Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Master's Thesis

Equivalence of Petri Games

submitted by
Jesko Hecking-Harbusch

submitted on
June 15th, 2016

Supervisor
Prof. Bernd Finkbeiner, Ph.D.

Advisor
Prof. Bernd Finkbeiner, Ph.D.

Reviewers
Prof. Bernd Finkbeiner, Ph.D.
Prof. Dr. Ernst-Rüdiger Olderog

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, _____ _____

(Datum/Date) (Unterschrift/Signature)

Abstract

Petri games represent games between the system and the environment which are distributed over several independent components, respectively. Independent components of the system model that decisions can be based on incomplete information, whereas each independent component of the environment represents one source of non-determinism. Information is gathered locally and only exchanged upon synchronization among components. The goal of Petri games is to synthesize a strategy for the system which satisfies the winning condition for every non-deterministic behavior of the environment.

Petri games are an extension of Petri nets. Bisimilarity defines an equivalence relation on Petri nets among other modeling formalisms. It defines two Petri nets as bisimilar if a bisimulation exists between the nets. The bisimulation includes pairs of markings of two Petri nets. For every pair, all possible transitions in one net can be simulated in the other net such that the following markings are again related and vice versa. This allows us to view bisimilar Petri nets as interchangeable.

In this Master's thesis, bisimilarity of Petri games is defined. This equivalence relation allows the translation of strategies between bisimilar Petri games which makes the games interchangeable. We extend the standard bisimulation between Petri nets to incorporate the extended expressiveness of Petri games. The bisimulation handles the possibly infinite history of Petri games in a finite manner.

Acknowledgments

I am very grateful to Prof. Bernd Finkbeiner for generously offering me this interesting and challenging topic. Especially, I am thankful for the aspiring guidance, invaluably constructive criticism, and friendly advice during the last months. I deeply appreciated the many and illuminating views on a number of issues related to my thesis.
Furthermore, I would like to thank Prof. Bernd Finkbeiner and Prof. Ernst-Rüdiger Olderog for reviewing this thesis.
Moreover, I also place on record, my sincere thank you to my family, friends, and fellow students for their support. Especially, I want to give a warm thanks to Marcel Maltry, Marius Mosbach, and Sebastian Schirmer for proof-reading the thesis. Nevertheless, all remaining errors are mine alone.

Contents

1 Introduction

Petri nets [12, 4] provide a modeling formalism for distributed systems consisting of several independent components. They explicitly model the concurrency between individual components. A component can develop locally as well as synchronously together with other components.

Petri games [7] define games between the system and the environment on an underlying Petri net. The winner of the game is determined based on a safety condition, i.e. the system has to avoid certain game states to win. Petri games are an extension of Petri nets by defining each component to either belong to the system or to the environment.

The system and the environment are divided into local components, respectively. Local components of the system realize incomplete information because each component gathers information locally. Information cannot be exchanged passively between components but only by synchronization. As information, we define the history of choices of components. Each local component of the environment represents one source of non-determinism. A non-deterministic choice creates history the system may be required to obtain to fulfill the safety condition. Petri games allow for infinitely long lasting games and can therefore be used to model reactive systems which interact with the environment.

In a Petri game, we try to synthesize a winning strategy for the system. The *realizability problem* ask whether there exists such a winning strategy. The *synthesis problem* further requires the automatic construction of the strategy if it exists. A winning strategy readily yields an implementation of the modeled reactive system.

Petri games with a single environment component are decidable [7]. A single environment component implies that there is only one source of non-determinism in the game. Therefore, the decision procedure can ensure that the system either has the required information to make decisions fulfilling the safety condition or that it is impossible to obtain these information. The case of two or more sources of non-determinism is significantly more difficult as one environment can proceed while the system ensures that it is informed about the other source of non-determinism.

A decision procedure for the case of more than one environment component in a Petri game represents an open research topic. This thesis takes an alternative approach of defining an equivalence notion on Petri games which allows the reduction of components. We thereby increase the range of solvable Petri games to such games with more than one environment component which are equivalent to a game with a single environment component. Our equivalence notion is based on bisimulation.

Bisimulations [16, 10] consist of pairs of distributions of tokens from Petri nets. Such distributions of tokens are called markings and represent states of the net. The bisimulation requires that each marking in a pair can simulate the labels of all transitions the related marking allows such that the respectively reached markings are again in the bisimulation. *Bisimilarity* based on the existence of a bisimulation including the pair of initial markings of two nets constitutes an equivalence relation on Petri nets [13]. The Petri nets are assumed equivalent because they can simulate each sequence of transitions in the other game. This equivalence relation allows for instance to find the minimal Petri net modeling a certain situation as well as showing that certain markings in a Petri net can be assumed to be repetitive.

In this thesis, we transfer and extend the definition of bisimulation between Petri nets to Petri games. The goal is to allow the translation of strategies between bisimilar Petri games. This allows us to tackle an easy, bisimilar Petri game instead of a harder one because winning strategies can be translated automatically between bisimilar games. Adam [6] represents an automatic solver for Petri games with a single source of non-determinism. A well-founded removal of environment components increases the range of solvable games by Adam.

Strategies are a restriction to so called unfoldings of Petri games. Unfoldings can become infinite as they explicitly represent all possible histories for each component. A loop already produces a new history during each iteration as the number of repetitions of a transition counts as history. Bisimulation between Petri games relates a newly developed strengthening of markings called *history markings* to represent the different histories of the same marking. We prove that bisimilarity based on the existence of a bisimulation between the history markings of Petri games relates these games as equivalent such that their (non-deterministic) winning strategies are translatable.

This thesis is structured as follows. In Section 2, an overview of the general framework of Petri nets, Petri games, and bisimulation between Petri nets is presented. Section 3 defines bisimulation and bisimilarity of Petri games. We develop the final form of bisimulation with the help of several strengthenings targeted at certain properties of Petri games. Applications of bisimilarity of Petri games are outlined in Section 4. In Section 5, it is proven that bisimilarity based on the developed bisimulation is an equivalence relation on Petri games and that it allows the translation of strategies between bisimilar Petri games. Related work is discussed in Section 6 and in Section 7, conclusions are drawn and future work is addressed.

2 Background

In the following, we outline the basics about Petri nets. In turn, we introduce Petri games[1]. Finally, we envision the basic definition of bisimulation between Petri nets. For this general framework we propose a bisimulation between Petri games in Section 3.

2.1 Petri Nets

Petri nets [12, 4] model the flow of independent components in distributed systems. They consist of places, represented by circles, and of transitions, represented by bars. Each place holds at most one token which represents the current state of a component of the distributed system. Transitions enable tokens to flow through the Petri net by defining arrows between the circles and bars. They illustrate the step-wise development of the distributed system. A transition can be used if all preceding places hold one token. When using the transition, all tokens from the preceding places are removed and then tokens are added to all places following the transition. Labels become assigned to transitions to have indistinguishable transitions with different places preceding and following the transitions.

2.1.1 Definition of Petri Nets

A *Petri net* $\mathscr{N}$ is a five-tuple $(\mathscr{P}, \mathscr{T}, \mathscr{F}, In, \mathscr{L})$, where:

- $\mathscr{P}$ is a finite non-empty set of *places*.
- $\mathscr{T}$ is a finite non-empty set of *transitions*.
- $\mathscr{F} \subseteq (\mathscr{P} \times \mathscr{T}) \cup (\mathscr{T} \times \mathscr{P})$ is the *flow relation*.
- $In \subseteq \mathscr{P}$ is the non-empty *initial marking*.
- $\mathscr{L} \colon \mathscr{T} \to L$ is a *labeling function* for transitions.

The set of places and the set of transitions are disjoint ($\mathscr{P} \cap \mathscr{T} = \emptyset$). Places and transitions alternate according to the flow relation. A pair of the form $\mathscr{P} \times \mathscr{T}$ defines that the place precedes the transition, whereas a pair of the form $\mathscr{T} \times \mathscr{P}$ defines that the transition is followed by the place.

[1] The introduction of Petri nets and Petri games is an extended and revised version of the respective subsections from the author's Bachelor's thesis [8]. We base our definition of Petri games on bad markings [5] instead of bad places [7, 6]. Furthermore, we introduce a labeling function for Petri nets and Petri games. Accordingly all examples presented here are entirely new in order to fit the remainder of this Master's thesis.

The initial marking describes in which places tokens reside before any transition is used. The progress of a Petri net can be described by *markings* $M_i \subseteq \mathscr{P}$. Any marking M_i describes a distribution of tokens after i transitions were used starting from the initial marking. This implies that $M_0 = In$ holds. When using markings, we identify a token in a certain place by the name of the place.

L is the set of *labels*. The number of labels might be smaller than the number of transitions ($|L| \leq |\mathscr{T}|$) in order to have several indistinguishable transitions. We use the *identity function* as labeling function if $L = \mathscr{T}$ and $\mathscr{L}(t) = t$ for each $t \in \mathscr{T}$. This function is used when we do not wish to utilize the extended expressiveness of labels. We assume the usage of the identity function if not mentioned otherwise.

2.1.2 Enabledness and Firing of Transitions

Using pairs of the form $\mathscr{P} \times \mathscr{T}$, the flow relation $\mathscr{F}$ defines in which places tokens are necessary in order to use a transition. A transition is called *enabled* if all preceding places hold one token, respectively. The use of transitions is called *firing*. When a transition is fired all preceding tokens are consumed and new tokens are produced based on the pairs of the form $\mathscr{T} \times \mathscr{P}$ from $\mathscr{F}$. A transition with several places preceding it is called *synchronous*, whereas a transition with a single place preceding it is called *local*.

If (p, t) and (t, p) are in $\mathscr{F}$ then a token in place p is required for the transition t and firing the transition does not remove the token from p but leaves it in place. This situation is identified by a double arrow between p and t. We use the notation $M \xrightarrow{t}_{\mathscr{N}} M'$ to represent that firing transition t in Petri net $\mathscr{N}$ leads from marking M to marking M'. We omit $\mathscr{N}$ if it can be uniquely inferred from the context.

2.1.3 Boundedness

A place is *k-bounded* by a variable k iff for all sequences of fired transitions there are at most k tokens in the place. A Petri net is *k-bounded* iff all its places are *k-bounded*. A Petri net which is 1-*bounded* is called *safe*. We restrict ourselves to safe Petri nets to use simpler notation when introducing bisimulation. By our definition of Petri nets, initially each place can hold at most one token and each transition can produce at most one token in a certain place. Thus, a Petri net is safe iff each transition is only enabled if all following places do not contain any tokens, respectively. Our definition of markings is already tailored to safe Petri nets.

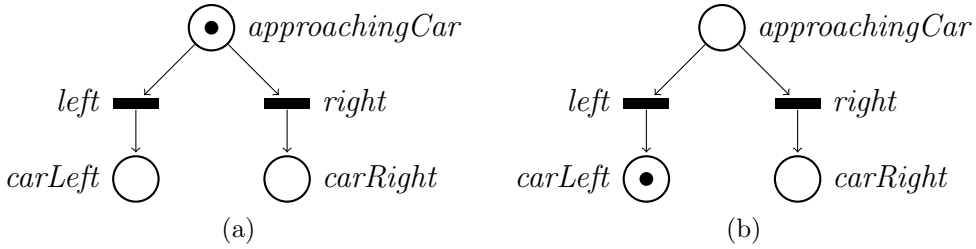

Figure 1: A Petri net is depicted with its initial marking (cf. subpanel (a)) and after firing the transition *left* (cf. subpanel (b)). The Petri net consists of three places interconnected by two transitions. It models the situation where a car approaches a junction and can decide to drive left or right.

2.1.4 Preset and Postset

We introduce the *preset pre* for a transition t of a safe Petri net which is defined by $pre(t) = \{ p \in \mathscr{P} \mid (p,t) \in \mathscr{F} \}$. It returns the set of places that are required to enable the transition. We abbreviate $pre(t)$ by ${}^\bullet t$. Analogously, we define for safe nets the *postset post* for a transition t by $post(t) = \{ p \in \mathscr{P} \mid (t,p) \in \mathscr{F} \}$ and abbreviate $post(t)$ by $t^\bullet$. It defines in which places a token is produced by firing the transition t.

Accordingly, we define which transitions can produce a token in a place p by $pre(p) = \{ t \in \mathscr{T} \mid (t,p) \in \mathscr{F} \}$ and which transitions can consume a token from place p by $post(p) = \{ t \in \mathscr{T} \mid (p,t) \in \mathscr{F} \}$. We use the abbreviations ${}^\bullet p$ and $p^\bullet$, respectively.

2.1.5 Example of a Petri Net

In Fig. 1a and in Fig. 1b, we see two markings of a Petri net. The example models a car approaching a three-way junction. The car is represented by a token in the place *approachingCar*. There are no other tokens in the initial marking (cf. situation depicted in subpanel (a)). The flow relation consists of the following four pairs: (*approachingCar*, *left*), (*approachingCar*, *right*), (*left*, *carLeft*), and (*right*, *carRight*).

Two transitions are enabled, namely *left* and *right*. Subpanel (b) illustrates the case after firing the transition *left*. One token in *approachingCar* has been consumed and one token in *carLeft* has been produced to model that the car decided to go left at the junction. We extend this example later on by assuming another car's behavior to be modeled. By this intuitive example, we can illustrate which complications arise from independent components which are either controllable or uncontrollable.

2.2 Petri Games

Petri games [7] define an extension to Petri nets [4]. They characterize the synthesis problem of distributed controllers where all sources of non-determinism are explicitly modeled. The distributed controllers can only utilize the information they obtained actively.

A Petri net represents the underlying model. Specifically, its places are divided to be either controlled by the *global system player* or by the *global environment player*. The environment constitutes the uncontrollable part of the model, whereas the system is controllable. The *realizability problem* decides whether a winning strategy for the system under a certain winning condition and despite the uncontrollable behavior of the environment exists. Deciding whether a winning strategy can be derived automatically is called the *synthesis problem*.

The winning condition of Petri games can be characterized as a safety condition based on bad markings[2]. These markings have to be avoided by the system in order to win the game. Conversely, the environment tries to reach such a bad marking. A token at a system place is controlled by the global system player and called a *local system player*. Obviously, there can be several local system players but only one global system player.

Petri nets model the information flow in a distributed system. This scheme of information flow is extended in Petri games where tokens (i.e. local players but *not* global players) exchange their complete *history* of fired transitions and visited places upon synchronous transitions. Each token created by a transition contains the history of all tokens consumed by said transition and the information about this most recent firing including all places reached by the firing.

The history is deployed by the local players when making decisions at places. We define that the global system player can only utilize the local information of a token when making a decision at a certain place but not elsewhere (i.e. it cannot use the information other independent tokens in places have although it is the same global player). This prohibits passive exchange of information.

The global environment player is assumed to behave non-deterministically. Nevertheless, an according distinction between global environment player and *local environment players* will be useful.

[2]The usage of bad markings instead of bad places is based on [5]. They allow a coarser equivalence notion while requiring less transitions to be checked when testing the existence of a bisimulation between Petri games.

2.2.1 Definition of Petri Games

A *Petri game* $\mathscr{G}$ is a seven-tuple $(\mathscr{P}_S, \mathscr{P}_E, \mathscr{T}, \mathscr{F}, In, \mathscr{B}, \mathscr{L})$, where:

- $\mathscr{P}_S$ is a finite set of places belonging to the *system*.
- $\mathscr{P}_E$ is a finite set of places belonging to the *environment*.
- $\mathscr{B} \subseteq 2^{\mathscr{P}_S \cup \mathscr{P}_E}$ is the set of *bad markings*.

In this definition, the set of transitions $\mathscr{T}$, the flow relation $\mathscr{F}$, the initial marking In, and the labeling function $\mathscr{L}$ remain the same as in a Petri net with $\mathscr{P} = \mathscr{P}_S \cup \mathscr{P}_E$ which shows that the extension is based on dividing places into two different groups (system and environment) and to determine certain markings as bad. The distribution of places is required to be disjoint ($\mathscr{P}_S \cap \mathscr{P}_E = \emptyset$).

The notation $\mathscr{P}$ in the context of Petri games identifies all places in the game, i.e. the union of system places and environment places. One of the two sets $\mathscr{P}_S$ and $\mathscr{P}_E$ can be empty but not both as the underlying Petri game requires $\mathscr{P}$ to be non-empty. When depicting Petri games, places belonging to the system are filled gray, whereas places belonging to the environment remain white.

We restrict ourselves to Petri games based on safe Petri nets. We adapt the labeling function $\mathscr{L}\colon \mathscr{T} \to L$ of Petri nets with $L \subseteq \mathscr{T}$. It allows two transitions in the Petri game to have the same label. This is necessary when modeling a certain, indistinguishable situation which can happen in different ways. Therefore, bisimulation is defined on these labels.

The progress of Petri games is described—as in Petri nets—by markings $M_i \subseteq \mathscr{P}_S \cup \mathscr{P}_E$. The set of sequences of transitions, where each sequence can be fired sequentially to reach the marking M from the initial marking, is defined by $\{ \langle t_1, \ldots, t_n \rangle \mid In \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} M_n = M \}$ and called the set of *runs* from In to M.

In the following, names providing a superscript like $\mathscr{G}^1$, $\mathscr{G}^2$, etc. are used to address individual Petri games. The superscripts percolate down to the corresponding tuple of the game, e.g. $\mathscr{G}^1 = (\mathscr{P}_S^1, \mathscr{P}_E^1, \mathscr{T}^1, \mathscr{F}^1, In^1, \mathscr{B}^1, \mathscr{L}^1)$.

2.2.2 Unfoldings

As already mentioned, it is essential for the system to find a winning strategy in order to win a Petri game. In this subsection, strategies for Petri games are defined. In particular, we characterize when they are called winning. Strategies are a restriction to the unfolding which explicitly represents the

possibly different histories known to a token in a place. Different histories for a token in a place arise when different transitions lead to the same place.

The *unfolding* $\beta_U = (\mathscr{G}^U, \lambda)$ of an underlying Petri game $\mathscr{G}$ consists of a Petri game $\mathscr{G}^U$ in which all joins of places in the underlying net have been resolved and of a homomorphism λ from $\mathscr{G}^U$ to $\mathscr{G}$. A join of a place occurs when two different transitions have an outgoing arc to the same place, respectively. A join also occurs at a place out of the initial marking if it has an incoming arc. The token can return to its initial position and has fired at least one transition.

Both types of joins are resolved by copying the place (including all transitions and places following it) and changing one transition responsible for the join to lead to the copied place instead of the original one. If a place included in a bad marking is copied an additional bad marking is added where the original place is replaced by its copy.

The removal of all joins results in a replication of places for each possible history they can be reached with. History refers to the places and transitions which the local player has taken along with the places and transitions other local players have taken up to the synchronous transition they participated in. On a synchronous transition, local players exchange their complete history including their knowledge about the history of other local players. Notice that the unfolding can differentiate transitions with the same label if they differ in their preset or their postset. We call $\mathscr{G}^U$ the *unfolded Petri game* of $\mathscr{G}$ and a marking in an unfolding a *cut*.

λ maps the possibly replicated places and transitions of $\mathscr{G}^U$ to their original places and original transitions in $\mathscr{G}$ to show the relationship between the two games. Note that the unfolding enumerates all possible choices of the system and of the environment. Consequently it delineates all possible ways the game can develop.

The initial marking is the same in both Petri games $\mathscr{G}$ and $\mathscr{G}^U$. A place which is not replicated is taken over from $\mathscr{G}$ to $\mathscr{G}^U$. The unfolding unwraps loops in the game because the history incorporates the taken transitions during each iteration of the loop. This implies that the unfolding stores how often the loop is iterated which results in infinite unfoldings.

A *sub-process* $\beta' = (\mathscr{G}^{U\prime}, \lambda^{U\prime})$ of an unfolding $\beta = (\mathscr{G}^U, \lambda^U)$ is produced by removing transitions and the subsequent places and transitions from $\mathscr{G}^U$ as well as closing unwrapped loops resulting in $\mathscr{G}^{U\prime}$. The homomorphism λ' relates the fewer places and transitions of the sub-process to the underlying Petri game of the unfolding. Notice that each sub-process is required by definition to be finite, whereas unfolding can be infinite.

2.2.3 Strategies

A *strategy* is a sub-process of the unfolding because for each system place, the unfolding provides all information available to make a decision. In a strategy, every system place has to decide to either stop movement, to activate all transitions based on the same transition of the original game, or to activate a maximal set of transitions out of which *one* environment player will choose when firing its transitions (i.e. the transitions are all synchronous with the single environment player). The later case defines that the strategy has to determine an order in which it reacts to two or more local environment players. For this case, it is ensured that the environment always chooses exactly one of the transitions the system allowed. The restriction to every system place ensures *deterministic strategies*. The unfolding also models all transitions of the environment. However, as the environment is assumed to behave non-deterministically it is not possible to restrict environment transitions.

Formally, a *(global) strategy* σ for all local system players in a Petri game $\mathscr{G}$ is a finite sub-process $\sigma = (\mathscr{G}^\sigma, \lambda^\sigma)$ of the unfolding $\beta^U = (\mathscr{G}^U, \lambda)$ of the underlying game $\mathscr{G}$ for which the following three conditions must hold:

(S1) if $p \in \mathscr{P}_S^\sigma$ then σ is deterministic at place p

(S2) if $p \in \mathscr{P}_E^\sigma$ then $\forall t \in \mathscr{T}^U. (p \in {}^\bullet t \wedge \forall p' \in {}^\bullet t.\, p' \in \mathscr{P}_E) \implies (p, t) \in \mathscr{F}^\sigma$

(S3) $\forall t \in \mathscr{T}^U. t \notin \mathscr{T}^\sigma \implies \exists p \in {}^\bullet t \cap \mathscr{P}_S^\sigma. \forall t' \in p^\bullet. \lambda(t') = \lambda(t) \Rightarrow t' \notin \mathscr{T}^\sigma$

A strategy σ is called *deterministic* at a place p when for all reachable markings M^σ in the underlying net $\mathscr{N}^\sigma$ of the strategy the following holds: $p \in M^\sigma \implies \exists^{\leq 1} t \in \mathscr{T}^\sigma. p \in {}^\bullet t \subseteq M^\sigma$, i.e. it is at most one transition enabled from the place for all reachable markings. The second condition (S2) ensures that the strategy does not impose restrictions on transitions which require only environment players to fire.

The third requirement (S3) ensures that an unfolded system place forbids all transitions which are based on the same transition in the original game if it forbids at least one of these transitions. This defines that the system does not get additional options from the unfolding of environment places and their transitions. In fact, it bases its decision solely on the history of the unfolded system place. Notice that during the collection of history the system recognizes the difference between equally labeled transitions if their preset and postset differ. The history therefore contains transitions and not labels of transitions. The decision to fire a transition is made before the system knows about the availability of tokens in the other places in the preset of the transition.

As already mentioned, the unfolding is infinite when it unwraps a loop because of the additional history for each iteration through the loop. Nevertheless, the strategy is required to be finite in the number of places by definition. This implies that the strategy can contain loops in order to deal with loops in the underlying Petri game. Notice that a loop can only be unwrapped finitely often resulting in different decisions by the system but there must exist a finite point from which on the system always repeats one certain decision.

The winner of a Petri game $\mathscr{G}$ and its strategy $\sigma = (\mathscr{G}^\sigma, \lambda^\sigma)$ is determined by checking whether there exists a possible sequence of transitions in the strategy which reaches a bad marking. In this case, the environment wins, otherwise the system wins. The global environment player does not benefit from stopping movement if it has not reached a bad marking. Due to this safety condition, the assumption is made that the environment does not stall the game but always picks a transition. This licenses the enumeration of all of its choices in order to check whether a strategy is winning for a Petri game.

The system has to be forced to engage in transitions because otherwise it would win every game, in which the environment cannot reach a bad marking locally, by not moving at all. For the remainder of this thesis, strategies are stipulated to be *deadlock-avoiding*. For all reachable markings M in the strategy, this requires: $\exists\, t_1 \in \mathscr{T}^U.{}^\bullet t_1 \subseteq M \implies \exists\, t_2 \in \mathscr{T}^\sigma.{}^\bullet t_2 \subseteq M$. Notice that it is not necessary that these transitions have the same name. This formulation only states that if there exists an enabled transition in the underlying unfolding of the strategy then there must also exist an enabled transition in the strategy. It can be necessary to unfold the finite strategy to map a reachable marking in the strategy to the corresponding marking in the infinite unfolding.

When no transition is enabled in the strategy the game has *terminated* and the winner of the game is determined depending on whether a bad marking is reached. A deadlock-avoiding strategy fulfilling (S1), (S2), and (S3) which never reaches bad marking is called a *winning strategy*.

2.2.4 Example of a Petri Game

In Fig. 2, an example Petri game is depicted. It is an extension of the example Petri net from Fig. 1. We add a second car to the previous scenario of a three-way junction. With Petri games, we can model whether or not the behavior of the cars is uncontrollable for the system. The first car from the previous example assumingly belongs to the environment. According to the convention introduced above, its places are depicted in white. The places of

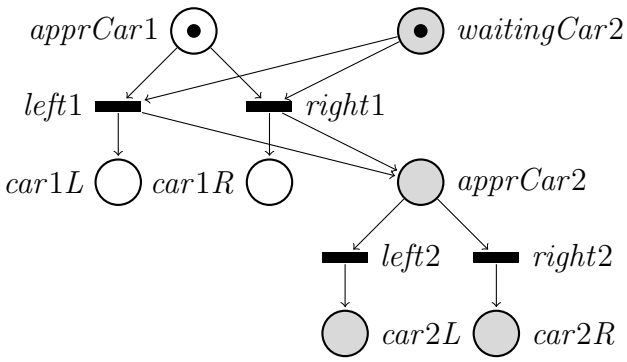

Figure 2: An extended model of the junction from Fig. 1 is displayed. A second car is added which has to react to the decision of the first car at the junction. The intended behavior of the second car is to follow the first car. This is ensured by defining $\{car1L, car2R\}$ and $\{car1R, car2L\}$ as bad markings.

the second car are depicted gray to indicate that we want to assume that they belong to the system. In our scenario, we model that the second car is a police car which is monitoring the junction. It spots which decision the approaching first car makes at the junction via the synchronous transitions *left*1 or *right*1. The police car then decides to follow the first car because the police suspects the car to be driven by a bank robber. The Petri game models how the police car can react at the junction using the information about the first car gathered through synchronous transitions.

The police car can choose between the transitions *left*2 and *right*2 at the junction which are local transitions since the first car has left the junction already. Depending on the decision of the first car either the place $car1L$ or $car1R$ is reached and depending on the second car either $car2L$ or $car2R$ is reached. Since the goal of the police car is to follow the suspected robber's car, we define the markings $\{car1L, car2R\}$ and $\{car1R, car2L\}$ as bad markings. They characterize that the second car went into the opposite direction of the first car.

The depicted Petri game uses the identity function as labeling function and thereby makes it visible which car went *left* or *right* based on the number following. Another possible labeling would be that *left*1 and *left*2 are labeled by *left* and *right*1 and *right*2 by *right* to emphasize that the order of the cars passing the junction is not important (but nevertheless predetermined by the structure this simple game).

Later on, we extend this real life example of the situation where the system tries to mimic the environment. A winning strategy of the system is

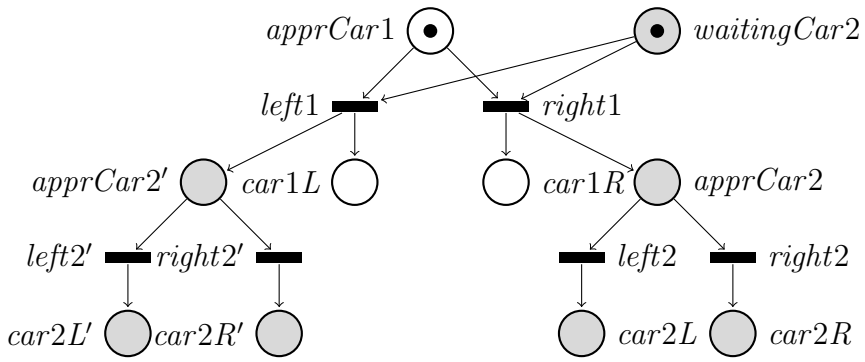

Figure 3: The unfolding of the Petri game from Fig. 2 is depicted. All places and transitions with an apostrophe appended represent a copy of the respective place or transition without it. The unfolding is caused by the two different transitions *left*1 and *right*1 leading to the place *apprCar*2. This place and the sub-game following it are copied exactly once. The unfolding adds $\{car1L, car2R'\}$ and $\{car1R, car2L'\}$ to the already existing bad markings $\{car1L, car2R\}$ and $\{car1R, car2L\}$.

achieved by following the behavior of the environment. In order to find the winning strategy formally, we first need to unfold the game. This is achieved by copying all places with joins of incoming transitions. The only place where this condition holds is *apprCar*2. The unfolding explicitly depicts whether the token in this place saw the robber's car go left or go right. The following sub-Petri game is copied together with the place in order to give the system player the chance to react differently to the two possible histories. The unfolding is depicted in Fig. 3.

The winning strategy corresponding to the unfolding from Fig. 3 is depicted in Fig. 4. The intuitive idea that the system should always mimic the behavior of the environment is stated formally here. At the place *apprCar*2′, the system has witnessed that the environment has fired the transition *left*1. Therefore, it reacts by firing transition *left*2′ (i.e. transition *right*2′ is deactivated). Analogously, at place *apprCar2* the system has seen that the environment has fired transition *right*1; therefore, it fires transition *right*2. The places and transitions which became unreachable because of the deactivation of *right*2′ and *left*2 are also removed.

From the strategy (cf. Fig. 4), it is easy to see that all bad markings are avoided since the only reachable markings where both cars made a decision are $\{car1L, car2L'\}$ and $\{car1R, car2R\}$ representing that both cars went into the same direction. Therefore, the given strategy avoids all bad markings and thus is winning for the system.

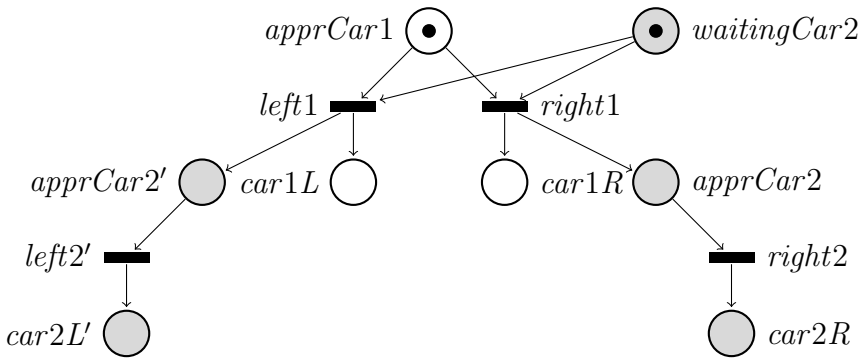

Figure 4: A winning strategy based on the unfolding from Fig. 3 of the Petri game from Fig. 2 is depicted. The system decides for the transitions *left*2′ and *right*2 at the unfolded places *apprCar*2′ and *apprCar*2, respectively.

2.3 Bisimulation between Petri Nets

A bisimulation between Petri nets relates markings of the nets such that transitions with the same label are enabled and their firing reaches markings which are again related. For each included pair of markings, this concept creates a simulation of the labels of transitions in both directions, i.e. for every enabled transition with a label in the first net there exists an enabled transition with the same label in the second net such that another pair of markings in the bisimulation is reached. The converse direction holds as well making it a *bi*simulation.

The existence of such a bisimulation between two nets with their initial markings related implies that every sequence of fired transition in the first net can occur in the second net and vice versa. Furthermore, the option of alternative labels is the same in both nets for every point of decision. This sets up an equivalence notion on Petri nets realized by the bisimilarity relation. Bisimilarity relates two Petri nets as bisimilar iff there exists a bisimulation including the pair of initial markings. Bisimilarity constitutes the equivalence relation on Petri nets we extend in the next section to be applicable for Petri games.

2.3.1 Definition of Bisimulation between Petri Nets

The standard definitions of bisimulation and bisimilarity [10] are given. We alter the original definition slightly to fit our notation for Petri nets.

Definition 2.3.1. (bisimulation between Petri Nets)

Given two Petri nets $\mathcal{N}^1$ and $\mathcal{N}^2$, a binary relation $R \subseteq 2^{\mathcal{P}^1} \times 2^{\mathcal{P}^2}$ is a *bisimulation* if for all $(M_1, M_2) \in R$:

1. $\forall\, t_1 \in \mathscr{T}^1, M_1' \subseteq \mathscr{P}^1.\, M_1 \xrightarrow{t_1}_{\mathscr{N}^1} M_1' \implies \exists\, t_2 \in \mathscr{T}^2, M_2' \subseteq \mathscr{P}^2.$
   $M_2 \xrightarrow{t_2}_{\mathscr{N}^2} M_2' \wedge \mathscr{L}^1(t_1) = \mathscr{L}^2(t_2) \wedge (M_1', M_2') \in R,$

2. $\forall\, t_2 \in \mathscr{T}^2, M_2' \subseteq \mathscr{P}^2.\, M_2 \xrightarrow{t_2}_{\mathscr{N}^2} M_2' \implies \exists\, t_1 \in \mathscr{T}^1, M_1' \subseteq \mathscr{P}^1.$
   $M_1 \xrightarrow{t_1}_{\mathscr{N}^1} M_1' \wedge \mathscr{L}^1(t_1) = \mathscr{L}^2(t_2) \wedge (M_1', M_2') \in R.$

Definition 2.3.2. (Bisimilarity of Petri Nets)

Two Petri nets $\mathscr{N}^1$ and $\mathscr{N}^2$ are *bisimilar* if there exists a bisimulation R relating their initial markings (i.e. $(In^1, In^2) \in R$).

For each marking, the bisimulation relates the markings which follow after firing transitions with the same label ($\mathscr{L}(t_1) = \mathscr{L}(t_2)$). The bisimilarity condition enforces that the initial marking is included in R. This implies that two bisimilar Petri nets have a relation between their reachable markings and two related markings can fire transitions with the same label which again result in related markings.

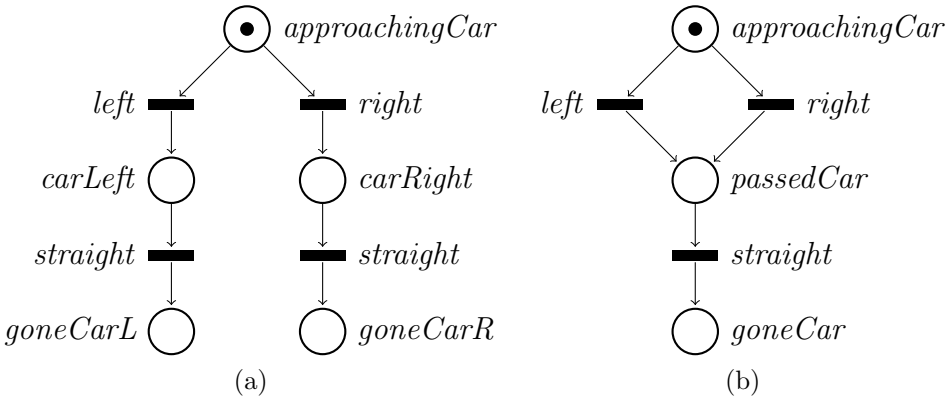

Figure 5: Two bisimilar Petri nets are depicted. Each marking in the bisimulation consists of exactly one place. In the second and third row of places, the place in the right net is in relation with the places in the left net, respectively. This implies that in total the bisimulation between the two nets consists of 5 elements.

2.3.2 Example of Bisimilar Petri Nets

Let us consider Fig. 5 as an example. The Petri net in subpanel (a) is an extension to the previous example Petri net (cf. Fig. 1). After the car has crossed the junction it goes *straight* until it is out of sight. The second Petri net in subpanel (b) is a different model of the same situation of a car

approaching a junction where it can go either left or right and after the junction the car can go only *straight*. The difference in the modeling occurs after the car made its decision in which direction to drive. In the second net, the very same place is reached from which the transition *straight* emits. This is in contrast to the first Petri nets where the transitions *left* and *right* lead to two different places each having an outgoing transition *straight*.

Notice that we use a labeling function different to the identity function in the left Petri net. The transitions emerging from *carLeft* and *carRight* (e.g. named *straightL* and *straightR*) are labeled with *straight*. For the remaining transitions, the labels are identical to the transitions names, respectively.

We claim that the two Petri nets in Fig. 5a and in Fig. 5b are bisimilar. To prove this we first give the relation R between markings from the first and the second Petri net and then argue why the conditions from Definition 2.3.1 are fulfilled. Our proposed relation looks as follows:

$$\begin{aligned} R = \{\ & (\{approachingCar\}, \{approachingCar\}), && (\{carLeft\}, \{passedCar\}), \\ & (\{carRight\}, \{passedCar\}), && (\{goneCarL\}, \{goneCar\}), \\ & (\{goneCarR\}, \{goneCar\})\ \} \end{aligned}$$

The relation R includes as first pair the initial markings of the two Petri nets. The transitions *left* and *right* are enabled, respectively. The transition *left* leads to the pair $(\{carLeft\}, \{passedCar\})$, contained in R, and the transition *right* to the pair $(\{carRight\}, \{passedCar\})$, also contained in R. Therefore, the pair relating the initial markings of the two Petri nets fulfills the bisimulation condition as no other transitions are enabled. For the pair $(\{carLeft\}, \{passedCar\})$ only the transitions *straightL* in the left net and *straight* in the right net are enabled. The transitions have the same label *straight* and lead to the pair $(\{goneCarL\}, \{goneCar\})$, contained in R. For $(\{carRight\}, \{passedCar\})$ the same holds analogously with *straightR* instead of *straightL* in the left net resulting in the pair $(\{goneCarR\}, \{goneCar\})$. For the last two mentioned pairs, no transitions are enabled in either net. Therefore, the bisimulation condition is fulfilled for all elements of R. The existence of R proves that the two Petri nets are in fact bisimilar.

2.3.3 Properties

From this example we can derive one important fact about bisimilar Petri nets: the bisimulation only checks that there exists a bisimilar relation of the markings based on their enabled transitions. However, it does not check whether the places are related. In fact, the names of the places do not matter for the bisimulation but only for the identification of places in the marking.

In the given example from Fig. 5, a token in the place *goneCarL* in the first Petri net clearly incorporates more information than the place *goneCar* in the second. As both places have no transitions, the bisimulation can relate the two places as they represent a marking in the respective net. Thus, *only* the labels of transitions are of importance for the modeling via Petri nets or Petri games in the remainder of this thesis.

The example also shows why we introduced labels for Petri nets and Petri games in the first place. It is possible to have two transitions in the first net which can be related to a single transition in the second net with the same label *straight*, respectively. Without the extension of labels the two nets could not be bisimilar solely because of the different number of transitions.

3 Equivalence

This section constitutes the first major achievement of this thesis. We define bisimilarity of Petri games. It relies on an underlying bisimulation which is a strengthening of the standard bisimulation between Petri nets from Section 2.3.

In Section 3.1, we explain the goal of strategy translation. In turn, we show why the bisimulation between Petri nets cannot be carried over to Petri games. We thereby identify specific properties of Petri games not covered by the standard bisimulation. Conversely, we develop a strengthening of the bisimulation to incorporate these properties. Then, we show that this strengthening does not suffice to deal with another property of Petri games which can be handled by a further strengthening. This approach is repeated until the bisimulation covers all properties of Petri games.

3.1 Goals of the Equivalence

The goal of an equivalence notion on Petri games is to replace equivalent Petri games with each other licensing to translate existing winning strategies between equivalent Petri games. We thereby can replace a hard-to-solve Petri game by an easier-to-solve equivalent one and tackle the easier game instead of the hard one.

A Petri game can be easier-to-solve if there exist fewer system places as there are fewer points where the system has to make a decision. Furthermore, the unfolding becomes smaller for each place less as fewer places can accumulate history. The game can become easier by having fewer transitions the system has to choose from. The same holds analogously for the environment as the number of choices is reduced to which the system has to react.

An even larger decrease in complexity can be expected if the equivalence relation is able to reduce the number of local players, i.e. the number of tokens. For a system token, this implies the removal of an unnecessary distributed component, whereas for an environment token, the number of sources of non-determinism is reduced. The solver for Petri games Adam [6] is only applicable to Petri games with exactly one environment player, i.e. one source of non-determinism. Therefore, the definition of an equivalence relation can lead to an increase of the range of solvable Petri games by Adam if the equivalence can reduce the number of environment tokens.

Unfoldings of Petri games can become infinite. Equivalence on finite structures is easier to define and test than equivalence on infinite structures. We set the goal to define our equivalence on Petri games instead of their unfolding.

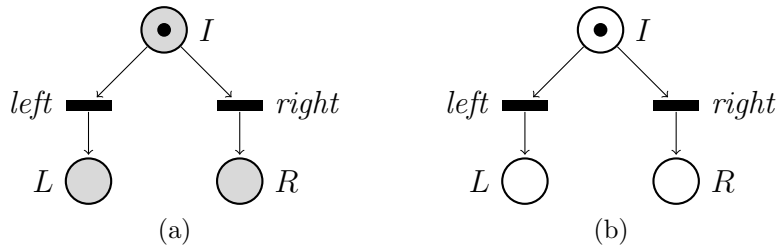

Figure 6: Two example Petri games $\mathscr{G}^a$ (cf. subpanel (a)) and $\mathscr{G}^b$ (cf. subpanel (b)) are displayed. Both games are based on the same underlying Petri net. The sets of bad markings $\mathscr{B}^a$ and $\mathscr{B}^b$ only contain the marking $\{R\}$, respectively.

3.2 Relating System Places

We consider Fig. 6 as a first example. It displays two Petri games $\mathscr{G}^a$ and $\mathscr{G}^b$ which are based on the same underlying Petri net. The net contains the place I in which one token resides. Two transitions *left* and *right* lead from I to the places L and R, respectively. The marking $\{R\}$ is defined as the only bad marking in both games. $\{I\}$, $\{L\}$, and $\{R\}$ are the reachable markings, respectively. Since the underlying Petri nets are identical they are also bisimilar with the bisimulation consisting of the three pairs $(\{I\},\{I\})$, $(\{L\},\{L\})$, and $(\{R\},\{R\})$.

The difference between the two games is the type of the places. All places in Petri game $\mathscr{G}^a$ belong to the system, whereas all places in Petri game $\mathscr{G}^b$ belong to the environment. This implies that in $\mathscr{G}^a$ the system can decide whether to fire transition *left* or transition *right* and thus avoid the bad marking $\{R\}$ by firing *left*. In $\mathscr{G}^b$, it remains uncontrollable for the system which transition is fired, i.e. the system cannot avoid that the bad marking $\{R\}$ is reached. Therefore, $\mathscr{G}^a$ is won by the system, whereas $\mathscr{G}^b$ is won by the environment.

This shows that we have to extend the bisimulation between Petri nets in order to deal with the extended expressiveness of Petri games due to the distinction between system and environment places.

3.3 Relating Bad Markings

We consider again Fig. 6a. If the only bad marking is $\{R\}$ then the game $\mathscr{G}^a$ is won by the system as we have seen before. If we add the bad marking $\{L\}$ (i.e. $\{L\}$ *and* $\{R\}$ are in the set of bad markings) we obtain the Petri game $\mathscr{G}^{LR}$ which is won by the environment. The single system player is forced to

make a decision between *left* and *right* although both choices lead to a bad marking. Firing no transition implies that the corresponding strategy is not deadlock-avoiding and thus not winning.

Since the bisimulation between Petri nets does not incorporate bad markings the game $\mathscr{G}^{LR}$ with $\{L\}$ and $\{R\}$ as bad markings is bisimilar to the game $\mathscr{G}^{a}$. However, the difference in the bad markings provides the only possibility to differentiate the two games as in both games, all places belong to the system. Thus, it does not suffice to relate markings based on having system places in the preset of enabled transitions. In fact, the firing of transitions has to lead to a bad marking either in both games or in none of the two games.

3.4 Deadlock-Avoiding Choices

The example from Fig. 6 illustrates the importance of knowing which places belong to the system and which belong to the environment. System places are controllable for the strategy, whereas environment places remain uncontrollable for the strategy.

A local system player can decide not to fire any transitions when an alternative transition is enabled in the current marking. This alternative transition has other local players in its preset which activate it. The local players in the preset of this alternative transition can be in system places, in environment places, or in a mixture of both. If a local system player decides not to fire any transition despite its transitions being the only ones enabled in the reachable marking of the underlying unfolding then the corresponding strategy is not deadlock-avoiding. Winning strategies are required to be deadlock-avoiding to prevent trivial solutions of not firing any transitions. In most cases, not firing any transitions by the system prevents the reaching of bad markings unless the initial marking is a bad marking or the environment can reach a bad marking only with local transitions.

Bisimulation ensures that two related markings have transitions enabled with the same label to simulate each other. Therefore, a label of a transition either has an alternative or has no alternative in both related markings. The possibility for places in markings to be deadlock-avoiding is already realized by the standard bisimulation between Petri nets. Therefore, the requirement of deadlock-avoiding strategies in Petri games does not require a strengthening of the bisimulation.

3.5 Strengthening towards System Decisions

We formally realize the afore motivated strengthening in the following way: Firstly, the definition of bisimulation is lifted to Petri games. Secondly, the equivalence has to check that every enabled transition from a reachable marking in the one game has an enabled transition in the other game with the same label such that both transitions either have a system place in their respective presets or both do not (motivated in Section 3.2). Thirdly, it has to be ensured that related markings are either both bad markings or both are not (motivated in Section 3.3). Extensions to the definitions of bisimulation and bisimilarity for Petri nets (cf. Section 2.3.1) are printed bold.

Definition 3.5.1. (bisimulation between Petri Games)

Given two **Petri games $\mathscr{G}^1$ and $\mathscr{G}^2$**, a binary relation $R \subseteq \{ (\boldsymbol{M_1}, \boldsymbol{M_2}) \mid \boldsymbol{M_1 \subseteq \mathscr{P}^1 \wedge M_2 \subseteq \mathscr{P}^2 \wedge (M_1 \in \mathscr{B}^1 \iff M_2 \in \mathscr{B}^2)} \}$ is a *bisimulation* if for all $\boldsymbol{(M_1, M_2) \in R}$:

(1) $\forall\, t_1 \in \mathscr{T}^1, M_1' \subseteq \mathscr{P}^1.\, M_1 \xrightarrow{t_1}_{\mathscr{N}^1} M_1' \implies \exists\, t_2 \in \mathscr{T}^2, M_2' \subseteq \mathscr{P}^2.$
$M_2 \xrightarrow{t_2}_{\mathscr{N}^2} M_2' \wedge (\exists\, \boldsymbol{p^1 \in \mathscr{P}^1_S.\, p^1 \in {}^\bullet t_1 \iff \exists\, p^2 \in \mathscr{P}^2_S.\, p^2 \in {}^\bullet t_2}) \wedge$
$\mathscr{L}^1(t_1) = \mathscr{L}^2(t_2) \wedge (M_1', M_2') \in R,$

(2) $\forall\, t_2 \in \mathscr{T}^2, M_2' \subseteq \mathscr{P}^2.\, M_2 \xrightarrow{t_2}_{\mathscr{N}^2} M_2' \implies \exists\, t_1 \in \mathscr{T}^1, M_1' \subseteq \mathscr{P}^1.$
$M_1 \xrightarrow{t_1}_{\mathscr{N}^1} M_1' \wedge (\exists\, \boldsymbol{p^1 \in \mathscr{P}^1_S.\, p^1 \in {}^\bullet t_1 \iff \exists\, p^2 \in \mathscr{P}^2_S.\, p^2 \in {}^\bullet t_2}) \wedge$
$\mathscr{L}^1(t_1) = \mathscr{L}^2(t_2) \wedge (M_1', M_2') \in R.$

Definition 3.5.2. (Bisimilarity of Petri Games)

Two **Petri games $\mathscr{G}^1$ and $\mathscr{G}^2$** are *bisimilar* if there exists a bisimulation R according to Definition 3.5.1 relating their initial markings.

As a consequence, the Petri games $\mathscr{G}^a$ and $\mathscr{G}^b$ from Fig. 6 with bad marking $\{R\}$, respectively, are not bisimilar following Definition 3.5.2. For the transition *left* in $\mathscr{G}^a$ with the system place I in its preset, it is impossible to find a corresponding system place in the preset of *left* in $\mathscr{G}^b$ as there are no system places in $\mathscr{G}^b$.

Accordingly, the Petri game $\mathscr{G}^{LR}$ with bad markings $\{L\}$ and $\{R\}$ and the Petri game $\mathscr{G}^a$ with bad marking $\{R\}$ are not bisimilar. In both games, all places belong to the system but after firing the transition *left*, the marking $\{L\}$ is reached which is only a bad marking in $\mathscr{G}^{LR}$.

3.6 History of the System

We show why the first strengthening made by Definition 3.5.1 does not yet fulfill all requirements for an equivalence notion on Petri games (cf. Section 3.1). In the following, we give a counterexample where the first strengthening relates two Petri games as bisimilar despite them having different winners. After that, we analyze what caused this disagreement and develop a further strengthening referring to the history of the system to fix it.

3.6.1 Example of Different Histories at System Places

We extend our running example by adding a second police car controlled by the system. Both police cars have to follow a car driven by a bank robber (controlled by the environment) at a three-way junction in order to win the game. The two police cars can monitor the junction until the robber crosses it. Depending on their observation, they can decide whether to go left or right. Alternatively, the police cars can decide to go left or right without monitoring the third car controlled by the environment. The corresponding Petri game is depicted in Fig. 7.

The police cars can fire the synchronous version of transitions *testL* or *testR* together and the remaining transitions independently of each other. There are three versions of the transitions *testL* and *testR*, respectively, in order to give the police cars the possibility to independently monitor the junction (two transitions) or to monitor the junction together (one transition). This implies that either both police cars, only one of them, or no police car at all monitors the junction. The later case occurs if *testL* and *testR* are never fired.

Notice that we utilize a labeling function $\mathscr{L}$ in order to have three indistinguishable transitions with different pre- and postsets. For *testL*, the game contains three transitions *testL*1, *testL*2, and *testL*3 which are all labeled by *testL*. The same holds analogously for *testR*. All further transitions of the game are assumed to be labeled with their transition names.

After firing *testL* or *testR* for the first time, the environment reaches the place *tooLateL* or *tooLateR* which indicates that the bank robber has already crossed the junction. We thereby enforce that the two police cars cannot recognize the robber independently of each other. We define every marking where two cars went into different directions to be in the set of bad markings. We deploy the distinction between *tooLateL* and *tooLateR* to store the decision of the environment after one or both police cars monitored the junction. Moreover, bad markings are defined for *car*1*L* and *car*1*R* for the case that no police car monitored the intersection.

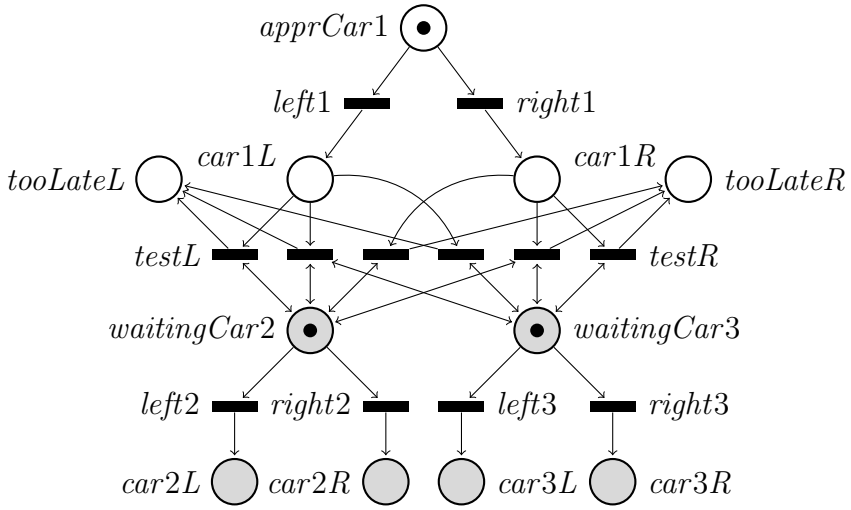

Figure 7: A new version of the running example is depicted where we added a third car controlled by the system. Bad behavior is characterized by those cases when not all three cars make the same decision (i.e. every marking containing a pair of places $car(i)L$ and $car(j)R$ with $i, j \in \{1, 2, 3\}, i \neq j$, a pair *tooLateL* with $car2R$ or $car3R$, respectively, or a pair *tooLateR* with $car2L$ or $car3L$, respectively, is defined as bad marking).

The game is won by the system by initially deactivating the transitions $left(i)$, $right(i)$, $i \in \{2, 3\}$, and allowing the transitions *testLeft* and *testRight* where *both* system players are included in the preset (i.e. the middle ones of the pair of the three transitions labeled by *testL* and *testR* in Fig. 7 are activated and the other two are deactivated, respectively). Both police cars actively monitor the junction which leads to the system re-reaching the places *waitingCar*2 and *waitingCar*3. These places are copied in the unfolding, respectively, i.e. the system gathers information about the choice of the environment. Thus, the system can react by *left*2 and *left*3 if the environment fired *left*1 and with *right*2 and *right*3 if the environment fired *right*1. All three cars drive into the same direction which implies that the system wins by avoiding all bad markings.

In Fig. 8, a closely related Petri game to the one in Fig. 7 is displayed. The difference consists of removing two transitions, namely those labeled by *testLeft* and *testRight* where both system players are in the respective preset. This subtle change removes the possibility for both police cars to monitor the junction together. A lack of parking space in sufficiently close proximity to the junction might cause this.

The change implies for both system players that it is not possible anymore

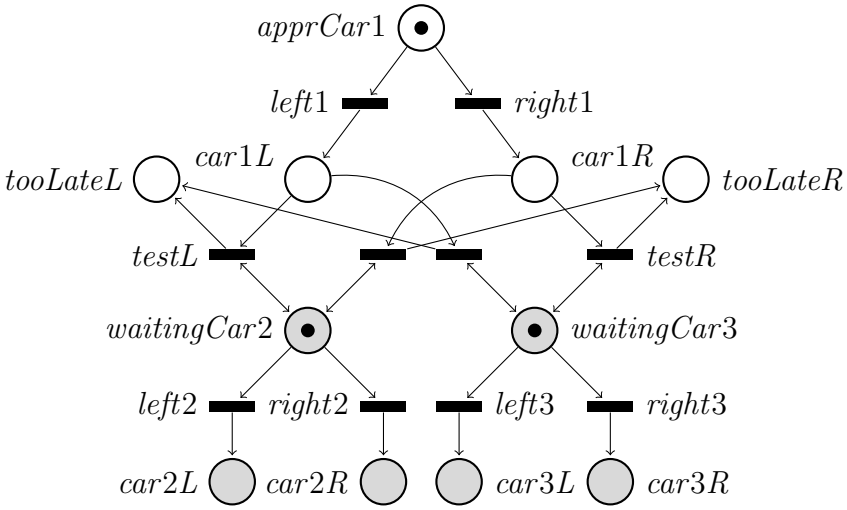

Figure 8: An altered version of the Petri game from Fig. 7 is depicted. We removed the transitions which enabled both system players to obtain information about the single environment player. This change is crucial when determining the winner of the game. The set of bad markings remains unchanged.

to gather information about the behavior of the environment. Without this information, it is impossible for them to make an informed decision as the two police cars cannot exchange information independent of the environment. At least one of the police cars has to choose either *left*(i) or *right*(i), $i \in \{2, 3\}$, in an uninformed manner. Both police officers have to make a decision because otherwise the corresponding strategy is not deadlock-avoiding. The system cannot ensure that both police cars follow the robber implying that the Petri game is won by the environment since the reaching of a bad marking cannot be prevented in general.

3.6.2 Bisimilarity Proof

We claim that the Petri games from Fig. 7 and Fig. 8 are bisimilar according to Definition 3.5.2. The sets of reachable markings in the two games coincide. For each reachable marking M, the bisimulation R includes the pair (M, M). The entire list of all 45 reachable markings for both games is enumerated in Appendix A. There are five positions where the environment token can reside. Transitions license nine ways how the two system tokens can be distributed over their places. Every combination of the position of the environment and of the system is enumerated in Appendix A.

To prove that this relation is indeed a bisimulation it suffices to look at

the markings from which the transitions *testL* and *testR* are enabled since the lack of such transitions with two system places in the preset indicates the only difference between Fig. 7 and Fig. 8. The respective markings are:

$\{car1L, waitingCar2, waitingCar3\}$, $\{car1R, waitingCar2, waitingCar3\}$,
$\{car1L, car2L, waitingCar3\}$, $\{car1R, car2L, waitingCar3\}$,
$\{car1L, car2R, waitingCar3\}$, $\{car1R, car2R, waitingCar3\}$,
$\{car1L, waitingCar2, car3L\}$, $\{car1R, waitingCar2, car3L\}$,
$\{car1L, waitingCar2, car3R\}$, and $\{car1R, waitingCar2, car3R\}$.

For all but the first two markings, one of the two system players has already made a decision between *left* and *right* and thus the additional transitions of the Petri game from Fig. 7 are not enabled. Obviously, these markings have identical behavior in both games. Notice that the transition *testL* and *testR* can only be fired once because the environment reaches the places *tooLateL* and *tooLateR* after the first firing, respectively.

We check the remaining two markings $\{car1L, waitingCar2, waitingCar3\}$ and $\{car1R, waitingCar2, waitingCar3\}$. Actually, it suffices to consider one of the markings since the treatment of the other one works analogously by exchanging L and R in names of places and labels of transitions, respectively. Without loss of generality, let us investigate the first marking $\{car1L, waitingCar2, waitingCar3\}$. The questionable situation occurs if we fire the transition *testL* in Fig. 7 where both system players participate. The two system players return to the same place, whereas the environment player reaches the place *tooLateL*. This implies that this particular situation cannot be distinguished from the firing of *testL* in the second game where only one system player participates. The environment player also reaches *tooLateL*, the participating system player re-reaches $waitingCar(i)$, $i \in \{2, 3\}$, and the non-participating system player remains in $waitingCar(j)$, $j \in \{2, 3\}, i \neq j$. One of those two transitions in Fig. 8 simulates the questionable transition in Fig. 7 as the resulting markings are identical.

Thus, we have shown that the two Petri games are bisimilar based on the bisimulation from Definition 3.5.1 despite the fact that the Petri game from Fig. 7 is won by the system whereas the Petri game from Fig. 8 is won by the environment. The crucial point is that bisimulation does not recognize differences in the history which a token in a place can utilize in order to make a decision. To overcome this problem we propose a space-efficient way for storing which tokens in two related markings of a bisimulation have witnessed the firing of transitions with the same labels, respectively.

3.6.3 History Markings

We define *history markings* by extending markings to group places together such that tokens (i.e. local players) in the places have fired the same transitions and visited the same places. We identify a token in a place by the name of the place in the same way as we do for markings.

Definition 3.6.1. (History Markings)

The function $\mathcal{H}$ returns the set of vectors denoting all possible history markings for a marking M of a Petri game $\mathscr{G}$ and is defined as follows: $\mathcal{H}(M) = \{ \langle s_1, \ldots, s_n \rangle \mid \forall\, 1 \leq i \leq n.\, s_i \subseteq M \wedge s_1 \cup \ldots \cup s_n = M \}$, where n denotes the number of different histories in each history marking and $\cup$ represents the disjoint union enforcing that each place of the marking M occurs exactly once in every corresponding history marking.

Each history marking is defined as a vector containing sets of places. We need the order of the vector in Section 3.7 when relating history markings in pairs of our bisimulation. The definition of history markings ensures that every place of the underlying marking occurs in exactly one set of the vector. A history marking thereby realizes a refinement of the underlying marking by distributing places among sets. Each set of the vector describes a unique history which identifies the history of tokens in all places in the set. The underlying marking of a history marking can be restored by union over the elements of the vector.

We introduce the following notation: History markings are represented by H as markings are represented by M. Each history marking H is based on a marking M. For a given history marking H, H^i, H_i, H' etc., we define that the underlying marking is given by the same suffix (e.g. M^5 is the underlying marking of the history marking H^5).

We say a transition t is enabled from a history marking H leading to another history marking H' if t is enabled from the underlying marking M of H and the firing results in the marking M' which is the underlying marking of a history marking H'. We introduce the notation $H \xrightarrow{t} H'$.

Let us consider Fig. 7 to exemplify the usage of history markings. Assume the marking $M = \{\mathit{tooLateL}, \mathit{waitingCar}2, \mathit{waitingCar}3\}$ is reached by firing $\mathit{left}1$ followed by a transition labeled by *testL*. From the marking, it remains unclear which of the three transitions labeled by *testL* was actually fired and how informed the two system players are. From the structure of the game, we can only derive that at least one system player is informed about the decision of the environment.

The following five history markings of length $n \leq 2$ are allowed by the structure of the game for the marking M:

$$\langle \{tooLateL, waitingCar2, waitingCar3\} \rangle, \quad (1)$$
$$\langle \{tooLateL, waitingCar2\}, \{waitingCar3\} \rangle, \quad (2)$$
$$\langle \{waitingCar3\}, \{tooLateL, waitingCar2\} \rangle, \quad (3)$$
$$\langle \{tooLateL, waitingCar3\}, \{waitingCar2\} \rangle, \quad \text{and} \quad (4)$$
$$\langle \{waitingCar2\}, \{tooLateL, waitingCar3\} \rangle. \quad (5)$$

The first history marking (1) describes that all three tokens are equally informed. This can only be achieved when the transition *testL* was fired where both system players participated. The next two history markings (2) and (3) describe that only *tooLateL* and *waitingCar*2 are equally informed, whereas *waitingCar*3 is informed differently. Only the first police officer participated in the synchronous transition *testL*. The last two history markings (4) and (5) work analogously assuming the second police officer becomes informed while the first officer remains uninformed. Based on each history marking, it is clear which transition was fired, i.e. the preset and the postset of the transition labeled by *testL* can be derived.

The difference between the games in Fig. 7 and in Fig. 8 becomes obvious by spelling out the history markings of length $n \leq 2$ based on the marking M for the second game. (1) is only a valid history marking for the first game but not for the second one. (2) to (5) represent all history markings of length $n \leq 2$ for the marking M in Fig. 8. An increased length only licenses the adding of empty sets for the history markings of both games.

3.7 Strengthening towards System History

We further strengthen the bisimulation between Petri games by forcing it to relate history markings (instead of markings). When checking for the enabledness of transitions the bisimulation requires that the system places in the preset of the transition come from sets from the same position in the related history markings. The bisimulation ensures inductively that being at the same position implies that tokens in the places from two sets of the same position incorporate *bisimilar history*, i.e. their knowledge of labels of fired transitions can be viewed as equivalent. Bisimilar history is introduced formally in Section 5.3 because the intuitive explanation suffices until then. This enables us to check whether the system places which are in the precondition of a transition are informed enough to simulate each others decision. Extensions to the previous strengthening (cf. Section 3.5) are printed bold. The formal specification looks as follows:

Definition 3.7.1. (bisimulation between Petri Games)

Given two Petri games $\mathscr{G}^1$ and $\mathscr{G}^2$, a binary relation $R \subseteq \{ (\boldsymbol{H_1}, \boldsymbol{H_2}) \mid M_1 \subseteq \mathscr{P}^1 \wedge M_2 \subseteq \mathscr{P}^2 \wedge (M_1 \in \mathscr{B}^1 \iff M_2 \in \mathscr{B}^2) \wedge \boldsymbol{H_1} \in \boldsymbol{\mathcal{H}(M_1)} \wedge \boldsymbol{H_2} \in \boldsymbol{\mathcal{H}(M_2)} \}$ is a *bisimulation* if for all $(H_1, H_2) \in R$:

(1) $\forall\, t_1 \in \mathscr{T}^1, H_1' \in \mathcal{H}(M_1').\, M_1 \xrightarrow{t_1}_{\mathcal{N}^1} M_1' \implies \exists\, t_2 \in \mathscr{T}^2, H_2' \in \mathcal{H}(M_2').$
$M_2 \xrightarrow{t_2}_{\mathcal{N}^2} M_2' \wedge \mathscr{L}^1(t_1) = \mathscr{L}^2(t_2) \wedge \boldsymbol{repeat\text{-}constraint} \wedge (H_1', H_2') \in R,$

(2) $\forall\, t_2 \in \mathscr{T}^2, H_2' \in \mathcal{H}(M_2').\, M_2 \xrightarrow{t_2}_{\mathcal{N}^2} M_2' \implies \exists\, t_1 \in \mathscr{T}^1, H_1' \in \mathcal{H}(M_1').$
$M_1 \xrightarrow{t_1}_{\mathcal{N}^1} M_1' \wedge \mathscr{L}^1(t_1) = \mathscr{L}^2(t_2) \wedge \boldsymbol{repeat\text{-}constraint} \wedge (H_1', H_2') \in R.$

where ***repeat-constraint*** is respectively defined as the conjunction of:

(a) $\forall\, \boldsymbol{s_i^1} \in \boldsymbol{H_1}. \forall\, \boldsymbol{s_i^2} \in \boldsymbol{H_2}. (\exists\, \boldsymbol{p^1} \in \boldsymbol{\mathscr{P}_S^1}. \boldsymbol{p^1} \in \boldsymbol{pre^1(t_1)} \wedge \boldsymbol{p^1} \in \boldsymbol{s_i^1} \iff \exists\, \boldsymbol{p^2} \in \boldsymbol{\mathscr{P}_S^2}. \boldsymbol{p^2} \in \boldsymbol{pre^2(t_2)} \wedge \boldsymbol{p^2} \in \boldsymbol{s_i^2})$,

(b) $\boldsymbol{H_1' = \langle\, post^1(t_1), s_1^1 \backslash pre^1(t_1), s_2^1 \backslash pre^1(t_1), \ldots, s_n^1 \backslash pre^1(t_1)\, \rangle}$,

(c) $\boldsymbol{H_2' = \langle\, post^2(t_2), s_1^2 \backslash pre^2(t_2), s_2^2 \backslash pre^2(t_2), \ldots, s_n^2 \backslash pre^2(t_2)\, \rangle}$.

The usage of the history markings is defined precisely in *repeat-constraint*: (a) defines how it is ensured that system places in the preset of enabled transitions with the same label have bisimilar history in both games. For such a pair of transitions t_1 and t_2, an iteration over the positions i of the history markings takes place. Note that we control that related history markings have the same length. For each pair of sets s_i^1 and s_i^2 from the history markings, it is checked that either both sets contain a system place which is in the preset of the respective transition (t_1 for s_i^1 and t_2 for s_i^2) or both do not entail such a place.

The definition of *repeat-constraint* utilizes the ordering of the vectors of two related history markings in the following way: Sets from the same position of two related vectors incorporate bisimilar history meaning that they have witnessed fired transitions with the same labels in their respective Petri game. We prove in Section 5.4 that this notion suffices to compare Petri games. The bisimulation checks the structure of the related games in such a way that the exact places visited in the respective game do not matter.

(b) and (c) of *repeat-constraint* define how the history markings are changed when firing a transition. These two steps build up the successor pairs of history markings which is required to be in the bisimulation. These steps are crucial for (a) to constitute a meaningful check. It is proven in Section 5.3.1 that history markings inductively realize bisimilar history when following these construction rules.

Let t be t_1 for (b) and t_2 for (c) as they work analogously. The idea is to put the most recently extended history to the beginning of the history marking. This is done by defining the set at the first position equal to the postset of the fired transition t in the respective game. If t is a synchronous transition then all participating places have exchanged their entire history and thus incorporate bisimilar history (including the firing of t and the visiting of the places in $t^\bullet$). If t is a local transition then only one place occurs in the postset and the token in this place incorporates a unique history because it is the only token knowing about the most recent firing of t.

Based on the preset of t, we can define the remaining part of the new history marking. The preset of t is removed from each element of the old history marking and then the old history marking is appended to the postset of t. All places that do not occur in the preset remain in their respective set of the vector. These sets are moved one position to the right because the postset of t is added to the front. The relation between the positions of the two history markings remains intact since the movement follows the same pattern in both history markings.

The current description leads to an increase in the vector size whenever a transition is fired. For convenience, we define that an empty set at the same position in *both* history markings can be removed including the movement of the following sets one position to the left. It is an arbitrary choice to add the postset to the front as any other position would work as well, as long as the choice of the position is consistent for all elements of the bisimulation. For the remainder of this thesis, we only add to the front of history markings.

Notice that the positions of the history markings do not impose an ordering on how well informed certain sets of places are, e.g. we cannot infer information whether the first set is better or worse informed than the second set of a history marking. In fact, we can only derive that the two sets of places are differently informed and that sets in the same position of two related history markings are informed bisimilar.

At last, we have to extend the definition of bisimilarity to include the initial pair of two history markings. This is simple because initially *all* places incorporate an empty history of no fired transitions.

Definition 3.7.2. (Bisimilarity of Petri Games)

Two Petri games $\mathscr{G}^1$ and $\mathscr{G}^2$ are *bisimilar* if there exists a bisimulation R according to Definition 3.7.1 with $(\langle \boldsymbol{In^1} \rangle, \langle \boldsymbol{In^2} \rangle) \in R$.

According to the new definition, the Petri games from Fig. 7 and from Fig. 8 are not bisimilar anymore. After firing the transitions *left*1 and then *testL* with both system players participating in the first game, the history

marking $\langle \{waitingCar2, waitingCar3, tooLateL\}, \{\} \rangle$ will be reached. The same firing sequence in the second game has to include a version of *testL* where only one system player participates. This implies that only the history markings $\langle \{waitingCar2, tooLateL\}, \{waitingCar3\} \rangle$ (the first system player participated in *testL*) and $\langle \{waitingCar3, tooLateL\}, \{waitingCar2\} \rangle$ (the second system player participated in *testL*) are reachable.

For $\langle \{waitingCar2, tooLateL\}, \{waitingCar3\} \rangle$, the distinction is caused by the enabled transitions *left*3 and *right*3 from *waitingCar*3 because the only system player in the preset of the respective transition in the other game is not at the same position in the only candidate history marking $\langle \{waitingCar2, waitingCar3, tooLateL\}, \{\} \rangle$. Analogously, the same holds for the second history marking of the second game and the transitions *left*2 and *right*2 from *waitingCar*2. The uninformed police car is recognized in both cases.

3.8 History of the Environment

We show that the previously deployed strengthenings of the bisimulation (cf. Definition 3.7.1) are still not sufficiently restricted to state equivalence on Petri games. We outline an example of two Petri games which are bisimilar according to Definition 3.7.2 but are won by different players. The Petri games can be found in Fig. 9 and Fig. 10.

3.8.1 Example of Environment Places Hiding History

The Petri game from Fig. 9 consists of two players. One is controlled by the environment and can be identified by its places and local transitions having names including "1". The second player is controlled by the system. Its names of places and local transitions have names including "2". The environment can decide between *left*1 or *right*1 reaching the places $car1L$ or $car1R$, respectively. In spirit of our running example, the police (system) still tries to follow the bank robber (environment) at a three-way junction. The system player can fire transitions labeled by *testL* and *testR*. We utilize the labeling function $\mathscr{L}$ to describe a local (to the system) and a synchronous version, respectively. The local version models that the police monitors the junction lazily and thus overlooks the robber's car, whereas the synchronous version models a thoroughly monitoring resulting in recognizing the robber. After firing *testL* or *testR*, the system reaches the place $waitingCar2'$ where it can make a decision between *left*2 and *right*2. It is defined as bad behavior if the system and environment go into opposite directions (i.e. the markings $\{car1L, car2R\}$ and $\{car1R, car2L\}$ are bad markings).

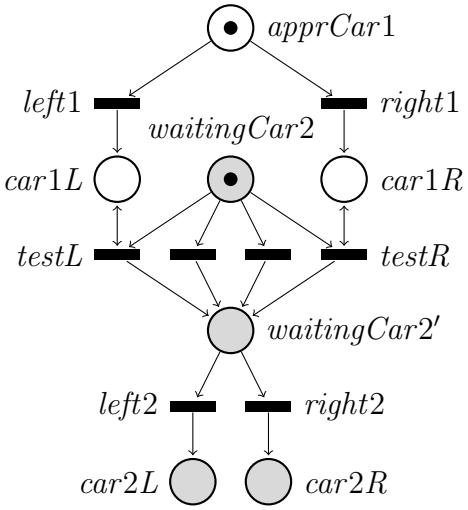

Figure 9: A Petri game is depicted which models the behavior of the police car controlled by the system in response to the robber's car controlled by the environment. The police car is forced to perform a transition *testL* or *testR* before making its decision between *left*2 or *right*2. Depending on how thoroughly it wants to monitor the junction, these transitions can be local or synchronous with the places $car1L$ and $car1R$ representing the environment's decision between *left*1 and *right*1, respectively. It is defined to be bad behavior when the two cars go into different directions by having $\{car1L, car2R\}$ and $\{car1R, car2L\}$ as bad markings.

The Petri game from Fig. 10 lacks the synchronous versions of *testL* and *testR*. Therefore, the system place $waitingCar2'$ has only two incoming transitions. Another consequence of the lack of the synchronous transitions is that the two players are disconnected in the game and cannot exchange information. It is impossible for the system to make an informed decision and to thereby win the game.

3.8.2 Bisimilarity Proof

We prove that the Petri games from Fig. 9 and Fig. 10 are bisimilar according to Definition 3.7.2 to motivate the next strengthening of the bisimulation towards including the environment of the history as well. The relation R between the two Petri games can be found in Appendix B. It contains 24 pairs of history markings from the two games.

The set of bad markings $\{\{car1L, car2R\}, \{car1R, car2L\}\}$ in both games coincide implying that we can neglect the check whether the underlying markings of history markings are bad markings.

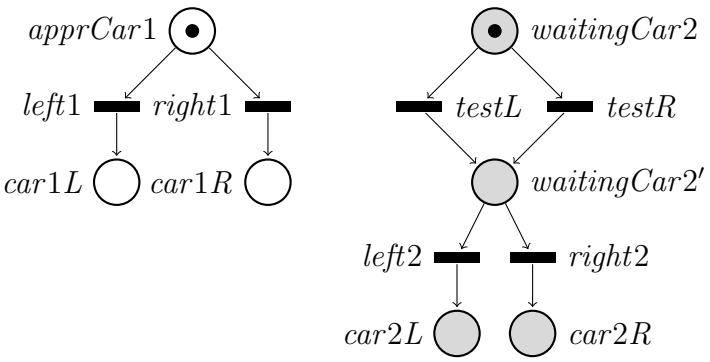

Figure 10: An altered version of the Petri game from Fig. 9 is displayed. The synchronous version of the transitions *testL* and *testR* has been removed. The bad markings $\{car1L, car2R\}$ and $\{car1R, car2L\}$ remain the same.

In the following, we focus on those pairs which represent how the synchronous versions of transitions *testL* and *testR* in the first game can be simulated by the local transitions in the second game. The synchronous transitions are enabled if the environment is the first player to make a decision. $(\langle \{car1L\}, \{waitingCar2\} \rangle, \langle \{car1L\}, \{waitingCar2\} \rangle)$ (cf. pair (2)) and $(\langle \{car1R\}, \{waitingCar2\} \rangle, \langle \{car1R\}, \{waitingCar2\} \rangle)$ (cf. pair (3)) are the corresponding history markings.

The firing of the synchronous transition *testL* from (2) results in the pair $\langle \{waitingCar2', car1L\}, \{\} \rangle, \langle \{waitingCar2'\}, \{car1L\} \rangle)$ (cf. pair (5)) and the firing of the synchronous transition *testR* from (3) leads to the pair $\langle \{waitingCar2', car1R\}, \{\} \rangle, \langle \{waitingCar2'\}, \{car1R\} \rangle)$ (cf. pair (7)). In these pairs, the reached system place ($car1L$ or $car1R$) and the reached environment place ($waitingCar2'$, respectively) of the left history marking are in the same set and thus both participated in the transition. This implies that the history marking represents that the system learned the environment's decision. In the right history marking, only the system place is in the first set meaning that the system did not learn the environment's decision. There is an empty set at the second position of the left element of the pairs to represent that there is no place in the first game not having learned the fired transition.

From the pairs (5) and (7) onward, only the transitions *left*2 and *right*2 representing the system's decision are enabled. These transitions make it impossible to distinguish between the history in the two games because only the respective system place participates in these transitions. (5) leads to $(\langle \{car2L\}, \{car1L\}, \{\} \rangle, \langle \{car2L\}, \{\}, \{car1L\} \rangle)$ (cf. pair (14) in Appendix B) and $(\langle \{car2R\}, \{car1L\}, \{\} \rangle, \langle \{car2R\}, \{\}, \{car1L\} \rangle)$ (cf. pairs (16)), whereas

(7) leads to $(\langle\{car2L\},\{car1R\},\{\}\rangle,\langle\{car2L\},\{\},\{car1R\}\rangle)$ (cf. pair (18)) and $(\langle\{car2R\},\{car1R\},\{\}\rangle,\langle\{car2R\},\{\},\{car1R\}\rangle)$ (cf. pair (20)). It is assumed that the system places have bisimilar history since they participated in the most recent transition. In fact, the difference of the environment place participating in the first game and thus making one transition synchronous including the exchange of history is not recognized.

After the respective firing of *left*2 and *right*2, the position of the environment in the related history markings of pairs (14), (16), (18), and (20) differs showing that the environment exchanged history in the first game but did not do so in the second game. No more transitions are enabled to recognize this difference.

All other pairs in R are caused by the remaining game being isomorphic. The system and the environment places have the same labels and the same behavior as long as the synchronous versions of transitions *testL* and *testR* are not fired. The remaining pairs of Appendix B cover the isomorphic parts of the two games. R is a bisimulation because the games are mostly isomorphic and the non-isomorphic part is covered by simulating the synchronous transitions with the respective local ones.

The previous proof shows a limitation to our current definition of bisimulation. The bisimulation does not recognize that the environment player carries information which the system learns via the synchronous versions of *testL* and *testR*. In contrast to the second game, the environment learns that the transition *testL* or *testR* is fired in the first game. As this is the last transition of the environment this difference cannot be used in order to differentiate the two Petri games. We tackle this problem by requiring that the environment places preceding a transition are in sets of history markings at the same position implying that they have bisimilar information. It then is no longer possible to simulate synchronous transitions between the system and the environment by local transitions.

3.9 Strengthening towards Environment History

The afore motivated strengthening is made precise in Definition 3.9.1. A fourth *repeat-constraint* is added (for symmetry reasons it is inserted at position (b)). This constraint states that the check regarding system places also has to be performed for environment places. If there exists an environment place in the preset of a transition which is enabled in one game (accordingly, the place is in a set of a history marking at a certain position) then there is a transition with identical label in the other game such that the preset of said transition includes an environment place at the same position in the related history marking.

We also add a fifth *repeat-constraint* at position (e) which defines formally how empty sets are removed from history markings. If empty sets occur at the same position in two related history markings the following sets to the right are shifted one position to the left. Thereby, the last set gets removed in both history markings and both empty sets get replaced by the respective shift. Although this procedure is not a strengthening, it is required to state formally how the intuition about removing empty sets is realized. Extensions to the previous strengthening (cf. Section 3.7) are printed bold.

Definition 3.9.1. (bisimulation between Petri Games)

Given two Petri games $\mathscr{G}^1$ and $\mathscr{G}^2$, a binary relation $R \subseteq \{\, (H_1, H_2) \mid M_1 \subseteq \mathscr{P}^1 \wedge M_2 \subseteq \mathscr{P}^2 \wedge (M^1 \in \mathscr{B}^1 \iff M^2 \in \mathscr{B}^2) \wedge H_1 \in \mathcal{H}(M_1) \wedge H_2 \in \mathcal{H}(M_2) \,\}$ is a *bisimulation* if for all $(H_1, H_2) \in R$:

(1) $\forall\, t_1 \in \mathscr{T}^1, H_1' \in \mathcal{H}(M_1').\, M_1 \xrightarrow{t_1}_{\mathscr{N}^1} M_1' \implies$ (FO)
$\exists\, t_2 \in \mathscr{T}^2, H_2' \in \mathcal{H}(M_2').\, M_2 \xrightarrow{t_2}_{\mathscr{N}^2} M_2' \wedge \mathscr{L}^1(t_1) = \mathscr{L}^2(t_2) \wedge$
repeat-constraint $\wedge\ (H_1', H_2') \in R$,

(2) $\forall\, t_2 \in \mathscr{T}^2, H_2' \in \mathcal{H}(M_2').\, M_2 \xrightarrow{t_2}_{\mathscr{N}^2} M_2' \implies$ (RE)
$\exists\, t_1 \in \mathscr{T}^1, H_1' \in \mathcal{H}(M_1').\, M_1 \xrightarrow{t_1}_{\mathscr{N}^1} M_1' \wedge \mathscr{L}^1(t_1) = \mathscr{L}^2(t_2) \wedge$
repeat-constraint $\wedge\ (H_1', H_2') \in R$.

where *repeat-constraint* is respectively defined as the conjunction of:

(a) $\forall\, s_i^1 \in H_1. \forall\, s_i^2 \in H_2. (\exists\, p^1 \in \mathscr{P}_S^1.\, p^1 \in pre^1(t_1) \wedge p^1 \in s_i^1$ (SE)
$\iff \exists\, p^2 \in \mathscr{P}_S^2.\, p^2 \in pre^2(t_2) \wedge p^2 \in s_i^2)$,

(b) $\boldsymbol{\forall\, s_i^1 \in H_1. \forall\, s_i^2 \in H_2. (\exists\, p^1 \in \mathscr{P}_E^1.\, p^1 \in pre^1(t_1) \wedge p^1 \in s_i^1}$ (EE)
$\boldsymbol{\iff \exists\, p^2 \in \mathscr{P}_E^2.\, p^2 \in pre^2(t_2) \wedge p^2 \in s_i^2)}$,

(c) $H_1' = \langle\, post^1(t_1), s_1^1 \backslash pre^1(t_1), s_2^1 \backslash pre^1(t_1), \ldots, s_n^1 \backslash pre^1(t_1) \,\rangle$, (H1)

(d) $H_2' = \langle\, post^2(t_2), s_1^2 \backslash pre^2(t_2), s_2^2 \backslash pre^2(t_2), \ldots, s_n^2 \backslash pre^2(t_2) \,\rangle$, (H2)

(e) $\boldsymbol{\forall\, s_i^1 \in H_1'. \forall s_i^2 \in H_2'. (s_i^1 = \emptyset \wedge s_i^2 = \emptyset) \implies}$ (RD)
$\boldsymbol{\forall\, i \leq j < n.\, s_j^1 := s_{j+1}^1 \wedge s_j^2 := s_{j+1}^2 \wedge s_n^1}$ ***and*** $\boldsymbol{s_n^2}$ ***are removed and*** $\boldsymbol{n}$ ***is reduced by one.***

Definition 3.9.2. (Bisimilarity of Petri Games)

Two Petri games $\mathscr{G}^1$ and $\mathscr{G}^2$ are *bisimilar* if there exists a bisimulation R according to Definition 3.9.1 with $(\langle In^1 \rangle, \langle In^2 \rangle) \in R$.

The two Petri games from Fig. 9 and from Fig. 10 are not longer bisimilar according to Definition 3.9.2. Transitions labeled by *testL* can be fired in both games immediately after firing *left*1. In the first game, there is a synchronous version of *testL* together with the environment player. This transition cannot be simulated in the second game because no environment player participates in the only transition labeled by *testL*. Therefore, the two games are not bisimilar anymore.

Bisimilarity covers all properties of Petri games to enable the translation of strategies. This is proven in Section 5.4. Some applications of bisimilarity are discussed in Section 4.

For the following, we stipulate $\sim$ to be the relation induced by Definition 3.9.2, i.e. $\mathscr{G}^1 \sim \mathscr{G}^2$ defines that there exists a bisimulation R following Definition 3.9.1 relating $\mathscr{G}^1$ and $\mathscr{G}^2$ such that $(\langle In^1 \rangle, \langle In^2 \rangle) \in R$.

As this is the final formulation of our definition of the equivalence we added bookmarks to parts of the equivalence to reference them easily during our proofs in Section 5. (FO) represents the forward direction of our bisimulation. (RE) does same for the return direction. (SE) and (EE) describe the additional constraints posed for the equal relation between system and environment places in the respective preset of a transition. (H1) and (H2) reference the additional constraints about the calculation of H'_1 and H'_2. (RD) is about the removal of empty sets in related history markings.

4 Applications

In the following, we discuss applications of bisimilarity. We give an example that it is possible to reduce the number of environment tokens. Furthermore, we report that allowing non-deterministic choices in strategies allow us to get rid of environment tokens in even more cases. Moreover, it is illustrated by an example how strategies between two bisimilar Petri games can be translated.

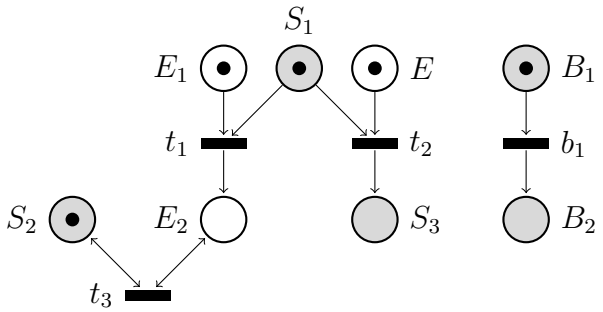

Figure 11: A Petri game with three system players and two environment players is depicted. The winner of the game cannot be determined using Adam because there are two environment players. All markings containing B_2 are bad markings.

4.1 Reduction of Environment Tokens

We show that two bisimilar Petri games can have a different number of environment tokens. This implies that the number of environment players can be reduced increasing the range of solvable Petri games by the automated solver Adam [6]. As already mentioned before, Adam determines the winner of Petri games which contain only a single environment player. It is written in Java and based on a fixed-point algorithm where the history of tokens is represented by *Binary Decision Diagrams (BDDs)* [2].

In Fig. 11 and Fig. 12, two bisimilar Petri games with a different number of environment tokens are depicted. Both games entail a system place B_1 which has to be prevented from firing its only transition b_1 reaching the place B_2. Thus, every marking containing B_2 is a bad marking. In the first game depicted in Fig. 11, the system player at S_1 can choose between t_1 and t_2. After firing t_1, the transition t_3 can be fired infinitely often, whereas after firing t_2, no transition except b_1 is enabled. In the second game depicted in Fig. 12, a choice between t_1 and t_2 takes place, too. However, each transition

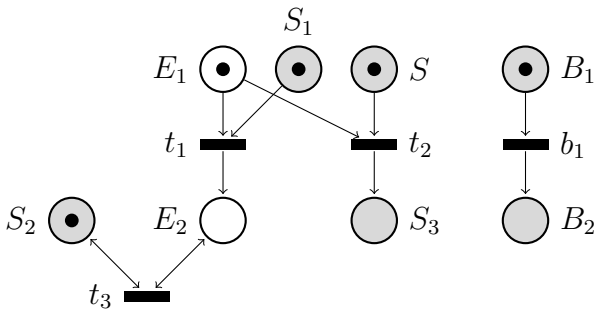

Figure 12: A Petri game with four system players and only one environment player is displayed. Therefore, ADAM can be utilized to find out that the system wins the game. All markings containing B_2 are bad markings.

has only an independent system place in its preset. After firing t_1, it is again possible to fire t_3 infinitely often, whereas firing t_2 leaves only b_1 enabled.

The difference between the two games is that the environment place E in the first game becomes the system place S in the second game and the preset of t_2 changes from $\{S_1, E\}$ in the first game to $\{E_1, S\}$ in the second game.

The bisimulation between the two games looks as follows:

$$
\begin{aligned}
R = \{\, &(\langle\, \{E, E_1, S_1, S_2, B_1\}\,\rangle, \langle\, \{E_1, S, S_1, S_2, B_1\}\,\rangle), && (1)\\
&(\langle\, \{E_2\}, \{E, S_2, B_1\}\,\rangle, \langle\, \{E_2\}, \{S, S_2, B_1\}\,\rangle), && (2)\\
&(\langle\, \{S_3\}, \{E_1, S_2, B_1\}\,\rangle, \langle\, \{S_3\}, \{S_1, S_2, B_1\}\,\rangle), && (3)\\
&(\langle\, \{B_2\}, \{E, E_1, S_1, S_2\}\,\rangle, \langle\, \{B_2\}, \{E_1, S, S_1, S_2\}\,\rangle), && (4)\\
&(\langle\, \{E_2, S_2\}, \{E, B_1\}\,\rangle, \langle\, \{E_2, S_2\}, \{S, B_1\}\,\rangle), && (5)\\
&(\langle\, \{B_2\}, \{E_2\}, \{E, S_2\}\,\rangle, \langle\, \{B_2\}, \{E_2\}, \{S, S_2\}\,\rangle), && (6)\\
&(\langle\, \{B_2\}, \{S_3\}, \{E_1, S_2\}\,\rangle, \langle\, \{B_2\}, \{S_3\}, \{S_1, S_2\}\,\rangle), && (7)\\
&(\langle\, \{E_2\}, \{B_2\}, \{E, S_2\}\,\rangle, \langle\, \{E_2\}, \{B_2\}, \{S, S_2\}\,\rangle), && (8)\\
&(\langle\, \{S_3\}, \{B_2\}, \{E_1, S_2\}\,\rangle, \langle\, \{S_3\}, \{B_2\}, \{S_1, S_2\}\,\rangle), && (9)\\
&(\langle\, \{B_2\}, \{E_2, S_2\}, \{E\}\,\rangle, \langle\, \{B_2\}, \{E_2, S_2\}, \{S\}\,\rangle), && (10)\\
&(\langle\, \{E_2, S_2\}, \{B_2\}, \{E\}\,\rangle, \langle\, \{E_2, S_2\}, \{B_2\}, \{S\}\,\rangle)\,\} && (11)
\end{aligned}
$$

The transition t_2 is the only difference between the two games. In both games, t_2 contains a system and an environment place in its preset. A token resides in all these places initially. Therefore, the tokens are uninformed about other players. We look at all pairs of history markings where t_2 is enabled as this is the only part where the bisimulation can possibly differentiate the two games.

From the initial pair (cf. pair (1) in R), pair (3) is reached after firing t_2. To fulfill (SE), we choose system place S_1 in the first game for system place S in the second game and vice versa. We do the same for E in the first game and E_1 in the second game to fulfill (EE). (3) is formally defined by (H1) and (H2). No empty sets occur implying that (RD) is trivially fulfilled. Therefore, t_2 fulfills the conditions of the bisimulation for pair (1). All other enabled transitions (t_1 and b_1) from said pair are the same in both games and R includes the correspondingly reached pairs (2) and (4).

The same argumentation referring to t_2 holds for (4) where said transition is enabled as well. Here, the only difference is that the disconnected transition b_1 was fired first which does not effect the informedness of S_1 and E in the first game and S and E_1 in the second game, respectively. The remaining pairs are caused by the different interleavings in which t_1, t_3, and b_1 can be fired. Therefore, R fulfills the definition of bisimulation and the games are bisimilar.

One can use the winning strategy for the second game derived by Adam and the bisimulation R to derive a winning strategy for the first game. For Adam, it is impossible to solve the first game because the solver only supports games with a single environment player. The winning strategy for both games is to activate t_1 and t_3 while deactivating t_2 and b_1. Firing t_3 infinitely often enables us to deactivate b_1 and thereby avoid all bad markings.

4.2 Non-Deterministic Strategies

We show that bisimilarity (cf. Definition 3.9.2) allows for further reductions of environment tokens when dropping the constraint of requiring deterministic choices at system places in a winning strategy.

Fig. 13 displays a Petri game which has no bad markings. There exists no deterministic and deadlock-avoiding decision for the system place S. Nevertheless, no bad behavior can occur as no such behavior is defined for the game. The environment decides between l, m, and r (abbreviations for *left*, *middle*, and *right*) and spawns another environment player such that the system has to decide between the two transitions the environment has not fired, respectively. These synchronous transitions have names in capital letters. After the environment fired l, the system has to decide between M and R. The crucial problem is that the environment can reach any place the system has to react to in two ways. The place A, from where R becomes enabled, can either be reached after the environment fired l or m.

After the decision of the environment, precisely two of the three places A, B, and C hold a token. The system has to decide for exactly one out of the two enabled transitions in order to be deterministic and deadlock-

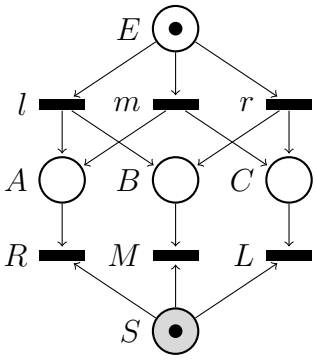

Figure 13: A Petri game is displayed where the system decides between two remaining transitions out of the three transitions l, m, and r which the environment did not fire. No deterministic, deadlock-avoiding, winning strategy exists even though the Petri game has no bad marking.

avoiding. Assume the system decides for transition R in reaction to the reachable marking $\{A, B, S\}$. Then, the system also has to make a choice for the reachable marking $\{B, C, S\}$. The only choice is L because M is already deactivated in response to the first marking. Activating R and L is not deterministic for the last remaining reachable marking $\{A, C, S\}$.

For transition M in response to the first marking $\{A, B, S\}$, L needs to be chosen for the third marking $\{A, C, S\}$. This again leaves only a non-deterministic choice for the second marking $\{B, C, S\}$. Deactivating all transitions is not deadlock-avoiding. We showed that no deterministic, deadlock-avoiding strategy for the game from Fig. 13 exists.

In Fig. 14, a bisimilar Petri game is depicted. The game also starts with an initial choice between the transitions l, m, and r by the environment. No additional environment players are spawned but one of the three separate places A, B, and C is reached. From these places, two transitions exist, respectively. The transitions are labeled in such a way that the system player at S can choose between the two transitions the environment has not fired before. All synchronous transitions can be identified by capital letters. This Petri game has no bad markings, either.

The game is won by the system because a deterministic, deadlock-avoiding strategy exists. The system simply chooses exactly one transition for each pair of transitions from the places A, B, and C. For instance, a winning strategy activates R in response to place A, L in response to B, and M in response to C while deactivating the other three transitions available as alternatives. The system can differentiate between the transitions as they share the labeling A, B, or C but differ in the preset, respectively.

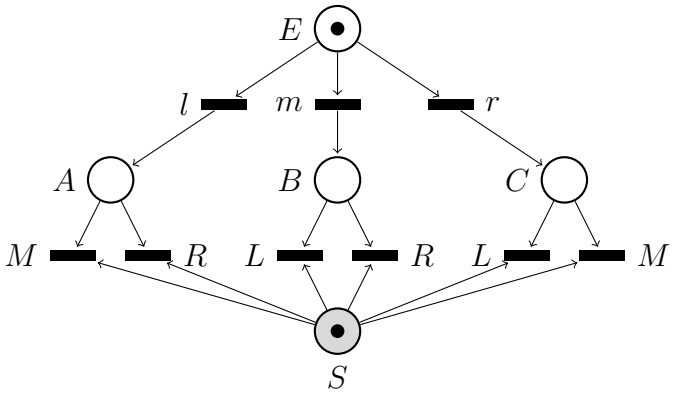

Figure 14: A bisimilar Petri game to the Petri game from Fig. 14 is depicted. This Petri game has no bad marking, either.

The two games from Fig. 13 and Fig. 14 are bisimilar according to Definition 3.9.2. The corresponding bisimulation R looks as follows:

$$R = \{\, (\langle\{S,E\}\rangle, \langle\{S,E\}\rangle), \quad (\langle\{A,B\},\{S\}\rangle, \langle\{A\},\{S\}\rangle),$$
$$(\langle\{A,C\},\{S\}\rangle, \langle\{B\},\{S\}\rangle), \quad (\langle\{B,C\},\{S\}\rangle, \langle\{C\},\{S\}\rangle),$$
$$(\langle\{A\}\rangle, \langle\{\}\rangle),\ (\langle\{B\}\rangle, \langle\{\}\rangle),\ (\langle\{C\}\rangle, \langle\{\}\rangle)\,\}$$

The first element relates the history markings of the two initial markings. From thereon, the three transitions l, m, and r are enabled in both games leading to the next three pairs. The first of these three pairs shows that tokens in the places S, A, and B in the first game can be simulated by S and A in the second game and vice versa. In both cases, the transitions labeled by M and R are enabled leading to the pairs $(\langle\{A\}\rangle, \langle\{\}\rangle)$ and $(\langle\{B\}\rangle, \langle\{\}\rangle)$ which are included in the bisimulation. For firing M and R, the two environment tokens in the first game are as informed as the one in the second game and the two system tokens in the two games are also informed in a bisimilar manner. Analog argumentations hold for the next two pairs. In both cases, an environment place from the first position of the history markings and a system place from the second position participate, respectively. From the last three pairs in the bisimulation, no transitions are enabled. Therefore, the relation fulfills the bisimulation condition.

The question arises whether the restriction to deterministic choices at system places is a good idea. Formally, the two Petri games from Fig. 13 and Fig. 14 have different winners despite both games lacking bad markings. In both games the same transitions can be fired and bisimilarity relates the two games as bisimilar. The second game can be solved by Adam which returns a deterministic winning strategy. This strategy can be translated into

a non-deterministic strategy for the first game. Both strategies are deadlock-avoiding and avoid bad markings. Therefore, we drop the constraint of deterministic strategies for the remainder of this thesis. This shows that ADAM and labelings can be used to derive (non-deterministic) winning strategies for games with more than one environment token.

4.3 Equivalence Example

We show how the winning strategies for the Petri games in Fig. 15 and Fig. 16 can be translated. The two Petri games model the situation where two system players have to mimic one environment player. In spirit of our running example, the environment models a bank robber arriving at a three-way junction where it can decide to either go *left* or *right*. Afterwards, the two system players can become informed about the environment's decision via the transitions *testL* and *testR*. The two system players can always decide together to either go *Left* or *Right*. The police should follow the bank robber in order to catch them. Therefore, the markings $\{EL, SR\}$, $\{TL, SR\}$, $\{ER, SL\}$, and $\{TR, SL\}$ are defined as bad markings representing that the bank robber and the police went into opposite directions.

The police is modeled as two independent officers which may observe certain behaviors of the environment via the transition *testL* and *testR*. In order to go into a certain direction, the two police officers get in the car and make a decision together in which direction to drive.

The difference between the two games is which police officer can observe which decision of the environment. In the Petri game from Fig. 15, each police officers monitor one direction of the junction independently. This implies that the first officer S_1 can identify when the bank robber goes *left*, whereas the second officer can identify the bank robber going *right*. Both police officers may sit in the front row of a single police car and look outside the window on the side they are sitting on. The Petri game from Fig. 16 models a different monitoring behavior of the police officers. The first police officer S_1 is lazy and cannot recognize the environment at all, whereas the second police officer S_2 can monitor both directions of the junction at the same time.

4.3.1 Proof of Bisimilarity

We give a bisimulation R between the two games in order to show that they are bisimilar. Afterwards, we unfold both games and give the corresponding winning strategies for the system. This gives us all necessary tools to show how to translate the winning strategies between the two games.

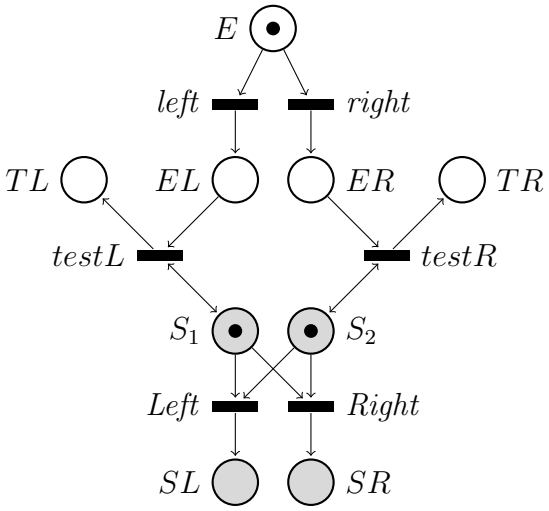

Figure 15: A Petri game is depicted where two police officers independently monitor a junction where a bank robber can go left or right. The first police officer can recognize the robber going left, whereas the second police officer can recognize the robber going right. The police officers can only start the pursuit of the bank robber together. It is bad behavior if the police officers go into a different direction than the bank robber. The corresponding bad markings are $\{EL, SR\}$, $\{TL, SR\}$, $\{ER, SL\}$, and $\{TR, SL\}$.

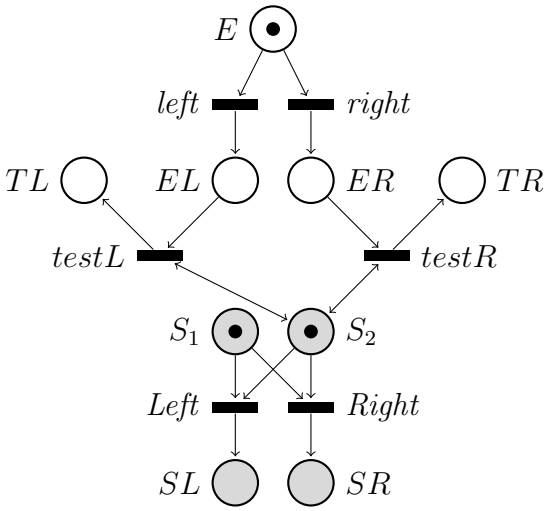

Figure 16: An alternation to the Petri game from Fig. 15 is displayed. The second police officer can recognize every behavior of the environment, whereas the first police officer cannot interact with the bank robber. The two police officers still need to follow the bank robber together. The bad markings remain $\{EL, SR\}$, $\{TL, SR\}$, $\{ER, SL\}$, and $\{TR, SL\}$.

The bisimulation R between the two games looks as follows:

$$
\begin{aligned}
R = \{\, &(\langle\{E, S_1, S_2\}\rangle, \langle\{E, S_1, S_2\}\rangle), && (\langle\{SR\}, \{EL\}\rangle, \langle\{SR\}, \{EL\}\rangle),\\
&(\langle\{EL\}, \{S_1, S_2\}\rangle, \langle\{EL\}, \{S_1, S_2\}\rangle), && (\langle\{SR\}, \{ER\}\rangle, \langle\{SR\}, \{ER\}\rangle),\\
&(\langle\{ER\}, \{S_1, S_2\}\rangle, \langle\{ER\}, \{S_1, S_2\}\rangle), && (\langle\{SL\}, \{EL\}\rangle, \langle\{SL\}, \{EL\}\rangle),\\
&(\langle\{TL, S_1\}, \{S_2\}\rangle, \langle\{TL, S_2\}, \{S_1\}\rangle), && (\langle\{SL\}, \{ER\}\rangle, \langle\{SL\}, \{ER\}\rangle),\\
&(\langle\{TR, S_2\}, \{S_1\}\rangle, \langle\{TR, S_2\}, \{S_1\}\rangle), && (\langle\{SL\}, \{TL\}\rangle, \langle\{SL\}, \{TL\}\rangle),\\
&(\langle\{SR\}, \{TL\}\rangle, \langle\{SR\}, \{TL\}\rangle), && (\langle\{SL\}, \{TR\}\rangle, \langle\{SL\}, \{TR\}\rangle),\\
&(\langle\{SR\}, \{TR\}\rangle, \langle\{SR\}, \{TR\}\rangle), && (\langle\{SL\}, \{E\}\rangle, \langle\{SL\}, \{E\}\rangle),\\
&(\langle\{SR\}, \{E\}\rangle, \langle\{SR\}, \{E\}\rangle), && (\langle\{EL\}, \{SL\}\rangle, \langle\{EL\}, \{SL\}\rangle),\\
&(\langle\{EL\}, \{SR\}\rangle, \langle\{EL\}, \{SR\}\rangle), && (\langle\{ER\}, \{SL\}\rangle, \langle\{ER\}, \{SL\}\rangle),\\
&(\langle\{ER\}, \{SR\}\rangle, \langle\{ER\}, \{SR\}\rangle)\,\}
\end{aligned}
$$

The games are identical except for the transition *testL* with which the police learns that the bank robber went left. In the first game (cf. Fig. 15), the police officer S_1 is in the preset of the transition, whereas in the second game (cf. Fig. 16), the police officer S_2 is in the preset. This difference is recognized by the bisimulation in the pair $(\langle\{TL, S_1\}, \{S_2\}\rangle, \langle\{TL, S_2\}, \{S_1\}\rangle)$ (cf. the first pair of the fourth line) which is reached after firing *left* followed by *testL*.

Only the transitions *Left* and *Right* are enabled from the underlying markings of the first pair of the fourth line. It holds that S_1 is as informed in the first Petri game as is S_2 in the second game as well as that S_2 is as informed in the first game as is S_1 in the second. Therefore, the places in the precondition fulfill the bisimulation condition for both transitions *Left* and *Right*. The reached pairs $(\langle\{SL\}, \{TL\}\rangle, \langle\{SL\}, \{TL\}\rangle)$ and $(\langle\{SR\}, \{TL\}\rangle, \langle\{SR\}, \{TL\}\rangle)$ are also in R (cf. the second pair of the fifth line and the first pair of the sixth line of R).

The remaining parts of the two games are identical and covered by R implying that the Petri games from Fig. 15 and Fig. 16 are bisimilar.

4.3.2 Unfoldings

The unfoldings of the two games can be found in Fig. 17 and in Fig. 18. We only need them to formally derive the respective winning strategies. They are not essential for the following discussion of strategy translation.

In the first game, S_1 and S_2 can each recognize one specific decision of the environment, respectively. Therefore, S_1 and S_2 are copied once including the following transitions and the thereby reached places. The resulting unfolding is displayed in Fig. 17. In the second game, only the second police officer S_2 can recognize any decision by the environment. Since there are two such

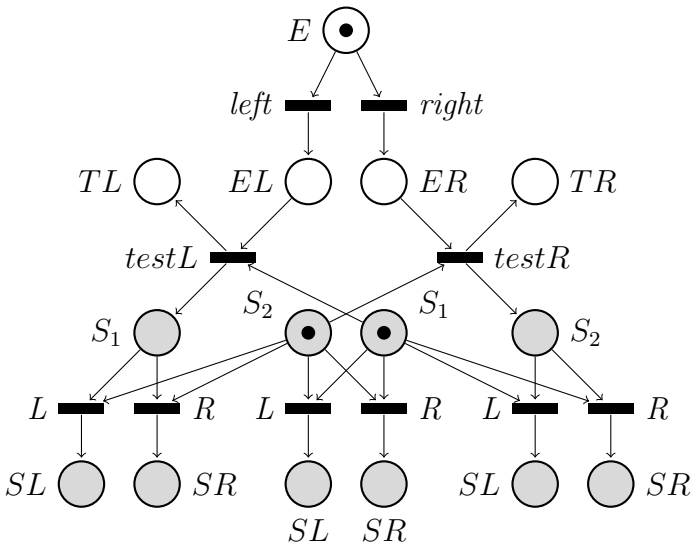

Figure 17: The unfolding of Fig. 15 is displayed. The places S_1 and S_2 are duplicated once for the returning arrows of the transitions *testL* and *testR*, respectively. Each of these two places has their outgoing transition *Left* and *Right* including the following places copied. We abbreviate *Left* by L and *Right* by R. The positions of the original places S_1 and S_2 have been swapped.

decisions, the place including the following transitions and thereby reached places is copied twice. The corresponding unfolding is depicted in Fig. 18.

4.3.3 Winning Strategies

The winning strategies for both games look similar. They are outlined in Fig. 19 and Fig. 20. In both cases, the system waits with its decision in which direction to drive until it has recognized the behavior of the environment via *testL* or *testR*, respectively. After that, both system players mimic the environment's decision together since one of the two players is guaranteed to have witnessed the bank robber's behavior.

For the first game, the system activates the transitions *testL* in S_1 and *testR* in S_2 to ensure that the environment's decision is noticed. Furthermore, S_2 activates *Left* and S_1 activates *Right* in case the other police officer has already made a decision, respectively. The unfolded places S_1 and S_2 can ensure that the environment either went left or right. Therefore, the system activates *Left* at the unfolded place S_1 and *Right* at the unfolded place S_2. These transitions represent that one police officer makes a decision while the other police officer is still in its initial place. The only reachable markings where all players made a decision are $\{TL, SL\}$ and $\{TR, SR\}$ which are no

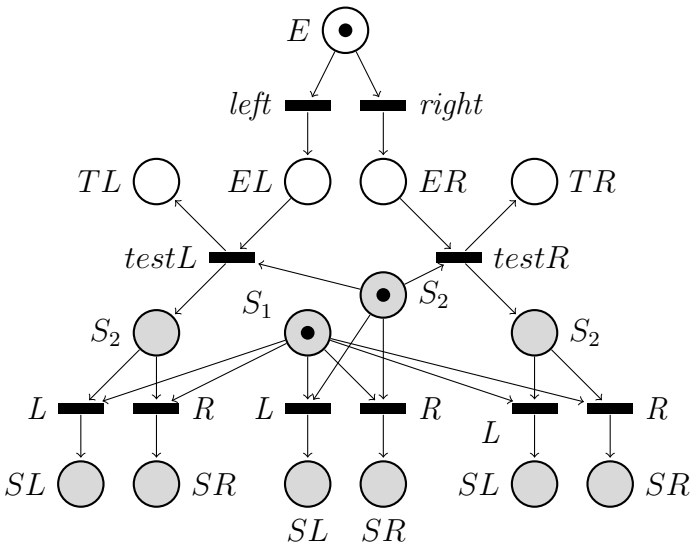

Figure 18: The unfolding of Fig. 16 is depicted. Both transitions *testL* and *testR* return to the place S_2 in the original game. Therefore, two copies are introduced including the following transitions *Left* and *Right* which are abbreviated by L and R. The following places SL and SR are also copied.

bad markings. The strategy is winning for the system.

The strategy for the second game is almost the same except that the second police officer S_2 always gets informed. Therefore, S_1 simply follows S_2's decision. S_1 activates all its transitions waiting for the decision of its partner, whereas the initial place S_2 activates *testL* and *testR*. In the place S_2 reached after firing *testL*, the transition *Left* is activated, whereas in the place S_2 reached after firing *testR*, the system activates *Right*. Again, the only reachable markings where all players made a decision are $\{TL, SL\}$ and $\{TR, SR\}$. This implies that the strategy is winning for the system.

4.3.4 Example Strategy Translation

In the following, we show how to translate the winning strategy for the Petri game from Fig. 15 into a winning strategy for the Petri game from Fig. 16. This example becomes generalized in Section 5.4 to an algorithm for all bisimilar Petri games. For the strategy translation, we utilize the winning strategy from Fig. 19, the Petri game from Fig. 16, and the bisimulation R.

The basic idea is to initially deactivate all transitions at system places in the second game. They can become activated based on a breadth-first search over the activated transitions in the given strategy. The search iterates over reachable history markings in the strategy. Places are unfolded only

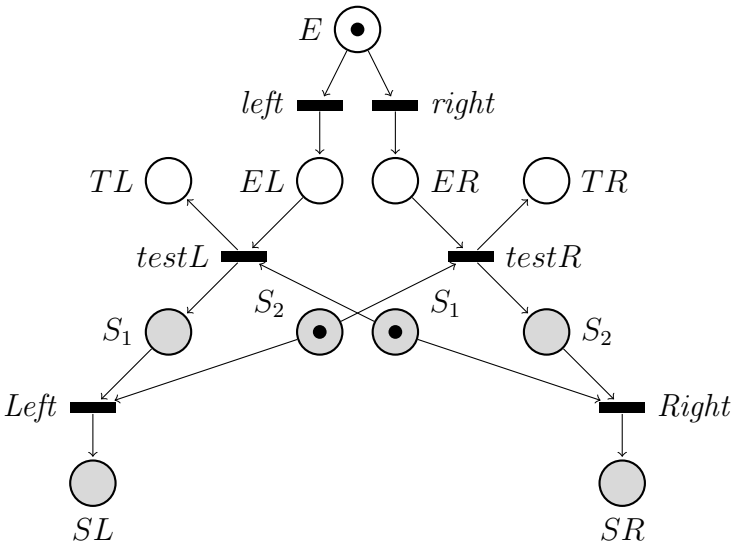

Figure 19: The winning strategy based on the unfolding from Fig. 17 of the Petri game from Fig. 15 is depicted. It is based on the system independently trying to collect information and immediately following the other system player if this player claims to have gathered information.

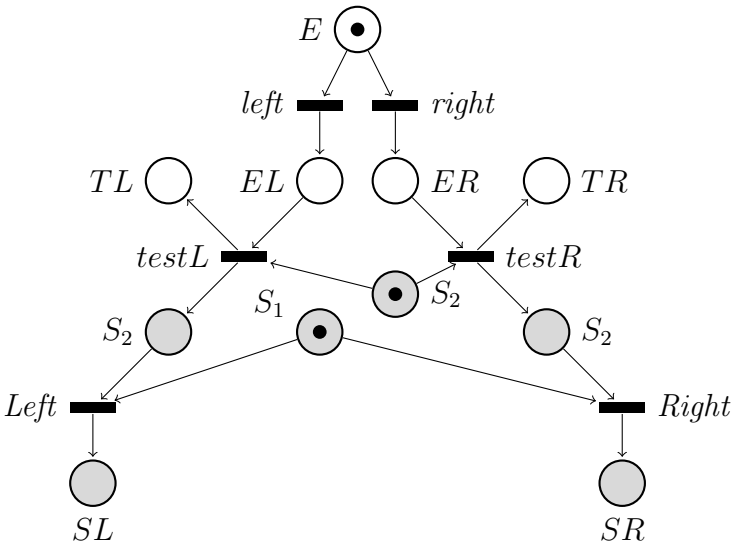

Figure 20: The winning strategy based on the unfolding from Fig. 18 of the Petri game from Fig. 15 is depicted. It is based on the one police officer doing all the work of recognizing the behavior of the environment and then taking the other (lazy) officer with her.

on demand at the second reaching. For a history marking, we mimic the transitions of the given strategy at the most recently unfolded version of the places in the preset. Bisimulation R defines which transition in the second game simulates the transitions allowed by the strategy.

The strategy for the first game allows exactly two runs of fired transitions: *left*, *testL*, *Left* and *right*, *testR*, *Right*. We start from the pair of initial history markings. The transitions *left* and *right* are enabled. We know that they are environment transitions (i.e. they have no system places in their preset, respectively). We therefore do not activate any local transitions at the initial marking in the second game and disallow to do so in the future.

We reach the history markings $\langle\,\{EL\},\{S_1,S_2\}\,\rangle$ and $\langle\,\{ER\},\{S_1,S_2\}\,\rangle$. For the first element, we know that the history marking $\langle\,\{EL\},\{S_1,S_2\}\,\rangle$ is reached in the second game by simulating the transition *left*. This is based on the first pair of the second row in R. We are sure about this reaching as said pair is the only pair in the bisimulation containing $\langle\,\{EL\},\{S_1,S_2\}\,\rangle$ at the first position. The treatment of a history marking occurring in multiple pairs in R is made precise in the general algorithm in Section 5.4.

Only the transition *testL* is enabled and it contains a system place in its preset. Therefore, we activate all transitions with the same label from the related history marking in the second game. Here, *testL* in the second game is the only candidate. As S_2 cannot be unfolded yet, we activate *testL* for the initial place S_2. The pair of history markings reached after firing *testL* is $(\langle\,\{TL,S_1\},\{S_2\}\,\rangle,\langle\,\{TL,S_2\},\{S_1\}\,\rangle)$ (cf. first pair in the fourth row of R).

We continue with the history marking $\langle\,\{ER\},\{S_1,S_2\}\,\rangle$ due to breadth-first search. The same treatment for *testR* as for *testL* is performed. The resulting pair of history markings is $(\langle\,\{TR,S_2\},\{S_1\}\,\rangle,\langle\,\{TR,S_2\},\{S_1\}\,\rangle)$ (cf. first pair in the fifth row of R).

For $(\langle\,\{TL,S_1\},\{S_2\}\,\rangle,\langle\,\{TL,S_2\},\{S_1\}\,\rangle)$, *testL* re-reaches the place S_2 in the second game. Therefore, we unfold the place by copying it and all following transitions. We change *testL* to lead to the unfolded place. In the strategy, the last transition *Left* is fired from the unfolded place S_1. For the current history marking of the second game, we know that the only possible transition with the same label *Left* has only S_2 in its preset. As S_2 was unfolded, we activate *Left* at the unfolded place. Transitions of unfolded places are deactivated, i.e. *Right* is deactivated at the unfolded place S_2.

For $(\langle\,\{TR,S_2\},\{S_1\}\,\rangle,\langle\,\{TR,S_2\},\{S_1\}\,\rangle)$, *testL* re-reaches the place S_2 in the second game for the second time. We therefore unfold it for the second time and allow only *Right* at the unfolded place. No transitions are enabled from the remaining pairs of history markings in the open list. The search therefore terminates and has constructed exactly the winning strategy from Fig. 20 for which we already argued why it is winning.

5 Characteristics

This section constitutes the second major achievement of this thesis. We prove thoroughly that bisimilarity as introduced by Definition 3.9.2 gives rise to sharing winning strategies between bisimilar Petri games. With the help of strategy translation, one can choose either—especially the easier to find—strategy for both games (cf. Section 3.1). First, we show that for two bisimilar Petri games, there exists a finite bisimulation according to Definition 3.9.1. In turn, we prove that bisimilarity is in fact an equivalence relation (i.e. it is reflexive, symmetric, and transitive).

The next step is to show that strategies of bisimilar Petri games are translatable. Therefore, we need to define in a precise manner what the term *bisimilar history* means and between which places of two related history markings bisimilar history is realized by a bisimulation. Finally, we give a construction for translating strategies between bisimilar Petri games and prove its correctness.

5.1 Existence of Finite Bisimulations

We prove that there exists a bisimulation of finite size between any two bisimilar Petri games. With this property, we can prove that finite bisimulations suffices to show the reflexivity of bisimilarity of finite Petri games.

Theorem 1. *(Existence of Finite Bisimulations)*
$\forall \mathscr{G}^1, \mathscr{G}^2. \mathscr{G}^1 \sim \mathscr{G}^2 \implies \exists$ *bisimulation* R *between* $\mathscr{G}^1$ *and* $\mathscr{G}^2. |R| < \infty$

Proof. The bisimilarity of $\mathscr{G}^1$ and $\mathscr{G}^2$ implies the existence of a bisimulation R' between the two games. We show how to reduce the size of R' to obtain a finite bisimulation R. We claim that every pair of history markings can be removed from R' where two empty sets occur at the same position to obtain R. (RD) enforces to remove empty sets at the same position for every enabled transition and the following pair of history markings. By definition, the initial pair of history markings cannot contain any empty sets. Therefore, two empty sets at the same position are never required for the reachable pairs of a bisimulation starting from the pair of initial history markings. Thus, all these pairs of history markings can be removed from R' resulting in R where it holds that $|R| \leq |R'|$.

For any two Petri games, we prove that the number of different pairs of history markings without empty sets at the same position is finite. Remember that we assumed Petri games to be finite and safe. A finite Petri games has only finitely many places. A safe Petri game has at most one tokens residing in each place for each sequence of enabled transitions. From these properties,

it follows that the maximum number of tokens for every game is restricted to the number of places. The maximum length of each history marking in R is $m = |\mathscr{P}^1| + |\mathscr{P}^2|$ as no two empty sets can be related at the same position. In the maximal case, both games have a token in each place and each token has a unique history, i.e. it is related to the empty set in the other history marking.

Let m be the length of a given history marking (i.e. the number of sets in the history marking) and let k be the number of tokens in the underlying marking. There exist maximally $m^k < \infty$ possibilities to distribute k tokens over m positions where more than one token can be at a certain position while certain positions can be empty but all tokens have to be distributed.

An upper bound for the number of pairs in R is $\sum_{i=2}^{m}(2 * i)^i$. In the worst case, the number of tokens in both games can vary between one and $|\mathscr{P}^i|, i = 1, 2$, respectively. The variation in the number of tokens occurs if it is possible to spawn new players. Therefore, the number of tokens i iterates from two to the maximum $m = |\mathscr{P}^1| + |\mathscr{P}^2|$. The number of tokens i is equal to the maximal length of one element of the pair. The other element of the pair has to be of the same length, hence $2 * i$. We over-approximate because we assume that the tokens can be freely moved between the two elements of a related pair from R. Nevertheless, the upper bound is finite as m is finite. □

5.2 Equivalence Relation

We prove that bisimilarity induced by our strengthening of the bisimulation between Petri nets is an *equivalence relation*.

Theorem 2. *(Equivalence Relation)*
The bisimilarity relation $\sim$ induced by Definition 3.9.2 is an equivalence relation (i.e. it is reflexive, symmetric, and transitive).

5.2.1 Reflexivity

$\sim$ is reflexive (i.e. $\forall \mathscr{G}. \mathscr{G} \sim \mathscr{G}$).

Proof. We build the bisimulation R by relating every history marking to itself, i.e. $R = \{ (H, H) \mid \exists M \subseteq \mathscr{P}_S \cup \mathscr{P}_E. H \in \mathcal{H}(M) \}$. This relation is infinite as $\mathcal{H}(M)$ is infinite for a single marking. It is proven in Section 5.1 that we can calculate a bound to the length of reachable history markings which makes the relation finite. This can be seen as an optimization but is not essential for the proof's correctness.

We claim that R is a bisimulation between $\mathscr{G}$ and $\mathscr{G}$. Requirements (FO) and (RE) of Definition 3.9.1 hold for every pair in R since the related Petri games are identical. Therefore, the same transitions are enabled having the same label and leading to the same underlying marking, respectively. (SE) and (EE) hold true for every pair in R by always choosing the same place. (H1) and (H2) are satisfied as every possible history marking is related to itself meaning that the needed one is also included in R. The last *repeat-constraint* (RD) holds for the same reason. □

5.2.2 Symmetry

$\sim$ is symmetric (i.e. $\forall \mathscr{G}^1, \mathscr{G}^2. \mathscr{G}^1 \sim \mathscr{G}^2 \implies \mathscr{G}^2 \sim \mathscr{G}^1$).

Proof. Let R be the bisimulation between the Petri games $\mathscr{G}_1$ and $\mathscr{G}_2$. For each pair $(q_1, q_2) \in R$, we include the pair (q_2, q_1) in R'. We claim that R' is a bisimulation between $\mathscr{G}^2$ and $\mathscr{G}^1$. The definition of bisimulation is symmetric for (FO) and (RE), i.e. the forward direction (FO) becomes the return direction (RE) and vice versa. The *repeat-constraints* (SE), (EE), and (RD) are independent of the proof direction (i.e. they hold for $\mathscr{G}^2 \sim \mathscr{G}^1$ iff they hold for $\mathscr{G}^1 \sim \mathscr{G}^2$). (H1) is the converse direction of (H2) and vice versa. Therefore, all constraints are fulfilled for R'. □

5.2.3 Transitivity

Let R be the bisimulation between $\mathscr{G}^1$ and $\mathscr{G}^2$ and R' the bisimulation between $\mathscr{G}^2$ and $\mathscr{G}^3$. The obvious approach of building R'' between $\mathscr{G}^1$ and $\mathscr{G}^3$ by including pairs (H_1, H_3) in R'' for all matching pairs $(H_1, H_2) \in R$ and $(H_2, H_3) \in R'$ does *not* work. The problem is that removing empty sets based on (RD) is defined between two games. This removing of empty sets can differ for two pairs of games.

Let us consider Fig. 21 as an example. Three Petri games $\mathscr{G}^a$, $\mathscr{G}^b$, and $\mathscr{G}^c$ are depicted consisting of various numbers of system tokens. In all three games, the transition t can be fired once. The three games only differ in the number of existing and participating system players.

The three games are pairwise bisimilar. We show that the aforementioned construction for proving transitivity does not work. The relation R between the Petri games $\mathscr{G}^a$ and $\mathscr{G}^b$ consists of the pairs $(\langle \{P_1, P_2\} \rangle, \langle \{P_4, P_5\} \rangle)$ and $(\langle \{P_3\}, \{P_2\} \rangle, \langle \{P_6, P_7\}, \{\} \rangle)$. The relation R' between the Petri games $\mathscr{G}^b$ and $\mathscr{G}^c$ entails the pairs $(\langle \{P_4, P_5\} \rangle, \langle \{P_8\} \rangle)$ and $(\langle \{P_6, P_7\} \rangle, \langle \{P_9\} \rangle)$.

The construction of the relation R'' between the Petri games $\mathscr{G}^a$ and $\mathscr{G}^c$ would result in the pair $(\langle \{P_1, P_2\} \rangle, \langle \{P_8\} \rangle)$. No further pair would be created because $\langle \{P_6, P_7\}, \{\} \rangle$ does not match $\langle \{P_6, P_7\} \rangle$. Nevertheless,

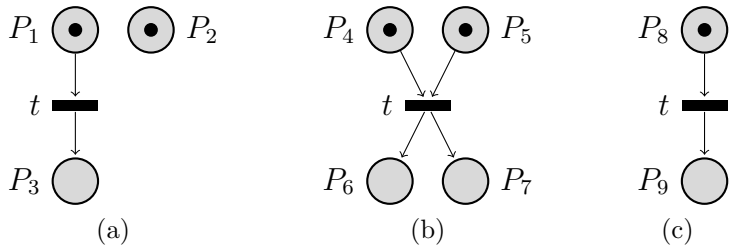

Figure 21: Three Petri games are depicted where a single transition t is enabled, respectively. The difference lays in the participating system players. In the first game, only one of two system players participates in the transition, whereas in the second game, both system players participate. In the third game, the only existing system player participates in t.

$(\langle\, \{P_3\}, \{P_2\} \,\rangle, \langle\, \{P_9\}, \{\} \,\rangle)$ with the added empty set for the second history marking has to be included in R'' in order to specify a bisimulation. As a way out, we add empty sets before performing the check for equality of the sets in the following proof of transitivity for bisimilarity.

$\sim$ is transitive (i.e. $\forall \mathscr{G}^1, \mathscr{G}^2, \mathscr{G}^3. \mathscr{G}^1 \sim \mathscr{G}^2 \wedge \mathscr{G}^2 \sim \mathscr{G}^3 \implies \mathscr{G}^1 \sim \mathscr{G}^3$).

Proof. Let R be the bisimulation between $\mathscr{G}^1$ and $\mathscr{G}^2$ and R' the bisimulation between $\mathscr{G}^2$ and $\mathscr{G}^3$. We construct the relation R'' between $\mathscr{G}^1$ and $\mathscr{G}^3$ by including (H_1^x, H_3^x) for all pairs $(H_1, H_2^1) \in R$ and $(H_2^2, H_3) \in R'$. The sets at each position of H_2^1 and H_2^2 have to match after adding empty sets such that no two added empty sets occur at the same position. This implies that the empty sets are added in the minimal manner and that the resulting pair consists of two finite history markings. H_1^x and H_3^x are H_1 and H_3 with empty sets added at the same positions as for H_2^1 and H_2^2, respectively.

For the Petri game from Fig. 21, $\langle\, \{P_6, P_7\}, \{\} \,\rangle$ and $\langle\, \{P_6, P_7\} \,\rangle$ match after we add the empty set at the second position in the second history marking. H_3 gets an empty set added at the second position such that $H_3^x = \langle\, \{P_9\}, \{\} \,\rangle$, whereas $H_1 = \langle\, \{P_3\}, \{P_2\} \,\rangle = H_1^x$ remains the same.

Before adding (H_1^x, H_3^x) to R'', we remove empty sets at the same position which may occur because $\mathscr{G}^2$ contains places which are simulated by empty sets in $\mathscr{G}^1$ and $\mathscr{G}^3$, respectively. This step is unrelated to the adding of empty sets in the previous step.

We claim that R'' is a bisimulation between $\mathscr{G}^1$ and $\mathscr{G}^3$. The set of history markings referring to the second Petri game from R and R' have to match except for empty sets because otherwise a certain transition is not enabled either in R or R'. Therefore, R or R' would not be a bisimulation in the first place.

For each enabled transition $t_1 \in \mathscr{T}^1$ from H_1 to H_1', the same transition is enabled from H_1^x to $H_1^{x\prime}$ because H_1 and H_1^x only differ in empty sets by construction and empty sets do not affect the enabledness of transitions. According to R, there exists a transition t_2 and a history marking H_2^1 such that t_2 is enabled from H_2^1 leading to $H_2^{1\prime}$, the transitions t_1 and t_2 have the same label, and $(H_1', H_2^{1\prime}) \in R$.

t_2 is enabled from H_2^2 to $H_2^{2\prime}$ as H_2^1 and H_2^2 only differ in the placement of empty sets. Because of $(H_2^2, H_3) \in R'$, firing t_2 from H_2^2 to $H_2^{2\prime}$ implies that there exists a transition t_3 and a history marking H_3' such that t_3 is enabled from H_3 to H_3', the transitions t_2 and t_3 have the same label, and $(H_2^{2\prime}, H_3') \in R'$. For the transition t_1 enabled from H_1^x to $H_1^{x'}$, we pick the transition t_3 and history marking $H_3^{x\prime}$. We know that t_3 is enabled from H_3^x to $H_3^{x\prime}$, that $\mathscr{L}(t_1) = \mathscr{L}(t_2) = \mathscr{L}(t_3)$, and that $(H_1^{x\prime}, H_3^{x\prime}) \in R''$ because we added the necessary empty sets by construction. This concludes our proof of the forward direction (FO). The converse direction (RE) works analogously.

Next, we inspect the *repeat-constraints*. (SE) and (EE) hold true because of the transitivity of the " $\Longleftrightarrow$ "- operator and the existence of the underlying history markings H_2^1 and H_2^2 for each pair $(H_1^x, H_3^x) \in R''$ which is based on $(H_1, H_2^1) \in R$ and $(H_2^2, H_3) \in R'$. We added empty sets in the minimal way for H_2^1 and H_2^2 to match. The added empty sets are taken over for H_1 and for H_3 such that they exactly match regarding their position when choosing places. Conditions (H1) and (H2) are fulfilled because we add empty sets in the minimal way, i.e. we never add empty sets in both markings H_2^1 and H_2^2 at the same position.

The last condition (RD) holds because we removed empty sets at the same position from $(H_1^x, H_3^x) \in R''$ as the last step of our construction. These empty sets do not occur because of our construction but because $\mathscr{G}^2$ can contain places which are simulated by empty sets both in $\mathscr{G}^1$ and $\mathscr{G}^3$. □

5.3 Bisimilar History

We define the term *bisimilar history* which is realized by bisimulation between two Petri games $\mathscr{G}^1$ and $\mathscr{G}^2$. Based on this concept we prove the possibility to translate strategies of bisimilar Petri games in Section 5.4.

Two history markings H_1 and H_2 incorporate bisimilar history iff $H_1 = \langle\, s_1^1, s_2^1, \ldots, s_n^1 \,\rangle$ and $H_2 = \langle\, s_1^2, s_2^2, \ldots, s_n^2 \,\rangle$ are of equal length n as well as $\forall\, 1 \leq i \leq n.\, s_i^1$ and s_i^2 incorporate bisimilar history, i.e. two sets of places s_i^1 and s_i^2 from the same position i have to incorporate bisimilar history.

Two sets of places s^1 and s^2 out of two history markings H_1 and H_2 incorporate bisimilar history iff $\forall\, j \in \{1,2\}.\, \forall\, p, p' \in s^j.\, p$ and p' incorporate bisimilar history and $\forall\, p^1 \in s^1.\, \forall\, p^2 \in s^2.p^1$ and p^2 incorporate bisimilar

history. This definition forces that both sets incorporate bisimilar history pair-wise among their elements, respectively, and that each pair of places including one place from the one set and one place from the other set incorporates bisimilar history.

The question whether two places from the same history marking H_i incorporate bisimilar history is positively answered by verifying that the sequences of fired transitions witnessed by the two places reaching the underlying marking of H_i are the same. By arguing about transitions, we realize that the visited places *and* the labels of fired transitions are the same as well as the number of firings per transition.

The question whether two places from different history markings H_1 and H_2 incorporate bisimilar history is positively answered by verifying that the labels of the sequence of transitions fired to reach the respective underlying markings of H_1 and H_2 and witnessed by the two places are the same. When comparing two games, we cannot argue about the visited places in both games as these can have different names.

5.3.1 History Markings realize Bisimilar History

We show that bisimilarity realizes bisimilar history. For each reachable pair of history markings, we show that each sequence of labels of transitions enabled in the first game resulting in the first history marking can be simulated in the second game such that the reached second history marking realizes bisimilar history and vice versa.

Theorem 3. *(History Markings realize Bisimilar History)*
If R is a bisimulation between Petri games $\mathscr{G}_1$ and $\mathscr{G}_2$, then the following holds for each pair $(H_1, H_2) \in R$: if the underlying markings of H_1 and H_2 are reachable in their respective Petri game via runs of the same length having the same labels then H_1 and H_2 incorporate bisimilar history for said runs.

Proof. By induction on the length n of the runs in the Petri game (i.e. the number of fired transitions):

$n = 0$: $(\langle In_1 \rangle, \langle In_2 \rangle)$ is in R by definition. In_1 and In_2 are obviously reachable in their respective Petri game. The history of each element of $\langle In_1 \rangle$ and $\langle In_2 \rangle$ is empty and thus In_1 and In_2 incorporate bisimilar history. Other markings are not reachable without firing a transition.

$n \to n + 1$: By the induction hypothesis, it is ensured that all pairs $(H_1, H_2) \in R$ reached after runs of length n with the same labels incorporate bisimilar history for the respective runs. These pairs are the only possibility to reach pairs $(H_1', H_2') \in R$ for runs of the length $n + 1$ according to (FO) and (RE). Let t_1 be the transition fired from H_1 to H_1' and t_2 the transition

from H_2 to H_2' such that t_1 and t_2 have the same label. The transitions either exist in both games or do not exist in either game as otherwise no bisimulation R between the games would exist. H_i' has the following form for $i = 1, 2$: $\langle\, post^i(t_i)\,\rangle^\frown\langle\, H_i \setminus pre^i(t_i)\,\rangle$ (cf. (H1) and (H2)). $^\frown$ denotes the concatenation of two vectors, $\langle\, H_i \setminus pre^i(t_i)\,\rangle, i = 1, 2$ represents a vector of length m for $i = 1, 2$, which preserves the obvious order from 1 to m, respectively, and the removal of elements of the places of the preset of t_i is performed per set of H_i. Empty sets are only removed if they occur in both history markings at the same position according to (RD).

The places of $post^i(t_i)$ incorporate bisimilar history per set and among the two sets because by (SE) and (EE), the participating system and environment places of the pre-condition are forced to have a partner with bisimilar history in the other Petri game. All participating places exchange their entire history when firing the transition. The pair (H_1, H_2) incorporates bisimilar history for the current run by the induction hypothesis.

Removing places keeps the property that the remaining places still have bisimilar history because empty sets are only removed if they occur at the same position in both history markings of a pair (cf. (RD)). Both history markings are shifted in the same way following (H1) for the first history marking and (H2) for the second history marking. This ensures that all respective positions in both pairs incorporate bisimilar history and shows that all pairs for runs of the length $n + 1$ incorporate bisimilar history.

This induction covers all *reachable* markings of the Petri game because it follows which transitions are enabled. □

Notice that (FO) and (RE) ensure the following property: if two sets that incorporate bisimilar history contain a different number of places then no transition is able to take advantage of this situation. For instance, a set with one element can have bisimilar history to the empty set and these two sets can be related by a pair of history markings in a bisimulation as long as a transition including the single place in its pre-condition is never enabled in any run of the underlying Petri game.

A pair of runs of same length is required to reach a pair of history markings. Such a pair of history markings can be reached via pairs of runs of different length. Therefore, history markings are defined to realize bisimilar history pairwise for runs of the same length. This re-reaching can happen in both games. Because the underlying marking is the same, the environment cannot take advantage of this situation despite the system possibly having the chance to make different decisions.

5.4 Algorithm for Strategy Translation

Two Petri games $\mathcal{G}^1$ and $\mathcal{G}^2$ have the *same winner* iff either both are won by the system or by the environment (i.e. $\exists$ winning strategy σ^1 for $\mathcal{G}^1$ iff $\exists$ winning strategy σ^2 for $\mathcal{G}^2$).

Bisimilarity realizes the following stronger claim for a different definition of strategies: $\mathcal{G}^1 \sim \mathcal{G}^2 \implies (\forall$ winning strategies σ^1 for $\mathcal{G}^1. \exists$ winning strategy σ^2 for $\mathcal{G}^2$ and $\forall$ winning strategies σ^2 for $\mathcal{G}^2. \exists$ winning strategy σ^1 for $\mathcal{G}^1$). The definition of winning strategies drops the requirement of deterministic decisions at system places. System places are allowed to activate more than one transition for a marking if a specific decision is not required to win the game.

The strategy is winning in the sense that it avoids bad markings and that it is deadlock-avoiding by always having at least one transition enabled in the strategy for all reachable markings if the same holds for the underlying unfolding. It further complies with (S2) and (S3).

Theorem 4. *(Strategy Translation)*
$\forall \mathcal{G}^1, \mathcal{G}^2. \mathcal{G}^1 \sim \mathcal{G}^2 \implies (\forall$ *winning strategies* σ^1 *for* $\mathcal{G}^1. \exists$ *winning strategy* σ^2 *for* $\mathcal{G}^2$ *and* $\forall$ *winning strategies* σ^2 *for* $\mathcal{G}^2. \exists$ *winning strategy* σ^1 *for* $\mathcal{G}^1$*) (assuming winning strategies without the requirement of deterministic choices at system places)*

Proof. The line of argumentation focusses on the first part of the conjunct since the second part works analogously. $\mathcal{G}^1 \sim \mathcal{G}^2$ implies that there exists a bisimulation R according to Definition 3.9.1 between $\mathcal{G}^1$ and $\mathcal{G}^2$. For each σ^1, we build σ^2 by allowing the same decisions at related history markings. We realize a breadth-first search over the strategy σ^1 to handle infinite strategies. This search represents the formal definition of the intuitive approach from Section 4.3.4. Based on R, we know where to simulate every decision of σ^1. We utilize the related history marking to unfold $\mathcal{G}^2$ (if necessary and possible) and allow transitions with the same label. This technique of allowing transitions is based on the fact that initially all system places deactivate all transitions, whereas no such restrictions are made for environment places. The bisimulation R defines which transitions in the second game can be used to simulate activated transitions in the first game.

A pseudo-code representation of the algorithm is given in Algorithm 1. We begin by taking $\mathcal{G}^2$ as the starting point of the strategy σ^2 we will construct. All decisions at system places are deactivated at σ^2. The visited places in σ^2 are required to determine when it is possible to unfold places. Line 5 defines the four-tuples that we perform a search over. The first two elements H_1 and H_2 are history markings related according to R. The third

element $F\colon \mathscr{P}^{\mathscr{G}^2} \to \mathscr{P}^{\sigma^2}$ stores which unfolded place in σ^2 is used to simulate decisions of σ^1. The bisimulation R is calculated before σ^2 is unfolded during the algorithm. Initially, $\mathscr{G}^2$ and σ^2 contain the same places. We therefore use the identify function Id on places from $\mathscr{G}^2$ as F. The fourth element V stores which history markings have been visited in the given strategy σ^1. This information is required to determine if σ^1 refuses on a possible unfolding. The open list *open* is used to store the queue of four-tuples waiting to be processed. We remove elements exclusively from the front and add at the back to realize a breadth-first search. The closed list *closed* is used to prevent us from processing a four-tuple twice as the second processing repeats the already done steps.

For a reached history marking H_1 in σ^1 which has been simulated in σ^2 by reaching H_2, F, and V (cf. Line 7), we search for each enabled transition t_1 for all partner transitions t_2 which fulfill the definition of bisimulation (cf. Line 9 to Line 13). The bisimulation property ensures that at least one such partner transition exists. Line 12 ensures that the transition can be activated. This may be forbidden if the transition is local and was not fired at a previously reached marking. In Line 34, the activation of transitions is forbidden. This can be required to ensure that the environment makes its decision before the system does. We simulate t_1 by activating t_2 at the places in its preset. We utilize the function F to find the most recent unfolding of the places for the currently simulated run (cf. Line 15).

Lines 17 to 29 inspect the necessity and possibility to unfold the places in the postset of t_2. We need to alter F when unfolding places resulting in F'. For each place *post* in the postset of t_2, the algorithm decides whether σ^1 requires an unfolding at this position. This is the case if σ^1 reaches a new history marking which can be identified by the history marking not being in V (cf. Line 18). The next decision to be made by the algorithm is whether the unfolding of the place *post* can lead to a new history marking in σ^2. This is only the case if *post* was already visited which we can check on behalf of $vp2$ which represents the set of so far visited places among all runs. If *post* was not visited before then an unfolding yields that the original place is unreachable. Therefore, no unfolding is necessary in this case.

The unfolding of *post* into *post*$'$ in Line 20 is performed by copying the place including all following transitions and thereby reached places. If *post* is a system place then we deactivate all copied transitions. For the case of *post* being an environment place, we are not justified to make such restrictions as the strategy is not allowed to restrict the environment. We update F' to return the place unfolded in this step for future decisions. The unfolded place *post*$'$ is added to set of visited places in σ^2. In case of not unfolding *post*, we add the original place to $vp2$ (cf. Line 27 and Line 29) as it is reached.

Algorithm 1 BFS-Algorithm for Strategy Translation

procedure BfsStrategyTranslation ($\mathscr{G}^1$, R, $\mathscr{G}^2$, σ^1):
  $\sigma^2 = \mathscr{G}^2$
  deactivate all system decisions in σ^2
  $vp2 = In^2$ // visited places in σ^2
  $open = [(In^1, In^2, Id, \{In^1\})]$
  $closed = \{\}$
  while $open \neq [\,]$ **do**
    $(H_1, H_2, F, V) = open.pop()$ // remove at first position
    for $t_1 \in \mathscr{T}^1$ **do**
      if $H_1 \xrightarrow{t_1}_{\sigma^1} H_1'$ **then** // enabled transition
        for $t_2 \in \mathscr{T}^2$ s.t. $\mathscr{L}(t_1) = \mathscr{L}(t_2)$ **do**
          if t_2 can be activated **then**
            if $(H_1, H_2) \in R \wedge H_2 \xrightarrow{t_2}_{\mathscr{G}^2} H_2' \wedge (H_1', H_2') \in R$ **then**
              for $pre \in {}^\bullet t_2$ **do**
                activate t_2 at $F(pre)$ // copy decision
              $F' = F$
              for $post \in t_2{}^\bullet$ **do**
                if $H_1' \notin V$ **then** // we should unfold
                  if $post \in vp2$ **then** // we can unfold
                    unfold $post$ into $post'$
                    if $post \in \mathscr{P}_S$ **then**
                      deactivate all decisions at $post'$
                    for p unfolded because of $post$ **do**
                      $F' = F'[\lambda^1(p) = p']$
                    add $post'$ to $vp2$
                  else
                    add $post$ to $vp2$
                else
                  add $post$ to $vp2$
              add (H_1, H_2, F, V) to $closed$
              $V' = V$ with H_1' added
              if $(H_1', H_2', F', V') \notin closed$ **then**
                add (H_1', H_2', F', V') to $open$
    Forbid the activation of not fired local system transition
  return σ^2

We extend V to V' by adding H_1' as a reached history marking in σ^1. Now, we update the closed and open list in the obvious way. The adding to the open list happens to the end of the list to realize breadth-first search.

For each transition $t_1 \in \mathscr{T}^1$ leading from H_1 to H_1', we know that there exists H_2 and a transition t^2 with $(H_1, H_2) \in R$ that can be used to simulate the label of t_1. We allow *all* of these possible transitions making the constructed strategy possibly non-deterministic at system places. The algorithm terminates when all elements from the open list have been processed.

For all Petri games $\mathscr{G}^1$ and $\mathscr{G}^2$, we claim that Algorithm 1 creates a winning strategy σ^2 for $\mathscr{G}^2$ given a bisimulation between the two games and a winning strategy σ^1 for $\mathscr{G}^1$. Each run of transitions allowed by σ^1 is either finite (i.e. it reaches a marking from which no transitions are enabled) or infinite. An infinite run has to repeat a history marking at some point as the number of reachable history markings is finite between the two games (cf. Theorem 1). This is an application of the pigeonhole principle [1]. The closed list identifies these repetitions in our algorithm. For two history markings, we can assume that the run repeats itself because a different run can only be caused by the environment as the strategy is fixed. Because we enumerate all runs, any deviating decision of the environment happens as soon as possible in another run and is handled there. The breadth-first search therefore iterates over runs of finite length.

It remains to show that there are only finitely many of such runs. The branching behavior of places is finite. For finite Petri games, we have a finite postset for each place. These properties assure that there are only finitely many choices before a history marking repeats itself. The interleaving of these choices is finite as well. Therefore, the algorithm terminates as it iterates over finitely many runs of finite length.

Next, we show that σ^2 allows the same runs as σ^1. This holds true because we allow exactly those transitions that have the same labels as transitions allowed by σ^1. The runs in σ^2 follow the unfolding of σ^1. Theorem 3 from Section 5.3.1 shows that the system places incorporate bisimilar history when making their decisions. Therefore, it is guaranteed that system places in σ^2 are as informed as the respective places in σ^1 when allowing a transition.

σ^2 cannot reach a bad marking because it only allows the same runs as σ^1. The existence of R proves that either both strategies reach a bad marking or none of them does and σ^1 never reaches a bad marking because it is winning. The constructed strategy σ^2 does not restrict environment transitions by construction. σ^2 is deadlock-avoiding because it decides for transitions with the same label as σ^1 and σ^1 is deadlock-avoiding by it being winning. R ensures that both strategies are based on games which have the same alternatives enabled when making a decision. Therefore, σ^2's decisions

are also deadlock-avoiding and σ^2 is winning. This proves that our algorithm not only terminates but produces a winning strategy for $\mathscr{G}^2$. □

In Appendix C, one can find the open list *open* before every iteration of the while-loop when using Algorithm 1 for the example from Section 4.3.4. In said example, we translate the winning strategy from Fig. 19 for the Petri game from Fig. 15 into a winning strategy for the bisimilar Petri game from Fig. 16. The algorithm terminates after seven iterations and returns the strategy from Fig. 20.

6 Related Work

Van Glabbeek and Vaandrager present equivalence notions on Petri nets [16]. These notions are divided to realize either true concurrency or interleaving semantics and to realize either branching time semantics or linear time semantics. The first distinction depends on whether transitions can take place in parallel (i.e. at the same time) or only one after the other. The second distinction spells out the fact whether the point in time when a decision is made is important (branching time) or not (linear time).

An *occurrence net equivalence* based on the isomorphism of the unfoldings of two nets realizes interleaving semantics and branching time. The *standard bisimulation equivalence* on Petri nets (cf. Section 2.3.1) leads to interleaving semantics and branching time semantics as well. The two equivalence notions leverage differently powerful algebraic tools in the sense of rules for equivalent Petri nets. The authors define another requirement for an equivalence called *causality*. All previously proposed equivalences fail to fulfill this requirement as they neglect which places causally depend on each other.

Partially ordered multisets (pomsets) represent the causality of places in a Petri net. *Pomset equivalence* based on two Petri nets having the same pomsets fails to realize branching time, i.e. it realizes linear time. By defining *pomset bisimulation equivalence*, this problem is fixed. The equivalence provides true concurrency semantics, realizes branching time, and conforms with causality. Another requirement the authors tackle is real-time consistency based on the nets real-time behavior which can be realized by *ST-bisimulation equivalence*. This equivalence handles the start and termination time of transitions and maintains the previously achieved features.

The equivalence presented in this thesis realizes interleaving semantics and branching time as it is an extension of the bisimulation between Petri nets. It further realizes some form of causality as pomsets can be judged to be more general than history markings. A history marking groups only equally informed places. However, it does not specify information about the corresponding informedness of two such sets which is in contrast to pomsets. When elaborating on future work, we discuss how to obtain a true concurrency semantics for Petri games.

Van Glabbeek extends the previous results by stating criteria to take into consideration when choosing an equivalence notion [15]. These criteria include the already mentioned branching time, concurrency, causality, and real-time consistency. The author further mentions inevitability and action refinement. Those are interesting for games as choices in two equivalent games should lead to the same (inevitable) game state. Action refinement forces the equivalence to be preserved when substituting sub-games which can

be viewed as another application of an equivalence notion on Petri games.

Jensen, Larsen, and Srba develop *Timed-Arc Petri Net Games* [11]. The underlying Timed-Arc Petri Nets introduce time to Petri nets. Transitions are guarded by time intervals for the places preceding them such that transitions are only enabled if all tokens in the preset have an age in the interval, respectively. Tokens possess an age, i.e. the time that passed since their creation. Per default, each firing of a transition resets the age of all participating tokens. This can be circumvented by *transport transitions*. Furthermore, Timed-Arc Petri Nets introduce a test of places for emptiness in order to fire transitions by so called *inhibitor arcs* to transitions. The continuous and discrete semantics for Timed-Arc Petri Nets coincide.

The game definition on Timed-Arc Petri Nets is based on dividing transitions to be either controllable or uncontrollable. The winning condition is based on safety, i.e. avoidance of certain markings which include the age of tokens. The authors show that Timed-Arc Petri Net Games with continuous and discrete time are incomparable. They prove that the games for both assumptions about time coincide when controllable transitions are restricted to be urgent. A transition is urgent if no time can pass as long as the transition is enabled. A synthesis algorithm is given for this special case which utilizes a timed bisimulation between markings to identify equivalent ages of tokens.

This work outlines another way to define games based on Petri nets. It introduces the concept of time constraints and shows that a bisimulation equivalence based on markings can be used to identify the equivalence of markings with different ages for tokens. Timed-Arc Petri Net Games do not model all non-deterministic choices explicitly. If the system and the environment decide to fire a transition at the same time then it is decided non-deterministically which of the two transitions is fired first.

Henzinger, Horowitz, and Majumdar define *rectangular hybrid games* as a restrictive generalization of hybrid automata to two-player games [9]. The restriction comprises constant lower and upper bounds for continuous time behavior in the underlying hybrid automata. These games utilize linear-time temporal logic (Ltl) [3] to define the winning sequences which have to be ensured by a strategy.

The paper shows decidability of the so-called Ltl *control problem*. For the purpose of the proof, *game trace equivalence*, *game similarity*, and *game bisimilarity* are defined as extensions of the respective notions from automata to two-player games. In contrast, Petri games do not realize a turn taking behavior of system and environment. Instead, they define the order of play by the structure of the underlying Petri net. Furthermore, the possibility of utilizing information from the history is more expressive in Petri games which does not allow us to use one of the aforementioned equivalences.

7 Conclusion

This section provides a summary of the presented material. We recall the most important facts during the development of bisimilarity. Moreover, we sum up the presented applications and the following theorems showing that bisimilarity induces an equivalence relation. In addition, we recall the algorithm for strategy translation between bisimilar Petri games. Regarding future work, we present possible extensions to the definition of bisimulation and further applications in the context of solving Petri games.

7.1 Summary

The starting point of our endeavor to an equivalence relation on Petri games is the standard bisimulation between Petri nets. The goal of our work is to translate winning strategies between bisimilar Petri games in order to solve easier games instead of hard ones. We especially focus on the reduction of environment tokens.

We prove that the standard bisimulation between Petri nets does not realize said goal as Petri games introduce system places as decision points for the strategy and bad markings as winning condition of the game. These points can be identified by a corresponding strengthening of the bisimulation (cf. Section 3.5).

A problem with a bisimulation between markings is that the system places do not only represent decisions points but they can accumulate history about fired transitions and visited places. This history can be essential to make decisions. For instance, a system place can decide for a transition *left* in response to being reached after firing *testL* and for *right* after witnessing *testR*. This example refers to our running example where the police tries to follow a bank robber.

Bisimulation based on markings cannot be used to identify the history of tokens in places. A possible way out is to work on unfoldings but this option is rejected because unfoldings can become infinite. Instead, we develop *history markings*. A history marking is a vector containing subsets of an underlying marking which incorporate bisimilar history.

Bisimulation is extended to consist of pairs of history markings. Such a pair represents that reaching an underlying marking via a sequence of transitions with certain labels in one game can be simulated in the other game with another sequence of transitions with the same labels and vice versa. Furthermore, the places are distributed in such a way that tokens in places from two sets of the same position of related history markings incorporate bisimilar history. The vector is used to realize an order of the

sets in a history marking. Thereby, places can be tested to originate from two sets of the same position of related history markings.

In order to utilize history markings, we need to check not only the enabledness of transitions but also that they have system places in the preset from the same position of history markings. This requirement represents that for each system place which has to decide on firing a transition in the one game there is a bisimilar informed system place in the other game such that the enabled transition has the same label as in the first game. If a system place in the one game decides to not fire a transition based on a certain history then there exists a system place in the other game which has witnessed bisimilar history and can simulate the decision. The respective strengthening is given in Section 3.7.

We show further that it does not suffice to perform this check for system places only but we rather need to track the history of environment places as well. At first, this may seem counterintuitive as environment players only perform non-deterministic choices. For these choices, the history is of no importance. However, the system can perform synchronous transitions with the environment in order to gain information about the environment. If we do not track the history of environment places we cannot ensure that after firing such a synchronous transition, the system places of two related history markings incorporate bisimilar history.

The problem can be solved by carrying out the same check for environment places as for system places when firing a transition, i.e. we search for each participating environment place of a transition in the one game for a corresponding environment place in the other game such that both places occur at the same position of the two related history markings. The corresponding strengthening is made precise in Section 3.9. The definitions outlined in this section establish the bisimilarity with the underlying bisimulation between history markings.

We depict an application where the bisimilarity relation defines two games as bisimilar where one game has a reduced number of environment players. Although the example is artificial, it shows that the concept of bisimilarity gives rise to a wider range of solvable games by Adam including some Petri games with two or more environment tokens. Furthermore, we show that the bisimilarity concept elicits further simplifications regarding the removing of environment tokens when allowing non-deterministic choices at system places.

On the one hand, the constraint for deterministic choices at system places forces the strategy to pick one particular direction even if another alternative is equally good as the picked one. Without the constraint for deterministic choices on the other hand, not all sources and developments of non-

determinism are modeled explicitly. To allow more simplifications, we allow non-deterministic strategies from this point onwards. We come back to this when discussing future work.

We prove that (non-deterministic) winning strategies between bisimilar Petri games are translatable. Together with the result that the bisimilarity relation based on the bisimulation between history markings is an equivalence relation, the goal of an equivalence on Petri games is achieved. We show by induction that two related history markings are reachable via runs resulting in bisimilar history for all tokens.

Given a winning strategy for one game, we therefore can translate this strategy into a strategy for another bisimilar game. The strategy predetermines an unfolding of the first game which is copied in the second game. The pairs of history markings determine uniquely where to simulate each decision.

7.2 Future Work

The first aspect of future work is affiliated to bisimilarity relating two Petri games as bisimilar although only one of the two games has a deterministic winning strategy. Additionally, we discuss an alternative to history markings. Furthermore, additional applications to the ones presented in Section 4 are considered.

7.2.1 Deterministic Strategies

It is an interesting task to strengthen bisimilarity in such a way that only Petri games are bisimilar with translatable deterministic strategies. One could build up a global store of necessary and possible decisions at system places when searching for a bisimulation. Such a strengthening would not require any alternation to the definition of Petri games but yield a stronger equivalence notion allowing less simplifications.

7.2.2 True Concurrency Semantics

We discussed related work about equivalence notions realizing true concurrency semantics [16]. It is challenging to realize such semantics and a corresponding equivalence for Petri games. Petri games could allow transitions of independent parts of the game to happen in parallel, i.e. more than one transition can be fired between two markings. This would reduce the number of markings to relate because all interleaving of a set of concurrent transitions would be replaced by one parallel execution.

It is necessary to alter history markings as the parallel execution of more than one transition should not exchange the history of players participating in different transitions. We presume that the notion for history markings and bisimulation can be extended to work in this case. Nevertheless, it is difficult to maintain a simple notation.

7.2.3 Applications

We have seen two applications of the equivalence relation that allowed to reduce the number of environment players for certain games. It is an open question whether it is possible to efficiently calculate the bisimulation between two games or prove the non-existence of such a bisimulation as outlined in [14] for standard bisimulation.

This problem leaves us with finding a smaller Petri game with which the existence of a bisimulation is tested. The general problem is based on finding the quotient of a Petri game. The quotient constitutes the smallest equivalent Petri games to a given one. A definition of smallest could be based on different concepts: (1) the number of environment tokens, (2) the number of system tokens, and (3) the number of places or transitions. This would define an order in which a Petri game is best reduced. If the existence of such a quotient can easily be calculated via the existence of a bisimulation to the given game this concept can run as a preprocessor of Adam.

An alternative approach of solving Petri games is centered on using *bounded synthesis* [5]. We limit the unfolding to a certain size and simulate runs of a certain length. This approach is suspected to find winning strategies faster than Adam but cannot prove the non-existence of a winning strategy. An equivalence notion on Petri games could be used to find places which have to be unfolded to find a winning strategy. A place could be unfolded and the resulting Petri game could be tested for equivalence to the game without the most recent unfolding. If the two games are equivalent the unfolding is unnecessary, whereas in the opposite case, a place is found which has to be unfolded.

8 Appendix

A Reachable Markings in Fig. 7 and Fig. 8

This is the set of reachable markings in the underlying nets $\mathscr{N}^1$ and $\mathscr{N}^2$ of the Petri games from Fig. 7 and Fig. 8.

$\mathscr{R}(\mathscr{N}^1) = \mathscr{R}(\mathscr{N}^2) = \{$
$\{apprCar1, waitingCar2, waitingCar3\}$,
$\{car1L, waitingCar2, waitingCar3\}$,
$\{car1R, waitingCar2, waitingCar3\}$,
$\{apprCar1, car2L, waitingCar3\}$,
$\{apprCar1, car2R, waitingCar3\}$,
$\{apprCar1, waitingCar2, car3L\}$,
$\{apprCar1, waitingCar2, car3R\}$,
$\{car1L, car2L, waitingCar3\}$,
$\{car1L, car2R, waitingCar3\}$,
$\{car1L, waitingCar2, car3L\}$,
$\{car1L, waitingCar2, car3R\}$,
$\{TTL, waitingCar2, waitingCar3\}$,
$\{TTR, waitingCar2, waitingCar3\}$
$\{car1R, car2L, waitingCar3\}$,
$\{car1R, car2R, waitingCar3\}$,
$\{car1R, waitingCar2, car3L\}$,
$\{car1R, waitingCar2, car3R\}$
$\{apprCar1, car2L, car3L\}$,
$\{apprCar1, car2L, car3R\}$,
$\{apprCar1, car2R, car3L\}$,
$\{apprCar1, car2R, car3R\}$,
$\{car1L, car2L, car3L\}$,
$\{car1L, car2L, car3R\}$,
$\{car1L, car2R, car3L\}$,
$\{car1L, car2R, car3R\}$,
$\{TLL, car2L, waitingCar3\}$,
$\{TRR, car2L, waitingCar3\}$
$\{TLL, car2R, waitingCar3\}$,
$\{TRR, car2R, waitingCar3\}$,
$\{TLL, waitingCar2, car3L\}$,
$\{TRR, waitingCar2, car3L\}$
$\{TLL, waitingCar2, car3R\}$,
$\{TRR, waitingCar2, car3R\}$,
$\{car1R, car2L, car3L\}$,
$\{car1R, car2L, car3R\}$,
$\{car1R, car2R, car3L\}$,
$\{car1R, car2R, car3R\}$,
$\{TLL, car2L, car3L\}$,
$\{TRR, car2L, car3L\}$
$\{TLL, car2L, car3R\}$,
$\{TRR, car2L, car3R\}$
$\{TLL, car2R, car3L\}$,
$\{TRR, car2R, car3L\}$
$\{TLL, car2R, car3R\}$,
$\{TRR, car2R, car3R\}$ $\}$

where TLL and TRR are abbreviations for $tooLateL$ and $tooLateR$.

B Bisimulation between Fig. 9 and Fig. 10

This is a bisimulation according to Definition 3.7.1 between the Petri games from Fig. 9 and from Fig. 10:

$$
\begin{aligned}
&R = \{ \\
&(\langle \{apprCar1, waitingCar2\} \rangle, \langle \{apprCar1, waitingCar2\} \rangle), && (1)\\
&(\langle \{car1L\}, \{waitingCar2\} \rangle, \langle \{car1L\}, \{waitingCar2\} \rangle), && (2)\\
&(\langle \{car1R\}, \{waitingCar2\} \rangle, \langle \{car1R\}, \{waitingCar2\} \rangle), && (3)\\
&(\langle \{waitingCar2'\}, \{apprCar1\} \rangle, \langle \{waitingCar2'\}, \{apprCar1\} \rangle), && (4)\\
&(\langle \{waitingCar2', car1L\}, \{\} \rangle, \langle \{waitingCar2'\}, \{car1L\} \rangle), && (5)\\
&(\langle \{waitingCar2'\}, \{car1L\} \rangle, \langle \{waitingCar2'\}, \{car1L\} \rangle), && (6)\\
&(\langle \{waitingCar2', car1R\}, \{\} \rangle, \langle \{waitingCar2'\}, \{car1R\} \rangle), && (7)\\
&(\langle \{waitingCar2'\}, \{car1R\} \rangle, \langle \{waitingCar2'\}, \{car1R\} \rangle), && (8)\\
&(\langle \{car2L\}, \{apprCar1\} \rangle, \langle \{car2L\}, \{apprCar1\} \rangle), && (9)\\
&(\langle \{car2R\}, \{apprCar1\} \rangle, \langle \{car2R\}, \{apprCar1\} \rangle), && (10)\\
&(\langle \{car1L\}, \{waitingCar2'\} \rangle, \langle \{car1L\}, \{waitingCar2'\} \rangle), && (11)\\
&(\langle \{car1R\}, \{waitingCar2'\} \rangle, \langle \{car1R\}, \{waitingCar2'\} \rangle), && (12)\\
&(\langle \{car2L\}, \{car1L\} \rangle, \langle \{car2L\}, \{car1L\} \rangle), && (13)\\
&(\langle \{car2L\}, \{car1L\}, \{\} \rangle, \langle \{car2L\}, \{\}, \{car1L\} \rangle), && (14)\\
&(\langle \{car2R\}, \{car1L\} \rangle, \langle \{car2R\}, \{car1L\} \rangle), && (15)\\
&(\langle \{car2R\}, \{car1L\}, \{\} \rangle, \langle \{car2R\}, \{\}, \{car1L\} \rangle), && (16)\\
&(\langle \{car2L\}, \{car1R\} \rangle, \langle \{car2L\}, \{car1R\} \rangle) && (17)\\
&(\langle \{car2L\}, \{car1R\}, \{\} \rangle, \langle \{car2L\}, \{\}, \{car1R\} \rangle), && (18)\\
&(\langle \{car2R\}, \{car1R\} \rangle, \langle \{car2R\}, \{car1R\} \rangle), && (19)\\
&(\langle \{car2R\}, \{car1R\}, \{\} \rangle, \langle \{car2R\}, \{\}, \{car1R\} \rangle), && (20)\\
&(\langle \{car1L\}, \{car2L\} \rangle, \langle \{car1L\}, \{car2L\} \rangle), && (21)\\
&(\langle \{car1R\}, \{car2L\} \rangle, \langle \{car1R\}, \{car2L\} \rangle), && (22)\\
&(\langle \{car1L\}, \{car2R\} \rangle, \langle \{car1L\}, \{car2R\} \rangle), && (23)\\
&(\langle \{car1R\}, \{car2R\} \rangle, \langle \{car1R\}, \{car2R\} \rangle) \} && (24)
\end{aligned}
$$

C Example Run of Algorithm 1

The open list *open* of Algorithm 1 is enumerated before each iteration of the while-loop for the example from Section 4.3.4. The algorithm translates the winning strategy from Fig. 19 for the Petri game from Fig. 15 into a winning strategy for the bisimilar game from Fig. 16. Termination occurs after seven iterations because *open* becomes empty. The algorithm returns the strategy from Fig. 20. S_2' denotes the first unfolding of S_2, S_2'' the second one.

open before 1st Iteration:

$$[\,(\langle\,\{In^1\}\,\rangle, \langle\,\{In^2\}\,\rangle, Id, V_1 := \{In^1\})\,]$$

open after 1st Iteration:

$$[\,(\langle\,\{EL\}, \{S_1, S_2\}\,\rangle, \langle\,\{EL\}, \{S_1, S_2\}\,\rangle, Id, V_2^L := V_1 \cup \{\langle\,\{EL\}, \{S_1, S_2\}\,\rangle\}),$$
$$(\langle\,\{ER\}, \{S_1, S_2\}\,\rangle, \langle\,\{ER\}, \{S_1, S_2\}\,\rangle, Id, V_2^R := V_1 \cup \{\langle\,\{EL\}, \{S_1, S_2\}\,\rangle\})\,]$$

open after 2nd Iteration:

$$[\,(\langle\,\{ER\}, \{S_1, S_2\}\,\rangle, \langle\,\{ER\}, \{S_1, S_2\}\,\rangle, Id, \{In^1, \langle\,\{EL\}, \{S_1, S_2\}\,\rangle\}),$$
$$(\langle\,\{TL, S_1\}, \{S_2\}\,\rangle, \langle\,\{TL, S_2\}, \{S_1\}\,\rangle, Id[\,S_2 = S_2'\,], V_3^L := V_2^L \cup \{\langle\,\{TL, S_1\}, \{S_2\}\,\rangle\})\,]$$

open after 3rd Iteration:

$$[\,(\langle\,\{TL, S_1\}, \{S_2\}\,\rangle, \langle\,\{TL, S_2\}, \{S_1\}\,\rangle, Id[\,S_2 = S_2'\,], V_3^L := V_2^L \cup \{\langle\,\{TL, S_1\}, \{S_2\}\,\rangle),$$
$$(\langle\,\{TR, S_2\}, \{S_1\}\,\rangle, \langle\,\{TL, S_2\}, \{S_1\}\,\rangle, Id[\,S_2 = S_2''\,], V_3^R := V_2^R \cup \{\langle\,\{TL, S_1\}, \{S_2\}\,\rangle\})\,]$$

open after 4th Iteration:

$$[\,(\langle\,\{TR, S_2\}, \{S_1\}\,\rangle, \langle\,\{TL, S_2\}, \{S_1\}\,\rangle, Id[\,S_2 = S_2''\,], V_3^R := V_2^R \cup \{\langle\,\{TL, S_1\}, \{S_2\}\,\rangle),$$
$$(\langle\,\{SL\}, \{TL\}\,\rangle, \langle\,\{SL\}, \{TL\}\,\rangle, Id[\,S_2 = S_2'\,], V_4^L := V_3^L \cup \{\langle\,\{SL\}, \{TL\}\,\rangle\})\,]$$

open after 5th Iteration:

$$[\,(\langle\,\{SL\}, \{TL\}\,\rangle, \langle\,\{SL\}, \{TL\}\,\rangle, Id[\,S_2 = S_2'\,], V_4^L := V_3^L \cup \{\langle\,\{SL\}, \{TL\}\,\rangle\}),$$
$$(\langle\,\{SR\}, \{TR\}\,\rangle, \langle\,\{SR\}, \{TR\}\,\rangle, Id[\,S_2 = S_2''\,], V_4^R := V_3^R \cup \{\langle\,\{SR\}, \{TR\}\,\rangle\})\,]$$

open after 6th Iteration:

$$[\,(\langle\,\{SR\}, \{TR\}\,\rangle, \langle\,\{SR\}, \{TR\}\,\rangle, Id[\,S_2 = S_2''\,], V_4^R := V_3^R \cup \{\langle\,\{SR\}, \{TR\}\,\rangle\})\,]$$

References

[1] M. Ajtai. The complexity of the pigeonhole principle. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 346–355. IEEE, 1988.

[2] S.B. Akers. Binary Decision Diagrams. *Computers, IEEE Transactions on*, 100(6):509–516, 1978.

[3] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

[4] J. Esparza and K. Heljanko. *Unfoldings - A Partial-Order Approach to Model Checking*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 2008.

[5] B. Finkbeiner. Bounded Synthesis for Petri Games. In *Correct System Design*, pages 223–237. Springer, 2015.

[6] B. Finkbeiner, M. Gieseking, and E.-R. Olderog. ADAM: Causality-Based Synthesis of Distributed Systems. In *Computer Aided Verification*, pages 433–439. Springer, 2015.

[7] B. Finkbeiner and E.-R. Olderog. Petri Games: Synthesis of Distributed Systems with Causal Memory. In *Proceedings Fifth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2014, Verona, Italy, September 10-12, 2014.*, volume 161 of *EPTCS*, pages 217–230, 2014.

[8] J. Hecking-Harbusch. A Game-Based Semantics for CSP. Bachelor's Thesis, Saarland University, 2015.

[9] T. Henzinger, B. Horowitz, and R. Majumdar. *Rectangular Hybrid Games*. Springer, 1999.

[10] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.

[11] P.G. Jensen, K.G. Larsen, and J. Srba. Real-Time Strategy Synthesis for Timed-Arc Petri Net Games via Discretization. In *Model Checking Software*, pages 129–146. Springer, 2016.

[12] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13(1):85–108, 1981.

[13] E.-R. Olderog. Strong bisimilarity on nets: a new concept for comparing net semantics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 549–573. Springer, 1988.

[14] R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[15] R.J. van Glabbeek. Structure Preserving Bisimilarity, Supporting an Operational Petri Net Semantics of CCSP. In *Correct System Design*, pages 99–130. Springer, 2015.

[16] R.J. van Glabbeek and F. Vaandrager. Petri Net Models for Algebraic Theories of Concurrency. In *PARLE Parallel Architectures and Languages Europe*, pages 224–242. Springer, 1987.