroy* one drone $d \in D$ and puts its name d into the place *Malfunction*. Small letters d, d', and p at the edges are variables ranging over elements from D and P, respectively. The functioning drones can deliver their assigned packages via transition *deliver*. After each delivery, a drone has the option to save the package of the crashed drone via transition *save*. The system players win when every package is delivered and thus by firing transition *end* the place *Bad* can be avoided

We validate the state space reduction of our approach on a set of benchmark families introduced in [21,22] in a prototype implementation. We develop the high-level representation of those benchmark families which had not already been introduced in [30] and introduce a completely new benchmark family about a package delivery service with drones. The experimental results calculated with a two hour time out show a state space reduction of a factor of up to 2366.

The remainder of the paper is structured as follows: We informally introduce the new benchmark family of a package delivery in Sect. 2 and motivate our approach by providing an intuition of the symmetries of the system. Section 3 recalls the formalism of low-level and high-level Petri games. In Sect. 4 the new solving technique for high-level Petri games is introduced and proven to be correct, before Sect. 5 shows the state-space reduction of this new technique on several benchmark families. We finish with the related work and a conclusion in Sects. 6 and 7.

2 Motivating example

We motivate our approach by introducing a new benchmark family for the synthesis of distributed systems. In this family, called *Package Delivery (PD)*, an armada of autonomous drones has the mission to deliver packages against a hostile environment. The environment is able to let one of the drones crash. The other drones, however, get informed of the crash and can recover the lost package after their own successful delivery. A visual representation as a high-level Petri game is depicted in Fig. 1.

The n drones are represented by a set $D = \{d_1, \ldots, d_n\}$ of n individual elements called *tokens*, which initially reside on the *place* with label *Drones*. Similarly, the m packages are represented by the set $P = \{p_1, \ldots, p_m\}$, which elements initially reside on the place *Packages*. The gray colored places belong to the controllable players, i.e., tokens residing

on such places belong to the team of system players. However, tokens residing on white places belong to the team of environment players, also called the uncontrollable players. A *transition* is depicted as a rectangular node. A *mode of a transition* is a valuation of the variables at the transition's edges. Those variables (like d, d', p in Fig. 1) are bound only locally to the transition. A transition is *enabled in a mode* iff on the places connected to its incoming edges the necessary tokens are available in this mode and a predicate, written as a dashed box connected to the transition, is satisfied in this mode. A missing dashed box represents the predicate *true*. The transition *fires in a mode* by removing the tokens from the places connected to its incoming edges and by putting tokens to the places connected to its outgoing edges. A transition can only fire in a mode if it is enabled in the mode.

Initially, the transitions *assign* and *destroy* are concurrently enabled. For simplicity reasons, in descriptions we omit the mode of a transition when it is clear from the context. Thus, the assignment of the packages to drones and the subsequent takeoff (via transition *takeoff*) may occur concurrently to the decision of the environment which drone will crash. In Petri games the players do not have a global view of the whole system. They only learn something about the other players by taking a joint transition. In this case the participating players exchange their complete knowledge, i.e., all places and transitions they had previously used as well as the knowledge they obtained from other player by previous joint transitions. This knowledge is called their *causal past*. For the example this means that even in the case that transition *destroy* fires before transition *assign*, the drones, each loaded with a package, take off without knowing which drone will crash. The non-functioning drone, say d, cannot take transition *deliver* due to the predicate $d \neq d'$. To avoid trivial solutions, where the system decides to just not move any further even though there is still a possible move and only by this is able to prevent ending up in a bad place, we are searching for *deadlock-avoiding* strategies. Hence, the drone d must take transition *crash* and has to put its package into the place *Lost*. The functioning drones can deliver their packages (via transition *deliver*) and meanwhile receive the emergency signal of the crashed drone. Thus, by taking transition *deliver*, the drone is getting the environment's decision for the first time because the environment token residing in place *Malfunction* has the transition *destroy* in its causal past. This information is passed to the system player via the joint transition. After each delivery, the drone can decide in place *Wait* to pickup another assigned package via transition *takeoff'*, or, to recover the lost package and call it a day via transition *save*. By taking transition *deliver*, a drone not only learns which drone has crashed but also exactly which package is lost and can therewith recover particularly this package. If no drone decides to recover the lost package, the system cannot avoid reaching the bad place *Bad* due to the deadlock-avoiding constraint. Only when each package is delivered, i.e., all $p \in P$ reside on place *Delivered*, the system can avoid the bad place and reach the place *End* via transition *end*. Even in the case that the environment decides to destroy a drone without an assigned package, all loaded drones can take transition *Delivered* and thus the place *End* is reached.

Consider for example the case of two drones ordered to deliver three packages. In this case, the system cannot win the game because two packages have to be assigned to the same drone. If the environment now destroys this drone, the other, functioning drone, cannot save both undelivered packages. Therefore, some package has to take transition *bad* or *bad'*, and the system loses.

In contrast, if three drones, say d_1, d_2, and d_3, have to deliver two packages, say p_1 and p_2, the system *can* win this game. Initially, the packages decide on an assignment, say p_1 is assigned to d_1 and p_2 is assigned to d_2, and the drones take off. In the case of the environment destroying drone d_1, package p_1 is lost. Then, d_2 can deliver its package, and afterwards can recover p_1 via transition *save*. Both packages now reside on *Delivered*, and therefore can

take the transition *end* to avoid taking any bad transition. In the case that the environment decides to let any other drone crash, the system still can win the game because either now the other drone saves the lost package or no package gets lost anyhow.

We see here that the behavior of the game is highly *symmetric*: it does not matter to which drones the packages p_1 and p_2 get assigned, as long as the two chosen drones are different. For example, there is no difference in the general behavior in the case that p_1 gets assigned to d_3 and p_2 to d_1. We say, this situation is *symmetric* to the situation above. It also does not matter which of the drones loaded with a package is destroyed by the environment – the other drone, after delivering its own cargo, will save the lost package.

These observations lead to the idea of exploiting symmetric behavior: we do not have to consider many different situations anymore, but can limit our examination to *representatives* of whole *classes* of symmetric situations. This makes it easier to determine whether the system can win the game.

3 Petri games

In this section we give an informal definition of low-level Petri games and their properties [24] and recall the definitions of high-level Petri games and their reduction technique to low-level Petri games from [30]. We assume some familiarity with Petri nets (e.g., [43,47]).

3.1 Low-level Petri games

A *Petri game* $\mathcal{G} = (\mathcal{P}_S, \mathcal{P}_E, \mathcal{T}, \mathcal{F}, In, \mathcal{B})$ models a multi-player game, where the tokens of the underlying Petri net $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$, with a finite set of places $\mathcal{P}$, a finite set of transitions $\mathcal{T}$, a flow relation $\mathcal{F}$, and an initial marking In, represent the players. Although the solving algorithm in [25] is presented for k-bounded Petri games, in this paper we assume that $\mathcal{G}$ is *1-bounded*, i.e., in any state of the game each place hosts at most one token. We do so to keep the paper less technical. We distinguish two teams of players: the (uncontrollable) *environment players* are the tokens residing on environment places $\mathcal{P}_E$ (depicted as white circles) and the (controllable) *system players* are the tokens residing on the system places $\mathcal{P}_S$ (depicted as gray circles). The disjoint union of these sets yields the places of the underlying net: $\mathcal{P} = \mathcal{P}_E \dot{\cup} \mathcal{P}_S$. Additionally, the Petri game identifies a set $\mathcal{B} \subseteq \mathcal{P}$ of *bad* places from the point of view of the system (depicted as double circled places). For each transition $t \in \mathcal{T}$ the *pre-* and *postset* of t are defined by $pre(t) = \{p \in \mathcal{P} \mid (p, t) \in \mathcal{F}\}$ and $post(t) = \{p \in \mathcal{P} \mid (t, p) \in \mathcal{F}\}$. Since we assume $\mathcal{G}$ to be *1-bounded*, a *marking* of $\mathcal{G}$ is a set $M \subseteq \mathcal{P}$. A transition t is *enabled* at M if $pre(t) \subseteq M$. *Firing* an enabled transition at M yields a new marking $M' = (M \backslash pre(t)) \cup post(t)$. This *firing relation* is denoted by $M\,[t\rangle\, M'$. We call transitions with a preset only consisting of system places *system transitions* and all other transitions *environment transitions*. Transitions with a preset only consisting of environment places are called *pure environment transitions*. Note that only pure environment transitions are under control of the environment player, whereas transitions which contain at least one system place in its preset are under control of the system players.

Each player knows its own *causal past*, i.e., the places and transitions which have been used to reach the current place. This information is exchanged with all players participating at a joint transition. The aim of the system players is to cooperate to avoid reaching any *bad place* $p \in \mathcal{B}$. To satisfy this *safety objective*, the players can solely use their locally available information.

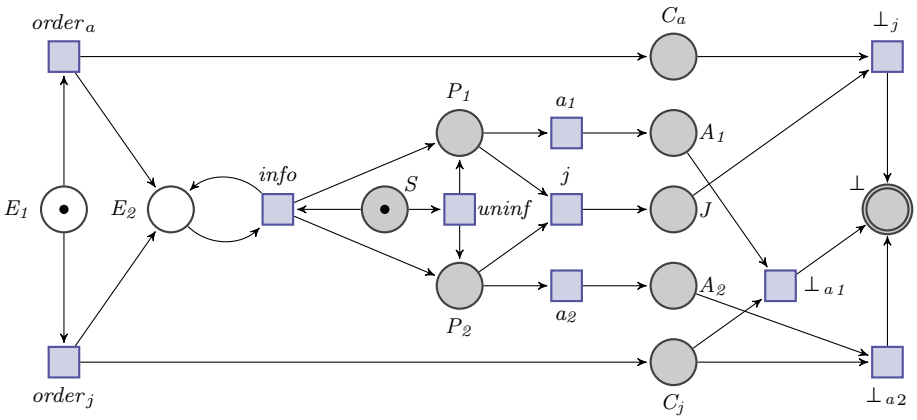

Fig. 2 A low-level Petri game in which the environment player (initially residing on place E_1) orders the system player (initially residing on place S) to let its processes work jointly or solely. The system can only win, i.e., avoid that any token ever resides on place $\perp$, by informing itself over the environment's decision via transition *info* and let its created processes work together or alone according to the environment's decision

Example 1 (Low-level Petri game) Figure 2 shows a simple Petri game with one environment player (initially residing on place E_1) and maximally three system players. At the outset there is only one system player (initially residing on place S). Later on, this player can create two independent processes. The environment however can decide whether these processes should work together (via transition $order_j$) or on their own (via transition $order_a$). In the beginning, the system player can decide whether it waits for the environment's decision and creates the two processes with this information attached (via transition *info*) or whether it just creates two processes without any further attached information (transition *uninf*). In both cases the processes can choose whether they want to work together (transition j) or alone (transition a_i for $i \in \{1, 2\}$). The system can only win, i.e., avoid reaching the bad place $\perp$, when it meets the order of the environment. Therefore, the processes must be created with the information about the environment's order. This knowledge is available in the causal past of the token residing on place E_2 and therewith is transmitted via transition *info* to the system.

The causal dependencies (and independencies) in $\mathcal{G}$ are formally represented by the *unfolding* of the underlying net $\mathcal{N}$ [17,19]. There, every loop in $\mathcal{N}$ is unrolled and every backward branching place is expanded by multiplying the place, so that every transition in the unfolding stands for the unique occurrence (instance) of a transition of $\mathcal{N}$ during an execution. The unfolding exhibits concurrency, causal dependency, and nondeterminism (forward branching of places) of the unique occurrences of the transitions in $\mathcal{N}$ during all possible executions. The unfolding is lifted to Petri games by keeping the distinction of environment, system, and bad places. The unfolding of the Petri game presented in Fig. 2 is shown in Fig. 3. The solid elements together with the lighter shaded ones form the complete unfolding of the Petri game.

A *strategy* for the system players in $\mathcal{G}$ describes a local controller for each system player which operates based on its currently available information about the whole system. A strategy is obtained from the unfolding by deleting some of the branches that are under control of the system players. Thus, it is technically a subprocess of the unfolding and describes for each place which transitions the player in that place can take. A strategy has to meet four conditions: (i) The strategy should not disallow any pure environment transition. This means

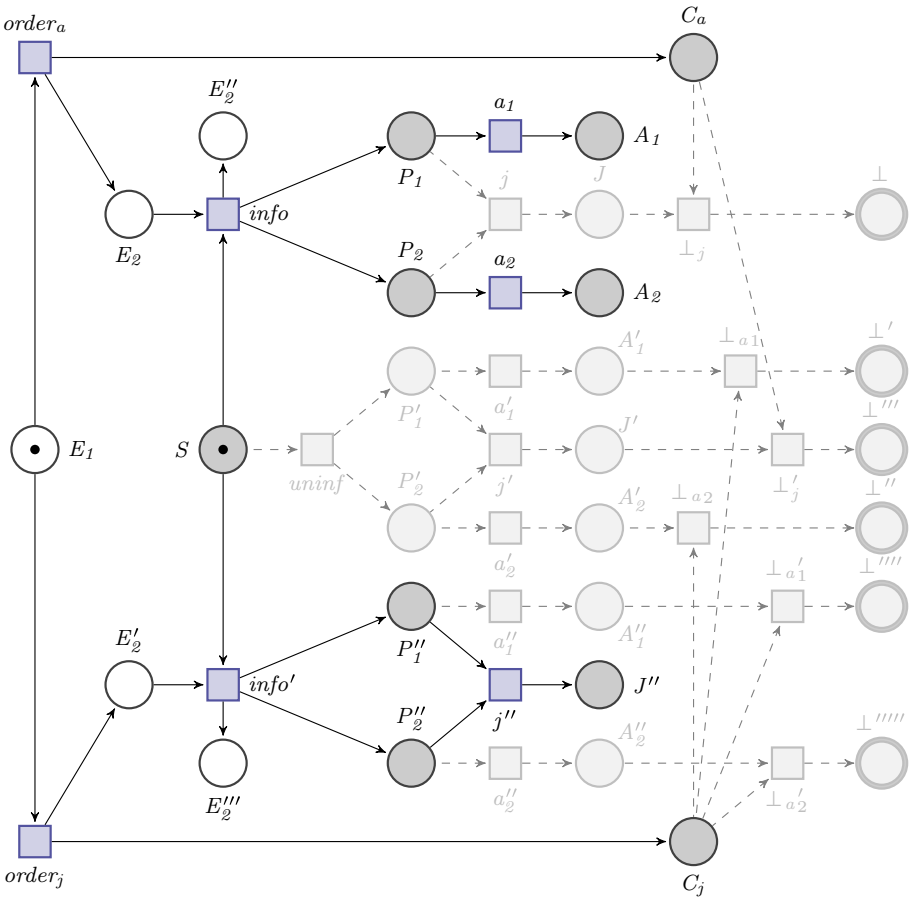

Fig. 3 The unfolding and a winning strategy for the system players in the Petri game presented in Fig. 2. The winning strategy is depicted by the solid elements

the system players cannot prevent the environment from working on its own. (ii) Each refusal of a transition must be *justified*, i.e., when a system player refuses an instance of a transition in a place of the strategy, no other instance of this transition is allowed to occur in the postset of this place. This means that in a specific state, the system can only allow or disallow a transition of the original net. It cannot choose between two instances of this transition in the unfolding, when these instances are indistinguishable due to the available knowledge for the system player. (iii) The strategies must be *deterministic*, i.e., in no state of the strategy two transitions are enabled involving the same system player. (iv) The strategy must be *deadlock-avoiding*, i.e., whenever the system can proceed in $\mathscr{G}$ there must also be a continuation in the corresponding situation in the strategy, to avoid trivial solutions. Since we consider a safety objective, the system players would win a non deadlock-avoiding strategy by just doing nothing.

In a *play* conforming to a given strategy for the system players all remaining nondeterminism has been resolved. The system players *win* this play when it avoids any bad place in $\mathscr{B}$. A strategy σ for the system players in $\mathscr{G}$ is *winning* iff all plays conforming to σ are winning. The formal definitions for unfolding, strategy, and plays are given in "Appendix A".

Example 2 (Strategy in a low-level Petri game) The solid elements of Fig. 3 show a winning strategy for the Petri game presented in Fig. 2. Every strategy has to contain transitions $order_a$ and $order_j$ because all pure environment transitions have to occur. When the system in place S decides to get informed about the environment's decision (as in this winning strategy), it has to do so uniformly in every indistinguishable situation. Thus either no instance of an *info* transition is present in the strategy or both *info* and *info*$'$. After getting informed of the environment's decision (which, due to the different causal pasts, results in two branches) the system can enable the appropriate transitions to avoid a bad place.

The game can only be won because every player memorizes its causal past and transmits this knowledge to all players participating in a joint transition. Suppose in place S the system decides to have *uninf* in its strategy. By the deterministic constraint, it includes neither transition *info* nor *info*$'$ and thus it is not informed of the environment's decision. Then the system cannot avoid a bad place while fulfilling the constraints of a winning strategy. Indeed, allowing all three transitions a'_1,a'_2, and j' in the postsets of P'_1, P'_2 exhibits nondeterminism. Thus, since the system must not deadlock, either A'_1 and A'_2, or J' must be part of the strategy. Disallowing the subsequent transitions ($\bot_{a_1}$ and $\bot_{a_2}$, or $\bot'_j$, respectively) leading to a bad place, however, would yield a deadlock in one of the environment's decision.

A decision taken by the strategy in a place p depends on the causal past of p, which may be arbitrarily large. Similar to model checking approaches based on net unfoldings [18], we use *cuts* (maximal subsets of pairwise concurrent places in the unfolding) as unique representatives of the causal past. The standard notion of cuts, however, collects places with possibly different knowledge of the individual players about the causal past. Consider for example the cut $\kappa = \{E_2, C_a, S\}$ in the unfolding presented in Fig. 3. The environment player residing on E_2 and the system player residing on C_a know the same about the causal past because their last move was the joint transition $order_a$. However, the system player residing on S does not know anything about the environment's decision. Therefore, the paper [25] introduced a new kind of cut, called *maximal cuts*, abbreviated *mcut*. For an environment place p an mcut is a cut including p such that for all places q in that cut either (1) the system players have *maximally progressed* at q, in the sense that any further system transition would require an additional environment transition starting from place p, or (2) the future starting at q does not depend on the environment. Mcuts are especially interesting for strategies in Petri games with a single environment player. Consider for example the loosing strategy where the system decides to take transition *uninf*. There the cut κ is not an mcut because the system player residing on S can still progress without moving the token residing on E_2. In the presented winning strategy however, κ is an mcut because only transition *info* with $E_2 \in pre(info)$ is enabled. Hence, for the winning strategy also the system player residing in S can be considered to be equally informed about the environment's decision because the only possible next move for this player will provide this information. In general for Petri games with one environment player every maximally progressed system player of an mcut [case (1)] can be considered to be equally informed about the environment because the next transition either directly involves the environment player or at least contains the current environment place in its causal past. This is exploited in [25] to create a two-player game with *complete* information which is used to solve Petri games. Also the construction of the later introduced symbolic two-player game for high-level Petri games rests on these cuts such that the complete informedness of the players does not cause any harm.

For simplicity, we restrict ourselves in this paper to Petri games $\mathscr{G}$ where alternative (2) does not arise. In the terminology of [25], we do not consider type-2 places here. In other words, we require that in every infinite sequence of transitions there are infinitely many

environment transitions. Formally, $\mathcal{G}$ has a *recurrently interfering* environment if in every infinite firing sequence

$$In = M_0\,[t_0\rangle\, M_1\,[t_1\rangle\, M_2[t_2\rangle \ldots$$

of $\mathcal{G}$ there are infinitely many $i \in \mathbb{N}$ with $pre(t_i) \cap \mathcal{P}_E \neq \emptyset$. This restriction allows us to keep the formal definitions of elements used in the solving algorithm for Petri games, namely decision sets and the two-player game, as simple as possible and saves to introduce an additional pre-processing algorithm similar to the one presented in [25].

3.2 High-level Petri games

Parameterized set-based high-level Petri games were introduced in [30]. The term *set-based* means that at no point in time two tokens of the same color reside on any place of the game. This corresponds to 1-bounded Petri games in the low-level case. In this paper we restrict ourselves to set-based high-level Petri games *without* parameters. We recall a slightly adapted version of the definition and its properties.

We consider data values that are called *colors*, following [36]. We refer to a finite set of colors as a *color domain*, with typical letter C. Let $\mathscr{C}$ denote the set of all considered color domains. We use *typed variables* that range over a specific color domain. Let Var denote the set of all variables. We use the function $ty : \mathsf{Var} \to \mathscr{C}$ to declare the type $ty(x)$ of a given variable $x \in \mathsf{Var}$, i.e., its color domain. In high-level Petri nets individual tokens will be represented by colors.

A *high-level Petri game* $\mathscr{H} = (\mathcal{P}_S^H, \mathcal{P}_E^H, \mathcal{T}^H, \mathcal{F}^H, ty, g, e, In^H, \mathcal{B}^H)$ is a structure with

- a finite set of *system places* $\mathcal{P}_S^H$,
- a finite set of *environment places* $\mathcal{P}_E^H$ (the disjoint union $\mathcal{P}^H = \mathcal{P}_S^H \dot{\cup} \mathcal{P}_E^H$ yields the set of all places),
- a finite set of *transitions* $\mathcal{T}^H$ satisfying $\mathcal{P}^H \cap \mathcal{T}^H = \emptyset \neq \mathcal{P}^H \cup \mathcal{T}^H$,
- a *flow relation* $\mathcal{F}^H \subseteq (\mathcal{P}^H \times \mathcal{T}^H) \cup (\mathcal{T}^H \times \mathcal{P}^H)$,
- a *type function* $ty : \mathcal{P}^H \to \mathscr{C}$, which maps each place $p \in \mathcal{P}^H$ to its color domain, i.e., the colors which can reside on p,
- a *guarding function* g, which assigns to each transition $t \in \mathcal{T}^H$ a *guard*, i.e., a Boolean expression $g(t)$ which restricts the firing of t,
- an *arc expression function* e, which assigns to each arc $(p, t) \in \mathcal{F}^H$ and $(t, p) \in \mathcal{F}^H$ an *arc expression* $e(p, t)$ or $e(t, p)$, respectively, describing which tokens are withdrawn or placed by t from or on the corresponding place p during the firing,
- an *initial marking* $In^H \subseteq \{(p, c) \mid p \in \mathcal{P}^H \wedge c \in ty(p)\}$, and
- a set of *bad places* $\mathcal{B}^H \subseteq \mathcal{P}^H$.

We require that two different color domains are disjunct, i.e., $ty(p_1) \neq ty(p_2) \Rightarrow ty(p_1) \cap ty(p_1) = \emptyset$ for all places $p_1, p_2 \in \mathcal{P}^H$. For a place $p \in \mathcal{P}^H$ and a color $c \in ty(p)$ we often use $p.c$ as abbreviation for (p, c) to state that the token c resides on the place p. For any set $X \subseteq \mathcal{P}^H$ of places, we denote by $X.ty = \{p.c \mid p \in X \wedge c \in ty(p)\}$ the set of all possible combinations of places in X with colors according to their types.

For a transition $t \in \mathcal{T}^H$ let $\mathsf{var}(t)$ denote the set of free variables occurring in $g(t)$ or any arc expression $e(p, t)$ or $e(t, p)$ for a flow $(p, t) \in \mathcal{F}^H$ or $(t, p) \in \mathcal{F}^H$, respectively. A *valuation* or *mode* v of a transition t assigns to each variable $x \in \mathsf{var}(t)$ a value $v(x) \in ty(x)$. We denote all valuations of a transition t by $Val(t)$. For a set $Y \subseteq \mathcal{T}^H$ of transitions, we

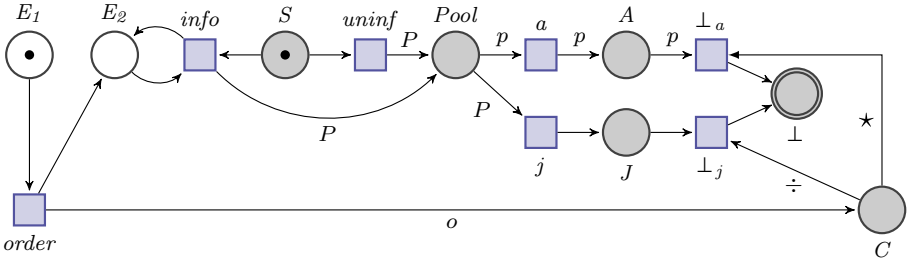

Fig. 4 A high-level Petri game with color domains $\{\bullet\}$, $P = \{p_1, \ldots, p_n\}$, and $\{\div, \star\}$ and the types of the variables $ty(p) = P$ and $ty(o) = \{\div, \star\}$. The environment player (initially residing on place E_1) decides whether a set of sytem processes P should work together (place J) or alone (place A). With a proper renaming and $n = 2$ the Petri game of Fig. 2 is an instance of this high-level version

denote by $Y.Val = \{t.v \mid t \in Y \wedge v \in Val(t)\}$ the set of all possible combinations of transitions in Y with their valuations. Each valuation v of t is inductively lifted from the variables in $\mathsf{var}(t)$ to the expressions around t. We denote by $v(t)$ the Boolean value assigned by v to $g(t)$, and by $v(p, t)$ or $v(t, p)$ the set of colors assigned by v to $e(p, t)$ or $e(t, p)$, respectively. For any combination of a transition $t \in \mathscr{T}^H$ and a valuation $v \in Val(t)$ with $v(t) = true$ we define the *pre-* and *postset*, by $pre(t.v) = \{p.c \in \mathscr{P}^H.ty \mid (p, t) \in \mathscr{F}^H \wedge c \in v(p, t)\}$ and $post(t.v) = \{p.c \in \mathscr{P}^H.ty \mid (t, p) \in \mathscr{F}^H \wedge c \in v(t, p)\}$, respectively. Analogously, we define for any combination of a place $p \in \mathscr{P}^H$ and a color $c \in ty(p)$ the corresponding sets $pre(p.c) = \{t.v \in \mathscr{T}^H.Val \mid (t, p) \in \mathscr{F}^H \wedge c \in v(t, p) \wedge v(t) = true\}$ and $post(p.c) = \{t.v \in \mathscr{T}^H.Val \mid (p, t) \in \mathscr{F}^H \wedge c \in v(p, t) \wedge v(t) = true\}$.

Example 3 (High-level Petri game) Figure 4 shows a high-level version of the scenario presented as low-level Petri game in Fig. 2. Here we use the set $\{\bullet\}$ as color domain for the environment and for the initial system player, the set $P = \{p_1, \ldots, p_n\}$ for the created processes, and the set $\{\div, \star\}$ for the orders of the environment. The type of the places can easily be inferred by the surrounding arc expressions, for instance, $ty(A) = P$. The type of the variables is given by $ty(p) = P$ and $ty(o) = \{\div, \star\}$. The guarding function maps every transition to *true*. Arcs without a depicted expression are by convention implicitly labeled with $\bullet$. The idea is the same as in the low-level case: The environment decides via transition *order* whether the processes should work together (token $\star$) or alone (token $\div$) and the system can decide to create them with or without this knowledge. The processes again choose to do the work jointly (transition j) or solely (transition a). In the end, the system can only win by getting informed and following the environment's order.

Since we consider set-based high-level Petri games, a *marking* M of $\mathscr{H}$ is a set $M \subseteq \mathscr{P}^H.ty$. We denote by $\mathscr{M}(\mathscr{H}) = \mathbb{P}\left(\mathscr{P}^H.ty\right)$ the set of all markings. An element $p.c \in M$ states that in marking M a player of color c resides on place p. A transition t is *enabled* in mode $v \in Val(t)$ in marking M, denoted by $M[t.v\rangle$, iff $v(t) = true$ and $pre(t.v) \subseteq M$ holds. The restriction to set-based Petri games yields that $M[t.v\rangle \Rightarrow post(t.v) \subseteq (\mathscr{P}^H.ty \backslash M) \cup pre(t.v)$ holds for every $M \in \mathscr{M}(\mathscr{H})$. The marking M' obtained after the firing of $t.v$ is computed as $M' = (M \backslash pre(t.v)) \cup post(t.v)$ and this *firing relation* is denoted by $M\,[t.v\rangle\, M'$. The game $\mathscr{H}$ has a *recurrently interfering* environment if in every infinite firing sequence

$$In^H = M_0\,[t_0.v_0\rangle\, M_1\,[t_1.v_1\rangle\, M_2[t_2.v_2\rangle \ldots$$

of $\mathscr{H}$ there are infinitely many $i \in \mathbb{N}$ with $pre(t_i.v_i) \cap \mathscr{P}^H_E.ty \neq \emptyset$.

A given set-based high-level Petri game $\mathscr{H}$ can be *transformed* into a 1-bounded P/T Petri game $\mathtt{L}(\mathscr{H}) = (\mathscr{P}_S, \mathscr{P}_E, \mathscr{T}, \mathscr{F}, In, \mathscr{B})$ with the set of system places $\mathscr{P}_S = \mathscr{P}_S^H.ty$, the set of environment places $\mathscr{P}_E = \mathscr{P}_E^H.ty$, the set of all places $\mathscr{P} = \mathscr{P}_S \cup \mathscr{P}_E = \mathscr{P}^H.ty$, the set of transitions $\mathscr{T} = \mathscr{T}^H.Val$, the flow relation $\mathscr{F} \subseteq (\mathscr{P} \times \mathscr{T}) \cup (\mathscr{T} \times \mathscr{P})$, such that $(p.c, t.v) \in \mathscr{F} \Leftrightarrow p.c \in pre(t.v)$ and $(t.v, p.c) \in \mathscr{F} \Leftrightarrow p.c \in post(t.v)$ holds for all $t.v \in \mathscr{T}$ and $p.c \in \mathscr{P}$, the initial marking $In = In^H$, and the set of bad places $\mathscr{B} = \mathscr{B}^H.ty$.

With this notation, the markings of $\mathtt{L}(\mathscr{H})$ are exactly the markings $\mathscr{M}(\mathscr{H})$ of $\mathscr{H}$. Here $p.c \in M$ for a marking M of $\mathtt{L}(\mathscr{H})$ means that a player resides on place $p.c \in \mathscr{P}$. The enabledness and firing of a transition also directly coincide. Also, $\mathscr{H}$ has a recurrently interfering environment player iff $\mathtt{L}(\mathscr{H})$ has a recurrently interfering environment player.

The *unfolding* of the high-level Petri game $\mathscr{H}$ is defined as the unfolding of $\mathtt{L}(\mathscr{H})$. Consequently, a *strategy* for the system players in $\mathscr{H}$ is defined as a strategy in $\mathtt{L}(\mathscr{H})$. By this, we know that the strategy is *winning* iff all plays conforming to the strategy avoid any bad place in $\mathscr{B} = \mathscr{B}^H.ty$.

Note that appropriately renaming the nodes of the low-level Petri game presented in Fig. 2 yields an instance of the high-level Petri game presented in Fig. 4 with $P = \{p_1, p_2\}$. Thus, Fig. 3 also shows (modulo renaming) a winning strategy for the system players in the high-level Petri game in Fig. 4. The concrete mapping is given by the following assignment: (i) $\forall x \in \{S, E_1, E_2, J, \bot\} : x \mapsto x.\bullet$, (ii) $\forall i \in \{1, 2\} : A_i \mapsto A.p_i$, $P_i \mapsto Pool.p_i$, $a_i \mapsto a.\{p \mapsto p_i\}$, $\bot_{ai} \mapsto \bot_a.\{p \mapsto p_i\}$, (iii) $C_a \mapsto C.\div$, $C_j \mapsto C.\star$, $order_a \mapsto order.\{o \mapsto \div\}$, $order_j \mapsto order.\{o \mapsto \star\}$.

Summarizing, a high-level Petri game $\mathscr{H}$ is a *succinct represention* of a detailed low-level Petri game $\mathtt{L}(\mathscr{H})$, but the semantic notions of markings, firing of transitions, unfoldings, strategies, and plays are all borrowed from $\mathtt{L}(\mathscr{H})$.

4 Solving high-level Petri games

In this section we show how to solve set-based high-level Petri games with a single recurrently interfering environment player and a bounded number of system players with a safety objective while exploiting the symmetries of the system. The key idea of the approach is the combination of two established concepts. Firstly, we use the techniques for the construction of a symbolic reachability graph (SRG) for Coloured Petri Nets with a significantly smaller size (for example presented in [8]). Secondly, we apply these techniques to the two-player game over a finite graph introduced in [24] which serves for solving a low-level Petri game with one environment player and a bounded number of system players with a safety objective. This results in a bisimilar game with a significantly smaller state space. Note that players can terminate and new players can be spawned during the game. The restriction to a *bounded* number of system players only limits the maximal number of system players in any state of the game, and not the total number of spawned and terminated players. The same applies to the restriction to a *single* environment player. Proofs that are omitted in this section can be found in "Appendix C".

Given a set-based high-level Petri game $\mathscr{H}$ with a single recurrently interfering environment player, a bounded number of system players and a safety objective, the solving algorithm proceeds in four steps:

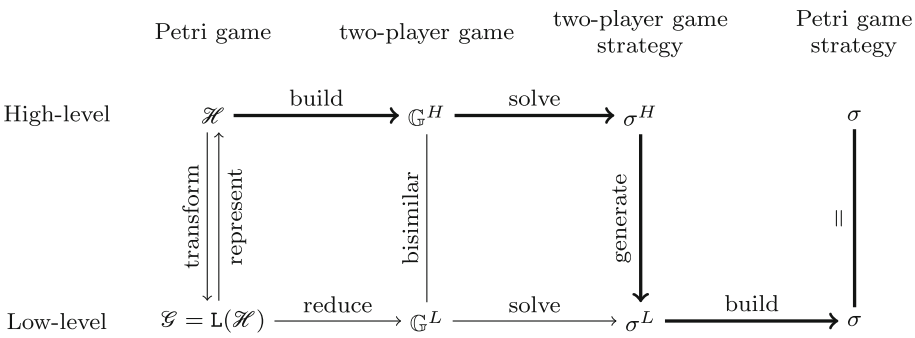

Fig. 5 The correlation of the games and strategies in the process of solving high-level and low-level Petri games. In the top of the figure the steps involving high-level elements are depicted, whereas the bottom shows the solving of low-level Petri games. The individual steps of the algorithm are marked bold. The edges between the top and the bottom layer show the relation of the high- and the low-level elements

1. The corresponding symbolic two-player game $\mathbb{G}^H$ is created with similar techniques as for the two-player game $\mathbb{G}^L$ described in [25] for a low-level Petri game. Moreover, in the case that $\mathbb{G}^L$ is created from $\mathrm{L}(\mathscr{H})$, the states of $\mathbb{G}^H$ are equivalence classes of the states of $\mathbb{G}^L$ with respect to the system's symmetries.
2. Since $\mathbb{G}^H$ is a two-player game with complete information, standard game solving algorithms are applied to gain a positional winning strategy σ^H in $\mathbb{G}^H$.
3. Resolving the symmetries of σ^H yields a winning strategy σ^L in $\mathbb{G}^L$.
4. The techniques in [25] yield a winning strategy σ in $\mathrm{L}(\mathscr{H})$ from σ^L.

Since the strategy in a high-level Petri game is defined as the strategy in the corresponding low-level Petri game, these four steps yield the strategy σ for the system players in $\mathscr{H}$. An overview of this algorithm and the interplay of the individual components is presented in Fig. 5. Note that step 3 and step 4 could be combined to obtain the Petri game strategy σ directly from the high-level two-player strategy σ^H. However, only introducing step 3 and showing its correctness yields together with [25] the same result and simplifies the presentation.

Step 1 is the crucial part of the algorithm and this section serves for its elaboration. We start by recalling the definition of $\mathbb{G}^L$ from [25] (with minor simplifications and adaptations). The definition of $\mathbb{G}^H$ is split into three parts. Firstly, we define how to apply symmetries on the states of $\mathbb{G}^L$ to obtain equivalence classes serving as states of $\mathbb{G}^H$. Secondly, we examine the interrelation of the classes to reduce the number of edges induced by $\mathbb{G}^L$. The result serves as edges of $\mathbb{G}^H$. Finally, we define the two-player game $\mathbb{G}^H$ and show the correctness of our approach by defining a bisimulation between $\mathbb{G}^L$ and $\mathbb{G}^H$, and generally proving that two bisimilar two-player Büchi games coincide regarding the existence of a winning strategy for Player 0. We start by introducing some general results and definitions for two-player games over a finite graph.

Preliminaries for two-player games: A *two-player Büchi game* is a structure $\mathbb{G} = (V, V_0, V_1, v_0, E, F)$ with the set of all *states* V, the set of *Player 0's states* V_0, the set of *Player 1's states* V_1, the *initial state* v_0, the *edge relation* E, and the set of *accepting states* F. The game is played between two players, namely Player 0 and Player 1. A *strategy* for Player i, for $i \in \{0, 1\}$, in $\mathbb{G}$ is a function $\sigma : V^* V_i \to V$ which maps each sequence of states ending in a state of Player i to some successor state, satisfying $(v, \sigma(wv)) \in E$ for all $w \in V^*$ and $v \in V_i$. A strategy σ for Player i in $\mathbb{G}$ is called *positional*, if $\sigma(wv) = \sigma(v)$ for all $w \in V^*$ and $v \in V_i$. A *play* on $\mathbb{G}$ is a possibly infinite sequence $\pi = v_0 v_1 v_2 \ldots$ of

states with $(v_j, v_{j+1}) \in E$ for all $j \in \mathbb{N}$. Player 0 wins a play if it infinitely often contains an accepting state. Otherwise, Player 1 wins. A *play* π *conforms* to a strategy σ for Player i if for all prefixes $wvv' \in V^*V_iV$ of π the strategy satisfies $\sigma(wv) = v'$. A strategy σ for Player i is *winning* if each play which conforms to σ is won by Player i.

From game theory we know that in a Büchi game $\mathbb{G}$, Player 0 has a winning strategy in $\mathbb{G}$ iff she has a *positional* winning strategy in $\mathbb{G}$. Deciding the question whether Player 0 has a winning strategy in a given Büchi game $\mathbb{G}$, and, if possible, generating a winning positional strategy, can be done in polynomial time in the number of edges in the game.

4.1 Solving low-level Petri games

In [25] Finkbeiner and Olderog reduce the problem of solving a low-level Petri game $\mathcal{G}$ with a single environment player and a bounded number of system players with a safety objective to the solving of a two-player game over a finite graph $\mathbb{G}^L$. They show that Player 0 has a winning strategy in $\mathbb{G}^L$ iff the system players have a winning strategy in $\mathcal{G}$.

In this section we simplify the definition of $\mathbb{G}^L$ for the subclass of Petri games with a single recurrently interfering environment. We define $\mathbb{G}^L$ specifically for a given low-level Petri game $\mathtt{L}(\mathcal{H}) = (\mathcal{P}_S^H.ty, \mathcal{P}_E^H.ty, \mathcal{T}^H.Val, \mathcal{F}, In, \mathcal{B}^H.ty)$ which is obtained from a set-based high-level Petri game $\mathcal{H}$ by the transformation presented in Sect. 3.2.

The general idea is that $\mathbb{G}^L$ simulates $\mathtt{L}(\mathcal{H})$ through a sequence of *decision sets*, i.e., enriched markings of $\mathtt{L}(\mathcal{H})$. In a decision set each system player is equipped with a *commitment set*, i.e., a set of transitions which are currently selected by the system player to be allowed to fire (or the special symbol $\top$). If the commitment set is $\top$, the system player has to select a new set of transitions. The key idea of the reduction is to delay the environment's moves until all future moves of each system player are dependent on the environment's decision. By this, we ensure that all system players get informed of the environment's decision during their next move and all system player's commitments, which should be made independently of the environment's decision, are made before the environment's choice. This allows for applying solving algorithms for games with complete information to $\mathbb{G}^L$.

Formally, a *decision set* is a set $D \subseteq \mathcal{P}^H.ty \times (\mathbb{P}(\mathcal{T}^H.Val) \cup \top)$ such that in a commitment set only transitions of the place's postset occur, i.e., $(p.c, C) \in D \wedge C \subseteq \mathcal{T}^H.Val \Rightarrow \forall t.v \in C : t.v \in post(p.c)$ holds. We denote the set of all decision sets by $\mathcal{D}(\mathtt{L}(\mathcal{H}))$, and define $\mathcal{M}$ to map a decision set $D \in \mathcal{D}(\mathtt{L}(\mathcal{H}))$ to its *corresponding marking* $\mathcal{M}(D) = \{p.c \mid \exists C : (p.c, C) \in D\}$.

A transition $t.v \in \mathcal{T}^H.Val$ is *enabled* in a decision set D iff $pre(t.v) \subseteq \mathcal{M}(D)$. The transition $t.v$ is *chosen* in D, denoted by $D(t.v\rangle$, iff $\forall (p.c, C) \in D : p.c \in pre(t.v) \Rightarrow t.v \in C$ holds. We call the transition $t.v$ *fireable* in D, denoted by $D[t.v\rangle$, iff $t.v$ is enabled and chosen in D. This condition is equivalent to $pre(t.v) \subseteq \mathcal{M}(D_{t.v})$, where $D_{t.v} = \{(p.c, C) \in D \mid t.v \in C\}$. This means that the transition $t.v$ not only needs to be enabled in the corresponding marking, but also that all players in the transition's preset must allow $t.v$. The decision set D' obtained after *firing* $t.v$, denoted by $D[t.v\rangle D'$, is given by $D' = \{(p.c, C) \mid (p.c, C) \in D \wedge p.c \notin pre(t.v)\} \cup \{(p.c, \top) \mid p.c \in post(t.v) \cap \mathcal{P}_S^H.ty\} \cup \{(e.d, post(e.d)) \mid e.d \in post(t.v) \cap \mathcal{P}_E^H.ty\}$. This means that the corresponding markings preserve the firing relation, i.e., $D[t.v\rangle D' \Rightarrow \mathcal{M}(D)[t.v\rangle \mathcal{M}(D')$ holds, and only the moved system players are allowed and have to decide on a new commitment set.

If a decision set D contains a $\top$ symbol, denoted by $D[\top\rangle$, the corresponding system players have to decide on a new commitment set before any other move is allowed. This

is denoted by $D[\top\rangle D'$ where D' is a decision set such that for a function $f : \mathscr{P}^H.ty \to \mathbb{P}\left(\mathscr{T}^H.Val\right)$, $D' = \{(p.c, C) \mid (p.c, C) \in D \wedge C \neq \top\} \cup \{(p.c, f(p.c)) \mid (p.c, \top) \in D\}$. We call this relation $\top$ *resolution*. The definition means that the $\top$ resolution of a decision set D yields one successor decision set for every possible combination of replacing each $\top$ in D with a possibly different commitment set $C \subseteq \mathscr{T}^H.Val$. The multiple successors are due to the several decisions the system players can make.

A decision set $D \in \mathscr{D}(\mathtt{L}(\mathscr{H}))$ can have the following properties: D is *environment-dependent* iff there is no $\top$ symbol in any tuple in D, there is a pair $(e.d, post(e.d)) \in D$ for some $e.d \in \mathscr{P}_E^H.ty$, and for all $t.v \in \mathscr{T}^H.Val$ holds that $\neg D[t.v\rangle$ or $e.d \in pre(t.v)$, D *contains a bad place* iff $\mathscr{M}(D) \cap \mathscr{B}^H.ty \neq \emptyset$, D is a *deadlock* iff there is a transition $t'.v' \in \mathscr{T}^H.Val$ such that $\mathscr{M}(D)[t'.v'\rangle$ and $\forall t.v \in \mathscr{T}^H.Val : \neg D[t.v\rangle$ holds, D is *terminating* iff $\neg\mathscr{M}(D)[t.v\rangle$ holds for all transitions $t.v \in \mathscr{T}^H.Val$, and D is *nondeterministic* iff there are two separate transitions $t_1.v_1, t_2.v_2 \in \mathscr{T}^H.Val$, with $t_1.v_1 \neq t_2.v_2$, that share a system place in their presets ($\mathscr{P}_S^H.ty \cap pre(t_1.v_1) \cap pre(t_2.v_2) \neq \emptyset$) and are both fireable in D, i.e., $D[t_1.v_1\rangle \wedge D[t_2.v_2\rangle$. Note that an mcut in a Petri game strategy corresponds to an environment-dependent decision set, i.e., all next moves of the system players are fixed (there is no $\top$ symbol in D) and each of these moves is only possible after a progress of the environment of which each player gets informed by this move.

The *game graph* for a 1-bounded Petri game $\mathtt{L}(\mathscr{H})$ with a single recurrently interfering environment player is a vertex labeled graph $\mathscr{A}^L = (\mathscr{V}^L, \mathscr{L}^L, D_0, \mathscr{E}^L)$ with

- the *vertices* $\mathscr{V}^L = \mathscr{D}(\mathtt{L}(\mathscr{H}))$,
- the *vertex labeling* $\mathscr{L}^L$, defined by $\mathscr{L}^L(D) = 1$, if D is environment-dependent, and $\mathscr{L}^L(D) = 0$ otherwise. The corresponding decision sets are collected in $\mathscr{V}_1^L = (\mathscr{L}^L)^{-1}(1)$ and $\mathscr{V}_0^L = (\mathscr{L}^L)^{-1}(0) = \mathscr{V}^L \backslash \mathscr{V}_1^L$,
- the *initial state* $D_0 = \{(p.c, \top) \mid p.c \in In \cap \mathscr{P}_S^H.ty\} \cup \{(e.d, post(e.d)) \mid e.d \in In \cap \mathscr{P}_E^H.ty\}$, i.e., the decision set containing all places of the initial marking and the system players still have to decide for a commitment set, and
- the *labeled edge relation* $\mathscr{E}^L \subseteq \mathscr{V}^L \times (\mathscr{T}^H.Val \cup \{\top\}) \times \mathscr{V}^L$ defined as follows: If D contains a bad place, is a deadlock, is terminating, or is nondeterministic, there is only a $\top$-labeled self-loop originating from D. Otherwise, we consider three disjunct cases for edges originating in D:
  Case $D \in \mathscr{V}_1^L$, i.e., all players have decided for a commitment set, but cannot proceed without the environment. Then for all $t.v \in \mathscr{T}^H.Val$, $(D, t.v, D') \in \mathscr{E}^L$ iff $D[t.v\rangle D'$.
  Case $D \in \mathscr{V}_0^L$ *and* $D[\top\rangle$, i.e., at least one system player has yet to decide for a commitment set. Then $(D, \top, D') \in \mathscr{E}^L$ iff $D[\top\rangle D'$.
  Case $D \in \mathscr{V}_0^L$ *and* $\neg D[\top\rangle$, i.e., all system players made their decisions and can proceed without the environment. Then for all $t.v \in \mathscr{T}^H.Val$ with $pre(t.v) \cap \mathscr{P}_E^H.ty = \emptyset$, $(D, t.v, D') \in \mathscr{E}^L$ iff $D[t.v\rangle D'$. The condition for $pre(t.v)$ ensures that only edges for system transitions are considered.

We define with $\mathscr{R}(\mathscr{A}^L)$ the elements in $\mathscr{V}^L$ that are reachable from D_0 under the edge relation $\mathscr{E}^L$. We finally define the *two-player Büchi game over a finite graph* $\mathbb{G}^L = (V^L, V_0^L, V_1^L, I^L, E^L, F^L)$ with the set of all *states* $V^L = \mathscr{R}(\mathscr{A}^L)$, the set of *Player 1's states* $V_1^L = \mathscr{V}_1^L \cap \mathscr{R}(\mathscr{A}^L)$, i.e., the environment-dependent decision sets, the set of *Player 0's states* $V_0^L = \mathscr{V}_0^L \cap \mathscr{R}(\mathscr{A}^L)$, the *initial state* $I^L = D_0$, as in the graph $\mathscr{A}^L$, the *edge relation* E^L with $(D, D') \in E^L$ iff $(D, \delta, D') \in \mathscr{E}^L$ for a $\delta \in \mathscr{T}^H.Val \cup \{\top\}$, and the set of *accepting states* F^L containing all $D \in V^L$ that are terminating or environment-dependent, but are not a deadlock, nondeterministic, or contain a bad place.

A *strategy* for Player i, for $i \in \{0, 1\}$, in $\mathbb{G}^L$ is a function $\sigma^L : (V^L)^* V_i^L \to V^L$ which maps each sequence of states ending in a state of Player i to some successor state. A *play* π on $\mathbb{G}^L$ is a possibly infinite sequence $\pi = v_0 v_1 v_2 \ldots$ of states with $v_0 = I^L$ and $(v_j, v_{j+1}) \in E^L$ for all $j \in \mathbb{N}$. Player 0 wins if π infinitely often contains a state $f \in F^L$. Otherwise, Player 1 wins. A *play* π *conforms* to a strategy σ^L for Player i if for all prefixes $wvv' \in (V^L)^* V_i^L V^L$ of π the strategy satisfies $\sigma^L(wv) = v'$. A strategy σ^L for Player i is *winning* if each play π which conforms to σ^L is won by Player i.

In [25], it is shown that there is a winning strategy for the system players in a P/T Petri game $\mathscr{G}$ with one environment player and a bounded number of system players if and only if there is a winning strategy for Player 0 in a two-player Büchi game over a finite graph. This proof caters for the more general case with type-2 places, i.e., where the environment player does not need to recurrently interfere. The proof rests on a link between mcuts in the strategy of the Petri game and the corresponding environment-dependent decision sets in the two-player game. For more details and insights we refer to [25]. Since the low-level Petri game $\mathtt{L}(\mathscr{H})$ considered here is a specific instance of the Petri games $\mathscr{G}$ studied in [25], the result still holds for $\mathtt{L}(\mathscr{H})$ and $\mathbb{G}^L$. In this paper, we do not reconsider this proof.

4.2 Symbolic decision sets

In [8], Chiola et al. construct a *Symbolic Reachability Graph* (SRG) for high-level Petri nets. In this graph the nodes are *equivalence classes* of markings with respect to symmetries, and instead of the ordinary firing relation between the markings, the *symbolic firing* relation is used. The following section introduces equivalence classes of decision sets with respect to symmetries and lifts results of [8] about markings (see "Appendix B") to decision sets. The representatives of the equivalence classes form the vertices of the high-level game graph $\mathscr{A}^H$. This is, analogously to the low-level case, the graph over which the symbolic two player game $\mathbb{G}^H$ is defined.

A *symmetry* $s_\mathtt{C}$ *on a color domain* $\mathtt{C} \in \mathscr{C}$ is a permutation on $\mathtt{C}$. A *symmetry* s *on a high-level Petri game* $\mathscr{H}$ is a family $(s_\mathtt{C})_{\mathtt{C} \in \mathscr{C}}$ (short $(s_\mathtt{C})_\mathtt{C}$) of symmetries on all color domains $\mathtt{C}$. Let $\mathscr{S}$ be the set of all symmetries on $\mathscr{H}$. Together with the function composition $\circ$, defined by $(s_\mathtt{C})_\mathtt{C} \circ (r_\mathtt{C})_\mathtt{C} = (s_\mathtt{C} \circ r_\mathtt{C})_\mathtt{C}$, the symmetries form a group $(\mathscr{S}, \circ)$ with $1_\mathscr{S} = (id_\mathtt{C})_\mathtt{C}$ and $(s_\mathtt{C})_\mathtt{C}^{-1} = (s_\mathtt{C}^{-1})_\mathtt{C}$. Let $s = (s_\mathtt{C})_\mathtt{C}$ be a symmetry. For any color $c \in \mathtt{C}$, $s(c)$ abbreviates $s_\mathtt{C}(c)$. The application of s to a set $A \subseteq \mathscr{P}^H\!ty$ is defined by $s(A) = \{p.s(c) \mid p.c \in A\}$ and to a valuation v by $s(v) = s \circ v$, i.e., if v assigns color c to a variable x then the valuation $s(v)$ assigns color $s_{ty(x)}(c)$ to x.

For a given high-level Petri game $\mathscr{H}$ we call a subset $S \subseteq \mathscr{S}$ of symmetries *admissible* iff

- $(S, \circ)$ is a subgroup of $(\mathscr{S}, \circ)$ such that
- $\forall s \in S\ \forall t \in \mathscr{T}^H\ \forall v \in Val(t) : v(t) = s(v)(t)$ and in the case $v(t) = true$,

  (i) $s(pre(t.v)) = pre(t.s(v))$ and
  (ii) $s(post(t.v)) = post(t.s(v))$

holds. This condition ensures that the symmetries are "compatible" with the firing of transitions: if a transition t, fireable in mode v, takes the color c from a place p (i.e., $p.c \in pre(t.v)$), then it should be fireable in mode $s(v)$ and, when fired, take color $s(c)$ from place p (i.e., $p.s(c) \in pre(t.s(v))$). The same applies to the postset of t. Hence, admissible symmetries on a high-level Petri game are those symmetries which are compatible with the game's semantics structure.

As for places, we can apply symmetries to sets of transitions $A \subseteq \mathscr{T}^H\!Val$ by $s(A) = \{t.s(v) \mid t.v \in A\}$. For a high-level Petri game $\mathscr{H}$ we fix one set S satisfying the conditions above, and call it the *set of admissible symmetries*.

Example 4 Consider the package delivery benchmark family of Fig. 1 with three packages $P = \{p_1, p_2, p_3\}$, three drones $D = \{d_1, d_2, d_3\}$, and the color domains of the places $\mathtt{C}_0 = \{\bullet\}$, $\mathtt{C}_1 = P$, $\mathtt{C}_2 = D$, and $\mathtt{C}_3 = D \times P$. Thus, every symmetry $s \in \mathscr{S}$ is of the form $s = (s_0, s_1, s_2, s_3)$, where s_i is a permutation on $\mathtt{C}_i$. This means that there is only one possibility for s_0 (namely, $id_{\{\bullet\}}$), $|P|! = 3! = 6$ possibilities for s_1, $|D|! = 3! = 6$ possibilities for s_2, and consequently there are $6 \cdot 6 = 36$ possibilities for permutations s_3 on the Cartesian product $D \times P$. Ultimately the set $\mathscr{S}$ of all symmetries contains $1 \cdot 6 \cdot 6 \cdot 36 = 1296$ elements.

We are now interested in the largest set S of admissible symmetries. Therefore, we have to consider the conditions the admissibility property imposes on symmetries. There is only one predicate not equal to *true*, namely $d \neq d'$ at transition *deliver*. For all drones $d, d' \in D$ holds that $d \neq d' \Leftrightarrow s(d) \neq s(d')$ because all symmetries $s \in \mathscr{S}$ are bijective. Thus, the predicates do not impose any restricting condition on the symmetries because $v(t) = s(v)(t)$ holds for all $t \in \mathscr{T}^H$, $v \in Val(t)$, and $s \in \mathscr{S}$.

Consider now the transition *deliver*. A valuation $v \in Val(deliver)$ assigns to all variables in $\mathsf{var}(deliver)$, namely d, d', and p, a value of the corresponding type. We denote $\mathsf{d} = v(d) \in D$, $\mathsf{d}' = v(d') \in D$, and $\mathsf{p} = v(p) \in P$. For $\mathsf{d} \neq \mathsf{d}'$ the transition *deliver* in mode v takes a token $(\mathsf{d}', \mathsf{p}) \in D \times P$ from the place *Fly* when firing. Condition (i) for the admissibility of symmetries imposes, regarding the arc expression (d', p), that $s_3(\mathsf{d}', \mathsf{p}) = (s_2(\mathsf{d}'), s_1(\mathsf{p}))$ has to hold for every $s \in S$ and all values d' and p that a valuation can assign to d' and p. Thus, s_3 is determined by the choice of s_1 and s_2. The condition (i) for the admissibility of symmetries regarding the arc expression d and condition (ii) for all corresponding arc expressions do not impose any restrictions on the symmetries because of the structure of the arc expressions. The conditions regarding all other transitions of the example do not introduce any other restrictions on the set of admissible symmetries. This means the admissible symmetries S consists of the 36 symmetries of the form $s = (id_{\{\bullet\}}, s_1, s_2, s_3)$ such that $s_3(d, p) = (s_2(d), s_1(p))$ for all $(d, p) \in D \times P$. □

We now apply symmetries to decision sets and show that their properties are invariant under the application. This means all decision sets in an equivalence class with respect to the admissible symmetries have the same properties. The representatives of these equivalence classes are called *symbolic decision sets*.

Due to the special syntax of $\mathtt{L}(\mathscr{H})$ for a high-level Petri game $\mathscr{H}$, and since both games have the same semantics, we define the decision sets of $\mathscr{H}$ as the decision sets of $\mathtt{L}(\mathscr{H})$, i.e., $\mathscr{D}(\mathscr{H}) = \mathscr{D}(\mathtt{L}(\mathscr{H}))$. For a decision set $D \in \mathscr{D}(\mathscr{H})$ and any symmetry $s \in S$ we define the *application of a symmetry to a decision set* by

$$s(D) = \{(p.s(c), s(C)) \mid (p.c, C) \in D\},$$

with $s(C) = \{t.s(v) \mid t.v \in C\}$ if $C \subseteq \mathscr{T}^H\!Val$ and $s(\top) = \top$ otherwise. This means if a player of color c on place p allows transition t in mode v in the decision set D (i.e., $(p.c, C) \in D$ and $t.v \in C$), then after the application of the symmetry s, the player of color $s(c)$ on place p allows transition t in mode $s(v)$ (i.e., $(p.s(c), s(C)) \in s(D)$ and $t.s(v) \in s(C)$). Since symmetries operate on the first coordinate of a decision set exactly as on markings, we obtain $s(\mathscr{M}(D)) = \mathscr{M}(s(D))$.

Two decision sets D and D' are *equivalent* iff there is an admissible symmetry $s \in S$ such that $s(D) = D'$ holds. This leads to the set of equivalence classes $\mathscr{D}(\mathscr{H})/S$ of the decision

sets. For a decision set $D \in \mathscr{D}(\mathscr{H})$, we denote an equivalence class in $\mathscr{D}(\mathscr{H})/S$ by $[D]$. We define $\overline{D} \in [D]$ as an arbitrarily chosen, but fixed representative of $[D]$. The representatives $\overline{D}$ are called *symbolic decision sets*. We fix with s_D a symmetry that maps a decision set D to its corresponding representative, i.e., $s_D(D) = \overline{D}$. Thus, the admissible symmetries on a high-level Petri game are a tool to transform equivalent situations into each other.

As a first property we consider the interplay of symmetries and the relations between decision sets.

Lemma 1 *The admissible symmetries are compatible with the firing of a transition $t \in \mathscr{T}^H$ in mode $v \in Val(t)$ in a decision set. The same is true for the resolution of a $\top$ symbol in a decision set.*

- $\forall t \in \mathscr{T}^H \; \forall v \in Val(t) \; \forall s \in S : D[t.v\rangle D' \Leftrightarrow s(D)[t.s(v)\rangle s(D')$.
- $\forall s \in S : D[\top\rangle D' \Leftrightarrow s(D)[\top\rangle s(D')$.

Both results follow from the admissibility of a symmetry s and from the equations $s(\mathscr{M}(D)) = \mathscr{M}(s(D))$ and $s(D_{t.v}) = s(D)_{t.v}$ (see "Appendix C").

From now on we assume w.l.o.g. that the initial marking In^H of a high-level Petri game $\mathscr{H}$ is *symmetric*, i.e., $s(In^H) = In^H$ for all $s \in S$. If this is not the case, we add an new transition t_0 with a single new environment place p_0 in its preset such that the firing of t_0 generates In^H or an equivalent marking. This way, the admissible symmetries remain unchanged. The new initial marking only consists of a fresh colored token c_0 residing on p_0 with a fresh singleton color domain and is thus trivially symmetric. For the explicit construction, see "Appendix B". Most examples, like the ones in Sects. 2 and 3.2, directly have a symmetric initial marking.

This assumption allows us to show that the following properties of a decision set are preserved by the application of admissible symmetries.

Lemma 2 *Let $D \in \mathscr{D}(\mathscr{H})$ and $s \in S$. Then D is environment-dependent, contains a bad place, is a deadlock, is terminating or is nondeterministic if and only if $s(D)$ has the same property.*

These properties can be proven by using Lemma 1 and the following facts about symmetries (see "Appendix C"). An admissible symmetry $s \in S$ applied to a set $A.ty$ with $A \subseteq \mathscr{P}^H$ leaves the set unchanged, i.e., $s(A.ty) = A.ty$. This also holds for a set $A.Val$ with $A \subseteq \mathscr{T}^H$. The application of s is compatible with intersections of sets $A, B \subseteq \mathscr{P}^H.ty$ or sets $A, B \in \mathscr{T}^H.Val$, i.e., $s(A \cap B) = s(A) \cap s(B)$.

Lemma 2 yields the uniform satisfaction of these properties throughout the complete equivalence class.

Corollary 1 *Let D be a decision set. Then D has one of the properties listed above if and only if all $D' \in [D]$ (and in particular $\overline{D}$) have the same property.*

The representatives $\overline{D}$ of the decision sets $D \in \mathscr{V}^L$ of the low-level game graph $\mathscr{A}^L$ form the vertices of the high-level game graph $\mathscr{A}^H$. Here you can already feel the spirit of the symbolic reachability graph SRG, where the nodes are *symbolic* markings $\overline{M}$ instead of ordinary markings M as in the reachability graph RG.

Usually, the relation on equivalence classes is given by all connections between the individual elements of the corresponding classes. Lemma 1 shows that for equivalence classes of decision sets, we only have to consider connections where the source is a representative of the class. The next section reduces this relation even further, by only considering *equivalence classes of firings*, local to the source decision set.

4.3 Symbolic firing and symbolic ⊤ resolution

In this section we define equivalence classes of transition firings to define the edge relation of the high-level game graph $\mathscr{A}^H$. In general, this relation is smaller than the relation containing an edge for every possible transition firing or ⊤ resolution between the corresponding equivalence classes of decision sets. Again, results of [8] are lifted from markings to decision sets.

Example 5 Consider the scenario of the package delivery of Fig. 1 with three packages $P = \{p_1, p_2, p_3\}$ and three drones $D = \{d_1, d_2, d_3\}$. Assume the packages assigned themselves to the drones according to their index, and the drones, loaded with their corresponding cargo, took off, i.e., the three tokens (d_1, p_1), (d_2, p_2), and (d_3, p_3) reside on the place *Fly*. Further assume that the environment decided that drone d_1 is defective via transition *destroy* in mode $v = \{d \mapsto d_1\}$, i.e., a token d_1 resides on *Malfunction*. The corresponding marking is $\{Fly.(d_1, p_1), Fly.(d_2, p_2), Fly.(d_3, p_3), Malfunction.d_1\}$.

Since the system players do not know which drone got destroyed and must not deadlock, they should allow both transitions *crash* and *deliver* in all possible modes to win the game. The environment player d_1 on *Malfunction* as usual also allows both transitions in all modes. We denote the modes $u \in Val(crash)$ by $u_{(i,j)} = \{d \mapsto d_i,\ p \mapsto p_j\}$, and the modes $v \in Val(deliver)$ by $v_{(i,j,k)} = \{d' \mapsto d_i,\ p \mapsto p_j,\ d \mapsto d_k\}$.

We abbreviate $crash.u_{(i,j)}$ by $cr_{(i,j)}$ and $deliver.v_{(i,j,k)}$ by $del_{(i,j,k)}$. The decision set described above is

$$\begin{aligned} D_1 = \big\{ & (Fly.(d_1, p_1), \{cr_{(1,1)}, del_{(1,1,2)}, del_{(1,1,3)}\}), \\ & (Fly.(d_2, p_2), \{cr_{(2,2)}, del_{(2,2,1)}, del_{(2,2,3)}\}), \\ & (Fly.(d_3, p_3), \{cr_{(3,3)}, del_{(3,3,1)}, del_{(3,3,2)}\}), \\ & (Malfunction.d_1, \{cr_{(1,j)}, del_{(i,j,1)} \mid i = 2, 3 \wedge j = 1, 2, 3\})\big\}. \end{aligned}$$

The fireable transitions from this decision set are $cr_{(1,1)}$, $del_{(2,2,1)}$, and $del_{(3,3,1)}$. This means, when the relation on decision sets is lifted to equivalence classes by collecting all connections between individual elements of the corresponding classes, these three transitions all induce an outgoing edge from $[D_1]$. This is illustrated in Fig. 6 where $[D_1]$ is depicted in the middle of the figure. The decision sets obtained after the respective firing are D_2, D_3, and D_4. Since D_2 and D_3 are in the same equivalence class, this results in a connection between $[D_1]$ and $[D_2]$ for $del_{(2,2,1)}$ and $del_{(3,3,1)}$ each.

Consider now the symmetry $s \in S$ that swaps p_2 and p_3 in P, swaps d_2 and d_3 in D, and accordingly operates on the Cartesian product $P \times D$. This symmetry leaves D_1 invariant. When applied to $del_{(2,2,1)}$ or $del_{(3,3,1)}$, the respective other transition is obtained. We call these two transitions therefore equivalent with respect to D_1 (since the symmetry s leaves D_1 invariant). Instead of considering both firings, we chose one of these transitions to represent both firings. □

From Lemma 1 we see that, if an admissible symmetry $s \in S$ leaves a decision set D invariant, then a transition t is fireable in mode $v \in Val(t)$ at D if and only if t is fireable in mode $s(v)$. These symmetries form a group (later called the isotropy group of D) and their application leads to equivalence classes of valuations which are locally belonging to the decision set. Instead of considering all valuations in which a transition is fireable from a decision set, it suffices to consider representatives of these equivalence classes. As a result, the size of the firing relation between equivalence classes of decision sets decreases. This

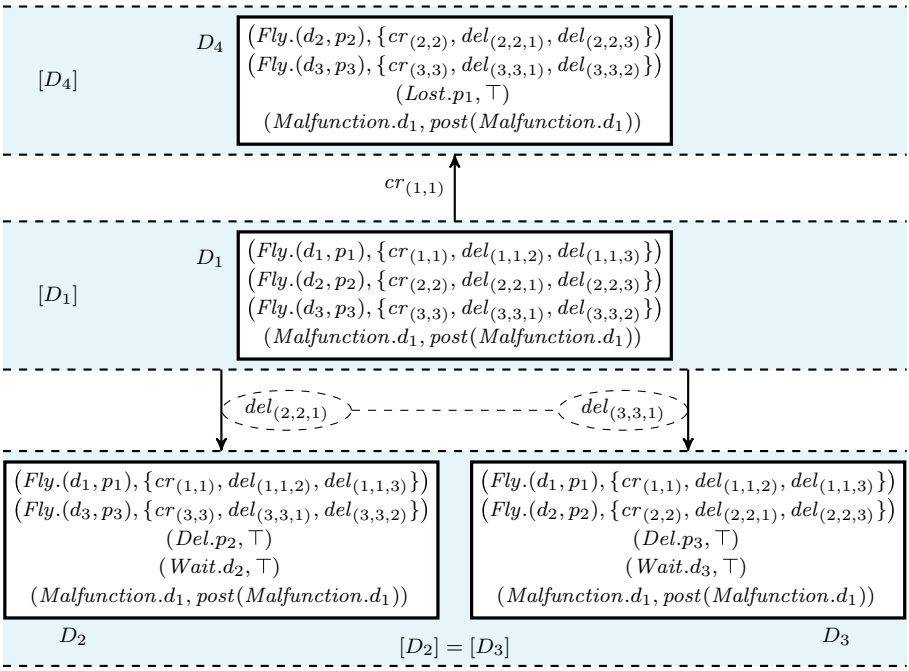

Fig. 6 An illustration of equivalences of transition firings. The decision set D_1 (depicted in the middle) has three successor decision sets according to the standard firing relation: D_2, D_3, which belong to the same symbolic decision set (depicted at the bottom), and D_4 (depicted at the top). Since the transitions $del_{(2,2,1)}$ and $del_{(3,3,1)}$ can be mapped to one another via a symmetry not affecting D_1, the relation between $[D_1]$ and $[D_2]$ can be represented by only one of them. The equivalence of the transitions is depicted by the dashed connection in the middle

reduced firing relation, called the symbolic firing relation, is the first part of the edge relation of the high-level game graph $\mathscr{A}^H$.

However, considering a symbolic decision set, after firing a transition in a representative of an equivalence class of valuations, the decision set obtained after the firing does not have to be a symbolic decision set itself. Since the symbolic firing relation will be defined between symbolic decision sets, this fact must be taken into account when defining the relation.

Additionally to the firing relation there is the relation of $\top$ resolution between decision sets. Thus, we also define the *symbolic* $\top$ resolution between *symbolic* decision sets. This relation forms the rest of the edge relation of $\mathscr{A}^H$.

Let $D \in \mathscr{D}(\mathscr{H})$ be a decision set. The *isotropy group* $S_D = \{s \in S \mid s(D) = D\}$ of D is the group of all admissible symmetries that preserve D. For a transition $t \in \mathscr{T}^H$, we denote by $Val(t)_D = Val(t)/S_D$ the set containing the equivalence classes of all modes of t, with respect to the isotropy group S_D. The individual modes in one class affect D in symmetric ways. For each class in $Val(t)_D$ we arbitrarily chose a representative mode $\overline{v}$ and define α_D as the function mapping each $v \in Val(t)$ to its representative $\alpha_D(v)$.

Note that for every representative $\overline{v}$ of a class in $Val(t)_D$, for every mode v belonging to $[\overline{v}]$, there is a symmetry $s \in S_D$ such that $s(v) = \alpha_D(v) = \overline{v}$. This means that, in a decision set D, a transition can fire in mode v if and only if it can fire in its representative $\alpha_D(v)$.

We now define the symbolic firing relation between symbolic decision sets. For that, instead of firing a transition in all modes, we only consider the representatives of equivalence

classes of modes, local to the symbolic decision set. The symbolic decision set obtained after the *symbolic* firing is corresponding to the decision set obtained after the *ordinary* firing of the transition in the representative mode.

We say a transition t can *fire symbolically* from the symbolic decision set $\overline{D}$ in mode $\alpha_{\overline{D}}(v)$ representing v in $Val(t)_{\overline{D}}$, denoted by $\overline{D}[\![t.\alpha_{\overline{D}}(v)\rangle\!\rangle$, iff $\overline{D}[t.\alpha_{\overline{D}}(v)\rangle$. The symbolic decision set $\overline{D'}$ obtained after the symbolic firing is determined as follows:

$$\overline{D}[\![t.\alpha_{\overline{D}}(v)\rangle\!\rangle\overline{D'} \Leftrightarrow \exists D'' \in [\,\overline{D'}\,] : \overline{D}[t.\alpha_{\overline{D}}(v)\rangle D''.$$

To define the symbolic $\top$ resolution between symbolic decision sets, we can not use representatives of the symbol $\top$. Instead, when symbolically resolving a $\top$ symbol in a symbolic decision set, we declare the possible targets as the representatives of possible targets of an ordinary $\top$ resolution.

We say a $\top$ can be *symbolically resolved* in a symbolic decision set $\overline{D}$, denoted by $\overline{D}[\![\top\rangle\!\rangle$, iff $\overline{D}[\top\rangle$. The possible symbolic decision sets obtained after the symbolic $\top$ resolution are the representatives of the decision sets D'' satisfying $\overline{D}[\top\rangle D''$:

$$\overline{D}[\![\top\rangle\!\rangle\overline{D'} \Leftrightarrow \exists D'' \in [\,\overline{D'}\,] : \overline{D}[\top\rangle D''.$$

In the following properties we compare the ordinary firing relation and the ordinary $\top$ resolution with their symbolic counterparts.

Property 1 Each ordinary transition firing is represented by a symbolic transition firing, and each ordinary $\top$ resolution is represented by a symbolic one.

- $D[t.v\rangle D' \Rightarrow \overline{D}[\![t.\overline{v}\rangle\!\rangle\overline{D'}$, where $\overline{v} = \alpha_{\overline{D}}(s_D(v))$.
- $D[\top\rangle D' \Rightarrow \overline{D}[\![\top\rangle\!\rangle\overline{D'}$.

The first property can be shown with the help of Lemma 1, analogously to [8]. The proof of the second property has the same structure.

Property 2 Each symbolic firing represents a set of ordinary firings, in which all source decision sets belong to the equivalence class of the symbolic source decision set of the symbolic firing. The same holds for the resolution of $\top$.

- $\overline{D}[\![t.\overline{v}\rangle\!\rangle\overline{D'} \Rightarrow$
  $(\forall D_1 \in [\,\overline{D}\,]\ \forall v' \in Val(t) : \alpha_{\overline{D}}(s_{D_1}(v')) = \overline{v} \Rightarrow \exists D_2 \in [\,\overline{D'}\,] : D_1[t.v'\rangle D_2.)$
- $\overline{D}[\![\top\rangle\!\rangle\overline{D'} \Rightarrow \forall D_1 \in [\,\overline{D}\,]\ \exists D_2 \in [\,\overline{D'}\,] : D_1[\top\rangle D_2$.

Again, the first property can be shown analogously to [8] using Lemma 1, and the proof of the second property uses the same ideas.

4.4 Symbolic two-player game

In this section we define the *high-level game graph* $\mathscr{A}^H$ and, based on its structure, the *symbolic two-player game* $\mathbb{G}^H$. We show that Player 0 has a winning strategy in $\mathbb{G}^H$ if and only if there is a winning strategy for Player 0 in the low-level two-player game $\mathbb{G}^L$ that corresponds to $\mathtt{L}(\mathscr{H})$. This is proven by introducing a bisimulation on the two-player games. We fix a set-based high-level Petri game with a single recurrently interfering environment player and a bounded number of system players $\mathscr{H} = (\mathscr{P}_S^H, \mathscr{P}_E^H, \mathscr{T}^H, \mathscr{F}^H, ty, g, e, In^H, \mathscr{B}^H)$ throughout the section.

Remember that the vertices of $\mathscr{A}^L$ are decision sets and an edge between two decision sets D and D' only exists if $D[t.v\rangle D'$ or $D[\top\rangle D'$ holds. We analogously define the high-level game graph $\mathscr{A}^H$ by considering the symbolic counterparts. This means, the vertices of $\mathscr{A}^H$ are the symbolic decision sets and there is an edge between two symbolic decision sets $\overline{D}$ and $\overline{D'}$ iff $\overline{D}[\![t.\overline{v}\rangle\!\rangle\overline{D'}$ or $\overline{D}[\![\top\rangle\!\rangle\overline{D'}$ holds. Note that we represent an equivalence class $[D]$ with respect to S by the symbolic decision set $\overline{D}$ when no confusion arises.

For the high-level Petri game $\mathscr{H}$, we define the vertex labeled *high-level game graph* $\mathscr{A}^H = (\mathscr{V}^H, \mathscr{L}^H, \overline{D_0}, \mathscr{E}^H)$ with

- the *vertices* $\mathscr{V}^H = \mathscr{D}(\mathscr{H})/S$, the set of all equivalence classes $[D]$ with respect to S,
- the *vertex labeling* $\mathscr{L}^H$, defined by $\mathscr{L}^H(\overline{D}) = 1$, if $\overline{D}$ is environment-dependent, and $\mathscr{L}^H(\overline{D}) = 0$ otherwise. The corresponding symbolic decision sets are collected in $\mathscr{V}_1^H = (\mathscr{L}^H)^{-1}(1)$ and $\mathscr{V}_0^H = \mathscr{V}^H \setminus \mathscr{V}_1^H$
- the *initial state* $\overline{D_0}$ with
  $D_0 = \{(p.c, \top) \mid p.c \in In^H \cap \mathscr{P}_S^H.ty\} \cup \{(e.d, post(e.d)) \mid e.d \in In^H \cap \mathscr{P}_E^H.ty\}$,
- the *labeled edge relation* $\mathscr{E}^H \subseteq \mathscr{V}^H \times (\mathscr{T}^H.Val \cup \{\top\}) \times \mathscr{V}^H$ is defined as follows: If $\overline{D}$ contains a bad place, is a deadlock, is terminating, or is nondeterministic, there is only a $\top$-labeled self-loop originating from $\overline{D}$. Otherwise, consider three disjunct cases for edges originating in $\overline{D}$:
  Case $\overline{D} \in \mathscr{V}_1^H$, i.e., all players have decided for a commitment set, but cannot proceed without the environment. Then for all $t \in \mathscr{T}^H$ and $\overline{v} \in Val(t)_D$, $(\overline{D}, t.\overline{v}, \overline{D'}) \in \mathscr{E}^H$ iff $\overline{D}[\![t.\overline{v}\rangle\!\rangle\overline{D'}$.
  Case $\overline{D} \in \mathscr{V}_0^H$ *and* $\overline{D}[\![\top\rangle\!\rangle$, i.e., at least one system player has yet to decide for a commitment set. Then $(\overline{D}, \top, \overline{D'}) \in \mathscr{E}^H$ iff $\overline{D}[\![\top\rangle\!\rangle\overline{D'}$.
  Case $\overline{D} \in \mathscr{V}_0^H$ *and* $\neg D[\![\top\rangle\!\rangle$, i.e., all system players made their decisions and can proceed without the environment. Then for all $t \in \mathscr{T}^H$ and $\overline{v} \in Val(t)_D$ with $pre(t.\overline{v}) \cap \mathscr{P}_E^H.ty = \emptyset$, $(\overline{D}, t.\overline{v}, \overline{D'}) \in \mathscr{E}^L$ iff $\overline{D}[\![t.\overline{v}\rangle\!\rangle\overline{D'}$.

Note that the labeling of the representatives is identical to the labeling in the low-level case and since the initial marking In^H is symmetric, $\overline{D_0} = D_0$ holds.

The symbolic two-player Büchi game $\mathbb{G}^H$ is defined on the structure of $\mathscr{A}^H$, as the definition of $\mathbb{G}^L$ is based on $\mathscr{A}^L$. This means that the states in $\mathbb{G}^H$ are the reachable *symbolic* decision sets, and there is an edge between two states, if these states are *symbolically* related.

Let $\mathscr{SR}(\mathscr{A}^H)$ be the set of vertices in $\mathscr{V}^H$ that are reachable from $\overline{D_0}$ under $\mathscr{E}^H$. The *high-level two player Büchi game over a finite graph* (or *symbolic Büchi game*) $\mathbb{G}^H = (V^H, V_0^H, V_1^H, \overline{D_0}, E^H, F^H)$ is defined with

- the set of all *states* $V^H = \mathscr{SR}(\mathscr{A}^H)$,
- the set of *Player 1's states* $V_1^H = \mathscr{V}_1^H \cap \mathscr{SR}(\mathscr{A}^H)$, i.e., the symbolic decision sets that are environment-dependent,
- the set of *Player 0's states* $V_0^H = \mathscr{V}_0^H \cap \mathscr{SR}(\mathscr{A}^H)$,
- the *initial state* $\overline{D_0}$, as in the high-level graph $\mathscr{A}^H$,
- the *edge-relation* E^H such that $(\overline{D}, \overline{D'}) \in E^H$ iff $(\overline{D}, \overline{\delta}, \overline{D'}) \in \mathscr{E}^H$ for any $\overline{\delta} \in \mathscr{T}^H.Val \cup \{\top\}$, and
- the set of *accepting states* F^H, containing all $\overline{D} \in V^H$ that are terminating or environment-dependent, but are not a deadlock, nondeterministic, or contain a bad place.

To show that the two-player games $\mathbb{G}^L$ and $\mathbb{G}^H$ are bisimilar (which yields the correctness of our approach), we compare the structures of $\mathscr{A}^H$ and $\mathscr{A}^L$. First, the edge relations of the two graphs correspond to each other (Lemma 3). Second, the set of reachable symbolic decision sets in $\mathscr{A}^H$ is exactly the set of representatives of decision sets reachable in $\mathscr{A}^L$ (Lemma 4).

Lemma 3 *For every edge in $\mathscr{A}^L$, there is a corresponding edge in $\mathscr{A}^H$, and vice versa:*

1. $(D, \delta, D') \in \mathscr{E}^L \Rightarrow (\overline{D}, \overline{\delta}, \overline{D'}) \in \mathscr{E}^H$, *where* $\overline{\delta} = t.\overline{v}$ *if* $\delta = t.v$ *and* $\overline{v} = \alpha_{\overline{D}}(s_D(v))$, *and* $\overline{\delta} = \top$ *if* $\delta = \top$.
2. $(\overline{D}, \overline{\delta}, \overline{D'}) \in \mathscr{E}^H \Rightarrow \forall D_1 \in [\,\overline{D}\,]\, \exists D_2 \in [\,\overline{D'}\,]\, \exists \delta : (D_1, \delta, D_2) \in \mathscr{E}^L$, *where* $\delta = t.v'$ *if* $\overline{\delta} = t.\overline{v}$ *and* v' *satisfies* $\alpha_{\overline{D}}(s_{D_1}(v')) = \overline{v}$, *and* $\delta = \top$ *if* $\overline{\delta} = \top$.

Since the self-loops of the edge relations, as well as the assignment of the vertices, depend on the properties of the (symbolic) decision sets and the other edges of the relations are induced by the (symbolic) firing relation and the (symbolic) $\top$ resolution, the proof mainly depends on Corollary 1 and on Properties 1 and 2 (see "Appendix C").

Lemma 4 *The representatives of decision sets in $\mathscr{R}(\mathscr{A}^L)$ are exactly the symbolic decision sets in $\mathscr{SR}(\mathscr{A}^H)$:* $\{\overline{D} \mid D \in \mathscr{R}(\mathscr{A}^L)\} = \{\overline{D} \mid [\,D\,] \in \mathscr{SR}(\mathscr{A}^H)\}$.

This lemma can be proven by an induction over the length of the shortest path from D_0 to a decision set D in $\mathscr{A}^L$, or from $\overline{D_0}$ to a symbolic decision set $\overline{D}$ in $\mathscr{A}^H$, respectively. The induction step follows from Lemma 3 (see "Appendix C").

We generally define a bisimulation relation on Büchi games and show that two bisimilar Büchi games coincide on the existence of a winning strategy. The instantiation of this result for the low-level two-player game $\mathbb{G}^L$ and the symbolic high-level game $\mathbb{G}^H$ yields the correctness of the main step of the solving algorithm for high-level Petri games.

We can view any Büchi game $\mathbb{G} = (V, V_0, V_1, v_0, E, F)$ as a *state-labeled transition system* (V, E, λ, v_0) with the set of states V, the transition relation E, and the initial state v_0 as defined in $\mathbb{G}$, and a labeling function $\lambda : V \to \mathbb{P}(\{g, f\})$ with propositions $\{g, f\}$, defined by $\forall v \in V : (g \in \lambda(v) \Leftrightarrow v \in V_0) \wedge (f \in \lambda(v) \Leftrightarrow v \in F)$. A *bisimulation* between two state-labeled transition systems $TS_1 = (\mathscr{S}_1, \to_1, \lambda_1, s_0)$ and $TS_2 = (\mathscr{S}_2, \to_2, \lambda_2, t_0)$ is a relation $B \subseteq \mathscr{S}_1 \times \mathscr{S}_2$ such that for all $(s, t) \in B$

- $\lambda_1(s) = \lambda_2(t)$,
- $\exists s' \in \mathscr{S}_1 : s \to_1 s' \Rightarrow \exists t' \in \mathscr{S}_2 : t \to_2 t' \wedge (s', t') \in B$, and
- $\exists t' \in \mathscr{S}_2 : t \to_2 t' \Rightarrow \exists s' \in \mathscr{S}_1 : s \to_1 s' \wedge (s', t') \in B$

holds. Two states $s \in \mathscr{S}_1$ and $t \in \mathscr{S}_2$ are called *bisimilar*, denoted by $s \sim t$, if there is a bisimulation B between TS_1 and TS_2 satisfying $(s, t) \in B$. The transition systems TS_1 and TS_2 are called *bisimilar*, denoted by $TS_1 \sim TS_2$, if $s_0 \sim t_0$.

Two Büchi games $\mathbb{G} = (V, V_0, V_1, v_0, E, F)$ and $\mathbb{G}' = (V', V_0', V_1', v_0', E', F')$ are *bisimilar*, denoted by $\mathbb{G} \sim \mathbb{G}'$, if the corresponding transition systems are bisimilar, i.e., $(V, E, \lambda, v_0) \sim (V', E', \lambda', v_0')$. Particularly, this means that for any such bisimulation B and every two states $v \in V$ and $v' \in V'$ with $(v, v') \in B$ the assignment of the states coincide, i.e., $v \in V_0 \Leftrightarrow v' \in V_0'$ and $v \in F \Leftrightarrow v' \in F'$ holds.

Lemma 5 *There is a bisimulation between $\mathbb{G}^L$ and $\mathbb{G}^H$.*

The bisimulation $B \subseteq V^L \times V^H$ is given by $B = \{(D, \overline{D}) \mid D \in V^L\}$. Lemma 4 shows that this relation is defined, Lemma 3 and Corollary 1 are used to show that B is in fact a bisimulation (see "Appendix C").

Lemma 6 *Let $\mathbb{G} = (V, V_0, V_1, v_0, E, F)$ and $\mathbb{G}' = (V', V_0', V_1', v_0', E', F')$ be two bisimilar Büchi games. Then Player 0 has a winning strategy in $\mathbb{G}$ if and only if Player 0 has a winning strategy in $\mathbb{G}'$.*

Proof For inductively defining a strategy on sequences of length n in a Büchi game, it suffices to only define it on paths that are consistent with the so far defined strategy. All other sequences are mapped to an arbitrary successor.

Let $B \subseteq V \times V'$ be the bisimulation between $\mathbb{G}$ and $\mathbb{G}'$ and σ a winning strategy for Player 0 in $\mathbb{G}$. We construct a winning strategy σ' for Player 0 in $\mathbb{G}'$ from σ. We define σ' inductively on paths of length n through the arena. For that we, also inductively, define a helper mapping τ, that maps paths of length $n+1$ in $\mathbb{G}'$ that are consistent with σ' to corresponding paths in $\mathbb{G}$.

The construction will ensure that, for all n,

- τ is defined for all paths of length $n+1$ consistent with σ' such that the image is consistent with σ,
- the states are *pairwise bisimilar*, i.e., if $\tau(v'_0 \dots v'_n) = v_0 \dots v_n$ then $(v_j, v'_j) \in B$ for all $0 \leq j \leq n$,
- σ' is defined for all consistent paths of length n that end in a state of V'_0.

(IB) Consider the case $n = 0$. Define $\tau(v'_0) = v_0$, and σ' is undefined since there are no paths of length 0.
(IH) Assume now, for an arbitrary n, that σ' is defined for all consistent paths of length n and τ is defined for all paths of length $n+1$ consistent with σ'.
(IS) Consider a path $v'_0 \dots v'_n$ of length $n+1$ in $\mathbb{G}$ that is consistent with σ'. Let $v_0 \dots v_n = \tau(v'_0 \dots v'_n)$.
Case $v'_n \in V'_0$. We define $\sigma'(v'_0 \dots v'_n)$ as follows: since $(v_n, v'_n) \in B$, we have that $v_n \in V_0$. Let now $v_{n+1} = \sigma(v_0 \dots v_n)$. This implies $(v_n, v_{n+1}) \in E$ and therefore, since B is a bisimulation, there is a $v'_{n+1} \in V'$ such that $(v'_n, v'_{n+1}) \in E'$ and $(v_{n+1}, v'_{n+1}) \in B$. We define $\sigma'(v'_0 \dots v'_n) = v'_{n+1}$ and $\tau(v'_0 \dots v'_n v'_{n+1}) = v_0 \dots v_n v_{n+1}$.
Case $v'_n \in V'_1$. We define, for every $v' \in V'$ with $(v'_n, v') \in E'$, $\tau(v'_0 \dots v'_n v') = v_0 \dots v_n v$ for an arbitrary v such that $(v_n, v) \in E$ and $(v, v') \in B$.

Let $\pi' = v'_0 v'_1 v'_2 \dots$ be a play in $\mathbb{G}'$ that is consistent with σ'. By defining v_j as the last element in $\tau(v'_0 \dots v'_j)$ for every $j \geq 0$, we obtain a play $\pi = v_0 v_1 v_2 \dots$ in $\mathbb{G}$ that is consistent with σ. Therefore, Player 0 wins π in $\mathbb{G}$, and since for all j, $v_j \in F$ iff $v'_j \in F'$, Player 0 wins π' in $\mathbb{G}'$.

Since $B^{\mathsf{T}} = \{(v', v) \mid (v, v') \in B\}$ is a bisimulation between $\mathbb{G}'$ and $\mathbb{G}$, the converse direction follows analogously. □

Lemma 6 together with Lemma 5 yields the conformity of the symbolic high-level game $\mathbb{G}^H$ and the corresponding low-level game $\mathbb{G}^L$ regarding the existence of a winning strategy.

Lemma 7 *Player 0 of $\mathbb{G}^H$ has a winning strategy in $\mathbb{G}^H$ if and only if Player 0 of $\mathbb{G}^L$ has a winning strategy in $\mathbb{G}^L$.*

The construction of a winning strategy in the proof of Lemma 6 yields a *nonpositional* strategy $\sigma^L : (V^L)^* V_0^L \to V^L$ in $\mathbb{G}^L$ for a strategy σ^H in $\mathbb{G}^H$. For the introduced solving algorithm of high-level Petri games we are interested in positional winning strategies. The following construction serves for the creation of a *positional* strategy $\sigma^L : V_0^L \to V^L$ in the low-level game $\mathbb{G}^L$ from a positional strategy $\sigma^H : V_0^H \to V^H$ in the symbolic high-level game $\mathbb{G}^H$.

Let $B = \{(D, \overline{D}) \mid D \in V^L\}$ be the bisimulation on $\mathbb{G}^L$ and $\mathbb{G}^H$, $\sigma^H : V_0^H \to V^H$ a positional winning strategy for Player 0 in $\mathbb{G}^H$ and $D \in V_0^L$. Then $\overline{D} \in V_0^H$ and $\sigma^H(\overline{D})$ is defined. Let $\overline{D'} = \sigma^H(\overline{D})$. This implies $(\overline{D}, \overline{D'}) \in E^H$, and since $(D, \overline{D}) \in B$, we have

$$\exists D_1 \in V^L : (D, D_1) \in E^L \wedge (D_1, \overline{D'}) \in B \text{ (i.e., } \overline{D_1} = \overline{D'}).$$

We define $\sigma^L(D) = D_1$. Hence, σ^L is positional. The strategy σ^L is also winning: Let $\rho = D_0 D_1 \cdots \in (V^L)^\omega$ be a play of $\mathbb{G}^L$ that is consistent with σ^L. Then $\overline{\rho} := \overline{D_0}\,\overline{D_1} \cdots \in (V^H)^\omega$ is a play of $\mathbb{G}^H$, that by definition is consistent with σ^H. Therefore, $\overline{\rho}$ is winning in $\mathbb{G}^H$. This, as in the proof of Lemma 6, implies that ρ is winning in $\mathbb{G}^L$.

Since the definition of a winning strategy in a high-level Petri game $\mathscr{H}$ is defined as a winning strategy in the corresponding low-level Petri game $\mathtt{L}(\mathscr{H})$ (cp. Sect. 3.2), Lemma 7 yields the final result.

Theorem 1 *Let $\mathscr{H}$ be a set-based high-level Petri game with a single recurrently interfering environment player and a bounded number of system players and $\mathbb{G}^H$ the corresponding symbolic two-player game. Then the system players have a strategy in $\mathscr{H}$ if and only if Player 0 has a winning strategy in $\mathbb{G}^H$.*

We construct a winning strategy for the system players in $\mathscr{H}$, i.e., a winning strategy for the system players in the corresponding low-level Petri game $\mathtt{L}(\mathscr{H})$, from the positional winning strategy σ^H for Player 0 in $\mathbb{G}^H$ in two steps. First, we create a positional winning strategy σ^L for Player 0 in $\mathbb{G}^L$ from σ^H as described above. Second, we apply the algorithm presented in [25] to σ^L, i.e., traversing σ^L in breadth-first order while adding the corresponding places and transition of the decision sets, to create a winning strategy for the system players in $\mathtt{L}(\mathscr{H})$. Note that the last step would take infinitely long for infinite Petri game strategies such that a practical algorithm has to provide a finite representation of the strategy.

5 Experimental results

In this section we report on our prototype implementation for generating the symbolic two-player game $\mathbb{G}^H$. We implemented three algorithms for the creation of the reduced state space and compare their runtime to the complete state space creation of ADAM [22]. These results are depicted in Table 1. All algorithms are integrated into the ADAM framework to exploit its data structures and functionality for Petri nets and Petri games.

ADAM uses Binary Decision Diagrams (BDDs) to answer the question of the existence of a strategy and to calculate a strategy in the positive case. In the original algorithm the explicit state space is never generated and thus, the concrete size of $\mathbb{G}^L$ could not directly be obtained. To have a proper comparison of the different sizes of the generated state spaces ($\mathbb{G}^H$ versus $\mathbb{G}^L$) we extended ADAM with a fixed point algorithm which calculates a BDD for the reachable states of the two-player game $\mathbb{G}^L$ and ask for the number of solutions to obtain the number of states of $\mathbb{G}^L$ as reference value. As input, the *Reference-Approach* takes the low-level version $\mathtt{L}(\mathscr{H})$ of a high-level Petri game $\mathscr{H}$. The results and used resources for the calculation of the reduced state space are given in column three and four of Table 1.

For the reduced state space generation we use Symmetric Nets (SNs)[1] [9,10] as underlying structure for the high-level Petri game. SNs are a subclass of high-level Petri nets with the same expressive power but allow for an easy and automatic creation of the system's symmetries from the modeled structure. In SNs the colors are grouped into *basic color classes* and *static subclasses*. The arc expressions, as well as the predicates, are restricted to some basic functions. This makes the modeling of practical examples only slightly more cumbersome.

The following three algorithm are all based on the algorithm originally presented in [35]:

HL-Approach This approach explicitly calculates the reduced state space from the high-level Petri game $\mathscr{H}$.

[1] Symmetric Nets have been formerly known as Well-Formed Nets (WNs). The renaming was part of the ISO standardization [33].

LL-Approach This approach first transforms the high-level Petri game $\mathcal{H}$ into the corresponding low-level Petri game $\mathtt{L}(\mathcal{H})$ and then uses this to explicitly calculate the reduced state space of $\mathbb{G}^H$. During this calculation $\mathcal{H}$ is still exploited to obtain the symmetries of the system.

BDD-Approach This approach uses, as in the Reference-Approach of ADAM, BDDs to symbolically calculate the number of states of the reduced two-player game. For this purpose, the high-level Petri game is also first transformed into the corresponding low-level one and then the high-level structure is used for the automatic generation of the system's symmetries.

Note that we neither calculate a winning strategy nor clarify its existence. We take a high-level Petri game and use three different approaches to calculate the reduced state space, i.e., to calculate the number of states of the high-level two-player game $\mathbb{G}^H$ while exploiting the symmetries of the high-level Petri game as described in Sect. 4.4. Furthermore, we use the adapted algorithm of ADAM to compare these sizes to the size of the previously existing low-level two-player game $\mathbb{G}^L$. Since the running time of any synthesis algorithm crucially depends on the size of the state space it has to explore, this gives a first impression of the potential of our new method.

We applied the algorithms on a set of five scalable benchmark families from applications in robotic control, workflow management, and other distributed domains. For each benchmark the elapsed CPU time (*time* in s) for calculating the size of the state space $|V^L|$ and the size of the reduced state space $|V^H|$ are listed in Table 1 for each approach. A timeout for a calculation time of more than two hours is indicated by TO. For each benchmark the time of the fastest of the new approaches is marked bold. The experiments are calculated on an Intel i7-2700K CPU with 3.50 GHz and 32 GB RAM and refer to the following scenarios:

Package delivery (PD) There are n drones which should deliver m packages. The packages get assigned to the drones. The hostile environment lets an arbitrary drone crash. Drones get informed of the crash and can decide on recovering the package. The system's goal is to deliver all packages (see Sect. 2). *Parameters:* n drones / m packages.

Alarm system (AS) There are n geographically distributed locations. Every location is secured by an alarm system. A burglar, modeled by the environment, can intrude an arbitrary location. The alarm systems can inform each other about burglaries. The goal is that no alarm system is triggered without an intrusion and all alarm systems indicate the correct intrusion point in case of an intrusion. *Parameters:* n alarm systems.

Concurrent machines (CM) There are n machines which should process m orders. The orders can be processed concurrently, but no machine is allowed to process more than one order. The hostile environment chooses one machine to be defective. The goal is that finally all orders are processed. *Parameters:* n machines / m orders.

Document workflow (DW) and (DWs) There are n clerks endorsing or rejecting a document. The document is circularly passed on by the clerks. The environment decides on which clerk receives the document first. The goal is that all clerks take an unanimous decision. In the simple variant DWs the goal is that all clerks endorse the document. *Parameters:* n clerks.

The package delivery benchmark family is newly presented in this paper. The alarm system benchmark family was introduced in [21] and its high-level version was presented in [30]. The benchmark families CM, DW, and DWs were introduced in [22], the high-level version for CM was already presented in [30], the high-level version for DW and DWs were developed for this paper.

Table 1 Experimental results of the benchmark families regarding the sizes of $\mathbb{G}^L$ and $\mathbb{G}^H$ and their calculation time (in s) for the three different approaches for $\mathbb{G}^H$ and the reference approach for $\mathbb{G}^L$

Ben.	*Par.*	Two-player game $\mathbb{G}^L$		Symbolic two-player game $\mathbb{G}^H$			
					HL-Appr.	LL-Appr.	BDD-Appr.
		time	$\lvert V^L \rvert$	$\lvert V^H \rvert$	*time*	*time*	*time*
PD	1/1	0.23	30	30	0.4	**0.21**	0.22
	1/2	0.3	262	138	1.02	0.46	**0.33**
	1/3	0.42	1988	420	3.99	1.7	**1.18**
	1/4	0.72	14010	1017	10.28	**5.32**	457.84
	1/5	1.09	94824	2122	46.08	**11.25**	TO
	1/6	3.82	6.266e5	4004	280	**50.54**	–
	1/7	29.24	4.079e6	6907	2629.75	**530.01**	–
	1/8	202.99	2.629e7	11115	TO	**7367.82**	–
	1/9	1815.35	1.683e8	–	–	TO	–
	...	...	...	...	...	...	...
	4/1	0.7	11473	695	6.86	**4.64**	13.04
	4/2	453.42	1.848e7	3.733e5	6224.85	**2165.59**	TO
	4/3	TO	–	–	TO	TO	–
	5/1	1.45	65713	1177	13.4	**12.02**	TO
	5/2	TO	–	–	TO	TO	–
AS	2	0.45	7445	3780	9.89	4.82	**1.22**
	3	1.36	5.802e7	-	TO	TO	TO
CM	2/1	0.28	157	80	0.39	0.27	**0.23**
	2/2	0.36	2617	685	2.16	1.06	**0.51**
	2/3	0.67	42657	4048	11.72	6.95	**4.64**
	2/4	1.41	6.794e5	18067	61.06	**21.37**	606.3
	2/5	3.6	1.061e7	67675	722.66	**306.49**	TO
	2/6	36.25	1.634e8	2.081e5	TO	**6722.21**	–
	2/7	1061.76	2.488e9	–	–	TO	–
	...	...	...	...	...	...	...
	4/1	0.39	2965	240	1.52	1.42	**0.7**
	4/2	0.81	4.553e5	11215	69.4	**23.61**	30.38
	4/3	2.51	6.973e7	3.824e5	TO	**2716.89**	TO
	4/4	28.91	1.175e10	–	–	TO	–
DW	1	0.27	58	58	0.45	0.3	**0.26**
	...	...	...	...	...	...	...
	6	1.05	7.557e5	1.201e5	582.44	**128.22**	TO
	7	1.41	4.055e6	5.199e5	3832.36	**1097.74**	–
	8	2.38	2.097e7	–	TO	TO	–
DWs	1	0.24	52	52	0.38	0.22	**0.22**
	...	...	...	...	...	...	...
	4	0.69	3.703e5	92647	131.01	**46.17**	TO
	5	1.42	5.638e6	1.125e6	3063.2	**1793.82**	–
	6	1.87	8.293e7	–	TO	TO	–

The results are calculated on an Intel i7-2700K CPU with 3.50 GHz, 32 GB RAM, and a timeout of 2 hours

The figures show a significant decrease on the size of the state space of the system. The new benchmark PD with parameters 1/8 shows the maximal reduction: 26,299,378 states for the standard state space versus 11,115 states for the reduced one. This is a factor of about 2366. The reason for a comparably small reduction for the DW and DWs benchmark family is the circularly passing of the document which restricts the admissible symmetries to rotations.

The decrease of the state space does not come without a cost. The calculation time of the new algorithms for the reduced state space (the last three columns) is in general notably higher than the ones of the reference algorithm for the standard state space (column three). On the one hand, this is due to the equivalence check which is done every time prior to the adding of a new state. On the other hand, this figures are not directly comparable. ADAM uses optimized symbolic algorithms for its calculations which generally outperform explicit algorithms like the ones of the HL- and the LL-Approach on large state spaces. Furthermore, all new algorithms are currently in an early development state.

The main reason of the low performance of the BDD-Approach on larger models is that the current algorithm checks for each newly created state whether there already exists an equivalent one. It is not possible to directly encode this check into a Boolean function for the representation of the two-player game's transition relation. Thus, in this prototype implementation of a symbolic algorithm exploiting the symmetries of the system, the equivalence check is done explicitly. This means, in every round of the fixed point calculation, each explicit state of the BDD representing the successors of this round is calculated. These costly solving steps of the BDDs thwart the use of a symbolic algorithm.

Generally, the LL-Approach outperforms the HL-Approach. This is explicable by the structure of the decision sets. A decision set of the high-level two-player game consists of the concrete instances of the places and transitions of the high-level net. Hence, the HL-Approach calculates these instances over and over each time a high-level transition is requested. An improvement is to buffer these data, but this nearly results in the LL-Approach.

Overall, these figures already show a big step towards a faster practical solving of Petri games because a smaller state space significantly reduces the running time of the synthesis algorithms. Standard algorithms for solving two-player Büchi games with complete information are polynomial in the number of edges of the game and can be applied to the symbolic two-player game $\mathbb{G}^H$. The remaining steps for solving high-level Petri games, i.e., resolving the symmetries of the two-player strategy and creating the Petri game strategy, are linear in the number of edges of the strategy and quadratic in the number of admissible symmetries. Given that the presented algorithms are still in a prototype stadium, these results are very encouraging for further work.

6 Related work

An active research area is Petri net synthesis [2]. Two-player games are studied under the name Petri net supervisory control [2], inspired by the work of Ramadge and Wonham on discrete event systems [45]. A significant body of work on synthesis and control based on Petri nets is in this area (cf. [6,31,46,55]), also for structured Petri nets like modules of signal nets [15]. However, these approaches solve the single-process synthesis problem, as opposed to the multi-process synthesis problem for concurrent systems considered in this paper.

The synthesis of distributed systems (short: distributed synthesis) is much more difficult because one must construct multiple processes that, individually, do not have access to the full system state. Most prominent is the model of Pnueli and Rosner [44], where processes communicating via single-writer single-reader shared variables with synchronous concurrency are considered. After a series of isolated decidability results [37,44], *information forks* [26]

were identified as the necessary and sufficient criterion for undecidability. For architectures without information forks, the synthesis problem can be solved, however, with nonelementary complexity in the number of processes.

Zielonka's *asynchronous automata* [56] have been proposed as an alternative setting for distributed synthesis [27,28,39,42]. The decidability of the control problem of asynchronous automata is open in general. There are various decidability results for restricted cases, e.g., concerning the dependencies of actions [27] or the synchronization behavior [39]. Decidability, albeit again with nonelementary complexity, has also been obtained for acyclic communication structures [28,42].

Petri games based on P/T Petri nets were introduced in [24,25]. They exploit concurrency and causality in defining a notion of informedness for the players. In [25] it is shown that the problem whether the system players have a winning strategy for a safety objective, is undecidable for unbounded Petri games. However, for Petri games with one environment player and a bounded number of system players the problem is EXPTIME-complete. The winning strategy can be obtained in single-exponential time by a reduction to a two-player graph game. In [23] it was shown that also for one system player and a bounded number of environment players the synthesis problem can be solved with the same complexity. In [20] a *bounded synthesis* approach was introduced. It sets a bound for the size of the strategy and constitutes a semi-decision procedure, optimized in finding small implementations. A formal connection between games on asynchronous automata and Petri games is established in [5].

For practical applications, higher-level Petri nets in the form of Coloured Petri Nets (CPN) have been introduced [29,36,47]. In CPNs, individual data values are represented by coloured tokens to describe concurrent systems succinctly. Boolean conditions on these tokens appear as guards of transitions, and expressions define which of these tokens are moved when a transition fires. In general, multisets of coloured tokens may appear as markings. In [36], a translation from CPNs back into normal P/T Petri nets is defined.

There is a significant body of work regarding symmetries. For high-level Petri nets the notion of equivalent markings and the idea of exploiting symmetries was originally introduced in [34,35]. For obtaining the symmetries of the system efficiently, several approaches on different subclasses of high-level Petri nets had been introduced, e.g., in [9,10,16,38,48]. In [8] the idea of using equivalent transitions in addition to the equivalent marking for the creation of the SRG is lifted to CPNs. For low-level Petri nets the reduction ideas are introduced in [51]. From then on lots of work has been done following that direction, e.g., [49,50,54]. Using symmetries for the alleviation of state-explosion problems are also common in model checking [12–14]. The complications that arise when using BDDs for the symmetric state space evaluation in this context is elaborated in [14].

7 Conclusion

We introduced a new, symmetry-exploiting solving algorithm for the subclass of set-based high-level Petri games with a single recurrently interfering source of external information. The main part of the algorithm is a reduction of the high-level Petri game to a two-player game which states consist of enriched equivalence classes of the Petri game's behavior. The key idea of the reduction is borrowed from the reduction of a low-level Petri games with a single external source of information to a two-player game presented in [24]. We proved the correctness of the new reduction by defining a bisimulation between the new game and the game obtained by converting the high-level Petri game to a low-level one and applying the reduction of [24].

Our experimental results show that the new two-player game is significantly smaller than the old one. Utilizing the symmetries of the system enabled us to reduce the state space needed for resolving the synthesis problem in the presented benchmark families by up to three orders of magnitude.

For future work we want to enhance the presented reduction technique to allow for an improved implementation of the solving algorithm. ADAM testifies the well-suited applicability of a symbolic game solving algorithm using BDDs for the synthesis of Petri games. As stated in Sect. 5, a drawback of the current approach is that BDDs cannot directly be used profitably for the calculation of the reduced two-player game. In [10] an algorithm for calculating canonical representatives of the equivalences classes of the reachability graph of a Petri net is presented. A corresponding algorithm for calculating canonical representatives of the equivalence classes of the decision sets could allow for a profitable use of BDDs.

Another step is to investigate several improvements regarding symmetries in high-level Petri nets existing in the literature. For example, the papers [1,3,4,7,32] introduce efficiency improvements for systems with a mixture of symmetric and asymmetric behaviors, or, in [53] the symmetries of entirely symmetric models are deduced from the system itself, i.e., the color classes of a SN can be partitioned automatically. It could be interesting to investigate to what extent the synthesis of high-level Petri games could profit from these results.

Acknowledgements We thank the anonymous reviewers for their insightful comments and detailed suggestions for improvement.

Funding Open Access funding enabled and organized by Projekt DEAL.

Appendix

In this appendix we recall the formal definitions for unfolding, plays, and strategies for low-level Petri games, definitions and properties of the construction of an SRG from [8], and provide technical details and proofs regarding Sect. 4.

A Further formal definitions for Petri games

In this section of the appendix we formally introduce unfoldings, strategies, and plays. Again, we focus only on 1-bounded Petri nets where the preset and the postset of any transition as well as the markings are sets rather than multisets.

Let $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$ be a Petri net. For two nodes $x, y \in \mathcal{P} \cup \mathcal{T}$ we call x a *causal predecessor* of y, written $x < y$, iff $x\,\mathcal{F}^+\,y$ holds. We write $x \leq y$ iff $x < y$ or $x = y$. We call x, y *causally related* iff $x \leq y$ or $y \leq x$ holds. Two nodes $x, y \in \mathcal{P} \cup \mathcal{T}$ are in *conflict*, written $x \sharp y$, iff there is a place $p \in \mathcal{P}\backslash\{x, y\}$ and two transitions $t_1, t_2 \in post(p)$ with $t_1 \neq t_2$, such that $t_1 \leq x$ and $t_2 \leq y$ holds. Two nodes $x, y \in \mathcal{P} \cup \mathcal{T}$ are *concurrent* iff they are neither in conflict nor causally related. A set of places $X \subseteq \mathcal{P}$ is called *concurrent* iff all places are pairwise concurrent.

We introduce occurence nets to represent the occurrences of transitions with their conflicts and causal dependencies. An *occurrence net* is a Petri net $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$ with the following constraints: (i) $\forall p \in \mathcal{P} : |pre(p)| \leq 1$, (ii) $\forall t \in \mathcal{T} : \neg(t \sharp t)$, (iii) $\forall x \in \mathcal{P} \cup \mathcal{T} : \neg(x < x)$, (iv) $\forall x \in \mathcal{P} \cup \mathcal{T} : |\{y \in \mathcal{P} \cup \mathcal{T} \mid y < x\}| < \infty$, and

(v) $In = \{p \in \mathscr{P} \mid pre(p) = \emptyset\}$. This means that each place only has one ingoing transition, no transition is in self-conflict, the flow relation is acyclic, the relation $<$ is well-founded, i.e., does not contain any infinitely decreasing sequence, and the initial marking contains exactly the places which do not have a predecessor. We call an occurrence net a *causal net*, when further (vi) $\forall p \in \mathscr{P} : |post(p)| \leq 1$ holds. In general we equip the function *pre* and *post* with a superscript, e.g., $pre^{\mathscr{N}}$, if we want to stress the dependency on a Petri net $\mathscr{N}$. Let $\mathscr{N}_1 = (\mathscr{P}_1, \mathscr{T}_1, \mathscr{F}_1, In_1)$ and $\mathscr{N}_2 = (\mathscr{P}_2, \mathscr{T}_2, \mathscr{F}_2, In_2)$ be two Petri nets. We call $\mathscr{N}_1$ a *subnet* of $\mathscr{N}_2$ iff $\mathscr{P}_1 \subseteq \mathscr{P}_2$, $\mathscr{T}_1 \subseteq \mathscr{T}_2$, $\mathscr{F}_1 \subseteq \mathscr{F}_2$, and $In_1 = In_2$ holds. A *homomorphism* from $\mathscr{N}_1$ to $\mathscr{N}_2$ is a mapping $h : \mathscr{P}_1 \cup \mathscr{T}_1 \to \mathscr{P}_2 \cup \mathscr{T}_2$ which preserve the types of the nodes and the pre- and postconditions of the transitions, i.e., (i) $h(\mathscr{P}_1) \subseteq \mathscr{P}_2$ and $h(\mathscr{T}_1) \subseteq \mathscr{T}_2$ and (ii) $\forall t \in \mathscr{T}_1 : h(pre^{\mathscr{N}_1}(t)) = pre^{\mathscr{N}_2}(h(t)) \wedge h(post^{\mathscr{N}_1}(t)) = post^{\mathscr{N}_2}(h(t))$, where the application of the homomorphism to a set $X \subseteq \mathscr{P}_1 \cup \mathscr{T}_1$ is defined component-wise: $h(X) = \{h(x) \mid x \in X\}$.

A homomorphism h is called *initial* iff also (iii) $h(In_1) = In_2$. If not differently stated, the elements of a superscripted Petri net $\mathscr{N}^X$ are also implicitly superscripted, i.e., $\mathscr{N}^X = (\mathscr{P}^X, \mathscr{T}^X, \mathscr{F}^X, In^X)$, and we abbreviate the pre- and postset functions by pre^X and $post^X$, respectively. A *branching process* $\beta = (\mathscr{N}^U, \lambda^U)$ of a Petri net $\mathscr{N}$ consists of on occurrence net $\mathscr{N}^U$ and a homomorphism $\lambda^U : \mathscr{P}^U \cup \mathscr{T}^U \to \mathscr{P} \cup \mathscr{T}$ such that $\forall t_1, t_2 \in \mathscr{T}^U : (pre(t_1) = pre(t_2) \wedge \lambda^U(t_1) = \lambda^U(t_2)) \Rightarrow t_1 = t_2$ holds. This means λ^U is injective on transitions with the same preset. If λ^U is initial, the branching process β is called *initial*. A branching process $\beta_R = (\mathscr{N}^R, \rho)$ of $\mathscr{N}$ with a causal net $\mathscr{N}^R$ is called *(concurrent) run* of $\mathscr{N}$. If furthermore ρ is an initial homomorphism, β_R is called an *initial (concurrent) run*. A run formalizes a single concurrent execution of the net. A branching process $\beta_1 = (\mathscr{N}_1, \lambda_1)$ is called a *subprocess* of a branching process $\beta_2 = (\mathscr{N}_2, \lambda_2)$ iff $\mathscr{N}_1$ is a subnet of $\mathscr{N}_2$ and $\lambda_1 = \lambda_2 \mid_{\mathscr{P}_1 \cup \mathscr{T}_1}$, where $h \mid_X$ restricts the domain of the function h to the set X.

An *unfolding* of a net $\mathscr{N}$ is an initial branching process $\beta = (\mathscr{N}^U, \lambda^U)$ of $\mathscr{N}$ satisfying $\forall t \in \mathscr{T}, C \subseteq \mathscr{P}^U : C$ concurrent $\wedge\, \lambda^U(C) = pre^{\mathscr{N}}(t) \Rightarrow \exists t^U \in \mathscr{T}^U : pre^{\mathscr{N}^U}(t^U) = C \wedge \lambda^U(t^U) = t$. This means that whenever a transition of the net can occur in the unfolding there is indeed a transition with the same label occurring in the unfolding. Note that an unfolding is unique up to isomorphism.

A *strategy* for the system players of a Petri game $\mathscr{G} = (\mathscr{P}_S, \mathscr{P}_E, \mathscr{T}, \mathscr{F}, In, \mathscr{B})$ with an underlying Petri net $\mathscr{N}$ is a subprocess $\sigma = (\mathscr{N}^\sigma, \lambda^\sigma)$ of the unfolding $\beta = (\mathscr{N}^U, \lambda^U)$ of $\mathscr{N}$ satisfying the properties: justified refusal, deterministic, and deadlock-avoiding as defined below. We lift the distinction of environment, system, and bad places of $\mathscr{G}$ to the strategy by $\mathscr{P}^\sigma_S = \{p \in \mathscr{P}^\sigma \mid \lambda^\sigma(p) \in \mathscr{P}_S\}$, $\mathscr{P}^\sigma_E = \{p \in \mathscr{P}^\sigma \mid \lambda^\sigma(p) \in \mathscr{P}_E\}$, and $\mathscr{B}^\sigma = \{p \in \mathscr{P}^\sigma \mid \lambda^\sigma(p) \in \mathscr{B}\}$ and analogously define those sets for unfoldings and runs. The conditions of a strategy are defined by

Justified refusal $\forall t \in \mathscr{T}^U : (t \notin \mathscr{T}^\sigma \wedge pre^U(t) \subseteq \mathscr{P}^\sigma) \Rightarrow (\exists p \in pre^U(t) \cap \mathscr{P}^\sigma_S \; \forall t' \in post^U(p) : \lambda^U(t') = \lambda^U(t) \Rightarrow t' \notin \mathscr{T}^\sigma)$, i.e., if an instance t of a transition is forbidden by σ, then the reason is that from a place p in the precondition of t, σ uniformly forbids all instances t' of this transition. This condition also ensures that a strategy does not restrict any pure environment transition.

Deterministic $\forall p \in \mathscr{P}^\sigma_S, M \in \mathscr{R}(\mathscr{N}^\sigma) : p \in M \Rightarrow \exists^{\leq 1} t \in post^\sigma(p) : pre^\sigma(t) \subseteq M$, i.e., there is no situation in the strategy, where a system player allows two separate, enabled transitions.

Deadlock avoiding $\forall M \in \mathscr{R}(\mathscr{N}^\sigma) : (\exists t \in \mathscr{T}^U : pre^U(t) \subseteq M) \Rightarrow \exists t \in \mathscr{T}^\sigma : pre^\sigma(t) \subseteq M$, i.e., whenever the system can proceed in $\mathscr{G}$ there must also be a continuation in the corresponding situation in the strategy.

A *play* π of the Petri game $\mathscr{G}$ is an initial concurrent run $\pi = (\mathscr{N}^R, \rho)$ of the underlying Petri net $\mathscr{N}$. Up to isomorphism π is a subprocess of the unfolding. The *system players win* a play $\pi = (\mathscr{N}^R, \rho)$ iff $\mathscr{B}^R = \emptyset$. Otherwise, the *environment players win*. A play π *conforms* to a strategy σ iff π is a subprocess of σ. A strategy σ is *winning* for the system players iff all plays conforming to σ are won by the system.

B Equivalence via symmetries

In this section of the appendix, the most important results from [8] are recalled to be available for some of the proofs of this paper. Two markings M and M' are *equivalent*, iff $\exists s \in S : M = s(M')$. This leads to equivalence classes $\mathscr{M}(\mathscr{H})/S$. For a marking M let $\overline{M}$ be the arbitrarily chosen, but fixed *representative* of $[M]$ (the equivalence class of M with respect to S), and let s_M be a fixed symmetry such that $s_M(M) = \overline{M}$.

Lemma 8 $\forall t \in \mathscr{T}^H \; \forall v \in Val(t) \; \forall s \in S : M[t.v\rangle M' \Leftrightarrow s(M)[t.s(v)\rangle s(M')$.

Let M be a marking and t a transition. Let $S_M = \{s \in S \mid s(M) = M\}$ be the isotropy group of M, The set of valuation equivalence classes obtained when quotienting $Val(t)$ by the group S_M is denoted by $Val(t)_M = Val(t)/S_M$. For each class in $Val(t)_M$ we arbitrarily chose a representative $\overline{v}$ and define α_M as the function mapping each $v \in Val(t)$ to its representative.

We say a transition t can be *fired symbolically* in the representative $\overline{M}$ for the valuation instance $\alpha_{\overline{M}}(v)$ representing v in $Val(t)_{\overline{M}}$, denoted by $\overline{M}[\![t.\alpha_{\overline{M}}(v)\rangle\!\rangle$, iff $\overline{M}[t.\alpha_{\overline{M}}(v)\rangle$. We obtain another symbolic marking $\overline{M'}$ after the symbolic firing:

$$\overline{M}[\![t.\alpha_{\overline{M}}(v)\rangle\!\rangle\overline{M'} \Leftrightarrow \exists M'' \in [\,\overline{M'}\,] : \overline{M}[t.\alpha_{\overline{M}}(v)\rangle M''.$$

The following properties hold for the symbolic firing of a transition $t \in \mathscr{T}^H$:

Property 3 $M[t.v\rangle M' \Rightarrow \overline{M}[\![t.\overline{v}\rangle\!\rangle\overline{M'}$, where $\overline{v} = \alpha_{\overline{M}}(s_M(v))$.

Property 4 $\overline{M}[\![t.\overline{v}\rangle\!\rangle\overline{M'} \Rightarrow \forall M_1 \in [\,\overline{M}\,] \; \forall v' \in Val(t) \; \exists M_2 \in [\,\overline{M'}\,] : M_1[t.v'\rangle M_2$ with $\overline{v} = \alpha_{\overline{M}}(s_{M_1}(v'))$.

Property 5 $\overline{M}[\![t.\alpha_{\overline{M}}(v)\rangle\!\rangle\overline{M'} \Rightarrow \forall M_2 \in [\,\overline{M'}\,] \; \exists M_1 \in [\,\overline{M}\,] \; \exists v' \in Val(t) : M_1[t.v'\rangle M_2$

Property 6 $\{M \mid M \in \mathscr{R}(\mathscr{H})\} = \{M \mid \overline{M} \in \mathscr{SR}(\mathscr{H})\}$, in the case that the initial marking M_0 is symmetric, i.e., $\forall s \in S : s(M_0) = M_0$. Here $\mathscr{R}(\mathscr{H})$ and $\mathscr{SR}(\mathscr{H})$ are the (symbolic) markings that are (symbolically) reachable from M_0 or $\overline{M_0}$, respectively.

Finally, we elaborate on the assumption that w.l.o.g. high-level Petri nets have a symmetric initial marking. If the initial marking in a high-level Petri net is not symmetric, we can add an initialization transition and a new initial place as follows.

Construction 1 Let $\mathscr{P}_0^H = \{p \in \mathscr{P}^H \mid \exists c \in ty(p) : p.c \in In^H\}$. For all $p \in \mathscr{P}_0^H$ we define $n_p = |\{c \in ty(p) \mid p.c \in In^H\}|$ as the number of color tokens initially residing on p. We name the colors in the initial marking by $In^H = \{p.c_1^p, \ldots, p.c_{n_p}^p \mid p \in \mathscr{P}_0^H\}$. We add a new place p_0 with $ty(p_0) = \{c_0\}$, a fresh singleton color set with a fresh color c_0, and a new transition t_0 such that $(p_0, t) \in \mathscr{F}^H \Leftrightarrow t = t_0$, and $(t_0, p) \in \mathscr{F}^H \Leftrightarrow p \in \mathscr{P}_0^H$, and $\forall t \neq t_0, p \neq p_0 : (p, t_0), (p_0, t) \notin \mathscr{F}^H$. Let further $e(p_0, t_0) = c_0$ and $\forall p \in \mathscr{P}_0^H$:

$e(t_0, p) = \{x_1^p, \ldots, x_{n_p}^p\}$ with variables x_i^p. The guard of t_0 now ensures that, by firing t_0, we generate a marking that is equivalent to In^H (with respect to S):

$$g(t_0) = \bigvee_{s \in S} \bigwedge_{p \in \mathscr{P}_0^H} \bigwedge_{1 \le i \le n_p} x_i^p = s(c_i^p).$$

By construction, we get

$$\forall s \in S\, \forall v \in Val(t_0) : v(t_0) = s(v)(t_0) \wedge [v(t_0) = true \Rightarrow s(post(t_0.v)) = post(t_0.s(v))].$$

Thus, the admissible symmetries S remain unchanged (except for the addition of $id_{\{c_0\}}$ to every symmetry). The new initial marking is $\{p_0.c_0\}$ and therefore trivially symmetric.

C Proofs for the construction of the reduced two-player game

In this section of the appendix, we provide proofs for lemmas of Sect. 4.

Proof (Lemma 1) Let $D[t.v\rangle$ and $s \in S$. We first check that $t.s(v)$ is fireable at $s(D)$: $D[t.v\rangle$ implies $pre(t.v) \subseteq \mathscr{M}(D_{t.v})$. Since s is admissible, we have

$$pre(t.s(v)) = s(pre(t.v)) \subseteq s(\mathscr{M}(D_{t.v})) = \mathscr{M}(s(D_{t.v})) = \mathscr{M}\left(s(D)_{t.s(v)}\right),$$

therefore $s(D)[t.s(v)\rangle$. Since

$$\begin{aligned} s(D') = &\{(p.c, C) \mid (p.c, C) \in s(D) \wedge p.c \notin pre(t.s(v))\} \\ &\cup \{(p.c, \top) \mid p.c \in post(t.s(v)) \cap \mathscr{P}_S^H.ty\} \\ &\cup \{(e.d, post(e.d)) \mid e.d \in post(t.s(v)) \cap \mathscr{P}_E^H.ty\}, \end{aligned}$$

which is exactly the decision set computed after firing $t.s(v)$ in $s(D)$ according to Sect. 4.1, the proof is complete. □

Proof (Lemma 2) We collect the results used to prove that the individual properties are invariant under the application of admissible symmetries. Since all symmetries $s \in S$ are bijective on $\mathscr{P}^H.ty$ (and especially injective) for all $A, B \subseteq \mathscr{P}^H.ty$

$$s(A \cap B) = s(A) \cap s(B) \tag{1}$$

holds. The same is true for $A, B \subseteq \mathscr{T}^H.Val$. That for two a set $A \subseteq \mathscr{P}^H$ or $A \subseteq \mathscr{T}^H$

$$s(A.ty) = A.ty \text{ or } s(A.Val) = A.Val, \text{ respectively} \tag{2}$$

is also true because all $s \in S$ are bijective.
We list, for each individual property, the needed results:
D is environment-dependent: Lemma 1.
D contains a bad place: (1) and (2).
D is a deadlock: Lemmas 1, 8 and (2).
D is terminating: Lemma 8 and (2).
D is nondeterministic: (2), (1), Lemmas 1, and 8. □

Proof (Lemma 3)

1. If D contains a bad place, is a deadlock, is terminating, or nondeterministic, so is $\overline{D}$ by Corollary 1, and there is also only a $\top$-labeled self-loop originating from D as well as from $\overline{D}$. Now consider the cases in which D has none of these properties.
   Case $D \in \mathscr{V}_1^L$: Then $\overline{D} \in \mathscr{V}_1^H$ by Corollary 1. Since $D \in \mathscr{V}_1^L$, $\exists t.v \in \mathscr{T}^H\!Val : \delta = t.v \wedge D[t.v\rangle D'$. Property 1 implies $\overline{D}[\![t.\overline{v}\rangle\!\rangle \overline{D'}$ for $\overline{v} = \alpha_{\overline{D}}(s_D(v))$, and then by definition $(\overline{D}, t.\overline{v}, \overline{D'}) \in \mathscr{E}^H$.
   Case $D \in \mathscr{V}_0^L$ and $D[\top\rangle$: Then $\overline{D} \in \mathscr{V}_0^H$ by Corollary 1, and by definition $\overline{D}[\![\top\rangle\!\rangle$. Since $D[\top\rangle$, we have $\delta = \top$ and $D[\top\rangle D'$. Property 1 implies $\overline{D}[\![\top\rangle\!\rangle \overline{D'}$, and then by definition $(\overline{D}, \top, \overline{D'}) \in \mathscr{E}^H$.
   Case $D \in \mathscr{V}_0^L$ and $\neg D[\top\rangle$: Works the same as the case $D \in \mathscr{V}_1^L$, only that we also use the simple fact that $pre(t.v) \cap \mathscr{P}_E^H.ty = \emptyset \Rightarrow pre(t.\overline{v}) \cap \mathscr{P}_E^H.ty = \emptyset$.
2. Works analogously to the proof of 1., but with the use of Property 2 instead of Property 1.

□

Proof (Lemma 4) We start with $\{\overline{D} \mid D \in \mathscr{R}(\mathscr{A}^L)\} \subseteq \{\overline{D} \mid \overline{D} \in \mathscr{SR}(\mathscr{A}^H)\}$:
$\forall D \in \mathscr{R}(\mathscr{A}^L)\ \exists D_1, \ldots, D_k \in \mathscr{V}_1^L\ \exists \delta_0, \ldots, \delta_k \in \mathscr{T}^H\!Val \cup \{\top\}$ such that

$$(D_i, \delta_i, D_{i+1}) \in \mathscr{E}^L \text{ for all } i = 0, \ldots, k, \text{ with } D_{k+1} := D.$$

We prove this by induction over the length of shortest path from D_0 to a decision set $D \in \mathscr{R}(\mathscr{A}^L)$: If the shortest path is empty we have $D = D_0$, and $\overline{D_0} \in \mathscr{SR}(\mathscr{A}^H)$. The induction step follows by Lemma 3 *1*.

To show $\{\overline{D} \mid \overline{D} \in \mathscr{SR}(\mathscr{A}^H)\} \subseteq \{\overline{D} \mid D \in \mathscr{R}(\mathscr{A}^L)\}$, we proceed analogously, with the induction step following by Lemma 3 *2*. □

Proof (Lemma 5) Note that in this lemma and proof, we explicitly use the notation of equivalence classes instead of their representatives for elements in V^H, so that no confusion arises. Let $B = \{(D, [D]) \mid D \in V^L\}$. This relation is defined on $V^L \times V^H$ by Lemma 4, since $V^L = \mathscr{R}(\mathscr{A}^L)$ and $V^H = \mathscr{SR}(\mathscr{A}^H)$, and we have $(D_0, [D_0]) \in B$.

Let now $(D, [D]) \in B$ and $(D, D') \in E^L$ for a $D \in V^L$. We have to prove that there is a $[D_2]$ such that $([D], [D_2]) \in E^H$ and $(D', [D_2]) \in B$. The obvious candidate is $[D_2] = [D']$, since $(D', [D']) \in B$ by definition. From $(D, D') \in E^L$ follows $\exists \delta \in \mathscr{T}^H\!Val \cup \{\top\} : (D, \delta, D') \in \mathscr{E}^L$. Then, by Lemma 3 *1*, we have $(\overline{D}, \overline{\delta}, \overline{D'}) \in \mathscr{E}^H$. This, again by definition, gives $([D], [D']) \in E^H$.

We can prove $(D, [D]) \in B \wedge ([D], [D']) \in E^H \Rightarrow \exists D_2 \in V^L : (D_2, [D']) \in B \wedge (D, D_2) \in E^L$ similarly, by using Lemma 3 *2*.

Since, by Lemma 4 and Corollary 1, $D \in V_0^L \Leftrightarrow [D] \in V_0^H$ and $D \in F^L \Leftrightarrow [D] \in F^H$, we finally have that B is a bisimulation between $\mathbb{G}^L$ and $\mathbb{G}^H$. □

References

1. Baarir, S., Haddad, S., Ilié, J.: Exploiting partial symmetries in well-formed nets for the reachability and the linear time model checking problems. In: 7th International Workshop on Discrete Event Systems (WODES'04), IFAC Proceedings, Reims, France, 22–24 Sept 2004, vol. 37, no. 18, pp. 219–224 (2004). https://doi.org/10.1016/S1474-6670(17)30749-8
2. Badouel, E., Bernadinello, L., Darondeau, P.: Petri Nets Synthesis. Springer, Berlin (2015)
3. Bellettini, C., Capra, L.: A quotient graph for asymmetric distributed systems. In: 12th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2004), pp. 560–568 (2004). https://doi.org/10.1109/MASCOT.2004.1348313

4. Bellettini, C., Capra, L.: Quotient graphs for the analysis of asymmetric distributed systems: Surveying two alternative approaches. In: 19th International Parallel and Distributed Processing Symposium (IPDPS 2005), CD-ROM / Abstracts Proceedings, 4-8 April 2005, Denver, CO, USA (2005). https://doi.org/10.1109/IPDPS.2005.371
5. Beutner, R., Finkbeiner, B., Hecking-Harbusch, J.: Translating asynchronous games for distributed synthesis. In: Fokkink, W., van Glabbeek, R. (eds.) 30th International Conference on Concurrency Theory (CONCUR 2019). LIPIcs, vol. 140, pp. 26:1–26:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPIcs.CONCUR.2019.26
6. Buy, U., Darabi, H., Lehene, M., Venepally, V.: Supervisory control of time Petri nets using net unfolding. Annu. Int. Comput. Softw. Appl. Conf. **2**, 97–100 (2005). https://doi.org/10.1109/COMPSAC.2005.148
7. Capra, L.: Colored Petri nets state-space reduction via symbolic execution. In: Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005), pp. 231–238 (2005). https://doi.org/10.1109/SYNASC.2005.26
8. Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S.: A symbolic reachability graph for coloured Petri nets. Theor. Comput. Sci. **176**(1), 39–65 (1997). https://doi.org/10.1016/S0304-3975(96)00010-2
9. Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S.: On well-formed coloured nets and their symbolic reachability graph. In: Jensen, K., Rozenberg, G. (eds.) High-level Petri Nets, Theory and Application, pp. 373–396. Springer, Berlin (1991)
10. Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S.: Stochastic well-formed colored nets and symmetric modeling applications. IEEE Trans. Comput. **42**(11), 1343–1360 (1993). https://doi.org/10.1109/12.247838
11. Church, A.: Applications of recursive arithmetic to the problem of circuit synthesis. In: Summaries of the Summer Institute of Symbolic Logic. vol. 1, pp. 3–50. Cornell University, Ithaca, NY (1957)
12. Clarke, E.M., Emerson, E.A., Jha, S., Sistla, A.P.: Symmetry reductions in model checking. In: 10th International Conference on Computer Aided Verification (CAV '98), pp. 147–158 (1998). https://doi.org/10.1007/BFb0028741
13. Clarke, E.M., Filkorn, T., Jha, S.: Exploiting symmetry in temporal logic model checking. In: 5th International Conference on Computer Aided Verification (CAV '93), pp. 450–462 (1993). https://doi.org/10.1007/3-540-56922-7_37
14. Clarke, E.M., Jha, S., Enders, R., Filkorn, T.: Exploiting symmetry in temporal logic model checking. Form. Methods Syst. Des. **9**(1/2), 77–104 (1996). https://doi.org/10.1007/BF00625969
15. Desel, J., Hanisch, H., Juhás, G., Lorenz, R., Neumair, C.: A guide to modelling and control with modules of signal nets. In: Ehrig, H., Damm, W., Desel, J., Große-Rhode, M., Reif, W., Schnieder, E., Westkämper, E. (eds.) Integration of Software Specification Techniques for Applications in Engineering. vol. 3147, pp. 270–300. Springer, Berlin (2004). https://doi.org/10.1007/978-3-540-27863-4_16
16. Dutheillet, C., Haddad, S.: Regular stochastic petri nets. In: Advances in Petri Nets 1990, 10th International Conference on Applications and Theory of Petri Nets, pp. 186–209 (1989). https://doi.org/10.1007/3-540-53863-1_26
17. Engelfriet, J.: Branching processes of Petri nets. Acta Inf. **28**(6), 575–591 (1991). https://doi.org/10.1007/BF01463946
18. Esparza, J.: Model checking using net unfoldings. Sci. Comput. Program. **23**, 151–195 (1994). https://doi.org/10.1016/0167-6423(94)00019-0
19. Esparza, J., Heljanko, K.: Unfoldings—A Partial-Order Approach to Model Checking. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-77426-6
20. Finkbeiner, B.: Bounded synthesis for Petri games. In: Meyer, R., Platzer, A., Wehrheim, H. (eds.) Correct System Design, vol. 9360, pp. 223–237. Springer, Berlin (2015)
21. Finkbeiner, B., Gieseking, M., Hecking-Harbusch, J., Olderog, E.R.: Symbolic vs. bounded synthesis for Petri games. In: Dana Fisman, S.J. (ed.) Proceedings Sixth Workshop on Synthesis, SYNT 2017, pp. 19–39. EPTCS (2017). https://doi.org/10.4204/EPTCS.202
22. Finkbeiner, B., Gieseking, M., Olderog, E.R.: ADAM: Causality-based synthesis of distributed systems. In: Kroening, D., Pasareanu, C.S. (eds.) Computer Aided Verification (CAV). LNCS, vol. 9206, pp. 433–439. Springer, Berlin (2015). https://doi.org/10.1007/978-3-319-21690-4_25
23. Finkbeiner, B., Gölz, P.: Synthesis in distributed environments. In: 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, Kanpur, India, pp. 28:1–28:14, 11–15 Dec 2017. https://doi.org/10.4230/LIPIcs.FSTTCS.2017.28
24. Finkbeiner, B., Olderog, E.R.: Petri games: Synthesis of distributed systems with causal memory. In: Peron, A., Piazza, C. (eds.) Proc. Fifth Intern. Symp. on Games, Automata, Logics and Formal Verification (GandALF). EPTCS, vol. 161, pp. 217–230 (2014). https://doi.org/10.4204/EPTCS.161.19
25. Finkbeiner, B., Olderog, E.R.: Petri games: synthesis of distributed systems with causal memory. Inf. Comput. **253**(2), 181–203 (2017). https://doi.org/10.1016/j.ic.2016.07.006

26. Finkbeiner, B., Schewe, S.: Uniform distributed synthesis. In: IEEE Symposium on Logic in Computer Science, pp. 321–330 (2005). https://doi.org/10.1109/LICS.2005.53
27. Gastin, P., Lerman, B., Zeitoun, M.: Distributed games with causal memory are decidable for series-parallel systems. In: Lodaya, K., Mahajan, M. (eds.) Foundations of Software Technology and Theoretical Computer Science (FSTTCS), LNCS, vol. 3328, pp. 275–286. Springer, Berlin (2005)
28. Genest, B., Gimbert, H., Muscholl, A., Walukiewicz, I.: Asynchronous games over tree architectures. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) Automata, Languages, and Programming (ICALP), LNCS, vol. 7966, pp. 275–286. Springer, Berlin (2013)
29. Genrich, H.J., Lautenbach, K.: System modelling with high-level Petri nets. Theor. Comput. Sci. **13**, 109–136 (1981). https://doi.org/10.1016/0304-3975(81)90113-4
30. Gieseking, M., Olderog, E.R.: High-level representation of benchmark families for Petri nets. CoRR arXiv:1904.05621 (2019)
31. Giua, A.: Petri Nets as Discrete Event Models for Supervisory Control. Ph.D. thesis, Rensselaer Polytechnic Institute (1992)
32. Haddad, S., Ilié, J., Taghelit, M., Zouari, B.: Symbolic reachability graph and partial symmetries. In: 16th International Conference on Application and Theory of Petri Nets 1995, pp. 238–257 (1995). https://doi.org/10.1007/3-540-60029-9_43
33. Hillah, L., Kordon, F., Petrucci-Dauchy, L., Trèves, N.: PN standardisation: A survey. In: Formal Techniques for Networked and Distributed Systems (FORTE 2006), pp. 307–322 (2006). https://doi.org/10.1007/11888116_23
34. Huber, P., Jensen, A.M., Jepsen, L.O., Jensen, K.: Towards reachability trees for high-level Petri nets. In: Advances in Petri Nets 1984, European Workshop on Applications and Theory in Petri Nets, pp. 215–233 (1984). https://doi.org/10.1007/3-540-15204-0_13
35. Huber, P., Jensen, A.M., Jepsen, L.O., Jensen, K.: Reachability trees for high-level Petri nets. Theor. Comput. Sci. **45**(3), 261–292 (1986). https://doi.org/10.1016/0304-3975(86)90046-0
36. Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, vol. 1. Springer, Berlin (1992). https://doi.org/10.1007/978-3-662-06289-0
37. Kupferman, O., Vardi, M.Y.: Synthesizing distributed systems. In: Symposium on logic in computer science (LICS), pp. 389–398. IEEE (2001)
38. Lindquist, M.: Parameterized reachability trees for predicate/transition nets. In: Advances in Petri Nets 1993, Papers from the 12th International Conference on Applications and Theory of Petri Nets, pp. 301–324 (1991). https://doi.org/10.1007/3-540-56689-9_49
39. Madhusudan, P., Thiagarajan, P.S., Yang, S.: The MSO theory of connectedly communicating processes. In: Foundations of Software Technology and Theoretical Computer Science (FSTTCS), pp. 201–212 (2005)
40. Missal, D.: Formal synthesis of safety controller code for distributed controllers. Ph.D. thesis, Martin Luther University of Halle-Wittenberg (2012), http://d-nb.info/1021582786
41. Missal, D., Hanisch, H.M.: A modular synthesis approach for distributed safety controllers, part A: modelling and specification. In: 17th IFAC World Congress, IFAC Proceedings, vol. 41, no. 2, pp. 14473–14478 (2008)
42. Muscholl, A., Walukiewicz, I.: Distributed synthesis for acyclic architectures. In: Proc. FSTTCS. LIPIcs, vol. 29, pp. 639–651. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2014). https://doi.org/10.4230/LIPIcs.FSTTCS.2014.639
43. Nielsen, M., Plotkin, G.D., Winskel, G.: Petri nets, event structures and domains, part I. Theor. Comput. Sci. **13**, 85–108 (1981). https://doi.org/10.1016/0304-3975(81)90112-2
44. Pnueli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In: 31st Annual Symposium on Foundations of Computer Science (FOCS'90), pp. 746–757 (1990). https://doi.org/10.1109/FSCS.1990.89597
45. Ramadge, P., Wonham, W.: Supervisory control of a class of discrete event processes. SIAM J. Control Optim. **25**(3), 206–230 (1987)
46. Raskin, J.F., Samuelides, M., Begin, L.V.: Petri games are monotone but difficult to decide. Technical report, Université Libre De Bruxelles (2003)
47. Reisig, W.: Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-33278-4
48. Schmidt, K.: Parameterized reachability trees for algebraic petri nets. In: Application and Theory of Petri Nets 1995, 16th International Conference, Proceedings, pp. 392–411 (1995). https://doi.org/10.1007/3-540-60029-9_51
49. Schmidt, K.: How to calculate symmetries of Petri nets. Acta Inf. **36**(7), 545–590 (2000). https://doi.org/10.1007/s002360050002

50. Schmidt, K.: Integrating low level symmetries into reachability analysis. In: Tools and Algorithms for Construction and Analysis of Systems, 6th International Conference (TACAS 2000), pp. 315–330 (2000). https://doi.org/10.1007/3-540-46419-0_22
51. Starke, P.H.: Reachability analysis of Petri nets using symmetries. Syst. Anal. Model. Simul. **8**(4–5), 293–303 (1991)
52. Tanenbaum, A.S., van Steen, M.: Distributed Systems—Principles and Paradigms, 2nd edn. Pearson Education, London (2007)
53. Thierry-Mieg, Y., Dutheillet, C., Mounier, I.: Automatic symmetry detection in well-formed nets. In: Applications and Theory of Petri Nets 2003, 24th International Conference (ICATPN 2003), pp. 82–101 (2003). https://doi.org/10.1007/3-540-44919-1_9
54. Wolf, K.: The Petri net twist in explicit model checking. Softw. Syst. Model. **14**(2), 711–717 (2015). https://doi.org/10.1007/s10270-014-0422-4
55. Zhou, Q., Wang, M., Dutta, S.P.: Generation of optimal control policy for flexible manufacturing cells: a Petri net approach. Int. J. Adv. Manuf. Technol. **10**, 59–65 (1995). https://doi.org/10.1007/BF01184279
56. Zielonka, W.: Notes on finite asynchronous automata. Theor. Inform. Appl. (ITA) **21**(2), 99–135 (1987)