
Saarland University

Faculty of Mathematics and Computer Science


Bachelor's Thesis

# Translating Asynchronous Games for Distributed Synthesis


**Author**
Raven Beutner

**Supervisor**
Prof. Bernd Finkbeiner

**Advisor**
Jesko Hecking-Harbusch



**Reviewers**
Prof. Bernd Finkbeiner
Prof. Gert Smolka

Submitted: 25th July 2019

**Eidesstattliche Erklärung**

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig 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, 25$^{\text{th}}$ July, 2019

# Abstract

Synthesis is the ambitious task of automatically generating an implementation fulfilling a given specification against all possible inputs or proving the unrealizability of the specification, i.e, the absence of a fulfilling implementation. As the user does not need to come up with an implementation herself but gets presented with a, by default, correct one, synthesis offers maximal convenience. In distributed systems, individual processes act both locally and communicate with one another without having full information about the global state of the system. For distributed synthesis we hence try to generate a family of individual implementations, that governs the processes such that the system as a whole satisfies a given objective, independent of received inputs. We explicitly consider systems where the processes share no information when acting concurrently but exchange their entire causal past, i.e., all information known to them, upon communication.

So far the synthesis problem for this class of systems has been studied in terms of either Petri games or control games. In both the overall system is modelled as either a Petri net or an asynchronous automaton and the synthesis problem consists of finding a restriction on the global behavior that accomplishes a given objective. Petri games partition the places of a net as either system or environment and allow only tokens on system places to forbid transitions, while tokens on environment places are regarded as adversary players that cannot be controlled. By contrast, control games split the actions of an asynchronous automaton as either controllable or uncontrollable. Individual processes can refuse controllable actions and thereby restrict the executions of the automaton, while uncontrollable ones cannot be thwarted and have to be accounted for in any execution.

In both frameworks it is an open question whether the existence of a correct implementation for the system is decidable. There are, however, interesting classes for which decision procedures exist. The precise connection between both games and therefore the question whether existing results can be transferred to the other type was, so far, unknown.

In this thesis we establish the first formal connection of Petri games and control games by providing exponential translations in both directions. We show that our translations yield structurally equivalent games, in the sense that they admit weak bisimilar implementations. In addition to our upper bound we provide lower bounds in both directions showing an intrinsic trade off between the two frameworks. Our translation allow for the transfer of existing decidability results to the respective other game type. We exemplary outline the newly identified decidable classes of control games where at most one process comprises controllable behavior as well as Petri games with acyclic communication architectures.

# Acknowledgements

Firstly, I would like to thank my advisor Jesko for his support during the last few months including the numerous interesting discussions. I would also like to express my gratitude for my supervisor Prof. Finkbeiner who provided me with this challenging topic and, at the same time, gave room to discover the exact scope of this thesis on my own. In addition to Prof. Finkbeiner, I would like to thank Prof. Smolka both for offering to be the second reviewer of this thesis but most importantly for being my academic advisor and helping me with important decision during the past three years. I also place on record, my sincere thank you to my friends for making life in Saarbrücken truly enjoyable. A particular thanks goes to Jule and Florian for proofreading this thesis. Last but not least I would like to thank my family for their unwavering support.

# Contents

## Chapter 1

# Introduction

Verification is the task of checking a given implementation against a provided specification, i.e., either prove that the specification is obeyed or return a counterexample that witnesses an execution that violates the specification. A user can thus write an implementation and, using verification, guarantee that it behaves as intended. Synthesis, on the other hand, is the ambitious task of *automatically* generating an implementation fulfilling a given specification or proving the non-existence of one. For a user, synthesis offers maximal convenience: Instead of coming up with an implementation herself and afterwards verifying it, she is presented with an automatically generated and, by default, correct solution or proof that her specification is unrealisable.

In this thesis we consider *reactive systems*, i.e., systems that might run forever and continuously interact with the environment. In a reactive setting, a synthesized implementation should thus fulfil a specification against all possible inputs received in a potentially infinite execution. It is therefore natural to consider synthesis as a game played between the system and the environment where winning objectives for the system are imposed. While the environment is responsible for generating continuous inputs, the system needs to react such that the overall execution fulfils the objective. System and environment can hence be seen as antagonists: The system tries to accomplish the winning objective, whereas the environment attempts to play such that the system fails to do so. An attempt to win a game from the view of the system is often condensed as a *strategy*, i.e., a description of how to react to different states of the game. Since a winning strategy needs to achieve the objective against all possible environment behavior, it corresponds to an implementation fulfilling the specification against all possible inputs.

More specifically, our work is concerned with *distributed* reactive systems. A distributed system comprises multiple individual processes that act independently and possess incomplete information about the state of the global system. As an example, we can consider a network as a distributed system where the individual components of the network are accommodated as distinct processes, that, as in real-world systems, act independently. We can view synthesis in this setting as a game played between a collection of system players against the environment. To synthesize distributed systems we are interested in a *distributed strategy* that governs the local processes such that the system as a whole satisfies an objective, independent of the inputs that are received from the environment. A distributed strategy hence prescribes the behavior of individual players based on the individual partial information.

The behavior of the overall system is often described by an underlying arena. Synthesis of implementations can then be considered as finding restrictions to this behavior where every execution in the conditioned system fulfils a given objective. To represent the aspects of environment and system responsibilities one often imposes additional constraints on the model that allow a strategy to restrict *parts* of the behavior, while other parts cannot be controlled. Finding a winning strategy hence consists of finding a restriction to the overall behavior that

achieves the objective against all behavior that cannot be averted, i.e., all environment controlled moves. As the underlying arena describes a distributed system, we are not interested in *global* restrictions of the behavior but aim for ones that originate from decisions that can be made by the *local* protagonists. The restriction must thus be made in accordance with the local information of each player. Distributed synthesis has firstly been studied in a synchronous setting [25] where the problem is given in terms of an architecture that comprises all processes as well as communication channels between them. The processes can synchronously read from and write to the communication channels that are associated to them and try to, as a whole, create execution admitted by an LTL specification. By contrast, this thesis is concerned with *asynchronous* distributed systems, i.e., systems where individual components progress concurrently and at different speeds. We hence consider models that describe concurrent executions of individual components in a system and try to restrict an underlying behavior in accordance with an objective. Two natural models for concurrent and asynchronous executions are Petri nets and asynchronous automata.

In distributed systems, individual players progress on their own and distributed strategies prescribe the next move of each. As the players act on incomplete information it is natural to contemplate on the explicit information primitive underlying such systems, i.e., ask the question of how the local information of the players changes during the course of a game. In interesting models the players should possess the possibility to communicate with one another, i.e., apart from playing concurrently, they ought to be admitted to synchronize and exchange information. One popular paradigm to model this exchange of information is *causal memory*: While acting independently processes share no information about one another, whereas they exchange their entire casual past upon synchronization. Decidability of asynchronous synthesis with causal memory has, so far, been studied independently from each other in two distinct game types: Petri games and control games.

**Petri Games [12]** Petri games are distributed games played between tokens on an underlying Petri net. As Petri nets are intuitive representations of concurrent executions, Petri games naturally define asynchronous games. The places of the net are partitioned into system and environment. The tokens progress as defined by the underlying net where communication between them is accommodated as shared transitions. The objectives of a strategy is to restrict the asynchronous execution of the Petri net such that a winning condition imposed on the tokens of the net is fulfilled. It is defined in terms of a branching process, which can be seen as a restriction on the possible behavior of the game based on the previous execution of the individual token. To distribute responsibilities of system and environment, i.e, declare which parts of the behavior can be controlled and which cannot we conceptually sort the tokens in the net as system and environment players. If a token resides on a system place we regard it as a system player for which a strategy can control what transitions to allow. A token on an environment place is viewed as an environment player whose moves cannot be prohibited. A strategy can hence govern the course of all system players while having to account for all decisions of the environment, observed as moves of the environment players.

**Control Games [14, 22]** Control games are played on asynchronous automata, which are compositions of local processes that synchronize on shared actions. The actions in the automaton are distributed as either controllable or uncontrollable. The processes in the automaton progress as prescribed by the automaton, i.e., move independently but synchronize if an action involving multiple processes is executed. A strategy should restrict the execution of the game, i.e., prohibit certain sequences in the automaton, such that a winning objective is accomplished on all unrestricted executions. The description of a strategy is defined as an individual set of rules for each process. Each of which is condensed as a function mapping previous executions to decisions of what actions to enable. To model the distributed character of asynchronous automata, each process makes its decision based on an explicit local view on the overall exe-

cution. At every point of the execution, all processes decide what actions to allow, proceeded by executing one where all involved processes agreed on. By proscribing actions, the overall executions are hence restricted. Unlike Petri games where players are regarded as either system players or environment players, the environment responsibilities in control games is phrased using uncontrollable actions. Even though all processes can decide which actions to allow, they can only do so for controllable actions. Uncontrollable ones, on the other hand, cannot be thwarted and are conceptually controlled by the environment. The strategy should hence govern the behavior of the global automaton such that the system as a whole satisfies an objective and while doing so has to account for all uncontrollable actions.

There has been a significant amount of work for both game types by identifying classes for which the existence of winning strategies is decidable. While both game types share decisive characteristics the precise connection between them was, so far, unknown and so was the question of whether existing results can be transferred between both types. Progress in terms of decidability results has been made separately in both "camps". In this thesis, we provide formal translations in both directions and thereby establish that both models are equally expressive. To this extent we introduce the intuitive notion of *strategy-equivalence* that relies on a weak bisimulation between the admitted executions in two strategies. We show that our translations yield strategy-equivalent games, and thereby preserves the structure of winning strategies.

**High-Level Descriptions** While both game types admit strategies based on causal information, the concrete formalisms describing which parts of the game are controlled by the system and respectively which by the environment, differs. In Petri games, the places are partitioned as system and environment ones. Since only players on system places can refuse transitions, this offers precise control about *who* can restrict a shared transition. A player in a system place has full control of all outgoing transitions and is not subject to any environment behavior. In contrast, in a control games the actions are partitioned in controllable and uncontrollable and can, therefore, be restricted by either all or none of the involved players. The environment is modelled by defining *which* actions can be averted. While this compromises fine control about which process can restrict an action, a state in a control game can comprise both system (controllable) and environment (uncontrollable) behavior. Apart from having a distinct formalism to sort responsibilities between system and environment, strategies in both games are fundamentally different. In Petri games, a strategy is defined as a global branching process inheriting concurrent behavior of the game. In contrast, a control game strategy is defined in terms of local strategies acting on an explicit local view on the overall execution. The challenge of a translation is to resolve the controllability aspects of the game types, while preserving the causal information and thus allow a strategy transfer between the two game types.

For both translations, we adopt the concept of commitment sets: Instead of having the players control behavior directly, they move to a state or place that explicitly encodes their decision. Using this explicit representation, we can express the controllability aspects of one game in the respective other one, i.e., make actions in a control game controllable by only a subset of players and allow places in Petri games that comprise both environment and system behavior.

In addition to the upper bounds established by our exponential translations, we provide matching lower bounds under the assumption that weak bisimilar strategies are required. While our translations show that the contrasting formalisms can be overcome, our lower bounds show an intrinsic difference between both and moreover highlight the key limitations of them. The equivalence of both models, as witnessed by our results, gives rise to more practical applications by allowing the transfer of existing decidability results between both models. As an example, we outline the transfer of decidability for single-process systems in distributed environments, originally obtained for Petri games [11], to control games and decidability for acyclic communication architectures, originally obtained for control games [21, 15], to Petri games.

Parts of the results in this thesis have been published [2, 3].

**Structure** The remainder of this thesis is structured as follows: In Chapter 2 we define the necessary preliminaries and thoroughly introduce both Petri games and control games. We proceed by outlining how the two models considered in this thesis compare to the existing attempts on distributed synthesis, summarize previous work for both Petri game and control games and give a high-level comparison of both formalisms (Chapter 3). Afterwards, we define a notion of game equivalence (Chapter 4) as our main reference of what "structure-preserving" translations should comply with. In Chapter 5 and Chapter 6 we then give our translations in both directions, prove them correct and, furthermore, provide lower bounds. We conclude by outlining some of the newly obtained decidable classes (Chapter 7) and close this thesis by discussing our results as well as future work in Chapter 8.

# Chapter 2

# Preliminaries

In this chapter we formally introduce Petri games and control games. We conclude each section by presenting small example games, modelling interesting real-world situations. We begin by recalling multiset and partially ordered sets.

**Multisets** Sets are collections of objectives defined in terms of membership "$\in$". A multiset generalizes this by allowing to quantify the number of occurrences of an element. We view a multiset $A$ over a set $\mathcal{X}$ as a function $A : \mathcal{X} \to \mathbb{N}$. For any $x \in \mathcal{X}$, $A(x)$ defines the number of occurrences of $x$ in $A$. We write $x \in A$ for $A(x) > 0$ and define $|A| = \sum_{x \in \mathcal{X}} A(x)$. An ordinary set can be seen as a multiset with domain $\{0, 1\}$. Similar to sets we can build the union and difference of multiset. To distinguish them from their set-counterparts we denote them with $+$ and $-$. They are defined in the obvious way: $(A + B)(x) = A(x) + B(x)$ and $(A - B)(x) = \max(0, A(x) - B(x))$. We define $A \subseteq B$ if $A(x) \leq B(x)$ for all $x \in \mathcal{X}$.

**Partially Ordered Sets** Throughout this thesis we deal with asynchronous games and thereby with concurrent executions. While such executions can be described in terms of totally ordered sequences of events, doing so feels unnatural and compromises information about the causal relationship of events. We will see that, while both game types are fundamentally different, they both admit comparable concurrent executions. As a natural representation we use partially ordered sets. Recall that a relation $\leq$ over some set $\mathcal{X}$ is a *partial order* if it is reflexive, transitive and antisymmetric. A *partially order set* (poset) is a tuple $(\mathcal{X}, \leq)$ where $\mathcal{X}$ is some set and $\leq$ is a partial order on elements from $\mathcal{X}$.

## 2.1 Petri Games

As the first asynchronous game we introduce Petri games [12]. Since Petri games are played on Petri nets [26] we begin by introducing the latter.

### Petri Nets

**Definition** A *Petri net* (*net*) is a tuple $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$ where

- $\mathcal{P}$ (*places*) and $\mathcal{T}$ (*transitions*) are disjoint sets.
- The *flow* relation $\mathcal{F}$ is a multiset over $(\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$
- The *initial marking* $In$ is a multiset over $\mathcal{P}$ with $|In| < \infty$

We call elements from $\mathcal{P} \cup \mathcal{T}$ *nodes* and $\mathcal{N}$ *finite* if $\mathcal{P} \cup \mathcal{T}$ is a finite set. For a node $x \in \mathcal{P} \cup \mathcal{T}$, the multiset $pre(x)$ defined as $pre(x)(y) = \mathcal{F}(y, x)$ is called the precondition and $post(x)$ defined as $post(x)(y) = \mathcal{F}(x, y)$ the postcondition. The precondition (postcondition) is the multiset of

all nodes having an ingoing (outgoing) arc to $x$ according to $\mathcal{F}$. Through this thesis we often encounter situations where multiple nets $\mathcal{N}^\sigma, \mathcal{N}^1, \cdots$ are considered, we refer to the components by, e.g., $\mathcal{P}^{\mathcal{N}^\sigma}$ and write $pre^{\mathcal{N}^\sigma}(x)$ unless clear from the context. A marking in a Petri net is a multiset over places and represents the current configuration of the net. $In$ is the initial marking, i.e., the initial configuration. We call a transition $t$ *enabled* from a marking $M$ if every place in the precondition contains at least as many tokens as required by the flow relation, i.e, $pre(t) \subseteq M$. If a transition is enabled in $M$ it can *fire* and thereby produce a new marking $M'$. Firing consumes a token from every place in the precondition (possibly multiple tokens from one place) and puts new tokens on the postcondition of the transition: $M' = M - pre(t) + post(t)$. We denote the fact that $M'$ is the marking reached by firing $t$ from $M$ by $M \,[\, t \,\rangle\, M'$. We generalize this to sequences of transitions: With $M \,[\, t_0, t_1, \cdots, t_{n-1} \,\rangle\, M'$ we denote that there exist markings $M = M_0, M_1, \cdots, M_n = M'$ such that $M_i \,[\, t_i \,\rangle\, M_{i+1}$ for all $0 \leq i < n$. A marking $M$ is called *final* if there are no transitions enabled from it. We define the set of reachable markings $\mathcal{R}(\mathcal{N})$ as all marking reachable by a finite sequence of transitions:

$$\mathcal{R}(\mathcal{N}) = \{M \mid \exists n \in \mathbb{N}, t_0, \cdots, t_{n-1} \in \mathcal{T} : \ In^{\mathcal{N}} \,[\, t_0, \cdots, t_{n-1} \,\rangle\, M\}$$

We call a sequence $\kappa$ of transitions *valid* from a marking $M$ if $M \,[\, \kappa \,\rangle\, M'$ for some $M'$. Given a valid sequence $\kappa$ from $M$ we introduce the shortcut $M[\triangledown\, \kappa\,]$ as the marking reached by firing $\kappa$ starting in $M$. It always holds that $M \,[\, \kappa \,\rangle\, (M[\triangledown\, \kappa\,])$. Note that $M[\triangledown\, \kappa\,]$ is unique. It can be computed by

$$\begin{aligned} M[\triangledown\, \epsilon\,] &= M \\ M[\triangledown\, \kappa'\, t\,] &= M[\triangledown\, \kappa'\,] - pre(t) + post(t) \end{aligned}$$

For the initial marking we abbreviate $In^{\mathcal{N}}[\triangledown\, \kappa\,]$ with $\mathcal{N}[\triangledown\, \kappa\,]$.

A Petri net is called *concurrency-preserving* if $\forall t \in \mathcal{T} : |pre(t)| = |post(t)|$, i.e., the number of tokens in the course of a play does not change. It is *k-bounded* if $\forall M \in \mathcal{R}(\mathcal{N}) : q \in \mathcal{P}.\ M(q) \leq k$, i.e., in each reachable marking there are at most $k$ tokens on each place. It is said to be *bounded* if it is $k$-bounded for some $k$ and *safe* if it is 1-bounded.
A net $\mathcal{N}'$ is a *subnet* of $\mathcal{N}$ (written $\mathcal{N}' \sqsubseteq \mathcal{N}$) if $\mathcal{P}' \subseteq \mathcal{P}$, $\mathcal{T}' \subseteq \mathcal{T}$, $In' \subseteq In$ and $\mathcal{F}' = \mathcal{F} \restriction (\mathcal{P}' \times \mathcal{T}') \cup (\mathcal{T}' \times \mathcal{P}')$. That is $\mathcal{N}'$ is obtained from $\mathcal{N}$ by removing places and transition but while preserving the flow between the non-removed nodes.

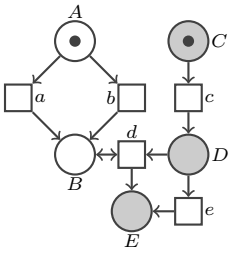



**Figure 2.1** Exemplary Petri net (Petri Game).

To represent a Petri net graphically we view the net as a bipatite, directed multigraph over $\mathcal{P} \cup \mathcal{T}$ and the flow relation as weighted arcs between nodes. Transitions are depicted as rectangle, whereas places are circles. A current marking (for instance the initial one) is illustrated by putting tokens/dots on the places of the marking. An example Petri net is depicted in Fig. 2.1. For now we ignore the fact that some places are coloured in gray. The net comprises places $A$ to $E$ and transitions, $a$ to $e$. The initial marking is $\{A, C\}$ and it, e.g., holds that $pre(d) = \{B, D\}$ and $post(D) = \{d, e\}$.

**Branching Processes** A Petri net progresses by firing transitions and thereby change the current marking. Even though we defined the set of reachable markings as the markings reached on a fixed sequence of transitions, Petri nets allow to model concurrent behavior that abstracts from concrete sequences of transitions. In Fig. 2.1 transitions $a$ and $c$ can be fired sequentially, however, as they are completely independent, the concrete order of execution does not matter. This motivates to look for characterisations of Petri net behavior that are better suited than sequences and captures the causal independence of transitions. It turns out that we can model this causal dependency in executions of a Petri net as a Petri net itself using so called occurrence nets. We begin by formally defining the causal dependency between nodes.

For nodes $x$ and $y$ we define $x \lessdot y$ iff $\mathcal{F}(x, y) > 0$, i.e., if there is at least one arc from $x$ to $y$. We denote the transitive and reflexive closure of $\lessdot$ by $\leq$ and the transitive closure by $<$. $x$ and $y$ are called *causally related* if $x \leq y$ or $y \leq x$. Two nodes $x$ and $y$ are in *conflict*, denoted $x \sharp y$, if there is a place $q$ different from $x$ and $y$ and two distinct transitions $t_1, t_2 \in post(q)$ s.t. $t_1 \leq x$ and $t_2 \leq y$. That is, $x$ and $y$ are reached leaving $q$ on two different transitions. They are hence exclusive in the sense that they result from two different nondeterministic choices. A node $x$ is in *self-conflict* if $x \sharp x$. If nodes $x$ and $y$ are neither casually related nor in conflict we call them *concurrent*.

**Definition** An *occurrence net* is a Petri net $\mathcal{N}$ where:

**(1)** $\forall t \in \mathcal{T} : pre(t)$ is a set **(4)** $In = \{q \in \mathcal{P} \mid pre(q) = \emptyset\}$

**(2)** $\forall q \in \mathcal{P} : |pre(q)| \leq 1$ **(5)** No transition is in self-conflict

**(3)** $\leq$ is well founded, i.e., there is no infinite descending sequence of nodes.

In occurrence nets transitions consume at most one token from each place **(1)**, each place is reached by an unique transition **(2)**, following the inverse of the flow always terminates **(3)** and the initial places of $\mathcal{N}$ coincide with the places with no ingoing transitions **(4)**. Furthermore, the absence of self-conflicts **(5)** guarantees that each choice of transitions from one place "separates" the remaining net: If two transitions are leaving a place $q$ then the successor nodes of both transitions are disjoint. As occurrence nets are, by definition, acyclic one might think of them as tree-like nets where each transition moves the current marking further down in the tree.

For a set of places $C$ the *causal past* of $C$ is the set

$$past(C) = \{y \in \mathcal{P} \cup \mathcal{T} \mid \exists x \in C, y \leq x\}$$

As we are mainly interested the events, i.e., transitions, we, furthermore, define

$$past_{\mathcal{T}}(C) = \{y \in \mathcal{T} \mid \exists x \in C, y \leq x\}$$

We abbreviate $past(x) = past(\{x\})$ and $past_{\mathcal{T}}(x) = past_{\mathcal{T}}(\{x\})$. The causal past of a node or set of nodes hence comprises all nodes that have been visit in the past. Since in occurrence nets the precondition of every place is unique **(2)**, every place therefore comprises precise information about its journey to get there. The causal past of a place (or a set of places), together with the causal relationship $\leq$ forms a partially ordered set $((past_{\mathcal{T}}(q), \leq))$ that naturally represents a concurrent execution in an occurrence net. Unordered transitions correspond to transition that can be executed concurrently.

We can define a homomorphism between two Petri nets as a mapping between nodes that preserves the structure on transitions.

**Definition** For two nets $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$ and $\mathcal{N}' = (\mathcal{P}', \mathcal{T}', \mathcal{F}', In')$ a function $\lambda : \mathcal{P} \cup \mathcal{T} \to \mathcal{P}' \cup \mathcal{T}'$ is called a *initial homomorphisms* from $\mathcal{N}$ to $\mathcal{N}'$ if

**(1)** $\lambda[\,\mathcal{P}\,] \subseteq \mathcal{P}'$ and $\lambda[\,\mathcal{T}\,] \subseteq \mathcal{T}'$

**(2)** For every $t \in \mathcal{T}$: $\lambda[\,pre^{\mathcal{N}}(t)\,] = pre^{\mathcal{N}'}(\,\lambda(t)\,)$ and $\lambda[\,post^{\mathcal{N}}(t)\,] = post^{\mathcal{N}'}(\,\lambda(t)\,)$

**(3)** $\lambda[\,In\,] = In'$

One can think of a homomorphism $\lambda$ as a labelling of $\mathcal{N}$ by nodes of the same type from $\mathcal{N}'$. For a homomorphism from $\mathcal{N}$ to $\mathcal{N}'$, $\mathcal{N}$ can be considered as a conceptually bigger net that can duplicate nodes. The labelling requires $\mathcal{N}$ to be structure-preserving on transitions: For each transition $t$ the pre- and postcondition must be equally labelled as the pre- and postcondition of $\lambda(t)$. Note that we only require transitions to preserve the structure so, in particular, places can copy outgoing transitions.

As occurrence nets are a natural concept to describe the causal relation between concurrent events, we use them to describe (parts of) the behavior of a Petri net. Using an initial homomorphism we label events (transitions) and places in the occurrence net with nodes in the original net. This is done formally by an initial branching process [5].

**Definition** An *initial branching process* for a net $\mathcal{N}$ is a pair $\jmath = (\mathcal{N}^\jmath, \lambda)$ where $\mathcal{N}^\jmath$ is an occurrence net and $\lambda$ is an initial homomorphism from $\mathcal{N}^\jmath$ to $\mathcal{N}$ with the additional requirement that

$$\forall t_1, t_2 \in \mathcal{T}^{\mathcal{N}^\jmath} : \ \lambda(t_1) = \lambda(t_2) \wedge pre(t_1) = pre(t_2) \Rightarrow t_1 = t_2$$

While the initial homomorphism requires transitions in the occurrence net to preserve the structure of the original transitions, an occurrence net necessitates each nondeterministic choice of transitions to result in distinct places. If a node can be reached on two different paths it is therefore split up in the branching process. The resulting copies of nodes are labelled by $\lambda$ with the corresponding original node in $\mathcal{N}$. The additional injectivity requirement avoids copies to be added multiple times from the same precondition: Each transition must either be labelled differently or occur from different preconditions. As an occurrence net preserves the structure of pre- and postcondition from the original net, every valid sequence of transitions in $\mathcal{N}^\jmath$ is (by $\lambda$) mapped to a valid sequence of transitions in $\mathcal{N}$. On the other hand, a branching can restrict behavior by not adding certain transitions. Thus not every valid sequence of transitions in $\mathcal{N}$ must be matched in $\mathcal{N}^\jmath$.

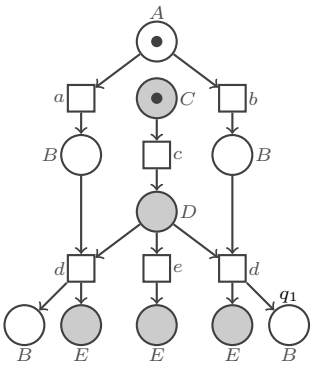



**Figure 2.2** Unfolding of the Petri net from Fig. 2.1. The gray label is the one given by $\lambda$. The name of nodes are omitted with the exception of $q_1$.

An exemplary branching process of the example net in Fig. 2.1 is depicted in Fig. 2.2 (when the colouring of places in gray is ignored). The label given by the homomorphism $\lambda$ is indicated in gray. Note that all transitions in the branching process preserve the structure of the transition in the underlying net. Place $B$ in Fig. 2.1 can be reached on two distinct paths, i.e., via $a$ or via $b$. In the branching process it is split into two copies, both labelled with $B$. In the branching process the causal past of any place, if paired with $\leq$, describes the concurrent execution that lead to a token on that place. As an example, the causal past of place $q_1$ comprises transitions $c, b$ and $d$, where $c$ and $b$ are unordered.

In this thesis we only deal with initial branching processes and thus refer to them simply as branching processes. Note that branching processes, even for finite nets, can be infinite.

**Unfolding** While a branching process can describe restrictions of the behavior of a Petri net, there is an unique maximal branching process, called the *unfolding* [6], representing every behavior of a net:

**Definition** The unfolding of a net $\mathcal{N}$ is the (unique) branching process $\mathfrak{U} = (\mathcal{N}^\mathfrak{U}, \lambda)$ satisfying: For every set of pairwise concurrent places $C$ in $\mathcal{N}^\mathfrak{U}$ with $\lambda[C] = pre^\mathcal{N}(t)$ for some $t \in \mathcal{T}^\mathcal{N}$ there exists a $t'$ with $\lambda(t') = t$ and $C = pre^\mathfrak{U}(t')$

In the unfolding from every set $C$ of reachable places all transitions possible from $\lambda[C]$ in the underlying net are added as some copy. Every sequence of transitions in the original net hence corresponds to at least one equally labelled sequence in the unfolding[1]. Fig. 2.2 depicts the unfolding of the net in Fig. 2.1.
Conceptually we can view any branching process $\jmath$ as a subprocess of the unfolding, i.e, $\mathcal{N}^\jmath \sqsubseteq \mathcal{N}^\mathfrak{U}$

[1] For safe nets one can show that there is exactly one.

and $\lambda^{\jmath} = \lambda^{\mathfrak{U}} \restriction (\mathcal{P}^{\jmath} \cup \mathcal{T}^{\jmath})$[2]. This goes well with the intuition that, while the unfolding characterizes every behavior of a net, a branching process describes a restriction of it.

## Petri Games

A *Petri game*, first introduced in [12], is a distributed game played between the system and the environment on a Petri net. Formally a game $\mathcal{G}$ is a tuple $\mathcal{G} = (\mathcal{P}_{\mathcal{S}}, \mathcal{P}_{\mathcal{E}}, \mathcal{T}, \mathcal{F}, In, Sp)$ where $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$ with $\mathcal{P} = \mathcal{P}_{\mathcal{S}} \uplus \mathcal{P}_{\mathcal{E}}$ is the underlying, finite Petri net. We extend notation from the underlying net to $\mathcal{N}$ by, e.g., defining $pre^{\mathcal{G}}(\cdot) = pre^{\mathcal{N}}(\cdot)$ and $\mathcal{P}^{\mathcal{G}} = \mathcal{P}^{\mathcal{N}}$. We can view a game as the underlying net where the places ($\mathcal{P}$) are partitioned into system places ($\mathcal{P}_{\mathcal{S}}$) and environment places ($\mathcal{P}_{\mathcal{E}}$). If a token resides on a system place it belongs to the system and is regarded as a system player. A token on an environment place, on the other hand, is controlled by the environment. The tokens can move through the game by firing transitions as defined by the underlying net. The Petri nets describes all possible behavior of the (asynchronous) system. A *strategy* can intuitively control the behavior of tokens on system places, i.e., restrict which transition to take and which to forbid. Opposed to that, a token on an environment place is controlled by the environment. A strategy cannot thwart the behavior of such tokens and has to account for every possible move. The last component of a Petri game, $Sp$, marks a set of special places ($Sp \subseteq \mathcal{P}$) used to define a winning objective for the system. In this thesis we consider either *reachability* or *safety* objectives. In the former $Sp$ comprises a set of winning places, whereas in the latter it defines losing ones. The strategy should restrict the overall behavior such that either a winning configuration is reached or a losing one is avoided.
We graphically represent a Petri game as the underlying net and depict system places in gray and environment places in white. The special places are marked by a double circle. When we take the colouring of nodes into account, Fig. 2.1 depicts a possible Petri game. The game does not comprise any special places.

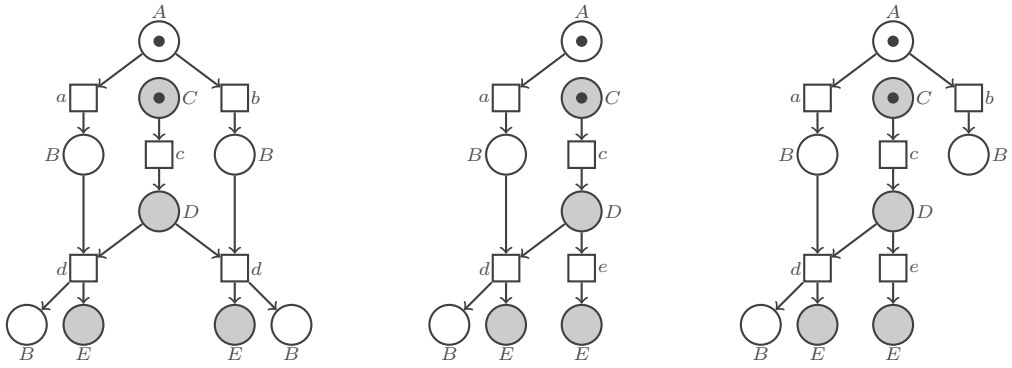
Before we precisely set out how a game can be won by either the system or environment we define the possibilities a system player has in a game. As we argued informally a strategy can control the behavior of system tokens but has to account for every environment move. We define a strategy as a branching process of the underlying net, that fulfils additional restrictions corresponding to the structure of the game.

**Definition** A *strategy* for a Game $\mathcal{G}$ is a branching process $\sigma = (\mathcal{N}^{\sigma}, \lambda)$ of the underlying net that satisfies the following condition:

*Justified Refusal:* If $C$ is a set of pairwise concurrent places in $\mathcal{N}^{\sigma}$ and $t \in \mathcal{T}^{\mathcal{G}}$ a transition with $\lambda[C] = pre^{\mathcal{G}}(t)$ then there either exists a transition $t'$ with $\lambda(t') = t$ and $C = pre^{\mathcal{N}^{\sigma}}(t)$ or there exists a system place $q \in C \cap \lambda^{-1}[\mathcal{P}_{\mathcal{S}}]$ with $t \notin \lambda[post^{\mathcal{N}^{\sigma}}(q)]$

The unfolding of a Petri net represents every possible behavior of a net, whereas a branching process is obtained by removing parts of the unfolding. A strategy, defined by a branching process, is therefore a restriction of the possible behavior in a game[3]. An ordinary branching process alone could restrict the behavior of a game in any way imaginable. A strategy should, however, respect the intended meaning of the partition of the places, i.e., only restrict behavior from system places. Justified refusal requires every system place in the strategy to decide what transitions to allow based on its causal past alone. To see this, it is easiest to compare the definition of justified refusal with that of the unfolding: Suppose there is a set of pairwise concurrent places $C$ in a branching process and a transition $t$ possible from the label of the places in $C$, i.e., $\lambda[C] = pre^{\mathcal{G}}(t)$. The unfolding requires a copy of $t$ to be added from $C$. Justified refusal, on the other hand, either requires a copy $t$ to be allowed, i.e., added to the strategy, but

[2] One can prove that every branching process is isomorphic to a subprocess of the unfolding. Where two branching processes $\jmath$ and $\jmath'$ are isomorphic if there is a bijective homomorphism $\lambda$ between $\mathcal{N}^{\jmath}$ and $\mathcal{N}^{\jmath'}$ and $\lambda^{\jmath} = \lambda^{\jmath'} \circ \lambda$.
[3] Our definition of a strategy differs from the one used in existing Petri game literature [12, 11] in the sense that our definition of a strategy is independent of the safety or determinism condition that we discuss later.




(a) (b) (c)

**Figure 2.3** Branching processes of the Petri game from Fig. 2.1. The gray label is given by $\lambda$ and places are distributed as system and environment. The explicit name of nodes are omitted. The branching process in (a) is also a strategy for Fig. 2.1. Both (b) and (c) violate justified refusal.

allows for situations in which $t$ can be restricted. This is, however, only admitted if there is a system place $q$ in $C$ that *never* allows a copy of $t$ ($t \notin \lambda[post^{\mathcal{N}^\sigma}(q)]$). Any transition $t$ not added to the strategy must hence be justified by at least on system place $q$ that uniformly forbids it, i.e., forbids it independently of the concrete situation of the game. If there is a different set of pairwise concurrent places $C'$ where the precondition of $t$ coincides with $\lambda[C']$ and $q \in C'$ then there cannot be a copy of $t$ added from $C'$. Every system place must thus allow either all copies of a transition or none. This implicitly forces places to base their decision on their causal past only and not restrict transitions as a reaction to the current situation of the game. If we see a shared transition as communication, a system place needs to decide whether or not to allow the particular communication without yet knowing in which situation the communication partner(s) is. Note that, in particular, a transition that only involves environment places cannot be restricted by the strategy.

As an example, we come back to our example from Fig. 2.1. Every branching process for this game can be seen as a subprocess of the unfolding in Fig. 2.2. In Fig. 2.3 three branching processes of the Petri game are depicted. While the one in (a) is also a strategy both branching processes in (b) and (c) violate justified refusal. In (b) transition $b$ is not added to the strategy even though there is no system place involved in it. For the branching processes in (c) transition $d$ is not added provided that the environment used transitions $b$. While there is a system place in the precondition of $d$ this place does not forbid the transition uniformly, i.e., it allows the transition from the other copy of $B$.

A strategy allows the system to restrict the behavior in the Petri game in a way that confirms with our intended meaning of system and environment places. To obtain interesting games the strategy should restrict the behavior such that a given objective is fulfilled. Throughout this thesis we deal with two kinds of system-objectives:

**Reachability Petri Games** In reachability games the special places describe a set of winning places $\mathcal{W} \subseteq \mathcal{P}$. The objective of the strategy is to manoeuvrer into a situation (reach a marking) where every token is on a winning place:

> **Definition** A strategy $\sigma$ is *(reachability) winning* if $\mathcal{N}^\sigma$ is finite and for every reachable, final marking $M$ in $\mathcal{N}^\sigma$, $\lambda[M] \subseteq \mathcal{W}$

**Safety Petri Games** In safety games the special places describe a set of bad places $\mathcal{B} \subseteq \mathcal{P}$. A winning strategy should never allow for a situation where a token resides on a bad place. The best chance of winning such a game is for all player on system places to refuse all possible

transitions. To avoid such trivial strategies we require deadlock avoidance: A strategy cannot refuse transitions if this would deadlock the system even though the underlying net could still execute a transition.

**Definition** A strategy $\sigma$ is *deadlock-avoiding* if for every *final* marking $M \in \mathcal{R}(\mathcal{N}^\sigma)$, $\lambda[M]$ is final.

Deadlock avoidance states that a strategy is only allowed to terminate if the underlying Petri net does. Note that deadlock avoidance describes a global property: Individual tokens might refuse all transitions if there is at least one token that can continue playing.

**Definition** A strategy $\sigma$ is *(safety) winning* if it is deadlock-avoiding and for every $M \in \mathcal{R}(\mathcal{N}^\sigma)$, $\lambda[M] \cap \mathcal{B} = \emptyset$

We remark at this point that all existing results for Petri games [12, 11] are stated in terms of safety objectives.[4]

**Determinism** Having established a winning objective and a notion of what the system is able to control we can already ask the question whether a system has a winning strategy. In addition to behave as intended (i.e., fulfil justified refusal) we can ask a strategy to be deterministic.

**Definition** A strategy $\sigma$ is *deterministic*, if for every $M \in \mathcal{R}(\mathcal{N}^\sigma)$ and every system place $q \in M \cap \lambda^{-1}[\mathcal{P}_\mathcal{S}]$ there is at most one transition $t \in post^{\mathcal{N}^\sigma}(q)$ enabled in $M$.

While a place might allow multiple outgoing transitions, i.e., $post^{\mathcal{N}^\sigma}(q)$ contains multiple transitions, determinism requires that in any situation at most one of them can be taken. An individual token on a system place hence never has to make a decision of what transition to take: Once the strategy is fixed the execution progresses deterministically from the view of the system. Requiring deterministic strategies is purely optional.

## Example

As an example, we consider the (reachability) Petri game depicted in Fig. 2.4. It describes possible police-behavior in a mafia investigation. The game consists of four player: A mafia boss at place $M$, a burglar initially in the $B$ place and an undercover-cop that behaves like a burglar, initially at $U$. Furthermore, there is a single police officer starting in place $P$. The undercover cop and the police are both modelled as a system player, i.e., their places are system place, whereas the mafia boss and the loyal burglar belong to the environment. The game progresses as follows: The mafia boss can choose whether she wants to burgle house $X$ or $Y$ by moving to place $D_x$ or $D_y$. Afterwards, she initiates the burglary by either synchronizing with the burglar ($i_x$) or the undercover cop ($i_y$). The instructed player commits the crime, i.e., moves to place $H_x$ or $H_y$. The mafia boss progresses to place $K$. The police officer in place $P$ is, as of right now, not involved in any part of the game. Upon arrival of the mafia boss at place $K$ she can, however, interrogate the boss using transition $i$, resulting in the police officer knowing the location of the burglary. Using transitions $s_x$ or $s_y$ she can afterwards send the mafia boss to location $L_x$ or $L_y$. To ensure all players to terminate on winning places the police is required to send the mafia boss to the correct location of burglary as only then transitions leading to winning places $(w_x, w_y)$ are enabled.

This game can be won by the system: The undercover cop does not blow her cover and commits the burglary even though she could forbid it. The police officer interrogates the mafia boss and uses the obtained information to send the boss to the correct location of the burglary. The existence of such a winning strategy depends on the fact that information is exchanged

[4] Our definition follows the one given in [12]: We define bad situations in terms of *local* bad places where every marking containing such a state should be avoided. [11] allowed a more expressive characterisation by defining bad situation in terms of a set of marking. This allows to characterise *global* situations that the system should avoid. The translations presented in this thesis can be adopted to both formalisms.

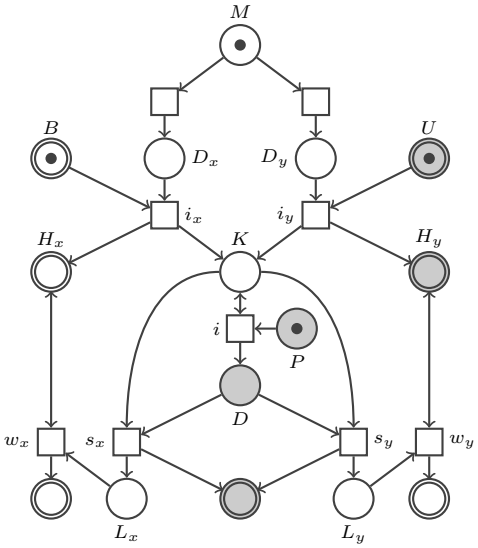



**Figure 2.4** A (reachability) Petri game describing a mafia-police investigation that involves a mafia boss, a burglar, an undercover cop and police officer. The mafia boss can decide the location of the burglary, the police can interrogate her and send her to the location of burglary. To win the police uses the information obtained in the interrogation and afterwards send her to the correct location.

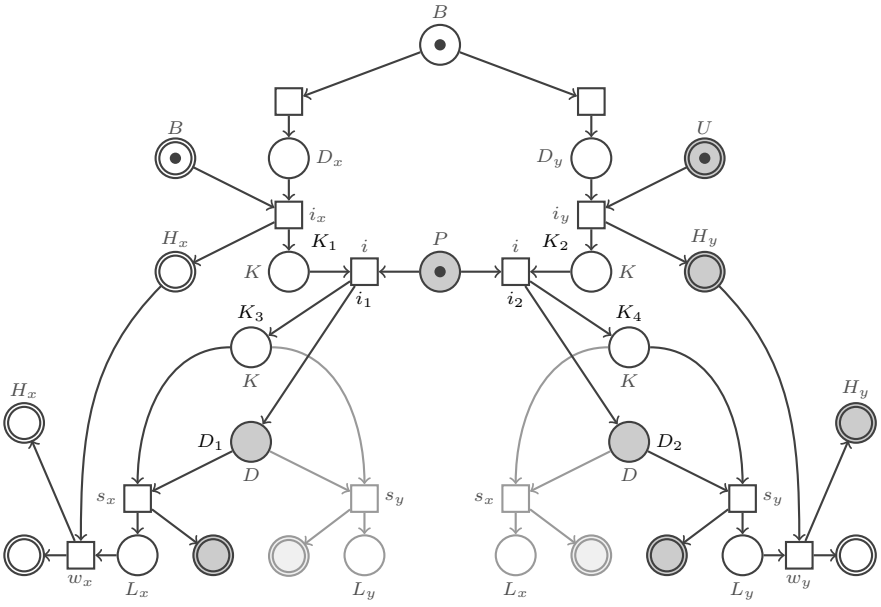



**Figure 2.5** The unfolding of the Petri game in Fig. 2.4 if all gray parts are included. The gray label is the $\lambda$-label to the original game. Some of the nodes nodes are named explicitly in black. If the grayed out parts are removed, a winning (deterministic) strategy for Fig. 2.4 is obtained.

upon every communication. The interrogation transition $i$ transfers the entire causal past of the mafia boss, in particular, the location of the burglary. Only given this information the police can act appropriately.

In Fig. 2.5 the unfolding of the Petri game from Fig. 2.4 is depicted. Note that every place or

transition that can be reached on different paths is split up into copies of that place/transition. Recall that causal past describes the information local to each player: As long as the players progress independently they share no information but exchange their entire history upon communication. We observe that every place in the unfolding *encodes* the causal past of a token on it: Consider the place $K$ in the original game. It can be reached on two different paths of transitions, depending on whether the burglary occurred in house $X$ or $Y$. In the unfolding the place $K$ is therefore split into two copies, $K_1$ and $K_2$. A token on a place can hence deduce its entire causal past and comprehended how it got their: If a token is, e.g., on place $K_1$ the burglary happened in house $X$ and vice versa. Taking the interrogation transition $i$ results in two additional copies of $K$, $K_3$ and $K_4$. This goes well with the idea of places encoding the causal information of a token on it: For instance, place $K_3$ carries the information that the burglary occurred in house $X$ *and* the police already interrogated her once. Places $K_1$ to $K_4$ therefore encode the causal information of the mafia boss if on the respective place.

As independent parts in the Petri game are added as disjoint parts in the unfolding the police office in place $P$ is, at the point of burglary, not involved in transition. His place does not encode any information about the burglary, in particular, no information whether the mafia boss resides on $K_1$ or $K_2$. We can, however, observe that taking transition $i$ transmits all causal information previously local to the mafia boss to the police. As we saw the causal information of the mafia is encoded in places $K_1$ and $K_2$. Since the unfolding adds transitions from all possible combination of copies, transition $i$ is duplicated into $i_1$ and $i_2$ and so are the places in the postcondition. In particular, the duplication of $K$ results in a duplication of $D$ ($D_1$ and $D_2$). While the initial place of the police does not encode any information, utilizing $i$ leads to copies of the resulting place encoding the newly obtained knowledge as the new causal information of a token on that place. If a token resides on $D_1$, it knows that the burglary occurred at $X$, if on $D_2$ the mafia boss instructed to burgle house $Y$.

A strategy is a subprocess of the unfolding and can therefore forbid certain behavior by removing transitions (and all subsequent nodes) from the unfolding. Since places in the unfolding carry precise information about the causal past, removing parts of the unfolding results in strategies that depend on causal information. A winning strategy for our example game is obtained by removing the gray parts in the unfolding. This corresponds to our high level idea of a winning strategy: If a token is on $D_1$ the burglary must have occurred in $X$ so the police officer sends the mafia boss there using $s_x$, and vice versa for $D_2$.

## 2.2 Control Games

As the second asynchronous game we introduce control games on asynchronous automata. The control problem in general has been first studied in [28] by Ramadge and Wonham and has been extended to the asynchronous setting [20, 14]. Similar to Petri Games being played on Petri nets, control games are played on asynchronous automata. We begin by introducing (Zielonka's) asynchronous automata [29] and fundamental trace theory [4].

### Asynchronous Automata

Asynchronous automata (Zielonka's automata) can be thought of as a family of local automata, called processes, synchronizing on shared actions. With $\boldsymbol{P}$ we denote a finite set of processes. A distributed alphabet is a pair $(\Sigma, dom)$ where $\Sigma$ is a finite set of actions and $dom : \Sigma \to 2^{\boldsymbol{P}} \setminus \{\emptyset\}$ assigns to each action in $\Sigma$ a non-empty set of processes that share this action. For an action $a$, $dom(a)$ comprises all processes that synchronize on $a$. For a process $p$, $\Sigma_p$ denotes the set of actions that $p$ is directly involved in, i.e., $\Sigma_p = \{a \in \Sigma \mid p \in dom(a)\}$.

**Definition** An *asynchronous automaton* is a tuple $\mathcal{A} = \langle \{S_p\}_{p\in \boldsymbol{P}}, s_{in}, \{\delta_a\}_{a\in\Sigma}\rangle$ where

- $S_p$ is a finite set of (local) states of process $p$
- $s_{in} \in \prod_{p\in \boldsymbol{P}} S_p$ is the initial (global) state
- $\delta_a : \prod_{p\in dom(a)} S_p \stackrel{\cdot}{\to} \prod_{p\in dom(a)} S_p$ is a partial transition function

We call an element $\{s_p\}_{p\in \boldsymbol{P}} \in \prod_{p\in \boldsymbol{P}} S_p$ a global state. Formally this is a dependent function mapping a process $p$ to a local state from $S_p$. For a set of processes $R \subseteq \boldsymbol{P}$, $\{s_p\}_{p\in R} \in \prod_{p\in R} S_p$ denotes the state of the processes in $R$, i.e., the restriction of the dependent function to $R$. We abbreviate $\{s_p\}_{p\in R}$ with $s_R$. We denote that a local state $s' \in S_p$ is part of a global state $s$ by $s' \in s$.
An asynchronous automata can be seen as a global automaton with state space $\prod_{p\in \boldsymbol{P}} S_p$, initial state $s_{in}$ and a transition from $s \stackrel{a}{\mapsto} s'$ if $\delta_a(s_{dom(a)}) = s'_{dom(a)}$ and $s_{\boldsymbol{P}\setminus dom(a)} = s'_{\boldsymbol{P}\setminus dom(a)}$. The processes start in their local initial states and progresses by firing actions and thereby change their current state. When playing an action $a$ the states of all processes in $dom(a)$, i.e., all processes synchronizing on $a$, is updated simultaneously according to $\delta_a$. By $Plays(\mathcal{A}) \subseteq \Sigma^* \cup \Sigma^\omega$ we refer to the set of finite and infinite sequences[5] in this global automaton. With $act(s')$ for a local state $s' \in \bigcup_{p\in \boldsymbol{P}} S_p$ we denote every action that can occur involving $s'$. Formally $act(s') = \{a \in \Sigma \mid \exists B \in domain(\delta_a),\ s' \in B\}$. For a finite $u \in Plays(\mathcal{A})$, $state(u)$ denotes the global state reached on $u$ and $state_p(u)$ the local state of process $p$.

**Composition of local automata** In our translation (Chapter 5), an alternative characterization of a subset of asynchronous automata turns out to be practical: We describe every process $p$ by a local automaton $\circlearrowleft_p$, acting on actions from $\Sigma_p$. A local (deterministic) automata is tuple $\circlearrowleft_p = (Q_p, \vartheta_p, s_{0,p})$ where $Q_p$ is a finite set of states, $s_{0,p} \in Q_p$ the initial state and $\vartheta_p \subseteq Q_p \times \Sigma_p \times Q_p$ a deterministic transition relation (labelled by $\Sigma_p$).

**Definition** For a family of processes modelled as local automata $\{\circlearrowleft_p\}_{p\in \boldsymbol{P}}$ we define the *parallel composition* as the asynchronous automaton $\otimes_{p\in \boldsymbol{P}} \circlearrowleft_p = \langle \{S_p\}_{p\in \boldsymbol{P}}, s_{in}, \{\delta_a\}_{a\in\Sigma}\rangle$ with:

**(1)** For every $p \in \boldsymbol{P}$, $S_p = Q_p$ **(2)** $s_{in} = \{s_{0,p}\}_{p\in \boldsymbol{P}}$

**(3)** $\delta_a(\{s_p\}_{p\in dom(a)})$: If for all $p \in dom(a)$ there exists a $s'_p$ with $(s_p, a, s'_p) \in \vartheta_p$ then define $\delta_a(\{s_p\}_{p\in dom(a)}) = \{s'_p\}_{p\in dom(a)}$ otherwise it is undefined.

In $\otimes_{p\in \boldsymbol{P}} \circlearrowleft_p$ each automaton acts locally but they synchronize on shared actions. For an action to be possible all processes must be in states where the action can occur from **(3)**. Viewing an asynchronous automaton as the parallel composition of local automata is most intuitive and well suited for graphical representation. It should, however, be noted that not every asynchronous automaton can be seen as the parallel composition of local automata: The transition functions $\{\delta_a\}_{a\in\Sigma}$ in an asynchronous automaton define transitions between global states of the processes in $dom(a)$. Such global transitions might not be representable in terms of local automata.

In Fig. 2.6 a possible asynchronous automaton is depicted. The three processes are all represented as local automaton that share actions. A possible play in this automaton is $a, d, c, c, b, e, c$.

## Traces

So far we have described plays in an asynchronous automaton as totally ordered sequences. Since transition in asynchronous automata only involve subsets of the processes we notice that, as for Petri nets, (totally ordered) sequences are not well suited to capture the concurrent nature

[5] The attentive reader might notice that a linear play is not well suited to describe behavior of a system allowing concurrent behavior. We fix this later.

admitted by an asynchronous automaton. We try to fix this now and recover the concurrency lost when considering sequential executions.

The distributed alphabet induces a natural dependency relation on $\Sigma$. Actions $a$ and $b$ are said to be independent (written $a \mathbb{I} b$) if $dom(a) \cap dom(b) = \emptyset$, that is they involve disjoint sets of processes. Since an asynchronous automata $\mathcal{A}$ is defined on the distributed alphabet it is easy to see that if $a \mathbb{I} b$ and $w_1\, a\, b\, w_2 \in \mathit{Plays}(\mathcal{A})$ for some $w_1$ and $w_2$ then $w_1\, b\, a\, w_2 \in \mathit{Plays}(\mathcal{A})$; Independent action can be swapped. Note that swapping independent actions does not change the global state reached on a sequence. We use this insight to define an equivalence relation $\sim_{\mathbb{I}}$ on $\Sigma$-sequences by defining $u \sim_{\mathbb{I}} w$ if $u$ and $w$ are identical up to multiple swaps of consecutive independent actions. That is $\sim_{\mathbb{I}}$ is the finest relation closed under $w_1\, a\, b\, w_2 \sim_{\mathbb{I}} w_1\, b\, a\, w_2$ for independent $a$ and $b$. The equivalence class of a sequence $u \in \Sigma^*$, denoted $[u]_{\mathbb{I}}$, is the smallest set that contains $u$ and is closed under $\sim_{\mathbb{I}}$. The equivalence class $[u]_{\mathbb{I}}$ is called a *trace*, its elements are called *linearisations* of $[u]_{\mathbb{I}}$. Many operations can be extended to traces by defining them in terms of linearisation of a trace: With $|\cdot|$ we denote the length of a trace and for traces $[u]_{\mathbb{I}}$ and $[w]_{\mathbb{I}}$, $[u]_{\mathbb{I}}\,[w]_{\mathbb{I}}$ denotes their concatenation. Note that for sequences with $u \sim_{\mathbb{I}} w$ it holds that $u \in \mathit{Plays}(\mathcal{A}) \Leftrightarrow w \in \mathit{Plays}(\mathcal{A})$, we hence say that the set $\mathit{Plays}(\mathcal{A})$ is *trace closed*. It is therefore natural to abstract from sequences of actions in $\mathcal{A}$ and instead view the plays in $\mathcal{A}$ as traces. A trace is a natural representation of the concurrent execution of an automaton, by abstracting from sequential executions. From now on, unless explicitly stated, we work with traces instead of sequences. In particular, $\mathit{Plays}(\mathcal{A})$ denotes the set of traces. If clear from the context we also write $u$ instead of $[u]_{\mathbb{I}}$. Note that $\mathit{state}(\cdot)$ ($\mathit{state}_p(\cdot)$) is invariant under $\sim_{\mathbb{I}}$ and can hence be extended to traces.

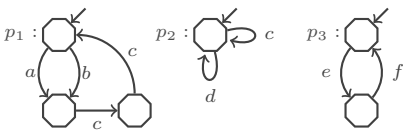



**Figure 2.6** Example asynchronous automaton depicted as the composition of local automata. It comprises three process $p_1, p_2, p_3$. The domain of all actions includes all processes whose local description contains that action.

**Traces as Partially Ordered Sets** While traces are a natural concept to describe concurrent executions, they are tedious to work with since they are defined in terms of a representation of the respective equivalence class. For Petri nets we already saw that partially ordered sets are well suited to capture the causal independence of events in an execution. In similar fashion, we can now consider a trace as a poset where the concurrency of actions in the trace is made explicit in the partial order. This gives us an unique representation of trace and is well suited for both graphically representation as well as comparison to concurrent executions in Petri nets.

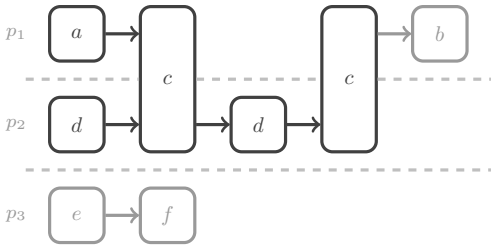



**Figure 2.7** Poset representation of play $u_{example} = [a, d, e, c, d, f, c, b]_{\mathbb{I}}$ in the asynchronous automaton in Fig. 2.6. Every element in the poset formally is a primed prefix and is labelled by its last action.

For a totally ordered sequence $u$ we denote the last action with $\mathit{last}(u)$. A trace $[u]_{\mathbb{I}}$ is called *prime* if the last action agrees on all its linearisations. A trace $[w]_{\mathbb{I}}$ is the prefix of $[u]_{\mathbb{I}}$, denoted $[w]_{\mathbb{I}} \sqsubseteq [u]_{\mathbb{I}}$, if $w' \sqsubseteq_{seq} u'$ for some $w' \in [w]_{\mathbb{I}}$ and $u' \in [u]_{\mathbb{I}}$ where $\sqsubseteq_{seq}$ denotes the prefix relation on sequences[6]. To each trace $[u]_{\mathbb{I}}$ we associate a partially ordered set $(\mathcal{X}, \leq)$ where

$$\mathcal{X} = \{[w]_{\mathbb{I}} \mid [w]_{\mathbb{I}} \sqsubseteq [u]_{\mathbb{I}} \ \wedge\ [w]_{\mathbb{I}} \text{ is prime}\}$$

[6]Prefixes on traces behave fundamentally different then prefixes of sequences. For example, an intuitive result for sequences is that two prefixes of the same sequence are comparable w.r.t. $\sqsubseteq_{seq}$. This does not hold for prefixes of traces.

and

$$[w_1]_{\mathbb{I}} \leq [w_2]_{\mathbb{I}} \text{ if } [w_1]_{\mathbb{I}} \sqsubseteq [w_2]_{\mathbb{I}}$$

When depicting the poset we are mainly interested in the actions associated with the element in the poset. We hence label each element from $\mathcal{X}$ with the last element of the trace. Since all elements in $\mathcal{X}$ are prime this is well defined. Computing the poset representation for a trace is a injective operation. A poset is hence a true alternative representation of a trace.

The poset representation of the example trace $u_{example}$ with

$$u_{example} = [\ a\ ,\ d\ ,\ e\ ,\ c\ ,\ d\ ,\ f\ ,\ c\ ,\ b\ ]_{\mathbb{I}}$$

in the automaton from Fig. 2.6 is depicted in Fig. 2.7. To aid readability the actions are grouped in rows of the processes involved in them. The concurrency of actions $a$ and $d$ is, e.g., made explicit as they are unordered in the poset representation. In particular, we can note that every total order of the partially ordered set forms a linearisation of the trace and, conversely, every linearisation corresponds to a total order compatible with the partial one.

## Control Games

Similar to Petri nets being the playing arena of Petri Games, asynchronous automata define the arena of a control game [20, 14]. The game is played between the system and the environment. Formally a control game is a tuple $\mathcal{C} = \langle \mathcal{A}, \Sigma^{sys}, \Sigma^{env}, \{\mathbb{S}_p\}_{p \in \boldsymbol{P}} \rangle$. $\mathcal{A}$ delineates an asynchronous automaton, called the plant, as the underlying arena of the game. $\Sigma = \Sigma^{sys} \uplus \Sigma^{env}$ is a partition of the set of action $\Sigma$ into controllable ($\Sigma^{sys}$) and uncontrollable ($\Sigma^{env}$). We define the set of possible plays as $\mathit{Plays}(\mathcal{C}) = \mathit{Plays}(\mathcal{A})$. While the asynchronous automaton describes all plays admitted by the underlying (asynchronous) model, a strategy can restrict this behavior by prohibiting actions from occurring. Unlike Petri games where we distinguish system and environment places and therefore charactered *who* can restrict behavior, in control games the actions are distributed to indicate *what* can be restricted. Intuitively a strategy can refuse controllable actions, whereas uncontrollable ones are conceptually controlled by the environment and can therefore not be restricted by a strategy. The last constitute $\{\mathbb{S}_p\}_{p \in \boldsymbol{P}}$ describes a set of special local states used to express winning objectives for the system. As for Petri games we operate with either reachability or safety objectives.

For graphical representation we depict control games as the underlying automaton, denote controllable actions with dotted arrows and mark special places with a double circle.

**Local View** Any play in the automaton, $u \in \mathit{Plays}(\mathcal{C})$, describes a global execution of the system. At every point, each process conceptually decides what next moves to allow. Apart from progressing asynchronously, the processes should also act on partially information, i.e., not act on the entire $u$. Instead every process bases its decision on a restrictive view on the overall play. Process $p_1$ in the example from Fig. 2.6 should not observe any of the actions played by $p_3$, so the local view should not comprise any of those actions. On the other hand, communication between $p_1$ and $p_2$ on $c$ should transmit information previously local to one process and therefore extend the local view. We define the $p$-view on $u$, denoted $\mathit{view}_p(u)$, as the part of the play that is observed by process $p$. Formally $\mathit{view}_p(u)$ is the smallest (shortest) trace $[v]_{\mathbb{I}}$ such that $u \sim_{\mathbb{I}} v\, w$ for some $w$ that does not contain any action from $\Sigma_p$. The $p$-view is hence the smallest prefix of $u$ containing all actions where $p$ is involved in. It is easy to check that $\mathit{view}_p(u)$ is well defined, i.e., there is an unique minimal element. We write $\mathit{Plays}_p(\mathcal{C}) = \{\mathit{view}_p(u) \mid u \in \mathit{Plays}(\mathcal{C})\}$ for the $p$-view of every play in $\mathit{Plays}(\mathcal{C})$.

By definition it always holds that the $p$-view on $u$ is a prefix of $u$. If we think of a trace as a concise concept to model the concurrent nature of an execution, we can view ever linearisation of a trace as execution where the processes act sequentially. As $\mathit{view}_p(u)$ is a prefix of $u$ there hence is a sequential execution of $u$ where all events (actions) from $\mathit{view}_p(u)$ proceed all remaining actions from $u$. As the remaining execution ($w$ in the definition above) does not contain any

action from $\Sigma_p$, process $p$ is not involved in any of those actions and can therefore not tell that they actually happened. The causal dependency comprised in a trace can, on the other hand, express that each communication transmits the entire causal past of the involved processes. If $p$ and another process $p'$ synchronize on an action $a$ their local views align: The shared action $a$ induces a causal dependency in the trace between the actions from $p$ and $p'$. The local view hence constitutes all actions a process is involved in herself as well as all parts of the execution transmitted upon communication.

Having the aforementioned poset representation of a trace $u$ is particularly useful when "computing" the local view. Since the local view is the minimal prefix containing every action from $\Sigma_p$ and the order of the poset agrees with the dependency in a trace, the local view in a poset is the minimal, downwards closed set that contains all actions from $\Sigma_p$. Since incomparable elements in the poset are not causally related, i.e., unordered in time, it is easy to see that this set of actions is indeed the causal past of a process.

When computing the local view of $p_2$ on the play $u_{example}$ we obtain

$$view_{p_2}(u_{example}) = [\ a\ ,\ d\ ,\ c\ ,\ d\ ,\ c\ ]_{\mathbb{I}}$$

In Fig. 2.7 the poset representation of the local view of process $p_2$ is depicted if the grayed out parts are removed. In particular, the actions $d$ and $e$ where $p_1$ is neither involved in nor can hear about in communication are removed from the local view. The trailing $b$ is removed as well as there is no communication informing $p_2$ about the occurrence. Furthermore, note that the local view comprises actions where $p_2$ is not involved in directly. In particular, action $a$ is included in her local view. This goes well with the idea of causal information as communication with $p_1$ on action $c$ transmits the entire causal past of $p_1$, including $a$.

**Controller** To avoid confusion between Petri games and control games we refer to strategies for control games as controllers. A (global) *controller* $\varrho$ is a family of local controller for each processor, $\varrho = \{f_p^\varrho\}_{p \in \boldsymbol{P}}$. A *local controller* for process $p$ is a function $f_p^\varrho : Plays_p(\mathcal{C}) \to 2^{\Sigma_p^{sys}}$ where $\Sigma_p^{sys} = \Sigma_p \cap \Sigma^{sys}$. The set of $\varrho$ compatible plays, $Plays(\mathcal{C}, \varrho)$, is the smallest set satisfying:

- $\epsilon \in Plays(\mathcal{C}, \varrho)$
- If $u \in Plays(\mathcal{C}, \varrho)$ and $u\,a \in Plays(\mathcal{C})$ for $a \in \Sigma^{env}$ then $u\,a \in Plays(\mathcal{C}, \varrho)$
- If $u \in Plays(\mathcal{C}, \varrho)$ and $u\,a \in Plays(\mathcal{C})$ for $a \in \Sigma^{sys}$ and $a \in f_p^\varrho(view_p(u))$ for all $p \in dom(a)$ then $u\,a \in Plays(\mathcal{C}, \varrho)$

The plays admitted by the controller, $Plays(\mathcal{C}, \varrho)$, form a subset of all plays possible in the game. On the other hand, not every play in the game must be admitted by the controller. A controller can hence restrict executions of the underlying plant. Our indented meaning of controllable and uncontrollable can be found in the definition: Uncontrollable actions can always happen, provided they are admitted by the underlying automaton. In order for a controllable actions to occur all involved processes need to agree. The local controllers can hence forbid such actions by excluding them from their decision. Note that the controller makes its decision based on the local view of that process. Even though the overall play has progressed further a local decision is always made based on the information known to one process.
We define the set of *maximal plays* compatible with $\varrho$, $Plays(\mathcal{C}, \varrho)^M$, as the (possible empty) set of all finite plays $u \in Plays(\mathcal{C}, \varrho)$ such that there is no $a$ with $u\,a \in Plays(\mathcal{C}, \varrho)$. Similarly we define $Plays(\mathcal{C})^M$, as the (possible empty) set of all finite plays $u \in Plays(\mathcal{C})$ such that there is no $a$ with $u\,a \in Plays(\mathcal{C})$. A global state $s$ is called *final* if for all $u \in Plays(\mathcal{C})$ with $state(u) = s$, $u \in Plays(\mathcal{C})^M$, that is, once in state $s$ no further action can be executed.

**Reachability Control Games** In reachability games the winning objectives is stated in terms of winning states $\{\mathcal{W}_p\}_{p \in \boldsymbol{P}}$. A controller fulfils a reachability objective if it ensures that every

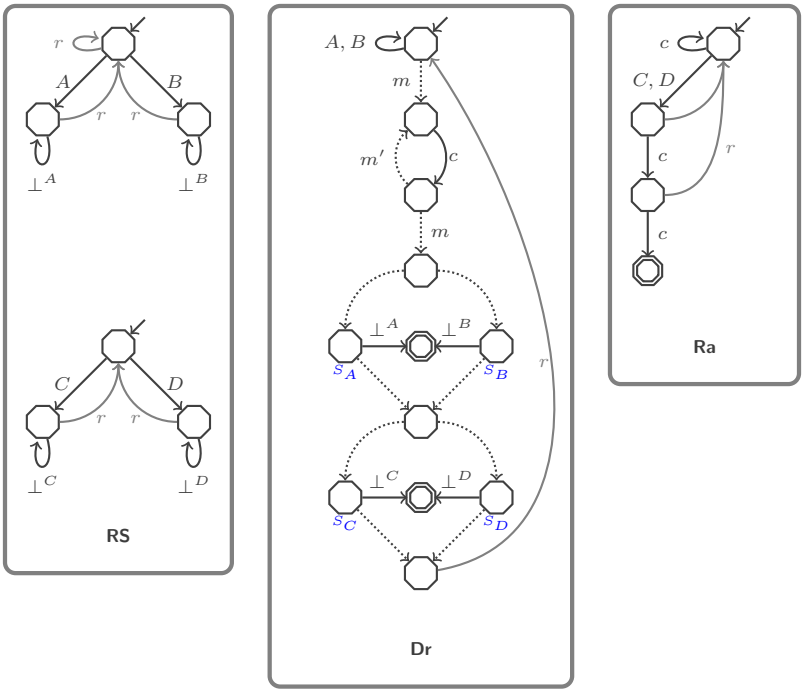



**Figure 2.8** A (safety) control game modelling the behavior of a driver in an unpredictable traffic situation. It comprises two road section *RS*, a news radio *Ra* and a driver *DR*. Each road section models the traffic at two location and can decide which of the construction sides $A$ to $D$ are blocked. The driver needs to gather information and circumvent both blocked sections.

process terminates in a winning state and therefore, in particular, does not admit infinite plays. All existing decidability results for control games [21, 15, 14, 19, 16] are stated in terms of reachability objectives.

**Definition** A controller $\varrho$ is *(reachability) winning* if every $u \in \mathit{Plays}(\mathcal{C}, \varrho)$ is finite and for every $u \in \mathit{Plays}(\mathcal{C}, \varrho)^M$ and $p \in \boldsymbol{P}$, $\mathit{state}_p(u) \in \mathcal{W}_p$

**Safety Control Games** In safety control games the winning objectives is stated in terms of losing states $\{\mathcal{B}_p\}_{p \in \boldsymbol{P}}$. A controller should not permit any play leading to a bad state. A trivial controller would, similar to Petri games, refuse all controllable actions. To avert such trivial controllers we require a controller to progress if possible. A controller is only allowed to terminate, i.e., if the underlying automaton does.

**Definition** A controller $\varrho$ is *deadlock-avoiding* if $\mathit{Plays}(\mathcal{C}, \varrho)^M \subseteq \mathit{Plays}(\mathcal{C})^M$.

A deadlock-avoiding controller hence only terminates in final states. As for Petri games, deadlock-avoidance is a global property.

**Definition** A controller $\varrho$ is *(safety) winning* if it is deadlock-avoiding and for every $u \in \mathit{Plays}(\mathcal{C}, \varrho)$ and $p \in \boldsymbol{P}$, $\mathit{state}_p(u) \notin \mathcal{B}_p$

## Example

Consider the (safety) control game depicted in Fig. 2.8. The game is depicted as a composition of local automata where controllable actions are indicated as dotted arrows. Losing states are marked with a double ring. The action $r$ is depicted in gray to improve readability. The game describes the behavior of a car-owner in the presence of two traffic-construction-sides. It consists of four players. Two of which represent the traffic situation at two distinct road-sections (RS),

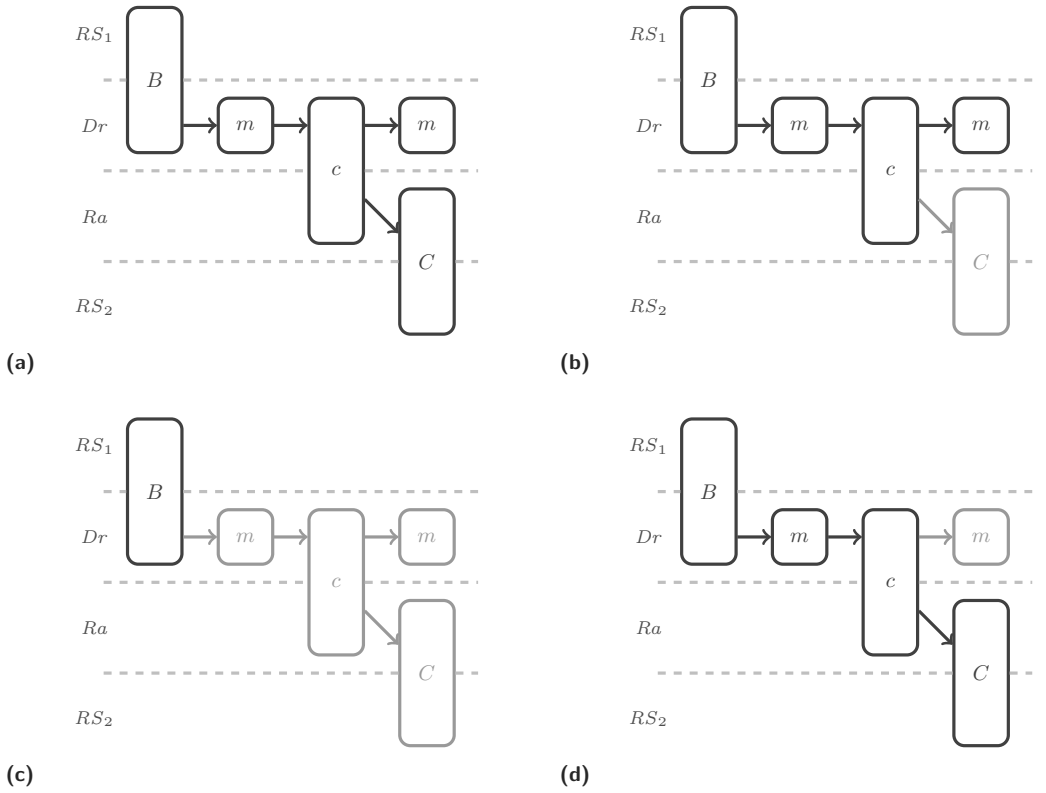



**Figure 2.9** The poset representation of the trace $u = [B, m, c, C, m]_{\mathbb{I}}$ is delineated in (a). The second $m$ and $C$ are independent and can be switched in $u$. They are hence not ordered. (b) depicts the local view of the driver ($Dr$) on $u$, (c) the local view of the first section ($RS_1$) and (d) the local view of the second section ($RS_2$) as well as the radio ($Ra$).

one modelling a local radio-channel (Ra) and the last describing the driver (Dr).
The processes representing the road sections comprise construction sides $A$ and $B$ or $C$ and $D$. Each road section can decide which of the two construction sides is blocked using $A$ to $D$. The radio-channel synchronizes with the latter road-section on $C$ or $D$, i.e., gets informed about the traffic situation at the later section. At all times the radio channel offers the driver to communicate using $c$. In the initial state the driver can synchronize with the first road-section on $A$ or $B$, i.e., observe the traffic situation at the first section. Using a controllable action ($m$) she can move on and synchronize with the radio channel on $c$. Upon communication on $c$ she can decide to either move back and repeat communication using $m'$ or move on using $m$. Utilising controllable action she can, in both sections, decide which of the respective construction sides she wants to pass. In a first stage she can either move through side $A$ or $B$ by passing through states $S_A$ or $S_B$. At the second section she can decide for $C$ or $D$. If she chooses a path that has been decided to be blocked by the corresponding road-section-processes one of the $\bot$-actions can fire, moving her to a losing state; she is locked in traffic jam. If she asks for the same information (using $c$) even though she is already up to date the radio crew is annoyed and causes a loss. Once she successfully passes both road sections the global system resets with the $r$ action and a new work-day with unpredictable traffic begins.

This game has a winning controller: The driver awaits the traffic information for the first section. Afterwards, she moves on and cycles through communication with the news challenge until she knows the traffic situation at the second section. This might take forever if the second road section never decides to block either construction side. Once informed about both sections she proceeds and bypasses both blocked road-sections using her controllable actions. Upon

a rest on $r$ she repeats her procedure. The existence of a winning controller rests upon the communication aspect associated with each synchronization. The communication action $c$ between the radio and driver transmit all available information about the situation at the second road section. A correct circumnavigation of both sections is based on this information.

A possible initial play in this game is the play

$$u = [\ B\ ,\ m\ ,\ c\ ,\ C\ ,\ m\ ]_{\mathbb{I}}$$

On this execution the driver communicated with the news radio before the second road section indicated which section is blocked ($C$). As $C$ causally succeeds the communication action $c$ the driver should, from her local information, not be able to deduce which road section is blocked. The poset representation of $u$ is depicted in Fig. 2.9 (a). Note that the independent actions $C$ and $m$ are not causal related in the trace and therefore unordered in the poset. Fig. 2.9(b)-(d) depict the local views of each process on $u$, i.e., the downwards closed set containing all actions from that process. We realize that our previous intuition about the local information of the driver are confirmed. The local view of $Dr$ is depicted in (b) and does not comprise action $C$. Even though the global play progressed in the sense that $SR_2$ decided which section to block, the driver needs to decide which way to circumvent the road section based on her local view, not including this decision. Fig. 2.9 (c) and (d) depict the local views of the remaining processes. Even though their view is not particular interesting as none of the processes can influences the game in terms of controllable actions, we can still study their local information. Note that $SR_1$ is barely informed as it has not taken part in any action apart from the initial decision to play $B$. Process $S_2$ does obviously know that it itself blocked section $C$ but it can, for instance, not deduce that the driver already progressed using $m$.

Fig. 2.10 depicts the poset representation of a slight variation of the previous trace: After executing $c$ and noticing that no interesting information is available the driver used $m'$ to repeat the process. As the second communication $c$ occurred after the road section played $C$, the information about this event is transmitted upon communication. Building the local view of the driver on the modified play agrees with the play itself. In particular, the occurrence of $C$ is included in her local view. The controller could now, successfully circumvent both sections.

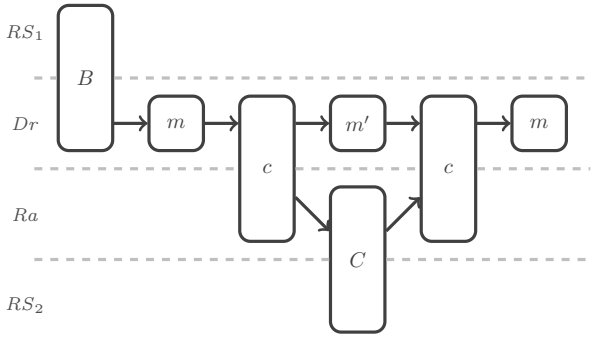



**Figure 2.10** Poset representation of the trace $u = [B, m, c, m', C, c, m]_{\mathbb{I}}$. The local view of the driver ($Rd$) on $u$ agrees with $u$ itself.

# Chapter 3

# Related Work and Comparison

In this chapter we give a quick overview on reactive synthesis in general and, in particular, on related work in the area of distributed synthesis. We outline the key difference between the models studied in this thesis compared to the most influential framework for distributed synthesis [25]. We then give a brief comparison *between* Petri games and control games and highlight their key differences. As one key achievement of this thesis is the transfer of existing results, we summarize the existing results for both Petri games and control games.

## 3.1 Context Of Our Framework

The existing results and attempt to synthesize programs are rich enough to warrant a work on its own. For a thorough introduction to the history and milestones of reactive synthesis we refer the interested reader to [8]. In this section we quickly discuss the origins of reactive synthesis and afterwards consider related work done in the area of *distributed*, reactive synthesis and pinpoint to some key differences that separates previous attempts from the models considered in this thesis.

The synthesis problem as it is known today has first been proposed by Church in the 1957 [8]. He asked for the automatic generation of programs that react to given inputs such that overall execution is admitted by a temporal specification. Church proposed Linear Temporal Logic (LTL) as a formalisms to specify relations between input and output. The objective of the system is hence to produce traces that are admitted by the formula independent of the received inputs. In particular, the implementation acts on *full information*, as it can base its decision on the entire previous execution. While Churchs problem sparked the entire field of reactive synthesis, this thesis focuses on distributed system, i.e, systems comprised of individual components acting on *partial information*.

Synthesis of distributed systems has first been pioneered by Pnueli and Rosner in the 90s. Their most influential work [25] still forms the standard setting for distributed synthesis. They tried to lift the traditional setting of synthesizing a program from a LTL-specification, as asked by Church, to distributed systems. A synthesis problem in their framework consists of both a temporal specification as well as a system architecture, i.e., a description of the structure of the underlying system. The architecture can be seen as directed graph where vertices indicate processes of the system and acrs between them shared variables (channels) used for communication. The processes can read from and write to the variables associated with them and try to produce, as a whole, execution admitted by the logical specification. In particular, they act *synchronously*. Synthesis aims to find finite-state programs that govern the local processes such that the entire system generates admitted executions in an unrestricted environment. As for Petri games and control games the individual processes act on partial information. The existing amount of information can be extended by reading values from the

associated channels. Communication between processes is hence accommodated as information exchange through shared variables. In their early work [25] they already showed that the existence of winning strategies is undecidable in general, even for simple architectures comprising as little as three processes. Later [13] extended this by outlining precise classes of architectures for which the problem is decidable.

While the Pnueli and Rosner framework is regarded "the standard" when it comes to distributed synthesis, it differs quite substantially from the more recent setting we are concerned with in this thesis. We briefly discuss three important ones. The most pressing difference being that they assumed the processes to progress in a *synchronous* lock-step execution, whereas the models we considered in this thesis proceed *asynchronously*, i.e., without a global clock. Secondly, their setting comprises more direct *communication primitive* as each process can explicitly decide what information to forward using the shared variables. This goes well with the intuitive notion of communication in distributed systems where information must, e.g., be send over a network. In contrast, both Petri games and control games consider communication that transmits the entire information available; their entire causal memory. This might, at first glance, seem unrealistic as distributed systems have no unlimited communication capacities allowing to transmit the, possibly unbounded, causal memory of a processes. It does, however, give a good over approximation of whether the prescribed communication structure, defined in the underlying Petri net or asynchronous automaton, is sufficient to admit winning strategies. If a winning strategy relying on causal memory is found one can manually analyse what information are really in need to be transmitted and which can be neglected. Conversely, the non-existence of a strategy in a setting modelled with causal memory already implies the absence of implementation with a more restrictive communication primitive. Lastly, in the Pnueli and Rosner setting the environment is regard as an adversary that can challenge the system by providing arbitrary, non structured inputs. The environment has no underlying assumption but is assumed to be able to produce every possible input at any time. In contrast, Petri games and control games allow to model the environment explicitly, i.e., abstract from viewing the environment as some omnipresent entity that can at all times generate all possible inputs, and, instead, encode assumptions on the environment.

The traditional Pnueli and Rosner work has been extended to the asynchronous setting [24], allowing for descriptions much closer to our setting. Asynchronicity is achieved by deploying an explicit scheduler that can direct the order in which the, originally synchronous, processes progress. As communication is modelled as explicit communication channels, a process might miss information, whereas in our setting, where communication is defied in terms of causal memory, information cannot vanish upon communication. In [27] they showed that the problem is doubly exponential even under strong assumptions on the scheduler and undecidable if more than one process in the system architecture is being synthesized.

## 3.2 Comparing the Two Game Types

In these early framework for distributed synthesis one quickly runs into the undecidability barrier[25, 13]. Control games, first studied in [14], and the more recent Petri games [12] try to get a broader class of decidable synthesis problems by relaxing the communication primitive and thereby avoid trivial barriers as encountered in previous results. We proceed by discussing some of their key similarities as well as their fundamental differences.

**Similarities** Petri games and control games share a lot of similarities. The two most important ones are the asynchronous nature of both games as well as the primitives with which information is transmitted, namely causal memory.

Both games are played *asynchronously*. From her local information a player cannot deduce how many transitions or actions the other players have played. The asynchronous scheduling is

conceptually controlled by the environment, since every possible execution has to be accounted for by the system. Secondly, both model distributed systems are intrinsically based on *partial information*. The processes in the system can possess different information about the global state of the system. Without having explicit local information the distributed components can be regarded as one big system; losing all distributivity in the system. Having an explicit information model motivates the natural question of how the local information change during the execution of the system. In the two asynchronous models we discuss in this thesis the arguably simplest communication model is used: Every communication transfers everything, i.e., the entire *causal past*. At the point of communication the local information of involved components aligns. While the formalism to ensure partial information differs quite substantially (local view on an execution vs. defining strategies in terms of branching processes), the underlying simple primitive is shared.

**Differences** Despite their key similarities both game types chose fundamentally different approaches to model the responsibilities of system and environment. As a Petri game distributes the places between system and environment, it is natural to distinguish the tokens in a game as system and environment player. In particular, the environment is modelled as dedicated players that can interact and communicate like every other player. A system player has full control about her next move, i.e., she decides herself what next move to allow. In the control game, on the other hand, the environment is not modelled as a dedicated player but rather via uncontrollable action. The environment can be regard as an omnipresent adversary that interacts and provides information using those actions. Conceptually we no longer think of a game played between system and environment players but a game played between system players that are subject to environment decisions beyond their control. Both approaches of modelling the environment can be justified by synthesis problems in the real-world.

Petri game strategies can be required to be *deterministic* and all previous results have been established under this premise. For control games, on the other hand, there has been no attempt to work with deterministic controllers and all existing results allow nondeterministic controllers. Unlike the previous differences, this disparity is not related to the game type itself but rather to the choice of the people who introduced them. There is no real reason to restrict Petri games to deterministic strategies and control games to non-deterministic controllers. As control games are conceptually not played between system and environment player there exist multiple valid attempts to define what determinism means in the presence of uncontrollable and controllable actions. In this work we do not attempt to come up with a definition by ourself but merely discuss some possibilities in our conclusion. Throughout this thesis we consider controllers that are no subject to any determinism criterion. Nondeterministic strategies for Petri games are in general more powerful as there exists games that can only be won by nondeterministic strategies. One such example is depicted in Fig. 3.1.

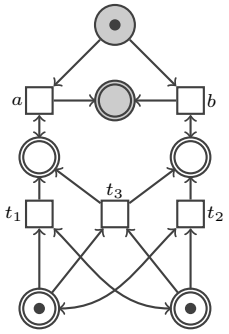



**Figure 3.1** (Reachability) Petri game that can only be won by non-deterministic strategies. To win the system must enabled transitions *a and b* since otherwise the environment can choose $t_1$ or $t_2$ to block the system from reaching a winning marking. If both are enabled the environment can create non-determinism using $t_3$.

## 3.3 Previous Results for Asynchronous Games

Both Petri games and control games are used to pose synthesis problems for distributed systems. As winning strategies/controllers conceptually correspond to correct implementations, we are

mainly interested in the existing of a winning strategy/controller. We thus call a class of games *decidable* if the existence of a winning strategy is decidable.

Unlike the traditional Pnueli and Rosner setting where the boundaries of decidability are well understood [13], decidability for both game types in this thesis is an open question and yet to be fully explored. The question whether (bounded) Petri games or control games are decidable in general is still open. There are, however, interesting classes in both types for which decision procedures exist. As mentioned before progress has, so far, been made independently for both game types, resulting in mostly incomparable results. In this section we briefly outline the existing results for both.

## Petri Games

Since Petri games utilize Petri nets as the underlying arena they offer more flexibility when it comes to modelling interactive ad-hoc systems where players join and leave. In particular, they allow for spawning and termination of players resulting in a flexible amount of players in the game that, with progressing execution, might grow arbitrary large. Even though reachability is still decidable in unbounded Petri nets [18], *unbounded* Petri games are in general undecidable [12].

For *bounded* Petri games there exist two known classes of decidability: Bounded Petri games with at most one system or at most one environment player [12, 11]. In both results they considered safety Petri games and required strategies to behave deterministic. Petri games are intrinsically based on the partial information of each player. [12, 11] achieved to reduce a game played on partial information to an ordinary two player game. Two player games are played on an underlying graph whose nodes are controlled either by the system or the environment. Whenever on a system node a strategy can decide where to go next, whereas the environment can appoint the next move from environment nodes [8]. In particular, the system always knows what the last moves of the environment have been, i.e., can base its decision on *full* information. To reduce a game on partial information to a game on full information without violating the information of each player they scheduled the game such that invalid information flows are prevented. For Petri games with at most one environment player [11], it suffices to postpone every environment decision as far as possible, i.e., until every player either gets informed about the environment in the next move anyway or can play without ever needing the environment again. For Petri games with at most one system player [12] the scheduling of transitions is left to the environment regarding the scheduling itself as adversary. While this might leak information, they showed that the environment can always schedule such that no illegal information are given to the system. The strategy needs to base its decision on the last known location of all other player. Both restrictions of the scheduling allow to model the local information in a fully informed setting. In addition to their exponential time decision procedure, both provided an exponential lower bound, thus establishing the EXPTIME-completeness of both classes.

For Petri games there also has been an increasing effort to investigate *bounded-synthesis* [7] where strategies are looked for up to a given size (bound). The size-bound can be raised incrementally giving a semi-decision procedure for general Petri games. Unlike the first two results, bounded synthesis *can* provide a winning strategy but is not able to prove the non-existence of one. There is a lot of exiting work on tool support for bounded synthesis [9, 17, 10].

## Control Games

Control games can be defined in two variations, called *process-based* and *action-based*. Throughout this thesis and, in particular, in Chapter 2 we work with process-based games, where the decision of what actions to enable is made by the processes. In contrast, in action-based control games the processes are conceptually regarded as memory cells and the actions as agents acting on them. All enabling-decision are therefore made by the actions. While a process decides what actions to enable based on its causal information, an action can make a decision based on the

causal information of *all* involved processes. Intuitively an action-based control game allows for more precise control as decision are based on a greater pool of information. A first translation of asynchronous games has been proposed in [22] where they showed that processes based games are reducible to the action-based variant.

Control games have, so far, been only considered for reachability objectives. Results were obtained for four distinct classes [19, 14, 15, 16]. In [19] they studied asynchronous automata, called connectedly communicating, where there is a bound on the number of actions possible between the communication of two processes, i.e., automata where the processes communicate "sufficiently many times". They showed that the MSO (monadic second order logic) theory of connectedly communicating automata is decidable and thereby every control game played on them.

The causal information during the execution of a control game can grow arbitrarily large. Throughout the execution of a game a controller might need to base its decision on a growing amount of information. [14] restricted the distributed alphabet to a special structure; a cograph. Using an inductive argument on the structure of the game they were able to show that the causal information in such games can be bounded, while not compromising winning controller. Whenever the information that needs to be considered for winning controller is bounded, decidability follows by enumeration of all possible controllers.

Given a control game one can analysis the communication structure between the processes by building the *communication architecture*. This is an undirected graph where the vertices comprise all processes and edges between them indicate shared actions between the processes. In [15, 21] they showed that control games played on automata with acyclic communication architecture are decidable. A process at the leaf of the architecture can be encoded in its parent by simulating the process locally in an exponentially bigger process. Using an involved set construction they showed that the combination of both processes does not leak information, i.e., the information flow of the original game is preserved. By repetitively encoding leafs in their parents, any acyclic game can be reduced to a control game consisting of a single process for which decidability is trivial[1]. Because each encoding of leaf and parent results in an exponentially bigger process the resulting game is of non-elementary size in the depth of the communication architecture. In addition to their upper bound, [15] proved the, to our knowledge, first non-elementary lower bound for control games.

Recently [16] proposed the new class of decomposable games and established decidability. They proved that this new class subsumes all previously known classes of decidability, while containing new intriguing classes such as four player games. Unlike previously identified decidable classes, which are mostly based on the structure of the game, decomposable games are defined in terms of executions making the results less intuitive and applicable.

## 3.4 Our Contributions

Our main contribution is a translation between Petri games and control games in both directions. To this extent we define a notion of game-equivalence that is based on weak-bisimilar behavior of possible strategies/controllers and show that our translation fulfils it. The games returned by our translation are therefore not only winning-equivalent, but preserve the structure of the game. Winning strategies can hence be transferred into bisimilar ones in the translated game. Our translations allow us to unify the scattered decidability landscape by using existing results for the respective different game type. For instance, the two examples given in the preliminaries (Fig. 2.4 and Fig. 2.8) correspond to the newly identified classes of acyclic Petri games and single system control games. We conjecture that there are many more decidable classes for both games that are yet to be discovered. Our translations are not limited to existing results and

[1] A single process control game can be seen as game between system and environment on full information.

make future results applicable in both types. We hope that the translation brings both "camps" together and leads to a joint effort to discover the decidability boundaries of asynchronous distributed synthesis.

In addition to our exponential translation we provide exponential lower bounds in both direction indicating a intrinsic difference of both game types. The lower bounds allow for a precise study of the modelling advantage of one game type over the other. While this can be overcome with exponential effort (as witnessed by our translations), it might motivate the study of unified game models that comprise the most intriguing features of both frameworks.

## Chapter 4

# Game Equivalence

We translate games from one class to the other one. The minimum requirement for any reasonable translation is *winning-equivalence*: The system has a winning strategy in one game if and only if it has a winning strategy in the translated game. A trivial translation fulfilling this is to first solve the games and then return a minimal winning equivalent game of the other type. Such a translation is not desirable, especially since decidability in both, control games and Petri games, is still an open question [12, 22].

Asynchronous games can be used to state synthesis problems for distributed systems where winning strategies correspond to correct implementations in the distributed system. From a very abstract point of view one would like to be able to extract a implementation from winning strategy. A translation between two game types should ideally be modular in the sense that a user can model a given problem in one game type (say control games), translate it to Petri games and can, from a winning strategy of a Petri game, construct her implementation. A translation should therefore not only preserve the existence of a winning strategy but ideally allow for structurally similar strategies. This motivates the search for translations that preserve the structure of a game. We propose *strategy-equivalence*, defined in terms of a weak bisimulation between strategies, as an adequate equivalence notion.

Bisimulation (and weak bisimulation) are well studied relations for concurrent systems [1, 23]. In two bisimular systems the states of both systems can be grouped together in classes of "equal behavior". Every move from a state can be matched with an equal move (preceded by possible internal computation) in every state of that group, resulting in states that are again grouped together. It is easy to define a bisimulation between the markings in a Petri net and the global states in an asynchronous automaton. To related dynamic executions in both games this is, however, not sufficient. Instead we want to express that any strategy for one game can be matched by a strategy in the respective different game that allows equivalent (bisimilar) behavior, i.e., allows identical actions/transitions. The current state of a game is described as either a marking in a strategy (in Petri games) or a global execution (play) of the automaton (in control games). We consider a strategy $\sigma$ for a Petri game and a controller $\varrho$ for a control game equivalent if there is a weak bisimulation between the reachable markings in the branching process of the strategy ($\mathcal{R}(\mathcal{N}^\sigma)$) and the plays that are compatible with the controller ($\mathit{Plays}(\mathcal{C}, \varrho)$). There hence is a relation between the current "situations" in both games where bisimulation requires equivalent situations to be extended in an identical form.

To compare Petri games and control games we require a shared core of transitions/actions between two games. We refer to them as *observable*. All transitions/actions that are not shared are considered *internal* ($\tau$), i.e., correspond to internal computation steps that must not be matched in related situations.

**Definition** A strategy $\sigma$ for $\mathcal{G}$ and controller $\varrho$ for $\mathcal{C}$ are *bisimilar* if there exists a relation $\approx_{\mathfrak{B}} \subseteq \mathcal{R}(\mathcal{N}^\sigma) \times \mathit{Plays}(\mathcal{C}, \varrho)$ such that:

- $\mathit{In}^\sigma \approx_{\mathfrak{B}} \epsilon$
- If $M \approx_{\mathfrak{B}} u$ and $M \left[ a \right\rangle M'$ for some $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ then there exists $u'$ with $u' = u\,\tau^*\,a\,\tau^* \in \mathit{Plays}(\mathcal{C}, \varrho)$ and $M' \approx_{\mathfrak{B}} u'$.
- If $M \approx_{\mathfrak{B}} u$ and $M \left[ \tau \right\rangle M'$ for some $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ then there exists $u'$ with $u' = u\,\tau^* \in \mathit{Plays}(\mathcal{C}, \varrho)$ and $M' \approx_{\mathfrak{B}} u'$.
- If $M \approx_{\mathfrak{B}} u$ and $u' = u\,a \in \mathit{Plays}(\mathcal{C}, \varrho)$ then there exists $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ with $M \left[ \tau^* a \tau^* \right\rangle M'$ and $M' \approx_{\mathfrak{B}} u'$.
- If $M \approx_{\mathfrak{B}} u$ and $u' = u\,\tau \in \mathit{Plays}(\mathcal{C}, \varrho)$ then there exists $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ with $M \left[ \tau^* \right\rangle M'$ and $M' \approx_{\mathfrak{B}} u'$.

The definition requires that there exist a relation that witnesses bisimulation ($\approx_{\mathfrak{B}}$) and relates markings and plays. The initial situation in both the strategy ($\mathit{In}^\sigma$) and the controller ($\epsilon$) are related and any move in one strategy/controller can be matched after some preceding $\tau$-steps.

**Definition** A Petri game $\mathcal{G}$ and a control game $\mathcal{C}$ are called *strategy-equivalent* if: For every winning strategy $\sigma$ for $\mathcal{G}$ there exists a bisimilar winning controller $\varrho_\sigma$ for $\mathcal{C}$ and for every winning controller $\varrho$ for $\mathcal{C}$ there exists a bisimilar winning strategy $\sigma_\varrho$ for $\mathcal{G}$

Note that strategy-equivalent games are by definition winning-equivalent. The converse does not hold in general.

The definition of strategy equivalence is, on its own, independent of translating a game but can be seen as a general notion to compare two games. In particular, we note that if the set of shared transitions/actions between both games is empty, there are no observable transitions/actions and two games are hence strategy-equivalent iff they are winning-equivalent. For a translation as attempted in this thesis, we use the notion of strategy-equivalence to justify structural identical games and therefore do not want to consider the set of shared events as empty. If we, e.g., attempt to translate a Petri game to a control game we aim for a game that comprises all transitions of the Petri game as actions, i.e., $\mathcal{T} \subseteq \Sigma$. Every behavior in the Petri game should hence be matched in the translated control game.

We already argued why we are interested in translations that permit similar strategies/controllers. Strategy-equivalence captures this very well as any winning strategy/controller corresponds to a bisimilar winning controller/strategy in the other game. In strategy-equivalent games we can hence grantee that any environment controlled move can be matched and everything in control of the system can be matched as well, i.e., a strategy/controller allows reaction in an equivalent way.

Chapter 5

# Translating Petri Games to Control Games

In this chapter we provide an exponential translation from Petri games to control games. We prove that our translation yields strategy-equivalent (and therefore winning-equivalent) games. Moreover we give an exponential lower bound showing that our translation is asymptomatically optimal when restricting to strategy-equivalent translations. The existing decidability results for control games [14, 19, 15, 16] are all stated in terms of reachability objectives. We therefore explicitly give the translation for games with reachability winning conditions. It is, however, easy to extend the translation to other winning objectives.

## 5.1 Construction

We describe the construction of the translated game. We begin by defining our translation for a restricting class of Petri games, called sliceable. In Sec. 5.5 we outline how the translation can be generalized to the broad class of concurrency-preserving games.

**Slices** A Petri game describes the *global* behavior of the players. By contrast a control game is defined in terms of *local* processes. Similarly, a Petri game strategy is a global branching process opposed to a family of local controllers for control games. This is due to the intrinsic nature of the underlying model: Petri nets model global behavior, whereas asynchronous automaton are defined locally. The first step towards a successful translation is to dismantle a global Petri net into local parts. To this extent we introduce *slices*.

**Definition** A *slice* of a Petri net $\mathcal{N}$ is a net $\varsigma = (\mathcal{P}^\varsigma, \mathcal{T}^\varsigma, \mathcal{F}^\varsigma, In^\varsigma)$ such that :

**(1)** $\varsigma \sqsubseteq \mathcal{N}$ **(2)** $|In^\varsigma| = 1$

**(3)** $\forall t \in \mathcal{T}^\varsigma : |pre^\varsigma(t)| = |post^\varsigma(q)| = 1$ **(4)** $\forall p \in \mathcal{P}^\varsigma : post^{\mathcal{N}}(p) \subseteq \mathcal{T}^\varsigma$

A slice is a subnet of $\mathcal{N}$ **(1)** that is concurrency-preserving **(2)** and contains exactly one player **(3)**. **(4)** requires the slice to allow every behavior, i.e., if a place is added to a slice then all outgoing transitions must also be added. Note that **(1)** requires the flow to be preserved between the nodes that are added to the slice. A slice describes the behavior of a single token in the global net. One can think of a slice as a finite state automaton where the token marks the current state.

We call a family of slices $\boldsymbol{S} = \{\varsigma_i\}_{i \in \mathfrak{I}}$ over some finite index set $\mathfrak{I}$, *compatible* if the places of the slices are disjoint. For a compatible family $\boldsymbol{S}$ we define the composition $\langle \|_{\boldsymbol{S}} \rangle$ as the Petri net with places $\biguplus_{i \in \mathfrak{I}} \mathcal{P}^i$, transitions $\bigcup_{i \in \mathfrak{I}} \mathcal{T}^i$, flow relation $\biguplus_{i \in \mathfrak{I}} \mathcal{F}^i$ and initial marking $\biguplus_{i \in \mathfrak{I}} In^i$. Since the places of slices are disjoint, all unions, except for the union of transitions, are disjoint. Transitions can, however, be shared between multiple slices, creating synchronization. A Petri net $\mathcal{N}$ is *sliceable* if there exists a compatible family of slices $\boldsymbol{S}$, s.t. $\mathcal{N} = \langle \|_{\boldsymbol{S}} \rangle$. Due to

the fact that the places in an a compatible family are disjoint, $\biguplus_{i\in\mathfrak{I}} \mathcal{P}^i$ also forms *partition* of $\mathcal{P}^{\mathcal{N}}$. Sliceable nets are nets that can be described as the composition of local tokens that move along individual slices. We call such a family of slices a *distribution* (or slice-distribution) of $\mathcal{N}$. Sliceable nets are always concurrency-preserving and safe. We extend slices to Petri games in the natural way by distinguishing between system and environment places and marking places in slices as winning. A Petri game is therefore sliceable iff the underlying net is sliceable. Fig. 5.3 depicts a Petri game (a) and a distribution into slices (b). Note that both slices synchronize on the shared transitions $a$ and $b$.

It is interesting and important to note that not every Petri net (Petri game) is sliceable, even in the concurrency-preserving, bounded case where each place is part of a reachable marking. For an example of this consider Fig. 5.1. Petri nets can hence describe complex behavior that cannot be modelled by viewing the net as a composition of local tokens that can share transitions. If a net is sliceable a distribution must not be unique.

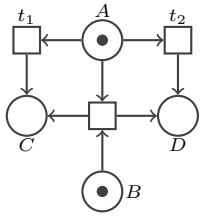



**Figure 5.1** Concurrency preserving, safe Petri net that is not sliceable. Because of transitions $t_1$ and $t_2$ both places $C$ and $D$ must belong to the same slice as place $A$. One of them must, however, also be in the same slice as $B$.

**Commitment Sets** In control games actions are either controllable or uncontrollable, whereas in Petri games transitions can only be forbidden from involved system places. If we translate a Petri game to a control game we want to represent transitions as actions. We hence require actions that can only be controlled by some of the involved player. This cannot be expressed directly using controllable and uncontrollable actions. We overcome this by using *commitment sets*. Every process that should be able to control an action does not restrict it directly but instead chooses a commitment set, i.e., moves to a state that explicitly encodes its decision.

**Description of $\mathcal{C}_{\mathcal{G}}$** For now we fix a sliceable (reachability) Petri game $\mathcal{G} = (\mathcal{P}_{\mathcal{S}}, \mathcal{P}_{\mathcal{E}}, \mathcal{T}, \mathcal{F}, In, \mathcal{W})$ and slice-distribution $\boldsymbol{S}$. We begin by defining our translated control game $\mathcal{C}_{\mathcal{G}}$. Afterwards, we describe a possible modification $\widehat{\mathcal{C}_{\mathcal{G}}}$ used in the context of determinism. Both translations are depicted in Fig. 5.2 where the former is obtained if the red parts are excluded and the latter with the red parts included. As we begin by describing $\mathcal{C}_{\mathcal{G}}$ the red parts should be ignored at first.

The causal information in a Petri game is carried by tokens. Since a slice describes the movement of precisely one token, it is natural to view every slice as a process in a control game. We therefore transform each slice $\varsigma \in \boldsymbol{S}$ into a local process that is described by a local automaton $\wr_S$[1]. Every place of the slice becomes a state in the process[2]. From now on we hence use the terms slice and process interchangeably. Every local automaton starts in the state that corresponds to the initial place of the slice. For every system place $q$ we add the aforementioned commitment sets. These are states of the form $(q, A)$ that encode every possible choice made by a token on place $q$, i.e., every $A \subseteq post^{\mathcal{G}}(q)$. A controller can later on achieve any possible combination of allowed transitions by moving to an appropriate commitment set.

We add every transition $t \in \mathcal{T}$ from the Petri game as an uncontrollable action. Action $t$ involves all processes that are build from a slice taking part in $t$, i.e., all slices $\varsigma$ where $t \in \mathcal{T}^{\varsigma}$. To choose commitment sets we add additional controllable $\tau_{(q,A)}$-actions that are local to one process. We assume that each process chooses at most one commitment set.

[1] Our final control game consists of the parallel composition of automata as defined in the preliminaries. Our translation therefore yields a more specific class of control games.

[2] To avoid clustered notation we view a place $q$ as both a place in the Petri game as well as a state in our translated game. We proceed similar for transitions/actions. If we refer to those elements it should be clear from the context whether we consider it as a transition in the Petri game or an action in the control game.

Define $\boldsymbol{P} = \boldsymbol{S}$ and the alphabet of the asynchronous automaton to be $(\Sigma, dom)$ with:

$$\Sigma = \mathcal{T} \cup \{\tau_{(q,A)} \mid q \in \mathcal{P}_\mathcal{S} \wedge A \subseteq post^\mathcal{G}(q)\}$$
$$\cup \{ \lightning^{(q,A)}_{[t_1,t_2]} \mid q \in \mathcal{P}_\mathcal{S} \wedge A \subseteq post^\mathcal{G}(q) \wedge t_1 \neq t_2 \in A\}$$

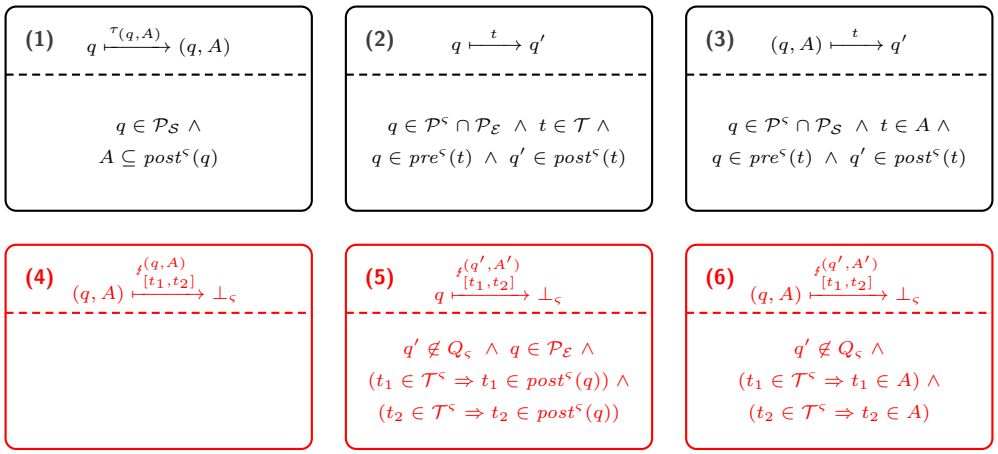
and $dom : \Sigma \to 2^{\boldsymbol{P}} \setminus \{\emptyset\}$:

$$dom(t) = \{\varsigma \in \boldsymbol{S} \mid t \in \mathcal{T}^\varsigma\} \quad \text{for } t \in \mathcal{T}$$
$$dom(\tau_{(q,A)}) = \{\varsigma\} \text{ where } \varsigma \in \boldsymbol{S} \text{ is the unique slice s.t. } q \in \mathcal{P}^\varsigma$$
$$dom(\lightning^{(q,A)}_{[t_1,t_2]}) = \{\varsigma \in \boldsymbol{S} \mid t_1 \in \mathcal{T}^\varsigma \vee t_2 \in \mathcal{T}^\varsigma\}$$

For each slice $\varsigma = (\mathcal{P}^\varsigma, \mathcal{T}^\varsigma, \mathcal{F}^\varsigma, In^\varsigma) \in \boldsymbol{S}$ we define the local process $\text{Ω}_\varsigma = (Q_\varsigma, \vartheta_\varsigma, q_{0,\varsigma})$ with $\vartheta_\varsigma \subseteq Q_\varsigma \times \Sigma_\varsigma \times Q_\varsigma$ as:

- $Q_\varsigma = \mathcal{P}^\varsigma \cup \{(q, A) \mid q \in \mathcal{P}^\varsigma \cap \mathcal{P}_\mathcal{S} \wedge A \subseteq post^\varsigma(q)\} \cup \{\bot_\varsigma\}$
- $q_{0,\varsigma}$ is the unique state s.t. $In^\varsigma = \{q_{0,\varsigma}\}$

and $\vartheta_\varsigma$ is given by:




Define $\mathcal{A}_\mathcal{G}$ as the parallel composition of each process $\bigotimes_{\varsigma \in \boldsymbol{S}} \text{Ω}_S$.
And $\mathcal{C}_\mathcal{G} = (\mathcal{A}_\mathcal{G}, \Sigma^{sys}, \Sigma^{env}, \{\mathcal{W}_\varsigma\}_{\varsigma \in \boldsymbol{S}})$ where

- $\Sigma^{sys} = \{\tau_{(q,A)} \mid q \in \mathcal{P}_\mathcal{S} \wedge A \subseteq post^\mathcal{G}(q)\}$
- $\Sigma^{env} = \mathcal{T} \cup \{ \lightning^{(q,A)}_{[t_1,t_2]} \mid q \in \mathcal{P}_\mathcal{S} \wedge A \subseteq post^\mathcal{N}(q) \wedge t_1 \neq t_2 \in A\}$
- $\mathcal{W}_\varsigma = (\mathcal{W} \cap \mathcal{P}^\varsigma \cap \mathcal{P}_\mathcal{S}) \cup \{(q, A) \mid q \in (\mathcal{W} \cap \mathcal{P}^\varsigma) \wedge A \subseteq post^\varsigma(q)\}$

**Figure 5.2** The translated control game for a (reachability) Petri game $\mathcal{G} = (\mathcal{P}_\mathcal{S}, \mathcal{P}_\mathcal{E}, \mathcal{T}, \mathcal{F}, In, \mathcal{W})$ and distributed in slices $\boldsymbol{S} = \{\varsigma_i\}_{i \in \mathfrak{I}}$. Excluding the red parts it depicts the definition of $\mathcal{C}_\mathcal{G}$. Including the red parts it depicts the definition of $\widehat{\mathcal{C}_\mathcal{G}}$.

The transition relation of $\mathfrak{Q}_\varsigma$ is given by three rules: If a process is on an environment place $q$ an action $t$ can fire if it involves this place in the Petri game, i.e., if $t \in pre^\varsigma(t)$. In this case the process is moved to the state $q'$ that corresponds to the place that is reached when firing transition $t$ in the slice $(q' \in post^\varsigma(t))$ **(2)**. For an environment place the local automaton $\mathfrak{Q}_\varsigma$ hence copies the structure of slice $\varsigma$ and simply replaces all transitions between places with actions between states. Since all processes that correspond to slices taking part in $t$ must allow $t$ in their local description, executing $t$ in the automaton has the same effect as firing $t$ in the Petri game. Since all the actions that correspond to transitions are uncontrollable, a controller on a state that corresponds to an environment place cannot avert any behavior. If a process is on a system place it should be allowed to control the outgoing actions using the aforementioned commitment sets. From every system place $q$ the process can move to every possible commitment set of that state using the local $\tau_{(q,A)}$-action **(1)**. The behavior from a state representing a commitment set is almost identical to that from an environment place, i.e., the transition fires if in the precondition $(q \in pre^\varsigma(t))$ and moves the process to the place reached when firing $t$ in the slice $(q' \in post^\varsigma(t))$. The only difference is that action $t$ is only added if included in the chosen commitment set **(3)**. While a process on an environment place cannot control any behavior a process on a system place can choose a commitment set and thereby implicitly restrict which of the outgoing should be allowed. Using the commitment set construction we can hence guarantee that only processes on system places can restrict actions.

An example translation is depicted in Fig. 5.3[3]. In the Petri game $\dot{\mathcal{G}}$ in (a) the token initially in $A$ can move to $B$ taking either $e_1$ or $e_2$. The system player starting in $C$ can shift to $D$ utilizing transition $i$. Afterwards, they can synchronize on either $a$ or $b$, returning them to the initial configuration or letting them stay in $B$ and $D$. This game can be won by the systems: It can play any combination of $a$ or $b$ as long as at some point it refuses to take both. The players hence eventually terminate in $B$ and $D$. Note that justified refusal forbids the system player to base its decision (enable $a$ and/or $b$) on the most recent decision of the environment player ($e_1$ or $e_2$). In (b) a possible (in this case unique) slice distribution is depicted. Transitions $a$ and $b$ are shared between both slices. In (c) the translated game is given if the red parts are excluded. Each slice in (b) defines a process in the control game where every place of a slice is added as a local state of one of the processes. The slice that only contains environment places results in the left process $p_1$ that comprises no controllable actions. Every transition between the places of the slice is added as an action between the states. The second slice results in process $p_2$. For the system places $C, D$ there are commitment sets $\{C\} \times 2^{\{i\}}$ and $\{D\} \times 2^{\{a,b\}}$ ranging over the set of outgoing transitions. A controller can for every system place precisely determine what to allow by choosing an appropriate commitment set using the controllable $\tau$-actions. The actions $a, b$ and $i$ can only occur from a commitment set that includes them[4]. In our construction only the second player is able to control $a$ or $b$, just like in the Petri game. Note that, as in $\dot{\mathcal{G}}$, the second process $p_2$ needs to make a decision (allow $a$ $b$ or both) without yet knowing the most recent decision of $p_1$.

## 5.2 On Nondeterminism

For Petri games one can focus on *deterministic* strategies. In such strategies every system place allows transitions such that in every situation at most one of them is enabled. We have not described yet how strategies and controllers can be translated between $\mathcal{G}$ and $\mathcal{C}_\mathcal{G}$. The high level idea is that every place allows exactly the transitions that the controller for $\mathcal{C}_\mathcal{G}$ chose as a commitment set. To yield deterministic strategies we hence want to restrict the commitment sets

[3] Throughout this section we switch between explaining concept on an example and talking about general games. We hence annotate the example with a dot to ease readability.

[4] The attentive reader has noticed that the commitment set for transition $i$ is redundant. Making $i$ controllable and add no commitment set would also be feasible. We decided to omit such special cases in our definition in the pursue of a concise construction.

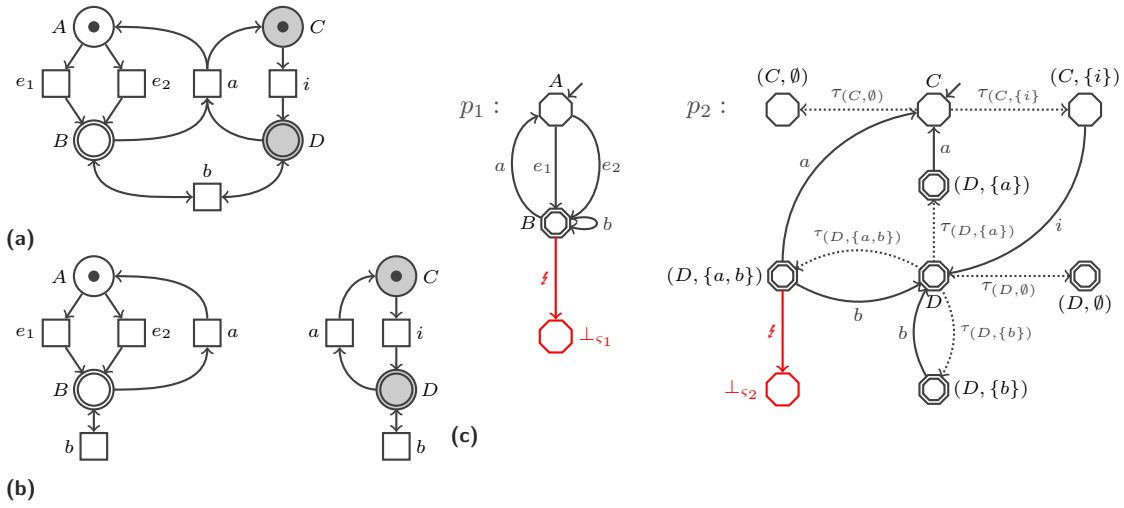



**Figure 5.3** Exemplary sliceable Petri game $\dot{\mathcal{G}}$ (a), a possible (in this case unique) distribution in slices (b) and the asynchronous automaton $\dot{\mathcal{C}}_{\mathcal{G}}$ obtained by our translation (c). Including the red parts (c) delineates $\widehat{\mathcal{C}_{\mathcal{G}}}$, which is identical to $\dot{\mathcal{C}}_{\mathcal{G}}$ except for additional $\lightning$-actions. The $\lightning$-action leaving from states $B$ and $(D, \{a, b\})$ is an $\lightning^{(D,\{a,b\})}_{[a,b]}$ action which is labelled as "$\lightning$" to improve readability.

such that from any chosen commitment set at most one action can occur. To put it differently: We want to penalize a controller if it allows commitment sets such that two distinct action from the same set can occur from the same global state.

We modify the previous $\mathcal{C}_{\mathcal{G}}$ slightly and obtain the new control game $\widehat{\mathcal{C}_{\mathcal{G}}}$. The structure of the latter is almost identical to the former but adds further uncontrollable actions. These actions enforce every winning controller to behave "deterministic" in the sense that from every commitment set at most one action is possible. The construction of $\widehat{\mathcal{C}_{\mathcal{G}}}$ is depicted in Fig. 5.2 by including the red parts. In $\widehat{\mathcal{C}_{\mathcal{G}}}$ we equip every process with an additional $\bot$-state. This state is neither winning nor has it any outgoing actions. As soon as a process has entered a $\bot$-state, no winning configuration is reachable and the game is therefore lost by the system. The situation to cover comprises a process that has chosen a commitment set, i.e., is in a state $(q, A)$ and two distinct actions $t_1, t_2$ in $A$. For every such situation we add a $\lightning^{(q,A)}_{[t_1,t_2]}$-action that involves all processes involved in $t_1$ or $t_2$ and can be taken exactly if there is a process in local state $(q, A)$ and $t_1$ and $t_2$ are both enabled from the current global state.

The three rules of $\vartheta$ add the $\lightning^{(q,A)}_{[t_1,t_2]}$-action to each process. The action can be executed if there is one process in state $(q, A)$ **(4)** and all others are in states such that both $t_1$ and $t_2$ can occur. To ensure that $t_1$ and $t_2$ are both enabled, we distinguish between system and environment places: Every process $\varsigma$ on an environment place needs to be in the right state, i.e., if $\varsigma$ is involved in $t_i$ ($t_i \in \mathcal{T}^{\varsigma}$), then $t_i$ is in the postcondition of its current state for $i = 1, 2$ **(5)**. If on a system place, $t_i$ must not only be in the postcondition but also in the currently chosen commitment set **(6)**. Whenever all processes involved in either $t_1$ or $t_2$ are in states that have an outgoing $\lightning^{(q,A)}_{[t_1,t_2]}$-action we know that $t_1$ and $t_2$ can both occur from the current global state. $\lightning^{(q,A)}_{[t_1,t_2]}$ is hence possible iff both $t_1$ and $t_2$ are possible.

$\widehat{\mathcal{C}_{\mathcal{G}}}$ for the example game in Fig. 5.3 (a) is depicted in (c) if the red parts are included. There is a $\lightning^{(D,\{a,b\})}_{[a,b]}$ action possible if the $p_1$ is in $B$ and $p_2$ in $(D, \{a, b\})$. In this case $p_2$ has chosen a commitment set such that both $a$ and $b$ are enabled. Recall the game in Fig. 3.1 from Chapter 3 that can only be won by non-deterministic strategies. The interested reader is advised to construct the translation of this game. With the added $\lightning$-action the translation has no winning controller.

## 5.3 Correctness

We still need to argue that our translation are correct, i.e., yield strategy-equivalent games. It thus remains to prove that $\mathcal{G}$ and $\mathcal{C}_\mathcal{G}$ are strategy equivalent and that $\mathcal{G}$ and $\widehat{\mathcal{C}_\mathcal{G}}$ equivalent if we restrict strategies for Petri games to be deterministic. Our final result is:

> **Theorem 1** $\mathcal{G}$ and $\mathcal{C}_\mathcal{G}$ are strategy-equivalent. $\mathcal{G}$ and $\widehat{\mathcal{C}_\mathcal{G}}$ are strategy-equivalent if we require deterministic Petri game strategies.

To prove this, so we need to show that we can translate winning strategies and controllers between both games in a bisimilar way. We begin by giving an informal description of how this translation looks like. This high level outline should already be sufficient to comprehend why the translation work. We then proceed by giving a formal description in Sec. 5.3.2, including proofs. As the latter section is very technical it might be helpful to skip it on first read and continue with the remainder of this thesis.

### 5.3.1 Intuition

**Controllability** Our translation needs to overcome the different formalism in both game types. In control games actions are either controllable or uncontrollable and can hence be controlled by either all or none of the involved player. When translating a Petri game to a control game we require actions that can be restricted by some processes but not all of them, i.e., only player on system places should control them. Our commitment set construction overcomes this difference. A process on a local state that corresponds to a system place can indirectly control all outgoing actions by using a commitment set, while a process on a state that corresponds to an environment place can restrict none of the (uncontrollable) actions. A process on a local state $s$ has the same control possibilities as the place that $s$ is build from, i.e., can either restrict all actions (system place) or none (environment place). The high level idea of a strategy/controller translation consists of simulating and copying the decision made by the existing strategy/controller.
Given a strategy for $\mathcal{G}$ we can build a controller for $\mathcal{C}_\mathcal{G}$ that copies the decision of the strategy: If a process is on a local state that corresponds to an environment place, it cannot control any behavior and neither can the strategy. If on a system place the controller can choose a commitment set and thereby implicitly control the outgoing actions. The controller can copy the system place by choosing the commitment set that contains all transitions allowed from the place. Using commitment sets it can copy the decision precisely.
For the reverse direction we are given a controller for $\mathcal{C}_\mathcal{G}$ and build a strategy for $\mathcal{G}$. Any system place in a strategy has to decide which transitions should be allowed. The controller can, if on a system place, choose a commitment set. The place now allows exactly the transitions that the controller has chosen as a commitment set. From an environment place a strategy cannot control any behavior and neither can the controller given it is on a state that corresponds to an environment place.

**On Strategy-Equivalence** Strategy Equivalence is defined by a relation $\approx_\mathfrak{B}$ between the markings in branching process of the strategy and plays compatible with the controller. It turns out that we can provide an even stronger statement than required by strategy-equivalence. Instead of relating markings in a branching process and controller compatible plays for fixed strategy and controller, we relate markings in the unfolding of the Petri game and all plays in the control game. Since every branching process can be seen as a subprocess of the unfolding and controller compatible plays are a subset of all possible plays, our relation extends naturally to a concrete strategy and controller.

The relation $\approx_\mathfrak{B}$ should intuitively relate markings and plays that represent a "similar situation". In these situations a strategy and controller should act identical, i.e., allow the same

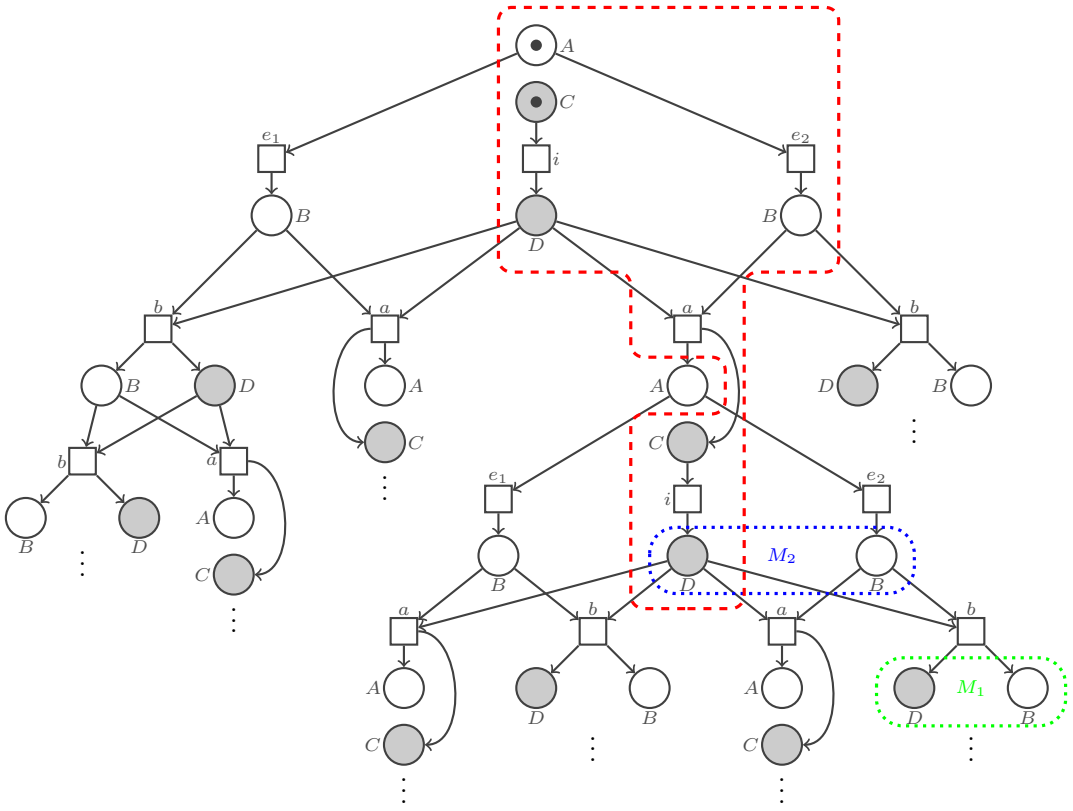



**Figure 5.4** An initial fragment of the unfolding of the Petri game $\dot{\mathcal{G}}$ in Fig. 5.3. The gray label is the one given by $\lambda$. Two markings $M_1$ and $M_2$ are surrounded in blue and orange. The causal past of the system place in $M_2$ is surrounded in red.

moves. We define $M \approx_{\mathfrak{B}} u$ if $M$ describes exactly the situation that results from the observable actions (non $\tau$-actions) in $u$. That is, $M$ is the marking reached in the unfolding by firing the observable actions from $u$.

As an example, we again consider the translation for Petri game $\dot{\mathcal{G}}$ depicted in Fig. 5.3. Fig. 5.4 shows an initial fragment of the unfolding of $\dot{\mathcal{G}}$. Every strategy for this game can be considered a subprocess of this unfolding. Each marking in the unfolding characterizes the precise situation of the game, i.e., what transitions have fired in what order. Every play in our translated control game $\dot{\mathcal{C}}_{\mathcal{G}}$ corresponds to a marking in the obvious way. As an example, consider the play

$$u_1 = [\, e_2 \,,\, \tau_{(C,\{i\})} \,,\, i \,,\, \tau_{(D,\{a,b\})} \,,\, a \,,\, e_2 \,,\, \tau_{(C,\{i\})} \,,\, i \,,\, \tau_{(D,\{a,b\})} \,,\, b \,,\, \tau_{(D,\{a,b\})} \,]_{\mathbb{I}}$$

The observable actions in $u_2$ are $e_2, i, a, e_2, i$ and $b$. $u_1$ naturally corresponds to the orange marking $M_1$ in Fig. 5.4. Both $M_1$ and $u_1$ describe the "same situation": The environment has played $e_2$ twice and communication occurred on $a$ and then on $b$. In fact firing the observable actions of $u_1$ in the unfolding results in $M_1$, so $M_1 \approx_{\mathfrak{B}} u_1$.

We can make a few observation regarding our definition. Firstly, if we have any play in $\mathcal{C}_{\mathcal{G}}$ then the observable actions in that play form a valid sequence in the unfolding of $\mathcal{G}$. For every play there hence exists a related marking. The simulation is, furthermore, invariant under elements in the trace-equivalent class: If two actions in a play are independent they can be fired in the unfolding in any order without changing the reached marking. Secondly, we observe that, by construction of $\mathcal{C}_{\mathcal{G}}$, in related situation the game is in equally labelled states (as long as we identify states that represent a commitment set with the underlying local state). We can see

both observations in our example: Firing the example $u_1$ from above results in marking $M_1$ no matter what concrete linearisation of $u_1$ is chosen. It holds that $state(u_1) = \langle B, (D, \{a, b\})\rangle$ which, if we ignore the commitment sets, is identical to the label of the places in $M_1$.

Having described $\approx_{\mathfrak{B}}$ we still need to argue that a strategy and a controller can actually achieve bisimilar behavior from related situations. Since by $\approx_{\mathfrak{B}}$-related situations are equally labelled our construction guarantees that both a strategy and a controller have identical control possibilities. That is if $M \approx_{\mathfrak{B}} u$ then every place in $M$ can control behavior (is on a system place) if and only if the process he corresponds to can control any behavior on $u$ (can choose commitment sets). Strategy and controller do, however, act on partial information. To show that bisimilar behavior is achievable we hence need to take the local information of each player into account. We can show that the information of each player aligns in $\approx_{\mathfrak{B}}$-related situations, i.e., if $M \approx_{\mathfrak{B}} u$ then every token in $M$ possess the "same information" as the process counterpart does on $u$. This is due to the underlying structure of $\mathcal{C}_{\mathcal{G}}$. Every observable action in this control game involves exactly the processes whose slices are involved in the underlying transition. In both game types each communication exchanges the entire causal history. Due to the communication behavior of $\mathcal{G}$ being reproduced truthfully in $\mathcal{C}_{\mathcal{G}}$, a process always posses comparable information to its counterpart place in $M$.

As an illustrative example, we consider the blue marking $M_2$ in Fig. 5.4 and a $\approx_{\mathfrak{B}}$-related play $u_2$ with

$$u_2 = [\ e_2\ ,\ \tau_{(C,\{i\})}\ ,\ i\ ,\ \tau_{(D,\{a,b\})}\ ,\ a\ ,\ e_2\ ,\ \tau_{(C,\{i\})}\ ,\ i\ ,\ \tau_{(D,\{b\})}\ ]_{\mathbb{I}}$$

We can compare the local information of process $p_2$ and the corresponding system player in $M_2$, i.e., the player on the place in $M_2 \cap \lambda^{-1}[S_{p_2}]$. We can not compare information formally yet but can argue informally with what information a player can base its decision on. While $p_2$ can deduce from its local view on $u_2$ that the environment chose $e_2$ the first time, it cannot tell the most recent decision of the environment. His local view does not include the trailing $e_2$ that has already been played. A possible controller can hence not base its decision (allow $a$, $b$, both or none) on the fact that $p_1$ already played $e_2$ a second time. Similarly justified refusal forbids the place in $M \cap \lambda^{-1}[S_{p_2}]$ to base its decision (i.e., enable $a$, $b$, both or none) on whether $e_1$ or $e_2$ occurred. The decision of the place can, however, be based on the knowledge that the environment chose $e_2$ the first time as this information is encoded inn the place itself. In both $M_2$ and $u_2$ the player hence possess a comparable amount of information: Both players cannot base their decision on the most recent move by the environment, whereas they can take the first environment decision, to play $e_2$, into account.

**Translating a Strategy for $\mathcal{G}$ to a Controller for $\mathcal{C}_{\mathcal{G}}$** Given a winning strategy $\sigma$ for $\mathcal{G}$ we construct a winning controller $\varrho_\sigma$ for $\mathcal{C}_{\mathcal{G}}$. The only states in $\mathcal{C}_{\mathcal{G}}$ from which a process $p$ can control any behavior (in terms of controllable actions) are of the form $q \in \mathcal{P}_{\mathcal{S}}$. If in such a state it can choose a commitment set using local $\tau$-actions. As we conceptually build our controller to achieve the same behavior as $\sigma$, the decision what commitment set to choose should be made in accordance with the strategy. $p$ wants to allow exactly those observable actions that $\sigma$ allowed in a similar situation. In order to do so, $p$ simulates the observable transitions in its local view in the branching process of $\sigma$. In the resulting marking there is a place $q'$ that belongs to $p$, i.e., a place of the slice that $p$ is build from. The transitions allowed from that place are $post^{\mathcal{N}^\sigma}(q')$. Process $p$ now copies this decision by choosing an appropriate commitment set, i.e., allow $\tau_{(q,\lambda[post^{\mathcal{N}^\sigma}(q')])}$ and forbid all other controllable actions.
For the bisimulation we need to show that if $M \approx_{\mathfrak{B}} u$ then $\sigma$ and $\varrho_\sigma$ allow the same behavior. Note that definition of $\approx_{\mathfrak{B}}$ and the construction of $\varrho_\sigma$ are completely independent. From $M \approx_{\mathfrak{B}} u$ conclude that firing the observable actions from $u$ results in $M$. To show bisimilarity we want to show that $p$ copies the decision made in $M$. Because $p$ does not posses the entire $u$ but only a local view on it ($view_p(u)$) simulation of the local view (as in the definition of $\varrho_\sigma$) results in a marking $M'$ that is different from the one reached when firing the observable actions

in $u$ (which is precisely $M$). One can, however, prove that the place that belongs to process $p$ is identical in $M$ and $M'$. Even though $p$ acts on partial information, it can replicate the decision made by the corresponding place in $M$. Since every process copies the decision of one place in $M$, *together* all processes hence allow the same behavior as possible in $M$.
The $\lightning$-actions in $\widehat{\mathcal{C}_\mathcal{G}}$ forbid nondeterminism as they can only fire if a commitment set has been chosen such that two actions from that set are possible. If we build our controller from a deterministic strategy the commitment sets are chosen in accordance with the strategy. In a deterministic strategy no two transitions are enabled from the same system place, so no two actions are possible from the same commitment set. In the translated controller the $\lightning$-action are hence never possible.

**Translating a Controller for $\mathcal{C}_\mathcal{G}$ to a Strategy for $\mathcal{G}$** Given a winning controller $\varrho$ we incrementally build a strategy $\sigma_\varrho$. We start by adding the initial marking and add the correct $\lambda$-labels. Every system place $q$ in a partially constructed strategy can decide what transitions to allow. To, in the end, be bisimilar $q$ should copy the decision of $\varrho$ made in a similar situations. The information local to $q$ is its causal past. We can take the transitions in that past and add $\tau$-actions to obtain a play in $\mathcal{C}_\mathcal{G}$. Place $q$ can hence translate its causal information (in terms of its causal past) to a play in $\mathcal{C}$. If this play is given to $\varrho$ the process that correspond to $q$ will choose a commitment set. We define the set of transitions allowed from $q$ to be exactly to be all transition in that commitment set. We then extend the strategy by adding all transitions where all involved system places have agreed on.
If $M \approx_\mathfrak{B} u$ we again need to show that $\varrho$ and $\sigma_\varrho$ allow equal behavior. One can show that if $M \approx_\mathfrak{B} u$ then the causal past of any place $q$ in $M$ with the added $\tau$-actions agrees with $\mathit{view}_p(u)$ for the corresponding process $p$. By translating its causal past to a play any place hence copies the local view of its corresponding process on $u$ and therefore duplicates the decision made by $p$. Every place in $M$ hence copies the decision of one process in $\mathcal{C}_\mathcal{G}$ on $u$. *Together* the places in $M$ therefore achieve the same behavior as $\varrho$.
A winning controller for $\widehat{\mathcal{C}_\mathcal{G}}$, furthermore, avoids all uncontrollable $\lightning$-actions, so at any point no two actions from a chosen commitment set are enabled. The constructed strategy copies the commitment sets and is therefore deterministic.

### 5.3.2 Proving Strategy Equivalence

Following this informal description we give a formal construction, including proofs establishing Theorem 1.

For convenience we abuse the notation: The transitions in a branching process of a Petri game are not the ones from the game but are merely equipped with a $\lambda$-label to them. Writing $M \,[\, t \,\rangle\, M'$ for some marking $M$ in a branching process and transitions $t$ in the underlying game is therefore not defined. Unless for very specific occasions we are, however, not interested in the precise transition in a branching process but solely on the label of it. In the proofs an notion defined below we hence always identify transitions in the branching process (strategy) with the original ones. $M \,[\, t \,\rangle\, M'$ should therefore be understood as: There is a transitions $t'$ in the branching process with $M \,[\, t' \,\rangle\, M'$ and $\lambda(t') = t$. Using this notational shortcut we could, for instance, write $\mathcal{N}^\mathfrak{U}[\triangledown\, \kappa\,]$ for a sequence $\kappa$ of transitions in the underlying game. It should be noted that this notional shortcut is not well defined for arbitrary Petri nets; there could be multiple equally labelled transitions enabled from the same marking in the branching process. For branching processes of safe games (and therefore of sliceable games), however, there is at most one transition with an matching $\lambda$-label enabled.

**On the relation $\approx_{\mathfrak{B}}$** Any state in $\bigcup_{\varsigma \in \boldsymbol{S}} Q_\varsigma \setminus \{\bot_\varsigma\}$ corresponds to a place in $\mathcal{G}$ in the natural way. This correspondence is formalized by $\zeta$ where:

$$\zeta(q) = q$$
$$\zeta(\,(q, A)\,) = q$$

We extend $\zeta$ to global states by defining for each global state $\{q_p\}_{p \in \boldsymbol{P}}$ a corresponding marking by: $\zeta(\{q_p\}_{p \in \boldsymbol{P}}) = \bigcup_{p \in \boldsymbol{P}} \{\zeta(q_p)\}$. For a process $p$ we define the shortcut $\boldsymbol{S}(p)$ for the slice that $p$ has been build from[5]. Conversely, for a slice $\varsigma$, $\boldsymbol{P}(\varsigma)$ denotes the process that is build from $\varsigma$. By definition of $\mathcal{C}_\mathcal{G}$ we have that $\mathcal{T} \subseteq \Sigma$. For a sequence of actions $u \in \Sigma^*$ we denote the projection on $\mathcal{T}$ by $\langle u \rangle_\downarrow^\mathcal{T}$. It is defined by:

$$\langle \epsilon \rangle_\downarrow^\mathcal{T} = \epsilon$$
$$\langle u\, \tau \rangle_\downarrow^\mathcal{T} = \langle u \rangle_\downarrow^\mathcal{T}$$
$$\langle u\, t \rangle_\downarrow^\mathcal{T} = \langle u \rangle_\downarrow^\mathcal{T}\, t \quad \text{if } t \in \mathcal{T}$$

We can now formalize our initial idea of the relation $\approx_{\mathfrak{B}} \subseteq \mathcal{R}(\mathcal{G}^\mathfrak{U}) \times \mathit{Plays}(\mathcal{C}_\mathcal{G})$ by defining:

$$M \approx_{\mathfrak{B}} u \text{ iff } \mathcal{G}^\mathfrak{U}[\triangledown \langle u \rangle_\downarrow^\mathcal{T}\,] = M$$

This captures the idea that a marking and play are similar/related if they are reached with the same observable trace. $\mathcal{G}^\mathfrak{U}[\triangledown \langle u \rangle_\downarrow^\mathcal{T}\,]$ should be understood as firing any linearisation of $\langle u \rangle_\downarrow^\mathcal{T}$. We hence need to prove that $\approx_{\mathfrak{B}}$ is *well defined*, i.e., $\mathcal{G}^\mathfrak{U}[\triangledown \langle u \rangle_\downarrow^\mathcal{T}\,]$ is invariant elements of the equivalence class $u$. Since the actions in $\mathcal{C}_\mathcal{G}$ are constructed from transitions they inherit the dependency from the transition. If two actions are independent the corresponding transitions are concurrent in the Petri net and can be executed in any order:

**Lemma 1** If $\mathcal{G}^\mathfrak{U}[\triangledown \langle u \rangle_\downarrow^\mathcal{T}\,] = M$ for $u \in \Sigma^*$ and $u \sim_\mathbb{I} w$ for some $w \in \Sigma^*$ then $\mathcal{G}^\mathfrak{U}[\triangledown \langle w \rangle_\downarrow^\mathcal{T}\,] = M$

**Proof** If actions $t_1, t_2 \in \mathcal{T}$ are independent in $\mathcal{C}_\mathcal{G}$ they belong to different slices (by definition of the dependency relation), so $(\mathit{pre}^\mathcal{G}(t_1) \cup \mathit{post}^\mathcal{G}(t_1)) \cap (\mathit{pre}^\mathcal{G}(t_2) \cup \mathit{post}^\mathcal{G}(t_2)) = \emptyset$. Swapping $t_1$ and $t_2$ hence results in the same marking in the unfolding $\mathcal{G}^\mathfrak{U}$. The claim follows by induction on the number of swaps in the proof of $u \sim_\mathbb{I} w$. ■

In our construction every place in the Petri game is represented as possibly many states in the control games. These additional copies, used to represent commitment sets, are equipped with the same $\zeta$ label. Every observable action $t$ in the control game precisely captures the movement of the tokens involved in $t$. We hence see that for a related marking and play the underlying net/automaton is in an equally labelled state:

**Lemma 2** If $M \approx_{\mathfrak{B}} u$ then $\zeta(\mathit{state}(u)) = \lambda[M]$.

**Proof** Follows by induction on the length of $u$ using the fact that for all $t \in \mathcal{T}$ and all $B \in \mathit{domain}(\delta_t)$ it holds that $\zeta(B) = \mathit{pre}^\mathcal{G}(t)$ and $\zeta(\delta_t(B)) = \mathit{post}^\mathcal{G}(t)$. ■

**Causal Information Flow** In our construction we represent each slice as a distinct process. The actions of a process $p$ ($\Sigma_p$) are precisely the transitions that $\boldsymbol{S}(p)$ is involved in (and additional $\tau$-actions). Now consider a marking $M$ and play $u$ where $M \approx_{\mathfrak{B}} u$. By construction firing the observable action from $u$ in the unfolding results in $M$. The marking $M$ and trace $u$ do not only represent the global state of the system but also include the information local of each token or process. The crucial observation of our translation is that this information is "the same". The

[5] In the construction $p$ is exactly this slice. However, having explicit notion makes the proofs more understandable.

local view of process $p$ on $u$ is the same as the causal past of the token in $M$ from slice $\boldsymbol{S}(p)$. We have already observed this phenomenon with the example from Fig. 5.4. This holds as in $\mathcal{C}_{\mathcal{G}}$ the communication behavior of $\mathcal{G}$ is modelled truthfully. Every process hence participates in exactly the actions that its slice takes part in. For our translation we need a more formal notion of what "having the same information" means. We thus need to find a way to relate causal information between both games. Unfortunately Petri games and control games represent causal information in a fundamentally different way utilizing either the causal past of a place or the local view on a play. We hence need to come up with an efficient intermediate representations that allow for comparison and transfer of causal memory.

**Partially Ordered Sets** To this extent we can use the poset representation established for both game types. Recall that a poset is a pair $(\mathcal{X}, \leq)$ where $\leq$ is a partial order on elements from $\mathcal{X}$. We introduce a labelled partially ordered set as a triple $(\mathcal{X}, \leq, \beta)$ where $(\mathcal{X}, \leq)$ is a poset and $\beta : \mathcal{X} \to \mathcal{Y}$ labels the elements from $\mathcal{X}$ in some set $\mathcal{Y}$. Two posets $(\mathcal{X}_1, \leq_1)$ and $(\mathcal{X}_2, \leq_2)$ are *isomorphic* if there is a bijection $g$ between $\mathcal{X}_1$ and $\mathcal{X}_2$ such that for all $x, y \in \mathcal{X}_1 : x \leq_1 y \Leftrightarrow g(x) \leq_2 g(y)$. In the literature such a function is refereed to as an *order isomorphism*. Two labelled posets $(\mathcal{X}_1, \leq_1, \beta_1)$ and $(\mathcal{X}_2, \leq_2, \beta_2)$ that are labelled in the same set $\mathcal{Y}$ are isomorphic if there exists an order isomorphism $g$ between $\mathcal{X}_1$ and $\mathcal{X}_2$ where $\forall x \in \mathcal{X}_1 : \beta_1(x) = \beta_2(g(x))$, i.e., the label agrees. We call two posets equal and write "=" between them if they are isomorphic[6]. We have seen that both the causal past of a place and a trace have intuitive poset characterisation:

- For the causal past of place $q$ in an branching process, we can define the labelled poset $(\mathit{past}_{\mathcal{T}}(q), \leq, \lambda)$ where $\leq$ is the causal dependency relation and $\lambda$ the homomorphism associated to each branching process[7].
- For a trace $u$ we can define the labelled poset $(\mathit{Pre}^{\mathit{prime}}(u), \sqsubseteq, \mathit{last})$ where $\mathit{Pre}^{\mathit{prime}}(u)$ are all primed prefixes of $u$, $\sqsubseteq$ is the prefix relation and *last* labels each prefix with its last action.

For both a play and the causal past the poset representation can be seen as a relaxation of the sequence of transitions that lead to a place $q$ or have been played in $u$, by leaving concurrent executions unordered. The poset hence hence describes the concurrent execution leading to this place. If we consider a play $u$ and the poset representation of $\langle u \rangle_{\downarrow}^{\mathcal{T}}$ we observe that the poset is labelled in $\mathcal{T}$. For any place $q$ in the unfolding of a $\mathcal{G}$, the poset is also labelled in $\mathcal{T}$. This allows us to express equality between the causal past of a pace and a trace. We can, for instance, write $\mathit{past}_{\mathcal{T}}(q) = \langle u \rangle_{\downarrow}^{\mathcal{T}}$, which should be understood as the fact that both sides have equal poset representations.

As an example, we consider the play $[e_2, \tau_{(C,\{i\})}, i, \tau_{(D,\{a,b\})}, a, \tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}]_{\mathbb{I}}$ whose poset representation is depicted in Fig. 5.5. Let $q_2$ be the system place in marking $M_2$ in Fig. 5.4. The causal past of this place is surrounded in red. The transitions in the causal past of $q_2$ form a poset in the natural way. Using our new formalism we can now state $\mathit{past}_{\mathcal{T}}(q_2) = \langle u_2 \rangle_{\downarrow}^{\mathcal{T}}$. The attentive reader is encouraged to comprehend why this equality holds.

We can now state the following result which gives us a direct characterisation of the local information of individual player. It tells us that in $\approx_{\mathfrak{B}}$-related situations, the local view of each process aligns with the causal past of the corresponding place.

[6]Throughout this thesis we never explicitly work with the order isomorphism between two posets.
[7]This is one of the rare occasion where we explicitly work with transitions in a branching process.

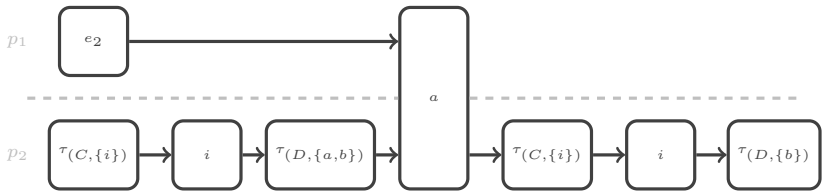



**Figure 5.5** The poset representation of $[e_2, \tau_{(C,\{i\})}, i, \tau_{(D,\{a,b\})}, a, \tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}]_{\mathbb{I}}$ in the control game $\mathcal{C}_{\mathcal{G}}$ from Fig. 5.3 (c).

**Lemma 3** If $M \approx_{\mathfrak{B}} u$ and $q \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$ (for some $p \in \boldsymbol{P}$) then $past_{\mathcal{T}}(q) = \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}$

**Proof** From $M \approx_{\mathfrak{B}} u$ we conclude that $\mathcal{G}^{\mathfrak{U}}[\triangledown \langle u \rangle^{\mathcal{T}}_{\downarrow}] = M$. The simulation is invariant under elements from $u$ as we argued in Lemma 1.
The local view of $p$ on $u$ is defined as the smallest trace $[v]_{\mathbb{I}}$ such that $u \sim_{\mathbb{I}} v\, w$ for some $w$ that contains no actions from $\Sigma_p$. We can hence write $u = view_p(u)\, w$. Since the $\tau$-actions are local to one process it holds that $\langle u \rangle^{\mathcal{T}}_{\downarrow} = \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow} \langle w \rangle^{\mathcal{T}}_{\downarrow}$ and $\langle w \rangle^{\mathcal{T}}_{\downarrow}$ contains no actions from $\Sigma_p$. We hence obtain that

$$\mathcal{G}^{\mathfrak{U}}[\triangledown \langle u \rangle^{\mathcal{T}}_{\downarrow}] = \mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow} \langle w \rangle^{\mathcal{T}}_{\downarrow}]$$

and, in particular,

$$\mathcal{G}^{\mathfrak{U}}[\triangledown \langle u \rangle^{\mathcal{T}}_{\downarrow}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] = \mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow} \langle w \rangle^{\mathcal{T}}_{\downarrow}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$$

The observable actions in $\Sigma_p$ are exactly the transitions that the slice $\boldsymbol{S}(p)$ is involved in. $\langle w \rangle^{\mathcal{T}}_{\downarrow}$ contains no action from $\Sigma_p$ and therefore contains no transition that involves $\boldsymbol{S}(p)$. We can hence see that

$$\begin{aligned} M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] &= \mathcal{G}^{\mathfrak{U}}[\triangledown \langle u \rangle^{\mathcal{T}}_{\downarrow}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\ &= \mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow} \langle w \rangle^{\mathcal{T}}_{\downarrow}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\ &= \mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \end{aligned}$$

Firing $\langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}$ and firing $\langle u \rangle^{\mathcal{T}}_{\downarrow}$ results in the same place for slice $\boldsymbol{S}(p)$. We later recover exactly this statement (Lemma 4) from our current lemma.
We next show that $\mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}] = \mathcal{G}^{\mathfrak{U}}[\triangledown past_{\mathcal{T}}(q)]$, i.e., firing the transitions in the causal past of $q$ results in the same marking as firing $\langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}$. Note that by definition every linearisation of $past_{\mathcal{T}}(q)$ results in the same marking, so $\mathcal{G}^{\mathfrak{U}}[\triangledown past_{\mathcal{T}}(q)]$ is well defined. It trivially holds that $\mathcal{G}^{\mathfrak{U}}[\triangledown past_{\mathcal{T}}(q)] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] = M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$, so we get that

$$\mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] = \mathcal{G}^{\mathfrak{U}}[\triangledown past_{\mathcal{T}}(q)] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \qquad \textbf{(1)}$$

We want to show the more general statement that not only the place that belongs to process $p$ is shared in $\mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}]$ and $\mathcal{G}^{\mathfrak{U}}[\triangledown past_{\mathcal{T}}(q)]$ but the place of every process.
We can first observe that $past_{\mathcal{T}}(q)$ is the smallest set of transitions that needs to fire to reach $q$. As soon as we remove a single transition from the set, the simulation will no longer reach place $q$. From **(1)** we get that simulating $\langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}$ also results in place $q$. Simulating $\langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}$ instead of $past_{\mathcal{T}}(q)$ therefore results in a marking that has progressed more, i.e., a marking where the game has progressed further **(2)**.
We assume for contradiction that $\mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}] \neq \mathcal{G}^{\mathfrak{U}}[\triangledown past_{\mathcal{T}}(q)]$. There hence is a process $p'$ with

$$\mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u) \rangle^{\mathcal{T}}_{\downarrow}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p')}] \neq \mathcal{G}^{\mathfrak{U}}[\triangledown past_{\mathcal{T}}(q)] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p')}]$$

Let $q_1$ and $q_2$ be the unique places with

$$q_1 \in \mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u)\rangle^{\mathcal{T}}_{\downarrow}\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p')}]$$
$$q_2 \in \mathcal{G}^{\mathfrak{U}}[\triangledown past_{\mathcal{T}}(q)\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p')}]$$

By assumption $q_1 \neq q_2$ and from **(2)** it is easy to see that $q_2 < q_1$, i.e., the token of slice $\boldsymbol{S}(p')$ has progressed further when firing $\langle view_p(u)\rangle^{\mathcal{T}}_{\downarrow}$ instead of $past_{\mathcal{T}}(q)$.
Let $t$ be the unique transition in $pre^{\mathcal{G}^{\mathfrak{U}}}(q_1)$. It holds that $q_2 < t < q_1$. We know that $t$ must be included in $view_p(u)$ and since $t$ has no successor transitions we observe that $view_p(u) = r\,t$ **(3)** for some play $r$, i.e., there is a linearisation of $view_p(u)$ that ends with $t$. Since $t$ does not involve the token from slice $\boldsymbol{S}(p)$ we can conclude that $t \notin \Sigma_p$. **(3)** is, however, a contradiction to the minimality of $view_p(u)$.
Hence $\mathcal{G}^{\mathfrak{U}}[\triangledown \langle view_p(u)\rangle^{\mathcal{T}}_{\downarrow}\,] = \mathcal{G}^{\mathfrak{U}}[\triangledown past_{\mathcal{T}}(q)\,]$. If two transitions in $past_{\mathcal{T}}(q)$ are unordered they are independent in $\langle view_p(u)\rangle^{\mathcal{T}}_{\downarrow}$. Conversely, consecutive independent actions in $\langle view_p(u)\rangle^{\mathcal{T}}_{\downarrow}$ involve disjoint set of slices and are hence unordered in $past_{\mathcal{T}}(q)$. It is therefore easy to see that $past_{\mathcal{T}}(q) = \langle (view_p(u))\rangle^{\mathcal{T}}_{\downarrow}$. ■

Lemma 3 tells us that our relation $\approx_{\mathfrak{B}}$ does not only capture the global configuration of both games (as stated in Lemma 2) but also respects the local information. This is of tremendous importance for a translation of strategies/controller. If $M \approx_{\mathfrak{B}} u$ then every process in $p$ possess the same information (in terms of the local view on $u$) as the corresponding place in $M$ has (in terms of the causal past).

To see this on an example we once more consider the blue marking $M_2$ in Fig. 5.4 and play $u_2$ with

$$u_2 = [\, e_2 \,,\, \tau_{(C,\{i\})} \,,\, i \,,\, \tau_{(D,\{a,b\})} \,,\, a \,,\, e_2 \,,\, \tau_{(C,\{i\})} \,,\, i, \tau_{(D,\{b\})} \,]_{\mathbb{I}}$$

It holds that $M_2 \approx_{\mathfrak{B}} u_2$. In both situations the environment has played $e_2$ twice. However, in both, $M_2$ and $u_2$ the second execution of $e_2$ occurred after the communication of $a$, so neither $p_2$ nor the system place in $M_2$ that corresponds to $p_2$ can deduce that the environment has played $e_2$. If we compute the local view of the process $p_2$ on $u_2$, we get that

$$view_{p_2}(u_2) = [\, e_2 \,,\, \tau_{(C,\{i\})} \,,\, i \,,\, \tau_{(D,\{a,b\})} \,,\, a \,,\, \tau_{(C,\{i\})} \,,\, i \,,\, \tau_{(D,\{b\})} \,]_{\mathbb{I}}$$

The poset representation of which is depicted in Fig. 5.4. The causal past from of the system place $q_2$ in $M_2$ (the place in $M_2 \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p_2)}]$) is surround in red in Fig. 5.4. As we observed before it holds that $past_{\mathcal{T}}(q_2) = \langle (view_p(u_2))\rangle^{\mathcal{T}}_{\downarrow}$, which is exactly the result stated in Lemma 3.

### 5.3.3 Translating Strategies to Controllers

In this section we provide a formal translation of strategies to controllers. Given a winning strategy $\sigma$ for $\mathcal{G}$. We construct a winning controller $\varrho_\sigma = \{f_p^{\varrho_\sigma}\}_{p \in \boldsymbol{P}}$ for $\mathcal{C}_{\mathcal{G}}$ and, furthermore, show that if $\sigma$ is deterministic, $\varrho_\sigma$ is a winning controller for $\widehat{\mathcal{C}_{\mathcal{G}}}$. The description of the local controller $f_p^{\varrho_\sigma}$ for process $p$ is depicted in Fig. 5.6.

Every process $p$ in $\varrho_\sigma$ does what we described informally. Given a play $u \in Plays_p(\mathcal{C}_{\mathcal{G}})$ it computes its current state. Only if this state correspondences to a system place of $\mathcal{G}$ (case **3.**) any controllable actions are available. In this case the observable action in $u$ are simulated in $\mathcal{N}^\sigma$, i.e., the branching process of $\sigma$. In Lemma 1 we already argued that simulation of traces is well defined, i.e., invariant under linearisations. For an arbitrary $u$, $\langle u\rangle^{\mathcal{T}}_{\downarrow}$ might not be a valid sequence in the strategy. We therefore include case **b)** to obtain a total function $f_p^{\varrho_\sigma}$. In case of a successful simulation (case **a)**) the simulation reaches some marking $M$. $p$ should now copy the decision of the strategy made in $M$. It therefore computes the place in $M$ that corresponds

For $p \in \boldsymbol{P}$ and $u \in \mathit{Plays}_p(\mathcal{C}_\mathcal{G})$:

1. If $\mathit{state}_p(u) \in \mathcal{P}_\mathcal{E}$ all outgoing transitions are uncontrollable. Define $f_p^{\varrho_\sigma}(u) = \emptyset$.
2. If $\mathit{state}_p(u) = (q, A)$ for some $q \in \mathcal{P}_\mathcal{S}$ and $A \subseteq \mathit{post}^\mathcal{G}(q)$ all outgoing transitions are uncontrollable. Define $f_p^{\varrho_\sigma}(u) = \emptyset$.
3. If $\mathit{state}_p(u) \in \mathcal{P}_\mathcal{S}$, we distinguish two cases
   - **a)** $\langle u \rangle_\downarrow^\mathcal{T}$ is a valid sequence of transitions in $\mathcal{N}^\sigma$:
     Let $M = \mathcal{N}^\sigma[\triangledown \langle u \rangle_\downarrow^\mathcal{T}]$. There exists an unique place $q \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$.
     Define $f_p^{\varrho_\sigma}(u) = \{\, \tau_{(\mathit{state}_p(u), \lambda[A])} \,\}$ where $A = \mathit{post}^{\mathcal{N}^\sigma}(q)$.
   - **b)** $\langle u \rangle_\downarrow^\mathcal{T}$ is no valid sequence of transition $\mathcal{N}^\sigma$:
     Define $f_p^{\varrho_\sigma}(u) = \emptyset$.
     *This case will never occur if $u$ is a controller compatible play.*
4. If $\mathit{state}_p(u) = \bot$ there are no outgoing transitions. Define $f_p^{\varrho_\sigma}(u) = \emptyset$.

**Figure 5.6** Description of local controller $f_p^{\varrho_\sigma}$ for process $p \in \boldsymbol{P}$. The controller is build from a strategy $\sigma$ for $\mathcal{G}$ with branching process $\mathcal{N}^\sigma$.

to the slice $p$ is build from, i.e., the place $q \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$. The set of transitions allowed by this place are $\mathit{post}^{\mathcal{N}^\sigma}(q)$. To copy $p$ hence chooses the commitment set that contains exactly those transitions. We later show that for controller compatibles plays $u$, $\langle u \rangle_\downarrow^\mathcal{T}$ is always a valid sequence, i.e., we never land in situation **b)**.

As an example, we consider the translation from Fig. 5.3 and the winning (non-deterministic) strategy $\dot{\sigma}$ depicted in Fig. 5.7. Whenever possible $\dot{\sigma}$ allows transition $i$ to move to place $D$. If in place $D$ the first time it allows communication on both $a$ and $b$. Upon communication on either $a$ or $b$ the strategy can, furthermore, deduce whether the environment played $e_1$ or $e_2$, as this information is conceptually transmitted in the communication. There are hence four different cases possible: In case of synchronization on $a$, $\dot{\sigma}$ allows the token on a system place to move to $D$ using transition $i$. It then distinguishes whether $e_1$ or $e_2$ have been played. In case of $e_1$ it terminates and in case of $e_2$ it allows communication on $b$ for one more time. If synchronization occurred on $b$ the strategy again distinguishes the two cases. If it can deduce $e_1$ it allows $b$ for one more time. In case of $e_2$ it terminates directly. Even though $\dot{\sigma}$ seems unnecessary complicated[8], strategy-equivalence requires us to build a controller that copies this behavior. We can now translate $\dot{\sigma}$ according to our translation of strategies and obtain a controller $\dot{\varrho}_\sigma$ for $\dot{\mathcal{C}}_\mathcal{G}$. Since there is no intuitive way to represent a controller graphically, we depict $\dot{\varrho}_\sigma$ as a table that summarizes a selection of plays in $\dot{\mathcal{C}}_\mathcal{G}$ and the decision made by $p_2$ (the local controller $f_{p_2}^{\dot{\varrho}_\sigma}$). The table is shown in Fig. 5.8. As we only depict the decision of $p_2$, we listed the $p_2$-view on all plays. $\dot{\varrho}_\sigma$ initially allows the $i$ action by choosing the $(C, \{i\})$ commitment set. Afterwards, it admits communication on both $a$ and $b$ by moving to the $(D, \{a, b\})$ commitment set. $\dot{\varrho}_\sigma$ copies the "cases analysis" of $\dot{\sigma}$. We can observe that every decision of the controller is made in accordance with our construction. As an example, consider the play $[\tau_{(C,\{i\})}, i, \tau_{(D,\{a,b\})}, e_1, b]_\mathbb{I}$ (play **(1)** in Fig. 5.8). The observable action on that play comprise $e_1, i$ and $b$. The simulation of this play in $\mathcal{N}^{\dot{\sigma}}$ results in the orange marking $M_1$. Since the system place (the place in $M_1 \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p_2)}]$) allows $b$ in its postcondition, the controller chooses $(D, \{b\})$ as a commitment set. The interested reader is advised to convince herself that all decision listed in Fig. 5.8 are in accordance with both our construction and $\dot{\sigma}$.

[8] In the sense that there are much simpler winning strategies.

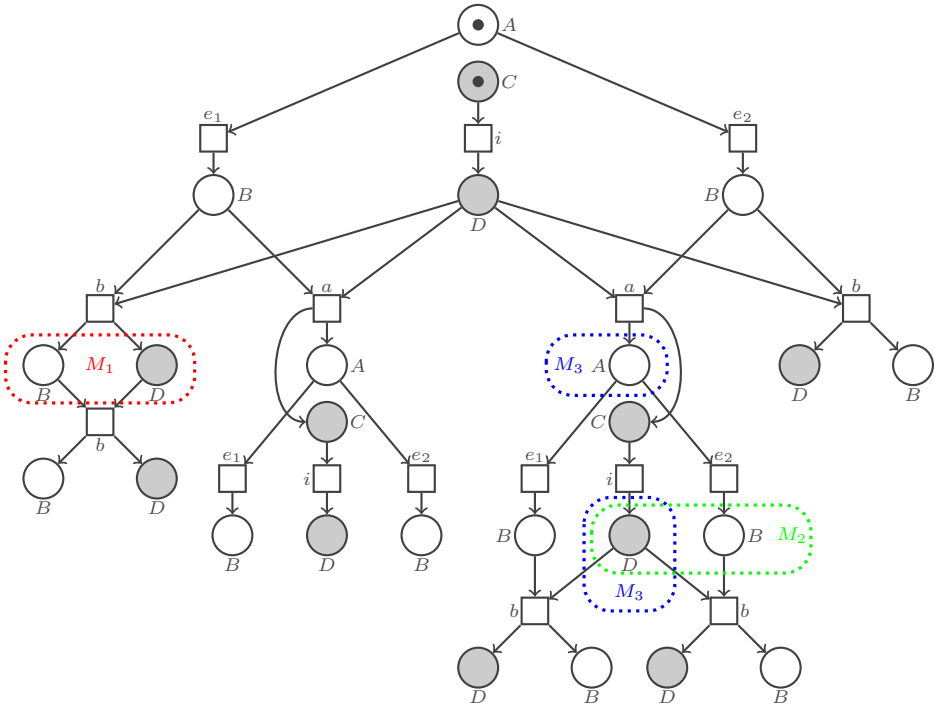



**Figure 5.7** A winning strategy $\dot{\sigma}$ for the Petri game $\dot{\mathcal{G}}$ in Fig. 5.3. The marking of winning places has been omitted. After first allowing both $a$ and $b$ the strategy makes a case distinction on which commination of $e_1$ ot $e_2$ and $a$ or $b$ occurred. We can view this strategy as a subprocess of the unfolding (depicted in Fig. 5.4). Reachable marking $M_1$, $M_2$ and $M_3$ are surrounded in red, green and blue.

**(1)**

| $u \in \mathit{Plays}(\dot{\mathcal{C}_\mathcal{G}}, \dot{\varrho}_\sigma) \cap \mathit{Plays}_{p_2}(\dot{\mathcal{C}_\mathcal{G}})$ | $f_{p_2}^{\dot{\varrho}_\sigma}(u)$ |
|---|---|
| $\epsilon$ | $\{\tau_{(C,\{i\})}\}$ |
| $\tau_{(C,\{i\})}$ | $\emptyset$ |
| $\tau_{(C,\{i\})}, i$ | $\{\tau_{(D,\{a,b\})}\}$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{a,b\})}$ | $\emptyset$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{a,b\})}, e_1, a$ | $\{\tau_{(C,\{i\})}\}$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{a,b\})}, e_1, b$ | $\{\tau_{(D,\{b\})}\}$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{a,b\})}, e_2, a$ | $\{\tau_{(C,\{i\})}\}$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{a,b\})}, e_2, b$ | $\emptyset$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{a,b\})}, e_2, a, \tau_{(C,\{i\})}, i$ | $\{\tau_{(D,\{b\})}\}$ |
| ... | |

**Figure 5.8** Controller $\dot{\varrho}_\sigma$ build from the winning strategy $\dot{\sigma}$ in Fig. 5.7. The controller is depicted by listing possible plays and the decision of $f_{p_2}^{\dot{\varrho}_\sigma}$ on them.

## Strategy-Equivalence

Given the constructed $\varrho_\sigma$ we can prove it strategy-equivalent to $\sigma$. For our bisimulation $\approx_\mathfrak{B}$ we use the one we already defined, but restrict it to reachable markings in $\mathcal{N}^\sigma$ and plays in $\mathit{Plays}(\mathcal{C}_\mathcal{G}, \varrho_\sigma)$. The previous statements (Lemma 2 and Lemma 3) extend to this restricted

relation. We begin by showing a direct consequence of Lemma 3:

**Lemma 4** If $M \approx_{\mathfrak{B}} u$ and $p \in \boldsymbol{P}$ then

$$\begin{aligned} M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] &= \mathcal{G}^{\mathfrak{U}}[\triangledown \langle u \rangle_{\downarrow}^{\mathcal{T}}\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\ &= \mathcal{G}^{\mathfrak{U}}[\triangledown \langle \mathit{view}_p(u) \rangle_{\downarrow}^{\mathcal{T}}\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \end{aligned}$$

**Proof** Let $q$ be the unique place with $q \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$. It holds that $M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] = \mathcal{G}^{\mathfrak{U}}[\triangledown \mathit{past}_{\mathcal{T}}(q)\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$ since firing the transitions in the past of $q$ is always sufficient to reach $q$. Note that writing down $\mathcal{G}^{\mathfrak{U}}[\triangledown \mathit{past}_{\mathcal{T}}(q)\,]$ is well defined. By Lemma 3 it holds that $\mathit{past}_{\mathcal{T}}(q) = \langle \mathit{view}_p(u) \rangle_{\downarrow}^{\mathcal{T}}$. We know conclude that

$$\begin{aligned} M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] &= \mathcal{G}^{\mathfrak{U}}[\triangledown \mathit{past}_{\mathcal{T}}(q)\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\ &= \mathcal{G}^{\mathfrak{U}}[\triangledown \langle \mathit{view}_p(u) \rangle_{\downarrow}^{\mathcal{T}}\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \end{aligned}$$

■

Our definition of $\varrho_\sigma$ is completely independent from the definition of $\approx_{\mathfrak{B}}$. Lemma 4, however, establishes an important relation between them. Suppose $u$ is the global play in $\mathcal{C}_{\mathcal{G}}$ and $M$ a marking such that $M \approx_{\mathfrak{B}} u$. From the definition of $\approx_{\mathfrak{B}}$ we know that $\mathcal{N}^{\sigma}[\triangledown \langle u \rangle_{\downarrow}^{\mathcal{T}}\,] = M$. Since $\mathit{view}_p(u)$ differs (in general) from $u$, simulating $\mathit{view}_p(u)$ instead of $u$ results in a different marking $M'$. Lemma 4 now states that for process $p$, the place that belongs to $\boldsymbol{S}(p)$ is identical in $M$ and $M'$. This establishes a connection to our controller definition as, in $\varrho_\sigma$, each process simulates its local view and copies the decision on the resulting marking. By Lemma 4 in related situations every process therefore copies the decision of one of the places in $M$.

We can see this at our example strategy $\dot{\sigma}$ and translated controller $\dot{\varrho}_\sigma$. Here the controller compatible play

$$u_2 = [\, e_2 \,,\, \tau_{(C,\{i\})} \,,\, i \,,\, \tau_{(D,\{a,b\})} \,,\, a \,,\, e_2 \,,\, \tau_{(C,\{i\})} \,,\, i, \tau_{(D,\{b\})} \,]_{\mathbb{I}}$$

and the green marking $M_2$ in Fig. 5.7 are related by $\approx_{\mathfrak{B}}$. When simulating the local view of $p_2$ on $u_2$ in $\mathcal{N}^{\dot{\sigma}}$ we reach the blue marking $M_3$, which is, as we observed earlier, different from $M_2$. The system place labelled $D$ (the place that belongs to $p_2$) is, however, shared in $M_2$ and $M_3$ (as stated in Lemma 4). In our construction of a controller $p_2$ would simulate its local view on $u_2$ and copy the decision in the resulting marking. It thereby copies the decision made in $M_2$, even though it computed the different $M_3$.

Lemma 4 allows us to show that the defined $\varrho_\sigma$ actually enable the same behavior if $M \approx_{\mathfrak{B}} u$. Essentially it allows us to conclude that in $\approx_{\mathfrak{B}}$-related situation $\varrho_\sigma$ copies $\sigma$. We can reason in both direction:

- If $u\,t \in \mathit{Plays}(\mathcal{C}, \varrho_\sigma)$ then all involved processes allowed $t$. So every process $p \in \mathit{dom}(t)$ either resides an environment place (a state corresponding to an environment place) where it has no control or it is on a system place where it must have chosen a commitment set where $t$ is included. $p$ chose its commitment set by simulating its local view on $u$ in the branching process of $\sigma$. By Lemma 4 it thereby copied the decision of a system place in $M$ (the system place $q \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$). As $t$ is in the commitment set of every process involved in $t$, it must be in the postcondition of every of every system place involved in $t$ so $t$ is enabled in $M$.
- If, on the other hand, the strategy allows a transition $t$ from $M$, all system places must have agreed, i.e., included $t$ in their postcondition. In $\varrho_\sigma$ each process decided on what to allow as a commitment set by simulating its local view and, by Lemma 4, therefore copies the decision of one system place in $M$. As $t$ is included in the postcondition of all involved places, every process involved in $t$ thus chooses a commitment set where $t$ is

included. We can hence see that $t$ is an extension of $u$ (after playing sufficiently many $\tau$-actions to choose a commitment set).

$\varrho_\sigma$ is a controller for both $\mathcal{C}_\mathcal{G}$ and $\widehat{\mathcal{C}_\mathcal{G}}$. To prove that $\sigma$ and $\varrho_\sigma$ are bisimilar we can treat $\mathcal{C}_\mathcal{G}$ and $\widehat{\mathcal{C}_\mathcal{G}}$ as the same, i.e., ignoring all $\lightning$-actions in $\widehat{\mathcal{C}_\mathcal{G}}$. We later show that, if $\sigma$ is deterministic, $\lightning$-actions are never part in any play compatible with $\varrho_\sigma$ and can hence be neglect for bisimulation.

**Lemma 5** If $M \approx_\mathfrak{B} u$ and $u' = u\,t \in \mathit{Plays}(\mathcal{C}_\mathcal{G}, \varrho_\sigma)$ then there exists a $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ with $M \,[\,t\,\rangle\, M'$ and $M' \approx_\mathfrak{B} u'$.

**Intuition** If $u' = u\,t \in \mathit{Plays}(\mathcal{C}_\mathcal{G}, \varrho_\sigma)$ then $t$ must be included in the chosen commitment set of every $p \in \mathit{dom}(t)$ that resides on a state that corresponds to a system place. By definition of $\varrho_\sigma$ every process chose the commitment set as the set of transitions leaving the corresponding place that results from simulation the local view in the branching process. By Lemma 4 every process thereby copies the decision of one system place in $M$. Since $t$ is in every commitment set, every system place in $M$ involved in $t$ has allowed $t$, so $t$ is possible from $M$.

**Proof** Since $M \approx_\mathfrak{B} u$, Lemma 2 allows us to conclude that $\zeta(\mathit{state}(u)) = \lambda[M]$.
We want to show that $t$ is enabled in $M$. This would imply that $M \,[\,t\,\rangle\, M'$ and $M' \approx_\mathfrak{B} u$ is a trivial consequence. From $\zeta(\mathit{state}(u)) = \lambda[M]$ and since $t$ is possible from $\mathit{state}(u)$, the construction of $\mathcal{C}_\mathcal{G}$ allows us to conclude that $t$ is enabled in $\lambda[M]$. There hence is a set $C \subseteq M$ with $\lambda[C] = \mathit{pre}^\mathcal{G}(t)$.
We assume for contraction $t$ is not allowed by the strategy. Because of justified refusal there hence is a *system* place $q \in C$ with $t \notin \lambda[\mathit{post}^{\mathcal{N}^\sigma}(q)]$ **(1)**. Place $q$ belongs to some process $p$, i.e., $q \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$. We know that $\lambda(q) = \zeta(\mathit{state}_p(u))$. By construction of $\mathit{dom}$ we know that $p \in \mathit{dom}(t)$. Since $u' = u\,t \in \mathit{Plays}(\mathcal{C}_\mathcal{G}, \varrho_\sigma)$ and $q$ is a system place we know that $\mathit{state}_p(u) = (\lambda(q), B)$ for some $B$ with $t \in B$, i.e., process $p$ has chosen a commitment set that includes $t$ **(2)**. We derive the contradiction by showing that the set of transitions leaving $q$ ($\lambda[\mathit{post}^{\mathcal{N}^\sigma}(q)]$) agrees with the decision of $p$ and must hence, by **(2)**, include $t$.
As $\mathit{state}_p(u) = (\lambda(q), B)$ there must be a $\tau_{(\lambda(q),B)}$ action in $u$, since this is the only action leading to state $(\lambda(q), B)$. Let $u_\tau \sqsubseteq u$ be the prefix obtained by removing the last such action. $u_\tau$ is a $\varrho_\sigma$ compatible play. It holds that $\mathit{state}_p(u_\tau) = \lambda(q)$. We conclude that $\tau_{(\lambda(q),B)} \in f_p^{\varrho_\sigma}(\mathit{view}_p(u_\tau))$
We can now study how $\varrho_\sigma$ chooses $B$ as its commitment set. By definition of $\varrho_\sigma$ we know that $B = \lambda[\mathit{post}^{\mathcal{N}^\sigma}(q')]$ for the unique system place $q'$ with

$$q' \in \mathcal{N}^\sigma[\triangledown\, \langle \mathit{view}_p(u_\tau)\rangle_\downarrow^\mathcal{T}\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$$

Now by Lemma 4:

$$\begin{aligned}
\{q'\} &= \mathcal{N}^\sigma[\triangledown\, \langle \mathit{view}_p(u_\tau)\rangle_\downarrow^\mathcal{T}\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\
&= \mathcal{N}^\sigma[\triangledown\, \langle \mathit{view}_p(u)\rangle_\downarrow^\mathcal{T}\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\
&= \mathcal{N}^\sigma[\triangledown\, \langle u\rangle_\downarrow^\mathcal{T}\,] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\
&= M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\
&= \{q\}
\end{aligned}$$

The system place reached by simulating $\langle \mathit{view}_p(u_\tau)\rangle_\downarrow^\mathcal{T}$ is hence exactly the system place in $M$. It follows that

$$B = \lambda[\mathit{post}^{\mathcal{N}^\sigma}(q')] = \lambda[\mathit{post}^{\mathcal{N}^\sigma}(q)]$$

This is a contradiction to $t \in B$ **(2)** but we assumed $t \notin \lambda[\mathit{post}^{\mathcal{N}^\sigma}(q)]$ **(1)**. ■

**Lemma 6** If $M \approx_{\mathfrak{B}} u$ and $u' = u\,\tau \in \mathit{Plays}(\mathcal{C}_{\mathcal{G}}, \varrho_\sigma)$ then $M \approx_{\mathfrak{B}} u'$.

**Proof** Obvious consequence from the definition of $\approx_{\mathfrak{B}}$. ■

In the proofs above we never have to cope with case **b)** in the definition of $\varrho_\sigma$: We always have to conclude statements under the assumption that $M \approx_{\mathfrak{B}} u$ for some $M$ and $u$. By definition of $\approx_{\mathfrak{B}}$, $\langle u \rangle_{\downarrow}^{\mathcal{T}}$ is a valid sequence in $\mathcal{N}^\sigma$ and therefore $\langle \mathit{view}_p(u) \rangle_{\downarrow}^{\mathcal{T}}$ as well (since $\langle \mathit{view}_p(u) \rangle_{\downarrow}^{\mathcal{T}}$ is a prefix of $\langle u \rangle_{\downarrow}^{\mathcal{T}}$). We can show the next corollary which shows that case **3b)** can be ignored for *any* $\varrho_\sigma$ compatible play[9].

**Corollary** If $u$ is a $\varrho_\sigma$ compatible play then $\langle \mathit{view}_p(u) \rangle_{\downarrow}^{\mathcal{T}}$ is a valid sequence in $\mathcal{N}^\sigma$.

**Proof** It holds that $\mathit{In}^\sigma \approx_{\mathfrak{B}} \epsilon$. By playing $u$ and using Lemma 5 and Lemma 6 we get a reachable marking $M$ in $\mathcal{N}^\sigma$ with $M \approx_{\mathfrak{B}} u$. By definition of $\approx_{\mathfrak{B}}$, $\langle u \rangle_{\downarrow}^{\mathcal{T}}$ is a valid sequence in $\mathcal{N}^\sigma$. Since $\langle \mathit{view}_p(u) \rangle_{\downarrow}^{\mathcal{T}}$ is a prefix of $\langle u \rangle_{\downarrow}^{\mathcal{T}}$ it is a valid sequence as well. ■

**Lemma 7** If $M \approx_{\mathfrak{B}} u$ and $M\ [t\rangle\ M'$ for some $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ there exists $u' = u\,\tau^*\,t \in \mathit{Plays}(\mathcal{C}_{\mathcal{G}}, \varrho_\sigma)$ with $M' \approx u'$.

**Intuition** If $M\ [t\rangle\ M'$ then every system place in $M$ that is involved in $t$ has allowed it, i.e., $t$ is in the postcondition of the place. By definition of $\varrho_\sigma$ every process that is on a state that corresponds to a system place simulates the local view on $u$. By Lemma 4 simulating the local view on $u$ results in a place in $M$ so every process involved in $t$ copies the decision of one place in $M$. Since all places in $M$ allow $t$ in their postcondition, $t$ is included in the commitment set of every process. After choosing a commitment set for all processes we can hence execute $t$.

**Proof** Since $M \approx_{\mathfrak{B}} u$ Lemma 2 allows us to conclude that $\zeta(\mathit{state}(u)) = \lambda[M]$ **(1)**. Transition $t$ is enabled in $M$ and hence for every place $q$ in $M$ with $\lambda(q) \in \mathit{pre}^{\mathcal{G}}(t)$ it holds that $t \in \lambda[\mathit{post}^{\mathcal{N}^\sigma}(q)]$ **(2)**.
Let $u_\tau$ be $u$ extended with as many $\tau$ actions as possible s.t. no $\tau$-action is possible after $u_\tau$. Since $\varrho_\sigma$ always allows a commitment set, after playing $u_\tau$ every process that can choose a commitment set, has chosen a commitment set, i.e., for every process $p$ with $\zeta(\mathit{state}_p(u_\tau)) \in \mathcal{P}_{\mathcal{S}}$ we know that $\mathit{state}_p(u_\tau) = (\_, \_)$.
Assume for contradiction $u_\tau\,t \notin \mathit{Plays}(\mathcal{C}_{\mathcal{G}}, \varrho_\sigma)$. Since **(1)** holds, every process on a system state has chosen a commitment set and action $t$ is uncontrollable, we can conclude that there is a process $p \in \mathit{dom}(t)$ with $\mathit{state}_p(u_\tau) = (q_p, B)$ but where $t \notin B$. That is $p$ has chosen a commitment set where $t$ is not included. Let $q'_p \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$ be the corresponding place in $M$. Since $p \in \mathit{dom}(t)$ and $t$ is enabled in $\lambda[M]$ we conclude that $\lambda(q'_p) \in \mathit{pre}^{\mathcal{G}}(t)$ and by **(2)** we get that $t \in \lambda[\mathit{post}^{\mathcal{N}^\sigma}(q'_p)]$ **(3)**, i.e., from the place in $M$ that corresponds to $p$, $t$ is enabled (in the postcondition).
Since $p$ is in state $(q_p, B)$ there is an $\tau_{(q_p,B)}$ action in $u_\tau$. Let $u_\tau^-$ be $u_\tau$ where the last such action is removed such that $p$ has not chosen a commitment set (i.e., $\mathit{state}_p(u_\tau) = q_p$). We can conclude that $\tau_{(q_p,B)} \in f_p(\mathit{view}_p(u_\tau^-))$.
By the definition of $\varrho_\sigma$ it holds that $B = \lambda[\mathit{post}^{\mathcal{N}^\sigma}(q''_p)]$ for the unique place $q''_p$ with

$$q''_p \in \mathcal{N}^\sigma[\triangledown \langle \mathit{view}_p(u_\tau^-) \rangle_{\downarrow}^{\mathcal{T}}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$$

[9] This is no statement required by strategy-equivalence.

Because of Lemma 4:

$$\begin{aligned}\{q_p''\} &= \mathcal{N}^\sigma[\bigtriangledown \langle view_p(u_\tau^-)\rangle_\downarrow^{\mathcal{T}}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\ &= \mathcal{N}^\sigma[\bigtriangledown \langle view_p(u_\tau)\rangle_\downarrow^{\mathcal{T}}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\ &= \mathcal{N}^\sigma[\bigtriangledown \langle u\rangle_\downarrow^{\mathcal{T}}] \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\ &= M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}] \\ &= \{q_p'\}\end{aligned}$$

We hence conclude that $q_p'' = q_p'$ and get

$$B = \lambda[post^{\mathcal{N}^\sigma}(q_p'')] = \lambda[post^{\mathcal{N}^\sigma}(q_p')]$$

The chosen commitment set, $B$, agrees with the transitions leaving $q_p'$. This is a contradiction to $t \in \lambda[post^{\mathcal{N}^\sigma}(q_p')]$ **(3)** and our assumption $t \notin B$. ■

**Corollary** $\sigma$ and $\varrho_\sigma$ are bisimilar.

**Proof** By definition $In^\sigma \approx_{\mathfrak{B}} \epsilon$. Since there are no local (unobservable) transitions in $\mathcal{G}$ the statement follows from Lemma 5, Lemma 6 and Lemma 7. ■

Having proved bisimilarity we can easily show that winningness is preserved by our translation.

**Lemma 8** If $\sigma$ is a winning strategy for $\mathcal{G}$ then $\varrho_\sigma$ is a winning controller for $\mathcal{C}_\mathcal{G}$.

**Proof** We first show that all plays in $Plays(\mathcal{C}_\mathcal{G}, \varrho_\sigma)$ are finite: Assume for contradiction there is an infinite play $u$. Due to $\mathcal{C}_\mathcal{G}$ not permitting infinite sequences of consecutive $\tau$-actions, $u$ must contain infinitely many observable actions. By bisimulation we therefore have an infinite sequence of markings $\mathcal{N}^\sigma$. This is a contradiction since $\sigma$ is by assumption winning and therefore by definition finite.
We now show that all maximal plays terminate in a winning configuration: Suppose $u \in Plays(\mathcal{C}_\mathcal{G}, \varrho_\sigma)^M$ is a maximal $\varrho_\sigma$-compatible play, i.e., cannot be extended by any action. Using our bisimulation, there exists a reachable marking $M$ in $\mathcal{N}^\sigma$ with $M \approx_{\mathfrak{B}} u$. Since $u$ is maximal, $M$ is final. Since $\sigma$ is winning $M$ must be a winning marking. Now $\zeta(state(u)) = \lambda[M]$ (by Lemma 2) and from our construction of the winning states in $\mathcal{C}_\mathcal{G}$ it follows that $state(u)$ is winning as well. ■

## Deterministic Strategies

So far we have ignored all $\lightning$-actions introduced with $\widehat{\mathcal{C}_\mathcal{G}}$. We can justify this by showing that the $\lightning$-actions can actually never be taken, if $\varrho_\sigma$ is constructed from a deterministic $\sigma$. The $\lightning$-transition can occur when the processes have choosen their commitment sets such that two transitions are enabled from the same set. By construction $\varrho_\sigma$ chooses its commitment sets in accordance with the strategy $\sigma$, i.e., the actions in a commitment set are exactly the ones that are enabled by a place in $\sigma$. If $\sigma$ is deterministic there is a most one transition enabled from every system place and thereby at most one action possible from each commitment set; the $\lightning$-actions are thus never enabled. Formally:

**Lemma 9** If $\sigma$ is deterministic, then there is no play in $Plays(\widehat{\mathcal{C}_\mathcal{G}}, \varrho_\sigma)$ that contains a $\lightning$-action.

**Proof** Suppose the opposite, i.e., there is a $\varrho_\sigma$ compatible play $u$ that contains a $\lightning$-action. W.l.o.g. $u = u' \lightning^{(q,A)}_{[t_1,t_2]}$ with $q \in \mathcal{P}_\mathcal{S}$, $A \subseteq post^\mathcal{G}(q)$, $t_1, t_2 \in A$, and there is no $\lightning$ action in $u'$. By construction of the $\lightning$-actions it is easy to see that if $u' \lightning^{(q,A)}_{[t_1,t_2]}$ is a play then $u'\, t_1$ and $u'\, t_2$ are as well.

If $\lightning^{(q,A)}_{[t_1,t_2]}$ is possible, the transition relation in $\widehat{\mathcal{C}_\mathcal{G}}$ also requires a process $p$ in state $(q, A)$, i.e., $state_p(u') = (q, A)$. Since there are no $\lightning$-action in $u'$ we can use the previous bisimulation result and obtain a marking $M \in \mathcal{R}(\mathcal{N}^\sigma)$ with $M \approx_{\mathfrak{B}} u'$. By bisimulation we know that $t_1$ and $t_2$ (transitions with that label) are enabled from $M$.
Let $q' \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$ be the *system* place in $M$ that corresponds to process $p$. From Lemma 2 know that $\lambda(q') = q$. Since $t_1, t_2 \in dom(p)$ we get $t_1, t_2 \in \mathcal{T}^{\boldsymbol{S}(p)}$. Place $q'$ is therefore involved in both $t_1$ and $t_2$ so we see that $t_1, t_2 \in \lambda[post^{\mathcal{N}^\sigma}(q')]$ and both can occur from $M$. This is a contradiction to the assumption that $\sigma$ is deterministic. ■

If $\sigma$ is deterministic, Lemma 9 shows that $\varrho_\sigma$ does not allow any $\lightning$-actions. We can hence neglect all $\lightning$-actions and extend our proofs for bisimulation and winningness from $\mathcal{C}_\mathcal{G}$ to $\widehat{\mathcal{C}_\mathcal{G}}$. We get that $\varrho_\sigma$ is a winning controller for $\widehat{\mathcal{C}_\mathcal{G}}$ (and also $\mathcal{C}_\mathcal{G}$) and, furthermore, bisimilar to $\sigma$. This gives us the first half of our proof of Theorem 1:

**Proposition 1** If $\sigma$ is a winning strategy for $\mathcal{G}$ then $\varrho_\sigma$ is a winning controller for $\mathcal{C}_\mathcal{G}$ and bisimilar to $\sigma$.
If $\sigma$ is a winning, deterministic strategy for $\mathcal{G}$ then $\varrho_\sigma$ is a winning controller for $\widehat{\mathcal{C}_\mathcal{G}}$ (and for $\mathcal{C}_\mathcal{G}$) and bisimilar to $\sigma$.

### 5.3.4 Translating Controllers to Strategies

In this section we provide the formal translation of controllers to strategies. We first need to restrict the possible controllers for $\mathcal{C}_\mathcal{G}$: We only consider controller that allow at most one commitment set (one $\tau$-action from each state). This restriction is needed to allow for bisimilar strategies[10]. Even though this constraint is not desirable, we can argue that it does not impose any relevant restriction on possible controller: Suppose controller $\varrho = \{f^\varrho_p\}_{p\in\boldsymbol{P}}$ allows more than one commitment set. We can build a modified controller $\varrho' = \{f^{\varrho'}_p\}_{p\in\boldsymbol{P}}$ by

$$f^{\varrho'}_p(u) = \{\tau_{(q,\ \bigcup_{i=1,\cdots,n} A_i)}\} \text{ when } f^\varrho_p(u) = \{\tau_{(q,A_1)}, \cdots, \tau_{(q,A_n)}\}$$

Whenever $\varrho$ allows multiple commitment sets, $\varrho'$ chooses the union of all of them as the new (unique) commitment set. $\varrho'$ admits the same observable sequences as $\varrho$. In particular, $\varrho'$ is winning if and only if $\varrho$ is winning. Allowing more commitment does not yield any advantage for a controller[11]. For convenience we also restrict controller even further by enforcing *exactly* one commitment set. If a controller $\varrho$ chooses no commitment set we can instead choose the empty one[12] We call this restriction on controller $\star$.

Assume now we are given a winning controller $\varrho$ for $\mathcal{C}_\mathcal{G}$ (or $\widehat{\mathcal{C}_\mathcal{G}}$) that satisfies $\star$. We need to construct a winning, bisimilar strategy $\sigma_\varrho$ for $\mathcal{G}$. Unlike controllers that are defined as functions evoked on an entire play, strategy for Petri games are defined as branching processes. We thus incrementally build a branching process for $\sigma_\varrho$. In our incremental strategy construction every system place needs to decide what transitions to allow from that place. This decision should be based on the causal past of that place and should be made in accordance with $\varrho$ in order to, in the end, obtain a bisimilar strategy. We would therefore like to be able to translate the causal past to a play in $\mathcal{C}_\mathcal{G}$, give this play to controller $\varrho$ and enable exactly the transitions that the controller chose as a commitment set. The crucial step is the translation of the causal past of place $q$ to a play in $\mathcal{C}_\mathcal{G}$ that is compatible with $\varrho$. When translating strategies to controller

[10] If two commitment sets are chosen, two states that are indistinguishable by weak-bisimulation allow different behavior. A strategy must hence allow the behavior of both states from a single place. This is in general not possible.
[11] Instead of building the union-commitment set it would be valid to simply choose one of the allowed commitment sets. An approach similar to this has been realised in [22].
[12] Unlike the restriction to at most one chosen commitment set, the further restriction to exactly one commitment set is not needed to maintain bisimilarity but purely for convenience.

**function** $extend_\varrho$
  **input:** $u \in Plays(\mathcal{C}_\mathcal{G})$
    **for all** $p \in \boldsymbol{P}$ **with** $state_p(u) \in \mathcal{P}_\mathcal{S}$ **do**
      Compute $f_p^\varrho(view_p(u)) = \{\tau_{(state_p(u),A)}\}$
      $u \leftarrow u\,\tau_{(state_p(u),A)}$
    **end for**
  **return:** $u$

**(a)**

**function** $rec'_\varrho$
  **input:** $\kappa \leftarrow \kappa_0, \cdots, \kappa_{n-1} \in \mathcal{T}^*$
    $u \leftarrow extend_\varrho(\epsilon)$
    **for** $i \leftarrow 0$ **to** $n-1$ **do**
      $u \leftarrow u\,\kappa_i$
      **assert** $u \in Plays(\mathcal{C}_\mathcal{G})$ **(A)**
      $u \leftarrow extend_\varrho(u)$
    **end for**
  **return:** $u$

**(b)**

**function** $rec_\varrho$
  **input:** $past_\mathcal{T}(q)$
    Order $past_\mathcal{T}(q)$ totally into a sequence
    $\kappa \leftarrow \kappa_0, \cdots, \kappa_{n-1} \in \mathcal{T}^*$
  **return:** $rec'_\varrho(\kappa)$

**(c)**

**Figure 5.9** Description of algorithm $rec_\varrho$ used to reconstruct a play in $\mathcal{C}_\mathcal{G}$ from the transitions in the causal past of a place.

(Sec. 5.3.3) we had to do conserve, i.e., translate a local view into the causal past of a place. We could easily do so by ignoring all $\tau$-actions using $\langle\cdot\rangle^\mathcal{T}_\downarrow$. In contrast, in our present translation we have to *add* $\tau$-actions to obtain a play in $\mathcal{C}_\mathcal{G}$. For a place $q$ with causal past $past_\mathcal{T}(q)$ we thus want to compute a $\varrho$ compatible play $u$ that contains the same observable actions, i.e., where $\langle u\rangle^\mathcal{T}_\downarrow = past_\mathcal{T}(q)$.

## Play Reconstruction

Given the causal past of $q$ we need to add $\tau$-actions to the play. We pursue an incremental construction of that play: We begin with an empty play and add the transitions in the past of $q$ one at a time. In between we need to play $\tau$-actions to allow all processes on system places to choose a commitment set. The incremental construction is done by a function $rec_\varrho$ that is depicted in Fig. 5.9. As the past of $q$ is a partially ordered set and it is intrinsically hard to write function on such sets, we begin by fixing a total order $\kappa$. The main work is then done by $rec'_\varrho$. The idea is to add the transitions in $\kappa$ one at a time and in between play as many $\tau$-actions as possible. The subroutine *extend* performs the addition of $\tau$-actions by moving every process that can choose a commitment set ($state_p(u) \in \mathcal{P}_\mathcal{S}$) to the chosen set by playing a $\tau$-action. Note that, as $\varrho$ satisfies $\star$, every process allows for exactly one commitment set. $rec'_\varrho$ then repetitively extends a play and adds a transition from $\kappa$. In the algorithm, we included an assertion **(A)** that requires the trace constructed so far to be a play in $\mathcal{C}_\mathcal{G}$. We discuss this assertion later. For now assume it is always fulfilled. It is easy to see that if $rec_\varrho$ does not trigger the assertion, the outputted play $u$ satisfies $\langle u\rangle^\mathcal{T}_\downarrow = past_\mathcal{T}(q)$.

The first step in $rec_\varrho$ consists in finding a total order of $past_\mathcal{T}(q)$. We can prove that the resulting trace does not depend on the concrete choice. $rec_\varrho$ is hence a deterministic procedure.

**Lemma 10** Let $q \in \mathcal{P}^{\mathcal{G}^{\mathfrak{U}}}$ be any place in the unfolding of $\mathcal{G}$. If $\kappa_1$ and $\kappa_2$ are two totally ordered sequences of $past_{\mathcal{T}}(q)$ then $rec'_{\varrho}(\kappa_1) = rec'_{\varrho}(\kappa_2)$.

**Proof** We first show that statements for two totally ordered sequences $\kappa$ and $\kappa'$ of $past_{\mathcal{T}}(q)$ that only differ at exactly one location, i.e., two consecutive transitions $t_1, t_2$ have been swapped. So $\kappa = \kappa^{r_1}, t_1, t_2, \kappa^{r_2}$ and $\kappa' = \kappa^{r_1}, t_2, t_1, \kappa^{r_2}$ for some sequences $\kappa^{r_1}$ and $\kappa^{r_r}$. Since $t_1$ and $t_2$ are unordered in $past_{\mathcal{T}}(q)$, we can conclude $(pre^{\mathcal{G}}(t_1) \cup post^{\mathcal{G}}(t_1)) \cap (pre^{\mathcal{G}}(t_2) \cup post^{\mathcal{G}}(t_2)) = \emptyset$. Since $t_1$ and $t_2$ can be fired exactly after another we can conclude that $t_1$ and $t_2$ involve different slices. By our construction of $\mathcal{C}_{\mathcal{G}}$ we hence know that $t_1 \mathbb{I} t_2$. We can see that $rec'_{\varrho}(\kappa)$ and $rec'_{\varrho}(\kappa')$ have the following form:

$$rec'_{\varrho}(\kappa) = u'\, t_1 \underbrace{\tau \cdots \tau}_{(1)}\, t_2 \underbrace{\tau \cdots \tau}_{(2)}\, u'' \text{ and } rec'_{\varrho}(\kappa') = u'\, t_2 \underbrace{\tau \cdots \tau}_{(3)}\, t_1 \underbrace{\tau \cdots \tau}_{(4)}\, u''$$

For some plays $u'$ and $u''$. The $\tau$-actions played in **(1)** and **(4)** only involve processes from $dom(t_1)$ and the ones in **(2)** and **(3)** from $dom(t_2)$. Since $t_1 \mathbb{I} t_2$ and all the $\tau$-actions are local to one process, both $rec'_{\varrho}(\kappa)$ and $rec'_{\varrho}(\kappa')$ describe identical traces.

We have shown the claim for two totally ordered sequences that differ at exactly one location. The proof for the general $\kappa_1$ and $\kappa_2$ follows by induction on the minimal number of swaps used to unify $\kappa_1$ and $\kappa_2$ using the insight from above. ■

We can now discuss the assertion. It can happen that adding a transition from $\kappa$ results in a play that is not in $Plays(\mathcal{C}_{\mathcal{G}})$. The controller could have chosen its commitment sets such that the action that is added from $\kappa$ cannot be taken. We can, however, show that *if* $rec_{\varrho}$ did not cause an assertion, the obtained play is a valid play in $\mathcal{C}_{\mathcal{G}}$ and moreover compatible with $\varrho$. We can, furthermore, observe that if there is some play in $Plays(\mathcal{C}_{\mathcal{G}}, \varrho)$ that contains exactly the observable actions from the past of a place, then $rec_{\varrho}$ is guaranteed to find such a play without causing an assertion.

**Lemma 11** Let $past_{\mathcal{T}}(q)$ be the causal past of some place $q \in \mathcal{P}^{\mathcal{G}^{\mathfrak{U}}}$ in the unfolding of $\mathcal{G}$.

1. Assume that $u = rec_{\varrho}(past_{\mathcal{T}}(q))$ and no assertion is triggered. Then $u \in Plays(\mathcal{C}_{\mathcal{G}}, \varrho)$.
2. If there is a play $u \in Plays(\mathcal{C}_{\mathcal{G}}, \varrho)$ with $\langle u \rangle^{\mathcal{T}}_{\downarrow} = past_{\mathcal{T}}(q)$ then $rec_{\varrho}(past_{\mathcal{T}}(q))$ does not trigger an assertion

**Proof** The first statement follows from the definition of $rec_{\varrho}$ and the fact that all observable actions are uncontrollable. The second claim follows since by $\star$ every process has chosen at most one commitment set and simulation is therefore unique. ■

We demonstrate *rec* on a quick example for our translation from Fig. 5.3. Consider a possible winning controller $\dot{\varrho}$ for $\dot{\mathcal{C}_{\mathcal{G}}}$ where $p_2$ allows the following: Whenever in state $C$ it chooses the commitment set including $i$ and hence allows a move to $D$. If in state $D$ for the first time, $p_2$ moves to the commitment set containing $b$, i.e., restricts communication to $b$. After executing $b$ it can deduce whether the environment played $e_1$ or $e_2$. In case of $e_1$ it allows $b$ for one more time and subsequentially terminates. In case of $e_2$ it allows communication on $a$, afterwards moves to state $D$ and terminates. The relevant plays and the decision of $\dot{\varrho}$ are depicted in Fig. 5.10. We want to observe what $rec_{\dot{\varrho}}$ does if applied to the causal pasts of some places in the unfolding. In Fig. 5.11 we depicted an initial fragment of the unfolding of the Petri game $\dot{\mathcal{G}}$. We consider the transitions in the causal pasts of the places $q$ and $q'$ which are surrounded in blue and red. Applying $rec_{\dot{\varrho}}$ to the transitions in the causal past of $q$ yields

$$rec_{\dot{\varrho}}(past_{\mathcal{T}}(q)) = [\ \tau_{(C,\{i\})}\ ,\ i\ ,\ \tau_{(D,\{b\})}\ ,\ e_1\ ,\ b\ ,\ \tau_{(D,\{b\})}\ ]_{\mathbb{I}}$$

which is a valid trace in $Plays(\dot{\mathcal{C}_{\mathcal{G}}}, \dot{\varrho})$ that agrees with the causal past of $q$ on all observable

| $u \in Plays(\dot{\mathcal{C}_\mathcal{G}}, \dot{\varrho}) \cup Plays_p(\dot{\mathcal{C}_\mathcal{G}})$ | $f^{\dot{\varrho}}_{p_2}(u)$ |
|---|---|
| $\epsilon$ | $\{\tau_{(C,\{i\})}\}$ |
| $\tau_{(C,\{i\})}$ | $\emptyset$ |
| $\tau_{(C,\{i\})}, i$ | $\{\tau_{(D,\{b\})}\}$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}$ | $\emptyset$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}, e_1, b$ | $\{\tau_{(D,\{b\})}\}$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}, e_2, b$ | $\{\tau_{(D,\{a\})}\}$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}, e_1, b, \tau_{(D,\{b\})}$ | $\emptyset$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}, e_e, b, \tau_{(D,\{a\})}$ | $\emptyset$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}, e_1, b, \tau_{(D,\{b\})}, b$ | $\{\tau_{(D,\emptyset)}\}$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}, e_e, b, \tau_{(D,\{a\})}, a$ | $\{\tau_{(D,\{i\})}\}$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}, e_e, b, \tau_{(D,\{a\})}, a, \tau_{(C,\{i\})}$ | $\emptyset$ |
| $\tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}, e_e, b, \tau_{(D,\{a\})}, a, \tau_{(D,\{i\})}, i$ | $\{\tau_{(D,\emptyset)}\}$ |

**Figure 5.10** Example winning controller $\dot{\varrho}$ for $\dot{\mathcal{C}_\mathcal{G}}$ in Fig. 5.3. The controller is depicted by listing plays and the decision of $f^{\dot{\varrho}}_{p_2}$ on them.

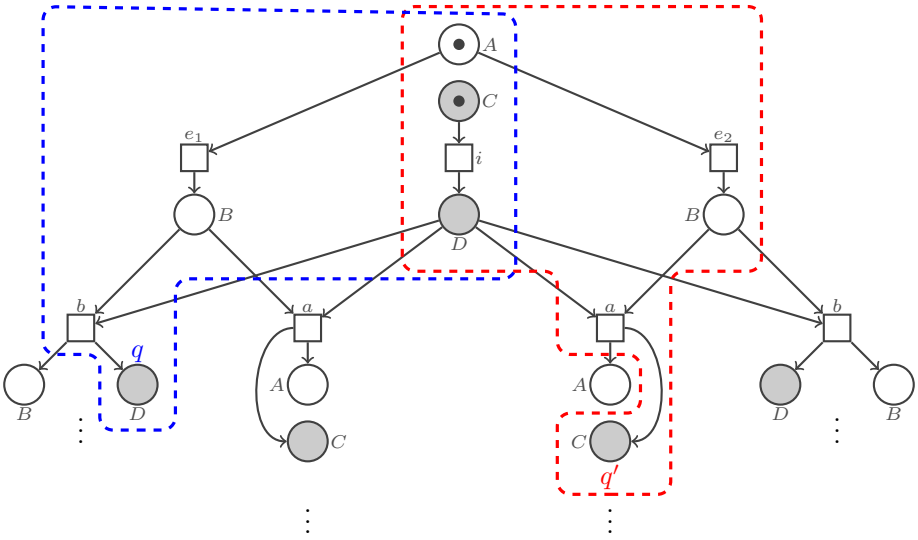



**Figure 5.11** Initial fragment of the unfolding of $\dot{\mathcal{C}_\mathcal{G}}$ from Fig. 5.3. The causal past of places $q$ and $q'$ are surrounded by blue or red.

actions. The concrete order of the causal past of $q$ is irrelevant as stated in Lemma 10. On the other hand, evoking $rec_{\dot{\varrho}}$ on the causal past of $q'$ (in red) causes the assertion to break. $\dot{\varrho}$ simply does not allow a play that contains $i, e_2, a$ as observable actions. The interested reader is encouraged to comprehend how $rec_{\dot{\varrho}}$ computes the play for the past of $q$ and why it fails for $q'$.

*rec* for itself is a rather unitive concept. The idea of why we need such an algorithm should become more clear in the context of the actual translation of a given controller to a strategy. One may view *rec* as a black box that allows us to transfer the transitions in the causal past of a place to a play in the control game.

## The construction of $\sigma_\varrho$

Using $rec_\varrho$ we can finally define the construction of $\sigma_\varrho$. It is depicted in Fig. 5.12 and does what we described informally. We incrementally build up a branching process by iterating over every reachable marking $M$ in the partially constructed strategy. Every place $q$ in a marking $M$ needs

Start by creating an initial marking $In^{\sigma_\varrho}$ and extend $\lambda$ s.t. $\lambda(In^{\sigma_\varrho}) = In^{\mathcal{G}}$.
Iterate over every unprocessed reachable marking $M$ in $\mathcal{N}^{\sigma_\varrho}$: Consider every $q \in M$:

- If $q$ is a system place, i.e., $q \in \lambda^{-1}[\mathcal{P}_{\mathcal{S}}]$:
  $q$ belongs to a process $p_q \in \boldsymbol{P}$, i.e., $q \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p_q)}]$ Compute $u = rec_\varrho(past_{\mathcal{T}}(q))$. Assume that no assertion is violated and that $\zeta(state_{p_q}(u)) = \lambda(q)$ **(A)**. Because of **(A)** and the fact that in $u$ every process has chosen a commitment set it holds that $state_{p_q}(view_{p_q}(u)) = (\lambda(q), B)$. Define $\mathbb{A}_q = B \subseteq \mathcal{T}^{\mathcal{G}}$.
- If $q$ is an environment place, i.e., $q \in \lambda^{-1}[\mathcal{P}_{\mathcal{E}}]$:
  Define $\mathbb{A}_q = post^{\mathcal{G}}(\lambda(q))$

Define

$$\Delta_M = \{t \in \mathcal{T}^{\mathcal{G}} \mid pre^{\mathcal{G}}(t) \subseteq \lambda[M] \wedge \forall q \in M : \lambda(q) \in pre^{\mathcal{G}}(t) \Rightarrow t \in \mathbb{A}_q\}$$

These are all transitions that can occur and where all places have agreed on. We want to add exactly the transitions from $\Delta_M$ from $M$: For every $t \in \Delta_M$: Check if there already exists a transition $t'$ with $pre^{\mathcal{N}^{\sigma_\varrho}}(t') \subseteq M$ and $\lambda(t') = t$:

- If it already exists, do not add anything.
- If it does not exist: Create a new transition $t'$ and extend the flow, s.t., $pre^{\mathcal{N}^{\sigma_\varrho}}(t') = \{q \in M \mid \lambda(q) \in pre^{\mathcal{G}}(t)\}$ and extend $\lambda$ with $\lambda(t') = t$. Add a new place $q'$ for every $q \in post^{\mathcal{G}}(t)$ with $\lambda(q') = q$ and extend the flow, s.t., $pre^{\mathcal{N}^{\sigma_\varrho}}(q') = \{t'\}$.

Mark $M$ as processed and continue with another, unprocessed marking.

**Figure 5.12** Construction of strategy $\sigma_\varrho$ for $\mathcal{G}$ from a given controller $\varrho$ for $\mathcal{C}_{\mathcal{G}}$

to decide what transitions to enable. This decision is stored in a set $\mathbb{A}_q$. Since an environment place cannot be restricted by a strategy all outgoing transitions are allowed ($\mathbb{A}_q = post^{\mathcal{G}}(\lambda(q))$). For each system place $q$ we consider its causal past and convert it to a play in $\mathcal{C}_{\mathcal{G}}$ using $rec_\varrho$. In this play the process that corresponds to $q$ has chosen a commitment set., i.e., is in a state of the form $(\_, \_)$. We define $\mathbb{A}_q$ to be the set of transitions that are in the current commitment set of that process. $q$ hence copies the decision made by the corresponding process on the reconstructed play. Once we have computed $\mathbb{A}$ for every place in the marking we add all transitions where all places agree on, i.e., compute $\Delta_M$. Since $\mathcal{G}$ is sliceable and therefore safe we can uniquely tell which places need to agree on a transition: $\Delta_M$ is the set of all transitions that are enabled in $\lambda[M]$ and where all places $q$ in the precondition of $t$ ($\lambda(q) \in pre^{\mathcal{G}}(t)$) have agreed on $t$ ($t \in \mathbb{A}_q$). For now we impose an assertion **(A)** in the construction. We later see that the assertion can be neglected, i.e., the causal past of any place in the partially constructed strategy can always be covered to a play using $rec_\varrho$.

To see our construction on an example we again consider the exemplary controller $\dot{\varrho}$ we explained before and whose description is depicted in Fig. 5.10. If we apply our construction we end up with the strategy $\dot{\sigma}_\varrho$ depicted in Fig. 5.13 (e). Note that $\dot{\sigma}_\varrho$ allows the same behavior as $\dot{\varrho}$: After moving to $D$ the system player allows communication on $b$ only. Depending on whether the environment chose $e_1$ or $e_2$, $\dot{\sigma}_\varrho$ either allows $b$ once more or allows $a$ and afterwards moves to $D$ using $i$. Apart from showing the final strategy, Fig. 5.13 also depicts possible indeterminate steps in the strategy construction. Next to each place the set $\mathbb{A}$ as computed in the construction is given in red. The gray label is the one given by $\lambda$. Places $q_1, \cdots, q_5$ are named explicitly in blue. The causal past of them is surrounded in blue.

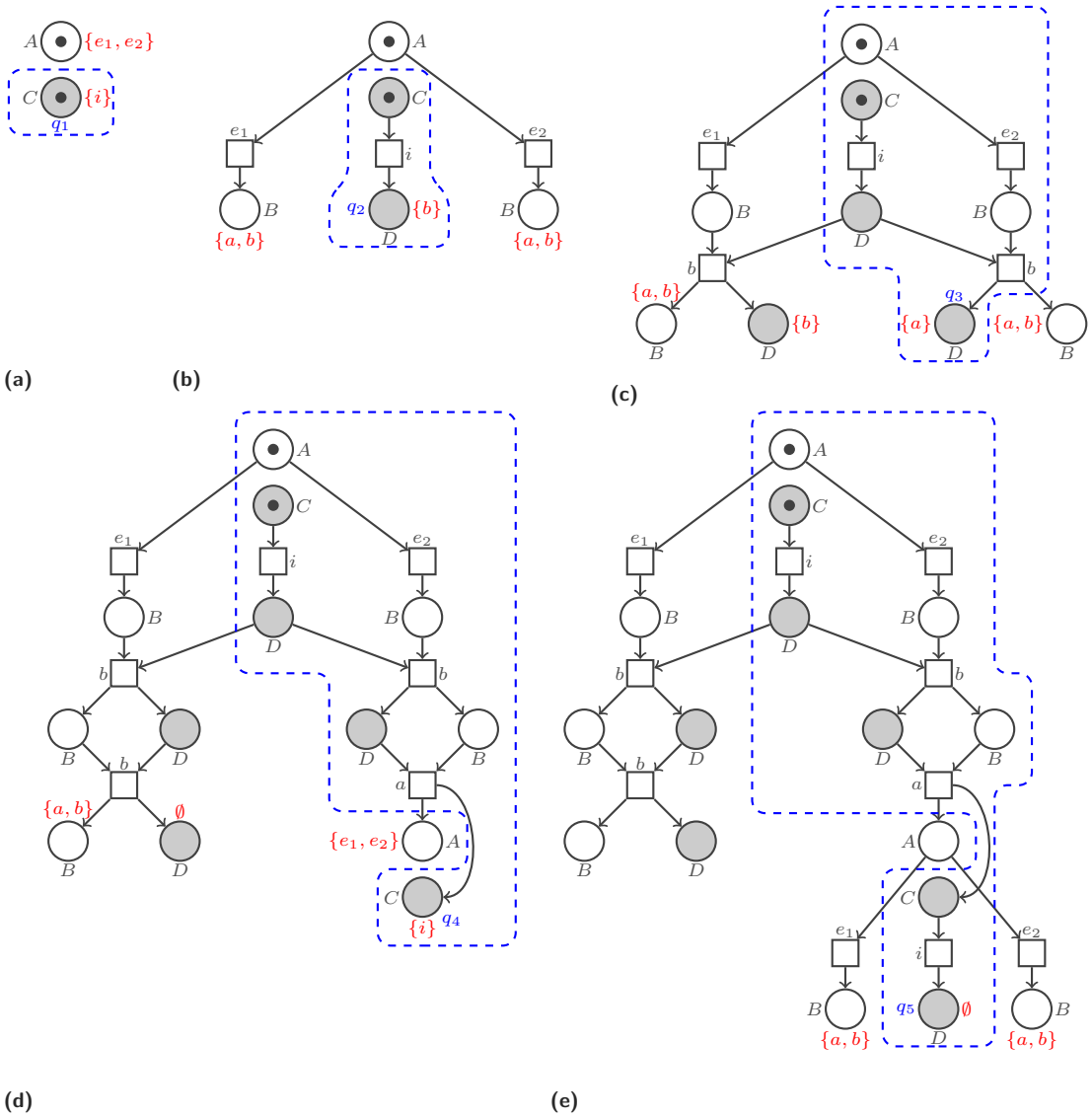



**Figure 5.13** Intermediate Steps in the construction of strategy $\dot{\sigma}_\varrho$ (for $\dot{\mathcal{C}}_\mathcal{G}$ from Fig. 5.3) from controller $\dot{\varrho}$ (Fig. 5.10). The gray label is given $\lambda$. The red label are the transitions that should be enabled, i.e., the set $\mathbb{A}$ computed in the construction. The causal past of places $q_1$ to $q_n$ is surrounded in blue. (e) illustrates the final strategy.

We try to comprehend the construction depicted in Fig. 5.13: Construction begins with an initial marking (a). For every system place in that marking we compute the transitions in the causal past and reconstruct a play using $rec_{\dot{\varrho}}$. For the system place $q_1$, $rec_{\dot{\varrho}}$ applied to the empty causal past gives us the play $[\tau_{(C,\{i\})}]_\mathbb{I}$. After playing $[\tau_{(C,\{i\})}]_\mathbb{I}$ the process that corresponds to $q_1$ (process $p_2$) is in state $(C, \{i\})$. In our construction we define $\mathbb{A}_{q_1}$ as the set of transitions in the commitment set of the corresponding process, so we derive that $\mathbb{A}_{q_1} = \{i\}$.. For the environment place we define $\mathbb{A}$ as the set of all outgoing transitions, i.e., $\{e_1, e_2\}$. After having computed the $\mathbb{A}$ sets for all places in the initial marking, we add all transitions that are allowed by all involved places and corresponding places for the postcondition. We end up with the branching process in (b). We repeat the same procedure: For both new environment places we define $\mathbb{A}$ as the set of outgoing transitions, in this case $\{a, b\}$. For the system place $q_2$ we compute $rec_{\dot{\varrho}}$ on the causal past (surrounded in blue) which gives us the play $[\tau_{(C,\{i\})}, i, \tau_{(D,\{b\})}]_\mathbb{I}$ explaining

why $\mathbb{A}_{q_2} = \{b\}$. As $q_2$ restricted its set $\mathbb{A}$ to $b$, only transition labelled $b$ are added from that place. We proceed this way and add more and more places and transitions. The construction terminates with the strategy in (e). At this point all $\mathbb{A}$-sets are such that no more transitions can be added and our construction terminates. The reader is encouraged to convince herself of this construction and, in particular, to comprehend how every $\mathbb{A}$-set is chosen.

Coming back to our general translation, we can show that the construction does indeed yield a strategy. The observation is that each place in $\sigma_\varrho$ decides which transitions to enable (i.e., chooses $\mathbb{A}$) based on its causal past only. The decision is therefore based solely on the place and not on the current marking.

**Lemma 12** $\mathcal{N}^{\sigma_\varrho}$ is a strategy for $\mathcal{G}$

**Proof** It is easy to verify that the constructed net $\mathcal{N}^{\sigma_\varrho}$ is a branching process of $\mathcal{G}$. We need to prove *Justified refusal*:
Suppose there is a reachable marking $M$ in $\sigma_\varrho$ and transition $t$ in $\mathcal{G}$ s.t. $t$ is enabled in $\lambda[M]$ (i.e., $pre^{\mathcal{G}}(t) \subseteq \lambda[M]$) but there is no $t'$ with $\lambda(t') = t$ enabled in $M$.
Since no such $t'$ has been added to $\sigma_\varrho$ we conclude that $t \notin \Delta_M$.
Since we know that $pre^{\mathcal{G}}(t) \subseteq \lambda[M]$, the definition of $\Delta_M$ gives us that there is a $q \in M$ with $\lambda(q) \in pre^{\mathcal{G}}(t)$ but $t \notin \mathbb{A}_q$. By construction of $\mathbb{A}_q$ we can conclude that $q$ is a system place. We, furthermore, know that $\mathbb{A}_q$ solely depends on the causal past of $M$. For every marking $M'$ that contains $q$ we always have that $t \notin \mathbb{A}_q$ and therefore $t \notin \Delta_M$. It hence holds that $t \notin \lambda[post^{\mathcal{N}^{\sigma_\varrho}}(q)]$. ■

## Strategy-Equivalence

We can now prove $\varrho$ and $\sigma_\varrho$ bisimilar. As relation $\approx_{\mathfrak{B}}$ we use the same one we used before and restrict it to the reachable markings in $\mathcal{N}^{\sigma_\varrho}$ and plays in $Plays(\mathcal{C}_\mathcal{G}, \varrho)$. We begin with a consequence of Lemma 3.

**Lemma 13** If $M \approx_{\mathfrak{B}} u$, $p \in \boldsymbol{P}$ and $q \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$, then computing $u' = rec_\varrho(past_\mathcal{T}(q))$ does not trigger any assertion. If $u$ is maximal w.r.t. $\tau$-actions, i.e., there is no $\tau$ s.t. $u\,\tau \in Plays(\mathcal{C}_\mathcal{G}, \varrho)$, it holds that

$$view_p(u') = view_p(u)$$

**Proof** We first show that computing $u' = rec_\varrho(past_\mathcal{T}(q))$ does not trigger any assertion: By definition from $\approx_{\mathfrak{B}}$ it holds that $M = \mathcal{N}^{\sigma_\varrho}[\triangledown\, \langle u \rangle^\mathcal{T}_\downarrow\,]$. By Lemma 3 we get that

$$past_\mathcal{T}(q) = \langle view_p(u) \rangle^\mathcal{T}_\downarrow \tag{1}$$

Since $u \in Plays(\mathcal{C}_\mathcal{G}, \varrho)$ we know that $view_p(u) \in Plays(\mathcal{C}_\mathcal{G}, \varrho)$. The claim that no assertion is triggered now follows from Lemma 11.

We can now show that $view_p(u') = view_p(u)$. By definition of $rec_\varrho$ it holds that $\langle u' \rangle^\mathcal{T}_\downarrow = past_\mathcal{T}(q)$. When using together with **(1)** we conclude that

$$\langle u' \rangle^\mathcal{T}_\downarrow = \langle view_p(u) \rangle^\mathcal{T}_\downarrow$$

It now follows that

$$\begin{aligned} view_p(\langle u' \rangle^\mathcal{T}_\downarrow) &= view_p(\langle view_p(u) \rangle^\mathcal{T}_\downarrow) \\ &= view_p(view_p(\langle u \rangle^\mathcal{T}_\downarrow)) \\ &= view_p(\langle u \rangle^\mathcal{T}_\downarrow) \\ &= \langle view_p(u) \rangle^\mathcal{T}_\downarrow \end{aligned}$$

since $view_p(\cdot)$ is idempotent and the $\tau$-actions removed by $\langle\cdot\rangle^{\mathcal{T}}_{\downarrow}$ are local, i.e., $\langle view_p(u)\rangle^{\mathcal{T}}_{\downarrow}) = view_p(\langle u\rangle^{\mathcal{T}}_{\downarrow})$. We, furthermore, know that $u$ and $u'$ are both maximal w.r.t. $\tau$-actions (by assumption and from definition of $rec_\varrho$). Because of $\star$ every process chooses exactly one commitment set. The $\tau$-actions in both $view_p(u')$ and $view_p(u)$ are hence unique and we get $view_p(u') = view_p(u)$. ■

The definition $\sigma_\varrho$ is completely independent to the definition of $\approx_{\mathfrak{B}}$. Lemma 13, however, characterizes a connection between both. In our construction of $\sigma_\varrho$ each place computes its decision (the set $\mathbb{A}$) by applying $rec_\varrho$ to the transition in its causal past. In $\approx_{\mathfrak{B}}$-related situations this results, according to Lemma 13, in the local view of one of the processes. This observation allows us to show that $\varrho$ and $\sigma_\varrho$ are bisimilar. We can reason in both direction:

- If $t$ is enabled in $M$ then by construction of $\sigma_\varrho$ every involved system place $q$ has allowed it, i.e., $t \in \mathbb{A}_q$. The set $\mathbb{A}_q$ was chosen by computing the causal past of that place and convert it to a play using $rec_\varrho$. By Lemma 13 each place therefore computes the local view of one of the processes on $u$ and copies the decision. Since $t$ is allowed by all involved system places, we can conclude that all involved processes must have chosen commitment set where $t$ is included. Hence $u$ can be extended by $t$ (after playing sufficiently many $\tau$-actions to choose a commitment set).

- If, on the other hand, $u$ can be extend with $t$ by $\varrho$ then all involved processes enable $t$. So every process $p \in dom(t)$ either resides on an environment place where it has no control or it is on a system place where it must have chosen a commitment set that includes $t$. Each system place in $M$ evokes $rec_\varrho$ on its causal past and, by Lemma 13, therefore computes the local view of on process on $u$. The place then copies the decision made on that local play, i.e., copies the chosen commitment set. Since $t$ is in the commitment of every involved process every place $q$ involved in $t$ will allows $t$ (i.e., chooses $\mathbb{A}_q$ such that $t \in \mathbb{A}_q$). So together the system places allow $t$ from $M$.

We can now prove this formally. Since $\varrho$ is, by assumption, winning we can neglect all $\lightning$-actions.

**Lemma 14** If $M \approx_{\mathfrak{B}} u$ and $M \,[\, t \,\rangle\, M'$ for some $M' \in \mathcal{R}(\mathcal{N}^{\sigma_\varrho})$ there exists $u' = u\,\tau^*\,t \in Plays(\mathcal{C}_\mathcal{G}, \varrho)$ and $M' \approx_{\mathfrak{B}} u'$.

**Intuition** If $M \,[\, t \,\rangle\, M'$ then every system place $q$ in $M$ must have allowed $t$, i.e., $t \in \mathbb{A}_q$. By construction the set $\mathbb{A}_q$ was obtained by computing the causal past, convert it via $rec_\varrho$ and check which commitment set was chosen. By Lemma 13 this yields the same commitment set that the processes would have chosen after $u$. So every process $p \in dom(t)$ that is on a system place chooses a commitment set where $t$ is included. After moving each process to their commitment-set state they hence all allow $t$.

**Proof** From $M \approx_{\mathfrak{B}} u$ we get $\lambda[M] = \zeta(state(u))$ **(1)** by Lemma 2. Since $M \,[\, t \,\rangle\, M'$ all places in $M$ that are involved in $t$ allow it, i.e., for every $q \in M$ with $\lambda(q) \in pre^{\mathcal{G}}(t)$ it holds that $t \in \lambda[post^{\mathcal{N}^{\sigma_\varrho}}(q)]$ . We hence conclude that $t \in \Delta_M$ **(2)**.
Let $u_\tau$ be the trace obtained from $u$ by playing as many $\tau$ actions as possible s.t. there are no $\tau$-actions enabled after $u_\tau$. It holds that $M \approx_{\mathfrak{B}} u_\tau$. By assumption $\star$ every process that can chooses a commitment set, so for every $p$ with $\zeta(state_p(u_\tau)) \in \mathcal{P}_\mathcal{S}$ we know that $state_p(u_\tau) = (\_, \_)$.
Assume for contradiction that $u_\tau\, t \notin Plays(\mathcal{C}_\mathcal{G}, \varrho)$. Because of **(1)** we know that $t$ would be possible after $u_\tau$ if the commitment sets are chosen appropriately. There hence is a process $p \in dom(t)$ that has chosen a commitment set that does not include $i$, i.e., $state_p(u_\tau) = (\lambda(q_p), B)$ where $t \notin B$.
Let $q_p \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$ be the place that corresponds to $p$ in $M$. By Lemma 13 and since

$u_\tau$ is by assumption maximal we now know that

$$view_p(rec_\varrho(past_\mathcal{T}(q_p))) = view_p(u_\tau)$$

From **(1)** and since $p \in dom(t)$ we can conclude that $\lambda(q_p) \in pre^\mathcal{G}(t)$ so since $t \in \Delta_M$ **(2)** we get that $t \in \mathbb{A}_{q_p}$.
We can now analyse the construction of $\sigma_\varrho$ to observe how $\mathbb{A}_{q_p}$ is derived. It is computed by matching

$$state_p(view_p(rec_\varrho(past_\mathcal{T}(q_p)))) = (\lambda(q_p), \mathbb{A}_{q_p})$$

But now

$$\begin{aligned}(\lambda(q_p), B) &= state_p(u_\tau)\\ &= state_p(view_p(u_\tau))\\ &= state_{\bar{a}}(view_p(rec_\varrho(past_\mathcal{T}(q_p))))\\ &= (\lambda(q_p), \mathbb{A}_{q_p})\end{aligned}$$

So $B = \mathbb{A}_{q_p}$., i.e., the transitions allowed by place $q_p$ are exactly the transitions that $p$ has chosen as a commitment set. This is a contradiction since $t \in \mathbb{A}_{q_p}$ **(2)** but by assumption $t \notin B$. ■

We can use the previous lemma to justify our assumption **(A)** made in the construction of $\sigma_\varrho$.

**Corollary** For any place $q \in \mathcal{P}^{\mathcal{N}^{\sigma_\varrho}}$ with $q \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p_q)}]$, computing $u = rec_\varrho(past_\mathcal{T}(q))$ does not trigger an assertion and $\zeta(state_{p_q}(u)) = \lambda(q)$.

**Proof** $q$ is part of some reachable marking $M$. Using Lemma 14 we get that there is some $u' \in Plays(\mathcal{C}_\mathcal{G}, \varrho)$ with $M \approx_\mathfrak{B} u'$. By Lemma 13 computing $rec_\varrho(past_\mathcal{T}(q))$ does not trigger any assertion. For the second part we know (from Lemma 13) that $view_p(u') = view_p(u)$. Now

$$\begin{aligned}\zeta(state_{p_q}(u)) &= \zeta(state_{p_q}(view_{p_q}(u)))\\ &= \zeta(state_{p_q}(view_{p_q}(u')))\\ &= \zeta(state_{p_q}(u'))\\ &= \lambda(q)\end{aligned}$$

where the second equability follows from $view_p(u') = view_p(u)$ and the third from Lemma 2 since $M \approx_\mathfrak{B} u'$. ■

**Lemma 15** If $M \approx_\mathfrak{B} u$ and $u' = u\,t \in Plays(\mathcal{C}_\mathcal{G}, \varrho)$ then there exists $M' \in \mathcal{R}(\mathcal{N}^{\sigma_\varrho})$ with $M\ [\,t\,\rangle\ M'$ and $M' \approx_\mathfrak{B} u'$.

**Intuition** If $u\,t \in Plays(\mathcal{C}_\mathcal{G}, \varrho)$ then every process that is involved in $t$ and is on a system place must have chosen a commitment set where $t$ is included. By Lemma 13 the system places in $M$ copy the commitment sets and use them as the set of action they want to enable ($\mathbb{A}_q$). As $t$ is in every commitment set of involved processes we get $t \in \mathbb{A}_q$ for every system place that is involved in $t$. By construction of $\sigma_\varrho$, $t$ is then enabled in $M$.

**Proof** From Lemma 2 we know that $\lambda[M] = \zeta(state(u))$, so $t$ is by construction enabled from $\lambda[M]$.
Assume for constriction $t$ is not enabled in $M$, i.e., forbidden by the strategy. Then $t \notin \Delta_M$. Since $t$ is enabled from $\lambda[M]$, by construction of $\Delta_M$ there must be a *system* place $q \in M$ with $\lambda(q) \in pre^\mathcal{G}(t)$ but $t \notin \mathbb{A}_q$ **(1)**, i.e., there is at least one place that hindered $t$ from being added to the strategy.

Let $p_q$ be the process $q$ belongs to, i.e., $q \in M \cap \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p_q)}]$. Since $q$ is involved in $t$ we get that $p_q \in dom(t)$. Since $u' = u\,t \in Plays(\mathcal{C}_\mathcal{G}, \varrho)$ and $q$ is a system place we get that $state_{p_q}(u) = (\lambda(q), B)$ with $t \in B$ **(2)**, i.e., $p_q$ has chosen a commitment set that contains $t$. We know that $\mathbb{A}_q$ for place $q$ is computed by matching

$$state_{p_q}(view_{p_q}(rec_\varrho(past_\mathcal{T}(q)))) = (\lambda(q), \mathbb{A}_q)$$

Let $u_\tau$ be $u$ extended with as many $\tau$-actions as possible (only necessary to fulfil the assumptions of Lemma 13). It holds that $M \approx_\mathfrak{B} u_\tau$. By Lemma 13 we get

$$view_{p_q}(rec_\varrho(past_\mathcal{T}(q))) = view_{p_q}(u_\tau)$$

It now holds that

$$\begin{aligned}(\lambda(q), \mathbb{A}_q) &= state_{p_q}(view_{p_q}(rec_\varrho(past_\mathcal{T}(q))))\\ &= state_{p_q}(view_{p_q}(u_\tau))\\ &= state_{p_q}(u_\tau)\\ &= state_{p_q}(u)\\ &= (\lambda(q), B)\end{aligned}$$

Where the fourth equality holds since $p_q$ has already chosen a commitment set after $u$, i.e., adding more $\tau$-actions to get from $u$ to $u_\tau$ does not affect $p_q$.
So $\mathbb{A}_q = B$. This is a contradiction to $t \notin \mathbb{A}_q$ **(1)** and $t \in B$ **(2)**. ■

**Lemma 16** If $M \approx_\mathfrak{B} u$ and $u' = u\,\tau \in Plays(\mathcal{C}_\mathcal{G}, \varrho)$ then $M \approx_\mathfrak{B} u'$.

**Proof** Obvious consequence from definition of $\approx_\mathfrak{B}$. ■

**Corollary** $\varrho$ and $\sigma_\varrho$ are bisimilar.

**Proof** By definition of $\approx_\mathfrak{B}$ it holds that $In^{\mathcal{N}^{\sigma_\varrho}} \approx_\mathfrak{B} \epsilon$. Since there are now $\tau$-transitions in $\mathcal{G}$ it follows from Lemma 14, Lemma 15 and Lemma 16. ■

We show next that a winning $\varrho$ results in a winning $\sigma_\varrho$. Since a winning controller for $\widehat{\mathcal{C}_\mathcal{G}}$ avoids all $\lightning$-actions the neglection of them in our bisimulation proofs is justified.

**Lemma 17** If $\varrho$ is a winning controller for $\mathcal{C}_\mathcal{G}$ or $\widehat{\mathcal{C}_\mathcal{G}}$ then $\sigma_\varrho$ is a winning strategy for $\mathcal{G}$.

**Proof** We first show that $\mathcal{N}^{\sigma_\varrho}$ is finite: Assume for contradiction it is infinite. Koenig's lemma and the fact that $\mathcal{N}^{\sigma_\varrho}$ is an occurrence net allow us to conclude that there is an infinite sequence of markings. By bisimilarity any infinite sequence of markings in $\mathcal{N}^{\sigma_\varrho}$ results in an infinite $\varrho$ compatible play. A contradiction since $\varrho$ is winning.
Now suppose $M$ is a reachable final marking in $\mathcal{N}^{\sigma_\varrho}$, i.e., there are no further transitions enabled. There is a $\varrho$ compatible play $u$ with $M \approx_\mathfrak{B} u$ and this play is maximal (up to $\tau$-actions). Since $\varrho$ is winning, $state(u)$ must be winning (Playing further $\tau$-actions does not move into winning states). It holds that $\zeta(state(u)) = \lambda[M]$ (by Lemma 2) so by construction of $\mathcal{C}_\mathcal{G}$, $\lambda[M]$ is winning. ■

## Deterministic Strategies

By Lemma 17 any winning controller $\varrho$ for either $\widehat{\mathcal{C}_\mathcal{G}}$ or $\mathcal{C}_\mathcal{G}$ results in a winning strategy $\sigma_\varrho$ for $\mathcal{G}$. If $\varrho$ is winning for $\widehat{\mathcal{C}_\mathcal{G}}$ it must additionally avoid all $\lightning$-actions. We now show that such a controller results in a deterministic $\sigma_\varrho$: The $\lightning$-action are designed such that they can occur if and only if a commitment set is chosen and two distinct actions from this set can occur. A

winning controller for $\widehat{\mathcal{C}_\mathcal{G}}$ must avoid every $\lightning$-action and therefore has to choose commitment set where at most one transitions from every set is possible. In $\sigma_\varrho$ every place decides what to enable in accordance with the commitment sets chosen by $\varrho$. If in $\varrho$ there is at most one action from each commitment set enabled, from every system place in $\sigma_\varrho$ there is at most one transition enabled.

**Lemma 18** If $\varrho$ is a controller for $\widehat{\mathcal{C}_\mathcal{G}}$ such that no play in $\mathit{Plays}(\widehat{\mathcal{C}_\mathcal{G}}, \varrho)$ contains a $\lightning$-action, then $\sigma_\varrho$ is deterministic.

**Proof** We assume for contradiction $\sigma_\varrho$ is not deterministic, i.e., there exists a reachable marking $M$ in $\mathcal{N}^{\sigma_\varrho}$ and a system place $q \in M$ from which two transitions $t_1, t_2 \in \mathit{post}^{\mathcal{N}^{\sigma_\varrho}}(q)$ are enabled.
By our previous bisimulation result there is a $u \in \mathit{Plays}(\widehat{\mathcal{C}_\mathcal{G}}, \varrho)$ with $M \approx_\mathfrak{B} u$. Choose this $u$ such that there are no more $\tau$-actions possible. Because of assumption $\star$ every process on a system place has chosen a commitment set. By bisimulation we know that $u\,t_1$ and $u\,t_2$ are both in $\mathit{Plays}(\widehat{\mathcal{C}_\mathcal{G}}, \varrho)$.
Let $p$ be the process that $q$ belongs to, i.e., $q \in \lambda^{-1}[\mathcal{P}^{\boldsymbol{S}(p)}]$. Since $q$ is in the precondition of $t_1$ and $t_2$ we have $t_1, t_2 \in \mathcal{T}^{\boldsymbol{S}(p)}$ so $p \in \mathit{dom}(t_1)$ and $p \in \mathit{dom}(t_2)$. Since $q$ is a system place we can conclude that $\zeta(\mathit{state}_p(u)) \in \mathcal{P}_\mathcal{S}$ (by Lemma 2) and, since in $u$ every process that can has chosen a commitment set, $\mathit{state}_p(u) = (\lambda(q), B)$. Since $t_1$ and $t_2$ are both enabled we derive $t_1, t_2 \in B$.
Now $t_1, t_2$ are both enabled from the same commitment set. By construction of the $\lightning$-actions it is easy to see that $u\,\lightning^{(\lambda(q),B)}_{[t_1,t_2]}$ is a play in $\mathit{Plays}(\widehat{\mathcal{C}_\mathcal{G}})$ and, since all $\lightning$-actions are uncontrollable, in $\mathit{Plays}(\widehat{\mathcal{C}_\mathcal{G}}, \varrho)$. A contradiction. ■

Lemma 17 together with Lemma 18 give us the second half of our correctness proof:

**Proposition 2** If $\varrho$ is a winning controller for $\mathcal{C}_\mathcal{G}$, then $\sigma_\varrho$ is a winning strategy for $\mathcal{G}$ and bisimilar to $\varrho$. If $\varrho$ is a winning controller for $\widehat{\mathcal{C}_\mathcal{G}}$, then $\sigma_\varrho$ is a winning, deterministic strategy for $\mathcal{G}$ and bisimilar to $\varrho$.

Combining Proposition 1 and Proposition 2 we conclude Theorem 1.

## 5.4 On Size and Lower Bounds

As we briefly discussed in Chapter 4 the simplest translation of a Petri game $\mathcal{G}$ is to return a minimal winning-equivalent control game $\mathcal{C}_\mathcal{G}$. Such a translation compromises all structure of $\mathcal{G}$ and thereby the behavior of any winning strategy. In contrast, we proposed strategy-equivalence as a notion of structure-perseverance and showed that it is fulfilled by our translation.

Apart form preferring structurally similar translation one is interested in both the computational effort required to compute the translation as well as the size of the translation itself. There is a natural trade of between both. The size of a computed trivial translation is minimal, whereas the game is required to be solved first, resulting in a high computational effort[13]. On the other hand, our translation aims to "truly" *translate* a game without needing to solving it. While this yields, in general, bigger game, the computational effort is significantly lower.

In this section we analyse the size of our translation as well as the computational effort needed to compute it. We, furthermore, prove that our translation is asymptomatically-optimal in size if we require strategy-equivalence by giving an exponential lower bound. While this does not answer the question whether there is a sub-exponential translation, it highlight that such a translation would inevitably destroy the structure of the game.

[13] The results of this thesis allow us to derive that Petri games inherit the non-elementary lower bound shown for control games [15].

## The size of $\mathcal{C}_\mathcal{G}$ and $\widehat{\mathcal{C}_\mathcal{G}}$

Before we can proceed with a concrete analyses we need to agree on the parameters used to define the size of a control game and Petri games. Two natural parameters that are well suited for measuring the size of a control game are the number of local states $|\bigcup_{p\in \boldsymbol{P}} S_p|$ as well as the number of actions $|\Sigma|$. Conversely, for Petri games the number of places $|\mathcal{P}|$ and transitions $|\mathcal{T}|$ are good candidates[14].

Since the computation of our translation is straightforward and local, the size gives a bound on the computational effort. We observe:

| | $\mathcal{C}_\mathcal{G}$ | $\widehat{\mathcal{C}_\mathcal{G}}$ |
|---|---|---|
| $\lvert\bigcup_{p\in \boldsymbol{P}} S_p\rvert$ | $\lvert\mathcal{P}\rvert + \lvert\mathcal{P}\rvert \cdot 2^{\lvert\mathcal{T}\rvert}$ | $\lvert\mathcal{P}\rvert + \lvert\mathcal{P}\rvert \cdot 2^{\lvert\mathcal{T}\rvert} + \lvert In\rvert$ |
| $\lvert\Sigma\rvert$ | $\lvert\mathcal{T}\rvert + \lvert\mathcal{P}\rvert \cdot 2^{\lvert\mathcal{T}\rvert}$ | $\lvert\mathcal{T}\rvert + \lvert\mathcal{P}\rvert \cdot 2^{\lvert\mathcal{T}\rvert} + \lvert\mathcal{T}\rvert^2 \cdot 2^{\lvert\mathcal{T}\rvert}$ |

The size of $\mathcal{C}_\mathcal{G}$ is exponential in the number of transitions but linear in the number of places. $\widehat{\mathcal{C}_\mathcal{G}}$ comprises additional $\lightning$-actions and $\bot$-states. The $\lightning$-action result in an only quadratic blow-up of $|\Sigma|$, caused by building pairs of transitions. In $\mathcal{C}_\mathcal{G}$ we can avoid the exponential blow-up in the size of the alphabet by using a tree-like structure to choose the commitment sets. Our translation can hence be modified to comprise a polynomial number of actions and exponential number of local states.

It is also worth noting that for Petri games with a (constant) bound on the number of outgoing transitions both translations ($\mathcal{C}_\mathcal{G}$ and $\widehat{\mathcal{C}_\mathcal{G}}$) are of polynomial size.

## Lower Bound

A natural question to ask is whether we can do better than that, i.e., keep the blow-up polynomial all together. While this might be possible in general we can show that a polynomial-translation inevitably destroys the structure of the game, i.e., does either yield non-strategy-equivalent games, or increases the number of players. We prove this by giving a family of Petri games where every strategy-equivalent control game must have exponential size.

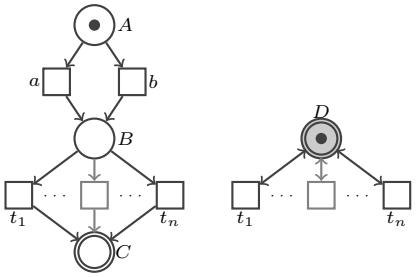



**Figure 5.14** Slices of a concurrency-preserving (reachability) Petri game family $\{\mathcal{G}_n\}_{n\in\mathbb{N}}$ where every strategy-equivalent control game is of exponential size.

Consider the Petri game family $\{\mathcal{G}_n\}_{n\in\mathbb{N}}$ obtained as the composition of the slices in Fig. 5.14. We fix any $n$ and refer to $\mathcal{G}_n$ as $\mathcal{G}$. In the initial marking of $\mathcal{G}$ both $a$ and $b$ can fire resulting in a marking $M = \{B, D\}$. From here all the transitions $t_1, \cdots, t_n$ are enabled. $\mathcal{G}$ is played between two players. Both of which possess different information, i.e., the first player (starting in $A$) knows whether $a$ or $b$ occurred, while the second (starting in $D$) does not. Only the second player can decide which of the transitions $t_1, \cdots, t_n$ should be possible. The decision of which of the transitions in $t_1, \cdots, t_n$ to allows can hence not be based on the occurrence of $a$ or $b$. Since a winning strategy for $\mathcal{G}$ can restrict any

[14] We remark at this point, that the more concise communication scheme of control games over Petri games allows us to hide additional complexity that is made explicit in Petri games. In a concurrency-preserving Petri game the size of the flow relation $\mathcal{F}$ is always polynomial in the number of places and transitions. In a control game the transition function $\{\delta_a\}_{a\in\Sigma}$, on the other hand, can be of exponential size in the number of local states. Our resulting game $\mathcal{C}_\mathcal{G}$ can, however, be described as the parallel composition of local automata. The local transitions relation is therefore again polynomial in $|\bigcup_{p\in \boldsymbol{P}} S_p|$ and $|\Sigma|$. For our analysis we hence restrict to the number of local states and actions in the alphabet.

combinations of $t_i$-transitions, every strategy-equivalent control game must admit controllers that can do the same. At the same time the decision, which of the $t_i$s to enable, cannot be based on the occurrence of $a$ or $b$, since this would imply a strategy for $\mathcal{G}$ that can do the same. Unlike Petri games that can natively express that the second player can restrict transitions $t_1, \cdots, t_n$, while the first one cannot (using system and environment places), control games are limited to controllable or uncontrollable actions. We show that this already results in exponentially many global states.

**Proving the lower Bound** For a strategy $\sigma$ for $\mathcal{G}$ we write $seq(\sigma)$ for the set of sequences admitted by $\sigma$. For every $\sigma$ it holds that $seq(\sigma) = \{\epsilon, a, b\} \cup \{a\,t, b\,t \mid t \in B\}$ for some $B \subseteq \{t_1, \cdots, t_n\}$. Conversely, for every such $B$ there exists a $\sigma$ s.t. $seq(\sigma)$ has exactly this form. In particular, a strategy cannot base the decision of what $t_i$s to enable on the occurrence of $a$ or $b$.

Consider any control game $\mathcal{C}$ that is strategy equivalent to $\mathcal{G}$. Due to $\mathcal{C}$ being a translation of $\mathcal{G}$, we assume $\mathcal{T} \subseteq \Sigma$. We treat $\mathcal{C}$ as a black box and deduce facts about it. A first step is to show that $a, b$ cannot be controlled by any controller, i.e., they are uncontrollable. Even though the statement looks fairly obvious, it requires some involved and careful reasoning[15].

**Lemma 19** $a$ and $b$ are uncontrollable.

**Proof** Choose the (winning) strategy $\sigma$ for $\mathcal{G}$ as the one that allows everything, i.e., $\mathcal{N}^\sigma = \mathcal{G}^{\mathfrak{U}}$. Let $\varrho_\sigma = \{f_p^{\varrho_\sigma}\}_{p \in \boldsymbol{P}}$ be a bisimilar controller for $\mathcal{C}$. There hence exists a relation $\approx_{\mathfrak{B}}$ with $In^{\mathcal{N}^\sigma} \approx_{\mathfrak{B}} \epsilon$.
Since $\varrho_\sigma$ is winning, every play must be finite. Now we consider every play $u$ in $Plays(\mathcal{C}, \varrho_\sigma)$ that only consist of $\tau$-actions and is maximal w.r.t. $\tau$-actions, i.e., $u$ cannot be extended by another $\tau$. By bisimulation we know that $In^{\mathcal{N}^\sigma} \approx_{\mathfrak{B}} u$. Since both $a$ and $b$ are possible from $In^{\mathcal{N}^\sigma}$, i.e., there is a marking $M$ with $In^{\mathcal{N}^\sigma} \,[\, a \,\rangle\, M$ and $In^{\mathcal{N}^\sigma} \,[\, b \,\rangle\, M$, we know that $a$ and $b$ must be the only extensions of such a play $u$. So $u\,a \in Plays(\mathcal{C}, \varrho_\sigma)$ and $u\,b \in Plays(\mathcal{C}, \varrho_\sigma)$. This holds for *every* play $u$ that solely consists of $\tau$-actions and is maximal w.r.t. them **(1)**. Now assume for contradiction and w.l.o.g. that $a$ is controllable. We build a slightly modified controller $\varrho' = \{f_p^{\varrho'}\}_{p \in \boldsymbol{P}}$ as follows:

$$f_p^{\varrho'}(u) = f_p^{\varrho_\sigma}(u) - \{a\}$$

$\varrho'$ behaves like $\varrho_\sigma$ but always forbids $a$.
As strategy-equivalence only considers winning strategies and controllers, our main objective is to show that $\varrho'$ is winning. It holds that $Plays(\mathcal{C}, \varrho') \subseteq Plays(\mathcal{C}, \varrho_\sigma)$. This alone does not allow us to conclude that $\varrho'$ is winning. It could happen that $\varrho'$ blocks action $a$ and therefore blocks itself from reaching a winning configuration. Suppose $u \in Plays(\mathcal{C}, \varrho') \subseteq Plays(\mathcal{C}, \varrho_\sigma)$ is any sequence that consist only of $\tau$-actions and is maximal w.r.t. $\tau$-actions, i.e., there is no $\tau$ with $u\,\tau \in Plays(\mathcal{C}, \varrho')$. Outside from always rejecting $a$, $\varrho'$ behaves like $\varrho_\sigma$. As $us$ is maximal w.r.z. $\tau$-actions in $\varrho'$, it is hence maximal w.r.t. $\tau$-actions for $\varrho_\sigma$, i.e., there is no $\tau$ with $u\,\tau \in Plays(\mathcal{C}, \varrho_\sigma)$. By **(1)** we now get that $u\,a \in Plays(\mathcal{C}, \varrho_\sigma)$ and $u\,b \in Plays(\mathcal{C}, \varrho_\sigma)$. While $\varrho'$ block $a$, we still conclude that $u\,b \in Plays(\mathcal{C}, \varrho')$. Starting in $u\,b$, $\varrho'$ again behaves like $\varrho_\sigma$, since after $u\,b$ there cannot be any $a$-action (by bisimulation). Therefore there is no maximal play in $Plays(\mathcal{C}, \varrho')$ that does not reach a winning configuration. $\varrho'$ is winning.
Since $\varrho'$ is winning, there exists a strategy $\sigma_{\varrho'}$ that is bisimilar to $\varrho'$. We can derive an easy contradiction: Let $u \in Plays(\mathcal{C}, \varrho')$ be any play that only consists of $\tau$-actions and is maximal w.r.t. them. It holds that $In^{\mathcal{N}^\sigma} \approx_{\mathfrak{B}} u$. By construction of $\varrho'$ we know that $u\,b$ is the only extension of $u$, i.e., no $\tau$ or $a$ is possible. This a contradiction since $In^{\mathcal{N}^\sigma} \,[\, a \,\rangle\, M$ for some $M$ as this is possible for any strategy for $\mathcal{G}$. ■

[15] Actually Lemma 19 is not needed for our lower bound. It is nevertheless crucial to get an idea of the proof used in the Lemma 20.

We can show next that all $t_i$-actions are uncontrollable. The insight is that if some of them were controllable, *all* processes can control them. There must be at least one process that can, from its local view, deduce whether $a$ or $b$ happened. This process can then base its decision of which $t_i$-actions to allow on the occurrence of $a$ or $b$. Such behavior cannot be achieved by a strategy for $\mathcal{G}$.

**Lemma 20** $t_1, \cdots, t_n$ are uncontrollable.

**Proof** Choose the strategy $\sigma$ for $\mathcal{G}$ as the one that allows everything, i.e., $\mathcal{N}^\sigma = \mathcal{G}^\mathfrak{U}$. It holds that $seq(\sigma) = \{\epsilon, a, b\} \cup \{a\,t, b\,t \mid t \in \{t_1, \cdots, t_n\}\}$. Let $\varrho_\sigma = \{f_p^{\varrho_\sigma}\}_{p \in \boldsymbol{P}}$ be a bisimilar winning controller for $\mathcal{C}$ that exists by assumption. Assume for contradiction and w.l.o.g. that $t_1$ is controllable.
We make an important observation: Any sequence of transitions in $\mathcal{G}$ or $\mathcal{N}^\sigma$ always begins with an $a$ or $b$ trailed by one of the $t_i$-actions. Since $\varrho_\sigma$ and $\sigma$ are bisimilar we conclude that for any play in $Plays(\mathcal{C}, \varrho_\sigma)$ the $t_i$-action must always trail the $a$ or $b$ action. To put it differently: Assume there is a play $u = u'\,t_1 \in Plays(\mathcal{C}, \varrho_\sigma)$, then there is an action $a$ or $b$ in $u'$. At every point where $t_1$ is executed there hence is some process in $dom(t_1)$ that can derive the occurrence of $a$ or $b$ from its local view. The process might not be the same on every execution, but at all times there is at least one.
We modify $\varrho_\sigma$ into a new controller $\varrho' = \{f_p^{\varrho'}\}_{p \in \boldsymbol{P}}$ as follows:

$$f_p^{\varrho'}(u) = \begin{cases} f_p^{\varrho_\sigma}(u) & \text{if } a \notin u \\ f_p^{\varrho_\sigma}(u) - \{t_1\} & \text{if } a \in u \end{cases}$$

Here $a \in u$ denotes that $a$ is an action in $u$. $\varrho'$ behaves just like $\varrho_\sigma$ with one difference: Whenever any process can deduce $a$ in its causal past it forbids $t_1$. Since there is at least one process that can deduce $a$ or $b$ we can conclude that $\varrho'$ never admits a sequence where $t_1$ is played after a previous $a$.
Similar to Lemma 19 the key is to argue that $\varrho'$ is a winning controller, i.e., blocking $t_1$ never results in a state from which no winning configuration can be reached. It holds that $Plays(\mathcal{C}, \varrho') \subseteq Plays(\mathcal{C}, \varrho_\sigma)$. By the same reasoning as for the previous lemma, any maximal play $u$ where no $t_i$-actions have been played can be extended by all $t_i$-actions, i.e., $(u\,t_1), \cdots, (u\,t_n) \in Plays(\mathcal{C}, \varrho_\sigma)$ (since $\sigma$ allows all $t_i$-transitions). Since $\varrho'$ only forbids $t_1$ we conclude that $(u\,t_2), \cdots, (u\,t_n) \in Plays(\mathcal{C}, \varrho')$. Bocking $t_1$ never blocks a winning state, so $\varrho'$ is winning.
Since $\mathcal{C}$ and $\mathcal{G}$ are strategy-equivalent there now is a strategy $\sigma_{\varrho'}$ for $\mathcal{G}$ that is bisimilar to $\varrho'$. Because of the bisimulation it is easy to see that

$$seq(\sigma_{\varrho'}) = \{\epsilon, a, b\} \cup \{a\,t_i \mid t_i \in \{t_2, \cdots, t_n\}\} \cup \{b\,t_i \mid t_i \in \{t_1, \cdots, t_n\}\}$$

Justified refusal forbids a strategy archiving this behavior, a contradiction. ■

Given this we can now show that there are exponentially many global states needed in $\mathcal{C}$. The idea is to simulate maximal $\tau$-sequences in the controller. The resulting *global* state should allow exactly all the actions in any subset of the $t_i$s. Since we know that the $t_i$ actions are uncontrollable the fact that exactly certain actions are enabled from a global state must be a "property" of the global state, i.e., it cannot be the result of a controller simply forbidding some $t_i$ actions.

**Lemma 21** For every $\emptyset \neq B \subseteq \{t_1, \cdots, t_n\}$ there is a *global* state $s_B = \{s_p\}_{p \in \boldsymbol{P}}$, s.t., for the set of actions $E$ that can fire from $s_B$ ($E = \{a \in \Sigma \mid \{s_p\}_{p \in dom(a)} \in domain(\delta_a)\}$) it holds that $E \cap \mathcal{T} = B$.

**Proof** Consider the (winning) strategy $\sigma$ s.t. $seq(\sigma) = \{\epsilon, a, b\} \cup \{a\,t, b\,t \mid t \in B\}$ and the bisimilar (winning) controller $\varrho_\sigma$. We know that $a$ is a $\sigma$-compatible sequence of transitions ($In^{\mathcal{N}^\sigma} [a\rangle M$ for some $M$), so by assumption there is a $\varrho_\sigma$-compatible play $u$ with $u = \tau^* a$ and $M \approx_{\mathfrak{B}} u$. We now extend the play $u$ as long as possible with $\tau$ actions. We obtain a play $u' \in Plays(\mathcal{C}_\mathcal{G}, \varrho_\sigma)$ s.t. $u'$ cannot be extend by another $\tau$-action. We can guarantee the existence of such a $u'$ since $\varrho_\sigma$ is winning and therefore does not admit infinite plays. It holds that $M \approx_{\mathfrak{B}} u'$. Since $\sigma$ does from $M$ allow exactly the transitions in $B$, $M \approx_{\mathfrak{B}} u'$ holds and $u'$ is maximal w.r.t. $\tau$-actions, we know that the possible extensions of $u'$ are exactly $B$. So

$$u'\,t_i \in Plays(\mathcal{C}_\mathcal{G}, \varrho_\sigma) \Leftrightarrow t_i \in B$$

Since all $t_i$-actions are uncontrollable (Lemma 20), we get that $u'\,t_i \in Plays(\mathcal{C}_\mathcal{G}, \varrho_\sigma)$ if and only if $u'\,t_i \in Plays(\mathcal{C}_\mathcal{G})$. So

$$u'\,t_i \in Plays(\mathcal{C}_\mathcal{G}) \Leftrightarrow t_i \in B$$

The global state $s_B = state(u')$ hence allows exactly the $t_i$-actions that are in $B$. ■

Consequently there must be exponentially many *global* states. For every strategy-equivalent control game $\mathcal{C}$ with a constant number of players there must hence be exponentially many local states.

**Theorem 2** There is a family of Petri games $\{\mathcal{G}_n\}_{n \in \mathbb{N}}$ with $|\mathcal{T}^{\mathcal{G}_n}| = n$, s.t., every strategy-equivalent control game (with an equal number of players) must have at least $\Omega(d^n)$ local states for $d > 1$.

**Proof** There are $\Omega(2^n)$ many sets $B \subseteq \{t_1, \cdots, t_n\}$. By the previous lemma any strategy-equivalent control games must hence have $\Omega(2^n)$ many global states. For any control game (asynchronous automaton) with two processes $p_1, p_2$ there are at most $|S_{p_1}| \cdot |S_{p_2}|$ many global states. Hence one of the two processes must have $\Omega((\sqrt{2})^n)$ many local states. ■

While our translation (Sec. 5.1) shows that the difference between controllable and uncontrollable actions vs. system and environment places can be overcome, the lower bound shows an intrinsic difficulty to do so. We can roughly analyse the main limitation of control games: A partition of the places (as in Petri games) allows for a direct control of which player is able to control certain moves. Using this formalism a transition can involve player of which some can restrict it, while others cannot. In a control game, on the other hand, an action is either controllable or uncontrollable and can therefore be restricted by either all or none of the involved processes.

## 5.5 Generalisation to Concurrency Preserving Petri Games

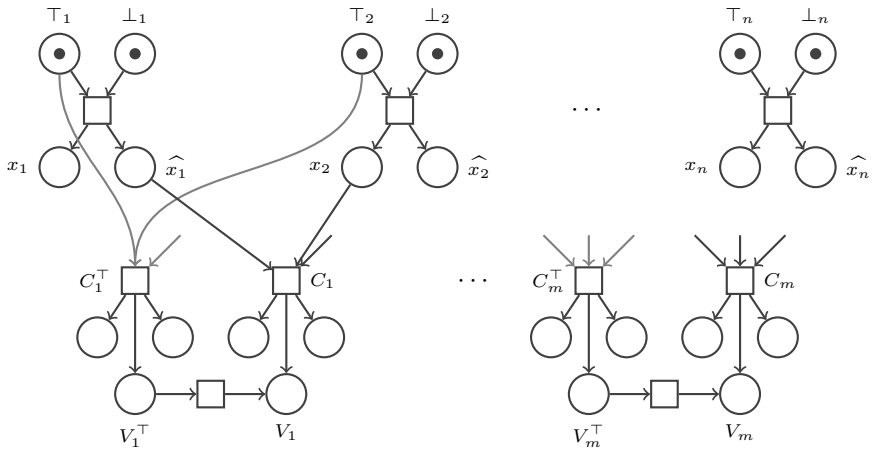
The translation as presented so far is limited to sliceable games, restricting its applicability. In this section we introduce introduce a new mechanism to distribute a game and thereby generalize our translation to all *concurrency-preserving* games. Before we proceed to our generalisation, we once more, contemplate on the concepts of slices. Figure 3.1 in Chapter 4 already showed that very simple nets (and hence games) must not be sliceable. We can show that the class of sliceable nets is actually fairly interesting and that deciding sliceability is not as easy as one might have thought.

**Proposition 3** Deciding whether a concurrency-preserving, safe, reachable Petri net is sliceable is NP-complete.

**Proof** A given slice-distribution (encoded as a partition of the places) can be verified efficiently so it is easy to see that the problem is in NP.
For the hardness we reduce from 3-SAT. Recall that 3-SAT is the problem of deciding whether a given propositional CNF formula, where each clause contains exactly 3 literals, is satisfiable. It is known to be NP-hard.
Fix a CNF-formula $\phi$ with

$$\phi = C_1 \wedge \cdots \wedge C_m$$

where each clause has the form $C_i = (L_i^1 \vee L_i^2 \vee L_i^3)$. Let $x_1, \cdots, x_i$ be the propositional variables in $\phi$
From $\phi$ we build a Petri net $\mathcal{N}_\phi$ as follows:




For every variable $x_i$ we create four places $x_i$, $\widehat{x_i}$, $\top_i$ and $\bot_i$, place a token on $\top_i$ and $\bot_i$ and add a transition that consumes tokens from the $\top_i$ and $\bot_i$ place and puts tokens on the $x_i$ and $\widehat{x_i}$ place. For each clause we create a new transition $C_i$ who's precondition is exactly the set that contains the three literals of the clause, i.e., the $x_i$ or $\widehat{x_i}$ place. For each of these transitions $C_i$ there are three outgoing places, two unimportant ones (only to stay concurrency-preserving) and one special one, $V_i$. For each clause we, furthermore, create a transition $C_i^\top$ who's precondition are the $\top$ places of each variable in the clause. As for the $C_i$ transition we add two unimportant outgoing places (only to stay concurrency-preserving) and one special one, $V_i^\top$. We then connect $V_i^\top$ with a transition to $V_i$.
This reduction with $4n + 6m$ places and $n + 3m$ transitions can be computed efficiently. It is easy to check that the net is reachable, safe and concurrency-preserving. In every slice distribution the places $x_i$ and $\widehat{x_i}$ must be split between the slices of $\top_i$ and $\bot_i$. By putting the variables places in slices we implicitly build an assignment for the variables where literals in the $\top$ slice are set to true and in the $\bot$ slice to false. The remaining construction guarantees that each clause contains one variable that is in the $\top$-slice. We prove that $\phi$ is satisfiable if and only if $\mathcal{N}_\phi$ has an acyclic distribution:
$\Rightarrow$: Let $\phi$ be satisfiable and $h : \{x_1, \cdots, x_n\} \to \{true, false\}$ a satisfying assignment.
For each variable $x_i$ that is mapped to *true* we put the $x_i$ place in the same slice as the $\top_i$ place. Otherwise put $x_i$ in the same slice as the $\bot_i$ place. Since $h$ is a satisfying assignment every clause contains a true literal. For each $C_i$ transition we can hence put the $V_i$ place in the same slice as the literal of that clause that is satisfied. Since this literal is in the same slice as the $\top$-place of the corresponding variable, we can put $V_i^\top$ in the same slice as $V_i$. This already forms a valid slice distribution. $\mathcal{N}_\phi$ is sliceable.

$\Leftarrow$: Now suppose $\mathcal{N}_\phi$ has a valid distribution. It is easy to see that $x_i$ and $\widehat{x_i}$ cannot be in the same slice. For a valid distribution we know that $V_i$ and $V_i^\top$ must be in the same slice. As the $V_i^\top$ place is always in the slice of one of the $\top$ places of the variable, the $V_i$ place must also be in the same slice as one of the $\top$ places. For every clause there hence is one literal in the same slice as the $\top$-place. By setting every variable $x_i$ to *true* iff the $x_i$ place is in the same slice as $\top_i$ we hence obtain a satisfying assignment. $\phi$ is satisfiable. ■

## Concurrency preserving Petri Games

We can now turn back and try to generalize our translation to a broader class. We can observe that the notion of slices is too strict for our purposes: Our translation requires to distribute the global movement of the Petri game in local behavior. A *partition* of the places (as prescribed by slice distributions) is not necessarily needed. Requiring such a partition is what enables proofs as the one above to work and hence limit the applicability of slices.

**Singular Net Distribution** We introduce the new concepts of *singular nets* (SN) and *singular net distributions* (SND). A singular net is similar to a slice but does not partition the places. Instead it is equipped with a labelling function $\pi$ assigning to each place and transition in the singular net a place or transition in the original net. Similar to a branching process this allows us to split up nodes into equally labelled ones and therefore enables us to assign the same place (copies of that place) to multiple tokens. Instead of dismantling a Petri net into a distribution of slices, we consider a singular net distribution. This is a composition of singular nets that show the same behavior as the original net: Every transition in the original Petri net can be matched with an equally labelled transition between the singular nets and vice versa. Similar to a slice distribution we can hence dismantle the global behavior of a Petri net, into local behavior of individual tokens. We later see how our translation can be modified to work with SNs instead of slices.

Even before giving a formal description we can consider the example in Fig. 5.15. The Petri net in (a) comprises three tokens of which two reside on the same place. As the net is not safe it is not sliceable. In (b) and (c) two possible singular net distributions of (a) are given. The black label (annotated with a hat) is the name of the node, whereas the gray label is the one given by $\pi$. The singular nets share transitions. If we, e.g., consider the SND in (b), the labelling of the initial marking agrees with the initial marking of (a) and both transitions $a$ and $b$ can be matched by some copy $(\hat{a_1}, \hat{a_2})$. By observing the SNDs in both (b) and (c) it becomes clear that both are valid distributions of the behavior in (a).

Throughout this section let $\mathcal{N}$ be a finite, concurrency-preserving Petri net. We now proceed and give a formal description of both SN and SNDs.

**Definition** A *singular net* (SN) of $\mathcal{N}$ is a pair $(\varsigma, \pi)$ where $\varsigma = (\mathcal{P}^\varsigma, \mathcal{T}^\varsigma, \mathcal{F}^\varsigma, In^\varsigma)$ is a Petri net satisfying

$$|In^\varsigma| = 1 \text{ and } \forall t \in \mathcal{T}^\varsigma : |pre(t)| = |post(t)| = 1$$

and $\pi : \mathcal{P}^\varsigma \cup \mathcal{T}^\varsigma \to \mathcal{P}^\mathcal{N} \cup \mathcal{T}^\mathcal{N}$ is a mapping with the following properties:

**(1)** $\pi(\mathcal{P}^\varsigma) \subseteq \mathcal{P}^\mathcal{N}$ and $\pi(\mathcal{T}^\varsigma) \subseteq \mathcal{T}^\mathcal{N}$ **(2)** $\forall q_1, q_2 \in \mathcal{P}^\varsigma : \pi(q_1) \neq \pi(q_2)$

**(3)** $\pi(In^\varsigma) \subseteq In^\mathcal{N}$ **(4)** $\forall q \in \mathcal{P}^\varsigma : post^\mathcal{N}(\pi(q)) \subseteq \pi(\mathcal{T}^\varsigma)$

**(5)** $\forall x, y \in \mathcal{P}^\varsigma \cup \mathcal{T}^\varsigma : (x, y) \in \mathcal{F}^\varsigma \Leftrightarrow (\pi(x), \pi(y)) \in \mathcal{F}^\mathcal{N}$

A singular net can be thought of as a generalized slice. The underlying net describes the movement of a single token. Instead of viewing it as a subnet of $\mathcal{N}$ (as we have done fro slices) we label it using $\pi$. This labelling should satisfy five properties, most of which correspond to proprieties lifted from the definition of a slice: $\pi$ must respect the node type **(1)** and copy each place at most once **(2)**. Singular nets of finite nets are hence finite. The initial marking must be

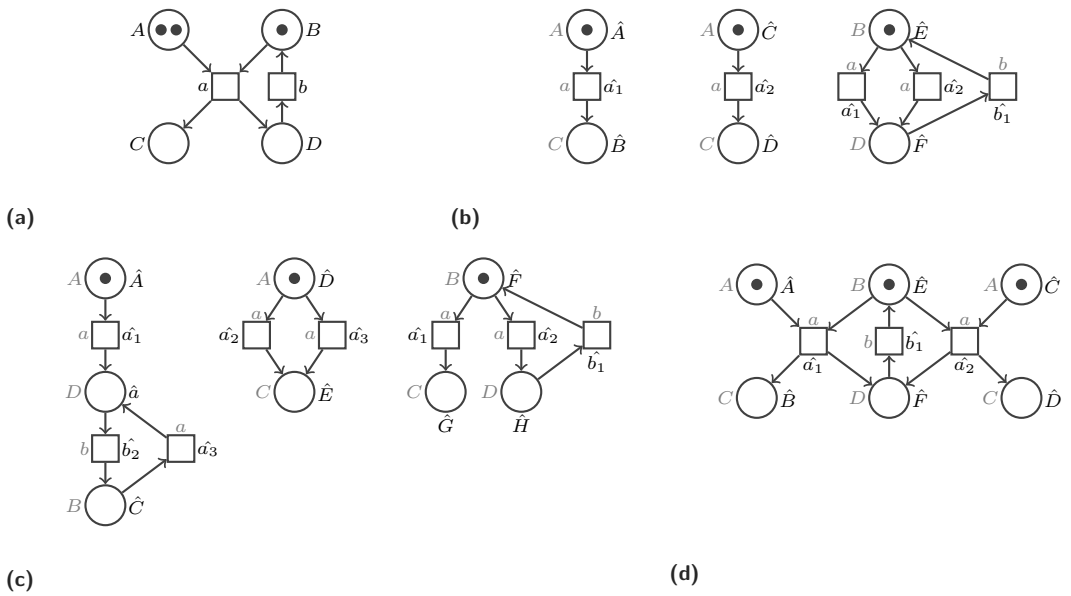



**Figure 5.15** A Petri game (a) and two possible distributions in singular nets (b) and (c). In (b) and (c), labels of the SND ($\pi$) are given in gray. The label of each node in a SND is annotated with a hat and labelled in black. Note that transitions can be shared between SNs. In (d) the composition of the SND in (b) is depicted.

labelled within the initial marking of $\mathcal{N}$ **(3)**. Similar to the definition of slices we require that all transitions leaving the label of some place are represented by at least one copy **(4)**. Lastly, the flow relation adds flow between to nodes if and only if there is some between the labels of the nodes in $\mathcal{N}$ **(5)**.

Singular nets are, similar to slices, defined as nets describing behavior of individual token. To, in the end, model global behavior, we want to compose multiple singular nets to obtain a description of a system involving more than one player.

**Definition** If $\mathcal{N}$ is a Petri net and $\boldsymbol{S} = \{(\varsigma_i, \pi_i)\}_{i\in\mathfrak{I}}$ with $\varsigma_i = (\mathcal{P}^i, \mathcal{T}^i, \mathcal{F}^i, In^i)$ is a finite family of singular nets for $\mathcal{N}$. We call $\boldsymbol{S}$ *compatible* if

$$\mathcal{P}^{\varsigma_i} \cap \mathcal{P}^{\varsigma_j} = \emptyset \text{ for all } i, j \in \mathfrak{I} \text{ with } i \neq j$$

and

$$\forall t : t \in \mathcal{T}^{\varsigma_i} \cap \mathcal{T}^{\varsigma_j} \Rightarrow \pi_{\varsigma_i}(t) = \pi_{\varsigma_j}(t)$$

If $\boldsymbol{S}$ is compatible we define the *composition* of $\boldsymbol{S}$ as the pair $(\langle\|_{\boldsymbol{S}}\rangle, \pi_{\langle\|_{\boldsymbol{S}}\rangle})$ where

$$\langle\|_{\boldsymbol{S}}\rangle = (\mathcal{P}^{\langle\|s\rangle}, \mathcal{T}^{\langle\|s\rangle}, \mathcal{F}^{\langle\|s\rangle}, In^{\langle\|s\rangle})$$

with $\mathcal{P}^{\langle\|s\rangle} = \biguplus_{i\in\mathfrak{I}} \mathcal{P}^i$, $\mathcal{T}^{\langle\|s\rangle} = \bigcup_{i\in\mathfrak{I}} \mathcal{T}^i$, $\mathcal{F}^{\langle\|s\rangle} = \biguplus_{i\in\mathfrak{I}} \mathcal{F}^i$ and $In^{\langle\|s\rangle} = \biguplus_{i\in\mathfrak{I}} In^i$ and

$$\pi_{\langle\|_{\boldsymbol{S}}\rangle} = \bigcup_{i\in\mathfrak{I}} \pi_i$$

As for slices we require a family of singular nets to contain disjoint sets of places. As each SN is furthermore labelled by $\pi$, we require that shared transitions are labelled equally among all singular nets. $\langle\|_{\boldsymbol{S}}\rangle$ is then defined as the Petri net obtained by taking the union of places, transitions, flow and initial marking. Since only transitions can be shared all unions except for them are disjoint. As $\mathcal{N}$ is compatible we know that for all transitions the label agrees in all

SNs. We can hence label the nodes in $\langle \|_{\boldsymbol{S}} \rangle$ with nodes in $\mathcal{N}$, i.e., design $\pi_{\langle \|_{\boldsymbol{S}} \rangle}$ as the union of all individual labelling functions. Note that unlike for slices the composition $\langle \|_{\boldsymbol{S}} \rangle$ in general differs from $\mathcal{N}$. In Fig. 5.15 (d) the composition of the SN-family in (b) is depicted. The $\pi$-label of the composition is given in gray.

The labelling of a SND allows us to split up places and transitions. We want to distribute a Petri net in a family of singular nets, that together shows the same behavior as the Petri net. We can hence define what a family of singular nets should suffices to be a valid *distribution* of a net:

**Definition** A *singular net distribution* (SND) for Petri net $\mathcal{N}$ is a compatible family $\boldsymbol{S}$ of singular nets for $\mathcal{N}$ where the composition $(\langle \|_{\boldsymbol{S}} \rangle, \pi_{\langle \|_{\boldsymbol{S}} \rangle})$ fulfils:

**(1)** $\pi(In^{\langle \|_S \rangle}) = In^{\mathcal{N}}$

**(2)** For every transition $t \in \mathcal{T}^{\langle \|_S \rangle}$, $\pi(pre^{\langle \|_S \rangle}(t)) = pre^{\mathcal{N}}(\pi(t))$, and $\pi(post^{\langle \|_S \rangle}(t))) = post^{\mathcal{N}}(\pi(t))$

**(3)** For every $t_1, t_2 \in \mathcal{T}^{\langle \|_S \rangle}$ with $pre^{\langle \|_S \rangle}(t_1) = pre^{\langle \|_S \rangle}(t_2)$ and $\pi(t_1) = \pi(t_2)$ it holds that $t_1 = t_2$

**(4)** For reachable marking $M \in \mathcal{R}(\langle \|_{\boldsymbol{S}} \rangle)$ and subset $C \subseteq M$ with $\pi(C) = pre^{\mathcal{N}}(t)$ for some $t \in \mathcal{T}^{\mathcal{N}}$ there exists a transition $t' \in \mathcal{T}^{\langle \|_S \rangle}$ with $\pi(t') = t$ and $pre^{\langle \|_S \rangle}(t') = A$

A singular net distribution is a compatible family of singular nets, i.e., a family with disjoint places and equally labelled shared transitions. The additional restrictions guarantee that the composition of the SNs shows the same behavior as the original net. They are reminiscent of the definition of a branching process and unfolding. Restriction **(1)** requires the initial marking of $\langle \|_{\boldsymbol{S}} \rangle$ to be labelled within the initial marking of $\mathcal{N}$, whereas **(2)** requires the composition to preserve the structure on transitions. Together **(1)** and **(2)** state that $\pi_{\langle \|_{\boldsymbol{S}} \rangle}$ is an initial homomorphism from $\langle \|_{\boldsymbol{S}} \rangle$ to $\mathcal{N}$. As for a branching process, **(3)** requires $\pi_{\langle \|_{\boldsymbol{S}} \rangle}$ to be injective on transitions with the same precondition: Equally labelled transitions must occur from distinct situations. A SND is almost identical to a branching process with the exception of not requiring an underlying occurrence net and, furthermore, being described in terms of local token movements. Lastly, requirement **(4)** is similar to the one found in the definition of an unfolding. It is a maximality criterion that requires that for every situation where there are tokens on places in $C$ that every transition possible from $\pi_{\langle \|_{\boldsymbol{S}} \rangle}(C)$ is matched by some copy. While we can split up places in a SND, **(4)** requires us to still add transitions from every possible combination of the new copies. Both families of singular nets in Fig. 5.15 (b) and (c) form singular net distributions of the net in (a).

Note that our notion of a singular net distribution agrees with slice distributions if we enforce to have only one copy of each place. In this case we can choose $\pi$ as the identity. Every sliceable net has a SND.

**Properties of SNDs** A SND is defined as a family of singular nets such that their composition is both structure-preserving **(2)** and at the same time capture all behavior **(4)**. It is easy to see that a SND describes the exact behavior of a Petri net.

**Corollary** $\mathcal{R}(\mathcal{N}) = \pi(\mathcal{R}(\langle \|_{\boldsymbol{S}} \rangle))$.

Our main motivation for defining SNs and SNDs is to generalize our previous translation to allow for a broader class of games. We already remarked that every sliceable net has a SND. In Fig. 5.15 we saw that even some non-sliceable nets have a SND. Nets with SND are thus a *strict superset* of sliceable nets. The next theorem shows, that the class of Petri nets, that can be distributed in singular nets, can be characterised precisely: It is exactly the class of concurrency-preserving Petri nets.

**Proposition 4** Every finite, concurrency-preserving Petri net has a SND.

**Proof** We present a constructive proof. Given a Petri net $\mathcal{N} = (\mathcal{P}^{\mathcal{N}}, \mathcal{T}^{\mathcal{N}}, \mathcal{F}^{\mathcal{N}}, In^{\mathcal{N}})$ we build $|In^{\mathcal{N}}|$ many singular nets. Each of these nets initially consists of a copy of the places in $\mathcal{N}$, i.e., $|In^{\mathcal{N}}|$-many copies of $\mathcal{N}$ without any transitions. So $\boldsymbol{S} = \{(\varsigma_1, \pi_i), \cdots, (\varsigma_{|In^{\mathcal{N}}|}, \pi_{|In^{\mathcal{N}}|})\}$ where $\varsigma_i = (\mathcal{P}^i, \mathcal{T}^i, \mathcal{F}^i, In^i)$ with $\mathcal{P}^i = \{q_i \mid q \in \mathcal{P}^{\mathcal{N}}\}$ Define $\pi_i(q_i) = q$. From each $\varsigma_i$ we select a single place and add it to a set $D$ s.t. $\pi(D) = In^{\mathcal{N}}$ (this is always possible). We put one token on each of these selected places, resulting in one token in the initial marking of each SN.
Now define $(\langle \| \boldsymbol{S} \rangle, \pi_{\langle \| \boldsymbol{S} \rangle})$ as the composition of the singular nets. We incrementally add transitions to the SNs: We iterate over every reachable marking $M$ in $\langle \| \boldsymbol{S} \rangle$ and consider every set $C \subseteq M$ where $\pi_{\langle \| \boldsymbol{S} \rangle}(C) = pre^{\mathcal{N}}(t)$ for some transition $t$ and there is no $t'$ with $\pi_{\langle \| \boldsymbol{S} \rangle}(t') = t$ and $pre^{\langle \| \boldsymbol{S} \rangle}(t') = C$. The set of SNs involved in $C$ is $\nabla_C = \{i \mid \mathcal{P}^i \cap C \neq \emptyset\}$ We create a new transition $t'$, define $\pi_{\langle \| \boldsymbol{S} \rangle}(t') = t$ and add it to all SNs whose place is contained in $C$.

$$\mathcal{T}^i = \begin{cases} \mathcal{T}^i & \text{if } i \notin \nabla_C \\ \mathcal{T}^i \cup \{t'\} & \text{if } i \in \nabla_C \end{cases}$$

We extend the flow of every SN $i$ in $\nabla_C$ s.t. $pre^{\varsigma_i}(t') = C \cap \mathcal{P}^i$. We pick a set of places $C'$ s.t. $\nabla_{C'} = \nabla_C$ and $\pi_{\langle \| \boldsymbol{S} \rangle}(C') = post^{\mathcal{N}}(t)$. We hence assign for each involved SN a place such that the label of $C'$ agrees with $post^{\mathcal{N}}(t)$. We note that there might be many such combinations but there is at least one. We extend the flow of all $i$ in $\nabla_C$ ($\nabla_{C'}$) such that $post^{\varsigma_i}(t') = C' \cap \mathcal{P}^i$. Afterwards, we recompute the composition $(\langle \| \boldsymbol{S} \rangle, \pi_{\langle \| \boldsymbol{S} \rangle})$ with the newly added transitions and repeat until no more transitions can be added.

We iterate this and thereby add more and more transitions. Since we deal with a finite number of places and transitions the construction terminates. Since we add exactly the transitions required in a SND it can easily be checked that each net is a singular net and the resulting family a singular net distribution. ■

## Branching processes of SNDs

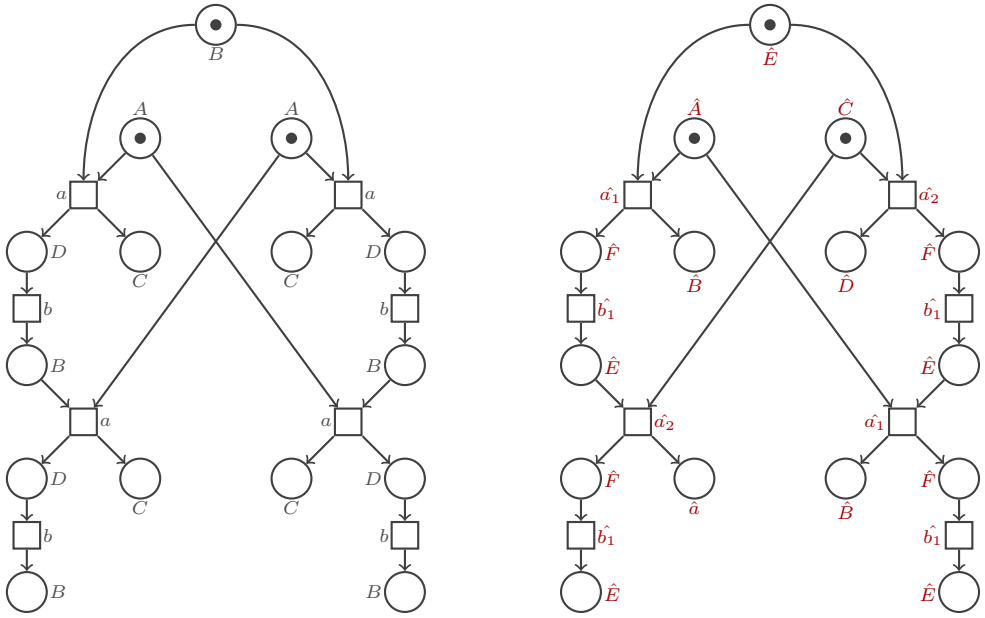
In the long run we want to extend SND to Petri games and use it for our translation. Since strategies are defined in terms of branching processes we begin by comparing branching processes for a SND with ones for the original net. Assume $\mathcal{N}$ is a Petri net, $\boldsymbol{S}$ is a SND for $\mathcal{N}$ and $(\langle \| \boldsymbol{S} \rangle, \pi)$ the composition of $\boldsymbol{S}$. We analyse and compare possible branching processes for both $\mathcal{N}$ and $\langle \| \boldsymbol{S} \rangle$.

Let $\iota = (\mathcal{N}^\iota, \lambda)$ be a branching process for $\mathcal{N}$ and $\iota_{\boldsymbol{S}} = (\mathcal{N}^\iota_{\boldsymbol{S}}, \lambda_{\boldsymbol{S}})$ a branching process for $\langle \| \boldsymbol{S} \rangle$. $\lambda$ labels the nodes from $\mathcal{N}^\iota$ with nodes in $\mathcal{N}$, while $\lambda_{\boldsymbol{S}}$ labels the nodes of $\mathcal{N}^\iota_{\boldsymbol{S}}$ in $\langle \| \boldsymbol{S} \rangle$. All nodes in $\langle \| \boldsymbol{S} \rangle$ are themself, by $\pi$, labelled in $\mathcal{N}$. A branching process of $\langle \| \boldsymbol{S} \rangle$ hence has finer label; instead of being labelled in nodes from $\mathcal{N}$ directly it is labelled in an intermediate entity, namely $\langle \| \boldsymbol{S} \rangle$, that is itself labelled in $\mathcal{N}$. We define

$$\iota \leftrightsquigarrow \iota_{\boldsymbol{S}} \;\Leftrightarrow\; \mathcal{N}^\iota = \mathcal{N}^\iota_{\boldsymbol{S}} \;\wedge\; \lambda = \pi \circ \lambda_{\boldsymbol{S}}$$

$\leftrightsquigarrow$ relates a branching processes for $\langle \| \boldsymbol{S} \rangle$ and $\mathcal{N}$ if the underlying occurrence net is identical and the labelling of $\iota_{\boldsymbol{S}}$ is finer than that of $\iota$, i.e., agrees when made coarser by applying $\pi$. It is intuitive that $\leftrightsquigarrow$-related branching processes describe equivalent restrictions of the Petri net.

For an example we come back to the Petri net from Fig. 5.15. Recall that the unfolding of a branching process is a branching process itself. In Fig. 5.16 (a) the unfolding of the Petri net in Fig. 5.15 (a) is depicted if the gray labels are used. Fig. 5.16 (b) shows the unfolding of the parallel composition in Fig. 5.15 (d). We note that both unfoldings are related by $\leftrightsquigarrow$. The labelling of the unfolding in (b) is finer, as we can always recover the gray label by applying $\pi$ to the red one.




**(a)** **(b)**

**Figure 5.16** Including the red label the unfolding of the SND in Fig. 5.15 (b) is depicted. Using the gray label it is the unfolding of the Petri net in Fig. 5.15 (a).

We can show that, as a SND preserves both the structure and add every transition possible, that:

> **Corollary** For every branching process $\iota_{\boldsymbol{S}}$ of $\langle \|_{\boldsymbol{S}} \rangle$ there exists a branching process $\iota$ for $\mathcal{N}$ with $\iota \leftrightsquigarrow \iota_{\boldsymbol{S}}$
> For every branching process $\iota$ of $\mathcal{N}$ there exists a branching process $\iota_{\boldsymbol{S}}$ for $\langle \|_{\boldsymbol{S}} \rangle$ with $\iota \leftrightsquigarrow \iota_{\boldsymbol{S}}$

For every branching process of either $\mathcal{N}$ or $\langle \|_{\boldsymbol{S}} \rangle$ there hence exists an equivalent one for $\langle \|_{\boldsymbol{S}} \rangle$ or $\mathcal{N}$, i.e., one with a finer or coarser labelling.

## Translating Games using SND

We can now adopt the previous concepts to Petri games, i.e., mark the places in a SN as either system and environment, and require that $\pi$ respects this distributions. Let $\mathcal{G}$ be a concurrency-preserving Petri game, $\boldsymbol{S}$ a SND for $\mathcal{G}$ and $(\langle \|_{\boldsymbol{S}} \rangle, \pi)$ the composition of $\boldsymbol{S}$. While for every *branching process* for $\mathcal{G}$ there exists an equivalent (defined by $\leftrightsquigarrow$) branching process for $\langle \|_{\boldsymbol{S}} \rangle$ and vice versa, this does not hold for *strategies*. It still holds that for every strategy of $\mathcal{G}$ there exists an equivalent one for $\langle \|_{\boldsymbol{S}} \rangle$, but the reverse does not hold in general: A strategy for $\langle \|_{\boldsymbol{S}} \rangle$ can distinguish between copies of transition even though they have the same $\pi$-label (i.e., belong to the same transition in $\mathcal{G}$). Since a SND splits up transitions a strategy for the composition can be more restricting without violating justified refusal. There even exist games where the composition of a SND has a winning strategy even though the original game has not.

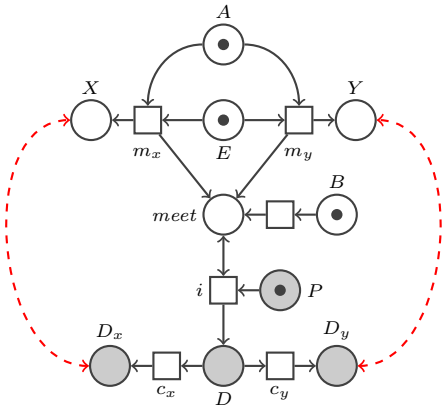



**Figure 5.17** Petri Game without a winning strategy. The player starting in $P$ should copy the decision made by the player in $E$, indicated with the red-arrows.

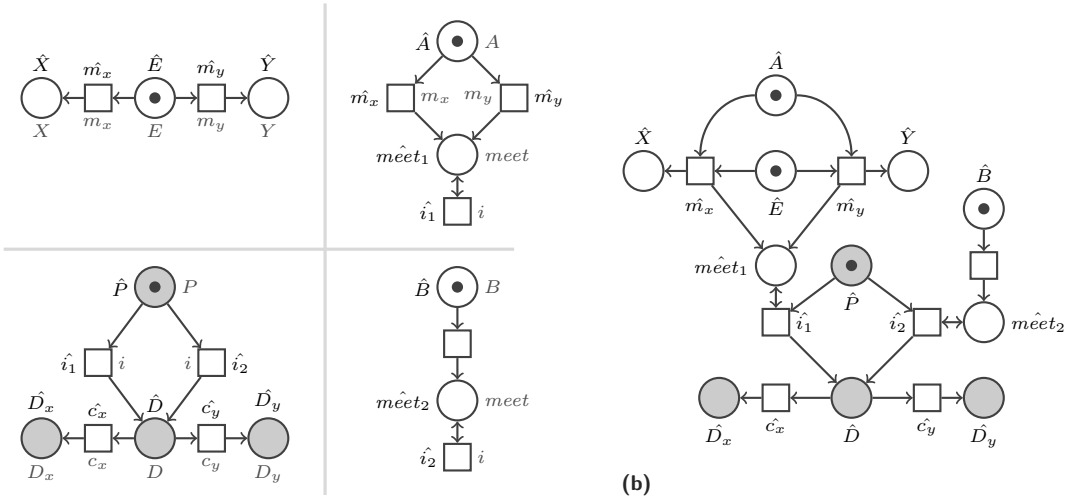



**Figure 5.18** A singular net distribution for the Petri game in Fig. 5.17 (a) and the composition of the distribution in (b). The black label is the name of the node, while the gray label is the one given by $\pi$. To aid readability the name of nodes in the SND are annotated with a hat and the $\pi$-label is omitted in (b).

To have an example for this, consider the Petri game in Fig. 5.17. It comprises four player: An environment that generates inputs starting in $E$, two dummy players starting in $A$ and $B$ as well as a system player starting in $P$. The player starting in $E$ can use transitions $m_x$ or $m_y$ and move to $X$ or $Y$ and thereby synchronizes with the dummy player in $A$. Upon synchronization $A$ hence moves to the *meet* place and the dummy player in $B$ moves there directly. The system player that is initially in place $P$ can synchronize with a token on *meet* on $i$ and afterwards use $c_x$ or $c_y$ to move to $D_x$ or $D_y$. To win the game the system player should copy the decision of the environment, i.e., move to $D_x$ iff the $E$ moves to $X$. This wining criterion can be expressed in either reachability and safety games. This game has *no* winning strategy: Both dummy players do not possess the same information since only the one starting in $A$ knows the decision that needs to be copied. To copy the player from $E$ reliably, the system player in $P$ needs to share transition $i$ with the player starting in $A$ since this is the only source of the much needed information. Communication with the player from $B$ does not provide any relevant information. Justified refusal, however, prohibits strategies that can guarantee that communication occurs with the token starting in $A$ and not with the one from $B$.

A possible singular net distribution of the Petri game in Fig. 5.17 is depicted in Fig. 5.18 (a). The gray label is the one given by $\pi$. The name of the node is depicted in black where each name is equipped with a hat to aid readability. The place *meet* is split up into two places $\hat{meet}_1$ and $\hat{meet}_2$. The transition $i$ is split up into $\hat{i_1}$ and $\hat{i_2}$. Fig. 5.18(b) delineates the composition of the SND in (a). To aid readability the $\pi$-label is omitted. Unlike the initial game from Fig. 5.17 the composition (b) has a winning strategy, since a strategy could forbid $\hat{i_2}$ while allowing $\hat{i_1}$ without violating justified refusal. When applying the coarser label to this winning strategy (i.e., apply $\pi$ point wise), the resulting branching process is no strategy for the original game.

While we cannot find equivalent strategies between $\mathcal{G}$ and $\langle \|_{\boldsymbol{S}} \rangle$, we can, however, find equivalent strategies, if strategies for $\langle \|_{\boldsymbol{S}} \rangle$ do not distinguish between equally $\pi$-labelled transitions. This motivates the following definition:

**Definition** A strategy $\sigma = (\mathcal{N}^\sigma, \lambda)$ for $\langle \|_{\boldsymbol{S}} \rangle$ is $\pi$-*insensitive*, if for any pairwise concurrent set of places $C$ with $\lambda[C] = pre^{\mathcal{G}}(t)$ for some transition $t$ there either is a transition $t'$ with $\lambda(t') = t$ and $pre^{\mathcal{N}^\sigma}(t') = C$ or there is a system place $q \in C \cap \lambda^{-1}[\mathcal{P}_{\mathcal{S}}]$ with $\pi(t) \notin \pi(\lambda[post^\sigma(q)])$

The definition is almost identical to the one of a strategy. While the original justified refusal requires that every transition that is not added to the strategy must be uniformly forbidden by a system place, in a $\pi$-insensitive strategy there must be a system place that uniformly forbids *all* transitions with the same $\pi$-label. Even though a transition is duplicated, a $\pi$-insensitive strategy considers all transitions with the same label as identical. The interested reader is advised to check that the composition in Fig. 5.18 (b) has no winning strategy that is $\pi$-insensitive.

We can show that if a strategy for $\langle\|_{\boldsymbol{S}}\rangle$ is $\pi$-insensitive and we apply the coarse label, the resulting branching process fulfils justified refusal, i.e., is a strategy:

> **Corollary** If $\sigma = (\mathcal{N}^\sigma, \lambda)$ is a $\pi$-insensitive strategy for $\langle\|_{\boldsymbol{S}}\rangle$ then $\sigma' = (\mathcal{N}^\sigma, \pi \circ \lambda)$ is a strategy for $\mathcal{G}$.

**Extending the Translation** We can do a similar translation as in Sec. 5.1 but work with singular net distributions instead of slice distributions by treating SNs as slices, i.e., ignore the $\pi$ label.

In Fig. 5.19 (a) the translated automaton for the SND from Fig. 5.18 is depicted. We already saw that a composition of a SND might have a winning strategy even though the original game has not (cf. Fig. 5.18 (b)). If we build our translation from a SND we run into the same problem: We give potential controllers to much power, by allowing them to distinguish equally $\pi$-labelled transitions (using their commitment sets) and therefore restrict the behavior in a way that the strategy of the Petri game cannot. For example, the control game in Fig. 5.19 (a) has a winning controller: As in composition of the SND, a controller can distinguish between $\hat{a_1}$ and $\hat{a_2}$ and therefore enforce communication with the player that possesses the information needed to win the game.

We fix this by modifying our translation slightly: We restrict the commitment sets for each process to transitions in the original game instead of the copies in the SND[16]. From such a commitment set all copies of a transition in the set are allowed. With the coarser commitment sets a controller can no longer distinguish equally labelled transitions and has to allow either all copies of a transition or none. If we translate the singular net distribution from Fig. 5.18 with the modified translation, the fourth singular net yields the process in Fig. 5.19 (b). If we substitute this process into the overall control game in (a) the resulting control game has no longer a winning controller, as $\hat{i_1}$ and $\hat{i_2}$ can no longer be restricted separately.

**Translating Strategies to Controllers** Given a winning strategy $\sigma$ for $\mathcal{G}$ we outline that there exist a bisimilar winning controller for the modified $\mathcal{C}_\mathcal{G}$. We can refine the labels of $\sigma$ to obtain a strategy $\sigma_{\boldsymbol{S}}$ for $\langle\|_{\boldsymbol{S}}\rangle$. It holds that $\sigma \leftrightsquigarrow \sigma_{\boldsymbol{S}}$. Since $\sigma$ satisfies justified refusal we get that $\sigma_{\boldsymbol{S}}$ is $\pi$-insensitive, i.e., if a place forbids a transition it forbids all transitions with the same $\pi$-label. We can now do the same controller construction as in Sec. 5.3.3 on the strategy $\sigma_{\boldsymbol{S}}$. Since $\sigma_{\boldsymbol{S}}$ is $\pi$-insensitive every transition that is not added must be forbidden together with all equally labelled transitions. The controller can hence choose an appropriate commitment set, even though the selection of sets does not allow to distinguish equally labelled transitions. As $\sigma \leftrightsquigarrow \sigma_{\boldsymbol{S}}$ it is easy to see that the obtained controller and $\sigma$ are bisimilar[17].

**Translating Controllers to Strategies** Given a controller for the modified $\mathcal{C}_\mathcal{G}$, we can construct a bisimilar strategy for $\mathcal{G}$. We first build a strategy $\sigma_{\boldsymbol{S}}$ for $\langle\|_{\boldsymbol{S}}\rangle$ using the construction from Sec. 5.3.4. As the commitment sets of the control game range over original transitions rather than copies, we observe that the resulting $\sigma_{\boldsymbol{S}}$ is $\pi$-insensitive, i.e., equally labelled transitions are not distinguished. When taking the coarser label we obtain a branching process $\sigma$ for $\mathcal{G}$. Since $\sigma_{\boldsymbol{S}}$ is $\pi$-insensitive, $\sigma$ fulfils justified refusal, i.e., is a strategy for $\mathcal{G}$. Bisimilar behavior follows since $\sigma \leftrightsquigarrow \sigma_{\boldsymbol{S}}$.

[16] For a place $q$ in the SND we do not allow all commitment sets $A \subseteq post(q)$ but $A \subseteq \pi(post(q))$

[17] For the bisimulation we identify every transition in the control game with its $\pi$-label.

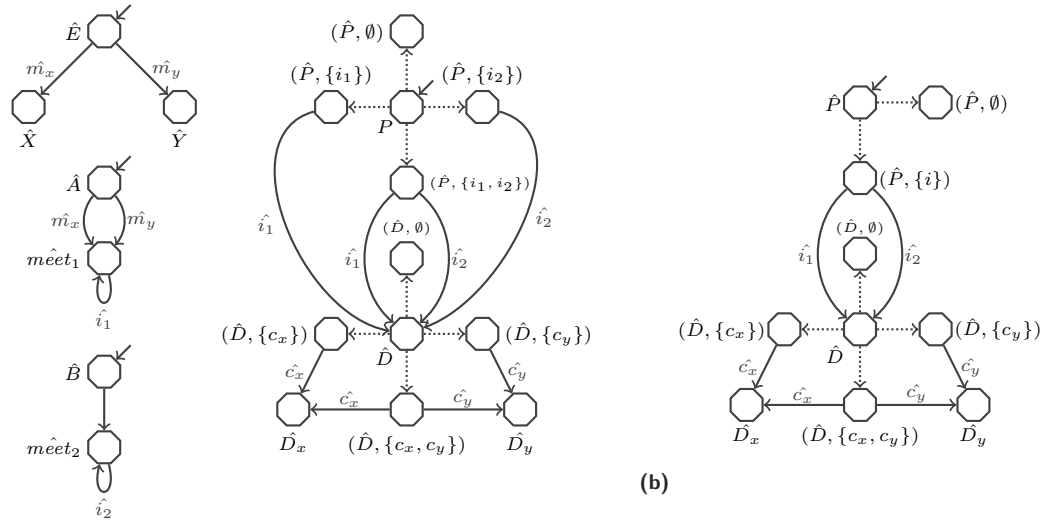



**Figure 5.19** Translation of the Petri game from Fig. 5.17 using the singular net distribution from Fig. 5.18 (a). The local automaton of the first three singular nets is depicted in (a). If we do our unmodified translation the fourth SN is translated to the automaton in (b). The modified translation designed for SNDs yields the automaton in (c).

## The General Result

We argued that we are able to generalize the translation from Sec. 5.1 to work with SNDs instead of slice distributions. In Proposition 4 we showed that all concurrency-preserving nets (and hence games) have a SND. We can hence derive the following generalisation of our initial result:

**Theorem 3** For every *concurrency-preserving* Petri game $\mathcal{G}$ there exist control games $\mathcal{C}_\mathcal{G}$ and $\widehat{\mathcal{C}_\mathcal{G}}$ with an equal number of player such that

- $\mathcal{G}$ and $\mathcal{C}$ are strategy-equivalent.
- $\mathcal{G}$ and $\widehat{\mathcal{C}_\mathcal{G}}$ are strategy-equivalent if we require deterministic Petri game strategies.

# Chapter 6

# Translating Control Games to Petri Games

We give our translation from control games to Petri games and show that the translation yields strategy-equivalent games. Since all existing Petri game results [12, 11] are stated for safety objectives we explicitly work with safety games. Adapting our translation for, e.g., reachability is, however, easy. Apart from our translation we provide an exponential lower bound.

## 6.1 Construction

In this section we describe our translation.

**Actions as Transitions** Our translation has to cope, not only with the different environment formalism, i.e., controllable and uncontrollable actions vs system and environment places, but also with a fundamental difference of both game arenas. By comparing Petri nets and asynchronous automata it becomes clear that they use fundamentally different synchronization primitives. An action $a$ in an asynchronous automaton can fire from different configurations of the processes in $dom(a)$, i.e., all configurations in $domain(\delta_a)$. A transition in a Petri net is only enabled from one fixed configuration; its precondition.

Instead of tackling the harder problem of translating the game variants of both models, it is helpful to first consider the somewhat simpler task of describing an asynchronous automaton $\mathcal{A}$ as an "equivalent" Petri net. A straightforward idea is so represent each local state in $\mathcal{A}$ as a place in a net. For an example we consider the automaton depicted in Fig. 6.1 (a). We define a Petri net with places $A$ to $D$ in (b). Action that move processes between states should then be added as transitions moving tokens between places. To model an action $a$ we have to duplicate it into multiple transitions, one for each configuration from which $a$ can fire ($|domain(\delta_a)|$ many copies). Action $b$ in the example in (a) can only occur from one configuration and is hence added as single transition leaving place $A$ and moving a token to place $B$. $a$, on the other hand, can occur from global states $\langle A, C\rangle$ and $\langle B, C\rangle$. We hence duplicate $a$ into two transitions $a_1$ and $a_2$ both corresponding to exactly one of these configurations and moving the tokens as the action would. Until now we always worked with asynchronous automaton that can be described in terms of a parallel composition of local automata. Fig. 6.1 (c) depicts the description of an automaton where we can not find such a composition, as the transition relation achieves behavior that depends on the global state of the system. The idea of adding places for states and transitions for all configurations in the domain of a transition function extends to such automaton. The resulting Petri net is depicted in Fig. 6.1 (d). The difference in synchronization schemes results in an intrinsic exponential blow up[1].

Following this very informal explanation of how to model asynchronous automata as Petri nets we can continue with defining our intended construction of the game variants.

[1] In asynchronous automaton the complexity is hidden in the transition function $\{\delta_a\}_{a\in\Sigma}$.

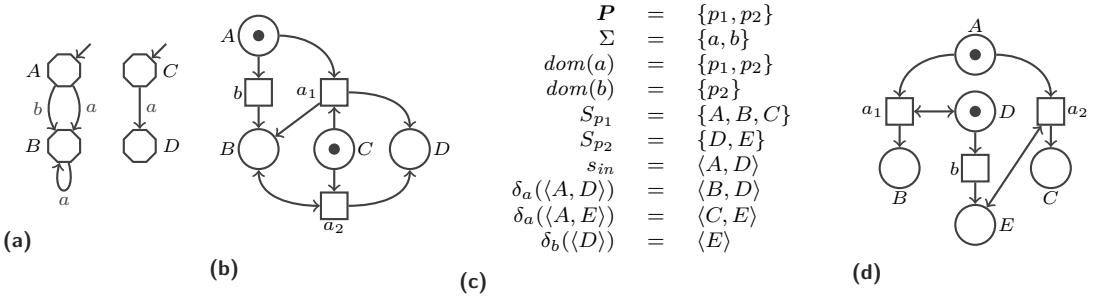



**Figure 6.1** An asynchronous automaton described as a composition of local processes (a) and a possible representation as a Petri net (b). The action $a$ can occur from two configurations ($|domain(\delta_a))| = 2$) resulting in two copies of $a$ ($a_1, a_2$). In (c) another asynchronous automaton is given that cannot be described in terms of the parallel composition of local processes. (d) depicts a possible representation as a Petri net where the action $a$ is duplicated in two transitions ($a_1, a_2$).

Define $\mathcal{G}_\mathcal{C} = (\mathcal{P}_\mathcal{S}, \mathcal{P}_\mathcal{E}, \mathcal{T}, \mathcal{F}, In, \mathcal{B})$ where

- $\mathcal{P}_\mathcal{S} = \bigcup_{p \in \boldsymbol{P}} S_p$
- $\mathcal{P}_\mathcal{E} = \{(s, A) \mid s \in \bigcup_{p \in \boldsymbol{P}} S_p \ \wedge \ A \subseteq act(s) \cap \Sigma^{sys}\} \cup \{\bot^p_{DL} \mid p \in \boldsymbol{P}\}$
- $\mathcal{T} = \{(a, B, \{A_s\}_{s \in B}) \mid a \in \Sigma^{env} \wedge B \in domain(\delta_a) \wedge$
  $\forall s \in B : A_s \subseteq act(s) \cap \Sigma^{sys}\} \cup$ **(1)**

  $\{(a, B, \{A_s\}_{s \in B}) \mid a \in \Sigma^{sys} \wedge B \in domain(\delta_a) \wedge$
  $\forall s \in B : A_s \subseteq act(s) \cap \Sigma^{sys} \wedge a \in A_s\} \cup$ **(2)**

  $\{\tau_{(s,A)} \mid s \in \bigcup_{p \in \boldsymbol{P}} S_p \ \wedge \ A \subseteq act(s) \cap \Sigma^{sys}\} \cup$ **(3)**

  $\{t^M_{DL} \mid M \in \mathfrak{D}_{DL}\}$ **(7)**
- $\mathcal{F} = \{((s, A), (a, B, \{A_s\}_{s \in B})) \mid s \in B \ \wedge \ A_s = A\} \cup$ **(4)**

  $\{((a, B, \{A_s\}_{s \in B}), s') \mid s' \in \delta_a(B)\} \cup$ **(5)**

  $\{(s, \tau_{(s,A)})\} \ \cup \ \{(\tau_{(s,A)}, (s, A)\} \cup$ **(6)**

  $\{(q, t^M_{DL}) \mid q \in M\} \ \cup \{(t^M_{DL}, \bot^p_{DL}) \mid p \in \boldsymbol{P}\}$ **(8)**
- $In = s^\mathcal{A}_{in}$
- $\mathcal{B} = \bigcup_{p \in \boldsymbol{P}} \mathcal{B}_p \ \cup \ \bigcup_{p \in \boldsymbol{P}} \bot^p_{DL}$

**Figure 6.2** The construction of the translated Petri game $\mathcal{G}_\mathcal{C}$ for a control game $\mathcal{C} = (\mathcal{A}, \Sigma^{sys}, \Sigma^{env}, \{\mathcal{B}_p\}_{p \in \boldsymbol{P}})$ where $\mathcal{A} = (\{S_p\}_{p \in \boldsymbol{P}}, s^\mathcal{A}_{in}, \{\delta_a\}_{a \in \Sigma})$. We view the initial state $s^\mathcal{A}_{in}$ of $\mathcal{A}$ as a set of states. The gray parts correspond to the detection of artificial deadlocks.

**Description of $\mathcal{G}_\mathcal{C}$** We fix a control game $\mathcal{C} = (\mathcal{A}, \Sigma^{sys}, \Sigma^{env}, \{\mathcal{B}_p\}_{p \in \boldsymbol{P}})$ with safety objective and transform it into a strategy-equivalent Petri game $\mathcal{G}_\mathcal{C}$. Our translated Petri game $\mathcal{G}_\mathcal{C}$ is depicted in Fig. 6.2. The gray parts describe a special treatment for deadlock detection. We advise to first ignore them. In the control game actions are either controllable or uncontrollable. Our first objective is therefore to achieve similar controllability even though a place in a Petri game can restrict either none or all of the outgoing transitions. We pursue our success from Chapter 5 and again use commitment sets to explicitly encode decisions.

We represent each local state $s$ as a system place. The initial marking of $\mathcal{G}_\mathcal{C}$ is the initial state of $\mathcal{C}$. The bad places are exactly the bad states in $\mathcal{C}$. For each state $s$ we additionally add environment places representing chosen commitment sets of the form $(s, A)$. The commitment sets represent any combination of actions a controller could enable from $s$, i.e., $A \subseteq act(s) \cap \Sigma^{sys}$. Recall that $act(s)$ are all actions that can occur invoking local state $s$. The idea is that, similar to a controller being able to decide which of the outgoing actions to allow from a state, a token on the system place can decide which of the controllable actions to allow by moving to a commitment set[2]. From the system places the strategy can choose a commitment set by using local $\tau$-transitions **(3, 6)**. The $\tau$-transitions are the only transitions leaving the system places. Every action $a$ in $\mathcal{C}$ is represented as possibly many transitions whose precondition comprises only environment places, i.e., places with chosen commitment set. As we observed in, e.g., Fig. 6.1 actions need to be duplicated for every global state in $\mathcal{C}$ from which $a$ can fire. By introducing commitment sets we also have to account for every possible combination of commitment sets and introduce distinct copies for each combination. Transitions hence have the form $t = (a, B, \{A_s\}_{s\in B})$ where $a \in \Sigma$ is the action in the control game, $B \in domain(\delta_a)$ is the configuration from which $a$ can fire, and $\{A_s\}_{s\in B}$ are the involved commitment sets of the local states in $B$ **(1, 2)**. When concerned with bisimulation we identify transition $t$ with the action $a$. Since transitions explicitly encode the commitment set from which they fire, we can read of their precondition: A place $(s, A)$ is in the precondition of transition $t$ if $s \in B$, i.e., the global state encoded in $t$ includes $s$ and $A_s = A$, i.e., the commitment set encoded in $t$ agrees with the current set $A$ **(4)**. When fired, $t$ moves every token to the system place that corresponds to resulting local state when executing $a$ in $\mathcal{C}$, i.e., a place in $\delta_a(B)$ **(5)**. We can bring this in a more understandable form by observing that

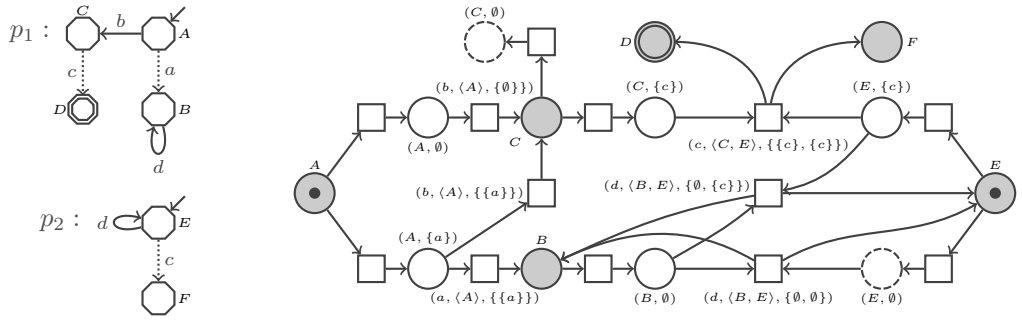
$$pre^{\mathcal{G}_\mathcal{C}}((a, B, \{A_s\}_{s\in B})) = \{(s, A_s) \mid s \in B\} \quad \text{and} \quad post^{\mathcal{G}_\mathcal{C}}((a, B, \{A_s\}_{s\in B})) = \delta_a(B)$$

Since the transitions explicitly encodes the commitment sets of the involved places we can achieve the intended meaning of the commitment sets by restricting when such transitions are added. If $t$ corresponds to a controllable action $a$ it should only be possible if the commitment sets of all places involved in it agree on $a$, that is $a \in A_s$ for every $s \in B$ **(2)**. If a token resides on a place that encodes a commitment set not containing $a$ there is no transitions that corresponds to $a$ added from this place. If action $a$ is uncontrollable it is added independently of the chosen commitment sets **(1)**. This goes well with the intuition that an uncontrollable actions should not be avertible using the commitment sets, but solely depend on the current place of token.

The transitions hence move the tokens though the game exactly as the actions in $\mathcal{C}$ move the processes. In particular, we can note that on every reachable marking in $\mathcal{G}_\mathcal{C}$ there is an one to one correspondence between the place in the marking and processes in the control game: Every place belongs to a process and for every process there is a place that corresponds to a state of that process. We remark that the translated Petri game $\mathcal{G}_\mathcal{C}$ is always concurrency-preserving, safe and, in particular, sliceable. Conceptually every token in $\mathcal{G}_\mathcal{C}$ hence moves along the states of exactly one process. Every token on a place can, using its commitment set, control which of the outgoing controllable actions to allow; just as the process counterpart can. As uncontrollable actions occur independent of the current commitment set but solely dependent the current state, a strategy for $\mathcal{G}_\mathcal{C}$ can not control them and neither can a controller for $\mathcal{C}$.

Figure 6.3 depicts an example translation. The control game in (a) consist of two processes $p_1$ and $p_2$. $p_1$ can actively decide to move to $B$ or is uncontrollably moved to $C$. If in $B$ it can synchronize with $p_2$ on the uncontrollable $d$ action, whereas from $C$ it can communicate with $p_2$ on the controllable $c$. The objective for the system is to avoid place $D$. Since winning controller for safety control games are necessarily deadlock-avoiding there does not exist a

[2]We note that a local controller for a control game is defined as a function $Plays_p(\mathcal{C}) \to 2^{\Sigma_p \cap \Sigma^{sys}}$. It, however, is sufficient if a controller only chooses from the controllable actions that are enabled from its current state. That is consider functions that map a play $u \in Plays_p(\mathcal{C})$ to a subset of $act(state_p(u)) \cap \Sigma^{sys}$.




**(a)** **(b)**

**Figure 6.3** Control game (a) and translated Petri game (b). Commitment sets without outgoing transitions are omitted. The label of $\tau$-transitions is not given as it can be infered from the context. The set of artificial deadlocks $\mathfrak{D}_{DL}$ comprises every final marking that contains at least one of place drawn with a dashed line ($(C, \emptyset)$ or $(E, \emptyset)$). The resulting $t^M_{DL}$-transitions are omitted.

winning controller. Upon execution of the uncontrollable $b$ action the processes must allow $c$, causing them to lose. In (b) our translated Petri game is depicted. The two tokens start in places $A$ and $E$. Both of them can decide which of the outgoing (controllable) actions should be allowed using their commitment sets. If we, for example, consider the token starting in $A$ we observe, that the transitions corresponding to $a$ can only be executed if the player has chosen the commitment set $\{a\}$, i.e., explicitly allowed $a$. Transitions corresponding to the uncontrollable $b$, on the other hand, are possible from both possible commitment sets. As action $c$ only occurs from one fixed configuration in the control game and there is only one combination of commitment sets, it is added as a single transition $(c, \langle C, E\rangle, \{\{c\}, \{c\}\})$ that can only fire if included in both commitment sets. Action $d$, on the other hand, is duplicated to account for all possible commitment set combinations.

**On Deadlocks and Committing** The observant reader might notice that the translation as presented so far is not correct. In the translated $\mathcal{G}_\mathcal{C}$ the system might have a winning strategy even though it has no winning controller in $\mathcal{C}$. The two problems can be characterised as "artificial deadlocks" and "commitment refusal".

The first problem are artificial-deadlocks: In our translation we explicitly work with safety objectives. It is therefore of utmost importance to require the system to behave deadlock avoiding. Without this requirement deciding the existence of a winning strategy is trivial[3]. Since a winning strategy for $\mathcal{G}_\mathcal{C}$ should correspond to a winning (and thereby deadlock-avoiding) controller for $\mathcal{C}$ we obviously need to impose deadlock-avoidance on any winning strategy. Because deadlocks are defined as situations where a strategy globally terminates even though the underlying model can still progress, deadlock-avoidance naturally depends on the underlying automaton or Petri net. The commitment sets introduced in the translation result in additional, "artificial" deadlocked states, i.e., markings in $\mathcal{G}_\mathcal{C}$ that are final but where the corresponding state in $\mathcal{C}$ could still act. By choosing certain commitment sets, e.g., always choose the empty one, the strategy could manoeuvrer into an artificial deadlock, without violating the deadlock-avoidance requirement. If we again consider Fig. 6.3 we can see that our translated game has a winning strategy even though the control game has not. A controller can not refuse to allow $c$ as this would result in a deadlock. A strategy for the translation, on the other hand, can move to place $(C, \emptyset)$ or $(E, \emptyset)$ and thereby effectively prohibit $c$ without being deadlocked. As we explicitly encode the decision in the commitment sets, we can detect and penalize such situations. We give a thorough explanation in Sec. 6.2 below.

[3]There is a winning strategy if and only if the strategy that blocks everything is winning.

The second, more involved, problem are strategies that refuse to commit, i.e., allow no commitment set for certain player. $\mathcal{G}_\mathcal{C}$ is designed such that transitions that represent uncontrollable actions are enabled independent of the chosen commitment but only occur from place that represented a chosen commitment set. If a player refuses to choose a commitment set, i.e., block all $\tau$-transitions from a system place, no uncontrollable actions can occur. A controller in a control game, on the other hand, has no such possibilities. In Sec. 6.4 we show how to enforce commitment by leveraging deadlock-avoidance. In the following, including the correctness proofs, we assume winning strategies for $\mathcal{G}_\mathcal{C}$ to always commit.

## 6.2 On Artificial Deadlocks

We want to hinder a strategy from terminating early by using the newly introduced commitment sets. Recall our use of $\lightning$-actions in Chapter 5: Back then we used uncontrollable actions to prohibit certain global configurations, namely nondeterministic ones, by moving the processes to a locked state and causing the system to lose. We pursue a similar approach by using (losing) transitions to prohibit configurations where the commitment sets are used to terminate early. The gray parts in the Fig. 6.2 describe this formalism. In contrast, to the $\lightning$-actions that were purely optional when considering deterministic strategies, the deadlock-detection mechanisms is inevitable to even allow for winning equivalent translations.

We begin by formally defining artificial deadlocks: Every place in $\mathcal{G}_\mathcal{C}$ (ignoring the $\bot^p_{DL}$ places being introduced in this section) corresponds to a state in $\mathcal{C}$. The correspondence is formalized by $\zeta$ as:

$$\zeta(s) = s$$
$$\zeta(\,(s, A)\,) = s$$

We extend this definition to markings by defining for each marking $M$ a corresponding global state in the asynchronous automaton by: $\zeta(M) = \bigcup_{q \in M} \{\zeta(q)\}$.

Recall that a state in a control game is final, if no further actions are possible once in that state. An *artificial deadlock* now comprises a situation where $M$ is final even though the corresponding state $\zeta(M)$ could still act, i.e., is not final. As $M$ is final a strategy would be allowed to terminate in that marking even though the controller would still be required to keep playing. We want to hinder and penalize a strategy that reaches such a situation. To this extent we equip $\mathcal{G}_\mathcal{C}$ with additional $\bot^p_{DL}$-places that are marked as losing. We define the set of all artificial deadlocks by

$$\mathfrak{D}_{DL} = \{M \in \mathcal{R}(\mathcal{G}_\mathcal{C}) \mid M \subseteq \mathcal{P}_\mathcal{E} \ \wedge \ M \text{ is final } \wedge \ \zeta(M) \text{ is not final}\}$$

Note that every marking in $\mathfrak{D}_{DL}$ contains only environment places, i.e., only places corresponding to chosen commitment sets[4]. For every $M \in \mathfrak{D}_{DL}$ we add a transition $t^M_{DL}$ **(7)** that fires exactly from $M$ and moves every token to a losing place $\bot^p_{DL}$ **(8)**. It holds that

$$pre(t^M_{DL}) = M \qquad \text{and} \qquad post(t^M_{DL}) = \{\bot^p_{DL} \mid p \in \boldsymbol{P}\}$$

Since all $\bot^p_{DL}$ places are losing, a winning strategy has to guarantee that none of the $t^M_{DL}$ transitions are enabled and therefore has to avoid all artificial deadlocks. We remark that this relies on the fact that all places in any marking $M \in \mathfrak{D}_{DL}$ belong to the environment and are therefore unrestrictable by a strategy. With the added deadlock detection a strategy can not manoeuvrer into a situation where it can terminate early, but is only allowed to end the game if the corresponding global state in $\mathcal{C}$ is final as well.

[4] We remark that the definition of $\mathfrak{D}_{DL}$ depends on the reachable markings in the very game we are just defining. Conceptually we first construct the game without the deadlock-mechanism and afterwards add the gray parts corresponding to the deadlock-detection.

In Fig. 6.3 the set of artificial deadlocks, $\mathfrak{D}_{DL}$, comprises all final markings that contain one of the two places highlighted with a dashed line. In such situations at least on player refused $c$ and hence ended the game, even though in the underlying automaton $c$ could still be executed, i.e., the corresponding state is not final. If we add losing transition from all those markings there no longer exists a winning strategy.

## 6.3 Correctness

We show that our translated Petri game $\mathcal{G}_\mathcal{C}$ and the control game $\mathcal{C}$ are strategy-equivalent. Our final result is:

**Theorem 4** $\mathcal{C}$ and $\mathcal{G}_\mathcal{C}$ are strategy-equivalent.

For our proofs we assume that any winning strategy for $\mathcal{G}_\mathcal{C}$ always commits. In Sec. 6.4 we see that this is a valid assumption. Similar to our first translation we begin by giving a high level idea of how a strategy/controller translation looks like and proceed with formal proofs in Sec. 6.3.2, which we advise to skip on first read.

### 6.3.1 Intuition

**Controllability** We first argue that the controllability of the players is identical in $\mathcal{C}$ and $\mathcal{G}_\mathcal{C}$. In $\mathcal{C}$ actions are either controllable or uncontrollable, whereas in $\mathcal{G}_\mathcal{C}$ only system places can restrict transitions. Our commitment set construction overcomes this difference: Every token in $\mathcal{G}_\mathcal{C}$ can decide which commitment set to choose and thereby implicitly restricts the controllable actions leaving this place. A controller for $\mathcal{C}$ can do the same by directly restricting which controllable actions to allow. Transitions corresponding to uncontrollable actions, on the other hand, can occur independent from the commitment set and can thus not be thwarted by a strategy. Conversely a controller for $\mathcal{C}$ can not control any of the outgoing uncontrollable actions. In both $\mathcal{C}$ and $\mathcal{G}_\mathcal{C}$ the system has the same control possibilities. A high level translation now consist in copying the decision of an existing strategy/controller.
Given a controller for $\mathcal{C}$ we can construct a strategy for $\mathcal{G}_\mathcal{C}$: For every local state the controller cam decide which of the controllable actions to allow. The strategy can now copy this decision by allowing the commitment set that comprises all actions allowed by the strategy. Both hence allow identical controllable behavior.
If we are provided with a strategy for $\mathcal{G}_\mathcal{C}$ we can construct a controller for $\mathcal{C}$: As the strategy always commits it always chooses a commitment set, i.e., selects a set of actions enabled from a state. A controller can copy this decision by allowing exactly for the actions that the strategy chose as a commitment set. The controller cannot restrict uncontrollable actions and neither can a strategy.

**On Strategy Equivalence** Strategy Equivalence is defined in terms of a relation $\approx_\mathfrak{B}$ that witnesses bisimilar behavior. As for the first translation we can give an even stronger result by defining $\approx_\mathfrak{B}$ not on a concrete strategy and controller but relate markings in the unfolding with plays in the control games. Since strategies are subprocesses of the unfolding and controller compatible plays subsets of all plays, our relation extends naturally.

Our previous example for our translation (Fig. 6.3) is well suited to highlight important points of our construction but, as it cannot be won by the system, it is exempted from strategy-equivalence and can hence not be used to demonstrate our strategy-controller translation. Instead we consider a translation of a slightly simpler game depicted in Fig. 6.4. The control game $\dot{\mathcal{C}}$ (a) comprises two processes $p_1$ and $p_2$ of which only one contains controllable actions. $p_1$ can uncontrollably take $a$ or $b$ to move to state $B$, while $p_2$ cannot restrict a move to $E$ (using $d$) but can control whether or not to move to state $D$. If both are in $B$ and $D$ they can

synchronize on $c$. This game is well suited to observe the different information of both processes as $p_2$ has to decide whether or not to enable $e$ without yet knowing if $a$ or $b$ occurred. In (b) our translated Petri game $\dot{\mathcal{G}}_{\mathcal{C}}$ is depicted. To aid readability the label of all $\tau$-transitions are omitted and transitions of the form $(a, B, \{A_s\}_{s\in B})$ are labelled by the underlying action $a$. Using commitment sets the token in $C$ can control $e$. All other places can only choose the empty commitment set. Communication on $c$ is represented as a shared transition between both token.

For our bisimulation we pursue a similar approach as we did in Chapter 5. There we related a marking and play if both describe the same situation, i.e., result from the same observable actions/transitions. To translate strategies and controllers we showed that in related situation the local information of each player are identical to its counterpart in the other game and they are therefore able to copy the decision of one another. For our present translation we pursue a similar idea of relating a marking $M$ and play $u$ if they describe the "same situation". We relate $M$ and $u$ if the observable transitions in the causal past of $M$ (when identifying transitions with the corresponding action) agree with $u$. More precisely if the poset structure of the transitions in the causal past of $M$ agree with the poset representation of $u$. This captures the idea, that in equivalent situations a strategy and controller should act equivalently.

As an example, we consider the Petri game $\dot{\mathcal{G}}_{\mathcal{C}}$ from Fig. 6.4. An initial fragment of the unfolding is depicted in Fig. 6.5. As we did in Fig. 6.4 we simplified the names of transitions by identifying transitions of the from $(a, B, \{A_s\}_{s\in B})$ with $a$ and omitted the label of $\tau$-transitions. Consider the blue marking $M_1$. The observable transitions in the past of this marking are $b$ and $e$. As both are completely unrelated and played by distinct player they are unordered when viewed as a poset. The situation of the game naturally corresponds to the play $u_1 = [b, e]_{\mathbb{I}}$. As actions $b$ and $e$ are independent they re unordered in the poset representation. Their poset representation hence agree and we get $M_1 \approx_{\mathfrak{B}} u_1$.

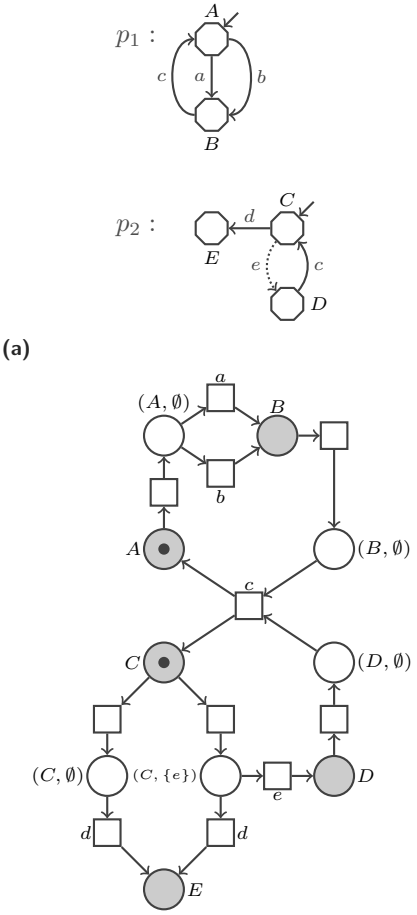



**Figure 6.4** Control game $\dot{\mathcal{C}}$ (a) comprised of two processes $p_1$ and $p_2$. (b) depicts the translated Petri game $\dot{\mathcal{G}}_{\mathcal{C}}$ (b). The labels of $\tau$-transitions in (b) are omitted and observable transitions are labelled with their underlying action.

We can state a few observation regarding $\approx_{\mathfrak{B}}$: Firstly, we can check that every reachable marking in the unfolding of $\dot{\mathcal{G}}_{\mathcal{C}}$ corresponds to a play in $\dot{\mathcal{C}}$. Moreover, on related situations the underlying state is equally labelled as long as we identify places that represent a commitment set with the underlying local state. For example it holds that $state([b, e]_{\mathbb{I}}) = \langle B, D\rangle$ which agrees with $\lambda[M_1]$ if we ignore commitment sets. If we consider the causal past of any reachable marking in the unfolding from Fig. 6.5 there always is related play in $\dot{\mathcal{C}}$.

Having fixed $\approx_{\mathfrak{B}}$ we need to show that we can translate strategies and controller such that they allow bisimilar behavior from related situations. As our commitment set constructions allows equal control possibilities, it remains to argue that the local information of each player are preserved in related situations. To this extend, it is helpful to think of the tokens in the

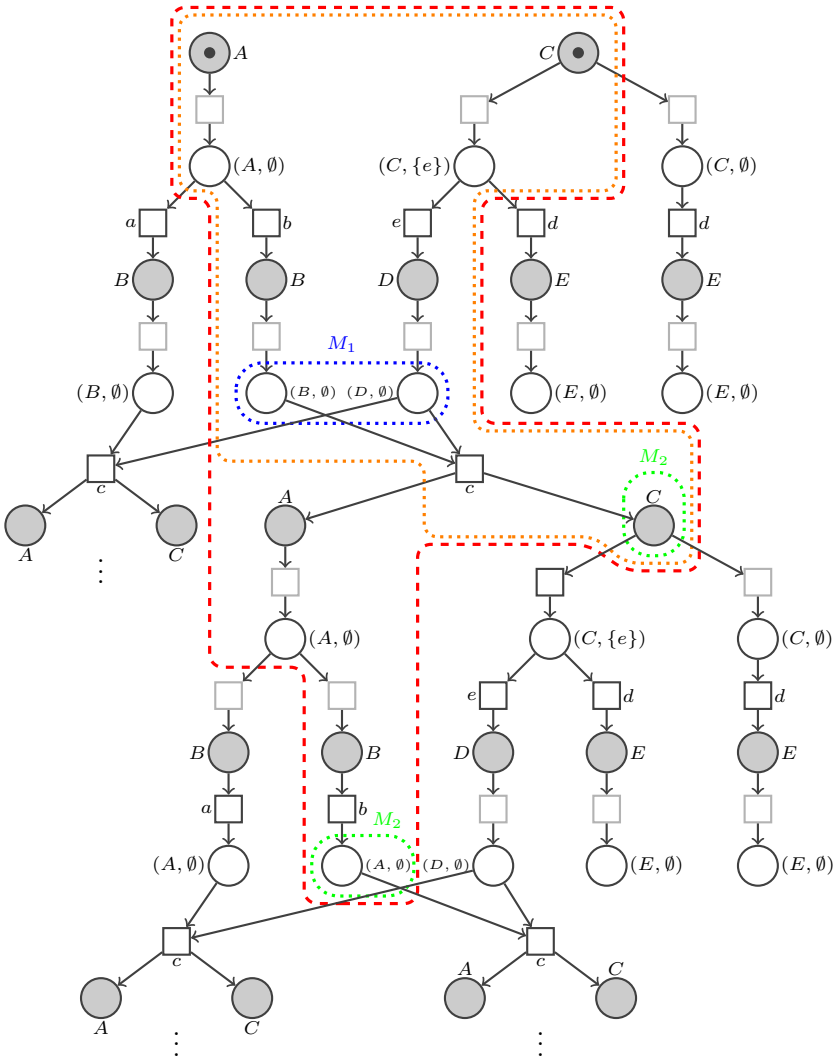



**Figure 6.5** An initial fragment of the unfolding of the Petri game in Fig. 6.4. Two markings $M_1$ and $M_2$ are emphasised in blue and green. The causal past of marking $M_2$ is surrounded in red (dashed) and the causal past of system place $C$ in $M_2$ in orange (dotted). To aid readability the label of transitions are simplified by only giving the name of the action rather than the entire transition. $\tau$-transitions are not labelled and depicted in gray.

Petri game as player moving along one slice and consider the game as a composition of slices. The game can be sliced up such that every token resides only on places of exactly one process. A token hence takes part in precisely the transitions that correspond to the action in which one process takes part in. In the example from Fig. 6.4 there is one token moving along the states of $p_1$ $(A, B)$ and another one along the states of $p_2$ $(C, D, E)$. The communication behavior of the control game is hence replicated truthfully as a Petri game. Since both game types rely on causal information the local information of on token agrees with the information of its process-counterpart.

As an illustrative example, we consider the green marking $M_2$ in Fig. 6.5. The system place $C$ in $M_2$ has to decide on a commitment set, i.e., needs to decide whether or not to allow $e$. The other player is in the $(A, \emptyset)$ place, i.e., has already played a second $b$. A strategy can now restrict the transitions leaving $C$ but has to do so independent of the decision made by the first player, i.e., without knowing that the first player has played $a$ or $b$. It can, however, base its decision

on the knowledge that the first player chose $b$ the first time, an information transmitted upon communication on $c$. The causal past of $M_2$ is surrounded in red. It comprises the observable transitions $b, e, c, b$. The corresponding play in $\dot{\mathcal{C}}$ is

$$u_2 = [\ b\ ,\ e\ ,\ c\ ,\ b\ ]_{\mathbb{I}}$$

It can easily be checked that $M_2 \approx_{\mathfrak{B}} u_2$ as their poset representations align. The local view of $p_2$ on $u_2$ is $[\ b\ ,\ e\ ,\ c\ ]_{\mathbb{I}}$. A controller therefore has to decide whether or not to enable $e$ without knowing if $p_1$ has already player the $a$ or $b$ action. $p_2$ can, however, from its local view deduce that $p_1$ chose $b$ the first time and hence base its decision on that knowledge. In both $M_2$ and $u_2$ the players hence possess the "same" information i.e., while knowing that $b$ occurred the first time, they cannot base their decision on the second occurrence of $a$ or $b$. In both situation the information about the first occurrence of $b$ has been transmitted upon communication as both game types rely on causal information.

**Translating Controllers to Strategies** Given a winning controller $\varrho$ for $\mathcal{C}$ we incrementally construct a (possibly infinite) winning, deterministic strategy $\sigma_\varrho$ for $\mathcal{G}_\mathcal{C}$. We construct our strategy incrementally by adding more and more nodes. As the decision of what to allow is made by system places, every system place $q$ should decide which of the possible transitions are admitted, i.e., decide for a commitment set. The decision should be made based on the causal information of $q$ and should be made in accordance with $\varrho$ to, in the end, obtain bisimilar behavior. In order to do so, $q$ considers the observable transition in its causal past. This past forms a play in $\mathcal{C}$ which can then be given to $\varrho$. The process that correspond to place $q$ then decides which of the controllable actions to allow. $q$ copies this decision by allowing exactly those actions as its commitment set.
For bisimulation we need to show that if $M \approx_{\mathfrak{B}} u$ then $M$ and $u$ allow the same behavior. By our definition of $\approx_{\mathfrak{B}}$ the observable transitions in the past of $M$ agree with $u$. While the causal past of $M$ agrees with $u$ the causal past of place $q$ in $M$ might differ from $u$. One can, however, show that, while the causal past does not agree with the entire $u$, it agrees with the local view of the respective process on $u$ ($view_p(u)$). Every system place in $M$ hence decides for a commitment set that agrees with the decision of one of the processes on $u$. *Together* the player achieving the same behavior as the processes do and allow bisimilar behavior.
Since $\varrho$ is deadlock avoiding the strategy never chooses the commitment sets such that an artificial deadlock is reached. The $t^M_{DL}$-transitions are therefore never enabled and can be neglected when proving bisimulation.

**Translating Strategies to Controllers** Given a winning, deterministic strategy $\sigma$ (that always commits) for $\mathcal{G}_\mathcal{C}$. We build a winning controller $\varrho_\sigma$ for $\mathcal{C}$. If a process $p$ is on any local state in $\mathcal{C}$, it can decide which controllable actions are admitted. The decision of what to enable should be made in accordance with $\sigma$, i.e., made by copying the commitment set that a token in $\sigma$ choose. To copy the decision a process simulates its local view $view_p(u)$ in the branching process of $\sigma$. By simulating the actions in $view_p(u)$ one at a time and simulate as many $\tau$-transitions as possible in between the simulation reaches a marking $M'$. The process then copies the commitment set that has been chosen by the corresponding place in this marking.
As before we need to show that if $M \approx_{\mathfrak{B}} u$, i.e., the observable transitions in the past of $M$ agree with $u$, $\sigma$ and $\varrho_\sigma$ allow the same behavior. Every process $p$ makes its decision based on its local view on $u$. By simulating this local view on $u$ it reaches some marking $M'$ that in general differs from $M$. One can, however, show that the place that corresponds to $p$ is, however, identical in $M'$ and $M$. $p$ hence allows for exactly those transitions that one of the players in $M$ chose as a commitment set . *Together* the process achieve bisimilar behavior.
Because $\sigma$ is winning, it avoids all $t^M_{DL}$ transitions and hence all artificial deadlocks. We can show that the resulting controller is then deadlock-avoiding.

### 6.3.2 Proving Strategy Equivalence

Following this informal description we proceed by giving formal proofs and precise descriptions of our translated strategy and controller. As we did in our first translation we abuse notation and do not distinguish between transitions in a branching process and transitions in the original net. We are hence able to simulate sequences of original transitions in a branching process. Since our obtained Petri game is safe we again obtain an unique simulation.

We can note that for any reachable marking in the unfolding there is an one to one correspondence between places in the marking and processes. For marking $M$ and $p$ we define $M^{\langle p \rangle}$ as the unique place in $M$ with $\zeta(\lambda(M^{\langle p \rangle})) \in S_p$.

**On the relation $\approx_{\mathfrak{B}}$** We use the $\zeta$ function defined in the context of deadlock detection to map places or markings in $\mathcal{G}_{\mathcal{C}}$ to states or global states in $\mathcal{C}$. Recall that $past_{\mathcal{T}}(C) = \{y \in \mathcal{T} \mid \exists x \in C, y \leq x\}$ are the transitions in the causal past of a set of places $C$. The transitions in the causal past can either be $\tau$-transitions or of the form $(a, B, \{A_s\}_{s \in B})$. Since $\tau$-transitions are only added in our translation and $(a, B, \{A_s\}_{s \in B})$-transitions were only needed because of the restrictive synchronization primitives of Petri nets, we define a point wise operation $\square$ that deletes $\tau$-transitions and maps transitions to the corresponding actions.

$$\square\big(\tau_{(s,A)}\big) = \epsilon$$
$$\square\big((a, B, \{A_s\}_{s \in B})\big) = a$$

$\square$ can be seen as projection on the observable actions followed by a mapping to the underlying action. $\epsilon$ denotes the deletion of an element. It should be noted that $\square$ is almost identical to the projection $\langle \cdot \rangle_{\downarrow}^{\mathcal{T}}$ defined in Chapter 5. The only difference is that $\square$ additionally maps $(a, B, \{A_s\}_{s \in B})$-transitions to the action $a$.

We can now express our informal ideas on $\approx_{\mathfrak{B}}$ properly:

$$M \approx_{\mathfrak{B}} u \text{ iff } \square(past_{\mathcal{T}}(M)) = u$$

As we have done in Chapter 5 we can express an equality between the causal past of a marking and a play by comparing the underlying poset representation. $\square(past_{\mathcal{T}}(M)) = u$ hence means that the poset of $\square(past_{\mathcal{T}}(M))$ and $u$ is equal. Note that both are labelled with $\Sigma$. $\approx_{\mathfrak{B}}$ agrees with what we argued informally. Given some marking $M$ in the unfolding, $\square(past_{\mathcal{T}}(M))$ is the partially ordered set that describes the observable transitions in the causal past of $M$. If this agrees with some play $u$ then $M$ and $u$ result from the same situation, i.e., they are reached on the same observable actions/transitions.

In our translation we represent each local state as a place and add a transition $(a, B, \{A_s\}_{s \in B})$ exactly from preconditions that correspond to configurations from which $a$ can occur. A transition in $\mathcal{G}_{\mathcal{C}}$ hence moves the tokens exactly as the corresponding actions would move the processes in $\mathcal{C}$. We can hence see that related marking and play result in equally labelled configurations.

**Lemma 22** If $M \approx_{\mathfrak{B}} u$ then $\zeta(\lambda[M]) = state(u)$.

**Proof** By induction on the length of a totally ordered sequence of $past_{\mathcal{T}}(M)$ using

$$post^{\mathcal{G}_{\mathcal{C}}}((a, B, \{A_s\}_{s \in B})) = \delta_a(B) \in image(\delta_a)$$

$$\zeta(pre^{\mathcal{G}_{\mathcal{C}}}((a, B, \{A_s\}_{s \in B}))) = \zeta(\{(s, A_s) \mid s \in B\}) = B \in domain(\delta_a)$$
$$\zeta(pre^{\mathcal{G}_{\mathcal{C}}}(\tau_{(s,A)}) = \zeta(post^{\mathcal{G}_{\mathcal{C}}}(\tau_{(s,A)})$$

■

**Causal Information Flow** The translated Petri game $\mathcal{G}_\mathcal{C}$ describes the global behavior of all player. It is nerveless helpful to view $\mathcal{G}_\mathcal{C}$ in terms of slices where one slice comprises all places added from the local states of one process. A token hence moves along one slice and thereby along the states of one process. Each transition $(a, B, \{A_s\}_{s\in B})$ added in $\mathcal{G}_\mathcal{C}$ involves exactly the tokens that correspond to the processes that take part in $a$.

Now consider a marking $M$ and play $u$ s.t. $M \approx_\mathfrak{B} u$. By our definition the observable transitions in the past of $M$ agree with $u$. We can observer that in both $M$ and $u$ the local information of a player can be seen as the minimal downwards closed set that contains all transitions/actions where the player is involved in directly. This goes well with the idea that both game types rely on causal information. The partial order orders the events in time. As each communication transmit everything, the entire previous execution is transmitted so all causally preceding events are transmitted. The local view of a player hence comprises all transitions/actions it is involved in directly as well as all causally preceding ones, a downwards closed set. We can state:

**Lemma 23** If $M \approx_\mathfrak{B} u$ and $p \in \boldsymbol{P}$ then

$$view_p(u) = \Box(past_\mathcal{T}(M^{\langle p\rangle})) = view_p(\Box(past_\mathcal{T}(M^{\langle p\rangle})))$$

**Proof** Form $M \approx_\mathfrak{B} u$ it follows that $\Box(past_\mathcal{T}(M)) = u$ where both are considered as partially ordered sets.
We first consider $view_p(u)$: We already argued that in the poset representation $view_p(u)$ is the smallest downwards closed subset of $u$ that contains all actions from $\Sigma_p$.
Now consider $past_\mathcal{T}(M^{\langle p\rangle})$: As $\mathcal{G}^\mathfrak{U}$ is an occurrence net there is an unique transition $t \in pre(M^{\langle p\rangle})$. It holds that $past_\mathcal{T}(M^{\langle p\rangle}) = past_\mathcal{T}(t)$. The token on place $M^{\langle p\rangle}$ moves along the places that correspond to process $p$ and thereby took part in all transitions that correspond to an action in $\Sigma_p$. All transitions in $past_\mathcal{T}(M^{\langle p\rangle})$ that correspond to an action in $\Sigma_p$ are hence causally related to $M^{\langle p\rangle}$. Since transition $t$ also corresponds to an action in $\Sigma_p$ and $past_\mathcal{T}(M^{\langle p\rangle}) = past_\mathcal{T}(t)$ we can characterize $past_\mathcal{T}(M^{\langle p\rangle})$ as the smallest downwards closed subset of $past_\mathcal{T}(M)$ that contains all transitions that correspond to action from $\Sigma_p$. The minimal downwards closed subset is unique and as both $view_p(u)$ and $past_\mathcal{T}(M)$ describe the minimal downwards closed subset contain all actions in $\Sigma_p$ or all transitions corresponding to actions in $\Sigma_p$ it holds that

$$view_p(u) = \Box(past_\mathcal{T}(M^{\langle p\rangle}))$$

We can then conclude $view_p(u) = view_p(\Box(past_\mathcal{T}(M^{\langle p\rangle})))$ as $view_p(\cdot)$ is idempotent. ■

Lemma 23 states that our definition of $\approx_\mathfrak{B}$ does not only capture the global configuration of both games (as stated in Lemma 22) but preserves causal information. In by $\approx_\mathfrak{B}$-related situations $M$ and $u$ the causal past of the place in $M$ that belongs to some process $p$ ($M^{\langle p\rangle}$) agrees with the local view of $p$ on $u$ (after applying $\Box$). Note that in general $past_\mathcal{T}(M^{\langle p\rangle}) \neq past_\mathcal{T}(M)$.

As an example, we again consider the green marking $M_2$ in Fig. 6.5. The causal past of $M_2$ is surrounded in red. It holds that $M_2 \approx_\mathfrak{B} u_2$ for

$$u_2 = [\ b\ ,\ e\ ,\ c\ ,\ b\ ]_\mathbb{I}$$

We can now consider the local information of each player. $M_2^{\langle p_2\rangle}$ is the system place in $M_2$, i.e., the place that corresponds to $p_2$. The causal past $M_2^{\langle p_2\rangle}$ is surrounded in orange. Note that the causal past does not agree with the past of $M_2$. $M_2^{\langle p_2\rangle}$ has to make a decision of what to allow based on this set only, in particular, it cannot base its decision on the fact that the other player already played $b$. Conversely, the local view of $p_2$ on $u_2$ is

$$view_{p_2}(u_2) = [\ b\ ,\ e\ ,\ c\ ]_\mathbb{I}$$

We create places for the initial marking $In^{\sigma_\varrho}$ and extend $\lambda$ s.t. $\lambda[In^{\sigma_\varrho}] = In^{\mathcal{N}}$.
We then iterate:

- For every system place $q$ with no outgoing transitions: $q$ belongs to a process, i.e., $\lambda(q) \in S_p$ for process $p$.
  Compute $u = \square(past_{\mathcal{T}}(q))$ in the already constructed strategy.

  We assume that $u$ is a $\varrho$ compatible play and $state_p(u) = \lambda(q)$ **(A)**. Now define $\mathbb{E} = f_p^{\varrho}(view_p(u)) \subseteq \Sigma_p^{sys}$. Add a transition $t'$ with $\lambda(t') = \tau_{(state_p(u),\mathbb{E})}$ and a new place $q'$ with $\lambda(q') = (state_p(u), \mathbb{E})$ and add the flow s.t. $q \in pre^{\mathcal{N}^{\sigma_\varrho}}(t')$ and $q' \in post^{\sigma_\varrho}(t')$.
  Afterwards, continue with a new unprocessed marking.
- For every set of concurrent environment places $C$ with $\lambda[C] = pre^{\mathcal{G}_\mathcal{C}}(t)$ for some $t$ we add a new copy of $t$ and places for the postcondition (if these nodes did not already exist).

**Figure 6.6** Construction of strategy $\sigma_\varrho$ for $\mathcal{G}_\mathcal{C}$ that is build from controller $\varrho$ for $\mathcal{C}$

We can build the poset representation this play and, e.g., observe that $b$ and $e$ are unordered. If we compare the poset with the causal past of $M_2^{\langle p_2 \rangle}$ (surrounded in orange) we see that they are identical (as stated in Lemma 23). In both situations the system needs to decide whether or not to enable $e$ without yet knowing the most recent decision of the environment ($a$ or $b$).

### 6.3.3 Translating Controllers to Strategies

In this section we provide a formal translation of strategies for $\mathcal{G}_\mathcal{C}$ to controller for $\mathcal{C}$. Given a winning controller $\varrho$ for $\mathcal{C}$ we define a strategy $\sigma_\varrho$ for $\mathcal{G}_\mathcal{C}$. The strategy construction is depicted in Fig. 6.6.

$\sigma_\varrho$ does what we sketched informally. It is build incrementally. We start by creating a branching process that only contains the initial marking and incrementally add more and more places and transitions. For every system place $q$ in the partially constructed strategy we need to add an environment place that represents a commitment set. To decide which to choose we apply $\square$ to the causal past of $q$, i.e., transform the transitions in the past to a trace of actions. There is a process $p$ that corresponds to $q$. The play obtained from the causal past is then given to the local controller of this process which decides for a set of controllable actions $\mathbb{E}$. $q$ the copies this decision by adding the commitment that contains exactly these actions. As soon as our construction adds a new system place we can hence choose a commitment set for that place. Apart from the $\tau$-transitions used to choose commitment sets, no observable transitions involve any system place. To add them we hence consider every set of pairwise concurrent places $C$ and add all transitions leaving from there. The construction hence proceeds by choosing commitment sets for every system place and afterwards add all transitions possible from these commitment sets.

The fact that $\sigma_\varrho$ is a branching process of $\mathcal{G}_\mathcal{C}$ follows directly from the construction: We observe that there is exactly one commitment set chosen. This directly gives us that $\sigma_\varrho$ is deterministic. Note that $\sigma_\varrho$ always commits, i.e., every token on a system place can move to a commitment set.

**Lemma 24** $\sigma_\varrho$ is a deterministic, deadlock-avoiding strategy that always commits.

**Proof** It is easy to see that the constructed net is a branching process of $\mathcal{G}_\mathcal{C}$. The only transitions that might not be added are local $\tau$-transitions. Since they are local and leave a system place we can refuse to add them without violating justified refusal. From any commitment set the construction adds all transitions possible from this set, i.e., does not restrict any transitions that can occur from the commitment sets. The constructed $\sigma_\varrho$ is hence a strategy. For every system place we add exactly one commitment set. $\sigma_\varrho$ is therefore deterministic and always chooses a commitment set. As $\sigma_\varrho$ always commits, it is also deadlock-avoiding. ■

## Strategy-Equivalence

Having constructed $\sigma_\varrho$, we can prove it strategy equivalent to $\varrho$. For our bisimulation we use the previously defined $\approx_\mathfrak{B}$ and restrict it to the reachable markings in $\mathcal{N}^{\sigma_\varrho}$ and plays compatible with $\varrho$. The previous lemmas (Lemma 22 and Lemma 23), established for the unfolding, extend to the restricted version.

The definition of $\sigma_\varrho$ and $\approx_\mathfrak{B}$ are, on its own, completely independent. Lemma 23, however, established an important connection between both: Assume that $M \approx_\mathfrak{B} u$. By definition it holds that $\square(past_\mathcal{T}(M)) = u$. In our construction of the strategy, every system place $q$ in $M$ decides what commitment set to choose by construction a play from its causal past and copy the decision of $\varrho$. According to Lemma 23, the computation of $view_p(\square(past_\mathcal{T}(q)))$ (as done in the strategy definition) agrees with $view_p(u)$. We hence conclude that in $\approx_\mathfrak{B}$-related situations the places in $M$ copy the decision made by $\varrho$ on $u$. We can argue in both direction:

- Suppose in $M$ there is a transition $(a, B, \{A_s\}_{s\in B})$ enabled. $a$ is either controllable or uncontrollable. If uncontrollable $u$ can be extended by $a$ as it is independent of the controller and the state reached on $u$ agrees with $\lambda[M]$ (Lemma 22). If $a$ is controllable all places $q$ involved in $(a, B, \{A_s\}_{s\in B})$ represent commitment sets that contain $a$. The commitment set of a place $q$ was chosen by computing $view_p(\square(past_\mathcal{T}(q)))$ for the respective process $p$ and define the commitment set as the set of action allowed by the controller. By Lemma 23 this is, however, identical to $view_p(u)$. Since $a$ is included in the commitment set of all involved places we can deduce that $a$ must have been allowed by the controller of each involved process. We get that $u\,a \in Plays(\mathcal{C}, \varrho)$.
- Suppose $u\,a \in Plays(\mathcal{C}, \varrho)$. Then $a$ is either uncontrollable or controllable. We first move all tokens from $M$ to a commitment set to be able to execute observable transitions. Call this marking $M'$. In case of $a$ being uncontrollable we can deduce that a transition corresponding to $a$ is possible from $M'$. If $a$ is controllable we observe that all processes involved in $a$ allowed $a$. By Lemma 22 every place in $M'$ involved in $a$ has chosen its commitment set in accordance with the controllers decision on $u$. As all involved processes allowed $a$, $a$ is included in all commitment sets. There hence is a transition corresponding to $a$ enabled in $M'$.

We can now give formal proofs: We begin by showing that it suffices to show that in $\approx_\mathfrak{B}$-related situations the same actions/transitions are possible. That is, if $M \approx_\mathfrak{B} u$ and we extend $M$ and $u$ by the same action/transition we obtain markings and plays that are again related[5].

[5]For the first translation Chapter 5 we have not needed such a results as we defined $\approx_\mathfrak{B}$-directly in terms of firing the actions in the branching process of a strategy. Extending a related marking and play with the same action/transition hence automatically resulted in related situation.

**Lemma 25** If $M \approx_{\mathfrak{B}} u$ and $M \left[\!\left[ \tau^*(a,B,\{A_s\}_{s\in B})\tau^* \right\rangle\!\right] M'$ for some $M'$ and $u' = u\,a$ then $M' \approx_{\mathfrak{B}} u'$

**Proof** From $M \approx_{\mathfrak{B}} u$ we conclude that $\Box(past_{\mathcal{T}}(M)) = u$. All $\tau$-transitions are local, i.e., involve only one place and hence not add any dependencies in the poset of $past_{\mathcal{T}}(M')$. An action $a$ adds dependency, i.e., a causal relation, to all actions from processes in $dom(a)$. As by construction $(a, B, \{A_s\}_{s\in B})$ involves places of tokens that correspond to $dom(a)$ it induces a dependency to the transitions that belong to action where process in $dom(a)$ are involved in. It hence holds that $\Box(past_{\mathcal{T}}(M')) = u'$. ■

We can now formally proof bisimilarity. $\mathcal{G}_{\mathcal{C}}$ comprises additional $t^M_{DL}$-transitions used to detect artificial deadlocks. For our bisimulation proofs we ignore them. We later show that they are indeed never enabled if we construct $\sigma_\varrho$ from a deadlock-avoiding controller $\varrho$.

**Lemma 26** If $M \approx_{\mathfrak{B}} u$ and $M \left[\!\left[ (a,\_,\_) \right\rangle\!\right] M'$ for some $M' \in \mathcal{R}(\mathcal{N}^{\sigma_\varrho})$ then $u' = u\,a \in Plays(\mathcal{C}, \varrho)$ and $M' \approx_{\mathfrak{B}} u'$

**Intuition** Assume $M \left[\!\left[ (a,B,\{A_s\}_{s\in B}) \right\rangle\!\right] M'$. We distinguish whether $a$ is controllable or uncontrollable. If uncontrollable we immediately get that $u' = u\,a \in Plays(\mathcal{C}, \varrho)$. If controllable we know that $a \in A_s$ for all $s \in B$. By Lemma 23 each system places copies the decision of one process on $u$. Since $a$ is contained in all commitment sets, every processes involved in $a$ must have allowed it so $u' = u\,a \in Plays(\mathcal{C}, \varrho)$.

**Proof** If $M \approx_{\mathfrak{B}} u$ then $\zeta(\lambda[M]) = state(u)$ **(1)** (by Lemma 22). We first remark that $a$ is a possible extension of $u$ in the underlying game arena, i.e, $u\,a \in Plays(\mathcal{C})$. This follows from

$$pre^{\mathcal{N}}((a, B, \{A_s\}_{s\in B})) \subseteq \lambda[M]$$

$$\zeta(pre^{\mathcal{N}}((a, B, \{A_s\}_{s\in B}))) = B \in domain(\delta_a)$$

and **(1)**. We now show that $u' = u\,a \in Plays(\mathcal{C}, \varrho)$. $M' \approx_{\mathfrak{B}} u'$ follows from Lemma 25. We distinguish two cases:

- If $a \in \Sigma^{env}$: Since $a$ is enabled an uncontrollable and $u\,a \in Plays(\mathcal{C})$ it follows that $u\,a \in Plays(\mathcal{C}, \varrho)$, i.e, $u\,a$ is a $\varrho$ compatible play.
- If $a \in \Sigma^{sys}$: We know that $a$ is enabled in $state(u)$. By our construction of $\mathcal{G}_{\mathcal{C}}$ because $a \in \Sigma^{sys}$ and since $M \left[\!\left[ (a,B,\{A_s\}_{s\in B}) \right\rangle\!\right] M'$ we can, furthermore, conclude that $a \in A_s$ holds for all $s \in B$ **(2)**.

  We assume for contradiction that $u\,a \notin Plays(\mathcal{C}, \varrho)$. Then there is a process $p \in dom(a)$ s.t. $a \notin f^{\varrho}_p(view_p(u))$. We derive the contradiction by showing that the set of allowed actions agrees with one of the commitment sets in $M$ which, by **(2)**, contains $a$.

  Since $(a, B, \{A_s\}_{s\in B})$ is enabled in $M$ and because of **(1)** we conclude that for the place $q = M^{\langle p \rangle}$ it holds that $\lambda(q) = (state_p(u), A_{state_p(u)})$. Since $p \in dom(a)$ we conclude that $state_p(u) \in B$ and from **(2)** we get that $a \in A_{state_p(u)}$. The place that belongs to process $p$ has choosen a commitment set that includes $a$.

  We can now observe how this commitment set was chosen. Let $q'$ be the predecessor (system) place of $q$, i.e., the place in the strategy from which we added $q$ as a committent set. When considering the construction of $\sigma_\varrho$ we observe that $q'$ computed what commitment set to choose by applying $\Box$ to its causal past. By looking at the definition it holds:

$$\begin{aligned} A_{state_p(u)} &= f^{\varrho}_p(view_p(\Box(past_{\mathcal{T}}(q')))) \\ &= f^{\varrho}_p(view_p(\Box(past_{\mathcal{T}}(q)))) \end{aligned}$$

where the last equality holds since the causal past of $q$ ad $q'$ only differ by one $\tau$-transition.

By Lemma 23 we get that

$$view_p(u) = view_p(\Box(past_{\mathcal{T}}(q)))$$

We can conclude

$$\begin{aligned} A_{state_p(u)} &= f_p^{\varrho}(view_p(\Box(past_{\mathcal{T}}(q)))) \\ &= f_p^{\varrho}(view_p(u)) \end{aligned}$$

The commitment set encoded in $q$ ($A_{state_p(u)}$), i.e, the set place $q'$ decided to add, agrees with the decision of $p$ on $u$. This is a contradiction to $a \in A_{state_p(u)}$ and our assumption $a \notin f_p^{\varrho}(view_p(u))$.

■

**Lemma 27** If $M \approx_{\mathfrak{B}} u$ and $M\ [\tau\rangle\ M'$ then $M' \approx_{\mathfrak{B}} u$.

**Proof** Obvious consequence from definition of $\approx_{\mathfrak{B}}$. ■

Using the two previous lemmas we already show that our assumption in the strategy construction (**(A)**) is justified:

**Corollary** For any place $q \in \mathcal{N}^{\sigma_\varrho}$ that belongs to process $p$, $u = \Box(past_{\mathcal{T}}(q))$ is a $\varrho$ compatible play and $state_p(u) = \lambda(q)$

**Proof** $q$ is part of some reachable marking $M$ and by Lemma 26 and Lemma 27 there is a play $u$ with $M \approx_{\mathfrak{B}} u$. By Lemma 23 it holds that $\Box(past_{\mathcal{T}}(q)) = view_p(u)$ so, as $view_p(u)$ is a $\varrho$ compatible play, $\Box(past_{\mathcal{T}}(q))$ is as well. $state_p(u) = \lambda(q)$ follows from Lemma 22. ■

**Lemma 28** If $M \approx_{\mathfrak{B}} u$ and $u' = u\,a \in Plays(\mathcal{C}, \varrho)$ then there exists $M' \in \mathcal{R}(\mathcal{N}^{\sigma_\varrho})$ with $M\ [\tau^*\,(a,\_,\_)\rangle\ M'$ and $M' \approx_{\mathfrak{B}} u'$.

**Intuition** We first move every token on a system place in $M$ to an environment place, i.e., to a place with chosen commitment set. We distinguish whether $a$ is controllable or uncontrollable. If $a$ is uncontrollable we have a transition of the from $(a, B, \{A_s\}_{s\in B})$ independent of the commitment sets. If $a$ is controllable we can only fire a $(a, B, \{A_s\}_{s\in B})$ transition if $a$ is included in all commitment sets $A_s$. By Lemma 23 the commitment sets are exactly the actions the processes allowed after $u$. Since $u\,a$ is in $Plays(\mathcal{C}, \varrho)$ we get that $a$ must be allowed by all involved processes and is hence included in all committent sets.

**Proof** Since $M \approx_{\mathfrak{B}} u$ we know that $\zeta(\lambda[M]) = state(u)$ (by Lemma 22). We first move every token in $M$ that resides on a system place to an environment place, i.e., move it to a chosen commitment set. Since $\sigma_\varrho$ is by construction deterministic and always commits there is exactly one $\tau$-transition possible from every system place. So $M\ [\tau^*\rangle\ M''$ and in $M''$ every token is on an environment place. It holds that $\zeta(\lambda[M'']) = \zeta(\lambda[M]) = state(u)$. We, furthermore, know that every $q \in M''$, $\lambda(q) = (s, A_s)$, i.e., every place represents commitment set. For every local state $s \in state(u)$ there hence is a commitment set $A_s$, s.t. there is a token on a place labelled $(s, A_s)$ **(1)**.
Let $B = \{state_p(u)\}_{p\in dom(a)}$. Since $a$ can occur from $state(u)$ (as $u\,a \in Plays(\mathcal{C}, \varrho)$) we know that $B \in domain(\delta_a)$. We now claim that there is a transition corresponding to $a$ possible from $M''$. As each transition explicitly encodes the configuration in $domain(\delta_a)$ and commitment sets, we need the global state $B$ and the current commitment sets $A_s$ from **(1)** to "design" the transition. We distinguish whether $a$ is controllable or uncontrollable:

- If $a \in \Sigma^{env}$: We consider the transition $t = (a, B, \{A_s\}_{s \in B})$ where $B$ and the $A_s$'s are the state and sets from above. Such a transition exists as for uncontrollable actions transitions are added independent of the commitment sets. Because of **(1)** we know that $t$ is enabled from $M''$, i.e., $M'' \,[\, t \,\rangle\, M'$ for some $M'$. $M' \approx_{\mathfrak{B}} u'$ follows from Lemma 25.

- If $a \in \Sigma^{sys}$: As $u' = u\,a \in \mathit{Plays}(\mathcal{C}, \varrho)$ we know that for every process $p \in \mathit{dom}(a)$, $a \in f_p^{\varrho}(\mathit{view}_p(u))$ **(2)**. We consider the transition $t = (a, B, \{A_s\}_{s \in B})$ where $B$ and the $A_s$'s are the state and sets from above. Since $a$ is controllable such a transition must not necessarily exists. To show the existence we need to show that $a \in A_s$ for every $s \in B$.

  Assume for contradiction that $a \notin A_{s'}$ for $s' \in B$ (and $p$ the process with $s' \in S_p$). We derive the construction by showing that the commitment set $A_{s'}$ agrees with the set of transitions allowed by $p$ after $u$ which, by **(3)**, contains $a$.

  Let $q = M''^{\langle p \rangle}$. It holds that $\lambda(q) = (s', A_{s'})$, which exists by **(1)**. We can study the construction of $\sigma_{\varrho}$ to see how the commitment set $A_{s'}$ was chosen. Let $q'$ be the predecessor of $q$, i.e., the place from which $q$ was added as a commitment set. By construction it holds that

  $$\begin{aligned} A_{s'} &= f_p(\mathit{view}_p(\Box(\mathit{past}_{\mathcal{T}}(q')))) \\ &= f_p(\mathit{view}_p(\Box(\mathit{past}_{\mathcal{T}}(q)))) \end{aligned}$$

  where the last equality holds since the pasts of $q$ and $q'$ differ only by a $\tau$-transition. From Lemma 23 we get

  $$\mathit{view}_p(u) = \mathit{view}_p(\Box(\mathit{past}_{\mathcal{T}}(q)))$$

  We conclude

  $$\begin{aligned} A_{s'} &= f_p(\mathit{view}_p(\Box(\mathit{past}_{\mathcal{T}}(q)))) \\ &= f_p(\mathit{view}_p(u)) \end{aligned}$$

  The commitment set $A_{s'}$ encoded in $q$ was added from $q'$ agrees with the decision of $p$ made on play $u$. This is a contradiction to $a \in f_p(\mathit{view}_p(u))$ **(2)** and our assumption that $a \notin A_{s'}$.
  Since $t$ exists and because of **(1)** we know that $t$ is enabled from $M''$, i.e., $M'' \,[\, t \,\rangle\, M'$ for some $M'$. $M' \approx_{\mathfrak{B}} u'$ follows from Lemma 25.

■

**Corollary** $\varrho$ and $\sigma_{\varrho}$ are bisimilar.

**Proof** By definition $\mathit{In}^{\sigma_{\varrho}} \approx_{\mathfrak{B}} \epsilon$. Since there are (unobservable) $\tau$-actions in $\mathcal{C}$ the statements follows from Lemma 26, Lemma 27 and Lemma 28. ■

## Deadlock-Avoidance

So far we have established bisimilarity under the assumption that no deadlock detecting transitions $t_{DL}^M$ is enabled. We now show that it is valid to assume. As discussed before a $t_{DL}^M$ is enabled if the strategy manoeuvred into an artificial deadlock. $\sigma_{\varrho}$ simulates $\varrho$ and copies each decision of the controller. Since the commitment sets are chosen according to the controller an

artificial-deadlock is never reached, as it would correspond to a deadlock of $\varrho$.

**Lemma 29** If $\varrho$ is deadlock-avoiding then there are no $t_{DL}^M$ transitions enabled in any reachable marking of $\mathcal{N}^{\sigma_\varrho}$.

**Proof** Suppose there is a reachable marking $M$ and a $t_{DL}^M$ transition enabled from $M$. By construction $t_{DL}^M$ only exists if $M$ is final. Using bisimulation of $\sigma_\varrho$ and $\varrho$ we get a play $u \in \mathit{Plays}(\mathcal{C}, \varrho)$ with $M \approx_{\mathfrak{B}} u$. By bisimilarity $u$ is maximal (since $M$ is final). By construction of $t_{DL}^M$ there is an action enabled from $\zeta(M)$ in the underlying automaton. By Lemma 22 it holds that $\zeta(M) = \mathit{state}(u)$, so $\mathit{state}(u)$ is not final. There hence is a play $u$ that is maximal w.r.t. $\varrho$ but ends in a non-final state $\mathit{state}(u)$. This is a contradiction as deadlock-avoiding requires a controller to only terminate in final states. ■

### Winning Equivalence

Finally we can show that $\sigma_\varrho$ is winning. Having already proved the bisimilarity this is rather easy.

**Lemma 30** If $\varrho$ is winning then $\sigma_\varrho$ is winning.

**Proof** As noticed in Lemma 24 $\sigma_\varrho$ is deadlock-avoiding.
Suppose there is a marking $M$ reachable in $\mathcal{N}^{\sigma_\varrho}$ that contains a bad place $q$. By construction of $\mathcal{G}_\mathcal{C}$ it either holds that $\lambda(q) \in \bigcup_{p \in \boldsymbol{P}} S_p$ or $\lambda(q) = \mathcal{B}_p$, i.e., the bad state must either be inherited from $\mathcal{C}$ or part of the deadlock-detection mechanisms.

- If $\lambda(q) \in \bigcup_{p \in \boldsymbol{P}} S_p$, i.e., $q$ is a place resulting from a bad state in $\mathcal{C}$: By bisimulation there is a $u \in \mathit{Plays}(\mathcal{C}, \varrho)$ with $M \approx_{\mathfrak{B}} u$. It holds that $\zeta(\lambda[M]) = \mathit{state}(u)$ (Lemma 22). By construction of $\mathcal{G}_\mathcal{C}$, $\mathit{state}(u)$ must contain a bad place. A contradiction ot the fact that $\varrho$ is winning.
- If $\lambda(q) = \mathcal{B}_p$, i.e., $q$ is a bad place added to detect artificial deadlocks. Since $\varrho$ is by definition deadlock-avoiding, Lemma 29 gives us that no $t_{DL}^M$ transition is enabled in any reachable marking in $\mathcal{N}^{\sigma_\varrho}$. $\lambda(q) = \mathcal{B}_p$ is hence not possible.

■

We can state the first half of Theorem 4:

**Proposition 5** If $\varrho$ is a winning controller for $\mathcal{C}$ then $\sigma_\varrho$ is a winning, deterministic strategy for $\mathcal{G}$ and bisimilar to $\varrho$.

## 6.3.4 Translating Strategies to Controllers

In this section we provide the formal translation of strategies for $\mathcal{G}_\mathcal{C}$ to controllers for $\mathcal{C}$. Given a winning, deterministic strategy $\sigma$ for $\mathcal{G}_\mathcal{C}$ that always chooses a commitment set. We construct a winning controller $\varrho_\sigma = \{f_p^{\varrho_\sigma}\}_{p \in \boldsymbol{P}}$ for $\mathcal{C}$. We refer to the fact that $\sigma$ always commits by $\star$. The description of $\varrho_\sigma$ is depicted in Fig. 6.7.

$\varrho_\sigma$ does what we argued informally. We depict the decision of a local controller $f_p^{\varrho_\sigma}$ for process $p$. Given some play $u$ it tries to simulate the actions in $u$ in the branching process of $\mathcal{N}^\sigma$. Since $\mathcal{G}_\mathcal{C}$ comprises additional local $\tau$-transitions the simulation needs to add them as well. After having played an action from $u$, $\varrho_\sigma$ hence simulates as many $\tau$-transitions as possible, i.e., moves every token to a place that corresponds to a chosen commitment set. It is easy to see that this is well-defined since all linearisations of a play result in the same marking: If two actions in trace are independent they correspond to distinct parts in the unfolding. The concrete order in which they are fired is hence irrelevant. The simulation of the actions in $u$ can fail, i.e., case **b)** can be reached. While we later show that if $u$ is a controller compatible

For $p \in \boldsymbol{P}$ and $u \in \mathit{Plays}_p(\mathcal{C})$.
Fix any linearisation $u_0, \cdots, u_{n-1}$ of $u$. Define $M_0 = In^\sigma$. Define $\hat{M}_0$ as the marking with $M_0 \,[\tau^*\rangle\, \hat{M}_0$ and there is no $\tau$-transition enabled in $\hat{M}_0$. Since all $\tau$ are local and $\sigma$ is deterministic, $\hat{M}_0$ is unique.
Check if there is an observable transition $t = (u_0, \_, \_)$ enabled from $\hat{M}_0$. There is at most one such transition $t$.

**a)** If such a $t$ exists: Let $M_1$ be the resulting marking, i.e., $\hat{M}_0 \,[t\rangle\, M_1$. Repeat the procedure of playing as many $\tau$-transitions as possible and all actions $u_1, \cdots, u_{n-1}$.

**b)** If no such $t$ exists: Define $f_p^{\varrho_\sigma}(u) = \emptyset$ **(break)**.

We iterate like this until the marking $\mathbb{M}_u = \hat{M}_n$ is reached. We observe that $\lambda((\mathbb{M}_u)^{\langle p\rangle}) = (\_, \mathbb{E})$ (Because of $\star$). Define $f_p^{\varrho_\sigma}(u) = \mathbb{E}$ .

**Figure 6.7** Definition of controller $\varrho_\sigma = \{f_p^{\varrho_\sigma}\}_{p \in \boldsymbol{P}}$ constructed from from strategy $\sigma$ for $\mathcal{G}_\mathcal{C}$. The figure depicts the description of each of the local controller $f_p^{\varrho_\sigma}$.

play the simulation always succeeds, we need to include **b)** to obtain a total function. In case of a successful simulation, a marking $\mathbb{M}_u$ is reached. It is easy to see that on this marking all tokens are on environment places, i.e., have chosen a commitment set. $(\mathbb{M}_u)^{\langle p\rangle}$ is the place in this marking that corresponds to $p$. Process $p$ now copies the decision by allowing exactly the action that are encoded in the commitment set of place $(\mathbb{M}_u)^{\langle p\rangle}$ $(\mathbb{E})$.

As an example, we consider our translation from Fig. 6.4. In Fig. 6.8 a possible winning strategy $\dot{\sigma}$ for $\dot{\mathcal{G}}_\mathcal{C}$ is depicted. $\dot{\sigma}$ always commits so, in particular, whenever only the empty commitment set is available it is allowed. The only place from which any decision can be made is $C$ as a strategy can decide on a commitment set and thereby implicitly decide if transition $e$ should be allowed. In the initial state $\dot{\sigma}$ allows $e$, i.e., admits the token on $C$ to move to place $(C, \{e\})$. The game then progresses: The first player can play either $a$ or $b$ and the second can either take the controllable $e$ or uncontrollable $d$. If they moved to $B$ and $D$ they can synchronize. In case of an communication of $c$, the token that corresponds to $p_2$ can deduce whether the other player played $a$ or $b$. In case of $a$ the strategy terminates, i.e., allows the player in $C$ to move to the empty commitment set. In case of $b$ the strategy allows $e$ for one more time. Note that $\dot{\sigma}$ is winning as $\dot{\mathcal{G}}_\mathcal{C}$ does not contain any bad places and $\dot{\sigma}$ avoids artificial deadlocks. To aid readability we label transitions of the form $(a, B, \{A_s\}_{s \in B})$ with action $a$ and omitted the label of $\tau$-transitions.

In Fig. 6.9 the resulting controller $\dot{\varrho}_\sigma$ is depicted. The controller is depicted by a table that lists all plays allowed by the controller (the local view of $p_2$) and the decision of the second process $p_2$. Just as $\dot{\sigma}$ the controller initially allows $e$. Upon synchronization on $c$ it only allows $e$ if the environment played $b$. If $a$ has been played it refuses $e$. All decision of the controller are in accordance with our description of the strategy translation. We can for exemplary comprehend the decision made on the three plays **(1, 2, 3)**. If the empty play is simulated according to the definition the green marking $M_1$ is reached. The place that corresponds to $p_2$ $(M_1^{\langle p_2\rangle})$ has chosen $\{e\}$ as a commitment set, so $\dot{\varrho}_\sigma$ allows exactly $e$. Play **(**2) results in the blue marking $M_2$ in which the corresponding place has chosen the empty commitment set. Lastly play **(3)** results in the pink marking $M_3$. As the place in $M_3^{\langle p_2\rangle}$ is on a commitment set containing $e$, the local controller of $p_2$ in $\dot{\varrho}_\sigma$ allows $e$. The interested reader is advised to comprehend all decision listed in Fig. 6.9.

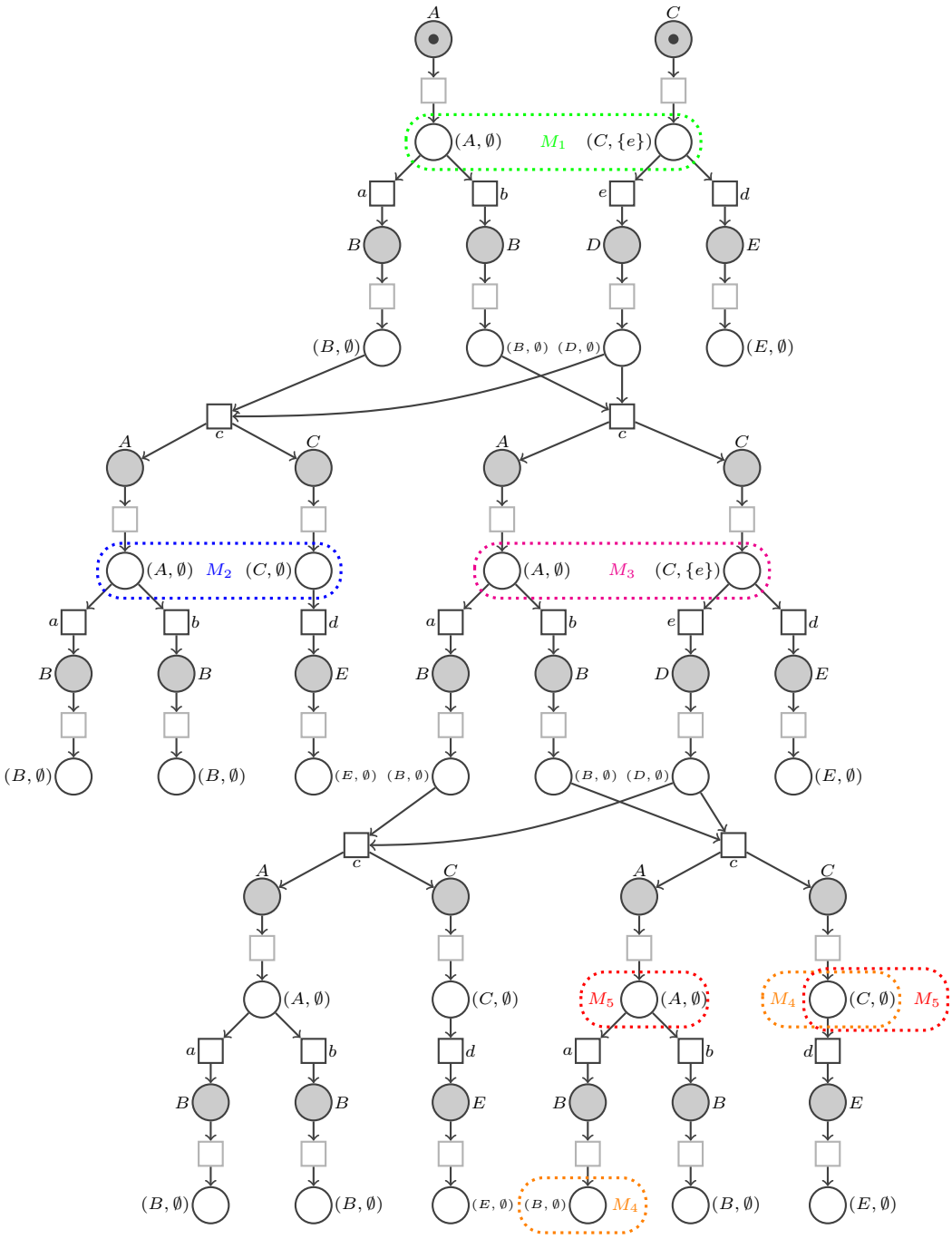



**Figure 6.8** Example winning strategy $\dot{\sigma}$ for the translated Petri game $\dot{\mathcal{G}}_{\mathcal{C}}$ from Fig. 6.4. Reachable marking $M_1, M_2, M_3, M_3$ and $M_5$ are surrounded in green, blue, magenta, orange and red. Transitions are labelled with the underlying actions. $\tau$-transitions are depicted in gray and the label of them is omitted.

| | $u \in Plays(\dot{\mathcal{C}}_{\mathcal{G}}, \dot{\varrho}) \cap Plays_{p_2}(\dot{\mathcal{C}}_{\mathcal{G}})$ | $f_{p_2}(u)$ |
|---|---|---|
| **(1)** | $\epsilon$ | $\{e\}$ |
| | $d$ | $\emptyset$ |
| | $e$ | $\emptyset$ |
| **(2)** | $e, a, c$ | $\emptyset$ |
| **(3)** | $e, b, c$ | $\{e\}$ |
| | $e, a, c, d$ | $\emptyset$ |
| | $e, b, c, d$ | $\emptyset$ |
| | $e, b, c, e$ | $\emptyset$ |
| | $e, b, c, e, a, c$ | $\emptyset$ |
| | $e, b, c, e, b, c$ | $\emptyset$ |
| | $e, b, c, e, b, c, d$ | $\emptyset$ |
| | $e, b, c, e, b, c, d$ | $\emptyset$ |

**Figure 6.9** Controller $\dot{\varrho}_\sigma$ obtained by our translation if applied to the strategy in Fig. 6.8. The table only depicts the decision of $p_2$ as only it involves any controllable actions.

## Strategy-Equivalence

Having defined $\varrho_\sigma$ we can prove it bisimilar to $\sigma$. We use the same relation $\approx_{\mathfrak{B}}$ and restrict it to the reachable markings in $\mathcal{N}^\sigma$ and the plays compatible with $\varrho_\sigma$. As before the existing results (Lemma 22 and Lemma 23) extend to the restricted version.

**Lemma 31** If $M \approx_{\mathfrak{B}} u$ and $p \in \boldsymbol{P}$ and $M^{\langle p \rangle}$ is an environment place, i.e., corresponds to a chosen committent set then: Simulating $view_p(u)$ as in the definition of $\varrho_\sigma$ succeeds and yields a marking $\mathbb{M}_{view_p(u)}$ where $M^{\langle p \rangle} = (\mathbb{M}_{view_p(u)})^{\langle p \rangle}$

**Proof** From Lemma 23 we get that $\Box(past_{\mathcal{T}}(M^{\langle p \rangle})) = view_p(u)$. The actions in $view_p(u)$ hence agree with the past of $q$ and since $\sigma$ is deterministic the simulation is deterministic as well. The simulation hence succeeds and yield a marking $\mathbb{M}_{view_p(u)}$. $M^{\langle p \rangle} = (\mathbb{M}_{view_p(u)})^{\langle p \rangle}$ follows since the simulation fires exactly the transitions in the past from $M^{\langle p \rangle}$. ■

Lemma 31 is a trivial consequence from Lemma 23 that allows us to prove the bisimilarity. It tells that simulating the local view of a process results in a marking $\mathbb{M}_{view_p(u)}$ and the decisive point in this marking is shared with $M$. This allows us to conclude a connection between our definition of $\approx_{\mathfrak{B}}$ and our construction of $\varrho_\sigma$. In $\varrho_\sigma$ every process simulates its local view and, according to Lemma 31, thereby copies the decision in a related marking.

For our example we consider marking $M_4$ in Fig. 6.8. It is easy to check that $M_4 \approx_{\mathfrak{B}} u_3$ for

$$u_4 = [\, e\,,\ \ b\,,\ \ c\,,\ \ e\,,\ b\,,\ c\,,\ a\,]_{\mathbb{I}}$$

In the simulation for our controller we simulate the local view of $p_2$ on $u_4$ which is $[e, b, c, e, b, c]_{\mathbb{I}}$, i.e., identical up to a removed $a$. Simulating this play does not result in marking $M_4$ but instead in marking $M_5$ coloured in red. The interesting place, i.e., place $(C, \emptyset)$, is, however, shared in both markings (as stated in Lemma 31). Controller $\dot{\varrho}_\sigma$ would on $u_4$ hence forbid $e$ and thereby copy the decision in the related marking $M_4$ even though he computed the different $M_5$.

We can use Lemma 31 to show bisimilarity: We can reason in both direction:

- If $M\ [_{(a,B,\{A_s\}_{s\in B})}\rangle\ M'$ we can do a case analysis if $a$ is uncontrollable or not. If it is uncontrollable we immediately get that $u\,a \in Plays(\mathcal{C}, \varrho_\sigma)$ since the underlying state reached on $u$ agrees with $\lambda[M]$ (Lemma 22). If $a$ is controllable we can deduce that $a \in A_s$ for all $s \in B$, i.e., every involved token has chosen a commitment set where $a$ is included.

By Lemma 31, $\varrho_\sigma$ now simulates the local view of a process and thereby reaches a place in $M$. Since all tokens involved in $a$ have chosen a commitment set where $a$ is included, all processes involved in $a$ will allow $a$. So $u\,a \in \mathit{Plays}(\mathcal{C}, \varrho_\sigma)$.

- If, on the other hand, $u\,a \in \mathit{Plays}(\mathcal{C}, \varrho_\sigma)$ we first move every token in $M$ to a commitment set which is always possible by $\star$. The new marking is $M'$. If $a$ is uncontrollable a transitions corresponding to $a$ is possible from this commitment set combination. If $a$ is controllable every involved process has allowed $a$. By construction the processes decided what to allow by simulating its local view, which, according to Lemma 31, results in a place of $M'$. Since every involved process allows $a$ every involved place must have chosen a commitment set where $a$ is included. We drive that a transitions corresponding to $a$ is possible from $M'$.

Since we assume that $\sigma$ is winning, there can never be any $t^M_{DL}$-transitions enabled. We can hence neglect them for our bisimulation proofs.

**Lemma 32** If $M \approx_{\mathfrak{B}} u$ and $M\ [\langle a,\_,\_\rangle\rangle\ M'$ for some $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ then $u' = u\,a \in \mathit{Plays}(\mathcal{C}, \varrho_\sigma)$ and $M' \approx_{\mathfrak{B}} u'$

**Intuition** Assume $M\ [\langle a,B,\{A_s\}_{s\in B}\rangle\ M'$. We distinguish whether $a$ is controllable or uncontrollable. If uncontrollable we immediately get that $u' = u\,a \in \mathit{Plays}(\mathcal{C}, \varrho_\sigma)$. If $a$ is uncontrollable we know that $a \in A_s$, i.e., $a$ is included in every commitment set. By Lemma 31 every process in $\mathit{dom}(a)$ simulates its local view and arrives at a place in $M$. Every process hence allows exactly the actions that are included in one of the commitment sets $A_s$. So every involved process allows $a$ and $u' = u\,a \in \mathit{Plays}(\mathcal{C}, \varrho_\sigma)$.

**Proof** Since $M \approx_{\mathfrak{B}} u$ we know that $\zeta(\lambda[M]) = \mathit{state}(u)$ **(1)** (by Lemma 22).
Because $(a, B, \{A_s\}_{s\in B})$ is enabled in $M$, **(1)** holds and our construction of transitions we know that $a$ is enabled from $\mathit{state}(u)$, i.e., $u\,a \in \mathit{Plays}(\mathcal{C})$. We distinguish two cases:

- If $a \in \Sigma^{env}$: Then $u' = u\,a \in \mathit{Plays}(\mathcal{C}, \varrho_\sigma)$ follows from the definition of control games. $M' \approx_{\mathfrak{B}} u'$ follows from Lemma 25.
- If $a \in \Sigma^{sys}$: Assume for contradiction that $u' = u\,a \notin \mathit{Plays}(\mathcal{C}, \varrho_\sigma)$. Then there is a $p \in \mathit{dom}(a)$ with $a \notin f_p^{\varrho_\sigma}(\mathit{view}_p(u))$. We derive the contradiction by showing that the set of allowed transitions by $p$ is exactly one of the commitment sets in $M$ which by assumption includes $a$.

  $M^{\langle p\rangle} \in M$ is the place that corresponds to process $p$. As this place is involved in $(a, B, \{A_s\}_{s\in B})$ and $(a, B, \{A_s\}_{s\in B})$ is enabled in $M$ we conclude that $\lambda(M^{\langle p\rangle})$ is an environment place, i.e., a chosen commitment set. Because of **(1)** we get that $\lambda(M^{\langle p\rangle}) = (\mathit{state}_p(u), A_{\mathit{state}_p(u)})$ By construction of $\mathcal{G}_\mathcal{C}$ and since $(a, B, \{A_s\}_{s\in B})$ is enabled we get that $a \in A_{\mathit{state}_p(u)}$.

  Let $\mathbb{E} = f_p^{\varrho_\sigma}(\mathit{view}_p(u))$ be this decision of process $p$ on $u$. We can now study how $\varrho_\sigma$ came to this decision. It does so by simulating $\mathit{view}_p(u)$ in the branching process of $\sigma$ and reaches a marking $\mathbb{M}_{\mathit{view}_p(u)}$ (by Lemma 31 the simulation is successfully).
  By construction $p$ then chooses $\mathbb{E}$ as the set with $\lambda((\mathbb{M}_{\mathit{view}_p(u)})^{\langle p\rangle}) = (\mathit{state}_p(u), \mathbb{E})$. That is $\varrho_\sigma$ copies the decision of the corresponding place in $\mathbb{M}_{\mathit{view}_p(u)}$.

  From Lemma 31 we now get that

$$M^{\langle p\rangle} = (\mathbb{M}_{\mathit{view}_p(u)})^{\langle p\rangle}$$

This allows us to conclude that

$$\begin{aligned}(state_p(u), \mathbb{E}) &= \lambda((\mathbb{M}_{view_p(u)})^{\langle p \rangle}) \\ &= \lambda(M^{\langle p \rangle}) \\ &= (state_p(u), A_{state_p(u)})\end{aligned}$$

We get that $f_p^{\varrho_\sigma}(view_p(u)) = \mathbb{E} = A_{state_p(u)}$. The decision of what to enable $(f_p^{\varrho_\sigma}(view_p(u)))$ hence agrees with the commitment set of place $M^{\langle p \rangle}$ which is $M^{\langle p \rangle}$. This is a contradiction to $a \in A_{state_p(u)}$ and our assumption $a \notin f_p^{\varrho_\sigma}(view_p(u))$.

So $u' = u\,a \notin Plays(\mathcal{C}, \varrho_\sigma)$. $M' \approx_{\mathfrak{B}} u'$ follows from Lemma 25.

■

**Lemma 33** If $M \approx_{\mathfrak{B}} u$ and $M \,[\, \tau \,\rangle\, M'$ for some $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ then $M' \approx_{\mathfrak{B}} u$.

**Proof** Obvious consequence from definition of $\approx_{\mathfrak{B}}$. ■

**Lemma 34** If $M \approx_{\mathfrak{B}} u$ and $u' = u\,a \in Plays(\mathcal{C}, \varrho_\sigma)$ then there exists $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ with $M \,[\, \tau^*(a,\_,\_) \,\rangle\, M'$ and $M' \approx_{\mathfrak{B}} u'$.

**Intuition** We first move every token to an environment place, i.e., to place that encodes a commitment set. This is possible because of $\star$. We distinguish if $a$ is controllable or uncontrollable. If uncontrollable there always is a $(a, B, \{A_s\}_{s \in B})$-transition, independent of the chosen commitment sets. If controllable we need to show that $a$ is in the commitment set of all paces involved in a $(a, B, \{A_s\}_{s \in B})$-transition. By Lemma 31 every process allows exactly the actions that are included in the commitment sets. Since $u' = u\,a \in Plays(\mathcal{C}, \varrho_\sigma)$ every involved process has allowed $a$ so every involved place has included $a$ in its commitment set.

**Proof** Since $M \approx_{\mathfrak{B}} u$ we know that $\zeta(\lambda[M]) = state(u)$ (by Lemma 22). We first move every token that resides on a system place to an environment one, i.e., to a commitment set place. Since $\sigma$ satisfies $\star$, i.e., always commits this is always possible. So $M \,[\, \kappa \,\rangle\, M''$ for some $M''$ and there are no enabled $\tau$-transitions in $M''$. It holds that $\zeta(\lambda[M'']) = \zeta(\lambda[M]) = state(u)$. For every $q \in M''$ it holds that $\lambda(q) = (s, A_s)$, i.e., all tokens have chosen a commitment set. For every local state $s \in state(u)$ there is a set $A_s$ such that there is a token on a place labelled $(s, A_s)$ **(1)**.
Since $u\,a \in Plays(\mathcal{C}, \varrho_\sigma)$ we know that $a$ can occur from $state(u)$. Let $B = \{state_p(u)\}_{p \in dom(a)}$. Since $a$ can occur from $state(u)$ (as $u\,a \in Plays(\mathcal{C}, \varrho)$) we know that $B \in domain(\delta_a)$. We now claim that there is a transition corresponding to $a$ possible from $M''$. As each transition explicitly encodes the configuration in $domain(\delta_a)$ and commitment sets, we need the global state $B$ and the current commitment sets $A_s$ from **(1)** to "design" the transition. We distinguish whether $a$ is controllable or uncontrollable.

- If $a \in \Sigma^{env}$: Consider the transition $t = (a, B, \{A_s\}_{s \in B})$ where $B$ is the global state from above and $A_s$ are the sets such that there is a token on $(s, A_s)$ **(1)**. By construction of $\mathcal{G}_\mathcal{C}$ such a $t$ exists. We conclude that $t$ is enabled in $M''$ and, as $t$ involves only environment places, it is allowed by $\sigma$ so there is a $M'$ with $M'' \,[\, t \,\rangle\, M'$. $M' \approx_{\mathfrak{B}} u'$ is follows from Lemma 25.
- If $a \in \Sigma^{sys}$: We know that for every $p \in dom(a)$, $a \in f_p^{\varrho_\sigma}(view_p(u))$ **(2)**. We again consider the transition $t = (a, B, \{A_s\}_{s \in B})$ where $B$ is the global state from above and $A_s$ are the sets such that there is a token on $(s, A_s)$ **(1)**. By construction of $\mathcal{G}_\mathcal{C}$ such a transition only exists if $a \in A_s$ for all $s \in B$.

Assume for contradiction that $a \notin A_{s'}$ for some $s' \in B$. Let $p$ be the process with $s' \in S_p$ (it holds that $state_p(u) = s'$). We derive the contradiction by showing that the set $A_{s'}$ is the set of actions allowed by $p$ on $u$ and $a$ must therefore, by **(2)**, be included.

For $M''^{\langle p \rangle} \in M$ it holds that $\lambda(M''^{\langle p \rangle}) = (s', A_{s'})$. Let $\mathbb{E} = f_p^{\varrho_\sigma}(view_p(u))$ be the decision made by $p$. We can study how $\varrho_\sigma$ came to this decision. It does so by simulating $view_p(u)$ in the branching process of $\sigma$ and reaches a marking $\mathbb{M}_{view_p(u)}$. $\mathbb{E}$ is then, by construction, the set with $\lambda((\mathbb{M}_{view_p(u)})^{\langle p \rangle}) = (state_p(u), \mathbb{E})$.

From Lemma 31 we get that

$$(\mathbb{M}_{view_p(u)})^{\langle p \rangle} = M''^{\langle p \rangle}$$

So we can derive that

$$\begin{aligned}(s', A_{s'}) &= \lambda(M''^{\langle p \rangle}) \\ &= \lambda((\mathbb{M}_{view_p(u)})^{\langle p \rangle}) \\ &= (state_p(u), \mathbb{E})\end{aligned}$$

So $f_p^{\varrho_\sigma}(view_p(u)) = \mathbb{E} = A_{s'}$, i.e., the commitment set $A_{s'}$ agrees with the decision of $p$ made on $u$. This is a contradiction to $a \in f_p^{\varrho_\sigma}(view_p(u))$ **(2)** and our assumption that $a \notin A_{s'}$.

We conclude that $t$ exists and is enabled in $M''$ so there is a $M'$ with $M'' \,[t\rangle\, M'$. $M' \approx_{\mathfrak{B}} u'$ follows from Lemma 25.

■

**Corollary** $\sigma$ and $\varrho_\sigma$ are bisimilar

**Proof** By definition it holds that $In^\sigma \approx_{\mathfrak{B}} \epsilon$. Since there are now unobservable $\tau$-actions in $\mathcal{C}$ the statements follows from Lemma 32, Lemma 33 and Lemma 34. ■

## Deadlock-Avoidance

We show that $\varrho_\sigma$ is deadlock avoiding. By construction $\varrho_\sigma$ allows exactly the actions that $\sigma$ has included in the commitment sets. To avoid all $t_{DL}^M$ transitions $\sigma$ has to choose commitment sets such that there is a transition possible if there is an action possible from the corresponding state in $\mathcal{C}$ (cf. Artificial deadlocks). By copying commitment sets there is no deadlock reachable.

**Lemma 35** If $\sigma$ is deadlock avoiding and avoids $t_{DL}^M$-transitions, $\varrho_\sigma$ is deadlock-avoiding.

**Proof** Assume for contradiction $\varrho_\sigma$ is not deadlock-avoiding. Then there exists a play $u \in Plays(\mathcal{C}, \varrho_\sigma)$ that is maximal w.r.t. the controller that could be extended in the underlying automaton. By bisimulation there exist a reachable marking $M$ in $\mathcal{N}^\sigma$ with $M \approx_{\mathfrak{B}} u$. Let $M'$ be the marking that results from $M$ by playing as many $\tau$ transitions as possible, i.e., where every token has chosen a commitment set. Because $\sigma$ satisfies $\star$ in $M'$ every token is on an environment place (i.e., has chosen a commitment set). It holds that $M' \approx_{\mathfrak{B}} u$.
Since $u$ is maximal, $M'$ is final (by bisimulation). Because $\sigma$ is deadlock avoiding, $\lambda[M']$ is final as well. Since $u$ can be extended in the underlying automaton there exist an action $a$ that is enabled in $state(u)$, i.e., $state(u)$ is not final. By Lemma 22 it holds that $\zeta(\lambda[M']) = state(u)$. By construction of the deadlock detection mechanism there hence is a transition $t_{DL}^{\lambda[M']}$ that is enabled in $\lambda[M']$ and can not be averted by strategy. A contradiction to the assumption. ■

### Winning Equivalence

Once we established the bisimilar behavior we can show that $\varrho_\sigma$ is indeed winning.

**Lemma 36** If $\sigma$ is winning, $\varrho_\sigma$ is winning

**Proof** Since $\sigma$ is winning it is by definition deadlock-avoiding. It, furthermore, avoids all $t^M_{DL}$ transitions so, by Lemma 35, $\varrho_\sigma$ is deadlock-avoiding.
Suppose $u \in \mathit{Plays}(\mathcal{C}, \varrho_\sigma)$ is a play that reaches a state that contains a bad state. By bisimulation there is reachable marking $M$ in $\mathcal{N}^\sigma$ with $M \approx_{\mathfrak{B}} u$. Let $M'$ be the marking where the last $\tau$-transition of ever place is reversed, i.e., $M'$ is almost identical to $M$ but every token is on a system place. It holds that $M' \approx_{\mathfrak{B}} u$. (We only need to do this reasoning because bad places in $\mathcal{G}_\mathcal{C}$ are restricted to the system places. ) Using Lemma 22 we conclude that $\zeta(\lambda[M']) = \lambda[M'] = \mathit{state}(u)$ so by construction of $\mathcal{G}_\mathcal{C}$, $M'$ contains a bad place. A contradiction to the fact that $\sigma$ is winning. ■

We can conclude the second half of the Theorem 4.

**Proposition 6** If $\sigma$ is a winning deterministic strategy for $\mathcal{G}_\mathcal{C}$ then $\varrho_\sigma$ is a winning controller for $\mathcal{C}$ and bisimilar to $\sigma$.

## 6.4 Enforcing Commitment

For our previous proofs to work and our results to hold we need strategies that always commit. In this section we outline how the winning objectives for the system can be used to enforce commitment.

For safety objectives it is difficult to force commitment since allowing fewer transitions is in general better. To avoid trivial strategies that refuse everything, strategies for safety games must be deadlock-avoiding. The system must hence allow some move in the global game if there is a move possible in the underlying arena. By definition deadlock-avoidance hence describes a *global* property of the strategy. In $\mathcal{G}_\mathcal{C}$ we require *local* deadlock-avoidance in the sense that the system tokens always have to choose a commitment set. It is important to note that both are not the same. To be deadlock-avoidant in the original sense it suffices if a single player can play continuously. In $\mathcal{G}_\mathcal{C}$ such a player would enable others to refuse commitment without being (globally) deadlocked.

As an example, consider the control game in Fig. 6.10 (a) and the translated Petri game in (b) (ignoring all grayed out parts). Even though the control game has no winning controller, the Petri game has a winning strategy: The token in $A$ refuses to commit and the token in $D$ plays transition $c$ for ever. We can note that if the token in $A$ chooses a commitment set, even if it is the empty one, the uncontrollable $a$ transitions can occur, causing a loss.

We can use the structure of $\mathcal{G}_\mathcal{C}$ to reduce local deadlock-avoidance to global one. The idea is to terminate certain player. For every process we add an additional place as a “safe-haven”, i.e., a place that is neither losing nor has it any outgoing transitions. We allow every token that is on an environment place, i.e., a place representing a chosen commitment set, to move to this new place. Every token that behaves as intended, i.e., always commits, can hence be moved to the haven and is therefore effectively removed from the game. All players that are locally deadlocked, i.e., refused to commit, could previously do so since some player continued playing. As soon as all other player terminate the locally deadlocked players do, however, cause a global deadlock since there no longer is a progressing player that can justify their refusal. If in Fig. 6.10 the second player corresponding to $p_2$ would be removed from the game the first player creates a global deadlock. Every strategy where a token is locally deadlocked hence results in a (globally) deadlocked strategy and is therefore by assumption not winning. We successfully reduced local deadlock-avoidance to global deadlock-avoidance. Our reduction relies on the fact that Petri games are conceptually scheduled by an adversary scheduler. While the transition

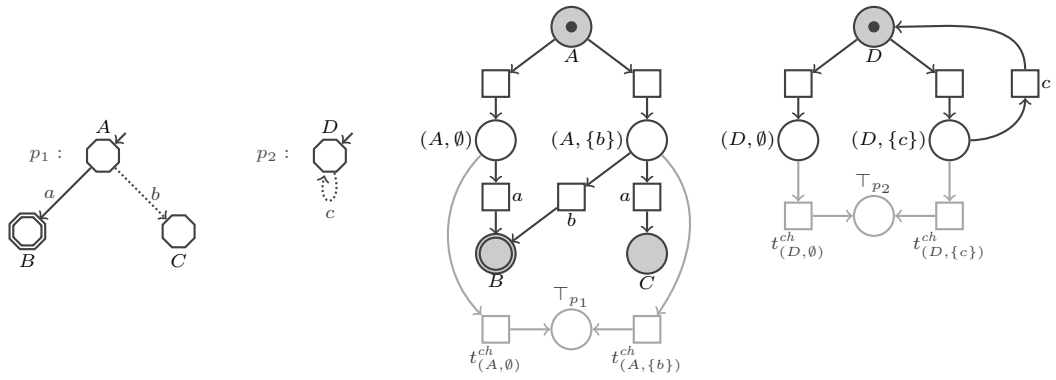



**Figure 6.10** (Safety) Control game $\mathcal{C}$ (a) and our translated Petri game $\mathcal{G}_\mathcal{C}$ (b). Non interesting commitment sets for places $B$ and $C$ are omitted. The greyed parts model the addition of the challenge transitions in $\mathcal{G}^{ch}_\mathcal{C}$. $\mathcal{G}_\mathcal{C}$ (without the grayed out parts) has a winning strategy. $\mathcal{G}^{ch}_\mathcal{C}$ does not admit a winning strategy.

leading to a safe haven is always possible, it might not be executed. Following this high level explanation we proceed by outlining the precise construction.

## Construction

We formally modify $\mathcal{G}_\mathcal{C}$ into a new game $\mathcal{G}^{ch}_\mathcal{C}$. For every process $p$ we add an additional place $\top_p$ to $\mathcal{G}_\mathcal{C}$. This place serves as the "safe-haven". We allow every token to move to this place whenever it has chosen a commitment set. We hence define a new set of transitions

$$\mathcal{T}^{ch} = \{t^{ch}_{(s,A)} \mid s \in \bigcup_{p \in \boldsymbol{P}} S_p \ \wedge \ A \subseteq act(s) \cap \Sigma^{sys}\}$$

and add them two the game. There is exactly one such $t^{ch}_{(s,A)}$-transition for every environment place, i.e., every place with commitment set $(s, A)$. We extend the flow such that these transitions fire from precisely the commitment set encoded in the transition:

$$pre^{\mathcal{G}^{ch}_\mathcal{C}}(t^{ch}_{(s,A)}) = \{(s, A)\}$$

Every $t^{ch}_{(s,A)}$ transition moves the token of the involved process to the safe haven, $\top_p$:

$$post^{\mathcal{G}^{ch}_\mathcal{C}}(t^{ch}_{(s,A)}) = \{\top_p \mid \text{ where } p \text{ is the process with } s \in S_p\}$$

Note that the precondition of all $t^{ch}_{(s,A)}$ comprises only environment places and can hence not be restricted by a strategy. In Fig. 6.10 the added transitions and places are depicted in gray. From every commitment set a token can always move to the $\top$ place. In the modified game the system no longer has a winning strategy, since the token in $D$ can be stopped at any point causing the token in $A$ to create a deadlock.

**Correctness** We can first observe that in any winning strategy $\sigma_{ch}$ for $\mathcal{G}^{ch}_\mathcal{C}$ every system place always commits. This follows directly from the construction: Suppose there is such a situation i.e., a marking $M$ in $\mathcal{N}^{\sigma_{ch}}$ where a system place $q \in M$ refuses to commit, i.e, $post^{\mathcal{N}^{\sigma_{ch}}}(q) = \emptyset$. We now consider one possible sequence starting in $M$: Every system place that can commit chooses a commitment set and afterwards terminates using a $t^{ch}$-transition. This results in a final marking $M'$ with $q \in M'$. $M'$ is, however, a deadlock as in the underlying Petri net since $q$ could still progress to a commitment set place.

Following this we can argue that $\mathcal{G}_\mathcal{C}$ has a winning strategy that always commits if and only if $\mathcal{G}_\mathcal{C}^{ch}$ has a winning strategy.
It is easy to see that any winning strategy $\sigma$ for $\mathcal{G}_\mathcal{C}$ that always commits results in a winning strategy $\sigma_{ch}$ for $\mathcal{G}_\mathcal{C}^{ch}$: The branching process of $\sigma_{ch}$ is just extended by all places and transitions introduced with the, i.e., from every commitment set place an outgoing $t^{ch}$-transition is added. Since only the $\top$ places are added there is no bad place reachable in $\sigma_{ch}$. Since $\sigma$ always commits, firing one of the $t^{ch}$ transitions does not result in a deadlock, since every token can always move to a commitment set and afterwards either progress further or use a $t^{ch}$ transition to terminate. A winning strategy $\sigma_{ch}$ for $\mathcal{G}_\mathcal{C}^{ch}$ results in a winning strategy $\sigma$ for $\mathcal{G}_\mathcal{C}$ that always commits: The branching process of $\sigma$ is obtained by removing all places and transitions that where added in the construction of $\mathcal{G}_\mathcal{C}^{ch}$. The idea is, that, while in $\sigma_{ch}$ the player can terminate early using a $t^{ch}$ transitions, there is also the possibility of them just playing as if there is no challenge transition. As there is no bad place reachable in $\sigma_{ch}$ there are no bad places in $\sigma$ either. Now assume for contradiction that in $\sigma$ some place refused to commit in some marking $M$. Since $\sigma$ is obtained by removing parts of $\sigma_{ch}$ we get that $M$ is also a marking in $\sigma_{ch}$ but by the previous consideration this is not possible. Finally because every system player always commits, $\sigma$ is also deadlock-avoiding.

**Proposition 7** $\mathcal{G}_\mathcal{C}^{ch}$ has a winning strategy iff $\mathcal{G}_\mathcal{C}$ has a winning strategy where every system place always chooses a commitment set.

We can use this result to justify the assumptions made in our correctness proves (Sec. 6.3.2) since we can always modify $\mathcal{G}_\mathcal{C}$ to enforce commitment of all system player. We remark that $\mathcal{G}_\mathcal{C}^{ch}$ is not strategy-equivalent to $\mathcal{C}$ as a deadlock challenges can end a game even though the controller can continue to play. For every winning strategy for $\mathcal{G}_\mathcal{C}^{ch}$ there, however, is a "identical" winning strategy for $\mathcal{G}_\mathcal{C}$ that, itself, is bisimilar to a controller for $\mathcal{C}$. Even though $\mathcal{G}_\mathcal{C}^{ch}$ and $\mathcal{C}$ are not strategy-equivalent, they are winning equivalent.

## 6.5 On Size and Lower Bounds

As for our first translation, computation of $\mathcal{G}_\mathcal{C}$ is straightforward and local and the computational effort hence bounded by the size of the translation: The size of our translated $\mathcal{G}_\mathcal{C}$ is substantially bigger than the size of our translation from Chapter 5. The main reason for that is the restricted communication scheme of Petri games which requires us to make every synchronization explicit, i.e., encode it as a distinct transition. In control game the same action can fire from distinct preconditions. In our first translation from Chapter 5 we could move most complexity in the transition function by describing the automaton as the parallel composition and thereby implicitly build all possible combinations without explicitly encoding them in the actions. The size of $\mathcal{G}_\mathcal{C}$ hence naturally depends on the number of reachable states in $\mathcal{C}$. Let $\mathcal{R}(\mathcal{C})$ be the set of reachable global states in $\mathcal{C}$ that is

$$\mathcal{R}(\mathcal{C}) = \{s \in \prod_{p \in \boldsymbol{P}} S_p \mid \exists u \in \mathit{Plays}(\mathcal{C}), \mathit{state}(u) = s\}$$

We can then observe observe:

| | (1) | | (2) | | (3) |
|---|---|---|---|---|---|
| $\lvert\mathcal{P}\rvert \leq$ | $\lvert\bigcup_{p\in\boldsymbol{P}} S_p\rvert + \lvert\bigcup_{p\in\boldsymbol{P}} S_p\rvert \cdot 2^{\lvert\Sigma\rvert}$ | $+$ | $\lvert\boldsymbol{P}\rvert$ | $+$ | $\lvert\boldsymbol{P}\rvert$ |
| $\lvert\mathcal{T}\rvert \leq$ | $\lvert\Sigma\rvert \cdot \lvert\mathcal{R}(\mathcal{C})\rvert \cdot 2^{\lvert\Sigma\rvert+\lvert\boldsymbol{P}\rvert} + \lvert\bigcup_{p\in\boldsymbol{P}} S_p\rvert \cdot 2^{\lvert\Sigma\rvert}$ | $+$ | $\lvert\mathcal{R}(\mathcal{C})\rvert \cdot 2^{\lvert\Sigma\rvert+\lvert\boldsymbol{P}\rvert}$ | $+$ | $\lvert\bigcup_{p\in\boldsymbol{P}} S_p\rvert \cdot 2^{\lvert\Sigma\rvert}$ |

To ease the understanding the size bounds are separated in three parts. The first column **(1)** shows the bounds for the original description, the second **(2)** for the added deadlock-detection mechanisms and the last **(3)** for the mechanism used to enforce commitment. Note that all three parts are needed to obtain even winning equivalent games. The number of places is hence linear in the number of local states from $\mathcal{C}$ and exponential in the size of the alphabet. The amount of transitions of the from $(a, B, \{A_s\}_{p \in dom(a)})$ is bounded by $|\Sigma| \cdot |\mathcal{R}(\mathcal{C})| \cdot 2^{|\Sigma|+|\boldsymbol{P}|}$ and the amount of $\tau$-transitions by $|\bigcup_{p \in \boldsymbol{P}} S_p| \cdot 2^{|\Sigma|}$. For the deadlock detection we added a single place for each process and one transition for every artificial deadlock in $\mathfrak{D}_{DL}$. Every marking in $\mathfrak{D}_{DL}$ encodes both the current situation of the players as well as their commitment sets. The number of $t^M_{DL}$-transitions is hence bounded by $|\mathcal{R}(\mathcal{C})| \cdot 2^{|\Sigma|+|\boldsymbol{P}|}$. For the commitment enforcing we also added one place per process and added a transition for every commitment set, of which there are at most $|\bigcup_{p \in \boldsymbol{P}} S_p| \cdot 2^{|\Sigma|}$ many.

Overall we can note that the size of $\mathcal{G}$ is linear in the number of local states, exponential in the size of the alphabet and number of processes and, furthermore, scales linear with the number of reachable states in $\mathcal{C}$[6].

## Lower Bound

We can ask whether we can obtain a strategy-equivalent translation of polynomial size. In the following we show that a polynomial translation must either comprise additional player or destroy the structure of the underlying game, i.e., be not strategy-equivalent. To show this, we give a family of control games, s.t., every strategy-equivalent Petri game must be of exponential size (in the size of $|\Sigma|$).

In our own translation we had to duplicate actions into multiple transitions, to overcome the restrictive communication scheme of Petri games. In our lower bound we offer a Petri game to do the same, i.e., allow the same transition to occur from multiple distinct situations. In particular, our proof does not depend on the fact that any transition can only occur from a fixed precondition.

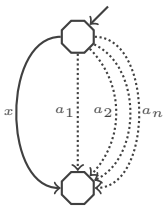



**Figure 6.11** Control game family $\{\mathcal{C}_n\}_{n \in \mathbb{N}}$ where every strategy-equivalent Petri game (with an equal number of player) must be of exponential size. The dotted action $(x)$ is uncontrollable. All other actions $a_1, \cdots, a_n$ are controllable.

Consider the control game family $\{\mathcal{C}_n\}_{n \in \mathbb{N}}$ depicted in Fig. 6.11. We fix $n$ and define $\mathcal{C} = \mathcal{C}_n$. The initial state has several outgoing controllable actions $(a_1, \cdots, a_n)$ and one outgoing uncontrollable action $(x)$. Let $\mathcal{G}$ be a Petri game that is strategy-equivalent to $\mathcal{C}$ and also contains only one player. For our lower bound we need the additional assumption that there are no infinite sequences of consecutive $\tau$-transitions possible in a winning strategy[7]. Whereas winning strategies in reachability games never permit such infinite sequences, we need to assume it for safety games. Since $\tau$-transitions conceptually correspond to internal computations of a model (not related to observable behavior), infinite consecutive $\tau$-sequences would correspond to infinite internal computations. As safety games require progress, infinite local computations are undesirable.

We can now show that there must be exponentially many places in $\mathcal{G}$. The idea is to consider any combination of actions from $a_1$ to $a_n$, i.e., any subset $B \subseteq \{a_1, \cdots, a_n\}$. There now is a controller for $\mathcal{C}$ that allows exactly the actions from $B$ and the uncontrollable $x$. By bisimulation we can show that there must be place where the observable outgoing transitions correspond to

[6] Note that the number of reachable states is exponential in $|\boldsymbol{P}|$

[7] There is in fact a Petri game that is strategy-equivalent to Fig. 6.11 of polynomial size (it permits possibly infinite $\tau$-transition sequences).

$B$, resulting in exponentially many places.

**Lemma 37** For every $\emptyset \neq B \subseteq \{a_1, \cdots, a_n\}$ there is a place $q_B$ such that

$$post^{\mathcal{G}}(q_B) = B \cup \{x\}$$

**Proof** Choose $\varrho$ as the (winning) controller that allows exactly the controllable actions in $B$ and $\sigma_\varrho$ as the bisimilar (winning) strategy for $\mathcal{G}$. Now let $M = \{q\}$ be a marking that is reached (in the strategy) by firing as many $\tau$-transitions as possible from the initial marking. This marking may not be unique but, by assumption, it exists.
Since $\varrho$ and $\sigma_\varrho$ are bisimilar and there are no $\tau$-transitions leaving $q$ we can conclude that $post^{\sigma_\varrho}(q) = B \cup \{x\}$ and therefore $B \cup \{x\} \subseteq post^{\mathcal{G}}(\lambda(q))$.
We claim that $\lambda(q)$ is an environment place. Assume for contradiction it is not, i.e., it is a system place. We now modify $\sigma_\varrho$ by removing every $x$ transition leaving place $q$. Call this modified strategy $\sigma'$. Note that, since $q$ is by assumption a system place, the resulting branching fulfils justified refusal, i.e., is indeed a strategy. It is, furthermore, easy to see that $\sigma'$ is still winning since whenever a token is in place $q$ all the other transitions in $B$ ($B \neq \emptyset$) are still possible and the behavior on them agrees with the behavior of the (winning) $\sigma_\varrho$.
By assumption there now is a bisimilar winning controller $\varrho_{\sigma'}$ to $\sigma'$. This is an immediate contradiction: In $\sigma'$ the place $q$ is still reachable (using only $\tau$-transitions) so it holds that $\{q\} \approx_{\mathfrak{B}} \epsilon$. We know that $x \in \mathit{Plays}(\mathcal{C}, \varrho_{\sigma'})$ as $x$ is uncontrollable but by construction of $\sigma'$ we get that $x \notin post^{\sigma'}(q)$. A contradiction to the bisimilarity of $\sigma'$ and $\varrho_{\sigma'}$. We hence know that $q$ is an environment place.
As $post^{\sigma_\varrho}(q) = B \cup \{x\}$, $q$ is an environments place and there is only one player it follows that $post^{\mathcal{G}}(\lambda(q)) = B \cup \{x\}$. $q_B = \lambda(q)$ now has the desired properties. ■

**Theorem 5** There is a family of control games $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ with $|\Sigma_n| = n$ such that every strategy-equivalent Petri Game (with an equal number of players) must have at least $\Omega(d^n)$ places for a $d > 1$.

**Proof** Follows from Lemma 37 with $d = 2$. ■

Our lower bound again indicates an exponential gap between control games and Petri games. While our translation from Sec. 6.3.2 shows that the difference between controllable and uncontrollable actions vs. system and environment places can be overcome, the lower bound outlines limitations and shows an intrinsic difference. Using our lower bound we can even pinpoint to the main restriction that requires exponentially many places: Places in Petri games allow for a restriction of either all or none of the behavior. As we saw, e.g., in Fig. 6.11, in control games a single state can comprise both controllable and uncontrollable behavior. This can not directly be matched by Petri games but must be explicitly encoded in the places. A distribution in system and environment places can hence not directly copy controllable and uncontrollable behavior of control games.

# Chapter 7

# New Decidable Classes

The main contribution of this work is the unification of control games and Petri games: Concurrency preserving Petri Games and asynchronous control games are equivalent. Apart from the theoretical insights this allows us to transfer existing results between both game types. In this chapter we exemplary outline newly identified decidable classes that can be obtained using our translations.

## 7.1 New Decidable Control Games

We call a process $p$ in a control game an *environment process* if all its actions are uncontrollable ($\Sigma_p \subseteq \Sigma^{env}$). A process that is not environment is called *system process*[1].

**Corollary** (Safety) control games with at most one environment process are decidable.

**Proof** We can adopt our translation from Chapter 6 slightly: For every state $s$ that has no outgoing controllable actions (i.e., $act(s) \subseteq \Sigma^{env}$) we do not add a system decision place as the only available commitment set would be the empty one anyway. Therefore all environment processes do not introduce any system places to the game. Decidability follows from [11]. ■

## 7.2 New Decidable Petri Games

Given a (reachability) Petri game $\mathcal{G} = (\mathcal{P}_\mathcal{S}, \mathcal{P}_\mathcal{E}, \mathcal{T}, \mathcal{F}, In, \mathcal{W})$ and a distribution in slices (or SNs) $\boldsymbol{S} = \{\varsigma_i\}_{i\in\mathfrak{I}}$. We can analyse the communication structure between the SNs by building the (undirected) *communication graph* $(V, E)$ where $V = \boldsymbol{S}$ and $E = \{(\varsigma_1, \varsigma_2) \mid \mathcal{T}^{\varsigma_1} \cap \mathcal{T}^{\varsigma_2} \neq \emptyset\}$. It is easy to see that this graph is isomorphic to the communication architecture of the constructed control game $\mathcal{C}_\mathcal{G}$ (as described in [15]). A slice distribution $\boldsymbol{S} = \{\varsigma_i\}_{i\in\mathfrak{I}}$ is called *acyclic* if the communication graph for this distribution is acyclic. We define the new class of Petri games $\mathcal{G}_{\text{ac}}$ as every Petri game that has an acyclic slice distribution. Using [15] we get the following results[2]:

**Corollary** Petri games in $\mathcal{G}_{\text{ac}}$ are decidable.

Acyclic Petri games are a very broad class of games that captures a lot of interesting situations. The results of this thesis are the first that allow for the synthesis of Petri games that comprise

[1] We remark that the following result holds even for the stronger formulation of bad situations used in [11]. We can hence translate control games where losing situations are defined globally.

[2] In [15] they imposed additional assumption on the automaton. Namely they required that every winning state has no outgoing actions. To be formal we can therefore only decide Petri games in $\mathcal{G}_{\text{ac}}$ where every winning place has no outgoing transitions.

two or more system and two or more environment players. Previously the result from [12] allowed to synthesize distributed startgies that interact on a single source of information. Opposed to that, [11] enabled synthesis of strategies that are themself not distributed but act in distributed environment and have to leverage communication to obtain information from different information sources. Our results allow us to, for the first time, use Petri games to synthesize strategies that are distributed themself *and* act in distributed environments. We can hence model interesting situations where the players need to gather information from distinct information sources and interact with each other by, e.g., exchanging the newly obtained information. We note that there are many interesting problems that can be described with an acyclic communication structure, e.g., client-server structures.

We do, however, need to remake that, while the communication graph of a slice distribution $\boldsymbol{S}$ and $\mathcal{C}_\mathcal{G}$ agrees, it might differ from the structure in $\widehat{\mathcal{C}_\mathcal{G}}$. We can therefore not generally extend our result to the synthesis of deterministic strategies but need to manually inspect whether the resulting $\widehat{\mathcal{C}_\mathcal{G}}$ has an acyclic structure[3].

## Lower Bound

We can also use our translation to transfer hardness results between both classes. In [15] they showed that deciding control games is non-elementary hard. As our translation is exponential, we immediately get a new hardness result for Petri games:

> **Corollary** Deciding concurrency-preserving, safe (reachability) Petri games is non-elementary hard.

There also exist hardness results for Petri games, namely the EXPTIME-hardness for bounded games with either at most one system or at most one environment player [12, 11]. Since our translation is exponential itself, this does, however, not yield any non-trivial lower bounds for control games.

[3]The added $\lightning$-actions add communication between processes that previously have not shared an action. While we can not give a general characteristics of situations where $\widehat{\mathcal{C}_\mathcal{G}}$ is acyclic, we can give a sufficient criterion: Let $(V, E)$ be the communication graph of a distribution $\boldsymbol{S}$. Define $E' = E \cup E^2$, i.e., add all transitive edges to bridge one transitive connection. If $(V, E')$ is acyclic then so is the communication architecture of $\widehat{\mathcal{C}_\mathcal{G}}$.

# Chapter 8

# Conclusion

Synthesis of distributed systems is of great relevance as it allows for automatic construction of real-world systems which are often of distributed nature. Synthesis in this framework is particularly hard because of the complicated information flow between the environment and the system. The existing approaches to identify classes for which the problem is decidable are either stated in terms of control games [19, 14, 15, 16] or Petri games [12, 11]. In both games, the system tries to use the limited controllability to restrict the behavior of the asynchronous system such that an objective is fulfilled. The decisive similarity in both is the transmission of complete information upon synchronization; the causal memory. In either formalism, general decidability is an open question.

In this thesis we used the concept of commitment sets, the idea of committing to a set of moves before they are actually executed, to overcome the different formalism in both games. We provided exponential translations between the games and showed that resulting games are strategy-equivalent, i.e., structure-preserving for strategies. In our correctness proofs we showed that, even if causal memory is modelled fundamentally different in both types, the underlying primitives are shared. We used partially ordered sets as a unified model of concurrent executions and showed that poset representations can be obtained for both Petri games as well as control games. This allowed us to translate the causal information from one player to the corresponding player in a different game type. As a consequence of our translations, we were able to translate existing decidability results to the respective different game type. Our findings hence allow for the unification of results in both types and hopefully lead to a unified effort to investigate the decidability-boundaries of asynchronous distributed synthesis. On the other hand, our lower bounds highlight an intrinsic difference between both types.

## 8.1 Future Work

**Translation Extensions** A translation, as proposed in this thesis, naturally brings up numerous questions. A first one being how the results of this work can be extended to a broader class of winning objectives. As control games previously focused on reachability and Petri games on safety objectives we presented one translation for safety and one for reachability. Since our proofs of bisimilarity are invariant under winning objectives it should be easy to extend the existing proofs to other objectives. We are certain that both translations work for safety *and* reachability objectives. In general, for both control games and Petri games it is worth studying generalisations to more expressive winning conditions such as LTL or Büchi.
Our translation from Petri games to control games (Chapter 5) is, furthermore, limited to concurrency-preserving games. On the other hand, having non-concurrency-preserving games, i.e., games modelling spawning and termination of player, enriches the expressiveness of Petri games. A future direction would be to try and extend the translation to such games, by, e.g.,

introducing dummy processes. On the side of Petri games there has been some effort to generate concurrency-preserving Petri games. Having a translation from non-concurrency-preserving Petri games to control games would immediately allow us to generate concurrency-preserving Petri games. Neither of both approaches seems easy.

**Determinism** Determinism is an interesting area for future development in both frameworks, as both game types, as of right now, restricts to either deterministic or nondeterministic strategies: While we have not attempted to define a suitable notion of determinism for control games, doing so and trying to extend existing results is fruitful. As control games do conceptually not distinguish between system and environment player there are many natural notions of determinisms: One could, for instance, require that in any situation from every local state at most one controllable action is possible. Alternatively one can demand that from every local state there is either at most one action possible or all possible actions are uncontrollable. Both exemplary attempts can be justified.
On the side of Petri games, the previous results [12, 11] all focused on deterministic strategies. As a future direction, it would be interesting to relax this notion and try to, e.g., extend existing results to the nondeterministic setting. Actually one minor side-result of this thesis is a reduction from deterministic Petri games to nondeterministic ones: We can translate a Petri game $\mathcal{G}$ to the control game $\widehat{\mathcal{C}_{\mathcal{G}}}$ (Chapter 5) and afterwards use our second translation (Chapter 6) to translate $\widehat{\mathcal{C}_{\mathcal{G}}}$ back into a Petri game $\mathcal{G}'$. It is easy to see that $\mathcal{G}'$ has (possibly nondeterminism) winning strategy if and only if $\mathcal{G}$ has a deterministic winning strategy. We can hence reduce the search for deterministic strategies to finding nondeterministic ones. While this requires to apply our translation twice and thus yields games of doubly exponential size, we conjecture that the same result can be achieved with an at most exponential blow-up.

**Deadlock Avoidance** Furthermore, it feels natural to extend the definition of deadlock-avoidance: Especially in the second translation, we noticed that deadlock-avoidance, stated as a global property, might not be well suited to require progress of a system as it allows players to terminate in unrealistic situations. In a system consisting of completely disjoint parts deadlocks are delusive as they entitle entire parts to deadlock as long as there is some, possibly completely unrelated player, that can continue playing. It might be worth considering stronger, local variants of deadlock-avoidance that require every player to progress locally[1].

**Unified Game Types** Our lower bounds showed an unavoidable difference between both types. While this is first and foremost of theoretical interest it motivates the pursue of unified game models, i.e., models that offer precise control about *who* can restrict behavior, as in Petri games, but at the same time allow to define *what* can be restricted, as in control games. Ideally one finds intermediate game representations that allow to encode both a Petri game and a control game without an exponential overhead. As our translations show that results can be applied to respective different formalisms, it is intriguing to look for minimal game descriptions that subsume both game types but where existing decidability results still yield interesting, transferable classes of decidability.

**New decidable classes** From a practical point of view, our translations allow to transfer decidability results between both models, two consequences of which were discussed in Chapter 7. An interesting direction for future work is to observe how the other previous results [19, 14, 16] relate to the respective different game, i.e., the question whether well defined, intuitive classes can be obtained. Especially for decomposable games [16], a rather unintuitive class of games, it is interesting to investigate whether the related class of Petri games is easier to grasp and offers a more structural definition. We can already give a negative answer to the question of

[1] Such formalism would allow us to omit the rather complicated mechanism used to enforce commitment from Sec. 6.4.

direct transferability of some classes: Petri games with at most one environment player [12] do not correspond to intuitive classes of control games (when using our presented translation). One can, however, attempt to modify our translation or find restrictions on control games (e.g., $\forall a \in \Sigma : |domain(\delta_a)| = 1$) to obtain transferable classes. This would, most likely, require more specific translations that are tailored towards one specific class.

**Action Based Control Games** As we briefly mentioned when summarizing previous work, control games can be seen as a game played between the actions rather than the process. In these action-based games all enabling decisions are made by actions, allowing decision that are based on the combination of the local information of all processes [22]. An interesting future direction would be to investigate how action-based control games relate to Petri games. That is, extending our translations to action-based control games. Doing so would, most likely, require to come up with a different notion of justified refusal that allows transitions to be prohibited based on the causal past of all involved tokens. A first idea would be to allow a strategy to forbid any transition as long as there is a system place in its precondition. Attempting a translation similar to the one done in this thesis is intriguing and would allow us to consider distributed synthesis from yet another perspective. If such a class of "transition-based" Petri games is found, one can, furthermore, investigate whether there is a similar relation as shown in [22] between both "types" of Petri games, i.e., show that ordinary Petri games can be reduced to "transition-based" ones.

# Bibliography


[1] Cyril Autant and Philippe Schnoebelen. Place bisimulations in petri nets. In *Proceedings of Application and Theory of Petri Nets*, pages 45–61, 1992.

[2] Raven Beutner, Bernd Finkbeiner, and Jesko Hecking-Harbusch. Translating asynchronous games for distributed synthesis. In *Proceedings of CONCUR*, pages 22:1–22:16, 2019.

[3] Raven Beutner, Bernd Finkbeiner, and Jesko Hecking-Harbusch. Translating asynchronous games for distributed synthesis (full version). *arXiv preprint arXiv:1907.00829*, 2019.

[4] Volker Diekert and Grzegorz Rozenberg. *The book of traces*. World scientific, 1995.

[5] Joost Engelfriet. Branching processes of Petri nets. *Acta Inf.*, 28(6):575–591, 1991.

[6] Javier Esparza and Keijo Heljanko. *Unfoldings – A Partial-Order Approach to Model Checking*. Springer, 2008.

[7] Bernd Finkbeiner. Bounded synthesis for petri games. In *Proceedings of Correct System Design - Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday*, pages 223–237, 2015.

[8] Bernd Finkbeiner. Synthesis of reactive systems. In Javier Esparza, Orna Grumberg, and Salomon Sickert, editors, *Dependable Software Systems Engineering*, volume 45, pages 72–98. IOS Press, 2016.

[9] Bernd Finkbeiner, Manuel Gieseking, Jesko Hecking-Harbusch, and Ernst-Rüdiger Olderog. Symbolic vs. bounded synthesis for petri games. In *Proceedings of SYNT@CAV*, pages 23–43, 2017.

[10] Bernd Finkbeiner, Manuel Gieseking, and Ernst-Rüdiger Olderog. Adam: Causality-based synthesis of distributed systems. In *Proceedings of CAV, Part I*, pages 433–439, 2015.

[11] Bernd Finkbeiner and Paul Gölz. Synthesis in distributed environments. In *Proceedings of FSTTCS*, pages 28:1–28:14, 2017.

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

[13] Bernd Finkbeiner and Sven Schewe. Uniform distributed synthesis. In *Proceedings of LICS*, pages 321–330, 2005.

[14] Paul Gastin, Benjamin Lerman, and Marc Zeitoun. Distributed games with causal memory are decidable for series-parallel systems. In *Proceedings of FSTTCS*, pages 275–286, 2004.

[15] Blaise Genest, Hugo Gimbert, Anca Muscholl, and Igor Walukiewicz. Asynchronous games over tree architectures. In *Proceedings of ICALP*, pages 275–286, 2013.

[16] Hugo Gimbert. On the control of asynchronous automata. In *Proceedings of FSTTCS*, pages 30:1–30:15, 2017.

[17] Jesko Hecking-Harbusch and Niklas O. Metzger. Efficient trace encodings of bounded synthesis for asynchronous distributed systems. In *Proceedings of ATVA*, 2019.

[18] Peep Küngas. Petri net reachability checking is polynomial with optimal abstraction hierarchies. In *6th International Symposium on Abstraction, Reformulation and Approximation*, pages 149–164, 2005.

[19] P. Madhusudan, P. S. Thiagarajan, and Shaofa Yang. The MSO theory of connectedly communicating processes. In *Proceedings of FSTTCS*, pages 201–212, 2005.

[20] Anca Muscholl. Automated synthesis of distributed controllers. In *Proceedings of ICALP*, pages 11–27, 2015.

[21] Anca Muscholl and Igor Walukiewicz. Distributed synthesis for acyclic architectures. In *Proceedings of FSTTCS*, pages 639–651, 2014.

[22] Anca Muscholl, Igor Walukiewicz, and Marc Zeitoun. A look at the control of asynchronous automata. *Perspectives in Concurrency Theory*, pages 356–371, 2009.

[23] Ernst-Rüdiger Olderog. *Nets, terms and formulas: three views of concurrent processes and their relationship*, volume 23. Cambridge University Press, 2005.

[24] Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In *Proceedings of ICALP*, pages 652–671, 1989.

[25] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *31st Annual Symposium on Foundations of Computer Science, 1990, Volume II*, pages 746–757, 1990.

[26] Wolfgang Reisig. *Petri Nets: An Introduction.* Springer, 1985.

[27] Sven Schewe and Bernd Finkbeiner. Synthesis of asynchronous systems. In *Proceedings of LOPSTR*, pages 127–142, 2006.

[28] W. M. Wonham and P. J. Ramadge. Modular supervisory control of discrete-event systems. *MCSS*, 1(1):13–30, 1988.

[29] Wieslaw Zielonka. Notes on finite asynchronous automata. *ITA*, 21(2):99–135, 1987.