Carl von Ossietzky Universität Oldenburg
Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Taking Synthesis of Distributed Systems via Petri Games to High Level, Symbolically

Von der Fakultät für Informatik, Wirtschafts- und Rechtswissenschaften der Carl von Ossietzky Universität Oldenburg zur Erlangung des Grades und Titels

Doktor der Naturwissenschaften (Dr. rer. nat.)

angenommene Dissertation

von Nick Würdemann
geboren am 12.01.1993 in Oldenburg

Gutachter:	Prof. Dr. Ernst-Rüdiger Olderog
	Prof. Dr. Javier Esparza
Tag der Disputation:	31.05.2024

Abstract

The manual implementation of local controllers for autonomous agents in a distributed and concurrent setting is an ambitious and error-prone task. Synthesis algorithms, however, allow for the automatic generation of such controllers, given a formal specification of the system's goal. P/T Petri games are a multi-player game model for the synthesis problem in distributed systems. The model represents causal memory of the players, which are tokens on a P/T Petri net and divided into two teams: the controllable system and the uncontrollable environment. For one environment player and a bounded number of system players, the problem of solving Petri games can be reduced to that of solving two-player games. This is achieved by constructing a corresponding two-player game for a given Petri game.

In this thesis we consider high-level Petri games, which provide concise representations of P/T Petri games. We show how symmetries, derived from a high-level representation, can be exploited to significantly reduce the state space in the corresponding two-player game. We present a solving algorithm for a subclass of high-level Petri games with one environment player and a bounded number of system players with a reachability or safety objective. The core of the algorithm is the construction of a symbolic two-player game whose states are symmetry-equivalence classes of the two-player game that corresponds to the represented P/T Petri game. We additionally present a second construction of this symbolic two-player game by defining unique, canonical representations of its states.

Strategies in a P/T Petri game are defined as prefixes of its unfolding. Unfoldings provide a well-known partial-order semantics of P/T Petri nets that can be applied to various model checking or verification problems. For high-level Petri nets, the so-called symbolic unfolding generalizes this concept. A complete finite prefix of the unfolding of a P/T Petri net contains all information to verify, e.g., reachability of markings.

In this thesis we define complete finite prefixes of the symbolic unfolding of high-level Petri nets. For a class of safe high-level Petri nets, we generalize the well-known algorithm by Esparza, Römer and Vogler for constructing small complete finite prefixes. Additionally, we identify a more general class of nets with infinitely many reachable markings, for which an approach with an adapted cut-off criterion extends the complete prefix methodology, in the sense that the original algorithm cannot be applied to the P/T net represented by a high-level net. Finally, we give an outlook on how to define symbolic strategies for high-level Petri games in the the symbolic unfolding.

Zusammenfassung

Die manuelle Implementierung lokaler Controller für autonome Agenten in einem verteilten und parallelen System ist eine ambitionierte und fehleranfällige Aufgabe. Synthesealgorithmen ermöglichen die automatische Generierung solcher Controller, basierend auf einer formalen Spezifikation des Ziels des Systems. P/T-Petri-Spiele sind ein Mehrspieler-Spiel-Modell für das Synthese-Problem in verteilten Systemen. Das Modell repräsentiert die kausale Erinnerung der Spieler, welche Tokens auf einem P/T-Petri-Netz sind und in zwei Teams unterteilt sind: das kontrollierbare System und die unkontrollierbare Umgebung. Für einen Umgebungsspieler und eine begrenzte Anzahl von Systemspieler kann das Lösen von Petri-Spielen auf das Lösen von Zwei-Spieler-Spielen reduziert werden. Dies wird durch die Konstruktion eines entsprechenden Zwei-Spieler-Spiels für ein gegebenes Petri-Spiel erreicht.

In dieser Arbeit befassen wir uns mit höheren Petri-Spielen, welche prägnante Repräsentationen von P/T-Petri-Spielen ermöglichen. Wir zeigen, wie Symmetrien, die aus einer solchen Repräsentation abgeleitet werden, genutzt werden können, um den Zustandsraum im entsprechenden Zwei-Spieler-Spiel signifikant zu reduzieren. Wir präsentieren einen Lösungsalgorithmus für eine Unterklasse von höheren Petri-Spielen mit einem Umgebungsspieler und einer begrenzten Anzahl von Systemspielern mit einem Erreichbarkeits- oder Sicherheitsziel. Der Kern des Algorithmus besteht in der Konstruktion eines symbolischen Zwei-Spieler-Spiels, dessen Zustände Symmetrieäquivalenzklassen des Zwei-Spieler-Spiels sind, das dem repräsentierten P/T-Petri-Spiel entspricht. Zusätzlich präsentieren wir eine zweite Konstruktion dieses symbolischen Zwei-Spieler-Spiels durch die Definition eindeutiger, kanonischer Darstellungen seiner Zustände.

Strategien in einem P/T-Petri-Spiel werden als Präfixe seiner Entfaltung definiert. Entfaltungen bieten eine wohlbekannte Partielle-Ordnungs-Semantik von P/T-Petri-Netzen, die auf verschiedene Model-Checking- oder Verifikationsprobleme angewendet werden kann. Für höhere Petri-Netze verallgemeinert die sogenannte symbolische Entfaltung dieses Konzept. Ein vollständiges endliches Präfix der Entfaltung eines P/T-Petri-Netzes enthält alle Informationen zur Überprüfung von beispielsweise der Erreichbarkeit von Markierungen.

In dieser Arbeit definieren wir vollständige endliche Präfixe der symbolischen Entfaltung von höheren Petri-Netzen. Für eine Klasse sicherer höherer Petri-Netze verallgemeinern wir den bekannten Algorithmus von Esparza, Römer und Vogler zur Konstruktion kleiner vollständiger endlicher Präfixe. Zusätzlich identifizieren wir eine allgemeinere

Klasse von Netzen mit unendlich vielen erreichbaren Markierungen, für die ein Ansatz mit einem angepassten Abbruchkriterium die Methodik des vollständigen Präfixes erweitert, da der ursprüngliche Algorithmus nicht auf das P/T-Netz angewendet werden kann, welches durch ein höheres Netz repräsentiert wird. Letztlich geben wir einen Ausblick darauf, wie man symbolische Strategien für höhere Petri-Spiele in der symbolischen Entfaltung definieren kann.

Acknowledgments

My heartfelt thanks to all who played a role in my PhD journey; though space limits recognition, your support is deeply appreciated, even if not expressly stated.

I owe my deepest gratitude to my supervisor, Ernst-Rüdiger Olderog. From the moment I aimlessly stumbled into his office inquiring about the possibilities of pursuing a PhD in Computing Science, he has been my guide and mentor. Since he is integral to nearly every group of people mentioned below, I wanted to express my gratitude to him separately inside this paragraph. I am especially thankful for providing me with many opportunities to attend conferences and summer schools, and most significantly, supported my extended research trip to Paris.

This brings me to Stefan Haar, who graciously welcomed me into his group in Paris for a total of eight weeks. This period became a highly enjoyable and fruitful time of scientific collaboration, for which I am profoundly grateful.

I extend my heartfelt thanks to Martin Fränzle, Javier Esparza, Astrid Rakow, Eike Best, Annegret Habel, and Heike Wehrheim for being part of my thesis committee and/or providing invaluable feedback on various occasions.

I thank Lukas, Manuel, Okan, Stefan and Thomas for the excellent scientific collaborations. I also want to express my gratitude to the many proofreaders of parts of my thesis, namely Anna, Christopher, Henk, Helen, and Paul. Their merciless yet effective feedback greatly improved the readability and quality of my thesis.

The members of the "Correct System Design" group and its friends made my time truly enjoyable, always fostering interesting discussions, both scientific and personal. A special thanks goes to Manuel, who guided me through my first two publications and served as a role model for research quality and teaching. Due to the countless hours I spent talking to him while standing in his doorway, I wouldn't be surprised if my outline is imprinted in the doorframe of his office. After Manuel completed his PhD, I found equally pleasant company in Christopher's and Paul's offices, where they always had an open ear for my rants about anything, especially when it came to writing my PhD thesis. I am also grateful to Cedric, Christoph, Christian, Erzana, Heinrich, Jan, Lara, Lars, Maike, Okan, and Uli for making my time in the department so pleasurable.

Being part of the research training group SCARE was the perfect starting environment for me at the beginning of my PhD journey. I appreciate all the other SCARIES, especially Ira, who warmly welcomed me on my first day at the department and was always there for me. Additionally, I thank Andrea, Grigorios, Marion, Nico, and Niko for their technical and administrative support over the years.

Without the unwavering support of my friends, who stood by me through both the highs and lows of this journey, from the conclusion of my master's studies to the successful defense of my PhD, this work would not have been possible. Thank you Anna, Aline, Carlsson, Caro, Carmen, Christopher, Elisa, Henk, Henning, Heike, Lara, Leah, Maike, Manuel, Niklas, Okan, Paul, Piet, and Ronja.

Mama, Papa, and Lynn, I am grateful for your endless and unconditional support in all of my endeavors, and especially during the last six years. I am very glad to have you in my life.

The greatest constant over all this time, my safety net and my rock in the storm, has been knowing that I am loved by the most important person in my life. Helen, you are the best. I love you.

Contents

Chapter 1

Introduction

Contents

Due to constant electronic and digital expansion, our society has become increasingly dependent on computer systems, such that they are no longer only specialized tools, but integral parts of our daily lives. From the integration of technology into electronic banking or the reliance on networked computers in flight control systems, reliability of these systems is paramount. While a malfunctioning self-service banking terminal may be a minor inconvenience, a fault in critical systems, such as a plane's navigation or communication systems, can have severe consequences [BK08].

The evolution of computer systems has led to the creation of complex, decentralized distributed systems composed of numerous interconnected computers [TS07]. This decentralization, while improving efficiency, comes at a cost: though an entire system may appear as one unit, the local controllers in a network often act autonomously on incomplete information to avoid constant communication. In particular, the challenges of implementing local controllers and facilitating mutual communication in asynchronous distributed systems, where components progress at individual rates, have become apparent, especially in manufacturing contexts [Mis12; MH08].

The increasing size and complexity of these systems create a major challenge for humans in implementing robust controllers accurately. *Synthesis* [Chu57; Chu62] addresses this challenge by automating the generation of controllers. This process involves modeling all potential system actions and behaviors from the system's environment, and a specification outlining the system's overarching goal. The derived controllers ensure resilience against all modeled environment behavior. For single-process models, synthesis approaches have found success in nontrivial applications (e.g., [BGJ+07], [KFP09]). In this case, we talk about *monolithic synthesis*.

However, as mentioned above, more and more systems today are decentralized. Here, the monolithic approach is not applicable. In this setting the task of *synthesis of distributed systems* [PR90; MT01; KV01] is to automatically construct a *set* of implementations, each for a specific system component. These implementations must collectively satisfy the given specification of the distributed system. An important aspect is determining at which states and to what extent information exchange between the individual processes can lead to the achievement of their goals [Fin16].

This thesis makes contributions to exploring the synthesis of distributed systems by employing state-space reduction techniques derived from *model checking*, which is the assessment of whether a given implementation adheres to a specified property. The techniques involved encompass the utilization of symmetries and abstraction, focusing particularly on systems characterized by substantial symmetric behavior.

1.1 Synthesis of Distributed Systems via Petri Games

Petri games constitute a model-based and game-based approach for the synthesis of distributed systems. We start by providing explanations for the terms model-based and game-based:

In a *model-based* approach, we work with mathematically precise and unambiguous descriptions of the system's behavior and the correctness requirements [BK08]. The effectiveness of a synthesis algorithm depends on the quality of the model and the formalization of the system. This problem is common in the world of model checking [BK08], where creating appropriate inputs is a significant and non-trivial task [Roz16]. However, this thesis does not deal with the details of modeling techniques, but focuses on the models themselves.

In the context of *game-based* synthesis [BL69], the synthesis problem is traditionally conceptualized as an infinite game on a finite graph with two players. One player, embodying the system, aims to fulfill the given specification, while the other player, representing the environment, tries to induce a violation. The states of the game correspond to the vertices of the graph which are uniquely assigned to either the environment or the system. In a game state, the player associated with the corresponding vertex selects the next vertex, respecting the graph's edges. This decision is made with complete knowledge of the graph structure and all preceding moves. In the realm of reactive systems, such games typically continue infinitely. As a result, these games are infinite in terms of moves but finite in terms of the state space, i.e., the graph in which they unfold. A strategy devised for the system player to ensure victory in the game against all possible behaviors of the environment constitutes an implementation that is assured to meet the specified requirements [Fin16; BCJ18].

In this thesis, the central model-based and game-based approach for the synthesis of distributed systems is constituted by *Petri games*. In the Petri games framework, the foundational model are *(P/T) Petri nets* [Rei13]. In a Petri net, the actions of a system are represented through entities known as *transitions*. These transitions are interconnected, establishing relationships among actions. The connection is facilitated by introducing conditions, represented by *places*, which are linked to transitions through directed edges, creating a directed, bipartite graph.

Each place has the capacity to hold *tokens*, serving as a formalization of the conditional nature of places. For an action to be executed, a specific number of tokens must be present on each place connected to the transition via an ingoing edge. During execution, the transition consumes these tokens from the places and deposits a token on every place linked to it by an outgoing edge. This, in turn, can potentially enable the execution of other transitions. A state in such a Petri net is called *marking*, and describes the number of tokens on each of the distributed places. Consequently, Petri nets offer a robust means of modeling distributed systems, as the interdependencies or independencies between actions can be visually represented through a bipartite graph featuring transitions and places.

Petri games, introduced by Finkbeiner and Olderog [FO14; FO17], extend the Petri net concept into a game-theoretic formalism. In such a game, tokens within an underlying Petri net assume the roles of *players*. The game is played between two teams: the *system players*, representing controllable behavior, and the *environment players*, embodying uncontrollable behavior. To facilitate this, the places in the Petri net are categorized into *system places* and *environment places*, determining a player's team based on the place the token occupies. Players situated in separate parts of the net lack information about each other until they engage in communication, i.e., a joint transition. During such interactions, players share knowledge about their *causal history*, encompassing the places and transitions where the player previously resided or contributed to the execution, as well as their individual histories.

The specification in Petri games is traditionally expressed through a *safety condition*: the objective for the system players is to consistently avoid reaching any designated *bad places* within the net. In a *strategy*, players can either permit or prohibit transitions that are dependent on their involvement. The decisions are (only) based on their causal history that defines their knowledge and place. Formally, a strategy is defined as a subprocess of the *unfolding*, which, in its turn, is an acyclic Petri net encompassing all potential executions and behaviors of the system as represented by the original Petri net. The definition of strategies as a subprocesses means that all unwanted behavior is "cut off" from the unfolding. From such a strategy, the set of implementations for the system components modeled in the original Petri net can be derived [FO17].

The task of *solving* a given Petri game involves determining the existence of a winning strategy for the system players, ensuring the given specification. In the affirmative scenario, the goal is to compute this strategy. Thus, this problem effectively models the synthesis problem for distributed systems. In this thesis, we primarily focus on an approach presented in [FO14; FO17; Gie22]: a given Petri game is reduced to an infinite game over a finite graph with complete knowledge. In this two-player game, a winning strategy for the system can be found using common game solving techniques [GTW02]. This strategy can then be translated back to a strategy for the system players in the Petri game.

High-level representations of Petri games, denoted as *high-level Petri games* [GO21], offer a concise means of describing Petri games. High-level Petri games are based on the formalism of *high-level Petri nets* [GL81; Jen96]. These high-level Petri nets allow the representation of several individual and distinguishable tokens residing on a single place by introducing the notion of *colors*. Arcs around transitions are labeled by *variables*, which can be instantiated in so-called *modes* of the transition, specifying the colors that are moved. In practical applications, the modeling of high-level Petri games often leads to corresponding P/T Petri games that exhibit a substantial amount of symmetric behavior. The reason for this is that, in many cases, individual tokens (representing robots, processes, workpieces) do not necessitate markedly different behaviors to achieve their objective in the game.

1.2 Contributions

This thesis contributes to the synthesis of distributed systems using Petri games by investigating the integration of state-space reduction techniques derived from model checking high-level Petri nets into solving algorithms for high-level Petri games. Two distinct approaches are explored: first, we investigate the application of the concept of "symmetries" to the two-player game used in the Petri game reduction from [FO14; FO17; Gie22]. Second, an examination of "complete finite prefixes" of the "symbolic unfolding" of high-level Petri nets is conducted, setting the foundation for forthcoming efforts in defining and synthesizing symbolic strategies.

Figure 1.1 serves as an overview of these contributions (marked dark gray) in the context of the existing framework (marked light gray), and its structure will be gradually explained during this section. We have two types of solid directed edges in this figure: without and with a hook. Directed edges without a hook can be read as "can be generated from", while directed edges with a hook can be read as "can be embedded into". The latter appear in the context of unfoldings, displayed on the right-hand side in the figure. The other relations (dashed and undirected, labeled edges) will be explained later.

The bottom left part illustrates the reduction of Petri games to two-player games from [FO14; FO17; Gie22] mentioned above: for a given Petri game, a corresponding two-player game is generated (written as "2-Player game" in the figure). This two-player game can be solved, and its strategy can then be translated into a Petri game strategy. This Petri game strategy is a subprocess of the Petri game's unfolding.

Exploiting Symmetries in Solving Petri Games. The focus of this thesis is on high-level Petri games, each of which represents a unique P/T Petri game we refer to as its "expansion" [GO21]. This expansion is obtained by explicitly listing all place-color combinations, and all transitions-mode combination, i.e., all variable instantiations. In Fig. 1.1, this is illustrated by a corresponding edge from "High-level Petri game" to "Petri game".

We introduce a solving algorithm designed for a specific subclass of high-level Petri games with a single environment player and a bounded number of system players. This subclass is defined by three key restrictions. Firstly, we exclusively consider "expansion safe" high-level Petri games, ensuring that, in the represented P/T Petri game, each reachable marking contains at most one token on every place. Secondly, we assume the single environment player to be "recurrently interfering", implying its participation in infinitely many transitions within every infinite transition sequence. Lastly, in alignment with [Gie22], we exclude Petri games involving "mixed communication", denoting a constraint on the structure of the underlying high-level Petri net.

The novel algorithm [GOW20] leverages the *symmetries* [Sch00a] inherent in the system, coming from the high-level representation. At its core, the algorithm combines the reduction technique employed in the associated class of P/T Petri games, as detailed in [FO17; Gie22], with the creation of a *symbolic reachability graph* for High-level Petri nets, as presented in [CDFH97]. In particular, we introduce the so-called *symbolic two-player game* (denoted as "symbolic 2-Player game" in Fig. 1.1). The vertices in this symbolic two-player game are equivalence classes w.r.t. symmetries of the vertices in the two-player game presented in [Gie22], corresponding to the High-level Petri game's expansion. The correctness of this construction is established through a bisimulation between these two two-player games (edge labeled by "$\sim$" in Fig. 1.1). Furthermore, we provide an algorithm to derive a winning positional strategy in the "original" two-player game from [Gie22] based on a winning positional strategy in the symbolic two-player game (directed edge between the two strategies in Fig. 1.1).

We validate the state space reduction of our approach using a prototype implementation on a set of benchmark families introduced in [FGO15; FGHO17; GOW20]. The experimental results, calculated with a two-hour timeout, demonstrate a state space reduction of up to three orders of magnitude.

Formally, the vertices in the symbolic two-player game are, as mentioned above,

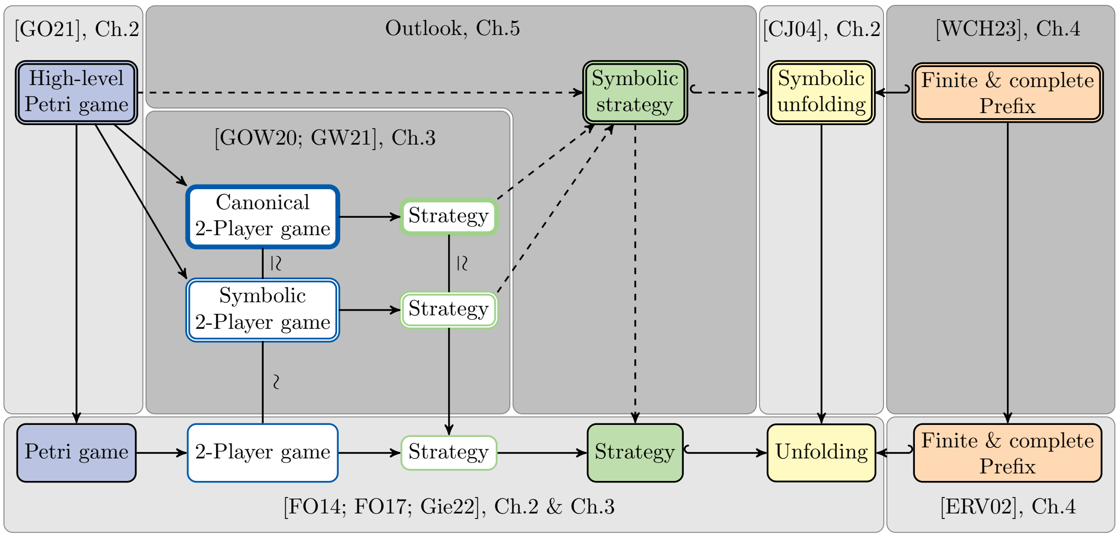


Figure 1.1: Contributions of this thesis (dark gray) in the context of the existing framework of high-level Petri games and symbolic unfoldings (light gray).

equivalence classes. However, in the construction of the game, we use *arbitrary representatives* of the classes to build the arcs between the vertices. Consequently, during the construction of the game, after reaching a new vertex, we have to compare it to all other representatives computed up to that point and test them for equivalence with respect to symmetry. This process is referred to as answering the *orbit problem* [CEJS98].

We explore a second approach [GW21a] that employs so-called *canonical representations* of the equivalence classes. We use techniques presented in [CDFH91a; CDFH93] that address a similar problem for markings in the symbolic reachability graph of a high-level Petri net. The canonical representation of a new vertex can be automatically calculated. Instead of comparing it to every other vertex in the game for *equivalence* with respect to symmetry, which can be expensive, we only have to check for *equality*. This is referred to as the *constructive orbit problem* [CEJS98]. By employing these canonical representations as vertices, we arrive at a two-player game known as the *canonical two-player game* (denoted by "Canonical 2-Player game" in Fig. 1.1). We demonstrate that this game is isomorphic to the symbolic two-player game, enabling the transfer of all achieved results (edges labeled by "$\simeq$" in Fig. 1.1). Finally, we compare the speed of construction for these two games on the benchmarks mentioned above, concluding that, in many cases, the canonical two-player game is faster to construct.

Complete Finite Prefixes of Symbolic Unfoldings. Since strategies in Petri games are defined as subprocesses of the underlying Petri net's unfolding, the latter is a crucial object in this thesis. The notion of subprocesses can be generalized to so-called *prefixes* of the unfolding. A *complete* prefix of a Petri net's unfolding contains all information to verify, e.g., reachability of markings. In [McM95], an algorithm for computing complete finite prefixes is presented. In [ERV02], a total order on states in the unfolding (called an *adequate order*) is used to give an improved algorithm calculating comparably small complete finite prefixes of the unfolding. We call this improved algorithm the "ERV-algorithm", after its authors Esparza, Römer, and Vogler. In Fig. 1.1, these prefixes are indicated right of the unfolding.

At first glance, high-level representations on the one hand and processes (resp. unfoldings) of P/T Petri nets on the other, seem to be conflicting concepts – one being a more concise, the other a more detailed description of the net('s behavior). However, in [CJ04], *symbolic branching processes* and *unfoldings* of high-level Petri nets are defined. The symbolic unfolding is represented above the unfolding in Fig. 1.1. In [Cha06], it is argued that in general, there exists no complete finite prefix of the symbolic unfolding of a high-level Petri net. However, this is only true for high-level Petri nets with infinitely many reachable markings such that the number of steps needed to reach them is unbounded, in which case the same arguments yield an example for P/T Petri nets.

In this thesis, we lift the concepts of complete prefixes and adequate orders to the level of symbolic unfoldings of high-level Petri nets [WCH23]. We consider the class of *safe* high-level Petri nets (i.e., in all reachable markings, every place carries at most one token) that have finitely many reachable markings. This class generalizes safe P/T Petri nets, and we obtain a generalized version of the ERV-algorithm creating a complete finite prefix of the symbolic unfolding of such a given high-level Petri net. Our results are a generalization of [ERV02] in the sense that if a P/T Petri net is viewed as a high-level Petri net then the new definitions of adequate orders and completeness of prefixes on the symbolic level, as well as the algorithm producing them, all coincide with their P/T counterparts. This part is marked in Fig. 1.1 by the elements in the top right dark gray area. We evaluate this generalized algorithm through a prototype implementation on four benchmark families introduced in [WCHP23].

We proceed to identify an even more general class of so-called *symbolically compact* high-level Petri nets, where we drop the assumption of finitely many reachable markings, and instead assume a uniform bound on the number of steps needed to reach any reachable marking. In such a case, the expansion is possibly not finite, in which case the original ERV-algorithm from [ERV02] is not applicable. We adapt the generalized ERV-algorithm by weakening the cut-off criterion to ensure finiteness of the resulting prefix. To check this cut-off criterion, we have to compare infinite sets of markings. We overcome this obstacle by symbolically representing these sets, making the cut-off criterion decidable.

An outlook on Symbolic Strategies. We present a brief preview of the integration of (complete finite prefixes of) symbolic unfoldings with high-level Petri games. These findings are currently unpublished and have not undergone peer review. We propose a definition of symbolic strategies (represented by the node "Symbolic Strategy" in Fig. 1.1) and provide a glimpse into the potential for synthesizing these strategies. The dashed lines in the figure indicate our exploration into whether the symbolic strategy can be formalized as a prefix of the symbolic unfolding, how it corresponds to a P/T Petri game strategy, and whether the definition aligns with the concept of the symbolic/canonical two-player game.

I come to the conclusion that symbolic strategies cannot be properly defined by just a prefix notation. Instead, I propose an alternative definition and then argue that the symbolic two-player game would have to be significantly modified to be able to generate symbolic strategies.

1.3 Structure of this Thesis

Following this introduction, Chapter 2 lays the groundwork with formal preliminaries for Petri nets and Petri games. In Chapter 3, we delve into the symmetry-exploiting solving algorithm tailored for high-level Petri games. This includes the introduction of the symbolic two-player game initially without (Sec. 3.2) and subsequently with canonical representations (Sec. 3.3). Chapter 4 presents the definition and construction of complete finite prefixes of symbolic unfoldings of high-level Petri nets. Chapter 3 and Chapter 4 are designed to not rely on each other, allowing readers to explore them independently. Chapter 5 presents the brief outlook into the definition and synthesis of symbolic strategies for high-level Petri games. This organizational structure is visually represented in Fig. 1.1 by the marking the corresponding areas with Ch.2, Ch.3, Ch.4, and Ch.5.

1.4 Publications

This thesis is based on the following peer-reviewed publications that I co-authored during my doctoral studies:

[GOW20] Manuel Gieseking, Ernst-Rüdiger Olderog, and Nick Würdemann. "Solving high-level Petri games". In: *Acta Informatica* 57.3-5 (2020), pp. 591–626. URL: https://doi.org/10.1007/s00236-020-00368-5

[GW21a] Manuel Gieseking and Nick Würdemann. "Canonical Representations for Direct Generation of Strategies in High-Level Petri Games". In: *Application and Theory of Petri Nets and Concurrency - 42nd International Conference, PETRI NETS 2021, Virtual Event, June 23-25, 2021, Proceedings.* Vol. 12734. Lecture Notes in Computer Science. Springer, 2021, pp. 95–117. URL: https://doi.org/10.1007/978-3-030-76983-3_6

[Wür21] Nick Würdemann. "Exploiting symmetries of high-level Petri games in distributed synthesis". In: *it Inf. Technol.* 63.5-6 (2021), pp. 321–331. URL: https://doi.org/10.1515/itit-2021-0012

[WCH23] Nick Würdemann, Thomas Chatain, and Stefan Haar. "Taking Complete Finite Prefixes to High Level, Symbolically". In: *Application and Theory of Petri Nets and Concurrency - 44th International Conference, PETRI NETS 2023, Lisbon, Portugal, June 25-30, 2023, Proceedings.* Vol. 13929. Lecture Notes in Computer Science. Full version available at https://inria.hal.science/hal-04029490. Springer, 2023, pp. 123–144. URL: https://doi.org/10.1007/978-3-031-33620-1_7

We were invited to submit an extended version of the last paper, [WCH23], to a special issue of Fundamenta Informaticae on Petri Nets 2023. This version, which in big parts is included in the present thesis, is currently under review, but already accessible online:

[WCHP23] Nick Würdemann, Thomas Chatain, Stefan Haar, and Lukas Panneke. *Taking Complete Finite Prefixes To High Level, Symbolically.* Submitted to Fundamenta Informaticae. 2023. arXiv: 2311.11443 [cs.LO]

Parts of this thesis are additionally based on the full version [GW21b] of [GW21a]. The content of the full version of [WCH23] is included in [WCHP23].

In all of the aforementioned papers, my contributions encompassed the development and formalization of the theoretical framework, including the provision of proofs. In [GOW20], the corresponding symbolic two-player game for a given high-level Petri game is defined, while [GW21a] is concerned with the canonical two-player game. In these two publications, the novel benchmark families were developed together with Manuel Gieseking. These benchmark families are not explicitly presented in this thesis. [Wür21] is an overview article that also gives gives a first outlook on symbolic strategies. [WCH23; WCHP23] describe the results about finite complete prefixes of the symbolic unfolding. In [WCHP23], the novel benchmark families were developed together with Lukas Panneke, and are presented in this thesis.

Two additional peer reviewed publications that I co-authored during my doctoral studies are not directly related to this thesis:

[ÖW21] Okan Özkan and Nick Würdemann. “Resilience of Well-structured Graph Transformation Systems”. In: *Proceedings Twelfth International Workshop on Graph Computational Models, GCM@STAF 2021, Online, 22nd June 2021*. Vol. 350. EPTCS. 2021, pp. 69–88. URL: `https://doi.org/10.4204/EPTCS.350.5`

[AHP+22] Giann Karlo Aguirre-Samboní, Stefan Haar, Loïc Paulevé, Stefan Schwoon, and Nick Würdemann. “Avoid One’s Doom: Finding Cliff-Edge Configurations in Petri Nets”. In: *Proceedings of the 13th International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2022, Madrid, Spain, September 21-23, 2022*. Vol. 370. EPTCS. 2022, pp. 178–193. URL: `https://doi.org/10.4204/EPTCS.370.12`

Chapter 2

Preliminaries – Petri Nets and Petri Games

Contents

In this chapter, we establish a foundational basis for the notation used in this work and introduce two primary objects: *Petri nets* [Rei13] and *Petri games* [FO14; FO17]. We discuss so-called *branching processes* [Eng91] of Petri nets, and explore their maximal version, termed *unfolding* [NPW81]. The unfolding process plays a crucial role in facilitating the behavior of Petri nets and Petri games.

Following this, we introduce a generalization of Petri nets and Petri games known as *high-level Petri nets* [Jen96] and *high-level Petri games* [GO21], respectively. We conclude the preliminaries by presenting the corresponding generalized notion of branching process and unfolding, referred to as *symbolic branching* and *symbolic unfolding* [CJ04].

Mathematical Notation and Multi-sets. As a prelude to our Preliminaries, we briefly establish mathematical notation, and recall a modification of the well-known mathematical concept of a set. Unlike a conventional set, this modification allows for multiple instances of each element and is referred to as a *multi-set.*

We denote the natural numbers by $\mathbb{N} = \{0, 1, 2, \dots\}$. For a set X, we denote by $\mathbb{P}(X)$ its power set. We write $X_1 \sqcup X_2$ for the *disjoint union* of two sets X_1, X_2, and generalize this by writing $\bigsqcup_{i=1}^{n} X_i$ for the disjoint union of sets $X_1, \dots, X_n$. For a set X, we denote by id_X the *identity function* given by $x \mapsto x$ for all $x \in X$. We omit the index and just write *id* when no confusion arises. For functions $f : X \to Y$ and $g : X' \to Y'$ with $Y \subseteq X'$, the *composition* $g \circ f : X \to Y'$ (read: "g after f") maps each $x \in X$ to $g(f(x))$. For $n \in \mathbb{N}$, $(x_i)_{i=1}^{n}$ denotes the tuple $(x_1, \dots, x_n)$. Analogously, for numbers $a_i \in \mathbb{N}$, $i = 1, \dots, n$,

$$((x_{i,j})_{j=1}^{a_i})_{i=1}^{n} := (x_{1,1}, x_{1,2} \dots, x_{1,a_1}, x_{2,1}, x_{2,2} \dots, x_{2,a_2}, x_{n,1}, x_{n,2} \dots, x_{n,a_n}).$$

For a set X, we call functions $A : X \to \mathbb{N}$ *multi-sets over* X, and denote $x \in A$ iff $A(x) \geq 1$. For two multi-sets A, A' over the same set X, we write $A \leq A'$ iff $\forall x \in X : A(x) \leq A'(x)$, and denote by $A + A'$ and $A - A'$ the multi-sets over X given by $(A + A')(x) := A(x) + A'(x)$ and $(A - A')(x) := \max(A(x) - A'(x), 0)$. We use the notation $\{\!\!\{ \dots \}\!\!\}$ as introduced in [KK03]: elements in a multi-set can be listed explicitly as in $\{\!\!\{ x_1, x_1, x_2 \}\!\!\}$, which describes the multi-set A with $A(x_1) = 2$, $A(x_2) = 1$, and $A(x) = 0$ for all $x \in X \setminus \{x_1, x_2\}$. A multi-set A is called finite iff there are finitely many $x \in X$ such that $A(x) > 0$. In such a case, $\{\!\!\{ f(x) \mid x \in A \}\!\!\}$, where $f(x)$ is an object constructed from $x \in X$, denotes the multi-set A' over $f(X)$ such that $\forall y \in f(X) : A'(y) = \sum_{x \in X \wedge f(x) = y} A(x)$. For a mapping $f : X \to Y$ and a finite multi-set A over X, we define the multi-set $f(A) := \{\!\!\{ f(x) \mid x \in A \}\!\!\}$ over Y. We denote by $|A| := \sum_{x \in X} A(x)$ the *cardinality* of A.

2.1 Place/Transition Petri Nets and Unfoldings

This subsection marks the actual beginning of our preliminaries. Given the thematic focus of this thesis, there is no better way to start than by defining what a Petri Net is. This definition is followed by a recap of some fundamental aspects of Petri Nets.

A *(P/T) net structure* is a tuple $\mathcal{N} = (\mathsf{P}, \mathsf{T}, \mathsf{F})$ with the disjoint sets of *places* P and *transitions* T, and a *flow function* $\mathsf{F} : (\mathsf{P} \times \mathsf{T}) \cup (\mathsf{T} \times \mathsf{P}) \to \mathbb{N}$. Places can hold so-called *tokens*: a *marking* in $\mathcal{N}$ is a multi-set $\mathsf{M} : \mathsf{P} \to \mathbb{N}$, that formalizes a state of $\mathcal{N}$ by indicating the number of tokens on each place. A *(P/T) Petri net* is a tuple $\mathsf{N} = (\mathcal{N}, \mathsf{M}_0)$ of a net structure equipped with an *initial marking* M_0. A net structure (resp. Petri net) is called *finite* if P and T are finite.

$\mathsf{F}(\mathsf{x}, \mathsf{y}) = n > 0$ means there is an *arc* of *weight* n from node x to y describing the flow of tokens in the net. We write $\mathsf{x} \to \mathsf{y}$ iff $\mathsf{F}(\mathsf{x}, \mathsf{y}) > 0$, and denote by $\leq$ and $<$ the reflexive and irreflexive closure of $\to$. For each transition $\mathsf{t} \in \mathsf{T}$ we define the *preset* and *postset* of t as the multi-sets over P given by $\mathit{pre}(\mathsf{t}) := \{\!\!\{ \mathsf{p} \mid (\mathsf{p}, \mathsf{t}) \in \mathsf{F} \}\!\!\}$, and analogously $\mathit{post}(\mathsf{t}) := \{\!\!\{ \mathsf{p} \mid (\mathsf{t}, \mathsf{p}) \in \mathsf{F} \}\!\!\}$. We assume that no transition has an empty preset. A transition $\mathsf{t} \in \mathsf{T}$ is *enabled* in a marking M iff $\mathit{pre}(\mathsf{t}) \leq \mathsf{M}$. If t is enabled then t can *fire* in M, leading to a new marking $\mathsf{M}' = (\mathsf{M} - \mathit{pre}(\mathsf{t})) + \mathit{post}(\mathsf{t})$. This is denoted by $\mathsf{M}[\mathsf{t}\rangle\mathsf{M}'$. By that, firing a transition means consuming a certain number of tokens (given by the arc weight) from the place on the other end of every incoming arc, and placing a certain number of tokens on the place at the other end of every outgoing arc. Analogously to transitions, we define for every place $\mathsf{p} \in \mathsf{P}$ its preset $\mathit{pre}(\mathsf{p}) := \{\!\!\{ \mathsf{t} \mid (\mathsf{t}, \mathsf{p}) \in \mathsf{F} \}\!\!\}$ and postset $\mathit{post}(\mathsf{p}) := \{\!\!\{ \mathsf{t} \mid (\mathsf{p}, \mathsf{t}) \in \mathsf{F} \}\!\!\}$ as mult-sets over T.

For a marking M, we denote by $\mathcal{R}(\mathscr{N}, \mathsf{M})$ the set of all markings in $\mathscr{N}$ that can be reached from M by firing a sequence of transitions. A Petri net $\mathsf{N} = (\mathscr{N}, \mathsf{M}_0)$ is called *k-bounded* if $\forall \mathsf{M} \in \mathcal{R}(\mathsf{N})\, \forall \mathsf{p} \in \mathsf{P} : \mathsf{M}(\mathsf{p}) \leq k$. 1-bounded Petri nets are also called *safe*.

A net structure $\mathscr{N} = (\mathsf{P}, \mathsf{T}, \mathsf{F})$ is called *ordinary* iff all arcs between nodes have a weight of at most 1, i.e., $\forall (\mathsf{x}, \mathsf{y}) \in (\mathsf{P} \times \mathsf{T}) \cup (\mathsf{T} \times \mathsf{P}) : \mathsf{F}(\mathsf{x}, \mathsf{y}) \leq 1$. In an ordinary net structure, two nodes x and y are said to be *in conflict*, denoted by $\mathsf{x} \sharp \mathsf{y}$, iff $\exists \mathsf{p} \in \mathsf{P}\, \exists \mathsf{t}_1, \mathsf{t}_2 \in \mathsf{T} : \mathsf{t}_1 \neq \mathsf{t}_2 \wedge \mathsf{p} \in \mathit{pre}(\mathsf{t}_1) \wedge \mathsf{p} \in \mathit{pre}(\mathsf{t}_2) \wedge \mathsf{t}_1 \leq \mathsf{x} \wedge \mathsf{t}_2 \leq \mathsf{y}$. Additionally, we call x and y *causally related* iff $\mathsf{x} \leq \mathsf{y}$ or $\mathsf{y} \leq \mathsf{x}$. Two nodes are called *concurrent*, iff they are neither in conflict, nor causally related. A set $\mathsf{X} \subseteq \mathsf{P} \cup \mathsf{T}$ is called *concurrent*, iff each two elements in X are concurrent. A concurrent set $\mathsf{X} \subseteq \mathsf{P}$ of places is called a *co-set*.

An essential tool in this thesis is the concept of the *unfolding* of a given Petri net. The unfolding can be seen as a process of "unwinding" the behavior of the original net. The foundation of an unfolding lies in a structurally restricted Petri net, referred to as an *occurrence net*. In this context, we use the term *conditions*, denoted by the symbol B (from the German 'Bedingungen'), instead of places. Similarly, we refer to *events*, denoted by E, instead of transitions. The role of the flow function is now played by the object denoted by H, and the initial marking is represented as K_0, which is now called the *initial cut*. The markings reachable from K_0 are referred to as *cuts*.

A Petri net $\mathsf{O} = (\mathsf{B}, \mathsf{E}, \mathsf{H}, \mathsf{K}_0)$ with an ordinary net structure $(\mathsf{B}, \mathsf{E}, \mathsf{H})$ is called an *occurrence net* if:

i) No event is in self conflict, i.e., $\forall \mathsf{e} \in \mathsf{E} : \neg(\mathsf{e} \sharp \mathsf{e})$.

ii) No node is its own causal predecessor, i.e., $\forall \mathsf{x} \in \mathsf{B} \cup \mathsf{E} : \neg(\mathsf{x} < \mathsf{x})$.

iii) The relation $<$ is well-founded, i.e., $\forall \mathsf{e} \in \mathsf{E} : |\{\mathsf{x} \mid \mathsf{x} < \mathsf{e}\}| < \infty$.

iv) For every $\mathsf{b} \in \mathsf{B}$, exactly one of the following holds:

   a) $\mathsf{K}_0(\mathsf{b}) = 1$ and $\mathit{pre}(\mathsf{b}) = \emptyset$.

   b) $\mathsf{K}_0(\mathsf{b}) = 0$ and there exists a unique event e s.t. $\mathit{pre}(\mathsf{b}) = \{\mathsf{e}\}$.

An occurrence net is called a *causal net* if additionally from every condition there is at most one outgoing event, i.e., $\forall \mathsf{b} \in \mathsf{B} : |\mathit{post}(\mathsf{b})| \leq 1$.

A *configuration* of an occurrence net is a causally closed set C of events that pairwise are not in conflict. Events in such a set describe a (possibly infinite) concurrent execution of the net. For a finite configuration C, the *cut* $\mathrm{cut}(\mathsf{C}) := (\mathsf{K}_0 \setminus ({\to}\mathsf{C})) \cup (\mathsf{C}{\to})$ of C describes the marking reached in the occurrence net after firing the events in C. Conversely, every cut K in an occurrence net can be identified with the finite configuration $\{\mathsf{e} \in \mathsf{E} \mid \exists \mathsf{b} \in \mathsf{K} : \mathsf{e} < \mathsf{b}\}$ containing the events that lie "before" K in the occurrence net. For every event e, we define its *cone configuration* $[\mathsf{e}] = \{\mathsf{e}' \in \mathsf{E} \mid \mathsf{e}' \leq \mathsf{e}\}$.

Let $\mathscr{N} = (\mathsf{P}, \mathsf{T}, \mathsf{F})$ and $\mathscr{N}' = (\mathsf{P}', \mathsf{T}', \mathsf{F}')$ be two net structures. A function $\mathsf{h} : \mathsf{P} \cup \mathsf{T} \to \mathsf{P}' \cup \mathsf{T}'$ is called a *(Petri net) homomorphism* from $\mathscr{N}$ to $\mathscr{N}'$, iff

i) it maps places and transitions in $\mathscr{N}$ into the corresponding sets in $\mathscr{N}'$, i.e., $\forall \mathsf{p} \in \mathsf{P} : \mathsf{h}(\mathsf{p}) \in \mathsf{P}' \wedge \forall \mathsf{t} \in \mathsf{T} : \mathsf{h}(\mathsf{t}) \in \mathsf{T}'$;

ii) it maps the pre- and postset of transitions correspondingly, i.e., $\forall \mathsf{t} \in \mathsf{T} : \mathit{pre}'(\mathsf{h}(\mathsf{t})) = \mathsf{h}(\mathit{pre}(\mathsf{t})) \wedge \mathit{post}'(\mathsf{h}(\mathsf{t})) = \mathsf{h}(\mathit{post}(\mathsf{t}))$.

For two Petri nets $\mathsf{N} = (\mathscr{N}, \mathsf{M}_0)$ and $\mathsf{N}' = (\mathscr{N}', \mathsf{M}'_0)$ the homomorphisms from N to N' are the homomorphisms from $\mathscr{N}$ to $\mathscr{N}'$. Such a homomorphism h is called *initial* iff

iii) it maps the initial marking of N to the initial marking of N', i.e., $\mathsf{h}(\mathsf{M}_0) = \mathsf{M}'_0$.

A(n *initial*) *branching process* $\beta = (\mathsf{O}, \mathsf{h})$ of a Petri net $\mathsf{N} = (\mathsf{P}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0)$ consists of an occurrence net $\mathsf{O} = (\mathsf{B}, \mathsf{E}, \mathsf{H}, \mathsf{K}_0)$ and a(n initial) homomorphism $\mathsf{h} : \mathsf{B} \cup \mathsf{E} \to \mathsf{P} \cup \mathsf{T}$ that is injective on events with same preset, i.e., $\forall \mathsf{e}_1, \mathsf{e}_2 \in \mathsf{E} : (\mathit{pre}(\mathsf{e}_1) = \mathit{pre}(\mathsf{e}_2) \wedge \mathsf{h}(\mathsf{e}_1) = \mathsf{h}(\mathsf{e}_2)) \Rightarrow \mathsf{e}_1 = \mathsf{e}_2$. If $\rho = (\mathsf{O}, \mathsf{h})$ is an initial branching process of N with a causal net O, then ρ is called a *(concurrent) run of* N. A run formalizes a single concurrent execution of the net. For two symbolic branching processes $\beta = (\mathsf{O}, \mathsf{h})$ and $\beta' = (\mathsf{O}', \mathsf{h}')$ of a Petri net, β is a *prefix* of β' if there exists an injective initial homomorphism ϕ from O into O', such that $\mathsf{h}' \circ \phi = \mathsf{h}$. In the case of $\phi = \mathit{id}$, i.e., O is a *subnet* of O', we call β a *subprocess* of β'.

In Theorem 23 of [Eng91] it is shown that for any given Petri net $\mathsf{N} = (\mathsf{P}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0)$ there exists a unique maximal branching process (maximal w.r.t. the prefix relation and unique up to isomorphism). This branching process is called the *unfolding of* N, and denoted by $\Upsilon(\mathsf{N}) = (\mathsf{B}^\mathsf{N}, \mathsf{E}^\mathsf{N}, \mathsf{H}^\mathsf{N}, \mathsf{K}^\mathsf{N}_0, \pi^\mathsf{N})$, where the contained occurrence net is called $\mathsf{U}(\mathsf{N})$, such that $\Upsilon(\mathsf{N}) = (\mathsf{U}(\mathsf{N}), \pi^\mathsf{N})$. The values of the homomorphism π^N are also called *labels* of the conditions/events.

This unfolding has the property that for every transition that can occur in the net, there is a transition in the unfolding with corresponding label, i.e., $\forall \mathsf{t} \in \mathsf{T} \forall \mathsf{X} \subseteq \mathsf{B}^\mathsf{N} : \mathsf{X}$ co-set $\wedge\, \mathit{pre}(\mathsf{t}) = \pi^\mathsf{N}(\mathsf{X}) \Rightarrow \exists \mathsf{e} \in \mathsf{E}^\mathsf{N} : \pi^\mathsf{N}(\mathsf{e}) = \mathsf{t} \wedge \mathit{pre}(\mathsf{e}) = \mathsf{X}$.

2.2 Place/Transition Petri Games

We extend a Petri net to a game by interpreting the tokens, present in the places of the Petri net, as players of the game. To the aim of having different teams of players, we divide the places into distinct categories. Specifically, we categorize the places as – controllable – system places and – uncontrollable – environment places, thereby forming two teams. The team-membership of a player is determined by the category of the place on which the player resides.

In Petri nets, firing a transition means consuming a certain number of tokens (given by the arc weight) from the place on the other end of every incoming arc, and placing a certain number of tokens on the place at the other end of every outgoing arc. Recognizing the correspondence between tokens and players, this implies that players can be "created" or "disposed" by transitions. In the context of Petri games, especially when the number of consumed system and environment players is equal to the corresponding number of placed players, we often identify the consumed and placed players with each other while describing how a Petri game models a scenario. By this notion, players can also "switch teams."

A play of the game is a concurrent execution of transitions in the net. During a play, the *knowledge* of each player is represented by its *causal history*, i.e., all visited places and used transitions to reach the current place. Players enrich this local knowledge when participating in joint transitions: at such an occasion, the complete knowledge of all participating players is exchanged. In the terminology of [Sch03], we have a game with "perfect recall" and "imperfect information".

Based on their knowledge, players allow or forbid transitions in their postset. A transition can only fire if every player in its preset allows the execution. The system players in a Petri game win a play if they satisfy a given objective. In the context of this thesis, this is either a safety objective, given by a designated set of bad places the system players must not reach, or a reachability objective, given by a set of target places one of them has to reach.

Formally, a *(P/T) Petri game* is a tuple $\mathsf{G} = (\mathsf{P}_\mathrm{S}, \mathsf{P}_\mathrm{E}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Obj}, \mathsf{P}_{\circledast})$, with a set of *system places* P_S, a set of *environment places* P_E, and a set of *special places* $\mathsf{P}_{\circledast} \subseteq \mathsf{P}_\mathrm{S}$. The set of all places is denoted by $\mathsf{P} = \mathsf{P}_\mathrm{S} \sqcup \mathsf{P}_\mathrm{E}$, and $\mathsf{T}, \mathsf{F}, \mathsf{M}_0$ are the remaining components of a Petri net $\mathsf{N}(\mathsf{G}) = (\mathsf{P}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0)$, called the *underlying net* of G. The *objective type* Obj belongs to a two-element set $\mathrm{Obj} \in \{\mathrm{Safety}, \mathrm{Reach}\}$, which, in combination with the set of special places $\mathsf{P}_{\circledast}$, defines the *objective* for the system players:

In the case Obj = Safety, the aim of the system players is for all of them to avoid reaching any special place $\mathsf{p} \in \mathsf{P}_{\circledast}$. The special places are in this case called *bad places*, and denoted by $\mathsf{P}_{\spadesuit}$.

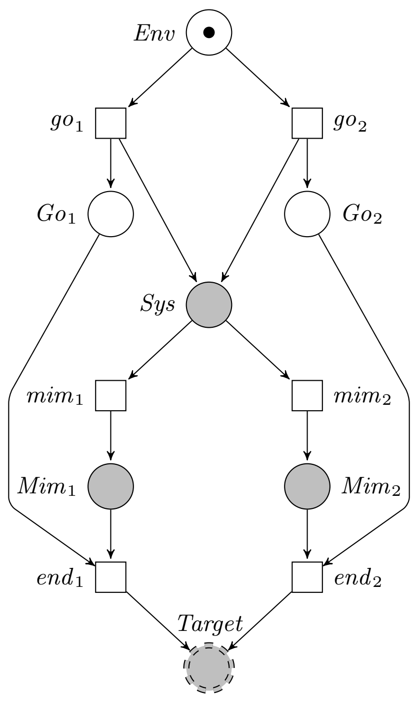
In contrast, if Obj = Reach, the aim of the system players is for at least one of them to reach a special place $\mathsf{p} \in \mathsf{P}_{\circledast}$. The special places are in this case called *target places*, and denoted by $\mathsf{P}_{\heartsuit}$.

In order to achieve this *safety objective* resp. *reachability objective*, the players often must cooperate: every player can solely use their locally available information which is their own *causal past*, i.e., the places and transitions which have been used to reach the current place. Cooperation among the players means that information is exchanged with all players participating at a joint transition.

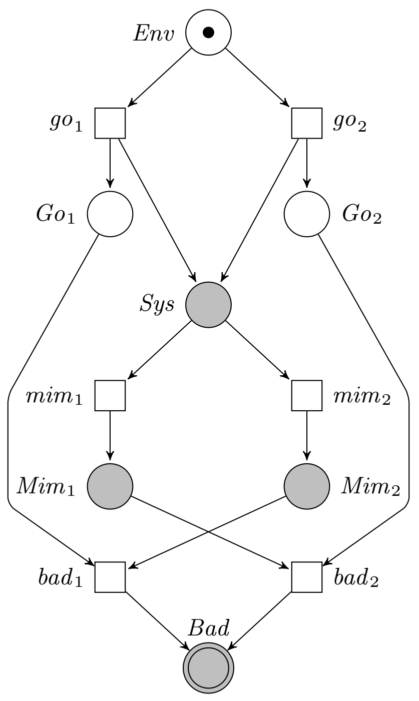
We consider Petri games with a finite underlying net. We call transitions with a preset only consisting of system places *system transitions* and all other transitions *environment transitions*. Transitions with a preset only consisting of environment places are called *pure environment transitions*. Note that only pure environment transitions are under control of the environment player, whereas transitions which contain at least one system place in its preset are under control of the system players.

When visualizing a Petri game, we extend the traditional way of visualizing Petri nets: we depict the system places P_S as gray circles and the environment places P_E as white circles. For special places $\mathsf{P}_{\circledast}$, we draw a double border – solid for bad places $\mathsf{P}_{\spadesuit}$ in the case Cond = Safety, and dashed for target places $\mathsf{P}_{\heartsuit}$ in the case Cond = Reach. Transitions are depicted as boxes, and the flow function is illustrated by arcs between transitions and places, with their weight as label. Arcs with weight 0 are omitted completely, whereas we omit the label on arcs with weight 1.

The following example gives a first impression about the semantics of a Petri game.



(a) Petri game with reachability objective.



(b) Petri game with safety objective.

Figure 2.1: First examples of Petri games..

Example 2.1 (P/T Petri game). In Fig. 2.1, two Petri games are shown: one with a reachability objective (Fig. 2.1a) and one with a safety objective (Fig. 2.1b) for the system players.

Initially, in the Petri game from Fig. 2.1a, there is only one environment player on place *Env*. This means, only the transitions go_1 and go_2 are activated. Fix an $\hat{\imath} \in \{1, 2\}$ and say the environment fires $go_{\hat{\imath}}$. Then the environment player moves from place *Env* to place $Go_{\hat{\imath}}$. Additionally, a token gets placed on the system place *Sys*, i.e., a system player is generated. This system player now has to make an analogous choice, i.e., taking transition mim_1 or mim_2 to go to place Mim_1 or Mim_2, respectively. The goal of the system player is to *mimic* the choice of the environment player, since only if it takes the transition $mim_{\hat{\imath}}$, transition $end_{\hat{\imath}}$ is activated from marking $\{\!|\, Go_{\hat{\imath}}, Mim_{\hat{\imath}} \,|\!\}$. Firing it leads to the marking $\{\!|\, Target \,|\!\}$, i.e., the system player reached the target space and therefore achieved the reachability objective. The system player can only mimic the environment's choice because the latter is in their *causal history* when they are generated by go_1 or go_2. They can therefore use that *local knowledge* to mimic the choice correctly.

The Petri game from Fig. 2.1b works analogously but with the difference that if the system player makes the wrong choice and fires transition $mim_{\hat{\jmath}}$ with $\hat{\jmath} \neq \hat{\imath}$, then it afterwards has no choice other than taking transition $bad_{\hat{\imath}}$, reaching the bad place and failing the objective. Only with the correct mimic, the net gets into a terminating marking and no bad place can be reached anymore. ◁

The causal dependencies in a Petri game G (and thus, the knowledge of the players) are represented in its *unfolding*, which is the unfolding $\Upsilon(\mathsf{G}) := \Upsilon(\mathsf{N}(\mathsf{G}))$ of the underlying net $\mathsf{N}(\mathsf{G})$. The *system-, environment-, and special conditions* are marked correspondingly: Let B be the conditions in $\Upsilon(\mathsf{G})$. Then $\mathsf{B}_\mathrm{S} := \{\mathsf{b} \in \mathsf{B} \mid \pi(\mathsf{b}) \in \mathsf{P}_\mathrm{S}\}$, $\mathsf{B}_\mathrm{E} := \{\mathsf{b} \in \mathsf{B} \mid \pi(\mathsf{b}) \in \mathsf{P}_\mathrm{E}\}$, and $\mathsf{B}_\circledast := \{\mathsf{b} \in \mathsf{B} \mid \pi(\mathsf{b}) \in \mathsf{P}_\circledast\}$. A *play* in G is an initial concurrent run $\rho = (\mathsf{O}, \mathsf{h})$ of the underlying net N of a Petri game G (i.e., all nondeterminism has been resolved) that is maximal with respect to the prefix relation.

A *strategy* for the system players in G describes a local controller for each system player whose operation is based on its currently available information about the whole system. Such a strategy is obtained from an unfolding by deleting some of the branches that are under control of the system players. Thus, it is technically a subprocess of the unfolding. Every condition describes a place in the net with a certain causal history. A strategy describes for each condition which transitions are available to a player in the corresponding place. This set of transitions depends on the players knowledge, i.e., the causal history of the corresponding condition, and is represented by the events in the strategy having this condition in their preset.

A *strategy* for the system players in G is a subprocess $\xi = (\mathsf{O}^\xi, \pi^\xi)$ with occurrence net $\mathsf{O}^\xi = (\mathsf{B}^\xi, \mathsf{E}^\xi, \mathsf{H}^\xi, \mathsf{K}_0^\xi)$ of the unfolding $\Upsilon = (\mathsf{B}, \mathsf{E}, \mathsf{H}, \mathsf{K}_0, \pi)$ of G satisfying the following conditions:

a) *Justified Refusal:* if an event is not in the strategy, then the reason is that a system player in its preset forbids *all* occurrences of this transition in the strategy. So to be more specific, there is no restriction on pure environment decisions, and system players can allow or forbid only transitions of the original net, based on only their knowledge:

$$\begin{aligned}\forall \mathsf{e} \in \mathsf{E} :& (\mathit{pre}(\mathsf{e}) \subseteq \mathsf{B}^\xi \wedge \mathsf{e} \notin \mathsf{E}^\xi)\\ &\Rightarrow (\exists \mathsf{b} \in \mathit{pre}(\mathsf{e}) \cap \mathsf{B}_\mathrm{S}\ \forall \mathsf{e}' \in \mathit{post}(\mathsf{b}) : \pi(\mathsf{e}') = \pi(\mathsf{e}) \Rightarrow \mathsf{e}' \notin \mathsf{E}^\xi).\end{aligned}$$

b) The system players must act in a *deterministic* way, i.e., in no reachable marking of the strategy two transitions involving the same system player are enabled:

$$\forall \mathsf{K} \in \mathcal{R}(\mathsf{O}^\xi)\ \forall \mathsf{b} \in \mathsf{K} \cap \mathsf{B}_\mathrm{S}\ \exists^{\leq 1} \mathsf{e} \in \mathsf{E}^\xi : \mathsf{b} \in \mathit{pre}(\mathsf{e}) \wedge \mathsf{K}[\mathsf{e}\rangle.$$

A strategy is *deadlock-free*, if it allows the possibility to continue whenever the system can proceed, i.e., $\forall \mathsf{K} \in \mathcal{R}(\mathsf{O}^\xi) : (\exists \mathsf{e} \in \mathsf{E} : \mathsf{K}[\mathsf{e}\rangle) \Rightarrow \exists \mathsf{e}' \in \mathsf{E}^\xi : \mathsf{K}[\mathsf{e}'\rangle$.

Figure 2.2 visualizes *violations* related of to the two strategy conditions and of the requirement for a strategy to be deadlock-free. We schematically depict occurrence nets O^a, O^b, O^c from prefixes (O^a, π^a), (O^b, π^b), (O^c, π^c) of an unfolding (U, π). When needed for illustrating the violation, values of π are written next to the conditions and events. In the prefix (O^a, π^a) we see two instances of each of the places p and q. From every combination of one instance of p and one of q, an instance of the transition t is fireable. Consider the most left instance of t. Since it is not part of the alleged strategy, there must a system place in its preset that refuses *all* instances of t. However, both places in

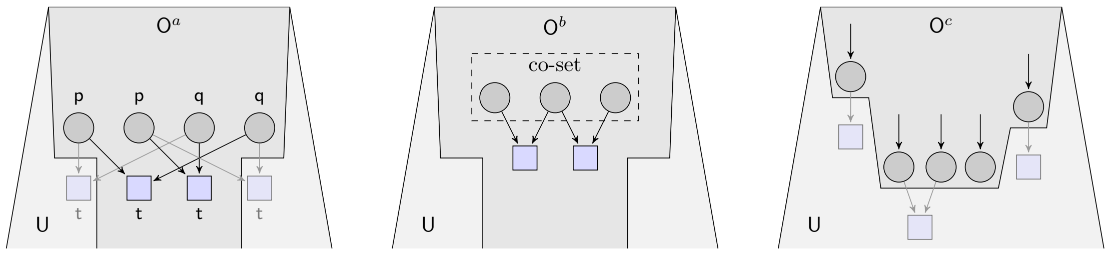


Figure 2.2: Visualization of three prefixes (O^a, π^a), (O^b, π^b), (and (O^c, π^c)) of an unfolding (U, π), each *violating* one of the conditions satisfied by a (deadlock-free) strategy.

its preset allow another instance of t, meaning that both only allow certain instances of t, depending on knowledge they do not have. Thus, it violates justified refusal condition a). The prefix (O^b, π^b) contains two concurrent transitions sharing a system place in their preset, violating the determinism condition b). Finally, the occurrence net O^c is finite, cutting off all possible continuations in the unfolding. This results in a deadlock which means that the prefix (O^c, π^c) is not deadlock-free.

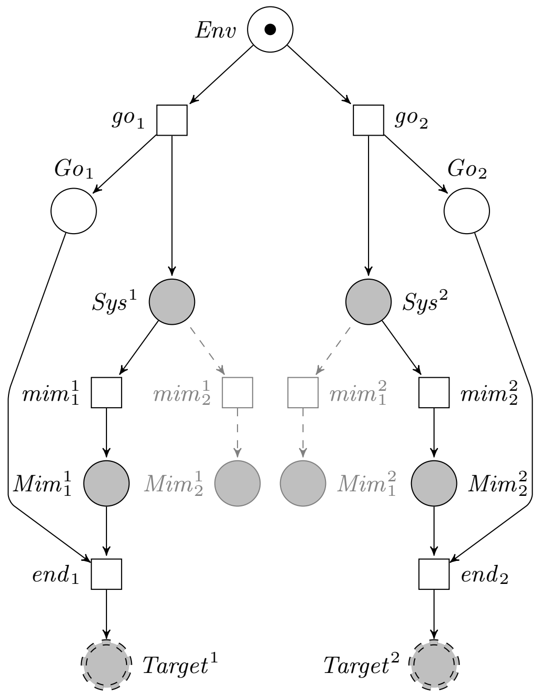
A play ρ *conforms* to a strategy ξ if ρ it is a prefix of ξ. The system players *win* ρ with conditions B^ρ depending on Cond:

- If Cond = Safety then the system players win ρ if it contains no instance of a bad place, i.e., $\mathsf{B}^\rho_\spadesuit = \emptyset$.
- If Cond = Reach then the system players win ρ if it contains an instance of a target place, i.e., $\mathsf{B}^\rho_\heartsuit \neq \emptyset$.

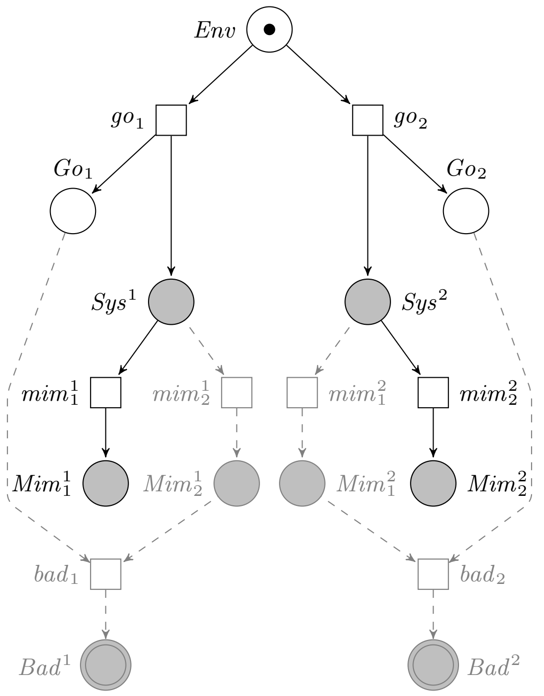
In the case of Cond = Safety, a strategy ξ is called *winning* if it is deadlock-free and all plays that conform to ξ are won by the system players. The latter is equivalent to $\mathsf{B}^\xi_\spadesuit = \emptyset$. Since we consider a safety objective, the system players would win with a non deadlock-free strategy by just doing nothing. In the case of Cond = Reach, we drop this assumption of deadlock-freedom and call a strategy ξ *winning* if all plays that conform to ξ are won by the system players.

Example 2.2 (Strategy in a P/T Petri Game). Fig. 2.3 shows the already informally described winning strategy (solid elements) for the Petri game presented in Fig. 2.1 as a prefix of the Petri games unfoldings (solid + dashed/grayed out elements). The names of conditions and events correspond to their labels (the homomorphism's values, i.e., places and transitions in the original net). If there is more than one instance of a place or transition in the unfolding, superscripts are added to the names of the respective conditions/events.

Consider the winning strategy in Fig. 2.3a. A strategy must not restrict any pure environment transition. Consequently, both events go_1 and go_2 are in the strategy. However, compared to the original net, they do not place a token on the same place *Sys*, but there is an output condition Sys^i for each of them. This corresponds to the knowledge of the system player placed on *Sys* in the original net: it can now make its choice which



(a) Winning strategy for the system players with reachability objective in the Petri game from Fig. 2.1a.



(b) Winning strategy for the system players with safety objective in the Petri game from Fig. 2.1b.

Figure 2.3: Winning strategies (solid) for the system players in the Petri games from Fig. 2.1, represented as prefixes in the respective unfolding (solid and dashed/grayed out).

events to allow depending on the causal history, i.e., for $i = 1, 2$ the system player on Sys^i only allows event mim^i_i and cuts off event mim^i_j with $j \neq i$ from the strategy. Therefore, every play reaches either the cut $\{\!|\, Mim^1_1, Go_1 \,|\!\}$ or the cut $\{\!|\, Mim^2_2, Go_2 \,|\!\}$, from where the events end_i, $i = 1, 2$ ensure that either the condition $Target^1$ or the condition $Target^2$ is in every play of the game conforming to the strategy, making it a *winning* strategy.

The winning strategy in Fig. 2.3b for the Petri game with safety objective for the system players again is built analogously. Just as the strategy in (a) it ensures every play to finally reach either the cut $\{\!|\, Mim^1_1, Go_1 \,|\!\}$ or the cut $\{\!|\, Mim^2_2, Go_2 \,|\!\}$, from which no event can be fired, thus terminating the net's execution and achieving the safety objective for the system players since they can never reach a bad place. ◁

Running Example: Communicating Copycats. We now extend the example of a Petri game with reachability objective for the system players from Fig. 2.1a and provide it with a twist to build our first running example, called "Communicating Copycats".

The result is shown in Fig. 2.4. At first glance, the upper part seems like two copies of the game from Fig. 2.1a: we initially have two environment players on places Env_{B}

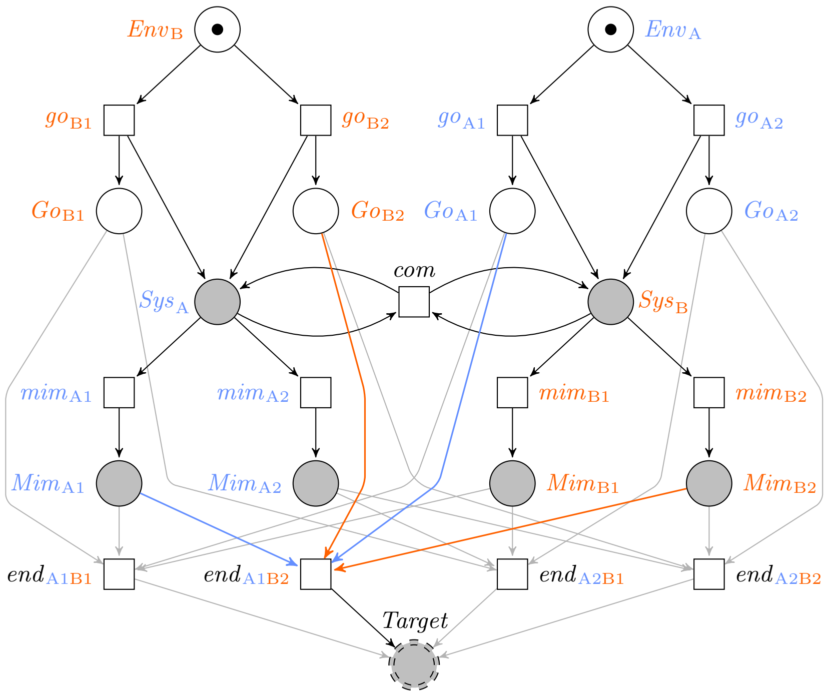


Figure 2.4: Petri game "Communicating Copycats" with communication between the system players needed in a winning strategy.

and Env_A, who both have the choice of taking transition go_{B1} or go_{B2} (resp. go_{A1} or go_{A2}). Firing such a transition creates a system player that can again mimic what the environment did. However, the twist is that now the system player on place Sys_A (on the left), generated by the firing of go_{B1} or go_{B2}, has to mimic the choice made by the environment player initially residing on place Env_A (on the right), and the other way around for the system player placed on the place Sys_B. However, directly after the system player gets placed on Sys_A, the only information it has is which one of the two transitions go_{B1} or go_{B2} was fired.

To get the information about which of the two transitions go_{A1} or go_{A2} was fired, the system player on place Sys_A must "communicate" with the system player on place Sys_B. This can be achieved by taking the joint transition *com*, which does not change the marking. However, by firing *com*, the whole causal history of both participating players gets shared between them. Thus, after one firing of this transition, the two system players can correctly mimic the respective environment player's choice. Finally, one of the four *end* transitions can be taken. Say the environment player on Env_A takes transition $go_{A\hat{\imath}}$ and the environment player on Env_B takes transition $go_{B\hat{\jmath}}$. Then the transition $end_{A\hat{\imath}B\hat{\jmath}}$ leading to the target place *Target* gets enabled if and only if the system player on Sys_A mimics the environment player from Env_A by taking $mim_{A\hat{\imath}}$ and the system player on Sys_B mimics the environment player from Env_B by taking $mim_{B\hat{\jmath}}$. Transition $end_{A\hat{\imath}B\hat{\jmath}}$ can then fire from the marking $\{\!\!\{\, Go_{A\hat{\imath}}, Go_{B\hat{\jmath}}, Mim_{A\hat{\imath}}, Mim_{B\hat{\jmath}} \,\}\!\!\}$.

To improve readability, the arcs to and from all *end* transitions except $\mathit{end}_{\mathrm{A1B2}}$ are grayed out. For $\mathit{end}_{\mathrm{A1B2}}$, the colors of the arcs emphasize that the transition is enabled when the system player on $\mathit{Sys}_{\mathrm{A}}$ mimics the choice of the environment player on $\mathit{Env}_{\mathrm{A}}$ to take transition $\mathit{go}_{\mathrm{A1}}$ by taking transition $\mathit{mim}_{\mathrm{A1}}$, and the system player on $\mathit{Sys}_{\mathrm{B}}$ mimics the choice of the environment player on $\mathit{Env}_{\mathrm{B}}$ to take transition $\mathit{go}_{\mathrm{B2}}$ by taking $\mathit{mim}_{\mathrm{B2}}$.

A winning strategy for the example Communicating Copycats is shown in Fig 2.5, again as a prefix of the whole unfolding. The parts of the unfolding not belonging to the strategy are dashed/grayed out. Again, the labels of the conditions/events are given by their names, which are the names of places/transitions in the original net, and with a superscript added when there is more than one occurrence of a node. To improve readability, each condition $\mathit{Go}_{\mathrm{X}i}$ with $\mathrm{X} \in \{\mathrm{A}, \mathrm{B}\}, i \in \{1, 2\}$ is drawn multiple times. All conditions with the same name are identified with each other; for all $\mathrm{X} \in \{\mathrm{A}, \mathrm{B}\}, i \in \{1, 2\}$ there is only one condition $\mathit{Go}_{\mathrm{X}i}$ in the unfolding. Additionally, we abbreviate $\mathit{mim}_{\mathrm{A}i}$ by $\mathrm{a}i$ and $\mathit{Mim}_{\mathrm{A}i}$ by $\mathrm{A}i$, with $i = 1, 2$, and analogously for B. Finally, for the system players, we chose the superscript of the conditions names such that it also indicates a players knowledge: for example, the system player on $\mathit{Sys}^{2}_{\mathrm{A}}$ knows that the environment player from $\mathit{Env}_{\mathrm{B}}$ took transition $\mathit{go}_{\mathrm{B2}}$, while a system player on $\mathit{Sys}^{12}_{\mathrm{A}}$ knows in addition that the environment player from $\mathit{Env}_{\mathrm{A}}$ took transition $\mathit{go}_{\mathrm{A1}}$.

Just as in the simpler example in Fig. 2.3a, the initial pure environment events must all be contained in the strategy. Generated by a $\mathit{go}_{\mathrm{Y}i}$ event, the system player on condition $\mathit{Sys}^{i}_{\mathrm{X}}$ with $\mathrm{X} \neq \mathrm{Y}$ could, in a strategy, allow *one* of the two events $\mathit{mim}^{i}_{\mathrm{X}j}$ (abbreviated by $\mathrm{x}j^{i}$), but not both, since this would violate determinism of the strategy. However, since the system player does not know which one of the two events $\mathit{go}_{\mathrm{X1}}$ or $\mathit{go}_{\mathrm{X2}}$ has fired, and therefore which event in the future of $\mathrm{x}j^{i}$ leading to a target condition would finally be enabled, it cannot make a deterministic choice that guarantees reaching a target place.

To resolve the situation described above, every system player on a condition $\mathit{Sys}^{i}_{\mathrm{X}}$ allows only all instances of *com* to fire. Although this may seem nondeterministic at first sight, this is allowed in a strategy since the nondeterminism is resolved by the environment's choices; no two instances of *com* in the unfolding are enabled in the same cut. After the firing of the event com^{ij}, the system players on $\mathit{Sys}^{ij}_{\mathrm{A}}$ and $\mathit{Sys}^{ij}_{\mathrm{B}}$ can then deterministically allow the correct events $\mathit{mim}^{ij}_{\mathrm{A}i}$ and $\mathit{mim}^{ij}_{\mathrm{B}j}$, whose firing leads to the cut $\Lbag\, \mathit{Mim}^{ij}_{\mathrm{A}i}, \mathit{Mim}^{ij}_{\mathrm{B}j}, \mathit{Go}_{\mathrm{A}i}, \mathit{Go}_{\mathrm{B}j} \,\Rbag$. From there, the event $\mathit{end}''_{\mathrm{A}i\mathrm{B}j}$ can fire, making the respective play winning by reaching the target condition $\mathit{Target}^{ij}{}''$.

Note that the unfolding is infinite and that the future of firing more than one instance of *com* is not shown here. This also means that there are infinitely many winning strategies for the system players, since they could take the transition *com* arbitrarily often before making the correct choice. However, after one firing, they have all the knowledge they need to make that choice.

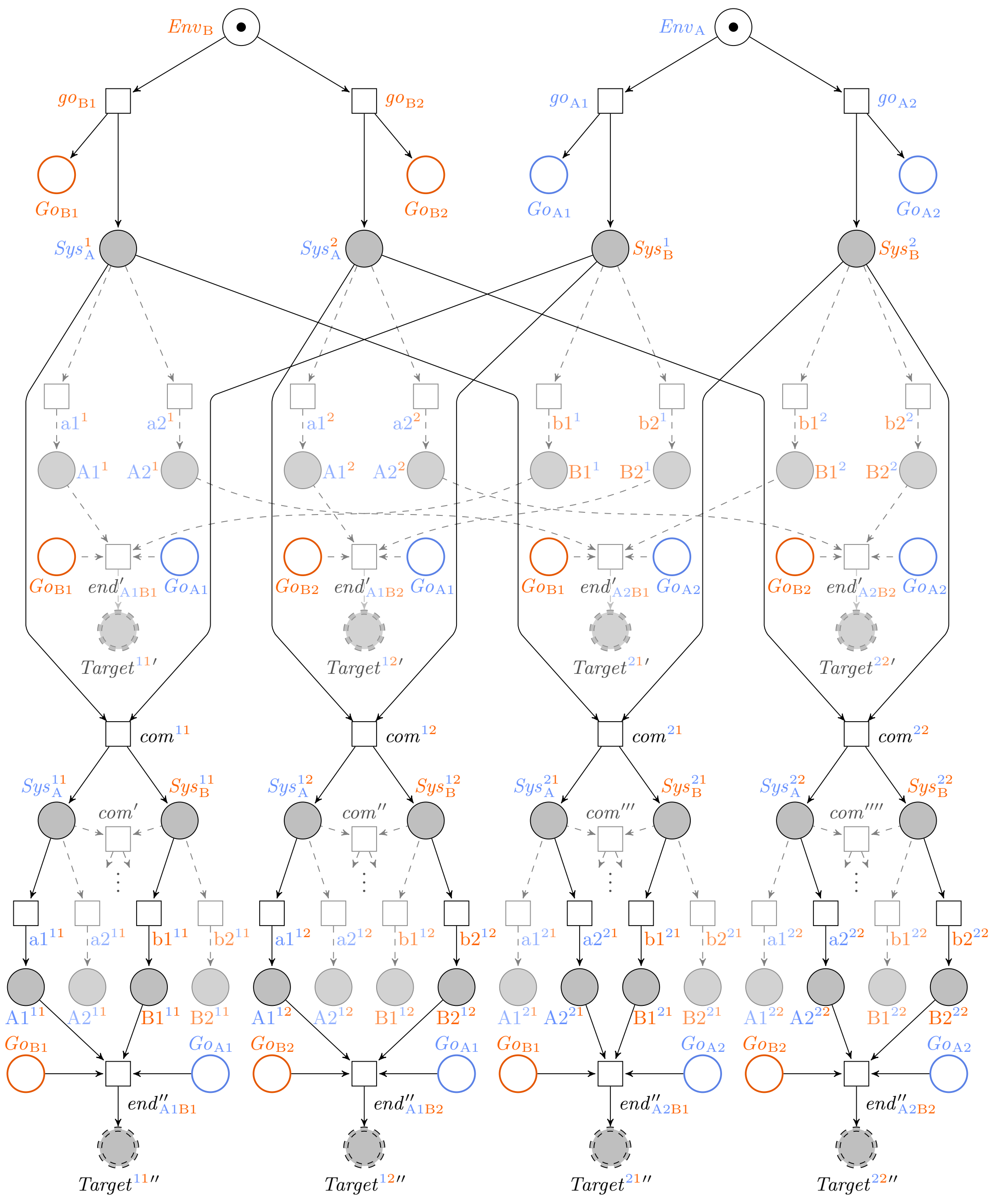


Figure 2.5: A winning strategy (solid) for the system players in running example “communicating copycats” from Fig. 2.4 as a prefix of the Petri games unfolding (solid and grayed out).

2.3 High-Level Petri Nets and High-Level Petri Games

In this subsection, we introduce a generalization of Petri nets and Petri games known as high-level Petri nets and high-level Petri games, respectively. We will begin by presenting high-level Petri nets and recalling their definitions and formalism. This serves as a foundation for subsequently presenting the definition of high-level Petri games as introduced in [GO21].

The generalization of Petri nets also leads to a generalization of the concepts of branching processes, and unfoldings. The generalizations are called symbolic branching and symbolic unfolding, respectively. We conclude this subsection with this generalization. Specifically, we recall its definition from [CJ04], where symbolic unfoldings for high-level Petri nets are introduced.

2.3.1 High-Level Petri Nets

High-level Petri nets generalize Place/Transition Petri nets in the sense that they allow places to hold tokens of different values, called *colors*. Arcs between nodes are labeled with variables, and a mode of a transition assigns colors to all variables "around it", describing which colors are moved from/to which places. Every transition has a *guard* restricting the possible modes. We structure of defining high-level Petri nets is analogous to the P/T Petri nets definition.

A *(high-level) net structure* is a tuple $\mathcal{N} = (\mathit{Col}, \mathit{Var}, P, T, F, g)$ with the following components. Col and Var are the sets of *colors* and *variables*, and P and T are sets of *places* and *transitions* such that the four sets are pairwise disjoint. The *flow function* is given by $F : (P \times \mathit{Var} \times T) \cup (T \times \mathit{Var} \times P) \to \mathbb{N}$. Let $\mathit{Var}(t) = \{v \in \mathit{Var} \mid \exists p \in P : (p, v, t) \in F \vee (t, v, p) \in F\}$. The function g maps each $t \in T$ to a predicate $g(t)$ on $\mathit{Var}(t)$, called the *guard* of t. By this, $g(t)$ can contain other (bounded) variables, but all free variables in $g(t)$ must appear on arcs to or from t. A *marking* in $\mathcal{N}$ is a multi-set $M : P \times \mathit{Col} \to \mathbb{N}$, describing how often each color $c \in \mathit{Col}$ currently resides on each place $p \in P$. A *high-level Petri net* $N = (\mathcal{N}, \mathcal{M}_0)$ is a net structure $\mathcal{N}$ together with a nonempty set $\mathcal{M}_0$ of *initial markings*, where we assume $\forall M_0, M_0' \in \mathcal{M}_0 : \{\!\mid p \mid (p, c) \in M_0 \mid\!\} = \{\!\mid p \mid (p, c) \in M_0' \mid\!\}$, i.e., in all initial markings, the same places are marked with *the same number of colors*. We often assume the two sets Col of colors and Var of variables to be given. In this case, we denote a high-level net structure (resp. high-level Petri net) by $\mathcal{N} = (P, T, F, g)$ (resp. $N = (P, T, F, g, \mathcal{M}_0)$).

For two nodes $x, y \in P \cup T$, we write $x \to y$, iff there exists a variable v such that $(x, v, y) \in F$. The reflexive and irreflexive transitive closures of $\to$ are denoted respectively by $\leq$ and $<$. For a transition $t \in T$, we denote by $\mathit{pre}(t) := \{\!\mid (p, v) \mid (p, v, t) \in F \mid\!\}$ and $\mathit{post}(t) := \{\!\mid (p, v) \mid (t, v, p) \in F \mid\!\}$ the *preset* and *postset* of t. A *mode* of t is a mapping $\sigma : \mathit{Var}(t) \to \mathit{Col}$ such that $g(t)$ evaluates to *true* under the substitution given by σ, denoted by $g(t)[\sigma] \equiv \mathit{true}$. We then denote $\mathit{pre}(t, \sigma) := \{\!\mid (p, \sigma(v)) \mid (p, v) \in \mathit{pre}(t) \mid\!\}$ and $\mathit{post}(t, \sigma) := \{\!\mid (p, \sigma(v)) \mid (p, v) \in \mathit{post}(t) \mid\!\}$. The set of modes of t is denoted by $\Sigma(t)$. Note that such a "binding" of variables to colors is always only local, when firing the respective transition. t can *fire* in a mode σ from a marking M if $M \geq \mathit{pre}(t, \sigma)$, denoted by $M[t,\sigma\rangle$. This firing leads to a new marking $M' = (M - \mathit{pre}(t,\sigma)) + \mathit{post}(t,\sigma)$, which is denoted by $M[t,\sigma\rangle M'$.

We collect in the set $\mathcal{R}(\mathcal{N},\mathcal{M})$ the markings reachable by firing a sequence of transitions in $\mathcal{N}$ from any marking in a set of markings $\mathcal{M}$. We say $\mathcal{N}$ resp. N is *finite* if P, T and F are finite. A high-level Petri net $N = (\mathit{Col}, \mathit{Var}, P, T, F, g, \mathcal{M}_0)$ is called *safe* if in every reachable marking, on every place there is at most one color, formally $\forall M \in \mathcal{R}(N) \forall p \in P : \sum_{c \in \mathit{Col}} M(p,c) \leq 1$. In this thesis, we in particular aim to analyze the behavior of high-level Petri nets. To avoid any issues concerning undecidability regarding the firing relation, we assume that guards are expressed in a decidable logic, with Col as its domain of discourse.

Let $\mathcal{N} = (P,T,F,g)$ and $\mathcal{N}' = (P',T',F',g')$ be two net structures with the same sets of colors and variables. A function $h : P \cup T \to P' \cup T'$ is called a *(high-level Petri net) homomorphism*, if:

i) it maps places and transitions in $\mathcal{N}$ into the corresponding sets in $\mathcal{N}'$, i.e., $h(P) \subseteq P'$ and $h(T) \subseteq T'$;

ii) it is "compatible" with the guard, preset, and postset, of transitions, i.e., for all $t \in T$ we have $g(t) = g'(h(t))$ and $\mathit{pre}(h(t)) = \{\!\!\{ (h(p),v) \mid (p,v) \in \mathit{pre}(t) \}\!\!\}$ and $\mathit{post}(h(t)) = \{\!\!\{ (h(p),v) \mid (p,v) \in \mathit{post}(t) \}\!\!\}$.

For $N = (\mathcal{N},\mathcal{M}_0)$ and $N' = (\mathcal{N}',\mathcal{M}'_0)$, the homomorphisms between N and N' are the homomorphisms between $\mathcal{N}$ and $\mathcal{N}'$. Such a homomorphism h is called *initial* if additionally

iii) $\{\{\!\!\{ (h(p),c) \mid (p,c) \in M_0 \}\!\!\} \mid M_0 \in \mathcal{M}_0\} = \mathcal{M}'_0$.

High-level Petri nets generalize Place/Transition Petri nets: Every P/T Petri net $(\mathsf{P},\mathsf{T},\mathsf{F},\mathsf{M}_0)$ can be interpreted as a high-level Petri net $(\{\bullet\},\{v^\bullet\},\mathsf{P},\mathsf{T},\mathsf{F}^\bullet,\mathit{true},\{\mathsf{M}_0^\bullet\})$ with $\mathsf{F}^\bullet(\mathsf{x},v^\bullet,\mathsf{y}) := \mathsf{F}(\mathsf{x},\mathsf{y})$. In a marking, every place holds a number of tokens $\bullet$, which is the only value ever assigned to the variable $v^\bullet$ on every arc (since $\bullet$ is the only color). The initial marking is given by $\mathsf{M}_0^\bullet(\mathsf{p},\bullet) := \mathsf{M}_0(\mathsf{p})$ for all $\mathsf{p} \in \mathsf{P}$, and the guard of every transition is *true*.

Conversely, we can *expand* a given high-level Petri net to a P/T Petri net with the same behavior. Often (and particularly in all examples in this thesis) we consider high-level Petri nets with only one initial marking, i.e., $\mathcal{M}_0$ is a singleton. Such a high-level Petri net is then also denoted by $N = (\mathit{Col}, \mathit{Var}, P, T, F, g, M_0)$ with the initial marking M_0. We now first define the expansion of high-level Petri nets with one initial marking, and explain how to adapt this definition for multiple initial markings at the end of this section. The *expansion*[1] of N is defined as the P/T Petri net $\mathrm{Exp}(N) = (\mathsf{P},\mathsf{T},\mathsf{F},\mathsf{M}_0)$ with

- $\mathsf{P} := \{p.c \mid p \in P, c \in \mathit{Col}\}$,

[1]What we call here "expansion" is often referred to as the "unfolding" of a high-level Petri net, e.g., in [KLP06; LHY12]. Due to the obvious clash of notion with the later defined (symbolic) unfolding of high-level Petri nets, we use the term "expansion", as, e.g., in [CJ04].

- $\mathsf{T} := \{t.\sigma \mid t \in T, \sigma \in \Sigma(t)\}$,
- $\mathsf{F}(p.c, t.\sigma) := \mathit{pre}(t,\sigma)(p,c)$ and $\mathsf{F}(t.\sigma, p.c) := \mathit{post}(t,\sigma)(p,c)$ for $p.c \in \mathsf{P}, t.\sigma \in \mathsf{T}$,
- $\mathsf{M}_0 := \{\!\!\{\, p.c \mid (p,c) \in M_0 \,\}\!\!\}$.

Every marking M in $\mathrm{Exp}(N)$ corresponds to a marking M in N, with $M(p,c) = \mathsf{M}(p.c)$. A transition $t \in T$ can fire in mode $\sigma \in \Sigma(t)$ from M iff $t.\sigma \in \mathsf{T}$ can fire from M. The thereby reached markings then also correspond to each other, i.e., if $\forall p.c \in \mathsf{P} : \mathsf{M}'(p.c) = M'(p,c)$ then $M[t,\sigma\rangle M' \Leftrightarrow \mathsf{M}[t.\sigma\rangle \mathsf{M}'$. Thus, we say that N and $\mathrm{Exp}(N)$ have the same behavior. If N is finite then the expansion $\mathrm{Exp}(N)$ is finite iff *Col* is finite. We call N a *high-level representation* of $\mathrm{Exp}(N)$.

A high-level Petri net N is called *expansion safe* if the place/transition Petri net $\mathrm{Exp}(N)$ is safe. This is the case if and only if in no reachable marking in N, there is a place that holds the same color twice. Every safe high-level Petri net therefore is also expansion safe. If N is expansion safe then we can w.l.o.g. write all reachable markings, as well as pre- and postsets of transitions as sets rather than multi-sets. Transitions with two copies of the same (p,v) in their pre- or postset could never fire.

In the case of two or more initial markings in the set $\mathcal{M}_0$, we implicitly assume there to be a special initial transition $\perp$ that fires only once to initialize the net with any initial marking $M \in \mathcal{M}_0$. We then add, for every $M \in \mathcal{M}_0$, a transition $\perp_M$ to T, and define $\mathsf{F}(\perp_M, p.c) := M(p,c)$ for $p.c \in \mathsf{P}$. To ensure that the transition only fires once, we add a place $\mathsf{p}_\perp$ to P, and define $\mathsf{F}(\mathsf{p}_\perp, \perp_M) := 1$ for all $M \in \mathcal{M}_0$. Finally, we define the initial marking of $\mathrm{Exp}(N)$ to be $\mathsf{M}_0 := \{\!\!\{\, \mathsf{p}_\perp \,\}\!\!\}$.

2.3.2 High-Level Petri Games

Having described high-level Petri nets, we now move on to high-level Petri games. *Parameterized expansion safe high-level Petri games* were introduced in [GO21]. We only make implicit use of the parameters, in the sense that we may describe a high-level Petri net with the help of parameters, e.g., for the definition of its color set, but after that always consider the parameters to be fixed. We recall a slightly adapted version of the definition of high-level Petri games.

A *high-level Petri game* is a tuple $G = (\mathit{Col}, \mathit{Var}, P_\mathrm{S}, P_\mathrm{E}, T, F, g, \mathcal{M}_0, \mathrm{Obj}, P_\circledast)$ such that $N(G) := (\mathit{Col}, \mathit{Var}, P, T, F, g, \mathcal{M}_0)$ with $P := P_\mathrm{S} \sqcup P_\mathrm{E}$ is a high-level Petri net, and $P_\circledast \subseteq P_\mathrm{S}$. This net is called the *underlying high-level Petri net* of G. As in the low-level case, the places are divided into *environment places* P_E and *system places* P_S, and the *special places* $P_\circledast$ which, together with $\mathrm{Obj} \in \{\mathrm{Safety}, \mathrm{Reach}\}$ give the *objective* for the system players in the game. Dependent on Obj, we call $P_\circledast$ the set of *bad places* or *target places*, and denote it by $P_\spadesuit$ or $P_\heartsuit$, respectively.

Just as for nets, we sometimes assume *Col* and *Var* to be given, and omit them in a tuple describing a high-level Petri game. Let now *Col* and *Var* be given. A given high-level Petri game $G = (P_\mathrm{S}, P_\mathrm{E}, T, F, g, \mathcal{M}_0, \mathrm{Cond}, P_\circledast)$ can be *expanded* to a P/T Petri game $\mathrm{Exp}(G) = (\mathsf{P}_\mathrm{S}, \mathsf{P}_\mathrm{E}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Cond}, \mathsf{P}_\circledast)$, called the *expansion* of G, where the underlying net of $\mathrm{Exp}(G)$ is given by the expanded underlyng net of G, i.e., for

$\mathsf{P} := \mathsf{P}_\mathrm{S} \sqcup \mathsf{P}_\mathrm{E}$ we have $(\mathsf{P}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0) = \mathrm{Exp}(N(G))$. The classification of places as a system-, environment-, or special place then depends on the respective high-level place, i.e., $\forall p.c \in \mathsf{P} : p.c \in \mathsf{P}_x \Leftrightarrow p \in P_x$ for every $x \in \{\mathrm{S}, \mathrm{E}, \circledast\}$.

As for high-level Petri nets, we often consider high-level Petri nets with *one* initial marking M_0. In this case, we denote a high-level Petri game by $G = (\mathit{Col}, \mathit{Var}, P_\mathrm{S}, P_\mathrm{E}, T, F, g, M_0, \mathrm{Cond}, P_\circledast)$.

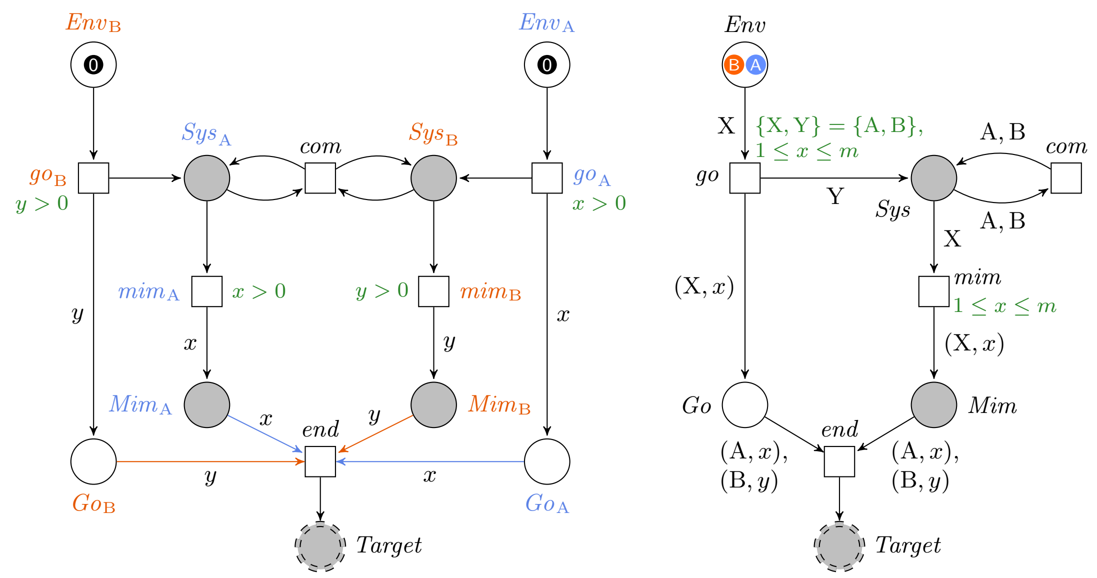
Illustrating High-level Petri games. While illustrating high-level Petri nets resp. -games (such as, e.g., in Fig. 2.6) we adopt from P/T Petri nets resp. -games the depiction of places as circles and transitions as squares, as well as the colors and border coding for the distinction between system places, environment places, and special places. Guards are written next to the respective transitions, and elements $(p, v, t), (t, v, p) \in F$ are illustrated as directed arcs with label v. To improve readability, several "tricks" are applied:

- We may write multiple variables on one arc to represent multiple arcs.
- The absence of a guard means that it always evaluates to *true*.
- In all our examples, we consider the color 0 to be special, by using it like a "neutral token" (as the token $\bullet$ in P/T Petri nets). We assume an unlabeled arc to be implicitly labeled by a variable v^0 (that is not used anywhere else), and the guard of the respective transition to additionally contain the proposition $v^0 = 0$.
- We further can label arcs with explicit colors $c \in \mathit{Col}$. Analogously to the situation described above this is shorthand for an arc labeled with a new variable v^c and the respective transition's guard containing the proposition $v^c = c$.
- Finally, we sometimes label the arcs with *tuples of variables* $(x_1, \dots, x_n)$ with $x_1, \dots, x_n \in \mathit{Var}$ although the set Col does not explicitly contain such tuples. In this case we implicitly extend Col to also contain tuples of colors, and view the tuple-labeled arc as shorthand for an arc labeled by a new variable $\vec{x}$ that is implicitly added to Var, and the respective transition having the proposition $\vec{x} = (x_1, \dots, x_n)$ in its guard. $\lhd$

We now proceed with our running example.

Example 2.3 (High-level "Communicating Copycats"). We fix $m = 2$ in this example. In Fig. 2.6, we see two high-level Petri games representing the running example "Communicating Copycats" (modulo renaming of nodes in the expansion). The high-level Petri game in Fig. 2.6a is a *safe* high-level representation of the Petri game in Fig 2.4. The representation in Fig. 2.6b is *not safe*, since in the initial merking, there are two colors on the same place. It is, however, *expansion safe*, meaning its expansion is a safe P/T Petri game.

The high-level Petri game in Fig. 2.6a with $\mathit{Col} = \{0, 1, 2\}$ has the initial marking $\{\!\!\{ (\mathit{Env}_\mathrm{A}, 0), (\mathit{Env}_\mathrm{B}, 0) \}\!\!\}$, which is illustrated by the tokens with color 0 on the respective places, and corresponds to the initial marking $\{\!\!\{ \mathit{Env}_\mathrm{A}, \mathit{Env}_\mathrm{B} \}\!\!\}$ in the P/T Petri net in Fig 2.4. The environment player on Env_B has the choice in which mode it wants to fire



(a) Safe high-level Petri game with $Col = \{0, 1, \dots, m\}$ and $Var = \{x, y\}$.

(b) Expansion safe high-level Petri game with $Col = \{A, B, 1, \dots, m\}$ and $Var = \{X, Y, x, y\}$.

Figure 2.6: Two high-level representations of the (underlying net of the) "Communicating Copycats" running example from Fig 2.4.

go_B, namely either $\{y \leftarrow 1\}$ or $\{y \leftarrow 2\}$. The assignment $\{y \leftarrow 0\}$ is not a mode of go_B, since the guard "$y > 0$" evaluates to *false* under this variable assignment. Independently of the environment's choice, the transition places a token of color 0 on the system place Sys_A. This system player knows in which mode go_B fired, and therefore which color, 1 or 2, has been placed on the place Go_B. The symmetric case is true for the system player that is placed on Sys_B; It knows which color $x \in \{1, 2\}$ was placed on Go_A. For the system players to be able to reach the target place *Target* by firing transition *end*, we see that there must be the same colors on Mim_A and Go_A, and on Mim_B and Go_B, respectively. This means that the transition mim_A must fire in the same mode as go_A did, and analogously for mim_B and go_B. For the system players on Sys_A and Sys_B to be able to allow the correct mode deterministically, the two must, as in the P/T case, take the joint transition *com*.

We continue our discussion of the running example by considering next the high-level Petri game in Fig. 2.6b (still for the fixed $m = 2$). It also is a high-level representation of the "Communicating Copycats" example. We immediately see that the net is not safe since initially, the environment players of color A and B are both residing on the place *Env*. From this initial marking, only transition *go* is fireable: In a mode σ of *go*, the color $\sigma(X) \in \{A, B\}$ is taken from *Env*, and the tuple $(\sigma(X), \sigma(x)) \in \{A, B\} \times \{1, 2\}$ is placed on *Go*. This represents the four initially activated transitions $go_{A1}, go_{A2}, go_{B1}, go_{B2}$ in the Petri game in Fig. 2.4. Note that *go* can fire twice and thus represents the choices of the two environment players. In the process of firing *go*, additionally a system player with color $\sigma(\mathrm{Y})$ is placed on *Sys*. The guard of *go* ensures that if $\sigma(\mathrm{X}) = \mathrm{A}$ then $\sigma(\mathrm{Y}) = \mathrm{B}$ and if $\sigma(\mathrm{X}) = \mathrm{B}$ then $\sigma(\mathrm{Y}) = \mathrm{A}$. This means that no two copies of the same color ever reside on *Sys*. Again, the goal of the two system players with colors A and B placed on *Sys* is to copy the choice of the environment player with the same color. They do this via transition *mim*. However, each system player placed on *Sys* by *go* only knows the choice made by the environment player with the respective *other* color. The system players can exchange their knowledge via the shared transition *com*. After firing it once, they both know the choice made by the environment player of their color, which they can then deterministically copy via transition *mim*. The transition *end* leading to the target place *Target* again only is activated when both system players correctly copied the respective environment player. We see that in every reachable marking, on every place there is at most one copy of each of the two colors A and B – either as an individual color or inside a tuple. Thus, the underlying high-level Petri net is expansion safe. $\lhd$

The *unfolding* of the high-level Petri game G is defined as the unfolding of $\mathrm{Exp}(G)$. Consequently, a *(winning) strategy* for the system players in G is defined as a (winning) strategy for the system players in $\mathrm{Exp}(G)$.

Summarizing, a high-level Petri game G (with fixed parameters) is a *succinct represention* of a P/T Petri game $\mathrm{Exp}(G)$, with the semantic of markings and firing of transitions being generalized versions from $\mathrm{Exp}(G)$, from which also the notions of unfoldings, strategies, and plays are borrowed.

2.3.3 Symbolic Branching Processes and the Symbolic Unfolding

We have reached the final topic of the preliminaries, where we explore the generalizations of branching processes and their associated maximal versions, unfoldings, to high-level Petri nets. These generalizations are referred to as symbolic branching process and symbolic unfolding [CJ04]. We start off by recalling the definition of symbolic branching processes and symbolic unfoldings from [CJ04].

A high-level net structure $\mathcal{N} = (\mathit{Col}, \mathit{Var}, P, T, F, g)$ is called *ordinary* iff there is at most one arc connecting any two nodes in $\mathcal{N}$, i.e., $\forall (x, y) \in (P \times T) \cup (T \times P) : \sum_{v \in \mathit{Var}} F(x, v, y) \leq 1$. For such an ordinary net structure, analogously to the low-level case, we say two nodes $x, y \in P \cup T$ are in *structural conflict*, denoted by $x \sharp y$, iff $\exists p \in P\, \exists t, t' \in T : t \neq t' \wedge p \to t \wedge p \to t' \wedge t \leq x \wedge t' \leq y$.

A *high-level occurrence net* is a high-level Petri net $O = (\mathit{Col}, \mathit{Var}, B, E, H, g, \mathcal{K}_0)$ with an ordinary net structure $(\mathit{Col}, \mathit{Var}, B, E, H, g)$, where B is a set of *conditions* (places), E is a set of *events* (transitions), H is a flow relation, and $\mathcal{K}_0$ is the set of initial *cuts* (reachable markings), such that the four properties below are satisfied.

The properties *i) – iii)* are exactly the same as in the P/T case and concern solely the net structure. Property *iv)* generalizes the corresponding requirement of low-level occurrence nets to the current situation, in which, just as in the low-level case every condition has at most one event in its preset, and that those conditions having an empty preset constitute the initial cut. Case *iv.a)* describes the conditions that initially hold a color, at the "top" of the net. Case *iv.b)* describes the conditions "deeper" in the net, which initially do not hold a color.

The properties that a high-level occurrence net must satisfy are:

i) No event is in structural self-conflict, i.e., $\forall e \in E : \neg(e \sharp e)$.

ii) No node is its own causal predecessor, i.e., $\forall x \in B \cup E : \neg(x < x)$.

iii) The flow relation is well-founded, i.e., $\forall x \in B \cup E : |\{y \mid y \leq x\}| < \infty$.

iv) For every $b \in B$, exactly one of the following holds:

- *a)* $\forall K_0 \in \mathcal{K}_0 : \sum_{c \in Col} K_0(b, c) = 1$ and $\{e \mid e \to b\} = \emptyset$.
  In this case we denote $pre(b) := (\bot, v^b)$.
- *b)* $\forall K_0 \in \mathcal{K}_0 : \sum_{c \in Col} K_0(b, c) = 0$ and there exists a unique pair (e, v) called $pre(b)$ s.t. $(e, v, b) \in H$, and for this pair we have $H(e, v, b) = 1$.

In a crucial notation for what follows in Chapter 4, we define in case *iv.a)* $\mathbf{e}(b) := \bot$, and $\mathbf{v}(b) := v^b$, and in case *iv.b)* we identify the event e by $\mathbf{e}(b)$ and the variable v by $\mathbf{v}(b)$. By this notation, we can say that whenever a condition b holds a color c, then it got placed there by firing $\mathbf{e}(b)$ in a mode binding $\mathbf{v}(b)$ to the color c.

We denote by $B_0 := \{b \in B \mid \exists K_0 \in \mathcal{K}_0 \, \exists c \in Col : (b, c) \in K_0\}$ the conditions from *iv.b)* occupied in all initial cuts. $\bot$ can be seen as a "special event" that fires only once to initialize the net, and produces the initial cuts $K_0 \in \mathcal{K}_0$ by assigning values to the variables v^b on "special arcs" $(\bot, v^b, b)$ towards the conditions $b \in B_0$. With the notation for $pre(b)$ introduced in *iv.a)* and *iv.b)* we have $\forall b \in B : pre(b) = (\mathbf{e}(b), \mathbf{v}(b))$.

In a high-level occurrence net, we define for every event e the predicates *loc-pred*(e) and *pred*(e). The *predicate pred*(e) is satisfiable iff e is not dead, i.e., there are cuts $K_0, \dots, K_n$ with $K_0 \in \mathcal{K}_0$ and events $e_1, \dots, e_n$, s.t. $K_0[e_1\rangle \dots [e_n\rangle K_n[e\rangle$. This predicate is obtained by building a conjunction over all so-called *local predicates* of events e' with $e' \leq e$, and the predicate of the special event $\bot$.

The local predicate of e is, in its turn, a conjunction of two predicates expressing that (i) the guard of the event e is satisfied, and (ii) that for every $(b, v) \in pre(e)$, the value of the variable v coincides with the color that the event $\mathbf{e}(b)$ placed in b. To realize this, the variables $v \in Var(e)$ are instantiated by the index e, so that v_e describes the value assigned to v by a mode of e. Having the definition of $\mathbf{e}(b)$ and $\mathbf{v}(b)$ from above in mind, for a condition b, we abbreviate $\mathbf{v_e}(b) := \mathbf{v}(b)_{\mathbf{e}(b)}$. Formally, we have

$$\begin{aligned} \textit{loc-pred}(e) \quad &:= \quad g(e)[v \leftarrow v_e]_{v \in Var(e)} \quad \wedge \bigwedge_{(b,v) \in pre(e)} v_e = \mathbf{v_e}(b) \\ \textit{pred}(e) \quad &:= \quad \textit{pred}(\bot) \wedge \bigwedge_{e' \leq e} \textit{loc-pred}(e'), \end{aligned}$$

where $\textit{pred}(\bot) := \textit{loc-pred}(\bot) := \bigvee_{K_0 \in \mathcal{K}_0} \bigwedge_{(b,c) \in K_0} (v^b_\bot = c)$ symbolically represents the set of initial cuts.

Since the notion of predicates is crucial for understanding symbolic unfoldings of high-level Petri nets, we demonstrate them on a generic example.

Example 2.4 (Predicates). Consider the generic example for a high-level occurrence net in Fig. 2.7. Next to this occurrence net, we progressively construct the local predicate of the contained event $\widetilde{e}$.

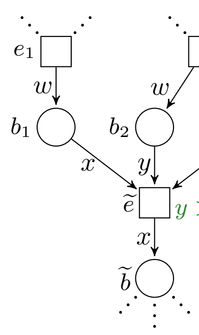


Figure 2.7: A generic high-level occurrence net.

$$\mathbf{e}(b_1) = e_1,\ \mathbf{v}(b_1) = w \Rightarrow \mathbf{v_e}(b_1) = w_{e_1}, \quad pre(b_1) = (e_1, w).$$
$$\mathbf{e}(b_2) = e_2,\ \mathbf{v}(b_2) = w \Rightarrow \mathbf{v_e}(b_2) = w_{e_2}, \quad pre(b_2) = (e_2, w).$$
$$\mathbf{e}(b_3) = e_2,\ \mathbf{v}(b_3) = y \Rightarrow \mathbf{v_e}(b_3) = y_{e_2}, \quad pre(b_3) = (e_2, y).$$

$$g(\widetilde{e}) = (y > x).$$
$$pre(\widetilde{e}) = \{(b_1, x), (b_2, y), (b_3, z)\}.$$

$$loc\text{-}pred(\widetilde{e}) = (y_{\widetilde{e}} > x_{\widetilde{e}}) \wedge (x_{\widetilde{e}} = w_{e_1}) \wedge (y_{\widetilde{e}} = w_{e_2}) \wedge (z_{\widetilde{e}} = y_{e_2}).$$

$$pred(\widetilde{e}) = loc\text{-}pred(\widetilde{e}) \wedge loc\text{-}pred(e_1) \wedge loc\text{-}pred(e_2) \wedge \dots\ .$$

$\triangleleft$

A *symbolic (initial) branching process* of a high-level Petri net N is a pair $\beta = (O, h)$ with an occurrence net $O = (Col, Var, B, E, H, g, \mathcal{K}_0)$ in which $pred(e)$ is satisfiable for all $e \in E$, and with a(n initial) homomorphism $h : O \to N$ that is injective on events with the same preset, i.e., $\forall e, e' \in E : (pre(e) = pre(e') \wedge h(e) = h(e')) \Rightarrow e = e'$.

For two symbolic branching processes $\beta = (O, h)$ and $\beta' = (O', h')$ of the same high-level Petri net, β is a *prefix* of β' if there exists an injective initial homomorphism ϕ from O into O', such that $h' \circ \phi = h$. In [CJ04] it is shown that for any given high-level Petri net N there exists a unique maximal symbolic initial branching process (maximal w.r.t. the prefix relation and unique up to isomorphism). This branching process is called the *symbolic unfolding*, and denoted by $\Upsilon(N) = (B^N, E^N, H^N, g^N, \mathcal{K}_0^N, \pi^N)$, where the high-level occurrence net $(B^N, E^N, H^N, g^N, \mathcal{K}_0^N)$ in $\Upsilon(N)$ is again denoted by $U(N)$, such that $\Upsilon(N) = (U(N), \pi^N)$.

We proceed by illustrating the various concepts about branching processes recalled above through a continuation of our running example.

Example 2.5 (Symbolic unfolding). Figure 2.8 shows the symbolic unfolding of the (underlying high-level Petri net of the) high-level Petri game Communicating Copycats from Fig. 2.6a. The names of events and conditions correspond to their label, and are distinguishable by superscripts if there is more than one occurrence of a node. Instead of its guard, next to every event the corresponding local predicate is indicated, where we again omit the parts about the "special variables" v^0 which can only be bound to the value 0, to improve readability. We depict the special event $\perp$ at the top of the net, and write its predicate $pred(\perp)$ below it. Firing $\perp$ *once* generates the initial cut of the symbolic unfolding, placing a 0 on both Env_{B} and Env_{A}, which enables the events go_{B} and go_{A}.

It follows from Fig. 2.6a that the initially enabled events go_{B} and go_{A} have the respective local predicates $y_{go_{\mathrm{B}}} > 0$ and $x_{go_{\mathrm{A}}} > 0$, respectively, which are indexed versions of the guards of the represented transitions. Here, we omitted for $loc\text{-}pred(go_{\mathrm{B}})$ the part

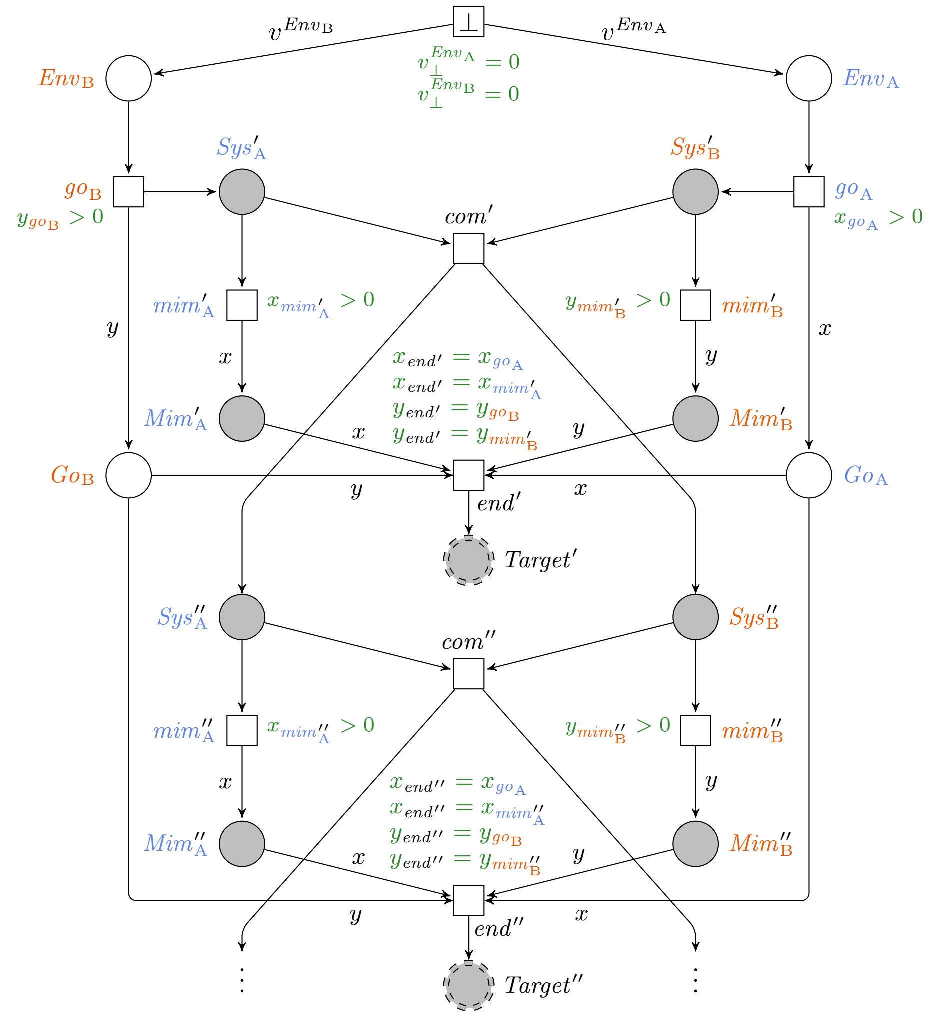


Figure 2.8: (A prefix of) the symbolic unfolding of the high-level version of our running example Communicating Copycats from Fig. 2.6a.

$v^0_{go_\mathrm{B}} = v^{Env_\mathrm{B}}_\perp$. Together with $pred(\perp) = (v^{Env_\mathrm{B}}_\perp = 0 \wedge v^{Env_\mathrm{A}}_\perp = 0)$, this would yield the predicate

$$\begin{aligned} pred(go_\mathrm{B}) &= loc\text{-}pred(go_\mathrm{B}) \wedge pred(\perp) \\ &= (y_{go_\mathrm{B}} > 0 \wedge v^0_{go_\mathrm{B}} = v^{Env_\mathrm{B}}_\perp) \wedge (v^{Env_\mathrm{B}}_\perp = 0 \wedge v^{Env_\mathrm{A}}_\perp = 0), \end{aligned}$$

and analogously for go_A. After firing go_B (or go_A), the respective system player of color 0 on Sys'_A (or Sys'_B) can directly make a choice of taking the event mim'_A (or mim'_B) in a mode assigning a value > 0 to x (or y). These two events represent the eight events ai^j, bi^j with $i, j \in \{1, 2\}$ in the (P/T) unfolding in Fig 2.5. We see that in contrast to the P/T unfolding, the output places of events are shared between all modes of an event.

The local predicate of end' contains four equalities, one "$v_{end'} = \mathbf{v_e}(b)$" for each

$(b, v) \in \mathit{pre}(\mathit{end}')$. For example, we have $(\mathit{Mim}'_{\mathrm{A}}, x) \in \mathit{pre}(\mathit{end}')$, with $\mathbf{e}(\mathit{Mim}'_{\mathrm{A}}) = \mathit{mim}'_{\mathrm{A}}$ and $\mathbf{v}(\mathit{Mim}'_{\mathrm{A}}) = x$. This results in the equality "$x_{\mathit{end}'} = x_{\mathit{mim}'_{\mathrm{A}}}$" contained in the local predicate of *end*′. The predicate of the event *end*′ is the conjunction of its own local predicate combined with all local predicates *loc-pred*(e) with $e < \mathit{end}'$. This predicate implies

$$x_{\mathit{go}_{\mathrm{A}}} > 0 \wedge x_{\mathit{mim}'_{\mathrm{A}}} = x_{\mathit{go}_{\mathrm{A}}} \quad \wedge \quad x_{\mathit{go}_{\mathrm{B}}} > 0 \wedge x_{\mathit{mim}'_{\mathrm{B}}} = x_{\mathit{go}_{\mathrm{B}}},$$

and this means that the event *end*′ can only fire if $\mathit{mim}'_{\mathrm{A}}$ fired in the same mode as go_{A}, and $\mathit{mim}'_{\mathrm{B}}$ fired in the same mode as go_{B}. We therefore directly see that "the system players must copy the choice of the respective environment players to reach the target place" in the Petri game.

If, instead of $\mathit{mim}'_{\mathrm{A}}$ and $\mathit{mim}'_{\mathrm{B}}$, the event *com*′ is fired, the tokens are afterwards placed on two new copies of $\mathit{Sys}_{\mathrm{A}}$ and $\mathit{Sys}_{\mathrm{B}}$, as in the low-level case. However, these two conditions now represent eight conditions in the P/T unfolding: The unique (purely structural) causal history of a node in the P/T unfolding is represented in the symbolic unfolding by the structural history together with all variable assignments needed to reach that node. As in the P/T case, we only display the prefix of the unfolding where at most two instances of *com* fired. ◁

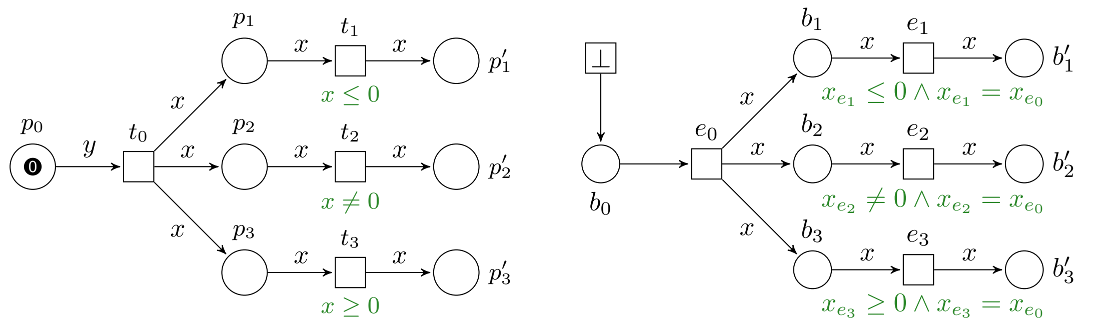
As we see in the definition of high-level occurrence nets, the notion of causality and structural conflict are the same as in the low-level case. However, a set of events in an occurrence net can also be in what we call *color conflict*, meaning that the conjunction of their predicates is not satisfiable. In a symbolic branching process, this means that the constraints on the values of the firing modes, coming from the guards of the transitions, prevent joint occurrence of all events from such a set in any *one* run of the net:

The nodes in a set $X \subseteq E \cup B$ are said to be in *color conflict* if $\bigwedge_{e \in X \cap E} \mathit{pred}(e) \wedge \bigwedge_{b \in X \cap B} \mathit{pred}(\mathbf{e}(b))$ is not satisfiable. The nodes of X are *concurrent* if they are *not* in color conflict, and for each $x, x' \in X$, neither $x < x'$, nor $x' < x$, nor $x \sharp x'$ holds. A set of concurrent conditions is called a *co-set*.

Note that while a set of nodes is defined to be in structural conflict if and only if two nodes in it are in structural conflict, the same does not hold for color conflict: it is possible to have a set $\{x_1, x_2, x_3\}$ of nodes that is in color conflict, but for which every subset of cardinality 2 is *not* in color conflict.

Example 2.6 (Color conflict). In Fig. 2.9a, a high-level Petri net with initial marking $\{\!|\, (p_0, 0) \,|\!\}$ is depicted. The only enabled transition is t_0, placing in every mode $\sigma = \{y \leftarrow 0, x \leftarrow \sigma(x)\}$, the same color $\sigma(x) \in \mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$ on each of the three places p_1, p_2, p_3. This color may be taken from the places by a respective transition from each of these places; The three transition t_1, t_2, t_3, however, each have a guard: $g(t_1) = (x \leq 0)$, $g(t_2) = (x \neq 0)$, and $g(t_3) = (x \geq 0)$. Depending on the mode σ in which t_0 fired, always two of the three transitions are fireable: if $\sigma(x) = 0$ then t_1 and t_3 can both fire (in mode $\{x \leftarrow \sigma(x)\}$), if $\sigma(x) < 0$ then t_1 and t_2 can fire, and if $\sigma(x) > 0$ then t_2 and t_3 can fire.

Since the high-level Petri net in Fig. 2.9a is a high-level occurrence net, it is structurally equal to its own symbolic unfolding in Fig. 2.9b. Again, instead of guards



(a) A high-level Petri net with $Col = \mathbb{Z}$ and $Var = \{x\}$.

(b) The symbolic unfolding of the net in (a), with the events $\{e_1, e_2, e_3\}$ in color conflict.

Figure 2.9: Illustration and example of nodes in color conflict.

of the events, the local predicates are written next to each event. To improve readability, we abbreviated for e_0 the associated local predicate $y_{e_0} = v_\perp^{b_0}$ together with $pred(\perp) = (v_\perp^{b_0} = 0)$ to $y_{e_0} = 0$. The set $\{b_1, b_2, b_3\}$ is a co-set, since the conditions are neither in conflict, nor causally related, and $\bigwedge_{b \in \{b_1,b_2,b_3\}} pred(\mathbf{e}(b))$, which is equivalent to $y_{e_0} = 0$, is satisfiable, i.e., the conditions are not in color conflict. Consequently, the set $\{e_1, e_2, e_3\}$ is also not in structural conflict, and the events are not causally related. However, we now have a color conflict between these three events, since $\bigwedge_{e \in \{e_1,e_2,e_3\}} pred(e)$ implies $x_{e_0} \leq 0 \wedge x_{e_0} \neq 0 \wedge x_{e_0} \geq 0$, which obviously is not satisfiable. However, each of the sets $\{e_i, e_j\}$ with $i, j \in \{1, 2, 3\}$, $i \neq j$ is *not* in color conflict. This makes each of the sets $\{b'_i, b'_j\}$ a co-set, while $\{b'_1, b'_2, b'_3\}$ is not a co-set. $\lhd$

Having employed the notions of conflict, we end this chapter with one the most important definitions when dealing with unfoldings, namely *configurations*.

Definition 2.7 (Configuration [CJ04]). A *(symbolic) configuration* is a set of high-level events that is free of structural conflict and color conflict, and causally closed. The configurations in a symbolic branching process β are collected in the set $\mathcal{C}(\beta)$. $\lhd$

For a configuration C, we define by $\mathrm{cut}(C) := (B_0 \cup (C \rightarrow)) \setminus (\rightarrow C)$ the *cut of* C, describing the high-level conditions that are occupied after any concurrent execution of C. Note that $\mathrm{cut}(C)$ is a co-set, and that $\emptyset$ is a configuration with $\mathrm{cut}(\emptyset) = B_0$.

Chapter 3

Exploiting Symmetries in High-Level Petri Games

Contents

In [FO17], Finkbeiner and Olderog reduce the problem of solving a low-level Petri game G with a single environment player and a bounded number of system players with a safety objective to the problem of solving a two-player game $\mathbb{G}(\mathsf{G})$ over a finite graph.

They show that Player 0 has a winning strategy in $\mathbb{G}(\mathsf{G})$ if and only if the system players have a winning strategy in G. In [Gie22], Gieseking generalizes this concept to Petri games with different objectives for the system players.

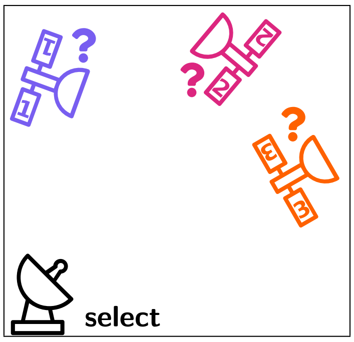
In practical scenarios, when high-level Petri games are used for modeling, the corresponding expansions (low-level Petri games) often exhibit a significant amount of symmetric behavior. This is primarily because individual players (tokens), such as robots, processes, or workpieces, do not necessarily need to exhibit distinct behaviors to achieve success in the game.

In this chapter, we introduce a solving algorithm tailored for a specific subclass of high-level Petri games that we call *proper*. These games feature a single environment player and a bounded number of system players. The novel algorithm takes advantage of the inherent symmetries within the system. The defined subclass is characterized by three key restrictions. Firstly, we focus on *expansion safe* high-level Petri games, where markings can be expressed as sets rather than multisets of individual colored tokens (cp. Sec. 2.3.1). Secondly, we consider Petri games where the single environment player is *recurrently interfering*, implying that in every infinite sequence of transitions, there are infinitely many environment transitions. Finally, we assume that there is no *mixed communication* in the net, meaning that a system player cannot communicate with on the one hand only the environment and on the other hand other system players in the same state.

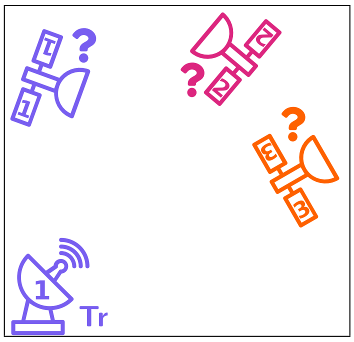
The algorithm's core concept combines the reduction technique employed in the related class of P/T Petri games, as detailed in [FO17; Gie22], with the creation of a *reduced reachability graph* for High-level Petri nets as presented in [CDFH97]. We introduce, for a given high-level Petri game G the *symbolic two-player game* $\overline{\mathbb{G}}(G)$, which is a two-player game over a finite graph with complete information. In this two-player game, Player 0 has a winning strategy if and only if in G the system players have a winning strategy. The vertices of $\overline{\mathbb{G}}(G)$ are equivalence classes with respect to symmetries of the vertices of the two-player game $\mathbb{G}(\mathsf{G})$ presented in [Gie22] of the represented low-level Petri game $\mathsf{G} = \mathrm{Exp}(G)$. The correctness of this construction is established through a bisimulation between these two two-player games. Additionally, we present an algorithm to derive a winning positional strategy in $\mathbb{G}(\mathsf{G})$ based on the winning positional strategy in $\overline{\mathbb{G}}(G)$.

Formally, the vertices in the symbolic two-player game $\overline{\mathbb{G}}(G)$ are, as mentioned above, equivalence classes of the vertices in the two-player game $\mathbb{G}(\mathsf{G})$, where $\mathsf{G} = \mathrm{Exp}(G)$. In the construction of $\overline{\mathbb{G}}(G)$, we use *arbitrary representatives* of the classes, from which we then build the arcs between the nodes. This has the consequence that during the construction of the game, after reaching a new node, we have to compare it to all other representatives computed up to that point and test them for equivalence with respect to symmetry.

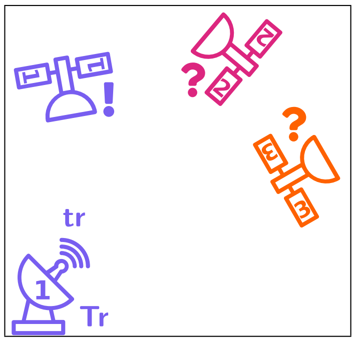
We therefore investigate employing so-called *canonical representations* of the equivalence classes instead. We use techniques presented in [CDFH91a; CDFH93], where the authors address a same problem for markings in the so-called *symbolic reachability graph* of a high-level Petri net. With these, the canonical representation of a new node can be automatically calculated, and instead of comparing it to every other node in the game for *equivalence* with respect to symmetry, which can be expensive, we only have to check for *equality*. Employing these canonical representations as nodes, we arrive at a two-player game that we call the *canonical two-player game* and denote it by $\widehat{\mathbb{G}}(G)$. We show that this game is isomorphic to $\overline{\mathbb{G}}(G)$, which means that we can transfer all results from the symbolic two-player game to the canonical two-player game. Finally, we compare the speed of construction for these two games, with the result that in most cases, the canonical two-player game is faster to construct.



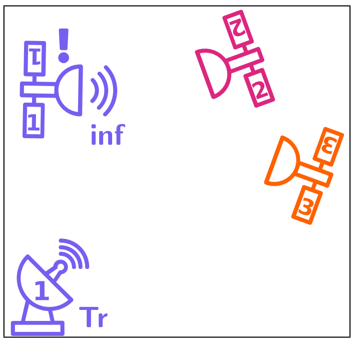
(a) The base station is approximating which satellite to transmit a message to.



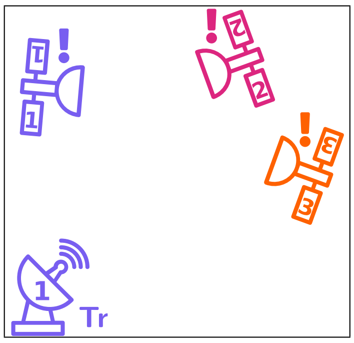
(b) The base station starts a transmission to satellite 1.



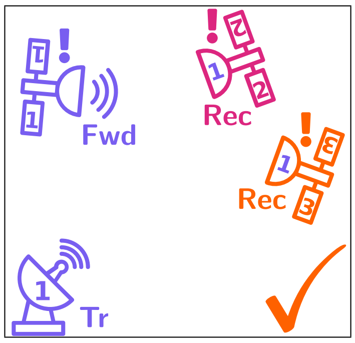
(c) Satellite 1 gets informed about the transmission.



(d) Satellite 1 informs the other two satellites about the transmission.



(e) All satellites know about the transmission.



(f) Satellite 1 forwards the transmitted message, satellites 2 and 3 receive it.

Figure 3.1: An exemplary sequence of events in the running example “Signal Sending Satellites”.

Running Example: Signal Sending Satellites. We introduce our second running example, called *Signal Sending Satellites* – an abstract model of a base station sending a message to one of three satellites (called Satellite 1, 2, 3). This satellite has to forward the message to the other two, which have to receive the signal. Before modeling this scenario as a high-level Petri game, we go through an exemplary sequence of events that is schematically illustrated in Fig. 3.1.

Initially, in Fig. 3.1a, the base station selects one of three satellites to send a message to. The number of each satellite is indicated on its "wings" (its solar panels). In this example, Satellite 1 is chosen and the transmission begins, as shown in Fig. 3.1b. At this point, none of the satellites knows about the transmission, as indicated by the question marks in the figure. In Fig. 3.1c, Satellite 1 detects the transmission (indicated by the exclamation mark) and identifies itself as the recipient of the message. Satellite 1 then shares this information with the other two satellites, as shown in Fig. 3.1d. After all satellites have been informed, as illustrated in Fig. 3.1e, Satellite 1 proceeds to forward the message, while the other two satellites receive it from Satellite 1 (as depicted in Fig. 3.1f). Once the transmission is complete, we return to the initial state in Fig. 3.1a, ready for the base station to select a different satellite to transmit to. The goal of the satellites is to ensure that, regardless of which satellite the base station chooses, the message must always be forwarded to the other two satellites.

We model this abstract scenario as a high-level Petri game with safety-objective for the system players shown in Fig. 3.2. Before we go into detail, we explain the ideas of the model. The base station serves as the environment, and the system components (the three satellites) have to react correctly under all possible behavior of the environment. In this case, the environment's behavior is to select one of the three satellites that must forward a message. Each satellite on the other hand must eventually choose between two modes of operation: "Forwarding mode", and "Receiving mode" during which they specify the satellite from which they wish to receive a message. Ideally, each satellite makes this choice informed about a currently ongoing transmission. To that goal, when a satellite is chosen by the environment, it can learn this information and inform the other satellites about it.

However, each satellite also has the option to make a direct decision without getting informed about a possible transmission. Similarly, when a satellite learns that it must forward a message, it has the option to make the decision without informing the other two satellites. It is crucial that the selected satellite shares this information, so that each satellite can forward or receive the message correctly. If this is achieved, the transmission can be completed and the base station can choose a new satellite for the next transmission.

The illustration in Fig. 3.2, employs again the readability improving "tricks" described in Sec. 2.3.2, with a "special color" $\mathbf{0}$, and multiple variables or tuples of variables on arcs. The set of colors is given by $\mathit{Col} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$, where $\mathbf{1}, \mathbf{2}, \mathbf{3}$ represent the three satellites.

Initially, the environment player of color $\mathbf{0}$ resides on the place *Env* and the three satellites, acting as system players, reside on the system place *Init*. Two transitions are activated in that marking: via *select*, the environment selects one of the satellites by assigning to x a value $\widetilde{x} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$. Transition *uninf* places a satellite on the place *Ch*, from where it then can proceed into the Forward mode via *fwd* or the Receiving mode via *rec*. The transition *uninf* therefore represents the satellite's possibility to make the choice of a mode uninformed of a possible transmission.

However, after the environment selected the satellite and placed $\widetilde{x}$ on place *Tr* (rep-

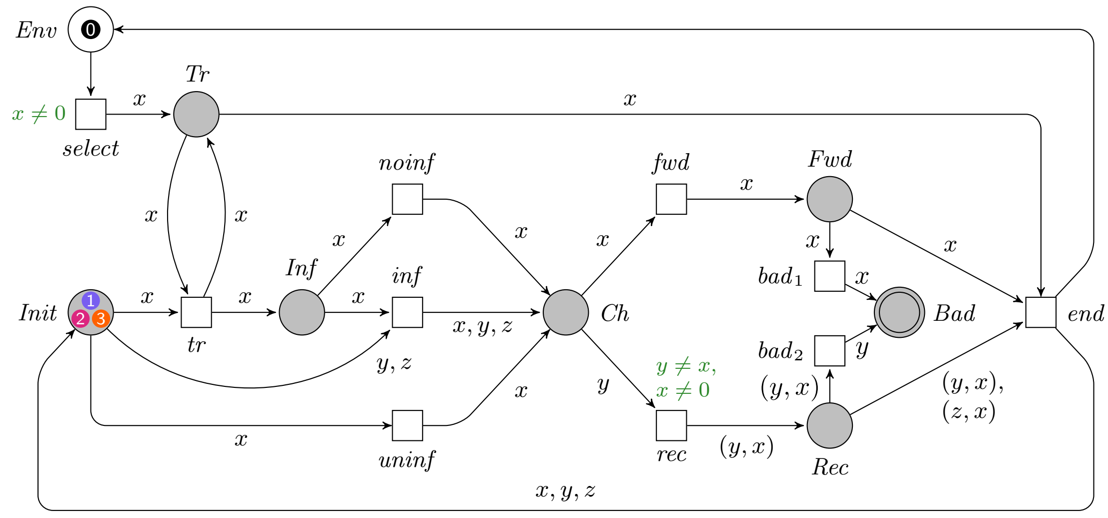


Figure 3.2: The running example "Signal Sending Satellites" with $\mathit{Col} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$ and $\mathit{Var} = \{x, y, z\}$

resenting a transmission to Satellite $\widetilde{x}$), the respective satellite can learn about the transmission by taking transition *tr* in mode $\{x \leftarrow \widetilde{x}\}$. Afterwards, from place *Inf*, it now has two possibilities, both leading to $\widetilde{x}$ being placed on *Ch*: it either takes transition *noinf*, not informing the other satellites, or transition *inf*. Here, the other two satellites are taken from *Init*, and all three are placed on *Ch*.

If *inf* fired, all three satellites now have the firing of *select* in their past, and therefore know the value $\widetilde{x}$. They can thus make an informed choice whether to go into Forwarding mode (*fwd*), or Receiving mode (*rec*) specifying the satellite from which they wish to receive a message. When acting correctly, Satellite $\widetilde{x}$ takes *fwd* to place *Fwd*, representing that it is in Forwarding mode, and the other two satellites y_1 and y_2 take transition *rec* in the modes $\{y \leftarrow y_i, x \leftarrow \widetilde{x}\}$, $i = 1, 2$. This places the tuples $(y_1, \widetilde{x})$ and $(y_2, \widetilde{x})$ on *Rec*, representing that the two satellites are in Receiving mode, wishing to receive a message from Satellite $\widetilde{x}$.

Finally, transition *end* can fire in mode $\{x \leftarrow \widetilde{x}, y \leftarrow y_1, z \leftarrow y_2\}$, restoring the initial marking and allowing the environment to again select one of the satellites. Notice that the marking $\{\!\!\{\, (\mathit{Tr}, \widetilde{x}), (\mathit{Fwd}, \widetilde{x}), (\mathit{Fwd}, (y_1, \widetilde{x})), (\mathit{Fwd}, (y_2, \widetilde{x}))\,\}\!\!\}$ from where *end* can fire could also have been reached if the system players coincidentally made the right choice *without* getting informed. However, since in a strategy they have to make their choice deterministically and dependent only on their respective knowledge, this would mean the choice is independent of the environment's selection. So if the environment chose another satellite then the net would arrive at a marking $\{\!\!\{\, (\mathit{Tr}, y_1), (\mathit{Fwd}, \widetilde{x}), (\mathit{Fwd}, (y_1, \widetilde{x})), (\mathit{Fwd}, (y_2, \widetilde{x}))\,\}\!\!\}$ or $\{\!\!\{\, (\mathit{Tr}, y_2), (\mathit{Fwd}, \widetilde{x}), (\mathit{Fwd}, (y_1, \widetilde{x})), (\mathit{Fwd}, (y_2, \widetilde{x}))\,\}\!\!\}$. From there, the transition *end* could not fire, and the system players would have to take transition bad_1 or bad_2, respectively, reaching the bad place *Bad* and loosing the play.

Note. Since a color $\widetilde{x}$ on *Tr* represents the base station (environment) transmitting a message to Satellite $\widetilde{x}$, it would be more intuitive for *Tr* to be an environment place. We model *Tr* as a system place so that the Petri game satisfies a property later called "having no mixed communication". This means that there is no system place that has pure system transitions as well as environment transitions in its postset. This restriction of Petri games has been introduced in [Gie22] to ensure correctness of the presented solving algorithms. Since in this chapter we build on these algorithms, we also assume the considered Petri games to have no mixed communication (cp. 3.1.2). In this example it makes no difference whether *Tr* is a system place or environment place. In general,

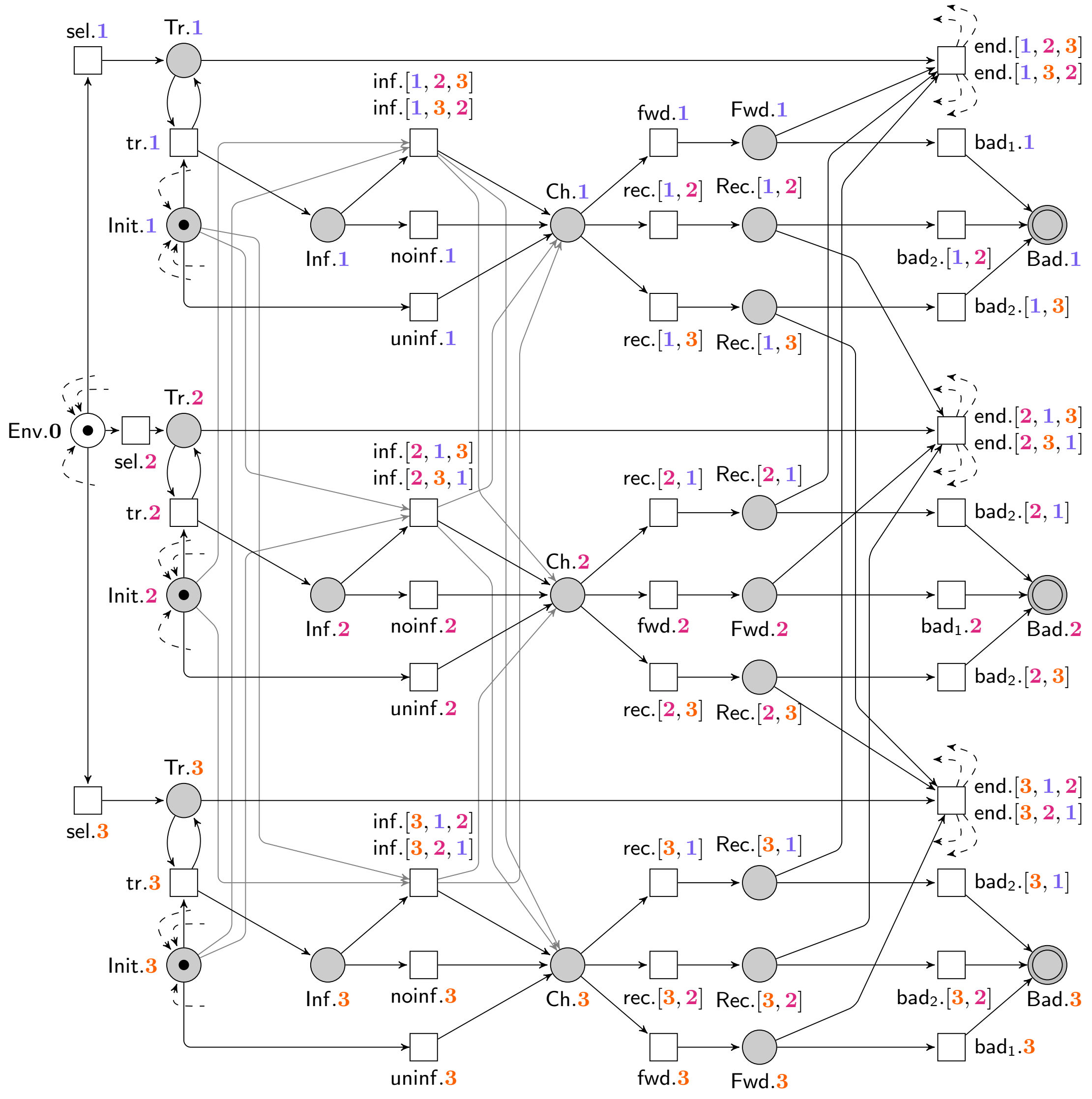


Figure 3.3: The P/T expansion of the high-level game Signal Sending Satellites from Fig. 3.2.

it is often possible to remodel a given Petri game to satisfy "no mixed communication" with minor adaptations (e.g., adding some places or transitions) while maintaining the general semantics of the Petri game. For further details, we refer the reader to [Gie22].

For the sake of completeness, we present in Fig. 3.3 the expansion of the running example Signal Sending Satellites. We abbreviate the modes of transitions by the assigned values, e.g., $\mathit{tr}.\{x \leftarrow \mathbf{1}\}$ is abbreviated by $\mathit{tr}.\mathbf{1}$, and $\mathit{inf}.\{x \leftarrow \mathbf{1}, y \leftarrow \mathbf{3}, z \leftarrow \mathbf{2}\}$ by $\mathit{inf}.[\mathbf{1}, \mathbf{3}, \mathbf{2}]$. A special case are the instances of *rec*, where we abbreviate the mode "non-alphabetically", e.g., $\mathit{rec}.\{y \leftarrow \mathbf{2}, x \leftarrow \mathbf{3}\}$ is abbreviated by $\mathit{rec}.[\mathbf{2}, \mathbf{3}]$, since firing this transition places a token on $\mathit{Rec}.(\mathbf{2}, \mathbf{3})$. The high-level transitions *inf* and *end* each have the same preset and postset for the two modes $\{x \leftarrow \widetilde{x}, y \leftarrow \widetilde{y}, z \leftarrow \widetilde{z}\}$ and $\{x \leftarrow \widetilde{x}, y \leftarrow \widetilde{z}, z \leftarrow \widetilde{y}\}$. We therefore have three pairs of low-level transitions with the same preset and postset. We draw one transition for each pair and label it by both names. Finally, for the instances of *end*, we only imply the postsets by the dashed arrows: every instance of *end* places one token on the environment place $\mathit{Env}.\mathbf{0}$ and one token on every system place $\mathit{Init}.\widetilde{x}$. To ensure readability, we restricted the transitions and places that can eventually fire resp. appear in any reachable marking.

3.1 Central Concepts

In the running example Signal Sending Satellites from Fig. 3.1 and Fig. 3.2, we observe that in all three possible cases of "satellite selection" by the environment, the situations are completely *symmetric* to each other. The concept of symmetry is discussed in detail in Sec. 3.1.1. In the solving algorithm for P/T Petri games in [FO14; FO17; Gie22], a two-player game on a graph is constructed from a given Petri game. Section 3.1.3 presents a simplified version of this two-player game for a restricted class of Petri games introduced in Sec. 3.1.2. These two concepts are combined in the later sections 3.2 and 3.3, by applying symmetries from a high-level Petri game to the vertices in the constructed two-player game.

3.1.1 Symmetries on High-Level Petri Nets

As mentioned above, the possible behavior of all satellites in the Signal Sending Satellites example is completely *symmetric*: Each satellite has the same set of actions available to it, including interactions with the environment and other satellites. Therefore, the specific satellite selected by the environment does not matter. The selected satellite should inform the other two and switch to Forwarding mode, while the other two should switch to Receiving mode, following the informally described strategy. This subsection formalizes the concept of symmetries in high-level Petri nets, which serves as the foundation for the results presented in this chapter.

A *symmetry* s in a high-level Petri net $N = (\mathit{Col}, \mathit{Var}, P, T, F, g, \mathcal{M}_0)$ is a permutation on the set of colors Col. Let $\mathbb{S}(N)$ be the set of all symmetries on N. Since permutations are bijective, together with the function composition $\circ$ the symmetries form a group $(\mathbb{S}(N), \circ)$ with $\mathbf{1}_{\mathbb{S}(N)} = \mathit{id}_{\mathit{Col}}$.

From the definition, we only get that symmetries map colors to colors. We now naturally lift this to different elements. A symmetry s can be applied to a mode σ of a transition by the function composition; $s(\sigma) := s \circ \sigma$. When applied to a set $A \subseteq P \times \mathit{Col}$ of place-color combinations (as, e.g., a marking or transition's preset), a symmetry acts only in the second coordinate, i.e., $s(A) := \{(p, s(c)) \mid (p, c) \in A\}$. Analogously, for a set A of transition-mode combinations (t, σ) with $\sigma \in \Sigma(t)$ we define $s(A) := \{(t, s(\sigma)) \mid (t, \sigma) \in A\}$. For a guard $g(t)$ of a transition t, we denote by $s(g(t))$ the predicate where every variable x is replaced by the term $s(x)$.

The purpose of symmetries is to express an equivalence of situations, so that we only have to consider one representative of each equivalence class when analyzing a high-level Petri net's behavior. However, the definition above is to general for this aim since it does not consider the syntax and semantics of the net. We now do exactly this and thereby restrict the symmetries to the ones that ensure that from "symmetric" situations we get equivalent possible behavior.

Definition 3.1 (Admissible symmetries). Let $N = (\mathit{Col}, \mathit{Var}, P, T, F, g, \mathcal{M}_0)$ be a high-level Petri net. The set of *admissible symmetries* is the biggest subset $S \subseteq \mathbb{S}(N)$ such that $(S, \circ)$ is a subgroup of $(\mathbb{S}(N), \circ)$ satisfying $\forall s \in S$:

- $s(\mathcal{M}_0) = \mathcal{M}_0$,
- $\forall t \in T: \quad s(g(t)) \equiv g(t), \quad \text{and} \quad \forall \sigma \in \Sigma(t)$
  - (i) $s(\mathit{pre}(t, \sigma)) = \mathit{pre}(t, s(\sigma))$, and
  - (ii) $s(\mathit{post}(t, \sigma)) = \mathit{post}(t, s(\sigma))$.

We denote this set S by $S(N)$. $\lhd$

The first condition means that the set of initial markings is invariant under the application of symmetries. In the special case $\mathcal{M}_0 = \{M_0\}$ of exactly one initial marking M_0, this implies $s(M_0) = M_0$ for all symmetries $s \in S(N)$. The second condition ensures that the symmetries are "compatible" with the firing of transitions: if a transition t, fireable in mode σ, takes the color c from a place p (i.e., $(p, c) \in \mathit{pre}(t, \sigma)$), then it should be fireable in mode $s(\sigma)$ and, when fired, take color $s(c)$ from place p (i.e., $(p, s(c)) \in \mathit{pre}(t.s(\sigma))$). The same applies to the postset of t. Hence, admissible symmetries on a high-level Petri net are those symmetries which are compatible with the net's semantics.

For a high-level Petri game G, we adopt the notion of (admissible) symmetries from its underlying high-level Petri net $N(G)$, i.e., $S(G) := S(N(G))$.

The following lemma formalizes the compatibility of admissible symmetries with the firing relation.

Lemma 3.2 ([CDFH97], Property 2.1). *Let N be a high-level Petri net with transitions T and admissible symmetries $S = S(N)$. These symmetries are* compatible *with transition firing, i.e., for markings M, M' in N, we have $\forall t \in T\ \forall \sigma \in \Sigma(t)\ \forall s \in S$:*

$$M[t, \sigma\rangle M' \Leftrightarrow s(M)[t, s(\sigma)\rangle s(M').$$

Example 3.3 (Symmetries in Signal Sending Satellites). In the running example G from Fig. 3.2, we wrote the set of colors as $\mathit{Col} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$, which would mean that the group of admissible symmetries is a subgroup of $\mathit{Sym}(\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\})$, the group of all permutations on $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$. However, we make use of an abbreviation by tuples on arcs, meaning that we implicitly extend the colors to contain tuples (as described in Sec. 2.3.2). This means the set of all symmetries is actually given by the set of all permutations on $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\} \cup (\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\} \times \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\})$. We now explicitly treat this "extended" notion. We thereby see that here the abbreviation is consistent with the concept of symmetries by showing that the group of admissible symmetries can in fact be identified with a subgroup of $\mathit{Sym}(\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\})$.

Let s be an *admissible* symmetry. We immediately see from the initial marking $M_0 = \{(\mathit{Env}, \mathbf{0}), (\mathit{Init}, \mathbf{1}), (\mathit{Init}, \mathbf{2}), (\mathit{Init}, \mathbf{3})\}$ that, since $s(M_0) = M_0$, it must hold that $s(\mathbf{0}) = \mathbf{0}$ and that s maps $\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$ into $\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$. Since s is bijective, this implies that s maps $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\} \times \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$ into $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\} \times \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$.

However, it must also hold condition (i) from Definition 3.1. In particular, this means for the transition bad_2 that $\forall \widetilde{x}, \widetilde{y} \in \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$:

$$\begin{aligned}\{(\mathit{Rec}, s((\widetilde{y}, \widetilde{x})))\} &= s(\mathit{pre}(\mathit{bad}_2, \{x \leftarrow \widetilde{x}, y \leftarrow \widetilde{y}\})) \overset{\text{(i)}}{=} \mathit{pre}(\mathit{bad}_2, s(\{x \leftarrow \widetilde{x}, y \leftarrow \widetilde{y}\})) \\ &= \mathit{pre}(\mathit{bad}_2, \{x \leftarrow s(\widetilde{x}), y \leftarrow s(\widetilde{y})\}) = \{(\mathit{Rec}, (s(\widetilde{y}), s(\widetilde{x})))\}.\end{aligned}$$

Therefore, we have that $\forall \widetilde{x}, \widetilde{y} \in \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\} : s((\widetilde{y}, \widetilde{x})) = (s(\widetilde{y}), s(\widetilde{x}))$, which means that the values of s on $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\} \times \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$ are given by its values on $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$. Together with the fixed element $\mathbf{0}$ (and after checking that no other transition restricts admissibility any further) we conclude that the set S of admissible symmetries can be identified by the permutation group $\mathit{Sym}(\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}) = S_3$ of permutations on $\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$.

Going back to the informal description, this makes perfect sense, since the satellites (colors $\mathbf{1}, \mathbf{2}, \mathbf{3}$) have symmetric behavior, and the base station (color $\mathbf{0}$) has a special role that is not symmetric to the satellites.

We now verify that the firing of transitions is compatible with the admissible symmetries, as stated in Lemma 3.2. Consider the case where the environment selected Satellite 2 via transition *select* in mode $\{x \leftarrow \mathbf{2}\}$, and the satellite in its turn got informed by that selection via transition *tr* in mode $\{x \leftarrow \mathbf{2}\}$. This leads to the marking $M = \{(\mathit{Tr}, \mathbf{2}), (\mathit{Init}, \mathbf{1}), (\mathit{Inf}, \mathbf{2}), (\mathit{Init}, \mathbf{3})\}$. Consider now the admissible symmetry s given by $s(\mathbf{1}) = \mathbf{2}$, $s(\mathbf{2}) = \mathbf{3}$, and $s(\mathbf{3}) = \mathbf{1}$. Then it is $s(M) = \{(\mathit{Tr}, \mathbf{3}), (\mathit{Init}, \mathbf{1}), (\mathit{Init}, \mathbf{2}), (\mathit{Inf}, \mathbf{3})\}$. From the marking M, we can fire, e.g., *inf* in mode $\sigma = \{x \leftarrow \mathbf{2}, y \leftarrow \mathbf{1}, z \leftarrow \mathbf{3}\}$:

$$M[\mathit{inf}, \sigma\rangle M' \quad \text{with} \quad M' = \{(\mathit{Tr}, \mathbf{2}), (\mathit{Ch}, \mathbf{1}), (\mathit{Ch}, \mathbf{2}), (\mathit{Ch}, \mathbf{3})\}.$$

With $s(\sigma) = \{x \leftarrow \mathbf{3}, y \leftarrow \mathbf{2}, z \leftarrow \mathbf{1}\}$ and $s(M') = \{(\mathit{Tr}, \mathbf{3}), (\mathit{Ch}, \mathbf{1}), (\mathit{Ch}, \mathbf{2}), (\mathit{Ch}, \mathbf{3})\}$ we verify $s(M)[\mathit{inf}, s(\sigma)\rangle s(M')$. $\lhd$

Note that we have now discussed the application of symmetries to colors and modes. Considering the expansion $\mathrm{Exp}(G) = (\mathsf{P}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0)$ of a high-level Petri game G, this corresponds to applying symmetries to (low-level) places and conditions. To see this, let $s \in S(G)$. For every place $\mathsf{p} = p.c \in \mathsf{P}$, we define $s(\mathsf{p}) := p.s(c)$. Analogously, for every transition $\mathsf{t} = t.\sigma \in \mathsf{T}$, we define $s(\mathsf{t}) := t.s(\sigma)$. With this definition, we get $\forall s \in S(G) : s(\mathsf{M}_0) = \mathsf{M}_0$, and $\forall s \in S(G) : \mathsf{M}[\mathsf{t}\rangle\mathsf{M}' \Leftrightarrow s(\mathsf{M})[s(\mathsf{t})\rangle s(\mathsf{M}')$.

3.1.2 Proper High-Level Petri Games

Although in the definition of high-level Petri nets we permit only a single variable on every arc, and assume guards to be decidable, we still allow for complicated behavior. We could for example deal with uncountable color sets, or nets with unbounded concurrent behavior caused by an unbounded number of colors in reachable markings. Since we want to solve Petri games with an underlying high-level Petri net structure, it is reasonable to exclude behavior from which we know that we cannot handle it.

To that aim we introduce a class of high-level Petri games called **G**, which are considered “proper”. In later sections, we will apply the concept of symmetries to the two-player games used in the solving algorithm from [FO17; Gie22] for this class. We now explain the motivation and benefits for each of the four assumptions that define proper high-level Petri games. After the definition, we give in Fig 3.4 four Petri games, each *violating* (at least) one of these assumptions, and proceed by showing that the running example Signal Sending Satellites from Fig. 3.2 is a proper high-level Petri game.

The algorithm for solving *k-bounded* P/T Petri games with a safety objective, as introduced in [FO17], is generalized and extended in [Gie22] to incorporate different objectives for the system players. However, [Gie22] only considers 1-*bounded* (i.e., safe) Petri games. In this chapter, we also restrict ourselves to safe Petri games, which simplifies the technical definitions in subsequent sections. Moreover, this allows us to represent markings, presets, and postset (as well as the later introduced decision sets), as *sets* instead of *multi-sets*. Recall that safe Petri nets correspond to **expansion safe** high-level Petri nets (cp. p. 25).

A decision taken by the strategy in a place p depends on the causal past of p, which may be arbitrarily large. Similar to model checking approaches based on net unfoldings [EH08], we use *cuts* (maximal subsets of pairwise concurrent places in the unfolding) as unique representatives of the causal past. Cuts are directly related to the concept of *configurations* – causally closed sets of events without conflicts: The markings generated by configurations in the unfolding are exactly all cuts. The standard notion of cuts, however, collects places with possibly different knowledge of the individual players about the causal past.

The paper [FO17] introduced a new kind of cut, called *maximal cuts*, abbreviated *mcut*. For an environment place p, an mcut is a cut including p such that for all places q in that cut either (1) the system players have *maximally progressed* at q, in the sense that any further system transition would require an additional environment transition starting from place p, or (2) the future starting at q does not depend on the environment. Mcuts are especially interesting for strategies in Petri games with a **single environment player**. For Petri games with one environment player, every maximally progressed system player of an mcut (case (1)) can be considered to be equally informed about the environment: the next transition either directly involves the environment player or at least contains the current environment place in its causal past.

This fact is exploited in [FO17; Gie22] to create a two-player game with *complete* information which is used to solve Petri games. Since we want to apply symmetries to these solving algorithms, we therefore consider Petri games with at most one environment player. For simplicity, we additionally restrict ourselves to Petri games where alternative (2) in mcuts does not arise. In the terminology of [FO17], we do not consider type-2 places here. In other words, we require that the system players cannot proceed infinitely without the environment. We call such an environment *recurrently interfering*:

Definition 3.4 (Recurrently interfering environment). We say a P/T Petri game $\mathsf{G} = (\mathsf{P}_\mathrm{S}, \mathsf{P}_\mathrm{E}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Obj}, \mathsf{P}_\circledast)$ has a *recurrently interfering* environment if in every infinite firing sequence $\mathsf{M}_0\,[\mathsf{t}_1\rangle\,\mathsf{M}_1\,[\mathsf{t}_2\rangle \dots$ in G, it is $\mathit{pre}(\mathsf{t}_i) \cap \mathsf{P}_\mathrm{E} \neq \emptyset$ for infinitely many $i \in \mathbb{N}$. Analogously, we say a high-level Petri game $G = (\mathit{Col}, \mathit{Var}, P_\mathrm{S}, P_\mathrm{E}, T, F, g, M_0, \mathrm{Obj}, P_\circledast)$ has a *recurrently interfering* environment if in every infinite firing sequence $M_0\,[t_1, \sigma_1\rangle$ $M_1\,[t_2, \sigma_2\rangle \dots$ in G, it is $\mathit{pre}(\mathsf{t}_i, \sigma_i) \cap (P_\mathrm{E} \times \mathit{Col}) \neq \emptyset$ for infinitely many $i \in \mathbb{N}$. $\lhd$

Thus, G has a recurrently interfering environment if and only if $\mathrm{Exp}(G)$ has a recurrently interfering environment. Assuming the environment to be recurrently interfering simplifies the formal definitions of elements used in the solving algorithm for Petri games, namely decision sets and the two-player game, and eliminates the need to introduce an additional pre-processing algorithm similar to the one presented in [FO17] or a round-robin component used in the algorithm in [Gie22]. Both of these handle the cases where the system players can proceed infinitely without interacting with the environment.

Finally, in accordance with [Gie22], we limit our attention to Petri games that adhere to the principle of "no mixed communication". Specifically, this means that the structure of the game (resp. its underlying net) ensures that system players never have the possibility to provide both a communication involving the environment or one involving only other system players.

Definition 3.5 (Mixed communication [Gie22]). We say a P/T Petri game $\mathsf{G} = (\mathsf{P}_\mathrm{S}, \mathsf{P}_\mathrm{E}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Obj}, \mathsf{P}_\circledast)$ has *mixed communication* if

$$\exists \mathsf{p} \in \mathsf{P}\ \exists \mathsf{t}_1, \mathsf{t}_2 \in \mathit{post}(\mathsf{p}) : \mathit{pre}(\mathsf{t}_1) \cap \mathsf{P}_\mathrm{E} \neq \emptyset \wedge \mathit{pre}(\mathsf{t}_2) \subseteq \mathsf{P}_\mathrm{S}.$$

Analogously, a high-level Petri game $G = (\mathit{Col}, \mathit{Var}, P_\mathrm{S}, P_\mathrm{E}, T, F, g, M_0, \mathrm{Obj}, P_\circledast)$ has *mixed communication* if

$$\exists p \in P\ \exists t_1, t_2 \in p{\rightarrow} : \mathit{pre}(t_1) \cap (P_\mathrm{E} \times \mathit{Var}) \neq \emptyset \wedge \mathit{pre}(t_2) \subseteq (P_\mathrm{S} \times \mathit{Var}). \qquad \lhd$$

As for recurrently interfering environments, a high-level Petri game G by definition has mixed communication if and only if $\mathrm{Exp}(G)$ has mixed communication. With the definitions above we can now define the class *proper* Petri games, for which we generalize and build upon the solving algorithms from [FO17; Gie22] in this chapter.

Definition 3.6 (Proper (high-level) Petri games). The class $\mathbf{G}$ consists of all finite high-level Petri games $G = (\mathit{Col}, \mathit{Var}, P_S, P_E, T, F, g, M_0, \mathrm{Obj}, P_{\circledast})$ such that $|\mathit{Col}| < \infty$ and

- G is expansion safe, i.e., $\forall M \in \mathcal{R}(N(G))\, \forall p \in P\, \forall c \in \mathit{Col} : M(p, c) \leq 1$,
- G has only one environment, i.e., $\forall M \in \mathcal{R}(N(G)) : \sum_{(p,c)\in P_E \times \mathit{Col}} M(p, c) \leq 1$,
- G has a recurrently interfering environment (cp. Definition 3.4),
- G has *no* mixed communication (cp. Definition 3.5).

We call the elements of $\mathbf{G}$ *proper high-level Petri games.* We denote by $\mathrm{Exp}(\mathbf{G})$ the class of P/T Petri games G such that $\mathsf{G} = \mathrm{Exp}(G)$ for some $G \in \mathbf{G}$. These P/T Petri games are safe, have only one recurrently interfering environment, and no mixed communication, and we call them *proper (P/T) Petri games.* $\lhd$

Notice that by definition, all proper high-level Petri games have a single initial marking M_0 instead of a set of initial markings (cp. p. 26).

Figure 3.4 shows four high-level Petri games, each violating one property of proper high-level Petri games. For all games, we assume $\mathit{Col} = \{0, 1, 2\}$. We again use 0 as a special number, and assume that every unlabeled edge is labeled implicitly by a variable v^0, with the guard of the corresponding transition implicitly containing the clause $v^0 = 0$. We do not give the specification (i.e., the objective of the system players) of the Petri games and only describe the respective violation, which always depends purely on the underlying structure.

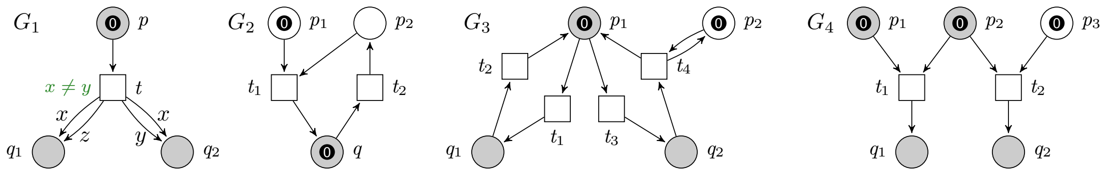


Figure 3.4: Four examples of high-level Petri games, each *violating* one property of *proper* high-level Petri games.

The high-level Petri game G_1 is not expansion safe: when t fires, the colors assigned to x and z are placed on q_1, and the colors assigned to x and y are placed on q_2. The guard of t ensures that two different colors are placed on q_2. However, when firing t, e.g., in mode $\{x \leftarrow 1, y \leftarrow 2, z \leftarrow 1\}$ the color 1 gets placed q_1 twice, resulting in the marking $\{\!|\, (q_1, 1), (q_1, 1), (q_2, 2) \,|\!\}$.

G_2 (while being expansion safe) has more than one environment player: firing t_2 results in the marking $\{\!|\, (p_1, 0), (p_2, 0) \,|\!\}$, with $p_1, p_2 \in P_E$.

The high-level Petri game G_3 is expansion safe and has only one environment player, but this environment player is not recurrently interfering: The infinite firing sequence

$$\begin{aligned} M_0 = &\{\!|\, (p_1, 0), (p_2, 0) \,|\!\} \quad [t_1, \{v^0 \leftarrow 0\}\rangle \quad \{\!|\, (q_1, 0), (p_2, 0) \,|\!\} \quad [t_2, \{v^0 \leftarrow 0\}\rangle \\ &\{\!|\, (p_1, 0), (p_2, 0) \,|\!\} \quad [t_1, \{v^0 \leftarrow 0\}\rangle \quad \{\!|\, (q_1, 0), (p_2, 0) \,|\!\} \quad [t_2, \{v^0 \leftarrow 0\}\rangle \quad \dots \end{aligned}$$

cycling the color 0 through the places p_1 and q_1, contains no transition with environment places in its preset.

Finally, G_4 has mixed communication, as transition t_1 only has system places in its preset and transition t_2 has both system places and environment places in its preset, and both share the system place p_2 in their preset.

After seeing examples for non-proper games, we now examine why the running example Signal Sending Satellites *is* a proper high-level game.

Example 3.7 (Proper high-level Petri game). The running example Signal Sending Satellites from Fig. 3.2 is a proper high-level Petri game:

- Transition *end* is the only one that places the color **0** on the *Env* place, which can only fire when there is a color on *Tr*. This means that there cannot be two copies of **0** on place *Env*.
- On every other place, there can be at most one copy of each satellite, either as an individual color or as the first component in a tuple. This follows from the structure of the high-level Petri net and the fact that transition *end* resets the initial marking. Thus, the example is *expansion safe*.
- Furthermore, these arguments imply that there is only *one environment*, which is *recurrently interfering* because every execution either ends or requires an instance of *sel* to fire infinitely often.
- As discussed in the note on p. 39, the example has *no mixed communication*. $\lhd$

3.1.3 Solving Place/Transition Petri Games

As already discussed above, in [FO14; FO17], it is shown that there is a winning strategy for the system players in a k-bounded P/T Petri game G with one environment player and a safety objective for the system players if and only if there is a winning strategy for Player 0 in a two-player game $\mathbb{G}(\mathsf{G})$ over a finite graph. The proof rests on a link between so-called *mcuts* ("maximal" cuts where every enabled event involves an environment player) in the strategy of the Petri game G and corresponding *environment-dependent decision sets* (vertices) in the two-player game $\mathbb{G}(\mathsf{G})$.

In [Gie22], Gieseking consolidates this work, and extends it for safe (i.e., 1-bounded) P/T Petri games without mixed communication to different objectives for the system players, namely: safety, reachability, Büchi, co-Büchi, and parity. Furthermore, each of these objectives is subdivided into an *existential* and a *universal* version, meaning that either *one* or *all* system players have to satisfy the objective. To express these objectives, the notion of *information flow* is introduced. In the terminology of [Gie22], the Petri games in [FO14; FO17] have a universal safety objective. The objectives of Petri games considered in this work (cp. Sec. 2.2) correspond to *universal safety* and *existential reachability*. To model these objectives, the notion of information flow is not needed.

The proper Petri games $\mathsf{G} \in \mathrm{Exp}(\mathbf{G})$ considered here are a specific instance of the Petri games studied in [FO17; Gie22]. The assumption of a recurrently interfering environment, and the restriction to the objectives universal safety and existential reachability, allow us to simplify the definition of $\mathbb{G}(\mathsf{G})$ from [Gie22]: due to the assumption of a recurrently interfering environment. we can omit the round-robin component from [Gie22] for handling "type-2 behavior" (infinite progress of the system without interaction with the environment), and a preprocess with the same purpose from [FO17].

Additionally, we directly integrate the automaton handling the different system objectives from [Gie22] into the two-player game. This is possible since we only consider universal safety and existential reachability, and realized by defining the winning condition in the two-player game $\mathbb{G}(\mathsf{G})$ depending on the system players' objective in the Petri game G.

We now start by briefly recalling the basics for two-player games with a safety or reachability condition for Player 0. We proceed by defining decision sets of a given Petri game G, and the relations between decision sets. These are then used to define the vertices and edges in (a simpler version of) the two-player game $\mathbb{G}(\mathsf{G})$ from [FO17; Gie22]. We conclude this section with a theorem stating there is a strategy for the system players in G if and only if there is a strategy for the Player 0 in $\mathbb{G}(\mathsf{G})$.

Preliminaries for two-player games. A *two-player game* $\mathbb{G} = (\mathbb{V}_0, \mathbb{V}_1, v_0, \mathbb{E}, \mathrm{Win})$ consists of the disjunct finite sets of *0-vertices* $\mathbb{V}_0$ and *1-vertices* $\mathbb{V}_1$ (and the set of all *vertices* is $\mathbb{V} = \mathbb{V}_0 \sqcup \mathbb{V}_1$), the *initial vertex* $v_0 \in \mathbb{V}$, the *edge relation* $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$, and *winning set* Win of plays in $\mathbb{G}$. A *play* in $\mathbb{G}$ is a possibly infinite sequence $\overline{v} = v_0 v_1 v_2 \cdots$ of states with $(v_j, v_{j+1}) \in E$ for all $j \in \mathbb{N}$. We assume that every vertex has an outgoing edge, i.e., $\forall v \in \mathbb{V} : v\mathbb{E} \neq \emptyset$. The game is played between two players, namely Player 0 and Player 1, with the respective set of i-vertices belonging to Player i. A *strategy* for Player i in $\mathbb{G}$ is a function $f : \mathbb{V}^*\mathbb{V}_i \to \mathbb{V}$ which maps each sequence of states ending in a state of Player i to some successor state, satisfying $(v, f(wv)) \in E$ for all $w \in \mathbb{V}^*$ and $v \in \mathbb{V}_i$. Such a strategy is called *positional*, iff $f(wv) = f(v)$ for all $w \in \mathbb{V}^*$ and $v \in \mathbb{V}_i$. Player 0 *wins* a play $\overline{v}$ iff $\overline{v} \in \mathrm{Win}$. Otherwise, Player 1 wins $\overline{v}$. A *play* $\overline{v}$ *conforms* to a strategy f for Player i iff for all prefixes $wvv' \in \mathbb{V}^*\mathbb{V}_i\mathbb{V}$ of $\overline{v}$ the play satisfies $f(wv) = v'$. A strategy f for Player i is *winning* iff every play which conforms to f is won by Player i.

For the solving algorithms for Petri games we are especially interested in so-called two-player *safety games* and two-player *reachability games* games, which are two-player games where the winning set is given by $\mathrm{Win} = \mathrm{Safety}(\mathbb{F})$ (resp. $\mathrm{Win} = \mathrm{Reach}(\mathbb{F})$) for sets of *special vertices* $\mathbb{F} \subseteq \mathbb{V}$ (also called *bad vertices* resp. *target vertices*). A play $\overline{v} = v_0 v_1 v_2 \cdots$ in such a two-player safety (resp. reachability) game is an element of $\mathrm{Win} = \mathrm{Safety}(\mathbb{F})$ (resp. $\mathrm{Win} = \mathrm{Reach}(\mathbb{F})$) if and only if $\forall i \in \mathbb{N} : v_i \notin \mathbb{F}$ (resp. $\exists i \in \mathbb{N} : v_i \in \mathbb{F}$).

From game theory we know that in a two-player safety/reachability game $\mathbb{G}$, Player 0 has a winning strategy iff Player 0 has a *positional* winning strategy. Deciding the question whether Player 0 has a winning strategy and, if possible, generating a winning positional strategy, can be achieved in linear time in the number of edges in the game (cp., e.g., [Maz01]). This process is called *solving* the two-player game. $\lhd$

Remark. In the context of this thesis, we represent runs in a given Petri game by plays in a corresponding two-player game. In this two-player game, the vertices represent states in the Petri game, and the edge relation partly is build from firing transitions. From a possible winning strategy for Player 0 we can then construct a winning strategy for the system players in the Petri game. For that, we need to know the transitions whose firings are simulated in the two-player game strategy. To access this information directly, we use a *labeled edge relation* $\mathbb{E} \subseteq \mathbb{V} \times \Lambda \times \mathbb{V}$ for a set Λ of labels. For example, in the case of P/T Petri games, this set is given by $\Lambda = \mathsf{T} \cup \{\dagger\}$, with a special symbol $\dagger$ that gets explained later. Having a labeled edge relation does not change any of the results above, and we can ignore the labels when we do not need them. In these cases we can write $(v, v') \in \mathbb{E}$, omitting an edge's label.

The solving algorithm for Petri games presented in [FO17; Gie22] executes the following (abstract) steps for a given P/T Petri game G:

- Construct a corresponding two-player game $\mathbb{G}(\mathsf{G})$.
- Solve the two-player game $\mathbb{G}(\mathsf{G})$;
  - If Player 0 has no winning strategy in $\mathbb{G}(\mathsf{G})$, then the system players have no winning strategy in G.
  - If Player 0 has a winning strategy f in $\mathbb{G}(\mathsf{G})$, then construct a corresponding winning strategy $\xi(f)$ for the system players in G.

We give a simplified definition of $\mathbb{G}(\mathsf{G})$ for the class of proper P/T Petri games $\mathsf{G} = (\mathsf{P}_\mathrm{S}, \mathsf{P}_\mathrm{E}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Obj}, \mathsf{P}_{\circledast}) \in \mathrm{Exp}(\mathbf{G})$. These Petri games are safe, have a single recurrently interfering environment, and no mixed communication. For the remainder of this section, let G be such a proper P/T Petri game.

The key idea of the reduction to a two-player game is to delay the environment's moves until no system player can proceed any further. By this, we ensure that all system players get informed of the environment's moves during their next move and all system player's commitments, which should be made independently are made before the environment's possible choice. This allows for applying solving algorithms for games with *complete information* to $\mathbb{G}(\mathsf{G})$.

In $\mathbb{G}(\mathsf{G})$, plays in G are simulated through a sequence of *decision sets*, i.e., enriched markings of G. In a decision set, each player (token) is equipped with a *commitment set*, i.e., a set of transitions which are currently selected by the player to be allowed to fire, or it is equipped with the special symbol $\dagger$. In the case of a $\dagger$-symbol, a system player has yet to select a commitment set of transitions.[1] No environment player is ever equipped with a $\dagger$-symbol. Instead, their commitment set always is their whole postset. This represents the unrestricted, nondeterministic choices of the environment.

[1]In [FO14; FO17; Gie22], instead of $\dagger$, the symbol $\top$ is used. We use $\dagger$ to avoid confusion with the set of transitions T.

Definition 3.8 (Decision set). A *decision set* in G is a set $\mathsf{D} \subseteq \mathsf{P} \times (\mathbb{P}(\mathsf{T}) \cup \{\dagger\})$ of pairs $(\mathsf{p}, \dagger)$ or $(\mathsf{p}, \mathcal{T})$, where $\mathcal{T} \in \mathbb{P}(\mathsf{T})$ is called a *commitment set*, s.t.

- in a commitment set only transitions of the respective place's postset occur, i.e., $(\mathsf{p}, \mathcal{T}) \in \mathsf{D} \wedge \mathcal{T} \subseteq \mathsf{T} \Rightarrow \forall \mathsf{t} \in \mathcal{T} : \mathsf{t} \in \mathit{post}(\mathsf{p})$, and
- a player on an *environment place* is always equipped with its whole postset as its commitment set, i.e., $(\mathsf{p}, \mathcal{T}) \in \mathsf{D} \wedge \mathsf{p} \in \mathsf{P}_{\mathrm{E}} \Rightarrow \mathcal{T} = \mathit{post}(\mathsf{p})$.

We denote the set of all decision sets by $\mathcal{D}(\mathsf{G})$, and define, for every decision set $\mathsf{D} \in \mathcal{D}(\mathsf{G})$ its *corresponding marking* $\mathsf{M}(\mathsf{D}) := \{\mathsf{p} \mid \exists \mathcal{T} \in (\mathbb{P}(\mathsf{T}) \cup \{\dagger\}) : (\mathsf{p}, \mathcal{T}) \in \mathsf{D}\}$. $\lhd$

A transition $\mathsf{t} \in \mathsf{T}$ is *enabled* in a decision set D if $\mathit{pre}(\mathsf{t}) \subseteq \mathsf{M}(\mathsf{D})$. The transition t is *chosen* in D if $\forall (\mathsf{p}, \mathcal{T}) \in \mathsf{D} : \mathsf{p} \in \mathit{pre}(\mathsf{t}) \Rightarrow \mathsf{t} \in \mathcal{T}$ holds. We call the transition t *fireable* in D, denoted by $\mathsf{D}[\mathsf{t}\rangle$, if t is enabled and chosen in D. This means that the transition t not only needs to be enabled in the corresponding marking, but also that all players in the transition's preset must allow t in their commitment set. The decision set D' obtained after *firing* t, denoted by $\mathsf{D}[\mathsf{t}\rangle\mathsf{D}'$, is given by

$$\begin{aligned}
\mathsf{D}' &= \{(\mathsf{p}, \mathcal{T}) \mid (\mathsf{p}, \mathcal{T}) \in \mathsf{D} \wedge \mathsf{p} \notin \mathit{pre}(\mathsf{t})\} && \text{(the players unaffected by } \mathsf{t} \text{ stay)}\\
&\cup \{(\mathsf{p}, \dagger) \mid \mathsf{p} \in \mathit{post}(\mathsf{t}) \cap \mathsf{P}_{\mathrm{S}}\} && \text{(}\dagger\text{-symbol for sys. players placed by } \mathsf{t}\text{)}\\
&\cup \{(\mathsf{p}, \mathit{post}(\mathsf{p})) \mid \mathsf{p} \in \mathit{post}(\mathsf{t}) \cap \mathsf{P}_{\mathrm{E}}\} && \text{(whole postset for env. players placed by } \mathsf{t}\text{)}
\end{aligned}$$

This means that the corresponding markings preserve the firing relation, i.e., $\mathsf{D}[\mathsf{t}\rangle\mathsf{D}' \Rightarrow \mathsf{M}(\mathsf{D})[\mathsf{t}\rangle\mathsf{M}(\mathsf{D}')$, and only the moved system players are allowed to (and must) decide on a new commitment set. The moved environment players are equipped with their whole postset as commitment set.

If a decision set D contains a $\dagger$-symbol, i.e., $\exists (\mathsf{p}, \dagger) \in \mathsf{D}$, denoted by $\mathsf{D}[\dagger\rangle$, the corresponding system players have to decide on a new commitment set before any other move is allowed in the two-player game. Such a decision is denoted by $\mathsf{D}[\dagger\rangle\mathsf{D}'$ where D' is a decision set such that

$$\mathsf{D}' = \{(\mathsf{p}, \mathcal{T}) \mid (\mathsf{p}, \mathcal{T}) \in \mathsf{D} \wedge \mathcal{T} \neq \dagger\} \cup \{(\mathsf{p}, \mathsf{d}(\mathsf{p})) \mid (\mathsf{p}, \dagger) \in \mathsf{D}\}$$

for a function $\mathsf{d} : \{\mathsf{p} \mid (\mathsf{p}, \dagger) \in \mathsf{D}\} \rightarrow \mathbb{P}(\mathsf{T})$. We call this relation $\dagger$-*resolution*. The definition means that the $\dagger$-resolution of a decision set D yields one successor decision set for *every possible combination* of replacing each $\dagger$ in D with a respective commitment set $\mathcal{T} \subseteq \mathsf{T}$. Each possibility is realized by a function d that sets a commitment set for every place in D that is equipped with a $\dagger$-symbol.

Example 3.9 (Relations between Decision sets). Figure 3.5 shows three decision sets D_0, D_1, and D_2 in the proper Petri game Signal Sending Satellites from Fig. 3.3, here called G. The decision set D_0 will later constitute the *initial vertex* in the two-player game $\mathbb{G}(\mathsf{G})$. It is constructed from the initial marking $\{\mathit{Env}.\mathbf{0}, \mathit{Init}.\mathbf{1}, \mathit{Init}.\mathbf{2}, \mathit{Init}.\mathbf{3}\}$. The commitment set of the environment player on $\mathit{Env}.\mathbf{0}$ is its entire postset, while the system players are equipped with a $\dagger$-symbol, meaning they have yet to decide for a commitment

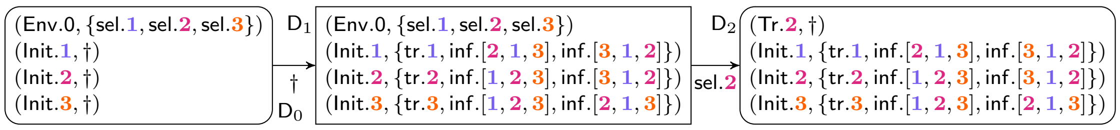

(Env.0, {sel.1, sel.2, sel.3})
(Init.1, †)
(Init.2, †)
(Init.3, †)
D₁
†
D₀
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
D₂
sel.2
(Tr.2, †)
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})

Figure 3.5: A sequence of two relations between three decision sets in the running example Signal Sending Satellites.

set. The different borders (rounded/sharp) indicate whether a vertex in the game $\mathbb{G}(\mathsf{G})$ belongs to Player 0 or Player 1. We will discuss this further down the line, and it is currently irrelevant.

Since D_0 contains a †-symbol, we can (and later must) resolve it, i.e., there is a relation $\mathsf{D}_0[\dagger\rangle\mathsf{D}'$ for every D' where the †-symbol of each system player is replaced by a subset of its postset. In particular, there is a relation $\mathsf{D}_0[\dagger\rangle\mathsf{D}_1$. In D_1, every system player on place $\mathit{Init}.\widetilde{x}$ allows transition $\mathit{tr}.\widetilde{x}$, as well as the two transitions $\mathit{inf}.[\widetilde{x},\widetilde{y},\widetilde{z}]$ with $\widetilde{x}=\widetilde{y}$ or $\widetilde{x}=\widetilde{z}$, and $\widetilde{y}<\widetilde{z}$.

However, none of the instances of *tr* or *inf* can fire from D_1. In fact, the only transitions fireable from D_1, namely *sel*.**1**, *sel*.**2**, and *sel*.**3**, are already fireable from D_0. Firing *sel*.**2** from D_1 leads to the decision set D_2 (relation $\mathsf{D}_1[\mathit{sel}.\mathbf{2}\rangle\mathsf{D}_2$). The tokens are moved corresponding to firing the transition from the decision set's marking, which in this case means that the token on *Env*.**0** is consumed, and a token is placed on the place *Tr*.**2**. Since this is a system place, the player gets equipped with a †-symbol. This means $\mathsf{D}_2[\dagger\rangle$. In the two-player game $\mathbb{G}(\mathsf{G})$, the next step then has to be another †-resolution. $\lhd$

In the two-player game $\mathbb{G}(\mathsf{G})$ for a Petri game G that we want to define, the edges are built from relations between decision set, depending on the source decision set's properties. To that aim we now define the possible properties a decision set $\mathsf{D}\in\mathcal{D}(\mathsf{G})$ can have:

- D is *environment-dependent* iff $\neg\mathsf{D}[\dagger\rangle$, and $\exists \mathsf{p}\in\mathsf{P}_\mathrm{E}\cap\mathsf{M}(\mathsf{D})\,\forall\mathsf{t}\in\mathsf{T}:\mathsf{D}[\mathsf{t}\rangle\Rightarrow\mathsf{p}\in\mathit{pre}(\mathsf{t})$. This means all next moves of the system players are fixed (there is no †-symbol in D) and each of these moves is only possible after a progress of the environment.
- D *contains a bad place resp. target place* iff $\mathsf{M}(\mathsf{D})\cap\mathsf{P}_\spadesuit\neq\emptyset$ resp. $\mathsf{M}(\mathsf{D})\cap\mathsf{P}_\heartsuit\neq\emptyset$.
- D is a *deadlock* iff $\forall\mathsf{t}\in\mathsf{T}:\neg\mathsf{D}[\mathsf{t}\rangle$ but $\exists\mathsf{t}\in\mathsf{T}:\mathsf{M}(\mathsf{D})[\mathsf{t}\rangle$. This means the corresponding marking enables a transition, but no enabled transition is chosen in D.
- D is *terminating* iff $\neg\mathsf{M}(\mathsf{D})[\mathsf{t}\rangle$ holds for all transitions $\mathsf{t}\in\mathsf{T}$,
- D is *nondeterministic* iff $\exists\mathsf{t}_1,\mathsf{t}_2\in\mathsf{T}:\mathsf{t}_1\neq\mathsf{t}_2\wedge\mathsf{P}_\mathrm{S}\cap\mathit{pre}(\mathsf{t}_1)\cap\mathit{pre}(\mathsf{t}_2)\neq\emptyset\wedge\mathsf{D}[\mathsf{t}_1\rangle\wedge\mathsf{D}[\mathsf{t}_2\rangle$. This means there are two separate transitions that share a system place in their presets and are both fireable in D.

These properties are demonstrated in Example 3.11, after the definition of the two-player game $\mathbb{G}(\mathsf{G})$.

We now have the components to define, for a proper Petri game G, the corresponding two-player game $\mathbb{G}(\mathsf{G})$. The decision sets of G form the vertices. The edges in $\mathbb{G}(\mathsf{G})$ are defined from relations between the decision sets, i.e., †-resolution or firing of transitions. Paths through the graphs simulate runs in G. Which relations between two decision sets are considered for the edges depends on the properties of the source decision set. By this, it is ensured that the environments moves in a simulated run in G are delayed until the system cannot proceed anymore on its own. Note that an mcut in a Petri game strategy corresponds to an environment-dependent decision set: each system player gets informed of the environment's progress in its next move. The winning condition for Player 0 in $\mathbb{G}(\mathsf{G})$ depends on the objective of system players in G.

Definition 3.10 (Two-player game $\mathbb{G}(\mathsf{G})$). Let $\mathsf{G} = (\mathsf{P_S}, \mathsf{P_E}, \mathsf{T}, \mathsf{F}, \mathsf{M_0}, \mathrm{Obj}, \mathsf{P}_{\circledast}) \in \mathrm{Exp}(\mathbf{G})$ be a proper P/T Petri game. The components of the *two-player game over a finite graph* $\mathbb{G}(\mathsf{G}) = (\mathbb{V}_0, \mathbb{V}_1, v_0, \mathbb{E}, \mathrm{Win})$ are defined as follows:

- The 0-vertices $\mathbb{V}_0 \subseteq \mathcal{D}(\mathsf{G})$ are all decision sets that are *not* environment-dependent.
- The 1-vertices $\mathbb{V}_1 := \mathcal{D}(\mathsf{G}) \setminus \mathbb{V}_0$ are all environment-dependent decision sets in G.
- The initial vertex $\mathsf{D}_0 := \{(\mathsf{p}, \dagger) \mid \mathsf{p} \in \mathsf{M}_0 \cap \mathsf{P_S}\} \cup \{(\mathsf{p}, \mathit{post}(\mathsf{p})) \mid \mathsf{p} \in \mathsf{M}_0 \cap \mathsf{P_E}\}$ is the decision set containing all places of the initial marking s.t. the system players still have to decide for a commitment set.
- The *labeled edge relation* $\mathbb{E} \subseteq \mathbb{V} \times (\mathsf{T} \cup \{\dagger\}) \times \mathbb{V}$ is defined as follows:

  If D is a deadlock, is terminating, or is nondeterministic, there is only a †-labeled self-loop originating from D. Otherwise, we consider three disjunct cases for edges originating in D:

  Case $\mathsf{D} \in \mathbb{V}_1$; i.e., all players have decided for a commitment set, but cannot proceed without the environment. Then for all $\mathsf{t} \in \mathsf{T}$, $(\mathsf{D}, \mathsf{t}, \mathsf{D}') \in \mathbb{E}$ iff $\mathsf{D}[\mathsf{t}\rangle\mathsf{D}'$.

  Case $\mathsf{D} \in \mathbb{V}_0$ *and* $\mathsf{D}[\dagger\rangle$; i.e., at least one system player has yet to decide for a commitment set. Then $(\mathsf{D}, \dagger, \mathsf{D}') \in \mathbb{E}$ iff $\mathsf{D}[\dagger\rangle\mathsf{D}'$.

  Case $\mathsf{D} \in \mathbb{V}_0$ *and* $\neg\mathsf{D}[\dagger\rangle$; i.e., all system players made their decisions and can proceed without the environment. Then for all $\mathsf{t} \in \mathsf{T}$ with $\mathit{pre}(\mathsf{t}) \cap \mathit{P}_\mathrm{E} = \emptyset$, $(\mathsf{D}, \mathsf{t}, \mathsf{D}') \in \mathbb{E}$ iff $\mathsf{D}[\mathsf{t}\rangle\mathsf{D}'$. The condition for $\mathit{pre}(\mathsf{t})$ ensures that only edges for system transitions are considered.
- Win depending on Obj and $\mathsf{P}_{\circledast}$ as follows:
  - if Obj = Safety then Win := Safety($\mathbb{F}$), where $\mathbb{F}$ contains all decision sets that are a deadlock, nondeterministic, or contain a bad place,
  - if Obj = Reach then Win := Reach($\mathbb{F}$), where $\mathbb{F}$ contains all decision sets that contain a target place.

We denote $\mathbb{V} := \mathbb{V}_0 \sqcup \mathbb{V}_1 = \mathcal{D}(\mathsf{G})$. ◁

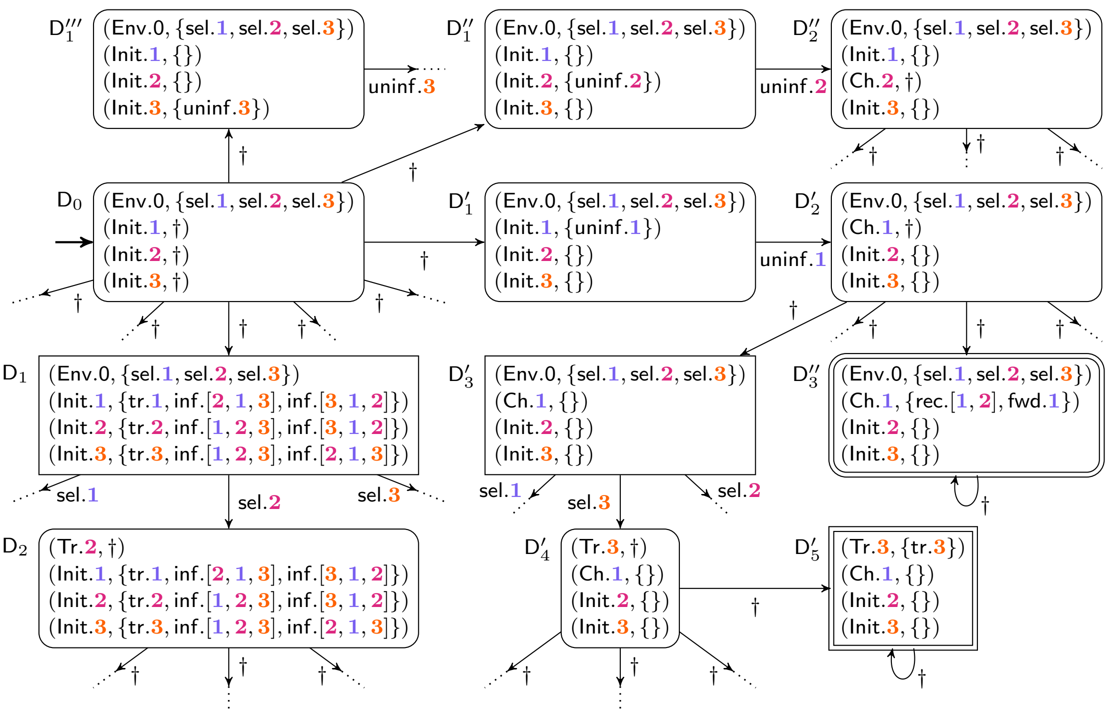

D‴1
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, {})
(Init.2, {})
(Init.3, {uninf.3})
uninf.3
D″1
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, {})
(Init.2, {uninf.2})
(Init.3, {})
uninf.2
D″2
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, {})
(Ch.2, †)
(Init.3, {})
D0
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, †)
(Init.2, †)
(Init.3, †)
D′1
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, {uninf.1})
(Init.2, {})
(Init.3, {})
uninf.1
D′2
(Env.0, {sel.1, sel.2, sel.3})
(Ch.1, †)
(Init.2, {})
(Init.3, {})
D1
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
D′3
(Env.0, {sel.1, sel.2, sel.3})
(Ch.1, {})
(Init.2, {})
(Init.3, {})
D″3
(Env.0, {sel.1, sel.2, sel.3})
(Ch.1, {rec.[1, 2], fwd.1})
(Init.2, {})
(Init.3, {})
sel.1
sel.2
sel.3
D2
(Tr.2, †)
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
D′4
(Tr.3, †)
(Ch.1, {})
(Init.2, {})
(Init.3, {})
D′5
(Tr.3, {tr.3})
(Ch.1, {})
(Init.2, {})
(Init.3, {})
†

Figure 3.6: Part of the two-player game for the running example Signal Sending Satellites.

Example 3.11 (Two-player game for Signal Sending Satellites). Consider the running example Signal Sending Satellites from Fig. 3.3. Denote this Petri game by G. Fig. 3.6 shows part of the two-player game $\mathbb{G}(\mathsf{G})$.

The initial decision set in $\mathbb{G}(\mathsf{G})$ is D_0. It is constructed from the initial marking $\{Env.\mathbf{0}, Init.\mathbf{1}, Init.\mathbf{2}, Init.\mathbf{3}\}$. The commitment set of the environment player on $Env.\mathbf{0}$ is its entire postset, while the system players are equipped with a †-symbol, meaning they have to decide for a commitment set in the next step. We abbreviate $select.\{x \leftarrow \widetilde{x}\}$ by $sel.\widetilde{x}$ for every $\widetilde{x} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$, and analogously for all other transitions. Since D_0 contains a †-symbol, it belongs to $\mathbb{V}_0$. In figures, this is illustrated by rounded corners of a vertex, whereas decision sets belonging to $\mathbb{V}_1$ have sharp corners. The system now has several options to resolve the †-symbol, from which we investigate two in particular – the ones leading to D_1 and D_1'.

In D_1, every system player on a place $Init.\widetilde{x}$ decided to allow transitions where it gains information: either from a player on $Tr.\widetilde{x}$ (via $tr.\widetilde{x}$), or from another system player of color $\overline{x}$ (transitions $inf.\{x \leftarrow \overline{x}, y \leftarrow \widetilde{y}, z \leftarrow \widetilde{z}\}$ with $\widetilde{y} = \widetilde{x}$ or $\widetilde{z} = \widetilde{x}$, abbreviated by $inf.[\overline{x}, \widetilde{y}, \widetilde{z}]$). To avoid nondeterminism, the system players do not allow *all* instances of *inf* satisfying the constraints above. We go more into detail on that in Example 3.13.

Since D_1 contains no †-symbol, and every fireable transition (i.e., all instances of *sel*) has an environment place in its preset, D_1 belongs to $\mathbb{V}_1$. We thus have an edge to every decision set reached after firing one of these transitions. For example, the decision set D_2 is reached from D_1 by firing $sel.\mathbf{2}$. Correspondingly, a system player is placed on $Tr.\mathbf{2}$, and equipped with a †-symbol. This †-symbol has to be resolved in the next step.

Consider now the †-resolution leading from D_0 to D'_1. Here, the system player on *Init*.1 decided to allow only transition *uninf*.1, while the system players on *Init*.2 and *Init*.3 disallowed all transitions in their postset to fire, which is represented by an empty commitment set. Since now *uninf*.1 is fireable and has no environment places in its preset, D'_1 belongs to $\mathbb{V}_0$, and its only outgoing edge corresponds to firing *uninf*.1, leading to D'_2.

In D'_2, the system player moved to place *Ch*.1 now again has to decide for a commitment set in the next step. If it decides to allow no transition in its postset to fire (†-resolution to D'_3), no transitions except the instances of *sel* can fire. Consider the firing of *sel*.3 leading to D'_4. If the system player placed on *Tr*.3 decides to only allow $\{tr.\mathbf{3}\}$ then we see in D'_5 that *tr*.3 is enabled, i.e., $\mathsf{M}(\mathsf{D}'_5)[tr.\mathbf{3}\rangle$, but not chosen (and therefore not fireable) since the player on *Init*.3 forbids the transition. This means D'_5 is a *deadlock*, and only hast an outgoing self-loop labeled with †. This vertex therefore is *bad* w.r.t. the safety condition, which we illustrate by a double border.

If from D'_2 the system player on *Ch*.1 decides to allow both transitions $rec.\{y \leftarrow \mathbf{1}, x \leftarrow \mathbf{2}\}$ and *fwd*.1, which leads to D''_3, both these transitions can fire and have the same system place in their preset. This makes D''_3 nondeterministic (also a bad vertex), which means the only outgoing edge is a self-loop.

The parts "after" the decision sets D''_1 and D'''_1 each are completely symmetric to the part after D_1 that was just partly discussed. They represent the cases where the system player on *Init*.2 resp. *Init*.3 decides to allow the corresponding instance of *uninf*, while the other two system players decide to allow no transitions.

Signal Sending Satellites is a Petri game with *safety objective*, which means in the two-player game we have $\mathrm{Win} = \mathrm{Safety}(\mathbb{F})$, where $\mathbb{F}$ contains all decision sets in $\mathbb{V}$ that are a deadlock, nondeterministic, or contain a bad place. We already marked the decision sets D'_5 and D''_3 by double borders, meaning they are in the set of bad vertices $\mathbb{F}$. Every decision set D where $\mathsf{M}(\mathsf{D})$ contains a place $Bad.\widetilde{x}$ would also be marked as a bad vertex. ◁

In [FO17] and [Gie22], more complex versions of the two-player game are presented, which cater for k-bounded Petri games and different system objectives, respectively. Additionally, both consider the possibility of so-called type-2 behavior, where the system can proceed infinitely without the environment. By assuming the Petri games to be safe and eliminating type-2 behavior, we can, for Petri games G with a safety or reachability objective, simplify $\mathbb{G}(\mathsf{G})$ to its core idea: delaying the environment's moves until the system cannot proceed without it. This idea allows us to assume complete information for the system players.

Compared to [FO17; Gie22], we therefore presented a simplified version of $\mathbb{G}(\mathsf{G})$ for the class of proper Petri games $\mathsf{G} \in \mathrm{Exp}(\mathbf{G})$. Thus, to state the following theorem, we have to show that the simplification of $\mathbb{G}(\mathsf{G})$ is corresponding to this class.

Theorem 3.12 ([FO17; Gie22]). *In a proper Petri game $\mathsf{G} \in \mathrm{Exp}(\mathbf{G})$, the system players have a winning strategy if and only if Player 0 has a winning strategy in $\mathbb{G}(\mathsf{G})$.*

For the sake of completeness, the rather lengthy proof of this theorem is provided in Appendix 3.A. The proof is primarily adapted from [Gie22] but modified to fit our definition of the two-player game.

In the proof of the respective analogue of Theorem 3.12 in [FO17; Gie22], it is demonstrated how to construct a winning strategy ξ for the system players in G based on a winning strategy f for Player 0 in $\mathbb{G}(\mathsf{G})$. The detailed algorithm is not provided here, but it is part of the proof of Theorem 3.12 in Appendix 3.A. The main idea is to attach an event with label t to the Petri game strategy ξ under construction whenever there is an edge $(\mathsf{D}, \mathsf{t}, \mathsf{D}')$ taken in the two-player game strategy f. We informally show this on an example:

Example 3.13 (Strategy translation). Consider the running example Signal Sending Satellites from Fig. 3.3. Denote this Petri game by G. In Fig. 3.7, we see on the left side a part of the winning strategy for Player 0 in the two-player game $\mathbb{G}(\mathsf{G})$, depicted as a tree. The algorithm now traverses the strategy tree in breadth first order. Depending on the edges taken, the strategy for the system players in G is built by successively attaching new events and conditions, starting with the initial cut of the unfolding.

The initial decision set in $\mathbb{G}(\mathsf{G})$ (and therefore contained in the strategy) is D_0. It is given by the initial marking $\{\mathit{Env}.\mathbf{0}, \mathit{Init}.\mathbf{1}, \mathit{Init}.\mathbf{2}, \mathit{Init}.\mathbf{3}\}$. The commitment set of the environment player on $\mathit{Env}.\mathbf{0}$ is its entire postset, while the system players are equipped with a †-symbol, meaning they have to decide for a commitment set in the next step. We abbreviate $\mathit{select}.\{x \leftarrow \widetilde{x}\}$ by $\mathit{sel}.\widetilde{x}$ for every $\widetilde{x} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$, and analogously for all other transitions.

From the initial decision set, we start constructing the strategy in G with the *initial cut* corresponding to the marking $\mathsf{M}(\mathsf{D})$. This correspondence is highlighted in gray. We say this cut is *associated* to the decision set D_0 in the strategy in $\mathbb{G}(\mathsf{G})$.

D_0 belongs to $\mathbb{V}_0$ since it contains a †-symbol. This means there is exactly one successor of D_0 in the strategy tree, which corresponds to replacing the †-symbol with commitment sets. As we have already described, no satellite should make an uninformed choice between Forwarding mode and Receiving mode. This means that a system player on a place $\mathit{Init}.\widetilde{x}$ should *not* allow the transition $\mathit{uninf}.\widetilde{x}$ to fire. Instead, the player should allow to be informed that itself must forward a transmission (via transition $\mathit{tr}.\widetilde{x}$), and to be informed by another satellite $\overline{x}$ that it should receive a message. This is done via a transition $\mathit{inf}.\{x \leftarrow \overline{x}, y \leftarrow \widetilde{y}, z \leftarrow \widetilde{z}\}$ (abbreviated by $\mathit{inf}.[\overline{x}, \widetilde{y}, \widetilde{z}]$) with $\widetilde{y} = \widetilde{x}$ or $\widetilde{z} = \widetilde{x}$.

For every $\overline{x}$, there are two instances of inf that satisfy the conditions above. For example, a system player on place $\mathit{Init}.\mathbf{1}$ can be informed that Satellite $\mathbf{2}$ must forward a message by $\mathit{inf}.[\mathbf{2}, \mathbf{1}, \mathbf{3}]$ and by $\mathit{inf}.[\mathbf{2}, \mathbf{3}, \mathbf{1}]$. However, due to the needed determinism, reaching a decision set where both these transitions are allowed would result in the system to lose. This is prevented by each system player on place $\mathit{Init}.\widetilde{x}$ allowing only the transitions $\mathit{inf}.[\overline{x}, \widetilde{y}, \widetilde{z}]$ with $\widetilde{y} = \widetilde{x}$ or $\widetilde{z} = \widetilde{x}$, and $\widetilde{y} < \widetilde{z}$. Fixing these commitment sets is done in the †-resolution leading to D_1.

During a †-resolution, the reached decision set is associated the same cut as the source decision set. This means D_1 has the same associated cut as D_0. Since D_1 contains

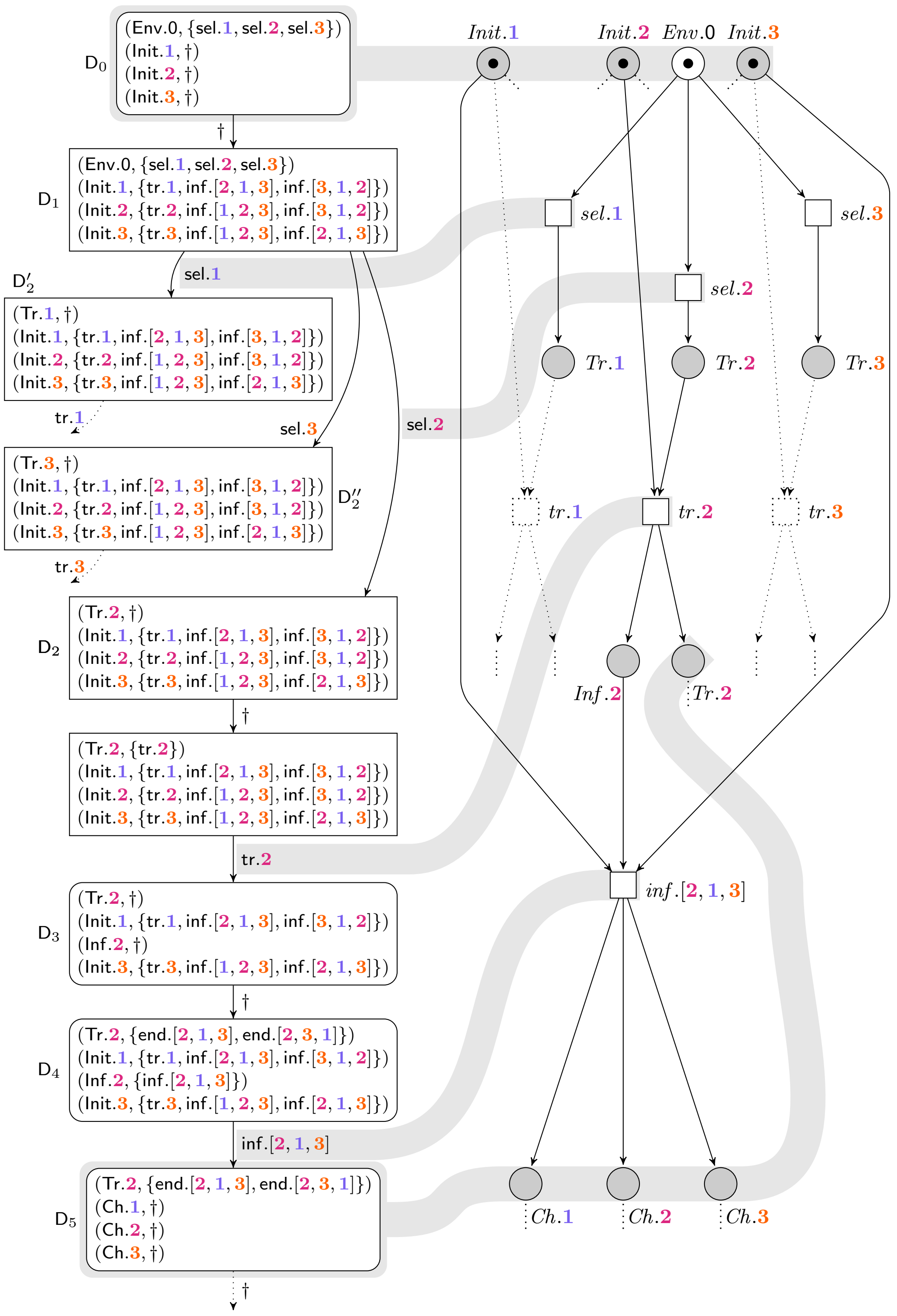

D₀
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, †)
(Init.2, †)
(Init.3, †)
†
D₁
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
sel.1
D′₂
(Tr.1, †)
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
tr.1
sel.3
sel.2
(Tr.3, †)
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
D″₂
tr.3
D₂
(Tr.2, †)
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
†
(Tr.2, {tr.2})
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
tr.2
D₃
(Tr.2, †)
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Inf.2, †)
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
†
D₄
(Tr.2, {end.[2, 1, 3], end.[2, 3, 1]})
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Inf.2, {inf.[2, 1, 3]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
inf.[2, 1, 3]
D₅
(Tr.2, {end.[2, 1, 3], end.[2, 3, 1]})
(Ch.1, †)
(Ch.2, †)
(Ch.3, †)
†
Init.1
Init.2
Env.0
Init.3
sel.1
sel.3
sel.2
Tr.1
Tr.2
Tr.3
tr.1
tr.2
tr.3
Inf.2
Tr.2
inf.[2, 1, 3]
Ch.1
Ch.2
Ch.3

Figure 3.7: Translating a strategy in $\mathbb{G}(\mathsf{G})$ to a strategy in G.

no †-symbol, and all fireable transitions ($\mathit{sel}.\mathbf{1}$, $\mathit{sel}.\mathbf{2}$, $\mathit{sel}.\mathbf{3}$) have the environment place $\mathit{Env}.\mathbf{0}$ in their preset, we have $\mathsf{D}_1 \in \mathbb{V}_1$. This means all possible successors in $\mathbb{G}(\mathsf{G})$ must also be in the strategy. The successors D_2, D_2', and D_2'', are each reached by firing a transition $\mathit{sel}.\widetilde{x}$.

When taking an edge corresponding to firing a transition in the strategy tree, the algorithm attaches an event with the respective label to the Petri game strategy under construction. From the cut associated to the source decision set, the conditions that have the transitions preset as labels are collected. These form the preset of the newly added event. Additionally, new conditions with the transitions postset as label are added to the strategy, and form the events postset. In our example, this means that when taking an edge corresponding to firing $\mathit{sel}.\widetilde{x}$, an event with label $\mathit{sel}.\widetilde{x}$ is added to the strategy, with the preset given by the condition with label $\mathit{Env}.\mathbf{0}$, and the postset consisting of a newly created condition with label $\mathit{Tr}.\widetilde{x}$. This is also highlighted in gray.

The decision set reached by taking the edge is then associated to the cut that is reached by firing the newly added event from the cut associated to the source decision set. For example, the cut of conditions with labels $\{\mathit{Tr}.\mathbf{2}, \mathit{Init}.\mathbf{1}, \mathit{Init}.\mathbf{2}, \mathit{Init}.\mathbf{3}\}$ (the upper condition with label $\mathit{Tr}.\mathbf{2}$) is associated to D_2.

The algorithm now further traverses the strategy tree in breadth first order. We consider only the future of D_2. The two cases of D_2' and D_2'' are symmetric, and indicated by the dotted parts of the two strategies. D_2 again is a decision set of Player 1, since the only fireable transition is $\mathit{tr}.\mathbf{2}$. After $\mathit{tr}.\mathbf{2}$, the system player on $\mathit{Inf}.\mathbf{2}$ in D_3 must again decide for a commitment set and chooses $\{\mathit{inf}.[\mathbf{2}, \mathbf{1}, \mathbf{3}]\}$ in the †-resolution leading to D_4. From there, the transition $\mathit{inf}.[\mathbf{2}, \mathbf{1}, \mathbf{3}]$ is the only fireable transition. The cut associated to the decision set D_5 (reached by this firing) is again highlighted in gray.

This part is enough to give the idea how the algorithm translates a strategy for Player 0 in $\mathbb{G}(\mathsf{G})$ to a strategy for the system players in G, so we end the example here. $\triangleleft$

3.2 The Symbolic Two-Player Game

In this section we show how to solve proper high-level Petri games G (i.e., high-level Petri games that are expansion safe, have a single recurrently interfering environment player, and no mixed communication, cp. Sec. 3.1.2) while exploiting the symmetries of the system. The key idea of the approach is to combine the two concepts established above: we apply *symmetries* in the net $N(G)$ (cp. Sec. 3.1.1) to the vertices in the *two-player game* $\mathbb{G}(\mathrm{Exp}(G))$ which serves for solving a proper P/T Petri game (cp. Sec. 3.1.3). The used techniques are borrowed from the construction of a "symbolic reachability graph" for high-level Petri Nets (for example presented in [CDFH97]) with a significantly smaller state space than the original reachability graph.

Correspondingly, our approach arrives at a significantly smaller two-player game, called the *symbolic two-player game* $\overline{\mathbb{G}}(G)$. The vertices in $\overline{\mathbb{G}}(G)$ are equivalence classes with respect to the symmetries of the high-level Petri game. The elements of these equivalence classes are the vertices of the two-player game $\mathbb{G}(\mathrm{Exp}(G))$ that the high-level Petri game's expansion can be reduced to. In particular, we show that the two two-player games are *bisimilar*.

Given a proper high-level Petri game $G \in \mathbf{G}$, the solving algorithm proceeds in four steps:

1. The corresponding symbolic two-player game $\overline{\mathbb{G}}(G)$ (Def. 3.26 on p. 68) is created with similar techniques as for the two-player game $\mathbb{G}(\mathsf{G})$ described in Sec. 3.1.3 for a low-level Petri game G. Moreover, the states of $\overline{\mathbb{G}}(G)$ are equivalence classes of the states in $\mathbb{G}(\mathrm{Exp}(G))$ with respect to the system's symmetries.
2. Since $\overline{\mathbb{G}}(G)$ is a two-player game with complete information, standard game solving algorithms are applied to gain a positional winning strategy $\overline{f}$ in $\overline{\mathbb{G}}(G)$.
3. Resolving the symmetries of $\overline{f}$ yields a winning strategy f in $\mathbb{G}(\mathrm{Exp}(G))$.
4. Applying the techniques from [FO17; Gie22] (demonstrated in Sec. 3.1.3) to f yields a winning strategy ξ in $\mathrm{Exp}(G)$.

Since strategies in a high-level Petri game are defined as strategies in its expansion, these four steps yield the strategy ξ for the system players in G. An overview of this algorithm and the interplay of the individual components is presented in Fig. 3.8. It can be seen as a zoomed in version of the left part of Fig. 1.1. In the top part of the figure the steps involving high-level elements are depicted, whereas the bottom part shows the solving of P/T Petri games from Sec. 3.1.3. The individual steps of the algorithm are marked bold. The edges between the top and the bottom layer show the relation of the high-level and the P/T elements. Note that step 3 and step 4 could be combined to obtain the Petri game strategy f directly from the symbolic two-player game strategy $\overline{f}$. However, introducing step 3 and showing its correctness yields, together with [FO17; Gie22], the same result and simplifies the presentation.

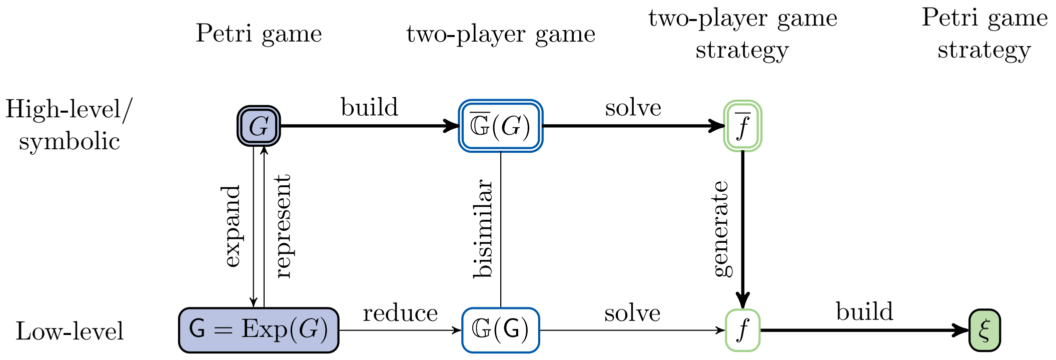


Figure 3.8: The correlation of the games and strategies in the process of solving high-level and P/T Petri games.

Step 1 is the crucial part of the algorithm and this section serves for its elaboration. The definition of $\overline{\mathbb{G}}(G)$ is split into three parts. Firstly, we define how to apply symmetries on the states of $\mathbb{G}(\mathrm{Exp}(G))$ to obtain equivalence classes serving as states of $\overline{\mathbb{G}}(G)$.

Secondly, we examine the relations between the classes to reduce the number of edges induced by $\mathbb{G}(\mathrm{Exp}(G))$. The result serves as edges of $\overline{\mathbb{G}}(G)$. Finally, we define the two-player game $\overline{\mathbb{G}}(G)$ and show the correctness of our approach by defining a bisimulation between $\mathbb{G}(\mathrm{Exp}(G))$ and $\overline{\mathbb{G}}(G)$, and generally proving that two bisimilar two-player games coincide regarding the existence of a winning strategy for Player 0.

In [CDFH97], Chiola et al. construct a *symbolic reachability graph* for high-level Petri nets. In this graph the nodes are *equivalence classes* of markings with respect to symmetries, and instead of the ordinary firing relation between the markings, the *symbolic firing* relation is used. The following section introduces equivalence classes of decision sets with respect to symmetries and lifts results of [CDFH97] about markings to decision sets. The equivalence classes form the vertices of the symbolic two-player game $\overline{\mathbb{G}}(G)$.

3.2.1 Symbolic Decision Sets

We define the application of symmetries (cp. Def. 3.1) to decision sets (cp. Def. 3.8). Furthermore, we establish that the properties of these decision sets remain invariant under the application of symmetries. This means all decision sets in an equivalence class with respect to the admissible symmetries have the same properties. These equivalence classes are called *symbolic decision sets*, and will later be the vertices of the symbolic two-player game we aim to construct.

Recall that for a high-level Petri game G, the (admissible) symmetries are defined to be the (admissible) symmetries of $N(G)$. In the following, let S always be the set of admissible symmetries of the currently considered high-level game.

Due to the special syntax of $\mathrm{Exp}(G)$ for a high-level Petri game G, and since both games have analogous semantics, we define the decision sets of G as the decision sets of $\mathrm{Exp}(G)$, i.e., $\mathcal{D}(G) := \mathcal{D}(\mathrm{Exp}(G))$. For a decision set $\mathsf{D} \in \mathcal{D}(G)$ and any symmetry $s \in S$ we define the *application of a symmetry to a decision set* by

$$s(\mathsf{D}) := \{(p.s(c), s(\mathcal{T})) \mid (p.c, \mathcal{T}) \in \mathsf{D}\},$$

with $s(\mathcal{T}) := \{t.s(\sigma) \mid t.\sigma \in \mathcal{T}\}$ if $\mathcal{T} \subseteq \mathsf{T}$, and $s(\dagger) := \dagger$. This means if a player of color c on place p allows transition t in mode σ in the decision set D (i.e., $(p.c, \mathcal{T}) \in \mathsf{D}$ and $t.\sigma \in \mathcal{T}$), then after the application of the symmetry s, the player of color $s(c)$ on place p allows transition t in mode $s(\sigma)$ (i.e., $(p.s(c), s(\mathcal{T})) \in s(\mathsf{D})$ and $t.s(\sigma) \in s(\mathcal{T})$).

From the definition of admissible symmetries (Def. 3.1) we can easily derive

$$\forall s \in S: \qquad t.\sigma \in pre(p.c) \quad \Leftrightarrow \quad t.s(\sigma) \in pre(p.s(c)).$$

This, together with the definition of decision sets (Def. 3.8), implies that admissible symmetries map $\mathcal{D}(G)$ into $\mathcal{D}(G)$. Since symmetries operate on the first coordinate of a decision set exactly as on markings, we obtain $\forall \mathsf{D} \in \mathcal{D}(G) : s(\mathsf{M}(\mathsf{D})) = \mathsf{M}(s(\mathsf{D}))$.

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

Lemma 3.14 (Relations under symmetries). *The admissible symmetries are compatible with the firing of a transition $t \in T$ in mode $\sigma \in \Sigma(t)$ in a decision set. The* *same is true for the resolution of a †-symbol in a decision set. Formally: let* D, D' *be two decision sets. Then*

(i) $\forall t \in T\ \forall \sigma \in \Sigma(t)\ \forall s \in S : \mathsf{D}[t.\sigma\rangle \mathsf{D}' \Leftrightarrow s(\mathsf{D})[t.s(\sigma)\rangle s(\mathsf{D}')$.

(ii) $\forall s \in S : \mathsf{D}[\dagger\rangle \mathsf{D}' \Leftrightarrow s(\mathsf{D})[\dagger\rangle s(\mathsf{D}')$.

Proof. Let $t \in T$, $\sigma \in \Sigma(t)$, and $s \in S$.

Assume $\mathsf{D}[t.\sigma\rangle \mathsf{D}'$. We first check that $t.s(\sigma)$ is fireable at $s(\mathsf{D})$. $\mathsf{D}[t.\sigma\rangle$ is equivalent to $\mathit{pre}(t.\sigma) \subseteq \{p.c \mid (p.c, \mathcal{T}) \in \mathsf{D} \wedge t.\sigma \in \mathcal{T}\}$. Since s is admissible, we have

$$\begin{aligned}\mathit{pre}(t.s(\sigma)) = s(\mathit{pre}(t.\sigma)) &\subseteq s(\{p.c \mid (p.c, \mathcal{T}) \in \mathsf{D} \wedge t.\sigma \in \mathcal{T}\}) \\ &= \{p.s(c) \mid (p.c, \mathcal{T}) \in \mathsf{D} \wedge t.\sigma \in \mathcal{T}\} \\ &= \{p.c \mid (p.c, \mathcal{T}) \in s(\mathsf{D}) \wedge t.s(\sigma) \in \mathcal{T}\},\end{aligned}$$

therefore $s(\mathsf{D})[t.s(\sigma)\rangle$. Additionally, we have

$$\begin{aligned}s(\mathsf{D}') =&\{(p.c, \mathcal{T}) \mid (p.c, \mathcal{T}) \in s(\mathsf{D}) \wedge p.c \notin \mathit{pre}(t.s(\sigma))\} \\ &\cup \{(p.c, \dagger) \mid p.c \in \mathit{post}(t.s(\sigma)) \cap \mathsf{P}_{\mathrm{S}}^{G}\} \\ &\cup \{(e.d, \mathit{post}(e.d)) \mid e.d \in \mathit{post}(t.s(\sigma)) \cap \mathsf{P}_{\mathrm{E}}^{G}\},\end{aligned}$$

which is precisely the decision set computed after firing $t.s(\sigma)$ in $s(\mathsf{D})$ according to Sec. 3.1.3, which shows $s(\mathsf{D})[t.s(\sigma)\rangle s(\mathsf{D}')$.

Conversely, assume $s(\mathsf{D})[t.s(\sigma)\rangle s(\mathsf{D}')$. Showing $\mathsf{D}[t.\sigma\rangle \mathsf{D}'$ works exactly as above with s^{-1} applied instead of s.

Finally, showing $\mathsf{D}[\dagger\rangle \mathsf{D}' \Leftrightarrow s(\mathsf{D})[\dagger\rangle s(\mathsf{D}')$ works analogously, completing the proof. □

Two decision sets D and D' are *equivalent (w.r.t. S)* iff there is a symmetry $s \in S$ such that $s(\mathsf{D}) = \mathsf{D}'$ holds. This leads to the set of equivalence classes $\mathcal{D}(G)/S$ of the decision sets. For a decision set $\mathsf{D} \in \mathcal{D}(G)$, we denote its equivalence class in $\mathcal{D}(G)/S$ by $[\mathsf{D}]$. These equivalence classes are called *symbolic decision sets.* For every such symbolic decision set $[\mathsf{D}]$, we define $\overline{\mathsf{D}} \in [\mathsf{D}]$ as an arbitrarily chosen, but fixed *representative.* We often identify each equivalence class with its representative. For every a decision set D, we fix with s_{D} a symmetry that maps D to its corresponding representative, i.e., $s_{\mathsf{D}}(\mathsf{D}) = \overline{\mathsf{D}} \in [\mathsf{D}]$.

Example 3.15 (Symbolic decision sets in Signal Sending Satellites). Consider the running example Signal Sending Satellites from Fig. 3.2. Recall that the admissible symmetries in this Petri game are given by $S_3 = \mathit{Sym}(\{1, 2, 3\})$, which are the following six permutations (written in “Cauchy’s two-line notation”, mapping the elements in the top row to the respective elements in the bottom row):

$$s_1 = \mathit{id}_{\{1,2,3\}} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, \qquad s_2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}, \qquad s_3 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix},$$

$$s_4 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}, \qquad s_5 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}, \qquad s_6 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}.$$

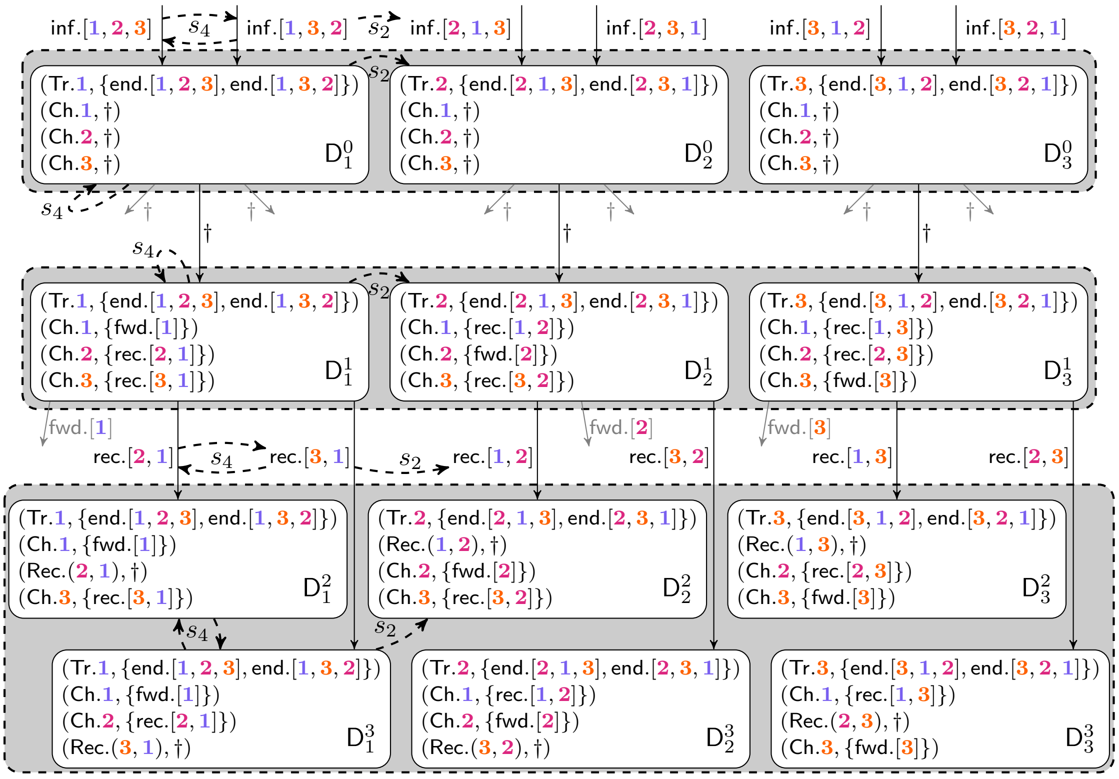

inf.[1, 2, 3]
s_4
inf.[1, 3, 2]
s_2
inf.[2, 1, 3]
inf.[2, 3, 1]
inf.[3, 1, 2]
inf.[3, 2, 1]
(Tr.1, {end.[1, 2, 3], end.[1, 3, 2]}) (Ch.1, †) (Ch.2, †) (Ch.3, †) D^0_1
(Tr.2, {end.[2, 1, 3], end.[2, 3, 1]}) (Ch.1, †) (Ch.2, †) (Ch.3, †) D^0_2
(Tr.3, {end.[3, 1, 2], end.[3, 2, 1]}) (Ch.1, †) (Ch.2, †) (Ch.3, †) D^0_3
†
(Tr.1, {end.[1, 2, 3], end.[1, 3, 2]}) (Ch.1, {fwd.[1]}) (Ch.2, {rec.[2, 1]}) (Ch.3, {rec.[3, 1]}) D^1_1
(Tr.2, {end.[2, 1, 3], end.[2, 3, 1]}) (Ch.1, {rec.[1, 2]}) (Ch.2, {fwd.[2]}) (Ch.3, {rec.[3, 2]}) D^1_2
(Tr.3, {end.[3, 1, 2], end.[3, 2, 1]}) (Ch.1, {rec.[1, 3]}) (Ch.2, {rec.[2, 3]}) (Ch.3, {fwd.[3]}) D^1_3
fwd.[1]
rec.[2, 1]
rec.[3, 1]
rec.[1, 2]
fwd.[2]
rec.[3, 2]
fwd.[3]
rec.[1, 3]
rec.[2, 3]
(Tr.1, {end.[1, 2, 3], end.[1, 3, 2]}) (Ch.1, {fwd.[1]}) (Rec.(2, 1), †) (Ch.3, {rec.[3, 1]}) D^2_1
(Tr.2, {end.[2, 1, 3], end.[2, 3, 1]}) (Rec.(1, 2), †) (Ch.2, {fwd.[2]}) (Ch.3, {rec.[3, 2]}) D^2_2
(Tr.3, {end.[3, 1, 2], end.[3, 2, 1]}) (Rec.(1, 3), †) (Ch.2, {rec.[2, 3]}) (Ch.3, {fwd.[3]}) D^2_3
(Tr.1, {end.[1, 2, 3], end.[1, 3, 2]}) (Ch.1, {fwd.[1]}) (Ch.2, {rec.[2, 1]}) (Rec.(3, 1), †) D^3_1
(Tr.2, {end.[2, 1, 3], end.[2, 3, 1]}) (Ch.1, {rec.[1, 2]}) (Ch.2, {fwd.[2]}) (Rec.(3, 2), †) D^3_2
(Tr.3, {end.[3, 1, 2], end.[3, 2, 1]}) (Ch.1, {rec.[1, 3]}) (Rec.(2, 3), †) (Ch.3, {fwd.[3]}) D^3_3

Figure 3.9: Symmetries on decision sets.

Figure 3.9 shows part of the two-player game corresponding to the Petri game. At the top, we see the three decision sets D^0_1, D^0_2 and D^0_3. Each of them represents a state in the Petri game where a color $\overline{x} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$ resides on the place *Tr*, meaning that the Base Station is transmitting a message to Satellite $\overline{x}$. In its commitment set are the two instances of transition *end* that consume $\overline{x}$ from *Tr*. We abbreviate the modes $\{x \leftarrow \widetilde{x}, y \leftarrow \widetilde{y}, z \leftarrow \widetilde{z}\}$ of *end* by $[\widetilde{x}, \widetilde{y}, \widetilde{z}]$, and analogously for the other transitions. In the case of *rec* we abbreviate the mode $\{y \leftarrow \widetilde{y}, x \leftarrow \widetilde{x}\}$ by $[\widetilde{y}, \widetilde{x}]$ ("non-alphabetically") since its firing places $(\widetilde{y}, \widetilde{x})$ on *Rec*. The three colors **1**, **2** and **3** reside on the place *Ch*, each equipped with a †-symbol instead of a commitment set, meaning that they have to decide for a commitment step by resolving the †-symbol.

Since all three satellites are equipped with a †-symbol, and †-symbols have to be resolved in $\mathbb{G}(\mathsf{G})$ before any transition can be fired, we know that all three colors have been placed on *Ch* in the previous step by firing an instance of *inf*, indicated by the incoming edges. Notice that D^0_2 is D_6 from Fig. 3.7. D^0_1 and D^0_3 can be reached on paths from D'_2 and D''_2 in Fig. 3.7, respectively. These paths are symmetric to the path from D_2 to D_6

From the previous discussions of symmetries, we directly notice that the three situations represented by $\mathsf{D}^0_1, \mathsf{D}^0_2, \mathsf{D}^0_3$ are symmetric. We see for example that $s_2(\mathsf{D}^0_1) = \mathsf{D}^0_2$. Since s_4 only switches **2** and **3**, and they "occur symmetrically" in D^0_1, we get $s_4(\mathsf{D}^0_1) = \mathsf{D}^0_1$.

In fact, D^0_1, D^0_2 and D^0_3 are in the same symbolic decision set, which is illustrated by the dashed (gray) rectangle.

At the second level we see the three decision sets D^1_1, D^1_2 and D^1_3, each resulting from resolving the †-symbol correspondingly to the informally described winning strategy for the system players: the color $\overline{x}$ on *Ch* to decides for the commitment set $\{\mathit{fwd}.[\overline{x}]\}$, meaning that the satellite which the environment transmits a message to wants to go into Forwarding mode in its next step. The other two satellites want to go into Receiving mode in their next step, represented by the commitment sets containing the respective instance of *rec*. The decision sets reached by other †-resolutions (i.e., where the system players chose different commitment sets) are omitted in the figure.

In the relations of the top and middle layer we can verify Lemma 3.14, property (ii): We have $\mathsf{D}^0_1[\dagger\rangle\mathsf{D}^0_2$. We see that $s_2(\mathsf{D}^0_1) = \mathsf{D}^0_2$ and $s_2(\mathsf{D}^1_1) = \mathsf{D}^1_2$, and, in fact, $\mathsf{D}^0_2[\dagger\rangle\mathsf{D}^1_2$. The same property can be seen for s_4, since we again have $s_4(\mathsf{D}^1_1) = \mathsf{D}^1_1$. Again, the three decision sets D^1_1, D^1_2 and D^1_3 are in the same equivalence class.

Finally, at the bottom, we see an equivalence class containing six decision sets. This is the maximal cardinality a symbolic decision set can have, due to the number of symmetries. Each of the decision sets D^j_i, $i = 1, 2, 3$, $j = 2, 3$ is reached from D^1_i by firing an instance of *rec*.

Since all of those instances are symmetric, this is an illustration of Lemma 3.14, property (i): An example for that is $\mathsf{D}^1_1[\mathit{rec}.[3, 1]\rangle\mathsf{D}^3_1$ which, together with $s_2(\mathsf{D}^1_1) = \mathsf{D}^1_2$, $s_2([3, 1]) = [1, 2]$, and $s_2(\mathsf{D}^3_1) = \mathsf{D}^2_2$, yields $\mathsf{D}^1_2[\mathit{rec}.[1, 2]\rangle\mathsf{D}^2_2$. We also see that, although $s_4(\mathsf{D}^1_1) = \mathsf{D}^1_1$, the two instances $\mathit{rec}.[2, 1]$ and $\mathit{rec}.[3, 1] = \mathit{rec}.s_4([2, 1])$ lead to different decision sets D^2_1 and D^3_1. These, however, are symmetric with $s_4(\mathsf{D}^2_1) = \mathsf{D}^3_1$.

The three instances of *fwd*, fireable from the decision sets in the second equivalence class, are also symmetric, and lead to symmetric decision sets. This part is omitted in the figure. ◁

Lemma 3.16 (Decision set properties under symmetries). *Let G be a high-level Petri game with admissible symmetries S. Let additionally $\mathsf{D} \in \mathcal{D}(G)$ and $s \in S$. Then D is environment-dependent, contains a special place, is a deadlock, is terminating or is nondeterministic if and only if $s(\mathsf{D})$ has the same property.*

Proof. Let $G = (P_S, P_E, T, F, g, M_0, \mathrm{Obj}, P_{\circledast})$ and $\mathrm{Exp}(G) = (\mathsf{P}_S, \mathsf{P}_E, \mathsf{T}, \mathsf{F}, \mathrm{Obj}, \mathsf{P}_{\circledast})$. For each property, we only show that if D has it, then so does $s(\mathsf{D})$. Applying s^{-1} then gives the other direction.

Let D be environment dependent. By definition, this means $\neg\mathsf{D}[\dagger\rangle$ and $\exists\widetilde{p}.\widetilde{c} \in \mathsf{P}_E \cap \mathsf{M}(\mathsf{D})\,\forall t.\sigma \in \mathsf{T} : \mathsf{D}[t.\sigma\rangle \Rightarrow \widetilde{p}.\widetilde{c} \in \mathit{pre}(t.\sigma)$. Aiming a contradiction, assume $s(\mathsf{D})[\dagger\rangle$. Then $\exists(p.c, \dagger) \in s(\mathsf{D})$, implying $(p.s^{-1}(c), \dagger) \in \mathsf{D}$, which contradicts $\neg\mathsf{D}[\dagger\rangle$. Therefore, we have $\neg s(\mathsf{D})[\dagger\rangle$. Additionally, we have $\widetilde{p}.s(\widetilde{c}) \in \mathsf{P}_E \cap \mathsf{M}(s(\mathsf{D}))$, and for all $t.\sigma \in \mathsf{T}$, from $s(\mathsf{D})[t.\sigma\rangle$ follows by Lemma 3.14 that $\mathsf{D}[t.s^{-1}(\sigma)\rangle$. Since D is environment dependent, this implies $\widetilde{p}.\widetilde{c} \in \mathit{pre}(t.s^{-1}(\sigma))$, which, in its turn means $\widetilde{p}.s(\widetilde{c}) \in \mathit{pre}(t.\sigma)$. Thus, $s(\mathsf{D})$ is environment dependent.

Let D contain a special place. By definition, this means $\exists p.c \in \mathsf{M}(\mathsf{D}) : (p, c) \in P_{\circledast} \times \mathit{Col}$. This implies $p.s(c) \in \mathsf{M}(s(\mathsf{D}))$ and $(p, s(c)) \in P_{\circledast} \times \mathit{Col}$, and thus, $s(\mathsf{D})$ contains a special place.

Let D be a deadlock. By definition, this means $\forall t.\sigma \in \mathsf{T} : \neg \mathsf{D}[t.\sigma\rangle$ but $\exists \widetilde{t.\sigma} \in \mathsf{T} : \mathsf{M}(\mathsf{D})[\widetilde{t.\sigma}\rangle$. Let $t.\sigma \in \mathsf{T}$. Assuming $s(\mathsf{D})[t.\sigma\rangle$ implies $\mathsf{D}[t.s^{-1}(\sigma)\rangle$ and therefore contradicts that D is a deadlock. Thus, $\forall t.\sigma \in \mathsf{T} : \neg s(\mathsf{D})[t.\sigma\rangle$. From $\mathsf{M}(\mathsf{D})[\widetilde{t.\sigma}\rangle$ we get with Lemma 3.2 and $s(\mathsf{M}(\mathsf{D})) = \mathsf{M}(s(\mathsf{D}))$ that $\mathsf{M}(s(\mathsf{D}))[\widetilde{t}.s(\widetilde{\sigma})\rangle$, making $s(\mathsf{D})$ a deadlock.

Let D be terminating. By definition, this means $\forall t.\sigma \in \mathsf{T} : \neg \mathsf{M}(\mathsf{D})[t.\sigma\rangle$. Aiming a contradiction, assume $\exists t.\sigma \in \mathsf{T} : \mathsf{M}(s(\mathsf{D}))[t.\sigma\rangle$. Lemma 3.2 and $s(\mathsf{M}(\mathsf{D})) = \mathsf{M}(s(\mathsf{D}))$ then give that $\mathsf{M}(\mathsf{D})[t.s^{-1}(\sigma)\rangle$, contradicting that D is terminating. Thus, $s(\mathsf{D})$ is terminating.

Let D be nondeterministic. By definition, this means $\exists t_1.\sigma_1, t_2.\sigma_2 \in \mathsf{T}$:

- $t_1.\sigma_1 \neq t_2.\sigma_2$,
- $\exists (\widetilde{p}, \widetilde{c}) \in (P_\mathrm{S} \times \mathit{Col}) \cap \mathit{pre}(t_1, \sigma_1) \cap \mathit{pre}(t_2, \sigma_2)$,
- $\mathsf{D}[t_1.\sigma_1\rangle \wedge \mathsf{D}[t_2.\sigma_2\rangle$.

We have that $t_1.s(\sigma_1), t_2.s(\sigma_2) \in \mathsf{T}$. Since s is bijective, it is $t_1.s(\sigma_1) \neq t_2.s(\sigma_2)$. We have that $(\widetilde{p}, s(\widetilde{c})) \in P_\mathrm{S} \times \mathit{Col}$, $(\widetilde{p}, s(\widetilde{c})) \in \mathit{pre}(t_1, s(\sigma_1))$, and $(\widetilde{p}, s(\widetilde{c})) \in \mathit{pre}(t_2, s(\sigma_2))$. Finally, by Lemma 3.14, we have $s(\mathsf{D})[t_1.s(\sigma_1)\rangle \wedge s(\mathsf{D})[t_2.s(\sigma_2)\rangle$, making $s(\mathsf{D})$ nondeterministic. □

Lemma 3.16 therefore yields the uniform satisfaction of these properties throughout the complete equivalence class:

Corollary 3.17. *Let* D *be a decision set. Then* D *has one of the properties listed in Lemma 3.16 if and only if all* $\mathsf{D}' \in [\mathsf{D}]$ *(and in particular* $\overline{\mathsf{D}}$*) have the same property.*

Having this corollary in mind, we say a symbolic decision set has the same properties (is environment-dependent, contains a bad/target place, is terminating, is a deadlock, or is nondeterministic) as all its elements, and in particular its representative.

The (representatives of) equivalence classes $\overline{\mathsf{D}}$ of the decision sets $\mathsf{D} \in \mathbb{V}$ of the two-player game $\mathbb{G}(\mathrm{Exp}(G))$ form the vertices of the symbolic two-player game $\overline{\mathbb{G}}(G)$ constructed in Sec. 3.2.3. This is precisely the concept of the symbolic reachability graph introduced in [CDFH97], where the nodes are *symbolic* markings $\overline{\mathsf{M}}$ instead of ordinary markings M as in the reachability graph.

Usually, the relation on equivalence classes is given by all connections between the individual elements of the corresponding classes. Lemma 3.14 shows that for equivalence classes of decision sets, we only have to consider connections where the source is a representative of the class. In the next section we reduce this relation even further, by only considering *equivalence classes of firings*, local to the source decision set.

3.2.2 Symbolic Relations Between Symbolic Decision Sets

In this section we define equivalence classes of transition firings to define the edge relation of the symbolic two-player game $\overline{\mathbb{G}}(G)$. This relation is often smaller than the relation containing an edge for every possible transition firing or †-resolution between the corresponding equivalence classes of decision sets. Again, results of [CDFH97] are lifted from markings to decision sets.

From Lemma 3.14 we see that, if an admissible symmetry $s \in S$ leaves a decision set D invariant, then a transition t is fireable in mode $\sigma \in \Sigma(t)$ at D if and only if t is fireable in mode $s(\sigma)$. These symmetries form a group (called the isotropy group of D) and their application leads to equivalence classes of modes which are locally belonging to the decision set. Instead of considering all modes in which a transition is fireable from a decision set, it suffices to consider representatives of these equivalence classes. As a result, the size of the firing relation between equivalence classes of decision sets decreases. This reduced firing relation, called the symbolic firing relation, is the first part of the edge relation of the symbolic two-player game $\overline{\mathbb{G}}(G)$.

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

In addition to the firing relation there is the relation of †-resolution between decision sets. Thus, we also define the *symbolic* †-resolution between (representatives of) *symbolic* decision sets. This relation forms the second part of the edge relation of $\overline{\mathbb{G}}(G)$.

Let $G \in \mathbf{G}$, and $\mathsf{D} \in \mathcal{D}(G)$ be a decision set. The so-called *isotropy group* $S_{\mathsf{D}} := \{s \in S \mid s(\mathsf{D}) = \mathsf{D}\}$ *of* D is the group of all admissible symmetries that preserve D. For a transition $t \in T$, we denote by $\Sigma(t)_{\mathsf{D}} := \Sigma(t)/S_{\mathsf{D}}$ the set containing the equivalence classes of all modes of t, with respect to the isotropy group S_{D}. The individual modes in one class affect D in symmetric ways. For each class in $\Sigma(t)_{\mathsf{D}}$ we arbitrarily choose a representative mode $\overline{\sigma}$, and define α_{D} as the function mapping each $\sigma \in \Sigma(t)$ to its representative $\alpha_{\mathsf{D}}(\sigma)$.

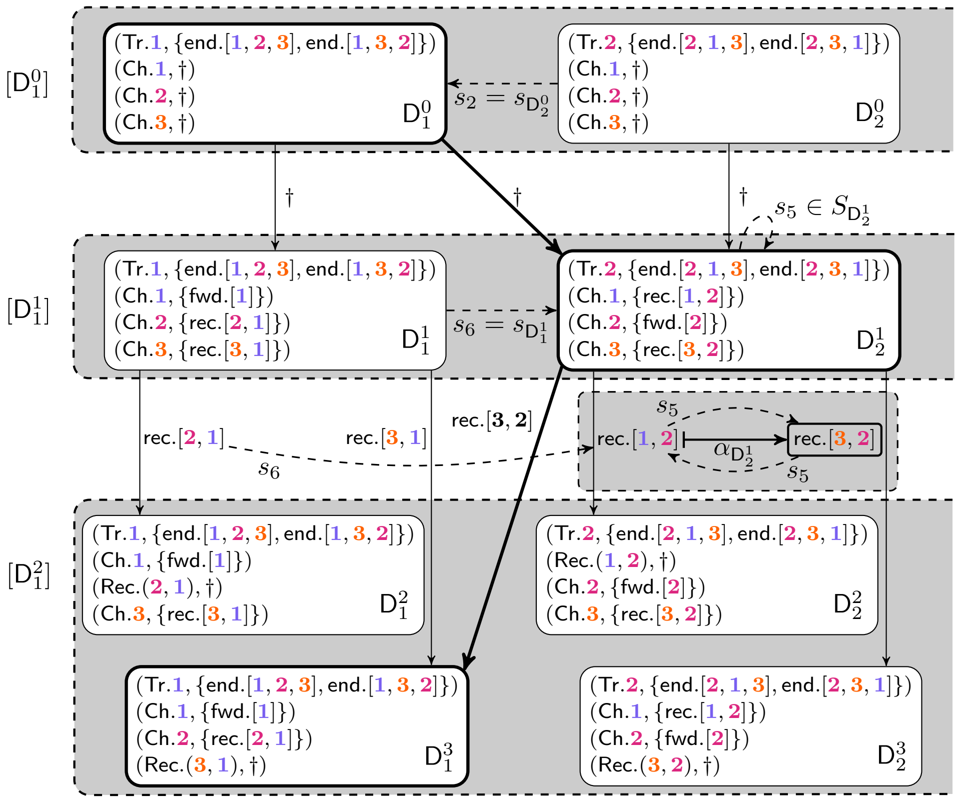
By this definition, and analogously to [CDFH97], we immediately get the following property. It implies that, in a decision set D, a transition can fire in mode σ if and only if it can fire in its representative $\alpha_{\mathsf{D}}(\sigma)$.

Property 3.18. *For every representative $\overline{\sigma}$ of a class in $\Sigma(t)_{\mathsf{D}}$, for every mode σ belonging to the equivalence class of $\overline{\sigma}$, there is a symmetry $s \in S_{\mathsf{D}}$ such that $s(\sigma) = \alpha_{\mathsf{D}}(\sigma) = \overline{\sigma}$.*

Example 3.19. Figure 3.10 shows again the decision sets (and the relations between them) from the left and middle part in Fig. 3.9. Assume that D_1^0, D_2^1, and D_1^3 are the representatives of their respective equivalence class, marked by the thick borders. The thick edges between the representatives will be explained in Example 3.21.

Recall the symmetries in G from Example 3.15 where $s_1 = id_{\{1,2,3\}}$ and $s_5 = \{1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 1\}$. We see that $s_5(\mathsf{D}_2^1) = \mathsf{D}_2^1$. This means $s_5 \in S_{\mathsf{D}_2^1}$. In fact, we have $S_{\mathsf{D}_2^1} = \{s_1, s_5\}$ (i.e., the set of symmetries that fix 2). We see that $s_5([1, 2]) = [3, 2]$, which means the two modes are in the same equivalence class in $\Sigma(rec)_{\mathsf{D}_2^1}$. Assume now $[3, 2]$ to be the arbitrary, but fixed, representative of this equivalence class (marked by the thick border). By definition, $\alpha_{\mathsf{D}_2^1}$ then maps $[1, 2]$ to the representative $[3, 2]$, and fixes $[3, 2]$.

By the above discussion, we have verified Property 3.18 for $\sigma = [1, 2]$, which belongs to the equivalence class of the representative $\overline{\sigma} = [3, 2] = \alpha_{\mathsf{D}_2^1}([1, 2])$ in $\Sigma(rec)_{\mathsf{D}_2^1}$; as we



[D_1^0]
(Tr.1, {end.[1, 2, 3], end.[1, 3, 2]}) (Ch.1, †) (Ch.2, †) (Ch.3, †) D_1^0
(Tr.2, {end.[2, 1, 3], end.[2, 3, 1]}) (Ch.1, †) (Ch.2, †) (Ch.3, †) D_2^0
s_2 = s_{D_2^0}
†
†
†
s_5 ∈ S_{D_2^1}
[D_1^1]
(Tr.1, {end.[1, 2, 3], end.[1, 3, 2]}) (Ch.1, {fwd.[1]}) (Ch.2, {rec.[2, 1]}) (Ch.3, {rec.[3, 1]}) D_1^1
s_6 = s_{D_1^1}
(Tr.2, {end.[2, 1, 3], end.[2, 3, 1]}) (Ch.1, {rec.[1, 2]}) (Ch.2, {fwd.[2]}) (Ch.3, {rec.[3, 2]}) D_2^1
rec.[2, 1]
rec.[3, 1]
rec.[3, 2]
s_6
s_5
rec.[1, 2]
α_{D_2^1}
rec.[3, 2]
s_5
[D_1^2]
(Tr.1, {end.[1, 2, 3], end.[1, 3, 2]}) (Ch.1, {fwd.[1]}) (Rec.(2, 1), †) (Ch.3, {rec.[3, 1]}) D_1^2
(Tr.2, {end.[2, 1, 3], end.[2, 3, 1]}) (Rec.(1, 2), †) (Ch.2, {fwd.[2]}) (Ch.3, {rec.[3, 2]}) D_2^2
(Tr.1, {end.[1, 2, 3], end.[1, 3, 2]}) (Ch.1, {fwd.[1]}) (Ch.2, {rec.[2, 1]}) (Rec.(3, 1), †) D_1^3
(Tr.2, {end.[2, 1, 3], end.[2, 3, 1]}) (Ch.1, {rec.[1, 2]}) (Ch.2, {fwd.[2]}) (Rec.(3, 2), †) D_2^3

Figure 3.10: Symbolic relations between symbolic decision sets.

have seen, it is $s_5([1, 2]) = [3, 2]$. ◁

Since for every symbolic decision set $D = [\mathsf{D}]$ there is exactly one representative $\overline{\mathsf{D}}$, we often identify the two with each other. For example, the symbolic relations between symbolic decision sets are now defined between their representatives. For the symbolic firing, instead of firing a transition in all modes, we only consider the representatives of equivalence classes of modes, local to the symbolic decision set. The symbolic decision set obtained after the *symbolic* firing is corresponding to the decision set obtained after the *ordinary* firing of the transition in the representative mode. To define the symbolic †-resolution between symbolic decision sets, we cannot use representatives of the symbol †. Instead, when symbolically resolving a †-symbol in a symbolic decision set, we declare the possible targets as the representatives of possible targets of an ordinary †-resolution.

Definition 3.20 (Symbolic Relations between Symbolic Decision Sets). We say a transition t can *fire symbolically* from the symbolic decision set $\overline{\mathsf{D}}$ in mode $\alpha_{\overline{\mathsf{D}}}(\sigma)$ representing σ in $\Sigma(t)_{\overline{\mathsf{D}}}$, denoted by $\overline{\mathsf{D}}[\![t.\alpha_{\overline{\mathsf{D}}}(\sigma)\rangle\!\rangle$, iff $\overline{\mathsf{D}}[t.\alpha_{\overline{\mathsf{D}}}(\sigma)\rangle$. The symbolic decision set $\overline{\mathsf{D}'}$ obtained after the symbolic firing is determined as follows:

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

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

$$\overline{\mathsf{D}}[\![\dagger\rangle\!\rangle\overline{\mathsf{D}'} \Leftrightarrow \exists \mathsf{D}'' \in [\,\overline{\mathsf{D}'}\,] : \overline{\mathsf{D}}[\dagger\rangle\mathsf{D}''. \qquad \lhd$$

By definition, we directly have $\mathsf{D}[\![t.\alpha_{\overline{\mathsf{D}}}(\sigma)\rangle\!\rangle \Rightarrow \exists^1\overline{\mathsf{D}'} : \overline{\mathsf{D}}[\![t.\alpha_{\overline{\mathsf{D}}}(\sigma)\rangle\!\rangle\overline{\mathsf{D}'}$. In the following properties we compare the ordinary firing relation and the ordinary $\dagger$-resolution with their symbolic counterparts.

Example 3.21 (Symbolic relations between symbolic decision sets). Consider again the decision sets in Fig. 3.10. The decision set D_1^0 is the representative of its equivalence class $[\mathsf{D}_1^0]$. The relation $\mathsf{D}_1^0[\dagger\rangle\mathsf{D}_1^1$, gives the symbolic relation $\mathsf{D}_1^0[\![\dagger\rangle\!\rangle\overline{\mathsf{D}_1^1}$, where $\overline{\mathsf{D}_1^1}$ is the representative of $[\mathsf{D}_1^1]$. Since we assumed $\overline{\mathsf{D}_1^1} = \mathsf{D}_2^1$, this means $\mathsf{D}_1^0[\![\dagger\rangle\!\rangle\mathsf{D}_2^1$. This is illustrated by the thick edge from D_1^0 to D_2^1

Consider now the fact that $\mathsf{D}_2^1[\mathit{rec}.[1,2]\rangle$ and $\mathsf{D}_2^1[\mathit{rec}.[3,2]\rangle$. We have $\alpha_{\mathsf{D}_2^1}([1,2]) = \alpha_{\mathsf{D}_2^1}([3,2]) = [3,2]$, which implies $\mathsf{D}_2^1[\![\mathit{rec}.[3,2]\rangle\!\rangle$ but *not* $\mathsf{D}_2^1[\![\mathit{rec}.[1,2]\rangle\!\rangle$ since $[1,2]$ is not the representative of its equivalence class in $\Sigma(\mathit{rec})_{\mathsf{D}_2^1}$. The symbolic decision set obtained after symbolically firing $\mathit{rec}.[3,2]$ is then given by D_1^3 since $\mathsf{D}_2^1[\mathit{rec}.[3,2]\rangle\mathsf{D}_2^3$ and (by assumption) $\overline{\mathsf{D}_2^3} = \mathsf{D}_1^3$. All in all this leads to the symbolic relation $\mathsf{D}_2^1[\![\mathit{rec}.[3,2]\rangle\!\rangle\mathsf{D}_1^3$, also illustrated by a thick edge in Fig. 3.10. $\lhd$

The following two properties discuss how the symbolic relations between symbolic decision sets represent the ordinary relations between ordinary decision sets. First, it is proved that every ordinary relation is represented by a symbolic one:

Property 3.22. *Each ordinary transition firing is represented by a unique symbolic transition firing, and each ordinary $\dagger$-resolution is represented by a symbolic $\dagger$-resolution:*

i) $\mathsf{D}[t.\sigma\rangle\mathsf{D}' \Rightarrow \overline{\mathsf{D}}[\![t.\overline{\sigma}\rangle\!\rangle\overline{\mathsf{D}'}$, *where* $\overline{\sigma} = \alpha_{\overline{D}}(s_D(\sigma))$.

ii) $\mathsf{D}[\dagger\rangle\mathsf{D}' \Rightarrow \overline{\mathsf{D}}[\![\dagger\rangle\!\rangle\overline{\mathsf{D}'}$.

Proof. Property *i)* can be shown with the help of Lemma 3.14, analogously to the proof of the corresponding Property 5.1 (for markings instead of decision sets) in [CDFH97]:

Let $\mathsf{D}[t.\sigma\rangle\mathsf{D}'$. Then Lemma 3.14 gives $\overline{\mathsf{D}}[t.s_{\mathsf{D}}(\sigma)\rangle s_{\mathsf{D}}(\mathsf{D}')$. Let $\overline{\sigma} = \alpha_{\overline{\mathsf{D}}}(s_{\mathsf{D}}(\sigma))$ be the representative of $s_{\mathsf{D}}(\sigma)$ in $\overline{\mathsf{D}}$. From Property 3.18 we get $\exists s \in S_{\overline{\mathsf{D}}} : s(s_{\overline{\mathsf{D}}}(c)) = \alpha_{\overline{\mathsf{D}}}(s_{\mathsf{D}}(\sigma))$. Since s preserves $\overline{\mathsf{D}}$, applying s to our relation yields $\overline{\mathsf{D}}[t.\overline{\sigma}\rangle s(s_{\mathsf{D}}(\mathsf{D}'))$. As $\overline{s(s_{\mathsf{D}}(\mathsf{D}'))} = \overline{\mathsf{D}'}$, we finally obtain $\overline{\mathsf{D}}[\![t.\overline{\sigma}\rangle\!\rangle\overline{\mathsf{D}'}$.

The proof of *ii)* works similarly, albeit easier, without the use of Property 3.18: Let $\mathsf{D}[\dagger\rangle\mathsf{D}'$. Lemma 3.14 gives $\overline{\mathsf{D}}[\dagger\rangle s_{\mathsf{D}}(\mathsf{D}')$. Since $\overline{s_{\mathsf{D}}(\mathsf{D}')} = \overline{\mathsf{D}'}$, we obtain $\overline{\mathsf{D}}[\![\dagger\rangle\!\rangle\overline{\mathsf{D}'}$. $\square$

Example 3.23. Consider the relation $\mathsf{D}_1^1[\mathit{rec}.[2,1]\rangle\mathsf{D}_1^2$ from Fig. 3.10, and recall $s_6 = \{1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 3\}$ from Example 3.15. Since $s_6(\mathsf{D}_1^1) = \mathsf{D}_2^1 = \overline{\mathsf{D}_1^1}$ we can set $s_{\mathsf{D}_1^1} = s_6$. We have $s_6([2,1]) = [1,2]$, and, as discussed in Example 3.19, $\alpha_{\mathsf{D}_2^1}([1,2]) = [3,2]$. Thus, we have $\alpha_{\overline{\mathsf{D}_1^1}}(s_{\mathsf{D}_1^1}([2,1])) = [3,2]$. Together with $\overline{\mathsf{D}_1^2} = \mathsf{D}_1^3$, Property 3.22 *i)* yields the symbolic relation $\mathsf{D}_2^1\llbracket rec.[3,2]\rangle\!\rangle\mathsf{D}_1^3$ that we discussed in Example 3.21.

Analogously, by Property 3.22 *ii)*, $\mathsf{D}_2^0[\dagger\rangle\mathsf{D}_2^1$ implies $\mathsf{D}_1^0\llbracket\dagger\rangle\!\rangle\mathsf{D}_2^1$, since $\overline{\mathsf{D}_2^0} = \mathsf{D}_1^0$ and $\overline{\mathsf{D}_2^1} = \mathsf{D}_2^1$. $\lhd$

The following Property 3.24 describes the reverse direction of Property 3.22, i.e., it is concerned with the set of ordinary relations represented by each symbolic relation:

Property 3.24. *Each symbolic firing represents a set of ordinary firings, in which all source decision sets belong to the equivalence class of the symbolic source decision set of the symbolic firing. The same holds for the resolution of* $\dagger$. *Formally:*

i) $\overline{\mathsf{D}}\llbracket t.\overline{\sigma}\rangle\!\rangle\overline{\mathsf{D}'} \Rightarrow (\forall \mathsf{D}_1 \in [\,\overline{\mathsf{D}}\,]\ \forall \sigma' \in \Sigma(t) : \alpha_{\overline{D}}(s_{\mathsf{D}_1}(\sigma')) = \overline{\sigma} \Rightarrow \exists \mathsf{D}_2 \in [\,\overline{\mathsf{D}'}\,] : \mathsf{D}_1[t.\sigma'\rangle\mathsf{D}_2.)$

ii) $\overline{\mathsf{D}}\llbracket\dagger\rangle\!\rangle\overline{\mathsf{D}'} \Rightarrow \forall \mathsf{D}_1 \in [\,\overline{\mathsf{D}}\,]\ \exists \mathsf{D}_2 \in [\,\overline{\mathsf{D}'}\,] : \mathsf{D}_1[\dagger\rangle\mathsf{D}_2.$

Proof. Again, *i)* can be shown analogously to the proof of the corresponding Property 5.2 in [CDFH97] for markings:

Let $\overline{\mathsf{D}}\llbracket t.\overline{\sigma}\rangle\!\rangle\overline{\mathsf{D}'}$ and $\mathsf{D}_1 \in [\overline{\mathsf{D}}]$. By definition, $\overline{\mathsf{D}}\llbracket t.\overline{\sigma}\rangle\!\rangle\overline{\mathsf{D}'}$ implies there exists a D'' such that $\overline{\mathsf{D}}[t.\overline{\sigma}\rangle\mathsf{D}''$ and $\overline{\mathsf{D}''} = \overline{\mathsf{D}'}$. Let σ' such that $\alpha_{\overline{D}}(s_{\mathsf{D}_1}(\sigma')) = \overline{\sigma}$, and denote $\sigma'' = s_{\mathsf{D}_1}(\sigma')$. As $\overline{\sigma}$ is the representative of σ'' in $\Sigma(t)_{\overline{\mathsf{D}}}$, there exists by Prop. 3.18 a $s \in S_{\overline{\mathsf{D}}}$ such that $s(\sigma'') = \overline{\sigma}$. Since $S_{\overline{\mathsf{D}}}$ is a group, we have $s^{-1} \in S_{\overline{\mathsf{D}}}$, and applying Lemma 3.14, we obtain $s^{-1}(\overline{\mathsf{D}})[t.s^{-1}(\overline{\sigma})\rangle s^{-1}(\mathsf{D}'')$. This can be written as $\overline{\mathsf{D}}[t.\sigma''\rangle s^{-1}(\mathsf{D}'')$. For all $\mathsf{D}_1 \in \overline{\mathsf{D}}$ we can apply $s_{\mathsf{D}_1}^{-1}$ to this firing and obtain $\mathsf{D}_1[t.s_{\mathsf{D}_1}^{-1}(\sigma'')\rangle s_{\mathsf{D}_1}^{-1}(s^{-1}(\mathsf{D}''))$. As $\overline{s_{\mathsf{D}_1}^{-1}(s^{-1}(\mathsf{D}''))} = \overline{\mathsf{D}''} = \overline{\mathsf{D}'}$, we finally obtain $\mathsf{D}_1[t.\sigma'\rangle\mathsf{D}_2$, where $\mathsf{D}_2 = s_{\mathsf{D}_1}^{-1}(s^{-1}(\mathsf{D}''))$.

The proof of *ii)* is again simpler: $\overline{\mathsf{D}}\llbracket\dagger\rangle\!\rangle\overline{\mathsf{D}'}$ means there is a D'' such that $\overline{\mathsf{D}}[\dagger\rangle\mathsf{D}''$ and $\overline{\mathsf{D}''} = \overline{\mathsf{D}'}$. By applying $s_{\mathsf{D}_1}^{-1}$ to this relation, Lemma 3.14 gives $\mathsf{D}_1[\dagger\rangle s_{\mathsf{D}_1}^{-1}(\mathsf{D}'')$, and since $\overline{s_{\mathsf{D}_1}^{-1}(\mathsf{D}'')} = \overline{\mathsf{D}'}$, we obtain $\mathsf{D}_1[\dagger\rangle\mathsf{D}_2$ for $\mathsf{D}_2 = s_{\mathsf{D}_1}^{-1}(\mathsf{D}'')$. $\square$

Example 3.25. We can go "backwards" through Example 3.23: We start with the symbolic relation $\mathsf{D}_2^1\llbracket rec.[3,2]\rangle\!\rangle\mathsf{D}_1^3$, and choose $\mathsf{D}_1 = \mathsf{D}_1^1 \in [\mathsf{D}_2^1]$ and the mode $\sigma' = [2,1] \in \Sigma(rec)$. Since $[2,1]$ satisfies $\alpha_{\overline{\mathsf{D}_1^1}}(s_{\mathsf{D}_1^1}([2,1])) = [3,2]$, we get by Property 3.24 *i)* that $\exists \mathsf{D}_2 \in [\mathsf{D}_1^3] : \mathsf{D}_1^1[rec.[2,1]\rangle\mathsf{D}_2$. This is satisfied by $\mathsf{D}_2 = \mathsf{D}_1^2 \in [\mathsf{D}_1^3]$. $\lhd$

Property 3.24 describes the ordinary relations represented by symbolic relations where the *source* decision set belongs to the respective symbolic decision set. In [CDFH97], it is additionally shown that, in the case of markings, an ordinary firing can be extracted from every symbolic firing such that the *destination* marking belongs to the class of the *destination* symbolic marking. This statement can also be translated to the symbolic relations between decision sets, and proven analogously. We do not need this property to show our results and therefore omit it here, and refer the interested reader to [CDFH97].

3.2.3 Construction of the Symbolic Two-Player Game

In this section we use the results from the preceding sections to define, for a high-level Petri game G, the *symbolic two-player game* $\overline{\mathbb{G}}(G)$. In the subsequent sections we show that Player 0 has a winning strategy in $\overline{\mathbb{G}}(G)$ if and only if there is a winning strategy for Player 0 in the low-level two-player game $\mathbb{G}(\mathrm{Exp}(G))$. This is proved by introducing a bisimulation on the two-player games. Recall the class $\mathbf{G}$ of proper high-level Petri games from Definition 3.6. For the rest of this section, we fix a proper high-level Petri game $G = (P_{\mathrm{S}}, P_{\mathrm{E}}, T, F, g, M_0, \mathrm{Obj}, P_{\circledast}) \in \mathbf{G}$. Let $\mathsf{G} = \mathrm{Exp}(G) = (\mathsf{P}_{\mathrm{S}}, \mathsf{P}_{\mathrm{E}}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Obj}, \mathsf{P}_{\circledast})$ be the expansion, and S the admissible symmetries of G.

Recall that the vertices of $\mathbb{G}(\mathrm{Exp}(G))$ are decision sets and an edge between two decision sets D and D' only exists if $\mathsf{D}[t.\sigma\rangle\mathsf{D}'$ or $\mathsf{D}[\dagger\rangle\mathsf{D}'$ holds. We analogously define the symbolic two-player game $\overline{\mathbb{G}}(G)$ by considering the symbolic counterparts. This means the vertices of $\overline{\mathbb{G}}(G)$ are symbolic decision sets, and each (labeled) edge between two symbolic decision sets $\overline{\mathsf{D}}$ and $\overline{\mathsf{D}'}$ comes from a symbolic relation $\overline{\mathsf{D}}[\![t.\overline{\sigma}\rangle\!\rangle\overline{\mathsf{D}'}$ or $\overline{\mathsf{D}}[\![\dagger\rangle\!\rangle\overline{\mathsf{D}'}$. Remember that we identify a symbolic decision $[\mathsf{D}]$ (i.e., an equivalence class with respect to S) by its representative $\overline{\mathsf{D}}$ when no confusion arises.

Definition 3.26 (Symbolic two-player game). The *symbolic two-player game* over a finite graph $\overline{\mathbb{G}}(G) = (\overline{\mathbb{V}}_0, \overline{\mathbb{V}}_1, \overline{v}_0, \overline{\mathbb{E}}, \overline{\mathrm{Win}})$ is has the following components:

- The 0-vertices $\overline{\mathbb{V}}_0 \subseteq \mathcal{D}(G)/S$ are all symbolic decision sets that are *not* environment-dependent.
- The 1-vertices $\overline{\mathbb{V}}_1 := (\mathcal{D}(G)/S) \setminus \overline{\mathbb{V}}_0$ are all environment-dependent symbolic decision sets in G.
- The initial vertex is $\overline{\mathsf{D}_0}$, where

$$\mathsf{D}_0 := \{(p.c, \dagger) \mid p.c \in \mathsf{M}_0 \cap \mathsf{P}_{\mathrm{S}}\} \cup \{(e.d, \mathit{post}(e.d)) \mid e.d \in \mathsf{M}_0 \cap \mathsf{P}_{\mathrm{E}}\}.$$

- the *labeled edge relation* $\overline{\mathbb{E}} \subseteq \overline{\mathbb{V}} \times (\mathsf{T} \cup \{\dagger\}) \times \overline{\mathbb{V}}$ is defined as follows: If $\overline{\mathsf{D}}$ contains a bad place, is a deadlock, is terminating, or is nondeterministic, there is only a †-labeled self-loop originating from $\overline{\mathsf{D}}$. Otherwise, consider three disjunct cases for edges originating in $\overline{\mathsf{D}}$:

  Case $\overline{\mathsf{D}} \in \overline{\mathbb{V}}_1$; i.e., all players have decided for a commitment set, but cannot proceed without the environment. Then for all $t \in T$ and $\overline{\sigma} \in \Sigma(t)_{\overline{\mathsf{D}}}$, $(\overline{\mathsf{D}}, t.\overline{\sigma}, \overline{\mathsf{D}'}) \in \overline{\mathbb{E}}$ iff $\overline{\mathsf{D}}[\![t.\overline{\sigma}\rangle\!\rangle\overline{\mathsf{D}'}$.

  Case $\overline{\mathsf{D}} \in \overline{\mathbb{V}}_0$ *and* $\overline{\mathsf{D}}[\![\dagger\rangle\!\rangle$; i.e., at least one system player has yet to decide for a commitment set. Then $(\overline{\mathsf{D}}, \dagger, \overline{\mathsf{D}'}) \in \overline{\mathbb{E}}$ iff $\overline{\mathsf{D}}[\![\dagger\rangle\!\rangle\overline{\mathsf{D}'}$.

  Case $\overline{\mathsf{D}} \in \overline{\mathbb{V}}_0$ *and* $\neg\overline{\mathsf{D}}[\![\dagger\rangle\!\rangle$; i.e., all system players made their decisions and can proceed without the environment. Then for all $t \in T$ and $\overline{\sigma} \in \Sigma(t)_{\overline{\mathsf{D}}}$ with $\mathit{pre}(t.\overline{\sigma}) \cap \mathsf{P}_{\mathrm{E}} = \emptyset$, $(\overline{\mathsf{D}}, t.\overline{\sigma}, \overline{\mathsf{D}'}) \in \overline{\mathbb{E}}$ iff $\overline{\mathsf{D}}[\![t.\overline{\sigma}\rangle\!\rangle\overline{\mathsf{D}'}$.
- $\overline{\mathrm{Win}}$ depending on Obj and $P_{\circledast}$ as follows:
  - if Obj := Safety then $\overline{\mathrm{Win}} = \mathrm{Safety}(\overline{\mathbb{F}})$, where $\overline{\mathbb{F}}$ contains all symbolic decision sets that are a deadlock, nondeterministic, or contain a bad place,
  - if Obj := Reach then $\overline{\mathrm{Win}} = \mathrm{Reach}(\overline{\mathbb{F}})$, where $\overline{\mathbb{F}}$ contains all symbolic decision sets that contain a target place. ◁

Analogously to the low-level case, we denote $\overline{\mathbb{V}} := \overline{\mathbb{V}}_0 \sqcup \overline{\mathbb{V}}_1$. Note that, since the initial marking M_0 is symmetric (i.e., $\forall s \in S(G) : s(\mathsf{M}_0) = \mathsf{M}_0$), $\overline{\mathsf{D}_0} = \mathsf{D}_0$ holds.

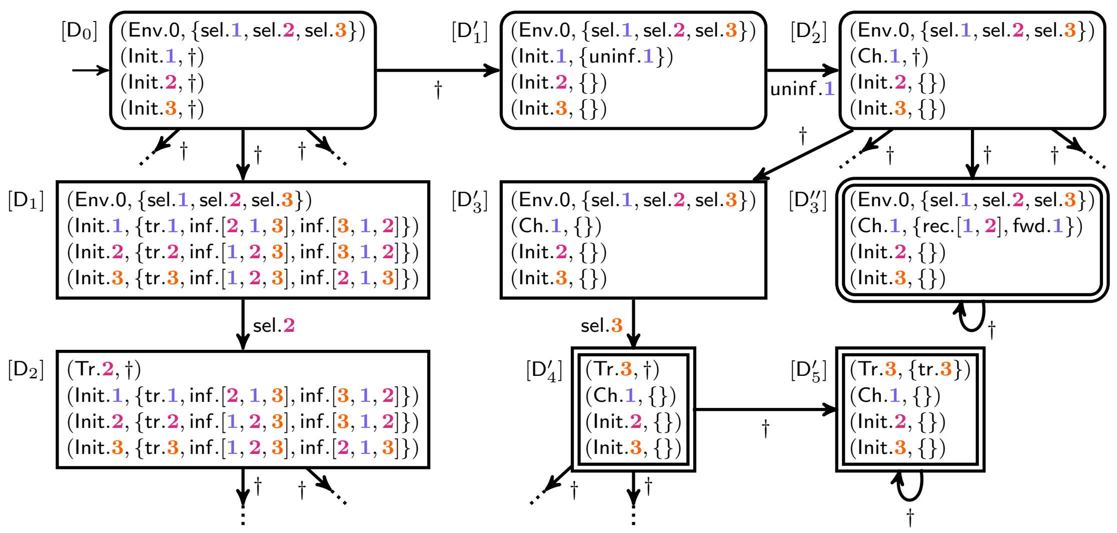

[D0]
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, †)
(Init.2, †)
(Init.3, †)
[D'1]
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, {uninf.1})
(Init.2, {})
(Init.3, {})
[D'2]
(Env.0, {sel.1, sel.2, sel.3})
(Ch.1, †)
(Init.2, {})
(Init.3, {})
uninf.1
†
[D1]
(Env.0, {sel.1, sel.2, sel.3})
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
[D'3]
(Env.0, {sel.1, sel.2, sel.3})
(Ch.1, {})
(Init.2, {})
(Init.3, {})
[D''3]
(Env.0, {sel.1, sel.2, sel.3})
(Ch.1, {rec.[1, 2], fwd.1})
(Init.2, {})
(Init.3, {})
sel.2
sel.3
[D2]
(Tr.2, †)
(Init.1, {tr.1, inf.[2, 1, 3], inf.[3, 1, 2]})
(Init.2, {tr.2, inf.[1, 2, 3], inf.[3, 1, 2]})
(Init.3, {tr.3, inf.[1, 2, 3], inf.[2, 1, 3]})
[D'4]
(Tr.3, †)
(Ch.1, {})
(Init.2, {})
(Init.3, {})
[D'5]
(Tr.3, {tr.3})
(Ch.1, {})
(Init.2, {})
(Init.3, {})

Figure 3.11: Part of the symbolic two-player game for the running example Signal Sending Satellites

Example 3.27 (Symbolic two-player game). Figure 3.11 shows a part of the symbolic two-player game for the running example Signal Sending Satellites. It corresponds to the part of the two-player game from Fig. 3.6 on p. 53, in the sense that every component shown in Fig. 3.6 is represented.

To emphasize that we consider the *symbolic* two-player game, the symbolic decision sets are labeled by the equivalence classes' names. However, instead of showing all instances of every class or depicting it abstractly, we show for each class the respective *representative*. As in Fig. 3.10, we draw these with thick borders, and draw thick edges between them to illustrate the *symbolic relations* between them.

Compared to Fig. 3.6, we see that the three edges from D_1, labeled by $sel.1$, $sel.2$, $sel.3$ are represented by one edge from $[\mathsf{D}_1]$. This edge corresponds to a symbolic relation $\mathsf{D}_1[\![sel.2\rangle\!\rangle\mathsf{D}_2$, where D_1 and D_2 are chosen to be the representatives of their respective equivalence class, and $sel.2$ is the representative of its equivalence class $\{sel.1, sel.2, sel.3\}$ in $\Sigma(sel)_{\mathsf{D}_1}$.

Analogously, the symbolic decision set $[\mathsf{D}'_1]$ represents D''_1 and D'''_1 with representative D'_1, which also means that the corresponding branches in Fig. 3.6 are now represented by the symbolic relations in the future of D'_1 in Fig. 3.11. ◁

3.2.4 Bisimulations between Two-Player Games

We now make a short excursus and generally define a bisimulations on two-player games. We proceed to show that two bisimilar two-player games coincide on the existence of a winning strategy for Player 0. In the subsequent Section 3.2.5, the instantiation of this result for the two-player game $\mathbb{G}(\operatorname{Exp} G)$ and the symbolic two-player game $\overline{\mathbb{G}}(G)$ yields the correctness of the solving algorithm for high-level Petri games $G \in \mathbf{G}$.

Let $\mathbb{G} = (\mathbb{V}_0, \mathbb{V}_1, v_0, \mathbb{E}, \mathrm{Win})$ be a two-player game with vertices $\mathbb{V} := \mathbb{V}_0 \sqcup \mathbb{V}_1$ and $\mathrm{Win} = \mathrm{Safety}(\mathbb{F})$ or $\mathrm{Win} = \mathrm{Reach}(\mathbb{F})$ for a set $\mathbb{F} \subseteq \mathbb{V}$ of special vertices. Then we can view $\mathbb{G}$ as a *state-labeled transition system* $TS(\mathbb{G}) := (\mathbb{V}, \mathbb{E}, \lambda, v_0)$ with the set of states $\mathbb{V}$, the transition relation $\mathbb{E}$, the initial state v_0 as defined in $\mathbb{G}$, and a labeling function $\lambda : \mathbb{V} \to \mathbb{P}(\{\mathfrak{s}, \mathfrak{f}\})$ with propositions $\{\mathfrak{s}, \mathfrak{f}\}$, defined by $\forall v \in \mathbb{V} : (\mathfrak{s} \in \lambda(v) :\Leftrightarrow v \in \mathbb{V}_0) \wedge (\mathfrak{f} \in \lambda(v) :\Leftrightarrow v \in \mathbb{F})$.

A *bisimulation* between two state-labeled transition systems $TS_1 = (\mathcal{S}_1, \to_1, \lambda_1, x_0)$ and $TS_2 = (\mathcal{S}_2, \to_2, \lambda_2, y_0)$ is a relation $R \subseteq \mathcal{S}_1 \times \mathcal{S}_2$ such that for all $(x, y) \in R$

- $\lambda_1(x) = \lambda_2(y)$,
- $\exists x \in \mathcal{S}_1 : x \to_1 x' \quad \Rightarrow \quad \exists y' \in \mathcal{S}_2 : y \to_2 y' \wedge (x', y') \in R$, and
- $\exists y' \in \mathcal{S}_2 : y \to_2 y' \quad \Rightarrow \quad \exists x' \in \mathcal{S}_1 : x \to_1 x' \wedge (x', y') \in R$

holds. Two states $x \in \mathcal{S}_1$ and $y \in \mathcal{S}_2$ are called *bisimilar*, denoted by $x \sim y$, iff there is a bisimulation R between TS_1 and TS_2 satisfying $(x, y) \in R$. The transition systems TS_1 and TS_2 are called *bisimilar*, denoted by $TS_1 \sim TS_2$, iff $x_0 \sim y_0$.

Definition 3.28 (Bisimulation between two-player games). Two two-player games $\mathbb{G} = (\mathbb{V}_0, \mathbb{V}_1, v_0, \mathbb{E}, \mathrm{Cond}(\mathbb{F}))$ and $\mathbb{G}' = (\mathbb{V}'_0, \mathbb{V}'_1, v'_0, \mathbb{E}', \mathrm{Cond}(\mathbb{F}'))$ with the *same* winning condition $\mathrm{Cond} \in \{\mathrm{Reach}, \mathrm{Safety}\}$ are *bisimilar*, denoted by $\mathbb{G} \sim \mathbb{G}'$, iff the corresponding transition systems are bisimilar, i.e., $TS(\mathbb{G}) \sim TS(\mathbb{G}')$. The bisimulations between $TS(\mathbb{G})$ and $TS(\mathbb{G}')$ are referred to as bisimulations between $\mathbb{G}$ and $\mathbb{G}'$ $\lhd$

In particular, this means that for any such bisimulation R and every two vertices $v \in \mathbb{V}$ and $v' \in \mathbb{V}'$ with $(v, v') \in R$ their classification as 0-vertices, 1-vertices or special vertices coincides, i.e., $v \in \mathbb{V}_0 \Leftrightarrow v' \in \mathbb{V}'_0$ and $v \in \mathbb{F} \Leftrightarrow v' \in \mathbb{F}'$ holds.

Lemma 3.29. *Let $\mathbb{G} = (\mathbb{V}_0, \mathbb{V}_1, v_0, \mathbb{E}, \mathrm{Cond}(\mathbb{F}))$ and $\mathbb{G}' = (\mathbb{V}'_0, \mathbb{V}'_1, v'_0, \mathbb{E}', \mathrm{Cond}(\mathbb{F}'))$ be two bisimilar two-player games. Then Player 0 has a winning strategy in $\mathbb{G}$ if and only if Player 0 has a winning strategy in $\mathbb{G}'$.*

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

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

The construction will ensure that, for all n,

- $\mathfrak{h}$ is defined for all paths of length $n+1$ consistent with f' such that the image is consistent with f,

- the states are *pairwise bisimilar*, i.e., if $\mathfrak{h}(v'_0 \cdots v'_n) = v_0 \cdots v_n$ then $(v_j, v'_j) \in R$ for all $0 \leq j \leq n$,
- f' is defined for all consistent paths of length n that end in a state of V'_0.

(IB) Consider the case $n = 0$. The only path of length $n + 1 = 1$ in $\mathbb{G}'$ is v'_0. Define $\mathfrak{h}(v'_0) := v_0$, and f' maps the empty path to v_0.
(IH) Assume now, for an arbitrary n, that f' is defined for all consistent paths of length n and $\mathfrak{h}$ is defined for all paths of length $n + 1$ consistent with f'.
(IS) Consider a path $v'_0 \cdots v'_n$ of length $n + 1$ in $\mathbb{G}'$ that is consistent with f'. Let $v_0 \cdots v_n = \mathfrak{h}(v'_0 \cdots v'_n)$.
Case $v'_n \in \mathbb{V}'_0$. We define $f'(v'_0 \cdots v'_n)$ as follows: since $(v_n, v'_n) \in R$, we have that $v_n \in \mathbb{V}_0$. Let now $v_{n+1} = f(v_0 \cdots v_n)$. This implies $(v_n, v_{n+1}) \in \mathbb{E}$ and therefore, since R is a bisimulation, there is a $v'_{n+1} \in \mathbb{V}'$ such that $(v'_n, v'_{n+1}) \in \mathbb{E}'$ and $(v_{n+1}, v'_{n+1}) \in R$. We define $f'(v'_0 \cdots v'_n) := v'_{n+1}$ and $\mathfrak{h}(v'_0 \cdots v'_n v'_{n+1}) := v_0 \cdots v_n v_{n+1}$.
Case $v'_n \in \mathbb{V}'_1$. We define, for every $v' \in \mathbb{V}'$ with $(v'_n, v') \in \mathbb{E}'$, $\mathfrak{h}(v'_0 \cdots v'_n v') = v_0 \cdots v_n v$ for an arbitrary v such that $(v_n, v) \in \mathbb{E}$ and $(v, v') \in R$.

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

Since the inverse relation $R^{-1} = \{(v', v) \mid (v, v') \in R\}$ is a bisimulation between $\mathbb{G}'$ and $\mathbb{G}$, the converse direction follows analogously. $\square$

Note that we have *not* shown how to create a *positional* strategy in $\mathbb{G}'$ from a *positional* strategy in $\mathbb{G}$. I suspect that this is not possible without further analysis of the strategies since the given bisimulation between $\mathbb{G}$ and $\mathbb{G}'$ could be a many-to-one relation. With this, we can have multiple possibilities for defining positional strategies in $\mathbb{G}'$. This is illustrated in the following example.

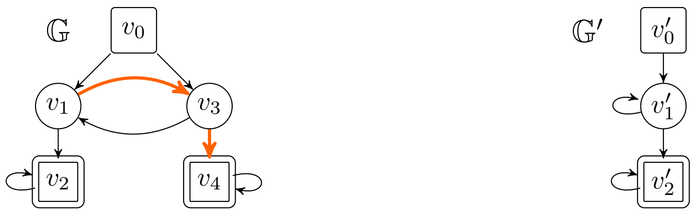


Figure 3.12: Two bisimilar two-player games $\mathbb{G}$ and $\mathbb{G}'$, with a positional strategy for Player 0 in $\mathbb{G}$.

Consider the two reachbility games $\mathbb{G}$ and $\mathbb{G}'$ from Fig. 3.12 and a bisimulation $R = \{(v_0, v'_0), (v_1, v'_1), (v_3, v'_1), (v_2, v'_2), (v_4, v'_2)\}$ relating the vertices respecting their layer position. Let the round vertices belong to Player 0. A positional winning strategy f in $\mathbb{G}$ is depicted as red, thick edges. Constructing a positional winning strategy f' in $\mathbb{G}'$, requires to define $f'(v'_1)$. Hence, we have to decide whether $f'(v'_1) = v'_1$ (considering v_1) or $f'(v'_1) = v'_2$ (considering v_3) should hold. The problem here is that R relates v_1 and

v_3 to the same vertex (v'_1), whereas the successors $f(v_1)$ and $f(v_3)$ are related by R to different vertices (v'_1 and v'_2).

We expect that we can always generate *another* positional winning strategy $\hat{f}$ in $\mathbb{G}$ which is compatible with the bisimulation (here $\hat{f}(v_1) = v_2$ and $\hat{f}(v_3) = v_4$), and therewith easily construct a positional winning strategy for $\mathbb{G}'$. However, this would probably be as complicated as the proof of Lemma 3.29, so I will not explore this possibility here.

Remark. Notice that we have not talked about possibly labeled edge relations in bisimilar two-player games. Concerning bisimulations, we completely ignore the labels on edges in such a case. This in particular means that labels do not play any role in the second and third condition of bisimulations.

In the next section we show that the symbolic two-player game $\overline{\mathbb{G}}(G)$ is bisimilar to the two-player game $\mathbb{G}(\mathsf{G})$, where $\mathsf{G} = \mathrm{Exp}(G)$. In this case, however, we have a *one-to-many* relation between $\overline{\mathbb{G}}(G)$ and $\mathbb{G}(\mathsf{G})$. This allows us to translate a positional winning strategy in $\overline{\mathbb{G}}(G)$ to a positional winning strategy in $\mathbb{G}(\mathsf{G})$, as we demonstrate in Construction 3.35.

3.2.5 Soundness of the Symbolic Two-Player Game

In this section we apply the theory from Sec. 3.2.4 to the two-player games $\mathbb{G}(\mathsf{G})$ from Sec. 3.1.3 and $\overline{\mathbb{G}}(G)$ from Sec. 3.2.3. We again assume a given high-level Petri game $G \in \mathbf{G}$, and denote its extension by $\mathsf{G} = \mathrm{Exp}(G) = (\mathsf{P}_\mathrm{S}, \mathsf{P}_\mathrm{E}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Obj}, \mathsf{P}_{\circledast}) \in \mathrm{Exp}(\mathbf{G})$. To show that the two-player games $\overline{\mathbb{G}}(G)$ and $\mathbb{G}(\mathsf{G})$ are bisimilar, we first compare the edge relation in the two. As we have seen, the edges are built from properties of and the relations between (symbolic) decision sets. Thus, Corollary 3.17, and the Properties 3.22 and 3.24, will be the main arguments in comparing the games' structures.

First we show that the edge relations of the two underlying graphs of $\mathbb{G}(\mathsf{G})$ and $\overline{\mathbb{G}}(G)$ correspond to each other (Lemma 3.30). Second we prove that the set of reachable symbolic decision sets in $\overline{\mathbb{G}}(G)$ is exactly the set of representatives of decision sets reachable in $\mathbb{G}(\mathsf{G})$ (Lemma 3.31). Let in this section $\mathbb{G}(\mathsf{G}) = (\mathbb{V}_0, \mathbb{V}_1, v_0, \mathbb{E}, \mathrm{Win})$ and $\overline{\mathbb{G}}(G) = (\overline{\mathbb{V}}_0, \overline{\mathbb{V}}_1, \overline{v}_0, \overline{\mathbb{E}}, \overline{\mathrm{Win}})$.

Lemma 3.30. *For every edge in $\mathbb{G}(\mathsf{G})$ there is a "corresponding" edge in $\overline{\mathbb{G}}(G)$, and vice versa, meaning:*

(1) $(\mathsf{D}, \tau, \mathsf{D}') \in \mathbb{E} \quad \Rightarrow \quad (\overline{\mathsf{D}}, \overline{\tau}, \overline{\mathsf{D}'}) \in \overline{\mathbb{E}},$ *where*
- *if $\tau = t.\sigma$ then $\overline{\tau} = t.\overline{\sigma}$, where $\overline{\sigma} = \alpha_{\overline{D}}(s_D(\sigma))$, and*
- *if $\tau = \dagger$ then $\overline{\tau} = \dagger$.*

(2) $(\overline{\mathsf{D}}, \overline{\tau}, \overline{\mathsf{D}'}) \in \overline{\mathbb{E}} \quad \Rightarrow \quad \forall \mathsf{D}_1 \in [\,\overline{\mathsf{D}}\,]\ \exists \mathsf{D}_2 \in [\,\overline{\mathsf{D}'}\,]\ \exists \tau : \ (\mathsf{D}_1, \tau, \mathsf{D}_2) \in \mathbb{E},$ *where*
- *if $\overline{\tau} = t.\overline{\sigma}$ then $\tau = t.\sigma'$ for a σ' satisfying $\alpha_{\overline{\mathsf{D}}}(s_{\mathsf{D}_1}(\sigma')) = \overline{\sigma}$, and*
- *if $\overline{\tau} = \dagger$ then $\tau = \dagger$.*

Proof. By Corollary 3.17, a decision set D contains a bad place, is a deadlock, is terminating, or nondeterministic, if and only if $\overline{\mathsf{D}}$ has the same property. In this case there is only a $\dagger$-labeled self-loop originating from D as well as from $\overline{\mathsf{D}}$.

Now consider the cases in which D resp. $\overline{\mathsf{D}}$ has none of these properties. We prove *(1)* and *(2)* in (1) and (2), respectively.

(1) Let $(\mathsf{D}, \tau, \mathsf{D}') \in \mathbb{E}$. We consider the three cases from the construction of the edge relation $\mathbb{E}$ in Def. 3.10.

Case $\mathsf{D} \in \mathbb{V}_1$: Then $\overline{\mathsf{D}} \in \overline{\mathbb{V}}_1$ by Corollary 3.17. Since $\mathsf{D} \in \mathbb{V}_1$ and D is not a deadlock, $\exists t.\sigma \in \mathsf{T} : \tau = t.\sigma \wedge \mathsf{D}[t.\sigma\rangle\mathsf{D}'$. Property 3.22 implies $\overline{\mathsf{D}}[\![t.\overline{\sigma}\rangle\!\rangle\overline{\mathsf{D}'}$ for $\overline{\sigma} = \alpha_{\overline{\mathsf{D}}}(s_{\mathsf{D}}(\sigma))$, and then by definition $(\overline{\mathsf{D}}, t.\overline{\sigma}, \overline{\mathsf{D}'}) \in \overline{\mathbb{E}}$.

Case $\mathsf{D} \in \mathbb{V}_0$ and $\mathsf{D}[\dagger\rangle$: Then $\overline{\mathsf{D}} \in \overline{\mathbb{V}}_0$ by Corollary 3.17, and by definition $\overline{\mathsf{D}}[\![\dagger\rangle\!\rangle$. Since $\mathsf{D}[\dagger\rangle$, we have $\tau = \dagger$ and $\mathsf{D}[\dagger\rangle\mathsf{D}'$. Property 3.22 implies $\overline{\mathsf{D}}[\![\dagger\rangle\!\rangle\overline{\mathsf{D}'}$, and then by definition $(\overline{\mathsf{D}}, \dagger, \overline{\mathsf{D}'}) \in \overline{\mathbb{E}}$.

Case $\mathsf{D} \in \mathbb{V}_0$ and $\neg\mathsf{D}[\dagger\rangle$: Works the same as the case $\mathsf{D} \in \overline{\mathbb{V}}_1$, only that we also use the simple fact that $\mathit{pre}(t.\sigma) \cap \mathsf{P}_{\mathrm{E}} = \emptyset \Rightarrow \mathit{pre}(t.\overline{\sigma}) \cap \mathsf{P}_{\mathrm{E}} = \emptyset$.

(2) This works analogously to (1), but considering the three cases in the edge relation $\overline{\mathbb{E}}$ (Def 3.26), and using Property 3.24 instead of Property 3.22. We demonstrate this only for the first case:

Let $(\overline{\mathsf{D}}, \overline{\tau}, \overline{\mathsf{D}'}) \in \overline{\mathbb{E}}$ and $\overline{\mathsf{D}} \in \overline{\mathbb{V}}_1$. Corollary 3.17 gives us $\mathsf{D} \in \mathcal{V}_1$. Since $\overline{\mathsf{D}} \in \overline{\mathbb{V}}_1$, and $\overline{\mathsf{D}}$ is not a deadlock, $\exists t.\overline{\sigma} \in \mathsf{T} : \overline{\tau} = t.\sigma \wedge \overline{\mathsf{D}}[\![t.\overline{\sigma}\rangle\!\rangle\overline{\mathsf{D}'}$. Let now $\mathsf{D}_1 \in [\overline{\mathsf{D}}]$. Property 3.24 implies that there is a σ' with $\overline{\sigma} = \alpha_{\overline{D}}(s_D(\sigma))$ and a $\mathsf{D}_2 \in [\overline{\mathsf{D}'}]$ s.t. $\mathsf{D}_1[t.\sigma'\rangle\mathsf{D}_2$, which by definition implies $(\mathsf{D}_1, t.\sigma', \mathsf{D}_2) \in \mathbb{E}$. □

By definition of $\mathbb{V}$ and $\overline{\mathbb{V}}$, we have that the symbolic decision sets in $\overline{\mathbb{V}} = \mathcal{D}(G)/S$ are the equivalence classes of decision sets in $\mathbb{V} = \mathcal{D}(G) = \mathcal{D}(\mathsf{G})$. We often implicitly restrict the sets of vertices $\mathbb{V}$ (resp. $\overline{\mathbb{V}}$) in $\mathbb{G}(\mathsf{G})$ (resp. $\overline{\mathbb{G}}(G)$) the ones reachable from D_0 (which is equal to $\overline{\mathsf{D}_0}$). This does not have any consequences for a game's semantics, since plays are defined to start in in the initial vertex. We now prove that, with the restriction to reachable vertices, $\overline{\mathbb{V}}$ still contains exactly the equivalence classes of $\mathbb{V}$. For that, let $\mathcal{R}(\mathbb{G}(\mathsf{G}))$ and $\mathcal{R}(\overline{\mathbb{G}}(G))$ be the reachable vertices in $\mathbb{G}(\mathsf{G})$ and $\overline{\mathbb{G}}(G)$, respectively.

Lemma 3.31. *The representatives of decision sets in* $\mathcal{R}(\mathbb{G}(\mathsf{G}))$ *are exactly the symbolic decision sets in* $\mathcal{R}(\overline{\mathbb{G}}(G))$*, i.e.,* $\{\overline{\mathsf{D}} \mid \mathsf{D} \in \mathcal{R}(\mathbb{G}(\mathsf{G}))\} = \{\overline{\mathsf{D}} \mid \overline{\mathsf{D}} \in \mathcal{R}(\overline{\mathbb{G}}(G))\}$.

Proof. We start with $\{\overline{\mathsf{D}} \mid \mathsf{D} \in \mathcal{R}(\mathbb{G}(\mathsf{G}))\} \subseteq \{\overline{\mathsf{D}} \mid \overline{\mathsf{D}} \in \mathcal{R}(\overline{\mathbb{G}}(G))\}$. For all $\mathsf{D} \in \mathcal{R}(\mathbb{G}(\mathsf{G}))$ there are $\mathsf{D}_1, \ldots, \mathsf{D}_k \in \mathbb{V}$ and $\tau_0, \ldots, \tau_k \in \mathsf{T}^{\mathsf{G}} \cup \{\dagger\}$ such that

$$(\mathsf{D}_i, \tau_i, \mathsf{D}_{i+1}) \in \mathbb{E} \text{ for all } i = 0, \ldots, k, \text{ with } \mathsf{D}_{k+1} = \mathsf{D}.$$

We prove $\overline{\mathsf{D}} \in \mathcal{R}(\overline{\mathbb{G}}(G))$ by induction over the length of shortest path from D_0 to a decision set $\mathsf{D} \in \mathcal{R}(\mathbb{G}(\mathsf{G}))$: If the shortest path is empty we have $\mathsf{D} = \mathsf{D}_0$, and $\overline{\mathsf{D}_0} \in \mathcal{R}(\overline{\mathbb{G}})$. The induction step follows by Lemma 3.30 *(1)*.
To show $\{\overline{\mathsf{D}} \mid \overline{\mathsf{D}} \in \mathcal{R}(\overline{\mathbb{G}})\} \subseteq \{\overline{\mathsf{D}} \mid \mathsf{D} \in \mathcal{R}(\mathbb{G}(\mathsf{G}))\}$, we proceed analogously, with the induction step following by Lemma 3.30 *(2)*. □

Lemma 3.32. *There is a bisimulation between* $\mathbb{G}(\mathsf{G})$ *and* $\overline{\mathbb{G}}(G)$.

Proof. Note that in this lemma and proof we explicitly use the notation of equivalence classes instead of their representatives for elements in $\overline{\mathbb{V}}$, so that no confusion arises. By definition of $\mathsf{G} = \text{Exp}(G)$ we have that the system players in G have the same kind of objective (safety resp. reachability) as in G. Thus, also by definition, the two-player games $\mathbb{G}(\mathsf{G})$ and $\overline{\mathbb{G}}(G)$ have the same winning condition (Cond $=$ Safety resp. Cond $=$ Reach) and thus satisfy the basic requirement for a bisimulation.

Let now $R = \{(\mathsf{D}, [\mathsf{D}]) \mid \mathsf{D} \in \mathbb{V}\}$. This relation is defined on $\mathbb{V} \times \overline{\mathbb{V}}$ by Lemma 3.31, from $\mathsf{D} \in \mathbb{V}$ it follows $\overline{\mathsf{D}} \in \overline{\mathbb{V}}$, and we have $(\mathsf{D}_0, [\mathsf{D}_0]) \in R$. We show that R is a bisimulation.

Let $(\mathsf{D}, [\mathsf{D}]) \in R$ and $(\mathsf{D}, \mathsf{D}') \in \mathbb{E}$ for a $\mathsf{D} \in \mathbb{V}$. We have to prove that there is a $[\mathsf{D}_2]$ such that $([\mathsf{D}], [\mathsf{D}_2]) \in \overline{\mathbb{E}}$ and $(\mathsf{D}', [\mathsf{D}_2]) \in R$. The obvious candidate for $[\mathsf{D}_2]$ is $[\mathsf{D}_2] = [\mathsf{D}']$, since $(\mathsf{D}', [\mathsf{D}']) \in R$ by definition. From $(\mathsf{D}, \mathsf{D}') \in \mathbb{E}$ follows $\exists \tau \in \mathsf{T}^{\mathsf{G}} \cup \{\dagger\} : (\mathsf{D}, \tau, \mathsf{D}') \in \mathbb{E}$. Then, by Lemma 3.30 *(1)*, we have $(\overline{\mathsf{D}}, \overline{\tau}, \overline{\mathsf{D}'}) \in \overline{\mathbb{E}}$. This, again by definition, gives $([\mathsf{D}], [\mathsf{D}']) \in \overline{\mathbb{E}}$.

We can prove $(\mathsf{D}, [\mathsf{D}]) \in R \wedge ([\mathsf{D}], [\mathsf{D}']) \in \overline{\mathbb{E}} \Rightarrow \exists \mathsf{D}_2 \in \mathbb{V} : (\mathsf{D}_2, [\mathsf{D}']) \in R \wedge (\mathsf{D}, \mathsf{D}_2) \in \mathbb{E}$ analogously, by using Lemma 3.30 *(2)*.

Since, by Lemma 3.31 and Corollary 3.17, $\mathsf{D} \in \mathbb{V}_0 \Leftrightarrow [\mathsf{D}] \in \overline{\mathbb{V}}_0$ and $\mathsf{D} \in \mathbb{F} \Leftrightarrow [\mathsf{D}] \in \overline{\mathbb{F}}$, we finally have that R is a bisimulation between $\mathbb{G}$ and $\overline{\mathbb{G}}$. □

Lemma 3.29 together with Lemma 3.32 finally yields the conformity of the symbolic high-level game $\overline{\mathbb{G}}(G)$ and the corresponding low-level game $\mathbb{G}(\mathsf{G})$ regarding the existence of a winning strategy:

Corollary 3.33. *Player 0 has a winning strategy in $\overline{\mathbb{G}}(G)$ if and only if Player 0 has a winning strategy in $\mathbb{G}(\mathsf{G})$.*

Since the definition of a winning strategy in a high-level Petri game G is defined as a winning strategy in the corresponding low-level Petri game $\text{Exp}(G)$ (cp. Sec. 2.3.2), Cor. 3.33 yields the main result of this section: Theorem 3.34 states that the construction of the symbolic two-player game is *sound* in the sense that solving the symbolic two-player game answers the question whether there exists a winning strategy for the system players in a given proper high-level Petri game.

Theorem 3.34. *Let $G \in \mathbf{G}$ be a proper high-level Petri game. Then the system players have a winning strategy in G if and only if Player 0 has a winning strategy in the corresponding symbolic two-player game $\overline{\mathbb{G}}(G)$.*

We construct a winning strategy for the system players in G, i.e., a winning strategy for the system players in the corresponding low-level Petri game $\mathsf{G} = \text{Exp}(G)$, from the positional winning strategy $\overline{f}$ for Player 0 in $\overline{\mathbb{G}}(G)$ in two steps. First, we create a positional winning strategy f for Player 0 in $\mathbb{G}(\mathsf{G})$ from $\overline{f}$, which is possible as given by Lemma 3.29. Second, we apply the algorithm presented in [FO17; Gie22] to f, i.e., traversing f in breadth-first order while adding the corresponding places and transition of the decision sets, to create a winning strategy for the system players in G. This algorithm was demonstrated in Example 3.13 and can also be found in Appendix 3.A. Note that this last step would take infinitely long for infinite Petri game strategies such that a practical algorithm has to provide a finite representation of the strategy, as demonstrated in [FO17]. The procedure that we just described corresponds to taking the thick edges in Fig. 3.8.

In Fig. 3.8, consider now the edge "generate" from a strategy $\overline{f}$ in the symbolic two player game $\overline{\mathbb{G}}(G)$ to a f in the two-player game $\mathbb{G}(\mathsf{G})$. The construction of a winning strategy in the proof of Lemma 3.29 yields a *nonpositional* strategy $f : \mathbb{V}^*\mathbb{V}_0 \to \mathbb{V}$ in $\mathbb{G}(\mathsf{G})$ for a such strategy $\overline{f}$. For the introduced solving algorithm of high-level Petri games we are interested in *positional* winning strategies. The following construction serves for the creation of a *positional* strategy $f : \mathbb{V}_0 \to \mathbb{V}$ in the two-player game $\mathbb{G}(\mathsf{G})$ from a positional strategy $\overline{f} : \overline{\mathbb{V}}_0 \to \overline{\mathbb{V}}$ in the symbolic two-player game $\overline{\mathbb{G}}(G)$. This is possible since our introduced bisimulation is a many-to-one relation.

Construction 3.35. Let $R^{-1} = \{(\overline{\mathsf{D}}, \mathsf{D}) \mid \mathsf{D} \in \mathbb{V}\}$ be the inverse of the bisimulation R on $\mathbb{G}(\mathsf{G})$ and $\overline{\mathbb{G}}(G)$. This bisimulation is a one-to-many relation with domain $\overline{\mathbb{G}}(G)$ and co-domain $\mathbb{G}(\mathsf{G})$. Let $\overline{f} : \overline{\mathbb{V}}_0 \to \overline{\mathbb{V}}$ be a positional winning strategy for Player 0 in $\overline{\mathbb{G}}(G)$ and $\mathsf{D} \in \mathbb{V}_0$. Then $\overline{\mathsf{D}} \in \overline{\mathbb{V}}_0$ and $\overline{f}(\overline{\mathsf{D}})$ is defined. Let $\overline{\mathsf{D}'} := \overline{f}(\overline{\mathsf{D}})$. This implies $(\overline{\mathsf{D}}, \overline{\mathsf{D}'}) \in \overline{\mathbb{E}}$, and since $(\overline{\mathsf{D}}, \mathsf{D}) \in R^{-1}$, we have

$$\exists \mathsf{D}_1 \in \mathbb{V} : (\mathsf{D}, \mathsf{D}_1) \in \mathbb{E} \wedge (\overline{\mathsf{D}'}, \mathsf{D}_1) \in R^{-1} \qquad \text{(i.e., } \overline{\mathsf{D}_1} = \overline{\mathsf{D}'}\text{)}.$$

We define $f(\mathsf{D}) := \mathsf{D}_1$. Hence, f is positional. The strategy f is also winning: Let $\rho = \mathsf{D}_0\mathsf{D}_1 \cdots \in (\mathbb{V})^\omega$ be a play of $\mathbb{G}$ that is consistent with f. Then $\overline{\rho} := \overline{\mathsf{D}_0}\,\overline{\mathsf{D}_1} \cdots \in (\overline{\mathbb{V}})^\omega$ is a play of $\overline{\mathbb{G}}(G)$, that by definition is consistent with $\overline{f}$. Therefore, $\overline{\rho}$ is winning in $\overline{\mathbb{G}}(G)$. This, as in the proof of Lemma 3.29, implies that ρ is winning in $\mathbb{G}(\mathsf{G})$. ◁

3.2.6 Experimental Results

This section briefly presents a prototype implementation for generating the symbolic two-player game $\overline{\mathbb{G}}(G)$ for a given proper high-level Petri game G. The primary work on this implementation was conducted by Manuel Gieseking in the context of [GOW20], and this section only serves as a showcase for the applicability of the presented concepts. The implementation comprises three algorithms to create the reduced, symbolic state space and compared their runtimes with the complete state space creation of $\mathbb{G}(\mathsf{G})$ in Adam [FGO15; Gie20], where $\mathsf{G} = \text{Exp}(G)$. The results of these benchmarks are shown in Table 3.1. All algorithms are integrated into the Adam framework, allowing for leveraging its data structures and functionalities for Petri nets and Petri games.

Adam employs Binary Decision Diagrams (BDDs) to determine the existence of a strategy and to compute a strategy when it exists. In the original algorithm, the explicit state space is never generated, which means the concrete size of $\mathbb{G}(\mathsf{G})$ cannot be directly obtained. To facilitate a proper comparison between the sizes of the generated state spaces ($\overline{\mathbb{G}}(G)$ versus $\mathbb{G}(\mathsf{G})$), Adam is extended with a fixed-point algorithm. This algorithm calculates a BDD for the reachable states of the two-player game $\mathbb{G}(\mathsf{G})$ and then counts the number of solutions to obtain the number of states of $\mathbb{G}(\mathsf{G})$ as a reference value. The *Reference-Approach* takes the low-level version G of a high-level Petri game G

as input. The results and the computational resources used for calculating the reduced state space are presented in columns three and four of Table 3.1.

To generate the reduced state space, Symmetric Nets [CDFH91a] (cp. Sec. 3.3.1) are utilized as the underlying structure for the high-level Petri game. Symmetric nets are a subclass of high-level Petri nets with equivalent expressive power but offer the advantage of allowing easy and automatic creation of the system's symmetries from the specification.

The following three algorithms are all based on the algorithm for a symbolic reachability tree originally presented in [HJJJ86]:

HL-Approach: This approach explicitly calculates the symbolic state space from the high-level Petri game G.

LL-Approach: This approach first expands the high-level Petri game G into the corresponding low-level Petri game $\mathsf{G} = \mathrm{Exp}(G)$ and then uses this to explicitly calculate the symbolic state space of $\overline{\mathbb{G}}(G)$. During this calculation, G is still exploited to obtain the symmetries of the system.

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

Note that neither a winning strategy is calculated nor its existence is determined. Instead, for a high-level Petri game, three different approaches to compute the reduced state space, which corresponds to determining the number of vertices in the high-level two-player game $\overline{\mathbb{G}}(G)$ are employed. These approaches leverage the symmetries of the high-level Petri game, as described in Sec. 3.2.3. Additionally, the adapted algorithm of Adam is used to compare the sizes of these reduced state spaces to the size of the previously existing low-level two-player game $\mathbb{G}(\mathrm{Exp}(G))$. The runtime of any synthesis algorithm is significantly influenced by the size of the state space it needs to explore. Therefore, this comparison provides an initial indication of the potential of our new method.

The algorithms were evaluated on a set of five scalable benchmark families, which encompass applications in robotic control, workflow management, and other distributed domains. For each benchmark, Table 3.1 presents the elapsed CPU time (*time* in seconds) for calculating the size of the state space $|\mathbb{V}|$ and the size of the reduced state space $|\overline{\mathbb{V}}|$ using each approach. If a calculation exceeds two hours, it is indicated as a timeout (TO). For each benchmark, the time of the fastest among the new approaches is highlighted in bold. The experiments were conducted on an Intel i7-2700K CPU with 3.50 GHz and 32GB RAM. The benchmark families describe the following scenarios:

Package Delivery (PD): In this scenario, we have n drones tasked with delivering m packages. The packages are assigned to the drones. However, due to the hostile environment, an arbitrary drone may crash during the delivery process. When a

Table 3.1: Experimental results of the benchmark families regarding the sizes of $\mathbb{G}(\mathsf{G})$ and $\overline{\mathbb{G}}(G)$ and their calculation time (in seconds) for the three different approaches for $\overline{\mathbb{G}}(G)$ and the reference approach for $\mathbb{G}(\mathsf{G})$.

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

drone crashes, other drones can get notified and can decide whether to recover the package or not. The ultimate goal of the system is to successfully deliver all the packages despite the potential crashes. *Parameters:* n drones / m packages.

Alarm System (AS): In this scenario, we have n geographically distributed locations, each of which is protected by an alarm system. The environment includes a burglar who can intrude into any of the locations. The alarm systems have the capability to communicate with each other about burglaries. The system's objective is twofold: to ensure that no alarm system is triggered without an actual intrusion, and that all alarm systems correctly indicate the location of the intrusion in case of a burglary. *Parameters:* n alarm systems.

Concurrent Machines (CM): In this scenario, we have n machines tasked with processing m orders. The orders can be processed concurrently, but each machine is limited to handling only one order. The hostile environment chooses one machine to be defective. Despite this challenge, the system's goal is to ensure that all orders are processed successfully in the end. *Parameters:* n machines / m orders.

Document Workflow (DW) and (DWs): In this scenario, there are n clerks responsible for either endorsing or rejecting a document. The document is passed in a circular manner among the clerks, and the environment decides which clerk receives the document first. The objective is to achieve a unanimous decision among all clerks. In the simplified variant DWs, the specific goal is for all clerks to endorse the document. *Parameters:* n clerks.

The package delivery benchmark family is originally described in [GOW20]. The alarm system benchmark family was first introduced in [FGHO17], and its high-level version was presented in [GO21]. The benchmark families CM, DW, and DWs were initially introduced in [FGO15] at the P/T level. The high-level version for CM was later presented in [GO21], and the high-level versions for DW and DWs were first defined for the benchmarks presented in [GOW20].

The figures demonstrate a substantial decrease in the size of the system's state space. The new benchmark PD, with parameters 1/8, shows the most significant reduction: from 26,299,378 states for the standard state space to 11,115 states for the reduced one. This amounts to a reduction factor of approximately 2,366. As for the DW and DWs benchmark family, the reduction is relatively smaller. This is due to the circular passing of the document, which limits the permissible symmetries to rotations, resulting in less potential for significant reductions.

The reduction in the state space does come with a trade-off. The computation time of the new algorithms for the reduced state space (the last three columns) is generally noticeably higher compared to the reference algorithm for the standard state space (column three). One reason is the equivalence check is performed each time before adding a new vertex, which adds to the computational overhead. However, one can argue that these figures are not directly comparable because Adam utilizes optimized symbolic algorithms that typically outperform explicit algorithms, such as those used in the HL- and the LL-Approach, especially for large state spaces.

The primary reason for the low performance of the BDD-Approach on larger models is that the current algorithm requires checking, for each newly created state, whether an equivalent state already exists. Unfortunately, the existing framework did not allow to directly encode this check into a Boolean function for representing the transition relation of the two-player game. As a result, in this prototype implementation of a symbolic algorithm that exploits the symmetries of the system, the equivalence check is performed explicitly. This means that in every round of the fixed point calculation, each explicit state of the Binary Decision Diagram (BDD) representing the successors of this round is calculated. These costly solving steps of the BDDs hinder the efficient use of a symbolic algorithm. As a consequence, the BDD-Approach exhibits lower performance on larger models.

In general, the LL-Approach outperforms the HL-Approach. This can be attributed to the structure of the decision sets. In the HL-Approach, a decision set of the high-level two-player game consists of concrete instances of the places and transitions from the high-level net. Consequently, the HL-Approach recalculates these instances each time a high-level transition is requested. Although an improvement can be made by buffering this data, it nearly leads to the LL-Approach in terms of efficiency.

Overall, these results indicate a significant advancement towards a faster practical solving of Petri games, as a smaller state space considerably reduces the runtime of the synthesis algorithms. Standard algorithms used for solving two-player games with complete information are polynomial in the number of edges of the game and can be applied to the symbolic two-player game $\overline{\mathbb{G}}(G)$. The remaining steps for solving high-level Petri games, namely, resolving the symmetries of the two-player strategy and creating the Petri game strategy, are linear in the number of edges of the strategy and quadratic in the number of admissible symmetries. Considering that the presented algorithms are still in a prototype stage, these results are highly promising and encouraging for further research and development.

3.3 Small, Canonical Representations in the Symbolic Two-Player Game

Before we go into the content of this section, we begin with a remark concerning terminology.

Remark. In the previous section, we discussed the concept of *(arbitrary) representat**ives***. This section, however, focuses on what we refer to as *(dynamic) representat**ions***. Although both terms share the common objective of representing an equivalence class of decision sets, they achieve this goal through different approaches. Therefore, we frequently draw comparisons between the two concepts. The aim of this remark is to highlight this subtle but crucial difference in terminology.

The idea of the symbolic two-player game discussed in the preceding sections is to reduce the state space's size by exploiting symmetries. To achieve this, we consider only one representative from each equivalence class induced by a Petri game's symmetries. This can be done either by checking whether a newly generated state is equivalent to any already generated one or by transforming each newly generated state into an equivalent, *canonical representation*.

These two approaches solve the so-called *Orbit Problem* and *Constructive Orbit Problem* [CEJS98], respectively. The Orbit Problem asks, for two given states, if the two are symmetric. The Constructive Orbit Problem asks, given a state, for a canonical representation that is equal for all symmetric states. Solving the Constructive Orbit Problem also solves the Orbit Problem, since two states are symmetric if and only if they have the same canonical representation.

In the preceding section, we explored the former approach. The vertices of the symbolic two-player game are symbolic decision sets. During the game's construction, an *arbitrary* representative $\overline{\mathsf{D}}$ is chosen for each of these equivalence classes. Usually, the first member of an equivalence class that is added to the game is declared the representative of that class. Consequently, when encountering a new vertex D', we must apply every admissible symmetry $s \in S$ to verify whether a representative $\overline{\mathsf{D}'} = s(\mathsf{D}')$ already exists in the game, or if D' belongs to a new symbolic decision set. We therefore solved the *Orbit Problem*.

Our focus now shifts to the second approach, wherein we introduce the concept of *dynamic representations* for symbolic decision sets. Once the dynamic representations are defined, we show how to select a canonical dynamic representation for each symbolic decision set. This selected representation is referred to as the *canonical representation*. These canonical representations will be employed in place of arbitrary representatives of symbolic decision sets during the construction of the symbolic two-player game. By defining such a canonical representation for every symbolic decision set we solve the *Constructive Orbit Problem*.

We employ techniques presented in [CDFH91a; CDFH93], where the authors address the constructive orbit problem for markings in the symbolic reachability graph of a high-level Petri net. The considered nets are given in a formalism known as *symmetric nets*. One major advantage of this formalism is that the symmetries of a net can be directly inferred from its specification, avoiding the necessity of further behavioral analysis. Additionally, it allows for the definition of *small* dynamic representations of markings, and facilitates the construction of a canonical representation by considering only a subset of the net's symmetries.

Thus, in this section, we consider high-level Petri games G with an underlying symmetric net. We extend the findings of [CDFH91a; CDFH93] from markings to decision sets, demonstrating that the notion of canonical representations is consistent with the symbolic two-player game. Consequently, we are able to construct a *canonical two-player game* $\widehat{\mathbb{G}}(G)$ that is isomorphic to the symbolic two-player game $\overline{\mathbb{G}}(G)$, but with canonical representations of symbolic decision sets serving as vertices. Fig. 1.1, illustrates the role of $\overline{\mathbb{G}}(G)$ in the frame work of the symbolic two-player game $\overline{\mathbb{G}}(G)$.

3.3.1 Symmetric Nets and Symmetric Petri Games

In this section, we revisit the definition of the symmetric net formalism as presented in [CDFH91a; CDFH93], and draw comparisons with the high-level formalism introduced in Sec. 2.3.1. Before this comparison, we offer a brief excursus to present another high-level Petri net formalism frequently referenced in the literature, such as [Jen96]. This particular formalism can be regarded as an intermediary step between Sec. 2.3.1 and symmetric nets. Its presentation, however, is solely aimed at enhancing the reader's comprehension of the symmetric net formalism, as all three formalisms have the same expressive power.

Excursus: Place Types and Arc Expressions in High-level Petri Nets. Our definitions in Sec. 2.3.1 of high-level Petri nets are based on [CJ04]. By choosing this formalism, we were able to adopt the notion of symbolic unfoldings, introduced also in [CJ04]. However, a more common definition of high-level Petri nets can be found, e.g., in [Jen96] (Definition 2.5). There, each place has a *type*, describing which kind of data (colors) the place can hold. Every arc is labeled by an *expression*, possibly containing variables. These expressions evaluate to *multi-sets* over the connected place's type, describing which tokens are placed on or taken from the place when firing the transition in a mode. The modes, as before, assign values to the variables, now inside the expressions.

In our definition, we have a *single set of colors* that can potentially be placed on *every* place. Simulating the type of a place involves using a corresponding guard on each transition connected to it. Additionally, we permit only a single variable on each arc. To simulate expressions, including (multi-)sets on arcs, there are two approaches. The first approach is to introduce multiple arcs, with each arc representing one element of a set. However, this method may not always work when the set's size depends on the evaluation of the expression. The more reliable approach to simulate an expression is to expand the set of (original) colors to include its *subsets* (and possibly multi-sets) and then add the expression to the transition's guard. Therefore, introducing expressions on arcs does not grant us any additional expressive power, but it allows us to potentially model scenarios in a more comprehensive manner.

We demonstrate this on our running example: Figure 3.13 shows the Signal Sending Satellites example from Fig. 3.2 with typed places and arc expressions. The type of each place is indicated next to its name. For example, the place *Init* and the place *Inf* now have the type **Sats** $= \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$. The place *Env* has the type $\{\mathbf{0}\}$, since this is the only color that will ever by placed there. We denote these *color classes* by $\mathscr{C}_1$ and $\mathscr{C}_2$. The rationale behind this (re)naming is to later characterize this high-level Petri net as a *symmetric net*, and will be further expounded in Example 3.37. Having the possibility to label arcs with expressions makes the role of some transitions more clear. Compared to Fig. 3.2 showing the same example in the formalism of Sec. 2.3.1, we have three transitions with changes on arcs or guards:

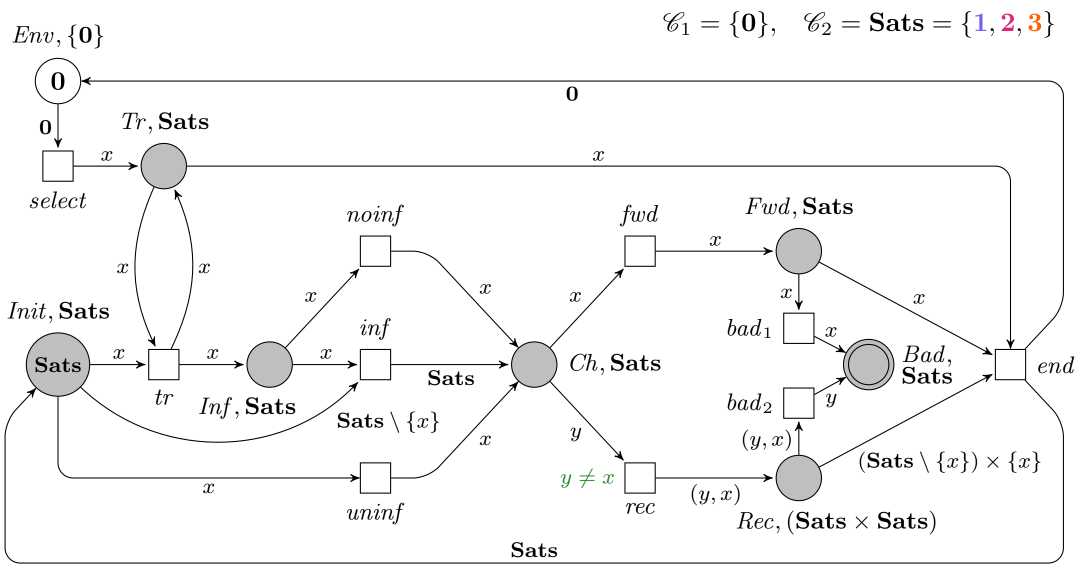


Figure 3.13: The running example Signal Sending Satellites as a high-level Petri game with place types and arc expressions.

- The arc from *Init* to *inf* is labeled by **Sats** $\setminus \{x\}$ instead of y, z. This says the satellite assigned to x (from place *Inf*) informs *all other* satellites by firing *inf*. This firing places *all* satellites on *Ch*, denoted by the arc expression **Sats** on the corresponding edge.
- The guard of *rec* does not contain the predicate "$x \neq \mathbf{0}$". This is not needed, since the type **Sats** × **Sats** of place *Rec* ensures that x (contained in the tuple (y, x)) must be in **Sats** and therefore not **0**.
- The arc from *Rec* to *end* is labeled by (**Sats**$\setminus\{x\}) \times \{x\}$. This again means all other satellites are in Receiving mode, expecting a message from the satellite assigned to x in Forwarding mode (from place *Fwd*).

We see that this formalism enables a more comprehensive expression of net behavior. However, the simpler formalism with single variables on arcs introduced in Sec. 2.3.1 considerably simplifies the definition of symbolic unfoldings for high-level Petri nets. This advantage proves beneficial in subsequent chapters. $\triangleleft$

High-level representations of a given P/T Petri net are often created by implicitly exploiting symmetric/equivalent behavior of components in the original net. This process is also referred to as *folding* the P/T Petri net (cf. e.g. [Jen96]). Conversely, in some high-level nets, symmetries can be read directly from the given specification. A class of nets which allow this are the so called *symmetric nets*[2] introduced in [CDFH91a]. The

[2]Symmetric Nets were formerly known as Well-Formed Nets (WNs). The renaming was part of the ISO standardization [HKPT06].

authors use this formalism to define small canonical representations of equivalence classes of markings in the symbolic reachability graph of a high-level net.

The aim of this Section 3.3 is to extend these results to symbolic decision sets. Thus, we introduce symmetric nets as the high-level Petri net formalism in this section. Choosing symmetric nets as a model does not lead to a loss of generality, and the presented methods work for any high-level Petri net, since symmetric nets have the same modeling power as general high-level Petri nets (cp. Prop. 3.38). Before we give the formal definition of symmetric nets in Def. 3.36, we explain the main ideas of the formalism.

- **Place types from basic color classes.** Symmetric nets are high-level Petri nets with place types and arc expressions as described above. A main concept now is the definition of (finite) *basic color classes* $\mathscr{C}_1, \dots, \mathscr{C}_n$.
  The type $ty(p)$ (here called the "color domain" $\mathscr{C}(p)$) of each place p is then built from these basic color classes as a Cartesian product. Formally, we have, for every place p that $\mathscr{C}(p) = \mathscr{C}_1^{J(p,1)} \times \cdots \times \mathscr{C}_n^{J(p,n)}$ for natural numbers $J(p,1), \dots, J(p,n) \in \mathbb{N}$, where $\mathscr{C}_i^x$ denotes the x-fold Cartesian product of $\mathscr{C}_i$. Every color in the type of p is a tuple of the form $c = ((c_i^j)_{j=1}^{J(p,i)})_{i=1}^n$ with $c_i^j \in \mathscr{C}_i$ for every i, j.
- **Color domains for transitions.** Analogously, there is a color domain $\mathscr{C}(t) = \mathscr{C}_1^{J(t,1)} \times \cdots \times \mathscr{C}_n^{J(t,n)}$ for every transition t. Compared to the definition of high-level Petri nets from Sec. 2.3.1, each entry in a tuple in $\mathscr{C}(t)$ corresponds to the value assigned to one of the variables around t. This means the color domain of a transition describes all possible assignments of colors to variables. The firing modes (i.e., the valuations for which the transition's guard evaluates to *true*) are a subset of t's color domain, i.e., $\Sigma(t) \subseteq \mathscr{C}(t)$. A typical firing mode is $\sigma = ((\sigma_i^j)_{j=1}^{J(t,i)})_{i=1}^n$ with $\sigma_i^j \in \mathscr{C}_i$ for every i, j.
- **Restricted syntax of arc expressions.** Arc expressions and transition guards are constrained to adhere to a structured form known as "standard functions" and "standard predicates". An arc expression can, for example, describe the j-th instance of a color from $\mathscr{C}_i$ in a firing mode σ. This is denoted by X_i^j and refers to the entry σ_i^j. The restricted syntax ensures that all colors in a basic color class are "treated equally". This comes with a benefit explained in the succeeding item.
- **Symmetries without analysis.** One of the main advantages of symmetric nets is that their symmetries can be read directly from the given specification. The ideas of symmetries in symmetric nets is to handle each basic color class independently. A symmetry is a tuple $s = (s_1, \dots, s_n)$, such that every s_i is a permutation on the basic color class $\mathscr{C}_i$. Thus, the definition of basic color classes directly yields the nets symmetries.

  The restricted syntax of guards and arc expressions, treating every basic color equally, now *ensures* that the firing of transitions is compatible with the application of symmetries. In this, the concept of symmetric nets differs from the approach in Sec. 3.1.1, where, for a given high-level net, the set of symmetries has to be *calculated* by analyzing which permutations are compatible with the firing of transitions.

In symmetric nets, no analysis of the net's behavior is necessary, since the syntax of colors and arc expressions guarantees this compatibility.

Before proceeding with the definition of symmetric nets, we give some more details about basic color classes. A basic color class $\mathscr{C}_i$ may be *ordered*, i.e., equipped with a *successor function* suc that orbits through the class, satisfying $\forall c \in \mathscr{C}_i : \mathrm{suc}^{|\mathscr{C}_i|}(c) = c \wedge \forall 1 \leq k < |\mathscr{C}_i| : \mathrm{suc}^k(c) \neq c$, where suc^k describes the k-fold successive execution of suc. Additionally, each basic color class $\mathscr{C}_i$ is possibly partitioned into *static subclasses* $\mathscr{C}_{i,q}$, such that $\mathscr{C}_i = \bigsqcup_{q=1}^{n_i} \mathscr{C}_{i,q}$ for a natural number n_i. The partition of basic color classes into static subclasses serves the purpose of precisely identifying which colors from the type of a place behave symmetrically to each other.

The structured form of arc expressions and guards guarantees that elements within the same static subclass exhibit symmetric behavior. For a detailed explanation of these structured forms, including the formal definition, refer to part at the end of this section, that starts on p. 91.

Definition 3.36 (Symmetric net). A *symmetric net* $N = (\mathscr{C}, P, T, J, F, g, M_0)$ has the following components:

- $\mathscr{C}$ is the family of finite *basic color classes*, $\mathscr{C} = \{\mathscr{C}_1, \ldots, \mathscr{C}_n\}$, with $\mathscr{C}_i \cap \mathscr{C}_j = \emptyset$.

  We use the shorthand $I := \{1, \ldots, n\}$. Each $\mathscr{C}_i$ is possibly partitioned into *static subclasses*, i.e., there is a fixed $n_i > 0$ and fixed $\mathscr{C}_{i,1}, \ldots, \mathscr{C}_{i,n_i}$ s.t. $\mathscr{C}_i = \bigsqcup_{q=1}^{n_i} \mathscr{C}_{i,q}$.

  For some $\mathbf{u} \in \{0, \ldots, n\}$, the first $\mathbf{u}$ color classes $\mathscr{C}_1, \ldots, \mathscr{C}_{\mathbf{u}}$ are ***u**nordered*, the remaining classes $\mathscr{C}_{\mathbf{u}+1}, \ldots, \mathscr{C}_n$ are *ordered*, i.e., equipped with a successor function suc orbiting through the class as explained above.
- P T denote the disjunct sets of *places* and *transitions*.
- $J : (P \cup T) \times I \to \mathbb{N}$ is the *color domain function*. By J, every node r is equipped with a so-called *color domain*

  $$\mathscr{C}(r) := \mathscr{C}_1^{J(r,1)} \times \cdots \times \mathscr{C}_n^{J(r,n)}.[3]$$

  For every $r \in P \cup T$, the value $J(r, i)$ describes "how often $\mathscr{C}_i$ appears in r's color domain".

  In terms of the excursus above, the color domain of a place in a symmetric net would be its *type*. For a place p, a typical element in $\mathscr{C}(p)$ is $c = ((c_i^j)_{j=1}^{J(p,i)})_{i=1}^n$ with $c_i^j \in \mathscr{C}_i$ for every i, j. Analogously, for a transition t, a typical element in $\mathscr{C}(t)$ is $\sigma = ((\sigma_i^j)_{j=1}^{J(t,i)})_{i=1}^n$ with $\sigma_i^j \in \mathscr{C}_i$ for every i, j.
- F is the *flow function* that indicates which colors $c \in \mathscr{C}(p)$ are taken from resp. placed on p when firing t in a color $\sigma \in \mathscr{C}(t)$.

[3]In these Cartesian products, we set $\mathscr{C}_i^0 = \{\varepsilon\}$, s.t. we "ignore" basic color classes $\mathscr{C}_i$ with $J(r, i) = 0$ (since $A \times \{\varepsilon\} \cong A$).

For all $p \in P$ and $t \in T$, the functions

$$F(p,t), F(t,p) \in (\mathscr{C}(t) \to (\mathscr{C}(p) \to \mathbb{N}))$$

are *standard functions* (cp. p. 91ff).

- g equips every transition $t \in T$ with a *guard* $g(t) : \mathscr{C}(t) \to \{\mathit{true}, \mathit{false}\}$ that is a *standard predicate* (cp. below). It indicates whether t can fire in a color $\sigma \in \mathscr{C}(t)$. By default we will assume for every $t \in T$ that $\forall \sigma \in \mathscr{C}(t) : g(t)(\sigma) = \mathit{true}$.

  For a transition $t \in T$, the colors σ in $\mathscr{C}(t)$ for which $g(t)(\sigma) = \mathit{true}$ are (analogously to Sec. 2.3.1) called the *modes* of t, and are denoted by $\Sigma(t)$.

- M_0 is the *initial marking.*

  A marking is a function M that for every place $p \in P$ describes how often which color lies on p. For $\mathit{Col} = \bigcup_{p \in P} \mathscr{C}(p)$, a marking M is a multi-set over $P \times \mathit{Col}$ (as in Sec. 2.3.1), that additionally satisfies $(p,c) \in M \Rightarrow c \in \mathscr{C}(p)$. ◁

The semantics of a symmetric net N is defined analogously to Sec. 2.3.1, but the notions of preset and postset are adapted to the syntax of F in symmetric nets: by denoting, for a mode $\sigma \in \Sigma(t)$ of a transition t, the sets $\mathit{pre}(t,\sigma) := \{\!\!\{ (p,c) \mid c \in F(p,t)(\sigma) \}\!\!\}$ and $\mathit{post}(t,\sigma) := \{\!\!\{ (p,c) \mid c \in F(t,p)(\sigma) \}\!\!\}$, we have arrived at the syntax from Sec. 2.3.1: we have $M[t,\sigma\rangle$ iff $\mathit{pre}(t,\sigma) \le M$, and $M[t,\sigma\rangle M'$ iff $M' = (M - \mathit{pre}(t,\sigma)) + \mathit{post}(t,\sigma)$.

At this point, we do not provide any further details about the structured forms of arc expressions and guards, i.e., about the syntax of standard functions and standard predicates. A brief overview about the possibilities in this syntax is shifted to the end of this Section 3.3.1, starting from p. 91, followed by the somewhat cumbersome formal definitions of standard forms starting from p. 94. Instead, we show the running example of Signal Sending Satellites in the formalism of symmetric nets.

Example 3.37. The (underlying net of the) running example Signal Sending Satellites from Fig. 3.13 can be transformed into a symmetric net with the same basic structure, same place types, and equivalent arc labeling. The result of this transformation is shown in Fig. 3.14. Note that the caption reads "...as a symmetric Petri *game*". Symmetric Petri games (explained later in this section) are to symmetric nets what high-level Petri games are to high-level nets in the formalism of Sec. 2.3.1 and Sec. 2.3.2. For now, we ignore the different colors and borders of places.

We notice the following differences when comparing the high-level net with places types in Fig. 3.13 to the symmetric net in Fig. 3.14:

- The denotation of $\mathscr{C}_2$ as **Sats** is omitted.
- Next to every transition t, a (Cartesian product of) basic subclass(es) is written, denoting the color domain $\mathscr{C}(t)$.
- The arc expressions and guards are different. In particular, they do not contain the variables x and y.

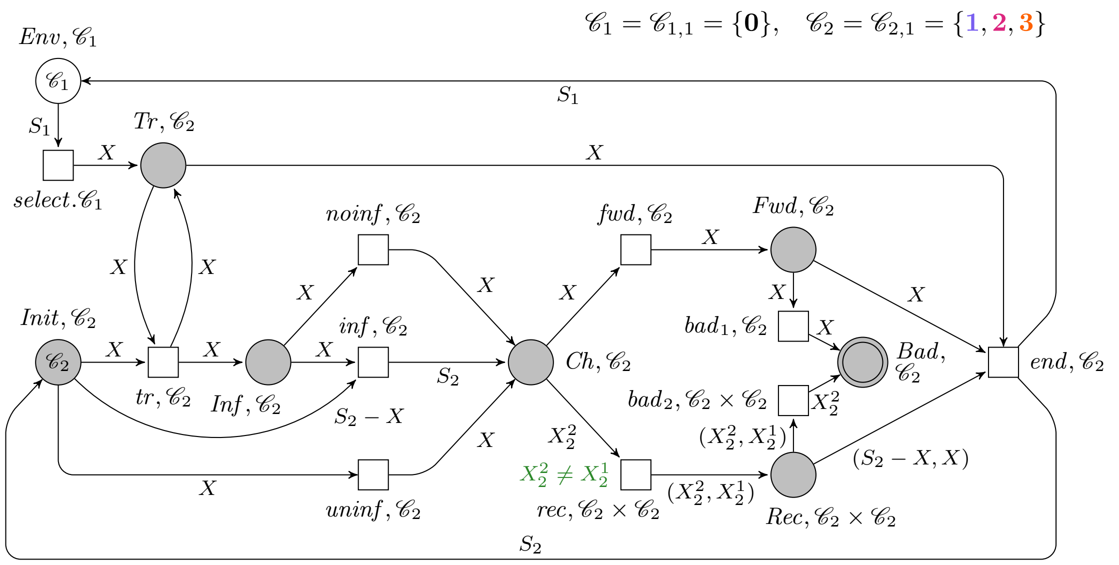


Figure 3.14: The running example Signal Sending Satellites as a symmetric Petri game.

We now go into detail about the second and third point, and in particular explain the *functions* $S_1, S_2, X, X_2^1, X_2^2$.

We begin with the transition that appears to have undergone the most complex changes, namely, transition *rec*. Next to *rec*, the color domain $\mathscr{C}(rec) = \mathscr{C}_2 \times \mathscr{C}_2$ is denoted. This means a color $\sigma \in \mathscr{C}(rec)$ is of the form $\sigma = (\sigma_2^1, \sigma_2^2)$, with $\sigma_2^1, \sigma_2^2 \in \mathscr{C}_2$. We see that in the guard of and the arc expressions around *rec*, the terms X_2^1 and X_2^1 appear, replacing the variables x and y. While this at first glance seams like just a renaming of variables (and in this case can be interpreted like this), the nature of these terms is a bit more complicated.

X_2^1 and X_2^2 are actually *functions*, and refer to σ_2^1 and σ_2^2, respectively. Consider the arc expression on the arc from *Ch* to *rec*, namely X_2^2. This formally is a function $X_2^2 : \mathscr{C}(rec) \to (\mathscr{C}(Ch) \to \mathbb{N})$. When *rec* fires in a mode $(\sigma_2^1, \sigma_2^2) \in \mathscr{C}(rec)$, then for all colors $c \in \mathscr{C}(Ch) = \mathscr{C}_2$, the value $X_2^2(\sigma)(c)$ is the number of copies of c that are taken from *Ch*. The function X_2^2 is given by

$$X_2^2(\sigma)(c) = \begin{cases} 1 & , \ c = \sigma_2^2 \\ 0 & , \ \text{else.} \end{cases}$$

Thus, when firing *rec* in a mode $\sigma = (\sigma_2^1, \sigma_2^2)$, the color σ_2^2 is taken from *Ch*.

Consider now the arc from *rec* to *Rec*, labeled by (X_2^2, X_2^1). Again, this expression is formally a function $(X_2^2, X_2^1) : \mathscr{C}(rec) \to (\mathscr{C}(Rec) \to \mathbb{N})$, describing that when *rec* is fired in mode $\sigma = (\sigma_2^1, \sigma_2^2)$, then for every $c = (c_2^1, c_2^2) \in \mathscr{C}(Rec) = \mathscr{C}_2 \times \mathscr{C}_2$, the number of copies of c that is placed on *Rec* is described by $(X_2^2, X_2^1)(\sigma)(c)$. In case of such a tuple, the value of $(X_2^2, X_2^1)(\sigma)(c)$ is given by the following product:

$$(X_2^2, X_2^1)(\sigma)(c_2^1, c_2^2) = X_2^2(\sigma)(c_2^1) \cdot X_2^2(\sigma)(c_2^1) = \begin{cases} 1 & , \ c_2^1 = \sigma_2^2 \wedge c_2^2 = \sigma_2^1 \\ 0 & , \ \text{else.} \end{cases}$$

This means, when firing *rec* in mode (σ_2^1, σ_2^2) then the color (σ_2^2, σ_2^1) is placed on *Rec*.

The guard of *rec* is a function $(X_2^2 \neq X_2^1) : \mathscr{C}(rec) \to \{true, false\}$ with $(X_2^2 \neq X_2^1)(\sigma) = (\sigma_2^2 \neq \sigma_2^1)$, since X_2^1 and X_2^2 refer to σ_2^1 and σ_2^2, respectively.

Next, consider transition *inf* with $\mathscr{C}(inf) = \mathscr{C}_2$. At the arc expressions around *inf*, we see two new terms: X and S_2. When we have only a single basic color class $\mathscr{C}(t) = \mathscr{C}_i$ as the color domain of a transition t (in this case $\mathscr{C}(inf) = \mathscr{C}_2$), then every mode σ of t is a basic color $\sigma = \sigma_i^1$ (in this case $\sigma = \sigma_2^1$). In such a case, X is an abbreviation for the function X_i^1. Thus, the X on the arc from *Inf* to *inf* abbreviates the function X_2^1, meaning that when *inf* fires in mode $\sigma = \sigma_2^1$ then σ_2^1 is taken from place *Inf*.

On the arc from *inf* to *Ch* we have a function $S_2 : \mathscr{C}(inf) \to (\mathscr{C}(Ch) \to \mathbb{N})$. Such a function is given by $\forall \sigma \in \mathscr{C}(inf) \forall c \in \mathscr{C}_2 : S_2(\sigma)(c) = 1$. This means when *inf* fires in mode $\sigma = \sigma_2^1$ then (independently of σ) the whole class $\mathscr{C}_2$ is placed on *Ch*.

Finally, on the arc from *Init* to *inf* there is a *sum* (or in this case, a difference) of the just described functions S_2 and X. Such a sum (or difference) of functions is defined as usual, in this case (for $\sigma = \sigma_2^1$ and $c = c_2^1 \in \mathscr{C}(Init) = \mathscr{C}_2$)

$$(S_2 - X)(\sigma_2^1)(c_2^1) = S_2(\sigma_2^1)(c_2^1) - X(\sigma_2^1)(c_2^1) = \begin{cases} 1 & , \ c_2^1 \neq \sigma_2^1 \\ 0 & , \ c_2^1 = \sigma_2^1. \end{cases}$$

All in all this means when *inf* fires, a satellite is taken from *Inf*, all *other* satellites are taken from *Init*, and *all* satellites are placed on *Ch*.

Having explained the details of the arc expressions around *rec* an *inf*, we can now understand all other arc expressions used in the net. We only consider one more arc expression in detail, namely the one on the arc from *Rec* to *end*, reading $(S_2 - X, X)$. Combining the concept of sums and tuples from above, we have for $\sigma = \sigma_2^1 \in \mathscr{C}(end)$ and $(c_2^1, c_2^2) \in \mathscr{C}(Rec) = \mathscr{C}_2 \times \mathscr{C}_2$ that

$$(S_2 - X, X)(\sigma_2^1)(c_2^1, c_2^2) = (S_2 - X)(\sigma_2^1)(c_2^1) \cdot X(\sigma_2^1)(c_2^2) = \begin{cases} 1 & , \ c_2^1 \neq \sigma_2^1 \wedge c_2^2 = \sigma_2^1 \\ 0 & , \ \text{else,} \end{cases}$$

meaning that *end* can only fire if all other satellites are in Receiving mode, expecting a message from the one in Forwarding mode.

Being used to the denotation of arc expressions, the graphical representation is nearly as easily readable as the one in Fig. 3.13 that is not written in the symmetric net formalism. This makes writing a high-level net as a symmetric net is only slightly more cumbersome than allowing arbitrary arc expressions. $\lhd$

When modeling this example as a symmetric Petri game, we make two observations. Firstly, an alternative approach would involve defining a single basic color class $\mathscr{C}_1 =$

$\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$ with two static subclasses $\mathscr{C}_{1,1} = \{\mathbf{0}\}$ and $\mathscr{C}_{1,2} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$. In this scenario, every place would be of type $\mathscr{C}_1$ or $\mathscr{C}_1 \times \mathscr{C}_1$. Consequently, we would need to add the predicate "$X_1^1 \in \mathscr{C}_{1,2}$" to the guard of *rec* to ensure that no satellite goes into the Receiving mode while waiting for a message from the base station. considering that all places can hold either only ever (tuples of) satellites or only ever the color $\mathbf{0}$, we opted to allocate these respective sets to individual base classes $\mathscr{C}_1$ and $\mathscr{C}_2$.

Secondly, note that the concept of basic color classes and the possibility of having tuples as colors is not a novel idea within this thesis. In Sec.2.3.2, we provided a list of readability-improving "tricks" when illustrating high-level Petri games. One of these tricks involved labeling arcs with tuples of variables, implicitly extending the color set *Col* to also contain tuples of colors. This was applied in Fig. 3.2, where we denoted the color set of Signal Sending Satellites as $Col = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$ and allowed tuples of these colors to be placed on or taken from the place *Rec*. This corresponds directly to having, in the case of the symmetric Petri game in Fig. 3.14, the type $\mathscr{C}(Rec) = \mathscr{C}_2 \times \mathscr{C}_2$.

The following proposition shows that this formalism is not restricting the expressive power of high-level nets with place types and arbitrary arc expressions.

Proposition 3.38 ([CDFH91b], Proposition 2.1). *Any high-level Petri net can be transformed into a symmetric net with the same basic structure, same place types, and equivalent arc labeling.*

The proof of this result can be accomplished through a straightforward encoding by introducing basic color classes with a singleton static subclass for every color present in the given high-level net. However, as we will observe later, this approach yields a symmetric net in which the only symmetry that can be read from the specification is the identity. Nonetheless, it is often feasible to construct a non-trivial symmetric net that preserves most of the symmetries present in the original net. We will revisit this concept in Example 3.40.

As we have seen for high-level Petri nets, a symmetric net N can be *expanded* into a P/T Petri net $\mathrm{Exp}(N) := (\mathsf{P}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0)$ with places $\mathsf{P} := \{p.c \mid p \in P, c \in \mathscr{C}(p)\}$, transitions $\mathsf{T} := \{t.\sigma \mid t \in T, \sigma \in \Sigma(t)\}$, the flow F defined by $\forall p.c \in \mathsf{P}\ \forall t.\sigma \in \mathsf{T}$: $\mathsf{F}(p.c, t.\sigma) := F(p,t)(\sigma)(c) \wedge \mathsf{F}(t.\sigma, p.c) := F(t,p)(\sigma)(c)$, and initial marking M_0 defined by $\forall p.c \in \mathsf{P} : \mathsf{M}_0(p.c) := M_0(p)(c)$. As before, the two nets then have the same semantics: the number of tokens on a place $p.c$ in a marking in $\mathrm{Exp}(N)$ indicates the number of colors c on place p in the corresponding marking in N. Firing a transition $t.\sigma$ in $\mathrm{Exp}(N)$ corresponds to firing transition t in mode σ in N. When translating a high-level Petri net into a symmetric net, this expansion definition coincides with the one for high-level Petri nets from Sec. 2.3.1.

Symmetries in Symmetric Nets and Symmetric Petri Games

We now come to one of the main advantages of having a structured form of guards and arc expressions: in Sec. 3.1.1, we gave a definition of admissible symmetries involving the semantics of the net. This means, to get the set $S(N)$ for a given high-level net N, an analysis of the guards and flow in N is necessary (as we did in Example 3.3). While it is more cumbersome to define a high-level net in the formalism of symmetric nets, one payoff is that no analysis is needed to get the net's symmetries: the structured form of guards and arc expressions *ensures* that all elements in the same basic subclass are treated equally. We therefore know that all symmetries that only permute elements *inside* their respective basic subclass are compatible with the semantics of the net, and are therefore admissible symmetries.

Definition 3.39 (Symmetries of symmetric net, [CDFH91a]). The *symmetries* S_N in a symmetric net N are all tuples $s = (s_1, \dots, s_n)$ such that each s_i is a permutation on $\mathscr{C}_i$ satisfying $\forall q = 1, \dots, n_i : s_i(\mathscr{C}_{i,q}) = \mathscr{C}_{i,q}$, i.e., it respects the partition into static subclasses. In case of an ordered basic class $\mathscr{C}_i$ with $i > \mathbf{u}$ we additionally require s_i to be a rotation w.r.t. the successor function. $\lhd$

Without loss of generality, we assume that for ordered basic color classes, i.e., $\mathscr{C}_i$ with $i > \mathbf{u}$, each static subclass contains is a set of subsequent elements, formally, $\forall 1 \leq q \leq n_i \, \exists c \in \mathscr{C}_{i,q} : \mathscr{C}_{i,q} = \{\mathrm{suc}^k(c) \mid 0 \leq k < |\mathscr{C}_{i,q}|\}$. Under this assumption, if $\mathscr{C}_i$ is ordered and divided into two or more static subclasses then the only possibility for s_i is the identity $id_{\mathscr{C}_i}$.[4] A symmetry s can be applied to an element $c \in \mathscr{C}_i$ of a basic color class by $s(c) := s_i(c)$. The application to tuples, e.g., colors on places or transition modes, is defined by the application in each entry.

In a symmetric net, we can without loss of generality assume that the initial marking M_0 is *symmetric*, meaning that $\forall s \in S_N : s(M_0) = M_0$. Achieving this can be done by either further partitioning the basic color classes with respect to the colors present in the initial marking or by introducing an additional place and transition, initializing the net with its initial marking or any symmetric marking. Note that the second procedure introduces additional (albeit symmetric to the original) behavior in the net.

It is shown in [CDFH91a] that for the symmetries defined above, the structured form of arc expressions and guards in symmetric nets ensures $\forall s \in S_N \, \forall t \in T \, \forall p \in P \, \forall \sigma \in \mathscr{C}(t) \, \forall c \in \mathscr{C}(p)$:

- $g(t)(\sigma) = \mathit{true} \quad \Leftrightarrow \quad g(t)(s(\sigma)) = \mathit{true}$,
- $F(p,t)(\sigma)(c) = F(p,t)(s(\sigma))(s(c)) \quad \wedge \quad F(t,p)(\sigma)(c) = F(t,p)(s(\sigma))(s(c))$.

These equations imply that symmetries are compatible with the firing relation, i.e.,

$$\forall s \in S_N : M[t,\sigma\rangle M' \Leftrightarrow s(M)[t,s(\sigma)\rangle s(M'),$$

where, as before, the marking $s(M)$ is given by $s(M)(p, s(c)) = M(p,c)$. The set S_N, together with the function composition $\circ$, forms a group with identity $(id_{\mathscr{C}_i})_{i=1}^n$.

This means that, when we transform a high-level net N into an equivalent symmetric net N' in the sense of Proposition 3.38, then the $(S_{N'}, \circ)$ is (isomorphic to) a subgroup of the admissible symmetries $(S(N), \circ)$. As already mentioned after Proposition 3.38, we can often give a transformation that preserves these symmetries, i.e., $S_{N'} \equiv S(N)$.

[4]In [CDFH91a], this result is stated without the assumption explicitly being made. My guess is that this assumption is made there implicitly, since for ordered classes that violate the assumption there can be more than one possible symmetry.

Example 3.40 (Symmetries in Signal Sending Satellites). We inspect the symmetries S_N in the symmetric Petri net N, Signal Sending Satellites, from Fig. 3.14. We have the two unordered basic color classes $\mathscr{C}_1 = \{\mathbf{0}\}$ and $\mathscr{C}_2 = \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$. Both basic color classes are not partitioned into subclasses, i.e., $\mathscr{C}_1 = \mathscr{C}_{1,1}$ and $\mathscr{C}_2 = \mathscr{C}_{2,1}$. Thus, by Def. 3.39 the set S_N contains all tuples $s = (s_1, s_2)$ where s_1 is a permutation on $\{\mathbf{0}\}$ (i.e., $s_1 = id_{\{\mathbf{0}\}}$), and s_2 is a permutation on $\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$. This means that S_N is isomorphic to $Sym(\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}) = S_3$. This is exactly the set $S(G)$ that we deduced in Example 3.3 by analyzing the (equivalent high-level) net's behavior. ◁

Symmetric Petri Games

The definition of symmetric Petri games proceeds analogously to Sec. 2.3.2: in a *symmetric Petri game* $G = (\mathscr{C}, P_S, P_E, T, J, F, g, M_0, \mathrm{Obj}, P_{\circledast})$ the places P of an *underlying symmetric net* $N(G) = (\mathscr{C}, P, T, J, F, g, M_0)$, are divided into *system places* P_S and *environment places* P_E. The set $P_{\circledast} \subseteq P_S$ indicates the special places used to define the objective for the system players together with $\mathrm{Obj} \in \{\mathrm{Safety}, \mathrm{Reach}\}$. All achieved concepts for high-level Petri games from Sec. 2.3.2 can be directly adopted in the case of symmetric Petri games. This is captured in the following corollary, which is a direct consequence of Proposition 3.38.

Corollary 3.41. *Any high-level Petri game can be transformed into a symmetric Petri game with the same basic structure, and equivalent arc labeling and behavior.*

Taking the different colors and borders into consideration, Fig. 3.14 shows a symmetric Petri game whose underlying net we discussed in detail above. It is the result of transforming the high-level Petri game from Fig. 3.13 into a symmetric Petri game.

The definitions of mixed communication and recurrently interfering environment from Sec. 3.1.2 can be directly adopted for symmetric Petri games. Thus, we define the class of *proper* symmetric Petri games exactly as in Def. 3.6. By this, whenever a proper high-level Petri game is transformed into a symmetric Petri game in the sense of Cor. 3.41, the result is a proper symmetric Petri game.

As before, we say the symmetries S_G in a symmetric Petri game G are the symmetries $S_{N(G)}$ in the underlying symmetric net. We have discussed above that it is often possible to translate a high-level Petri net N into a symmetric Petri net N' such that $S(N) \equiv S_{N'}$. Consequently, this is also the case for high-level Petri games. In particular, for the benchmark tests from [GOW20] presented in Sec. 3.3.5, we transformed high-level Petri games introduced in [GO21] into symmetric Petri games. In this process, no symmetries from the original nets were lost. From now on, we assume that the symbolic two-player game was built w.r.t. the subgroup S_G of all admissible symmetries $S(G)$. As all the constructions solely rely on the fact that $S(N)$ is a group, this substitution has no impact on the theoretical results of of Sec. 3.2.

Summary: Symmetric Nets

We summarize the content of this section. Symmetric nets are a formalism for describing high-level Petri nets with place types and arc expressions. The main concepts are

– giving types of places as a Cartesian product of fixed *basic color classes*, and
– restricting the syntax of arc expressions and guards to certain *standard forms.*

The basic color classes are partitioned into *static subclasses*, and the standard forms ensure that all objects in the same static subclass are handled equally. We have the following results for symmetric nets:

- Symmetric nets have the same expressive power as the high-level formalism from Sec. 2.3.1.
- The admissible symmetries of a symmetric net can be directly read from its specification, without further analysis of the net's behavior.
- The definition of (proper) symmetric Petri games with an underlying symmetric net proceeds exactly as in Sec. 2.3.2.
- The results of Sec. 3.2 still hold when considering high-level Petri games given in the symmetric net formalism.

For the rest of Section 3.3 we assume high-level Petri games to be given in the symmetric net formalism.

The content and discourse in this section have established a reasonable understanding of the concept of symmetric nets. The remainder of Section 3.3.1 is dedicated the details of their restricted syntax of arc expressions and transition guards that we already informally dealt with when transforming Signal Sending Satellites into a symmetric net. We first give an overview of the possibilities allowed by the so-called Standard Functions and Standard Predicates, and then, for the sake of completeness, formally define the syntax and semantic.

Standard Functions and Standard Predicates: Overview

We will now provide a brief overview of the possibilities for describing guards and arc expressions as standard predicates and standard functions, with a more elucidating example presented in Example 3.42. For the sake of completeness, we provide the formal and somewhat intricate definitions of these structured forms at the end of this section, beginning on p. 94. In essence, the permissible guards and expressions ensure equal (resp. *symmetric*) behavior of colors within the same basic color class.

A typical mode $\sigma \in \Sigma(t)$ of a transition t with $\Sigma(t) \subseteq \mathscr{C}(t) = \mathscr{C}_1^{J(t,1)} \times \cdots \times \mathscr{C}_n^{J(t,n)}$ is a tuple

$$\begin{aligned}\sigma &= \big((\sigma_i^j)_{j=1}^{J(t,i)}\big)_{i=1}^{n}\\ &= (\sigma_1^1, \dots, \sigma_1^{J(t,1)}, \sigma_2^1, \dots, \sigma_2^{J(t,2)}, \dots\dots, \sigma_n^1, \dots, \sigma_n^{J(t,n)}), \quad \text{where } \sigma_i^j \in \mathscr{C}_i.\end{aligned}$$

In the arc expressions around and the guard of t, the term X_i^j refers to the j-th instantiation of $\mathscr{C}_i$ in such a mode, i.e., to σ_i^j.

For guards, we have the three forms of basic predicates $X_i^j = X_i^k$, $X_i^j = \text{suc}(X_i^k)$ (for $i > \mathbf{u}$, i.e., if $\mathscr{C}_i$ is ordered), and $X_i^j \in \mathscr{C}_{i,q}$. All Boolean combinations with the operators $\vee, \wedge, \neg$ yield the so-called *standard predicates* on $\mathscr{C}(t)$ allowed as guards. They evaluate to true under a color σ if the respective entries σ_i^j, σ_i^k satisfy the predicate. In this case, σ is called a *mode* of t.

The arc expressions allowed on an arc connecting t to a place p, called *standard functions*, are finite sums of *guarded functions* $[g_{\mathscr{C}(t)}]f$, with $g_{\mathscr{C}(t)}$ being a standard predicate over $\mathscr{C}(t)$ and f being a *basic function* $f : \mathscr{C}(t) \to (\mathscr{C}(p) \to \mathbb{N})$. The guarded function assigns evaluates to f if $g_{\mathscr{C}(t)}(\sigma) = \mathit{true}$, and to the zero function, otherwise. Basic functions in their turn are given by tuples of functions

$$\begin{aligned} f &= \big((f_i^j)_{j=1}^{J(p,i)}\big)_{i=1}^n \\ &= (f_1^1, \dots, f_1^{J(p,1)}, f_2^1, \dots, f_2^{J(p,2)}, \dots\dots, f_n^1, \dots, f_n^{J(p,n)}), \end{aligned}$$

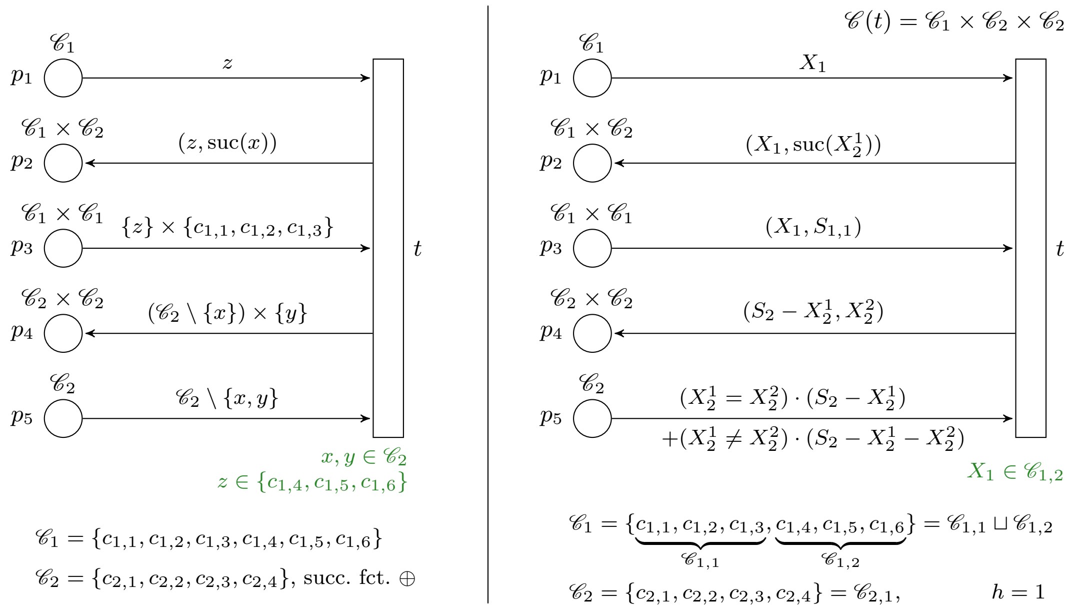
where finally every $f_i^j : \mathscr{C}(t) \to [\mathscr{C}_i \to \mathbb{N}]$ is a sum of functions of the following type:

- a synchronization resp. diffusion of all objects in a static subclass (independently of σ), i.e., a function $\alpha.S_{i,q}$ s.t. for $c_i \in \mathscr{C}_i$ it evaluates to $\alpha.S_{i,q}(\sigma)(c_i) = \alpha$ if $c_i \in \mathscr{C}_{i,q}$, and $\alpha.S_{i,q}(\sigma)(c_i) = 0$, else,
- selecting of one object that behaves independent of the other objects of the class, i.e., a function $\beta.X_i^k$ s.t. for $c_i \in \mathscr{C}_i$ it evaluates to $\beta.X_i^k(\sigma)(c_i) = \beta$ if $c_i = \sigma_i^k$, and $\beta.X_i^k(\sigma)(c_i) = 0$, else,
- the successor function, only applicable to ordered classes with $i > \mathbf{u}$, i.e., a function $\gamma.\,\text{suc}(X_i^k)$ s.t. for $c_i \in \mathscr{C}_i$ it evaluates to $\gamma.\,\text{suc}(X_i^k)(\sigma)(c) = \gamma$ if $c_i = \text{suc}(\sigma_i^k)$, and $\gamma.\,\text{suc}(X_i^k)(\sigma)(c) = 0$, else.

The application of $f(\sigma)$ to a color $c = \big((c_i^j)_{j=1}^{J(p,i)}\big)_{i=1}^n \in \mathscr{C}(p)$ is then given by $f(\sigma)(c) = \prod_{i=1}^n \prod_{j=1}^{J(p,i)} f_i^j(\sigma)(c_i^j)$ An illustration of these concepts will be presented shortly in the following example. There, we show the translation of certain common arc expressions into the syntax of symmetric nets.

Example 3.42 (Transforming high-level nets into symmetric nets). In Fig. 3.15 we see on the left a high-level Petri net with type places (written above the respective place) and arcs labeled with expressions, as described in the excursus above. On the right we see a symmetric Petri net equivalent behavior.

We have one transition t and five connected places $p_1, \dots, p_5$. Whether a place is in the preset or postset of t does not matter for transforming the high-level net into a symmetric net. In the high-level net, we see that the types of places are build from $\mathscr{C}_1 = \{c_{1,1}, \dots, c_{1,6}\}$ and $\mathscr{C}_2 = \{c_{2,1}, \dots, c_{2,4}\}$. On $\mathscr{C}_2$ we additionally have a successor function suc cycling through the set, i.e., $\text{suc}(c_{2,j}) = c_{2,j'}$ where $j' = (j \mod 4) + 1$. We see that not all elements in $\mathscr{C}_1$ behave the same: the guard of t allows z only to be in $\{c_{1,4}, c_{1,5}, c_{1,6}\}$, while the expression on the edge from p_3 says that, when t fires



$\mathscr{C}_1$
p_1
z
$\mathscr{C}_1 \times \mathscr{C}_2$
p_2
$(z, \mathrm{suc}(x))$
$\mathscr{C}_1 \times \mathscr{C}_1$
p_3
$\{z\} \times \{c_{1,1}, c_{1,2}, c_{1,3}\}$
t
$\mathscr{C}_2 \times \mathscr{C}_2$
p_4
$(\mathscr{C}_2 \setminus \{x\}) \times \{y\}$
$\mathscr{C}_2$
p_5
$\mathscr{C}_2 \setminus \{x, y\}$
$x, y \in \mathscr{C}_2$
$z \in \{c_{1,4}, c_{1,5}, c_{1,6}\}$
$\mathscr{C}_1 = \{c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4}, c_{1,5}, c_{1,6}\}$
$\mathscr{C}_2 = \{c_{2,1}, c_{2,2}, c_{2,3}, c_{2,4}\}$, succ. fct. $\oplus$
$\mathscr{C}(t) = \mathscr{C}_1 \times \mathscr{C}_2 \times \mathscr{C}_2$
$\mathscr{C}_1$
p_1
X_1
$\mathscr{C}_1 \times \mathscr{C}_2$
p_2
$(X_1, \mathrm{suc}(X_2^1))$
$\mathscr{C}_1 \times \mathscr{C}_1$
p_3
$(X_1, S_{1,1})$
t
$\mathscr{C}_2 \times \mathscr{C}_2$
p_4
$(S_2 - X_2^1, X_2^2)$
$\mathscr{C}_2$
p_5
$(X_2^1 = X_2^2) \cdot (S_2 - X_2^1)$
$+(X_2^1 \neq X_2^2) \cdot (S_2 - X_2^1 - X_2^2)$
$X_1 \in \mathscr{C}_{1,2}$
$\mathscr{C}_1 = \{\underbrace{c_{1,1}, c_{1,2}, c_{1,3}}_{\mathscr{C}_{1,1}}, \underbrace{c_{1,4}, c_{1,5}, c_{1,6}}_{\mathscr{C}_{1,2}}\} = \mathscr{C}_{1,1} \sqcup \mathscr{C}_{1,2}$
$\mathscr{C}_2 = \{c_{2,1}, c_{2,2}, c_{2,3}, c_{2,4}\} = \mathscr{C}_{2,1}$,
$h = 1$

Figure 3.15: A high-level Petri net and a symmetric net with the same behavior.

in mode σ, the three tuples $(\sigma(z), c_{1,1}), (\sigma(z), c_{1,2}), (\sigma(z), c_{1,3})$ are taken from p_3. By partitioning $\mathscr{C}_1$ into the respective *static subclasses* $\mathscr{C}_{1,1}$ and $\mathscr{C}_{1,2}$, we ensure that every color in the same static subclass behaves equally (or *symmetrically*). Since $\mathscr{C}_1$ is not ordered but $\mathscr{C}_2$ is, we have $\mathbf{u} = 1$.

Three variables, x, y, z, occur in the arc expressions around t. From the types of places, we see that $x, y \in \mathscr{C}_2$ and $z \in \mathscr{C}_1$ holds for every mode of t. Correspondingly, in the symmetric net, the color domain of t is given by $\mathscr{C}(t) = \mathscr{C}_1 \times \mathscr{C}_2 \times \mathscr{C}_2$. A color $\sigma = (\sigma_1^1, \sigma_2^1, \sigma_2^2) \in \mathscr{C}_1 \times \mathscr{C}_2 \times \mathscr{C}_2$ represents the variable assignment $\{z \leftarrow \sigma_1^1, x \leftarrow \sigma_2^1, y \leftarrow \sigma_2^2\}$ in the high-level net on the left. Thus, in the firing of t, the class $\mathscr{C}_1$ is instantiated once, and $\mathscr{C}_2$ is instantiated twice.

In the arc expressions and guard, the term X_i^j refers to the j-th instantiation of a variable from $\mathscr{C}_i$ in a mode of t, i.e., to σ_i^j. For color classes $\mathscr{C}_i$ that are only instantiated once (in this case $\mathscr{C}_1$) X_i^1 is short for X_i (in this case we abbreviate X_1^1 by X_1). We now see how we transformed the guard in the high-level net to the guard in the symmetric net. The term X_1 refers to σ_1^1 in a color $\sigma = (\sigma_1^1, \sigma_2^1, \sigma_2^2) \in \mathscr{C}(t)$ representing the assignment $\{z \leftarrow \sigma_1^1, x \leftarrow \sigma_2^1, y \leftarrow \sigma_2^2\}$ in the high-level net, and we have the static subclass $\mathscr{C}_{1,2} = \{c_{1,4}, c_{1,5}, c_{1,6}\}$. We can say that in this transformation into a symmetric net, X_1 takes the role of z, and X_2^1, X_2^2 take the role of x, y, respectively. The predicate "$z \in \{c_{1,4}, c_{1,5}, c_{1,6}\}$" is therefore replaced by "$X_1 \in \mathscr{C}_{1,2}$". The predicate "$x, y \in \mathscr{C}_2$" is ensured since X_2^1 and X_2^2 always refer to instantiations of $\mathscr{C}_2$.

Next, we will sequentially elaborate on the transformation of labels on the edges. Analogously to the guard of t, in the edge from p_1 we have replaced the expression z by X_1.

On the left, indicated by the label on the respective edge, a tuple containing the value assigned to z and the successor of the value assigned to x is placed on p_2. We see that the techniques from before also work in tuples, and we get an edge labeled by $(X_1, \mathrm{suc}(X_2^1))$. The successor function suc in the second coordinate refers to the respective successor function in $\mathscr{C}_2$.

The edge from p_3 is the first to contain a *set* of three tuples. In the symmetric net, however, we only have *one* tuple. In the first coordinate, this tuple contains X_1 for which we have replaced z now several times. In the second coordinate, we have the term $S_{1,1}$. Such a term $S_{i,q}$ means that for every element of the respective static subclass $\mathscr{C}_{i,q}$ one respective tuple is moved, in this case $\mathscr{C}_{1,1}$, which is exactly what happens in the high-level net on the left.

The edge from t to p_4 is labeled by $(\mathscr{C}_2 \setminus \{x\}) \times \{y\}$. This means that, when firing t in mode σ, all tuples $(c_{2,i}, \sigma(y))$ except $(\sigma(x), \sigma(y))$ are placed on p_4. In the symmetric net, where X_2^1 represents $\sigma(x)$ and X_2^2 represents $\sigma(y)$, this is realized by a tuple with a difference of functions in the first coordinate. We abbreviate $\sum_{q=1}^{n_i} S_{i_q}$ by S_i, and since $\mathscr{C}_2$ is not further partitioned, i.e., $\mathscr{C}_2 = \mathscr{C}_{2,1}$ we have $S_{2,1} = S_2$. Such a difference, however, must never indicate that a negative number of colors is placed on or taken from an edge. This is ensured here, since the value represented by X_2^1 is always in $\mathscr{C}_2$.

Finally, the expression "$\mathscr{C}_2 \setminus \{x, y\}$" is replaced by a much more complex expression on the right. If we would naively put $S_2 - X_2^1 - X_2^1$ in the symmetric net, then this would in the case of a mode $\sigma = (\sigma_1^1, \sigma_2^1, \sigma_2^2) \in \Sigma(t)$ with $\sigma_2^1 = \sigma_2^2$ take "-1 instance" of the respective color from p_5. This is prohibited by the syntax of standard functions (cp. below for the formal definition). Thus, we have to make the corresponding distinction of cases by using a sum of guarded functions. For every mode σ, exactly one of the two guards $(X_2^1 = X_2^2)$ and $(X_2^1 \neq X_2^2)$ is true, and the corresponding function after the guard takes the correct colors from p_5. $\lhd$

We close this section by providing the interested reader with the formal definition of the standard forms for guards and arc expressions used in symmetric nets.

Standard Functions and Standard Predicates: Formal Definition

In the definition of symmetric nets, the terms "standard functions" and "standard predicates" are used. The following definitions are taken directly from [CDFH91a].

Predicates. The guard $g(t) : \mathscr{C}(t) \to \{\mathit{true}, \mathit{false}\}$ of a transition t indicates in which modes

$$\sigma = ((\sigma_i^j)_{j=1}^{J(t,i)})_{i=1}^n \in \mathscr{C}(t) = \mathscr{C}_1^{J(t,1)} \times \cdots \times \mathscr{C}_n^{J(t,n)}$$

the transition t can fire. We only allow guards of a structured form, called *standard predicates.* The allowed predicates are

- "the j-th instantiation of $\mathscr{C}_i$ in σ is *equal* to the k-th instantiation of $\mathscr{C}_i$", denoted by $X_i^j = X_i^k$,
  $(X_i^j = X_i^k)(\sigma) = (\sigma_{i,j} = \sigma_{i,k})$

- "the j-th instantiation of $\mathscr{C}_i$ in σ is the *successor* of the k-th instantiation of $\mathscr{C}_i$" (for $i > \mathbf{u}$), denoted by $X_i^j = \mathrm{suc}(X_i^k)$,
  $(X_i^j = \mathrm{suc}(X_i^k))(\sigma) = (\sigma_{i,j} = \mathrm{suc}(\sigma_{i,k}))$
- 'the j-th instantiation of $\mathscr{C}_i$ in σ is in the *static subclass* $\mathscr{C}_{i,q}$", denoted by $X_i^j \in \mathscr{C}_{i,q}$,
  $(X_i^j \in \mathscr{C}_{i,q})(\sigma) = (\sigma_{i,j} \in \mathscr{C}_{i,q})$

and all the Boolean combinations with the operators $\wedge$, $\vee$, and $\neg$.

Color functions. The definition of color functions consists of three successive steps. First, *basic functions* are introduced. Equipping basic functions with a guard yields *guarded functions*. A sum of guarded functions is then called a *standard function*.

For $p \in P$ and $t \in T$, the flow function is given by $F(p,t), F(t,p) : \mathscr{C}(t) \to [\mathscr{C}(p) \to \mathbb{N}]$. This demonstrates that for every mode of the transition t, a multi-set of colors is taken from, resp. placed on, the place p. We now interpret these functions as $F(p,t), F(t,p) : \mathscr{C}(p) \times \mathscr{C}(t) \to \mathbb{N}$.

Basic functions. The *basic functions* are a subset of these functions, denoted by $F_{\mathscr{C}(p),\mathscr{C}(t)} \subset (\mathscr{C}(p) \times \mathscr{C}(t) \to \mathbb{N})$. The basic functions $F_{\mathscr{C}(p),\mathscr{C}(t)}$ are defined inductively over the size of $\mathscr{C}(p)$.

Case $\mathscr{C}(p = \{\varepsilon\})$. This means $\forall i : J(p,i) = 0$. The interpretation is that the color domain of p is $\{\bullet\}$. These functions are then equivalent to valuated arcs in an ordinary Petri net.

$$F_{\{\varepsilon\},\mathscr{C}(t)} = \{\overline{\delta} : \{\varepsilon\} \times \mathscr{C}(t) \to \mathbb{N} \mid \exists \delta \in \mathbb{N} \forall \sigma \in \mathscr{C}(t) : \overline{\delta}(\varepsilon, \sigma) = \delta\}$$

Case $\mathscr{C}(p = \mathscr{C}_i)$. This means there is a i such that $J(p,i) = 1$ and for all $i' \neq i$, $J(p,i') = 0$. We define three basic color functions, where the third one only applies to ordered color classes $\mathscr{C}_i$ with $i > \mathbf{u}$.

- The synchronization resp. diffusion of all objects in a static subclass.

$$\begin{aligned} \forall \sigma \in \mathscr{C}(t)\, \forall q \leq m_i\, \forall \alpha \geq 0 : \quad & \alpha.S_{i,q} : \mathscr{C}_i \times \mathscr{C}(t) \to \mathbb{N}, \\ & \alpha.S_{i,q}(c_i, \sigma) = \begin{cases} \alpha & \text{, if } c_i \in \mathscr{C}_{i,q} \\ 0 & \text{, else.} \end{cases} \end{aligned}$$

- The selection of one object that behaves independent of the other objects of the class.

$$\begin{aligned} \forall \sigma \in \mathscr{C}(t)\, \forall k \leq t_i\, \forall \beta \geq 0 : \quad & \beta.X_i^k : \mathscr{C}_i \times \mathscr{C}(t) \to \mathbb{N}, \\ & \beta.X_i^k(c_i, \sigma) = \begin{cases} \beta & \text{, if } v_{i,k} = c_i \\ 0 & \text{, else.} \end{cases} \end{aligned}$$

- The successor function, only applicable to ordered classes. Let $i > \mathbf{u}$.

$$\begin{aligned}\forall \sigma \in \mathscr{C}(t)\, \forall k \leq t_i\, \forall \gamma \geq 0 : \quad & \gamma.\operatorname{suc}(X_i^k) : \mathscr{C}_i \times \mathscr{C}(t) \to \mathbb{N},\\ & \gamma.\operatorname{suc}(X_i^k)(c_i, \sigma) = \begin{cases}\gamma & \text{, if } \operatorname{suc}(\sigma_{i,k}) = c_i\\ 0 & \text{, else.}\end{cases}\end{aligned}$$

We can now define the basic functions $F_{\mathscr{C}_i,\mathscr{C}(t)}$. In this set, the functions above are combined. The individual coefficients can be negative, provided that no negative number of objects is selected in each static subclass. The conditions are not necessary but sufficient, and can easily be checked. If $i \leq \mathbf{u}$ then

$$F_{\mathscr{C}_i,\mathscr{C}(t)} = \Big\{ \sum_{q=1}^{n_i} \alpha_{i,q}.S_{i,q} + \sum_{k=1}^{t_i} \beta_{i,k}.X_i^k \;\Big|\; \forall q \leq n_i\, \forall K \subset \{1, \dots, t_i\} : \alpha_{i,q} + \sum_{k \in K} \beta_{i,k} \geq 0 \Big\}.$$

If $i > \mathbf{u}$ then

$$F_{\mathscr{C}_i,\mathscr{C}(t)} = \Big\{ \sum_{q=1}^{n_i} \alpha_{i,q}.S_{i,q} + \sum_{k=1}^{t_i} (\beta_{i,k}.X_i^k + \gamma_{i,k}.\operatorname{suc}(X_i^k)) \;\Big|\; \forall q \leq n_i\, \forall K \subset \{1, \dots, t_i\} : \alpha_{i,q} + \sum_{k \in K} \min(\beta_{i,k}, \gamma_{i,k}) \geq 0 \Big\}.$$

Case $\mathscr{C}(p \neq \{\varepsilon\})$. This case involves and generalizes the case $\mathscr{C}(p) = \mathscr{C}_i$. In the generalized case, the functions $f \in F_{\mathscr{C}(p),\mathscr{C}(t)}$ are tuples of functions in $F_{\mathscr{C}_i,\mathscr{C}(t)}$, i.e., $F_{\mathscr{C}(p),\mathscr{C}(t)} = \big\{((f_i^j)_{j=1}^{J(p,i)})_{i=1}^{n}\big\}$ and for $f = ((f_i^j)_{j=1}^{J(p,i)})_{i=1}^{n}$, the application to (c, σ) is $f(c, \sigma) = \prod_{i=1}^{n} \prod_{j=1}^{J(p,i)} f_i^j(c_i^j, \sigma)$.

Guarded functions are basic functions equipped with guards, i.e., they are of the form
$F : \mathscr{C}(p) \times \mathscr{C}(t) \to \mathbb{N}$, $F = [g_{\mathscr{C}(t)}]f$, where $g_{\mathscr{C}(t)}$ is a standard predicate on $\mathscr{C}(t)$, and $f \in F_{\mathscr{C}(p),\mathscr{C}(t)}$ is a basic function. It is $F(c, \sigma) = f(c, \sigma)$ if $g_{\mathscr{C}(t)}(d) = \mathit{true}$, and $F(c, \sigma) = 0$, otherwise.

Standard functions are finite sums of guarded functions. They constitute, for a place p and transition t, the functions $F(p,t), F(t,p)$, i.e., $F(p,t), F(t,p) = \sum_\ell F_\ell$, where $F_\ell : \mathscr{C}(p) \times \mathscr{C}(t) \to \mathbb{N}$ are guarded functions.

3.3.2 Constructing Canonical Representations

In this section we introduce dynamic representations of symbolic decision sets. We again want to draw the reader's attention to the subtle but crucial name difference between (arbitrary) representatives vs. (dynamic) representations that we discussed in the remark on p. 79. *Canonical* dynamic representations will replace the arbitrary representatives in the symbolic two-player game. Later, we transfer relations between and properties of (symbolic) decision sets to the established canonical representations. Dynamic representations can be smaller than the representatives, in the sense that, in addition to taking into account the symmetry *between* different states in the Petri game, they also exploit symmetry *inside* each state.

Consulting our running example Signal Sending Satellites again, we now give the idea of dynamic representations vs arbitrary representatives. The idea of the representatives from the Sec. 3.2 is to say

> "Only consider the case where the environment chooses Satellite 1, which goes into Forwarding Mode. In this case, both Satellite 2 and Satellite 3 will go into Receiving Mode, expecting a message from Satellite 1. The other two cases (where the environment chooses Satellite 2 or Satellite 3) are symmetric to this case."

Dynamic representations, on the other hand, abstract from explicit colors. In their language, the above would read

> "There are two groups of satellites. The first group contains one satellite, the second group contains two satellites. The satellite in the first group constitutes the one chosen by the environment, and it goes into Forwarding Mode. The satellites in the second group go into Receiving Mode, expecting a message from the satellite in the first group. The explicit scenarios are the instantiations of this abstraction."

In the approach of dynamic representations, the two satellites not chosen by the environment are put into the same group since they behave completely symmetrically.

The term "groups" used above describes what we later term *dynamic subclasses.* They are the result of a *dynamic partition* of the basic colors in the given symmetric Petri game. The dynamic subclasses then replace explicit colors in a so-called *dynamic decision set.* A dynamic partition together with a dynamic decision set yields a *dynamic representation.* The instantiations of the dynamic decision set, each given by a *valid assignment* of explicit colors to dynamic subclasses, are then precisely the decision sets contained in the represented symbolic decision set.

In general, there is more than one dynamic representation of a given symbolic decision set. The goal is to find one that is "canonical". This is achieved by defining two criteria such a canonical representation must satisfy. Namely, it must be *minimal* w.r.t. the number of dynamic subclasses, and *ordered* in a certain way. We show how to construct a unique minimal and ordered dynamic representation, and prove that there is exactly one such dynamic representation for every symbolic decision set. This will then be the canonical representation. As mentioned before, these canonical representations will later replace the arbitrary representatives of symbolic decision sets as vertices in the symbolic two-player game.

Dynamic Representations

As explained above, a dynamic representation $\mathscr{R}$ of a symbolic decision set D consists of a dynamic partition $\mathcal{P}$ and a dynamic decision set $\mathscr{D}$. The dynamic partition describes

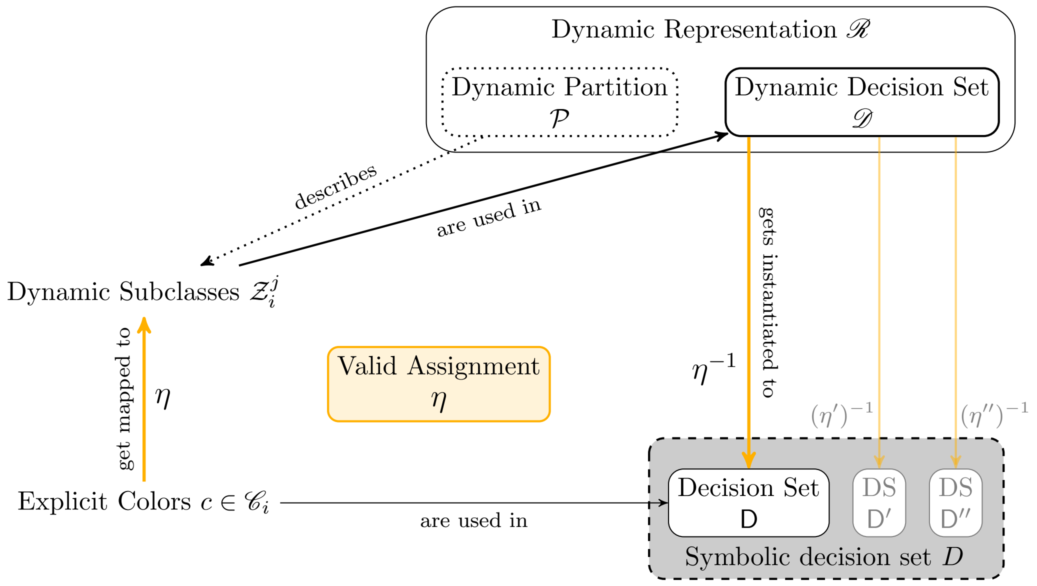

Dynamic Representation $\mathscr{R}$
Dynamic Partition $\mathcal{P}$
Dynamic Decision Set $\mathscr{D}$
describes
are used in
gets instantiated to
Dynamic Subclasses $\mathcal{Z}_i^j$
get mapped to
η
Valid Assignment η
η^{-1}
$(\eta')^{-1}$
$(\eta'')^{-1}$
Explicit Colors $c \in \mathscr{C}_i$
are used in
Decision Set D
DS D′
DS D″
Symbolic decision set D

Figure 3.16: An overview of the components constituting a dynamic representation, and their relation to the (elements of the) represented symbolic decision set.

dynamic subclasses $\mathcal{Z}_i^j$, that are used in the dynamic decision set in place of colors $c \in \mathscr{C}_i$. When the dynamic subclasses are assigned to explicit colors by a so-called valid assignment η, this describes an instance of the dynamic decision set. Each valid assignment describes an element D in the represented symbolic decision set D. An overview is pictured in Fig. 3.16.

We define dynamic representations by first giving the definitions of dynamic partition and valid assignments. Both these concepts are illustrated on a generic example. When we finally get to the definition of dynamic representations, we consult again our running example Signal Sending Satellites. We now start with the definition of dynamic partitions.

A dynamic partition consists of three elements. A function m that describes how many dynamic subclasses $\mathcal{Z}_i^j$ there are for each basic color class $\mathscr{C}_i$. These dynamic subclasses each represent a fixed number of basic colors from a single dynamic subclass $\mathscr{C}_{i,q}$. This is formalized by a function stat (describing that a dynamic subclass $\mathcal{Z}_i^j$ represents colors from $\mathscr{C}_{i,q}$ with $q = \mathrm{stat}(\mathcal{Z}_i^j)$) and a function card (describing the number $\mathrm{card}(\mathcal{Z}_i^j)$ of colors that each $\mathcal{Z}_i^j$ represents). The formal definition follows now.

Definition 3.43 (Dynamic partition). Let $\mathscr{C}$ be a family of basic color classes as in Def. 3.36. A *dynamic partition* of $\mathscr{C}$ is a tuple $\mathcal{P} = (m, \mathrm{stat}, \mathrm{card})$ with the following components:

- $m : I \to \mathbb{N}^+$ is a function describing for every color class $\mathscr{C}_i$ the *number* $m(i)$ (which will also be denoted m_i) *of dynamic subclasses in* $\mathscr{C}_i$.

  The set of *dynamic subclasses* in $\mathscr{C}_i$ is denoted $\mathscr{Z}_i = \{\mathcal{Z}_i^j \mid 1 \leq j \leq m(i)\}$. We additionally introduce the notation $\mathscr{Z} = \{\mathscr{Z}_i \mid i \in I\}$, analogously to the structure of $\mathscr{C}$ and $\mathscr{C}_i$ in symmetric nets.

- $\text{stat} : \left(\bigcup_{i\in I} \mathscr{Z}_i\right) \to \mathbb{N}^+$ is a function mapping every $\mathcal{Z}_i^j$ to an index $q = \text{stat}(\mathcal{Z}_i^j)$ of a static subclass $\mathscr{C}_{i,q}$.

  We denote $\mathscr{Z}_{i,q} = \{\mathcal{Z}_i^j \mid \text{stat}(\mathcal{Z}_i^j) = q\}$ such that, analogously to static color classes from Def. 3.36, we have $\mathscr{Z}_i = \bigsqcup_{q=1}^{n_i} \mathscr{Z}_{i,q}$.
- $\text{card} : \left(\bigcup_{i\in I} \mathscr{Z}_i\right) \to \mathbb{N}^+$ describes the *cardinality* of the dynamic subclasses.

Together, these components must satisfy

$$\forall i \in I\, \forall 1 \le q \le n_i : \sum_{\substack{1\le j\le m_i,\\ \text{stat}(\mathcal{Z}_i^j)=q}} \text{card}(\mathcal{Z}_i^j) = |\mathscr{C}_{i,q}|. \qquad \lhd$$

The last condition describes that for every $\mathscr{C}_{i,q}$, the dynamic subclasses represent exactly $|\mathscr{C}_{i,q}|$ colors from this static subclass. We will see this ensures that every *valid assignment* (cp. Def. 3.44 below) of dynamic subclasses to colors induces a *partition* of every $\mathscr{C}_i$.

We illustrate the concept of dynamic partitions, i.e., the role of m, stat and card, in a generic example, shown in Fig. 3.17. The figure can be seen as "zooming in" on the left part of Fig. 3.16. For the moment, disregard the yellow arrows depicted in the figure. At the bottom level, we see two basic color classes $\mathscr{C}_1$ (not divided into static subclasses, i.e., $\mathscr{C}_1 = \mathscr{C}_{1,1}$) and $\mathscr{C}_2$ (divided into the three static subclasses $\mathscr{C}_{2,1}, \mathscr{C}_{2,2}, \mathscr{C}_{2,3}$). The dots • are arbitrary colors belonging to the respective static subclass.

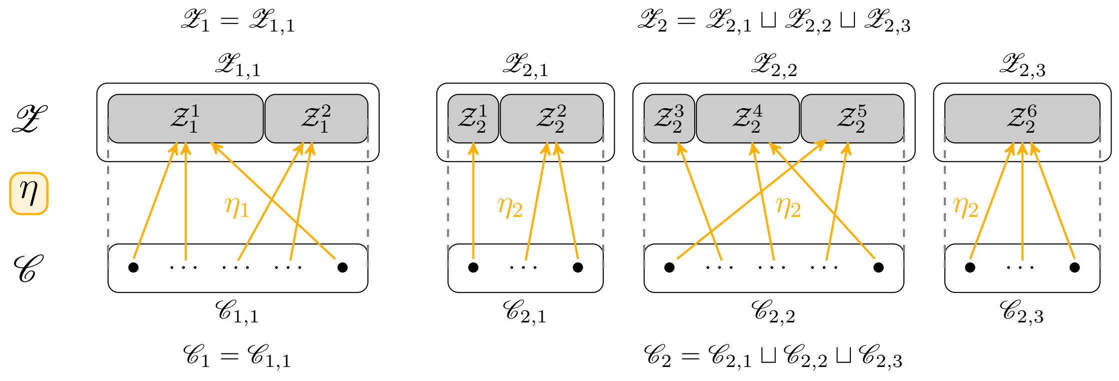


Figure 3.17: A schematic depiction of a dynamic partition of basic color classes and a valid assignment for a generic example.

Assume now $m(1) = 2$, $m(2) = 6$, This leads to the dynamic subclasses $\mathscr{Z} = \{\{\mathcal{Z}_1^1, \mathcal{Z}_1^2\}, \{\mathcal{Z}_2^1, \mathcal{Z}_2^2, \mathcal{Z}_2^3, \mathcal{Z}_2^4, \mathcal{Z}_2^5, \mathcal{Z}_2^6\}\}$, depicted at the top level. Since we have $\mathscr{C}_1 = \mathscr{C}_{1,1}$, it must trivially be $\text{stat}(\mathcal{Z}_1^1) = \text{stat}(\mathcal{Z}_1^2) = 1$. Additionally assume $\text{stat}(\mathcal{Z}_2^1) = \text{stat}(\mathcal{Z}_2^2) = 1$, $\text{stat}(\mathcal{Z}_2^3) = \text{stat}(\mathcal{Z}_2^4) = \text{stat}(\mathcal{Z}_2^5) = 2$, and $\text{stat}(\mathcal{Z}_2^6) = 3$. Every dynamic subclass $\mathcal{Z}_i^j$ is depicted above the respective static subclass $\mathscr{C}_{i,q}$ with $q = \text{stat}(\mathcal{Z}_i^j)$.

The visual size of a dynamic subclass $\mathcal{Z}_i^j$ represents the value $\text{card}(\mathcal{Z}_i^j)$. Analogously, the visual size of a static subclasses $\mathscr{C}_{i,q}$ represents $|\mathscr{C}_{i,q}|$. In the figure, these sizes illustrate the equation $\forall i \in I\, \forall 1 \leq q \leq m_i : \sum_{\text{stat}(\mathcal{Z}_i^j)=q} \text{card}(\mathcal{Z}_i^j) = |\mathscr{C}_{i,q}|$, emphasized by the vertical dashed lines.

Next, we proceed with the definition of *valid assignments* within a dynamic partition, which relate colors in the basic color classes and dynamic subclasses.

Definition 3.44 (Valid assignment). Given a dynamic partition $\mathcal{P} = (m, \text{card}, \text{stat})$ of a family of basic color classes $\mathscr{C}$ as in Def. 3.43, a *valid assignment* of dynamic subclasses to basic colors is a tuple $\eta = (\eta_i)_{i=1}^n$ of functions $\eta_i : \mathscr{C}_i \to \mathscr{Z}_i$ satisfying for all $\mathcal{Z}_i^j$:

- $\eta_i^{-1}(\mathcal{Z}_i^j) \subseteq \mathscr{C}_{i,q}$, where $q = \text{stat}(\mathcal{Z}_i^j)$, i.e., the colors mapped to $\mathcal{Z}_i^j$ are from $\mathscr{C}_{i,q}$,
- $|\eta_i^{-1}(\mathcal{Z}_i^j)| = \text{card}(\mathcal{Z}_i^j)$, i.e., the correct number of colors is mapped to $\mathcal{Z}_i^j$,
- if $i > \mathbf{u}$ then $\exists c \in \eta_i^{-1}(\mathcal{Z}_{i,q}^j)$:

$$\eta_i(\text{suc}(c)) = \mathcal{Z}_i^{\text{suc}(j)} \quad \wedge \quad \forall c' \in \eta_i^{-1}(\mathcal{Z}_i^j) : c' \neq c \Rightarrow \eta_i(\text{suc}(c')) = \mathcal{Z}_i^j,$$

where, for a dynamic subclass $\mathcal{Z}_i^j$, we define $\text{suc}(j) = (j \mod m_i) + 1$. $\lhd$

The third condition formalizes that in an ordered ($i > \mathbf{u}$) basic color class $\mathscr{C}_i$, two successive colors are either assigned to the same dynamic subclass, or to successive dynamic subclasses, i.e., successive colors are grouped in the assignment. The existentially quantified c is the “last” element assigned to a dynamic subclass $\mathcal{Z}_i^j$, before switching to the next dynamic subclass $\mathcal{Z}_i^{\text{suc}(j)}$.

We continue our illustration for a generic example from Fig. 3.17. Between the two levels, we see a schematic depiction of a valid assignment mapping colors in $\mathscr{C}_{i,q}$ to dynamic subclasses $\mathcal{Z}_i^j$. The first condition is visualized by seeing that every color in $\mathscr{C}_{i,q}$ is assigned to a $\mathcal{Z}_i^j$ with $\text{stat}(\mathcal{Z}_i^j) = q$, i.e., to a dynamic subclass depicted above the static subclass. For the second condition, recall that the visual size of a dynamic subclass $\mathcal{Z}_i^j$ corresponds to the value $\text{card}(\mathcal{Z}_i^j)$. This is visually depicted by a correspondence between the visual size of each dynamic subclass and the number of incoming arrows, which in turn represent the colors assigned to that dynamic subclass.

For simplicity, we assume that both color classes $\mathscr{C}_1$ and $\mathscr{C}_2$ are unordered, i.e., in the terms of Def. 3.36, we have $\mathbf{u} = 2$. In this case, the third condition is irrelevant. However, for $i = 1$, the illustration suggests that the third condition was satisfied if $\mathscr{C}_i$ was ordered with suc orbiting from left to right through $\mathscr{C}_i$, and mapping the rightmost color to the leftmost color. As a result, we observe that when starting from the rightmost color, all subsequent elements are assigned to $\mathcal{Z}_1^1$, until η_i switches to assigning all subsequent elements to $\mathcal{Z}_1^2$, ultimately reaching the rightmost color again.

By definition, a valid assignment only maps individual colors $c \in \mathscr{C}_i$ from basic subclasses to dynamic subclasses. However, color domains of places and transitions contain *tuples* of such colors, which are then used for example in decision sets. Since dynamic subclasses will replace the explicit colors, we now naturally lift the definition of valid assignments to map tuples of colors to tuples of dynamic subclasses.

Let $G = (\mathscr{C}, P_{\mathrm{S}}, P_{\mathrm{E}}, T, J, F, g, M_0, \mathrm{Obj}, P_{\circledast})$ be a symmetric Petri game, $\mathcal{P}$ a dynamic partition of $\mathscr{C}$, and $\eta = (\eta_i)_{i=1}^n$ a valid assignment mapping colors to dynamic subclasses. Let $p \in P$ and $c = \big((c_i^j)_{j=1}^{J(p,i)}\big)_{i=1}^n \in \mathscr{C}(p)$. Then we can apply η to c component-wise, i.e.,

$$\eta(c) := \Big(\big(\eta_i(c_i^j)\big)_{j=1}^{J(p,i)}\Big)_{i=1}^n.$$

Analogously, for a transition t and a color $\sigma = \big((\sigma_i^j)_{j=1}^{J(t,i)}\big)_{i=1}^n \in \mathscr{C}(t)$ we get

$$\eta(\sigma) := \Big(\big(\eta_i(\sigma_i^j)\big)_{j=1}^{J(t,i)}\Big)_{i=1}^n.$$

The definition of symmetries in a symmetric net (Def. 3.39) and the definition of valid assignments above are “compatible” with each other, so that one can easily derive the following property.

Property 3.45. *Let G be a symmetric Petri game with the family $\mathscr{C} = \{\mathscr{C}_1, \dots, \mathscr{C}_n\}$ of basic color classes. Let $\mathcal{P}$ be a dynamic partition of $\mathscr{C}$ and let $\eta = (\eta_i)_{i=1}^n$ be a valid assignment. Then the set of* all *valid assignments is given by*

$$\{(\eta_i \circ s_i)_{i=1}^n \mid s = (s_i)_{i=1}^n \in S_G\}.$$

The idea of dynamic partitions is to represent a *set* of partitions of the basic colors, instantiated by all valid assignments. In an instantiation of a dynamic representation of a decision set, all colors mapped to the same dynamic subclass behave equally. In contrast to regular decision sets, dynamic decision sets feature tuples of dynamic subclasses instead of tuples of explicit colors. In decision sets, these tuples are elements of the color domain associated with a place or transition. We now define the *symbolic color domains* of nodes.

The *symbolic color domain* w.r.t. a dynamic partition is of a node $r \in P \cup T$ is $\mathscr{Z}(r) := \mathscr{Z}_1^{J(r,1)} \times \cdots \times \mathscr{Z}_n^{J(r,n)}$. Analogously to the case of basic color classes, we call the symbolic color domain of a place p its *symbolic type*, and denote it by $\widehat{ty}(p)$. We will explore the notion of *symbolic modes* of transitions later.

With this definition in mind, we can now make the next step towards the definition of dynamic decision sets and thereby dynamic representations. Remember that a decision set $\mathsf{D} \in \mathcal{D}(G)$ is a set $\mathsf{D} \subseteq \mathsf{P} \times (\mathbb{P}(\mathsf{T}) \cup \{\dagger\})$, where we have $\mathsf{P} = \{p.c \mid P \in P, c \in \mathscr{C}(p)\}$ and $\mathsf{T} = \{t.\sigma \mid t \in T, \sigma \in \Sigma(t)\}$ with $\Sigma(t) \subseteq \mathscr{C}(t)$, i.e., $\mathsf{T} \subseteq \{t.\sigma \mid t \in T, \sigma \in \mathscr{C}(t)\}$ (cp. Def. 3.8). We now lift the notion of P and T to the symbolic level of dynamic subclasses. For that, we denote

$$\mathscr{P} = \{p.\widehat{c} \mid p \in P, \widehat{c} \in \mathscr{Z}(p)\} \quad \text{and} \quad \mathscr{T} = \{t.\widehat{\sigma} \mid t \in T, \widehat{\sigma} \in \mathscr{Z}(t)\}.$$

Note that, unlike for $t.\sigma \in \mathsf{T}$, we do permit for $t.\widehat{\sigma} \in \mathscr{T}$ that $\widehat{\sigma}$ is *any* element from $\mathscr{Z}(t)$. For $t.\sigma \in \mathsf{T}$, we required $\sigma \in \mathscr{C}(t)$ to satisfy $g(t)(\sigma) = \mathit{true}$, i.e., to be a *mode* $\sigma \in \Sigma(t)$. However, this distinction will not pose an issue within the definition of representations for symbolic decision sets. This is due to a requirement that every "instantiation" of the representation must constitute a decision set. This requirement ensures that only modes of t are represented. Given these sets, representations will contain what we term a *dynamic decision set* $\mathscr{D} \subseteq \mathscr{P} \times (\mathbb{P}(\mathscr{T}) \cup \dagger)$, such that $\mathscr{P}$ and $\mathscr{T}$ replace P and T as compared to the aforementioned recalled decision sets.

Before we come to the formal definition of dynamic representations, we now already give an example to provide the reader with some intuition.

Example 3.46 (Dynamic representation). Consider the running example Signal Sending Satellites as the symmetric Petri game from Fig. 3.14, with basic color classes $\mathscr{C}_1 = \mathscr{C}_{1,1} = \{\mathbf{0}\}$ and $\mathscr{C}_2 = \mathscr{C}_{2,1} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$. Assume a scenario where the satellite chosen by the environment informed the other two, and all satellites went into the correct mode. This describes one of the three (symmetric) scenarios where either Satellite 1, 2, or 3 has been chosen. The chosen satellite is in Forward Mode, and the other two satellites are in Receiving Mode. Assuming additionally (now at the level of P/T Petri game semantics) that all satellites allow the corresponding *end* transition to fire, this corresponds to the three decision sets $\mathsf{D}', \mathsf{D}'', \mathsf{D}'''$ at the bottom of Fig. 3.18. These decision sets, since they are symmetric, are in the same symbolic decision set $D = [\mathsf{D}']$, i.e., the same equivalence class.

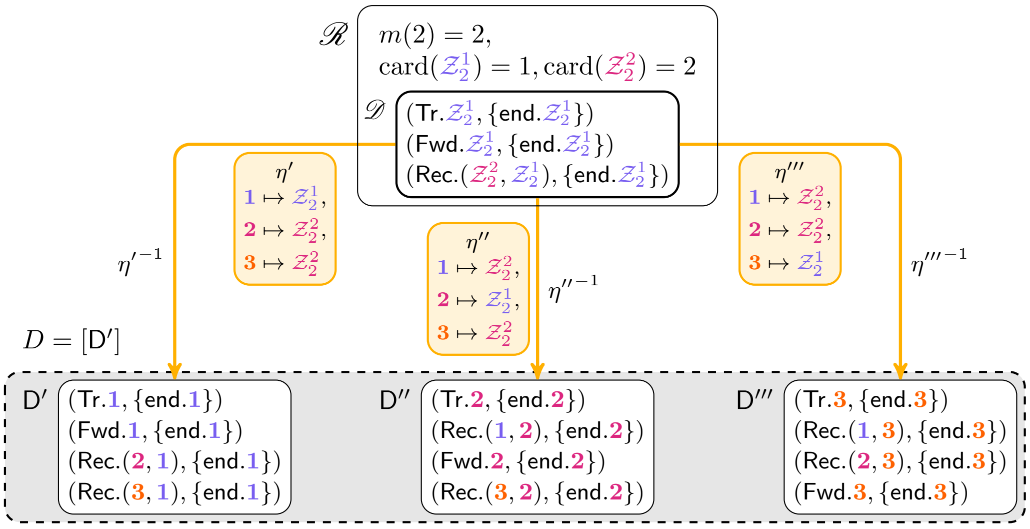

$\mathscr{R}$
$m(2) = 2$,
$\mathrm{card}(\mathcal{Z}_2^1) = 1, \mathrm{card}(\mathcal{Z}_2^2) = 2$
$\mathscr{D}$
$(\mathsf{Tr}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Fwd}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Rec}.(\mathcal{Z}_2^2, \mathcal{Z}_2^1), \{\mathsf{end}.\mathcal{Z}_2^1\})$
η' $\mathbf{1} \mapsto \mathcal{Z}_2^1$, $\mathbf{2} \mapsto \mathcal{Z}_2^2$, $\mathbf{3} \mapsto \mathcal{Z}_2^2$
η'' $\mathbf{1} \mapsto \mathcal{Z}_2^2$, $\mathbf{2} \mapsto \mathcal{Z}_2^1$, $\mathbf{3} \mapsto \mathcal{Z}_2^2$
η''' $\mathbf{1} \mapsto \mathcal{Z}_2^2$, $\mathbf{2} \mapsto \mathcal{Z}_2^2$, $\mathbf{3} \mapsto \mathcal{Z}_2^1$
η'^{-1}
η''^{-1}
η'''^{-1}
$D = [\mathsf{D}']$
D'
$(\mathsf{Tr}.\mathbf{1}, \{\mathsf{end}.\mathbf{1}\})$
$(\mathsf{Fwd}.\mathbf{1}, \{\mathsf{end}.\mathbf{1}\})$
$(\mathsf{Rec}.(\mathbf{2}, \mathbf{1}), \{\mathsf{end}.\mathbf{1}\})$
$(\mathsf{Rec}.(\mathbf{3}, \mathbf{1}), \{\mathsf{end}.\mathbf{1}\})$
D''
$(\mathsf{Tr}.\mathbf{2}, \{\mathsf{end}.\mathbf{2}\})$
$(\mathsf{Rec}.(\mathbf{1}, \mathbf{2}), \{\mathsf{end}.\mathbf{2}\})$
$(\mathsf{Fwd}.\mathbf{2}, \{\mathsf{end}.\mathbf{2}\})$
$(\mathsf{Rec}.(\mathbf{3}, \mathbf{2}), \{\mathsf{end}.\mathbf{2}\})$
D'''
$(\mathsf{Tr}.\mathbf{3}, \{\mathsf{end}.\mathbf{3}\})$
$(\mathsf{Rec}.(\mathbf{1}, \mathbf{3}), \{\mathsf{end}.\mathbf{3}\})$
$(\mathsf{Rec}.(\mathbf{2}, \mathbf{3}), \{\mathsf{end}.\mathbf{3}\})$
$(\mathsf{Fwd}.\mathbf{3}, \{\mathsf{end}.\mathbf{3}\})$

Figure 3.18: A dynamic representations of a symbolic decision set.

We see that in all three decision sets, the two satellites in Receiving Mode occur "in the same contexts". In D' for example, both $\mathbf{2}$ and $\mathbf{3}$ occur exactly at the position of $\triangledown$ in a tuple $(Rec.(\triangledown, \mathbf{1}), \{end.\mathbf{1}\})$. We now describe a dynamic representation where a dynamic subclass $\mathcal{Z}_2^1$ with $\mathrm{card}(\mathcal{Z}_2^1) = 1$ represents the satellite in Forward Mode, and $\mathcal{Z}_2^2$ with $\mathrm{card}(\mathcal{Z}_2^2) = 2$ represents the two satellites in Receiving Mode.

This means we need two dynamic subclasses for the basic color class $\mathscr{C}_2$ of satellites, thus $m(2) = 2$ in a corresponding dynamic partition. We therefore have $\mathscr{Z}_2 = \mathscr{Z}_{2,1} =$

$\{\mathcal{Z}_2^1, \mathcal{Z}_2^2\}$. Since for the basic color class $\mathscr{C}_1 = \mathscr{C}_{1,1}$ we have $|\mathscr{C}_1| = 1$, *every* dynamic partition has to satisfy $m(1) = 1$ (so that $\mathscr{Z}_1 = \mathscr{Z}_{1,1} = \{\mathcal{Z}_1^1\}$) and $\mathrm{card}(\mathcal{Z}_1^1) = 1$.

Both basic color classes in this example are not partitioned into static subclasses ($\mathscr{C}_1 = \mathscr{C}_{1,1}$ and $\mathscr{C}_2 = \mathscr{C}_{2,1}$). Thus, we trivially have for all $\mathcal{Z}_i^j$ that $\mathrm{stat}(\mathcal{Z}_i^j) = 1$.

With the intention that $\mathcal{Z}_2^1$ should represent the satellite in Forwarding Mode and $\mathcal{Z}_2^2$ should represent the two satellites in Receiving Mode, we set $\mathrm{card}(\mathcal{Z}_2^1) = 1$ and $\mathrm{card}(\mathcal{Z}_2^2) = 2$. The dynamic partition therefore satisfies $\mathrm{card}(\mathcal{Z}_2^1)+\mathrm{card}(\mathcal{Z}_2^2) = 3 = |\mathscr{C}_{2,1}|$, as required in Def. 3.47.

The concept of a dynamic representation involves the introduction of a *dynamic decision set* denoted as $\mathscr{D}$, wherein the distinct colors within decision sets are substituted with the dynamic subclasses derived from the specified dynamic partition. Each decision set D contained within the represented symbolic decision set D can be mapped to the dynamic decision set $\mathscr{D}$ through an appropriate valid assignment η.

In the illustration, the dynamic decision set $\mathscr{D}$ is presented at the top of Fig 3.18. This figure can be seen as "zooming in" on the right part of Fig. 3.16 for this scenario. Compared to each decision set at the lower level, the corresponding satellite in Forwarding Mode is substituted with $\mathcal{Z}_2^1$. Instead of having two tuples for the two satellites in Receiving Mode, the dynamic decision set $\mathscr{D}$ contains a single tuple $(\mathit{Rec}.(\mathcal{Z}_2^2, \mathcal{Z}_2^1), \mathit{end}.\mathcal{Z}_2^1)$. Above the dynamic decision set, the associated dynamic partition discussed earlier is indicated. Note that we haven't explicitly shown the trivial elements $m(1) = 1$ and $\mathrm{card}(\mathcal{Z}_1^1) = 1$, and the values $\mathrm{stat}(\mathcal{Z}_i^j) = 1$.

The presented valid assignments η', η'', and η''' respectively map the three decision sets D', D'', and D''' to $\mathscr{D}$. It is important to observe that $\mathscr{D}$ contains three tuples, while all the represented decision sets contains four tuples. This difference arises because each valid assignment $\eta^{(\ell)}$ maps the two tuples in $\mathsf{D}^{(\ell)}$ containing satellites in Receiving Mode to the same tuple (containing $\mathcal{Z}_2^2$) in $\mathscr{D}$. Additionally, note that every valid assignment trivially maps $\mathbf{0}$ to $\mathcal{Z}_1^1$, which therefore is not explicitly indicated in the figure. $\lhd$

We come back to this example after the following formal definition of dynamic representations. For a better understanding, we suggest keeping Fig. 3.16 in mind while reading the definition.

Definition 3.47 (Dynamic representation). A *dynamic representation* of a symbolic decision set D in a symmetric Petri game $G = (\mathscr{C}, P_\mathrm{S}, P_\mathrm{E}, T, J, F, g, M_0, \mathrm{Obj}, P_\circledast)$ is a tuple $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ with the following components:

- $\mathcal{P} = (m, \mathrm{card}, \mathrm{stat})$ is a dynamic partition of $\mathscr{C}$.
- $\mathscr{D} \subseteq \mathscr{P} \times (\mathbb{P}(\mathscr{T}) \cup \{\dagger\})$ is the *dynamic decision set*, satisfying that D contains exactly the decision sets D such that there is a valid assignment η satisfying that
  - for a tuple $(p.c, \dagger)$ with $c \in \mathscr{C}(p)$ we have
$$(p.c, \dagger) \in \mathsf{D} \quad \Leftrightarrow \quad (p.\eta(c), \dagger) \in \mathscr{D},$$
  - and for a commitment set $\mathcal{T} \in \mathbb{P}(\mathsf{T})$ of place $p.c$ we have
$$(p.c, \mathcal{T}) \in \mathsf{D} \quad \Leftrightarrow \quad \exists \widehat{\mathcal{T}} \in \mathbb{P}(\mathscr{T}) : (p.\eta(c), \widehat{\mathcal{T}}) \in \mathscr{D} \;\wedge\; \mathcal{T} = \{t.\sigma \mid t.\eta(\sigma) \in \widehat{\mathcal{T}}\}.$$

We denote such a decision set D by $\eta^{-1}(\mathscr{D})$. $\lhd$

We explain the above conditions on the dynamic decision set. The first case of tuples with a †-symbol is the easier one: a tuple $(p.\widehat{c}, \dagger) \in \mathscr{D}$ represents that for every valid assignment η, *all* tuples $(p.c, \dagger)$ with $\eta(c) = \widehat{c}$ are in the corresponding decision set D that is an element of the represented symbolic decision set D. Understanding that, the second case, handling commitment sets, is not much harder. It means a tuple $(p.\widehat{c}, \widehat{\mathcal{T}}) \in \mathscr{D}$ represents that all tuples $(p.c, \mathcal{T})$ with $\eta(c) = \widehat{c}$ are in D, where $\mathcal{T}$ in its turn contains *all* $t.\sigma$ with $t.\eta(\sigma) \in \widehat{\mathcal{T}}$.

Note that these conditions differ from a condition like "$\mathsf{D} \in D \Leftrightarrow \exists \eta : \eta(\mathsf{D}) = \mathscr{D}$", where η would be applied to D by applying it to every component. While this relation between D and $\mathscr{D}$ is true for every decision set $\mathsf{D} \in D$ (i.e., the direction "$\Rightarrow$" holds), not every D in this relation is represented by $\mathscr{R}$ (i.e., the direction "$\Leftarrow$" does not hold). The reason for this is that valid assignments are not necessarily bijections.

We explore this on the example from Fig. 3.18. We already described that each $\eta^{(\ell)}$ maps $\mathsf{D}^{(\ell)}$ to D. In each of these mappings, two tuples are mapped to $\mathscr{D}$. However, this does not imply that the decision sets are in the represented symbolic decision set. They are in the represented symbolic decision set because they satisfy the conditions from above. We exemplarily examine the case of η' and $\mathsf{D}' = (\eta')^{-1}(\mathscr{D})$. For the first two entries in $\mathscr{D}$ we get, since $\mathcal{Z}_2^1$ is only assigned to the color $\mathbf{1}$, that $(\mathit{Tr}.\mathbf{1}, \{\mathit{end}.\mathbf{1}\})$ and $(\mathit{Fwd}.\mathbf{1}, \{\mathit{end}.\mathbf{1}\})$ are in $(\eta')^{-1}(\mathscr{D})$. The dynamic subclass $\mathcal{Z}_2^2$, however, is assigned to *two* colors by η', namely $\mathbf{2}$ and $\mathbf{3}$. This means that from $(\mathit{Rec}.(\mathcal{Z}_2^2, \mathcal{Z}_2^1), \{\mathit{end}.\mathcal{Z}_2^1\}) \in \mathscr{D}$ follows $(\mathit{Rec}.(\mathbf{2}, \mathbf{1}), \{\mathit{end}.\mathbf{1}\}) \in (\eta')^{-1}(\mathscr{D})$ and $(\mathit{Rec}.(\mathbf{3}, \mathbf{1}), \{\mathit{end}.\mathbf{1}\}) \in (\eta')^{-1}(\mathscr{D})$.

A negative example, i.e., a decision set that is *not* represented by $\mathscr{R}$, is

$$\begin{aligned}\mathsf{D}^- = \{&(\mathit{Tr}.\mathbf{1}, \{\mathit{end}.\mathbf{1}\})\\ &(\mathit{Fwd}.\mathbf{1}, \{\mathit{end}.\mathbf{1}\})\\ &(\mathit{Rec}.(\mathbf{2}, \mathbf{1}), \{\mathit{end}.\mathbf{1}\})\}.\end{aligned}$$

Although we also have $\eta'(\mathsf{D}^-) = \mathscr{D}$, this decision set does not satisfy the conditions from Def. 3.47, since $(\mathit{Rec}.(\mathbf{3}, \mathbf{1}), \{\mathit{end}.\mathbf{1}\})$ is not contained in D^-.

We will see that in general, there are often several dynamic representations of a symbolic decision set. We start by stating that for every symbolic decision set, there is at least one dynamic representation.

Lemma 3.48. *Every symbolic decision set has a dynamic representation.*

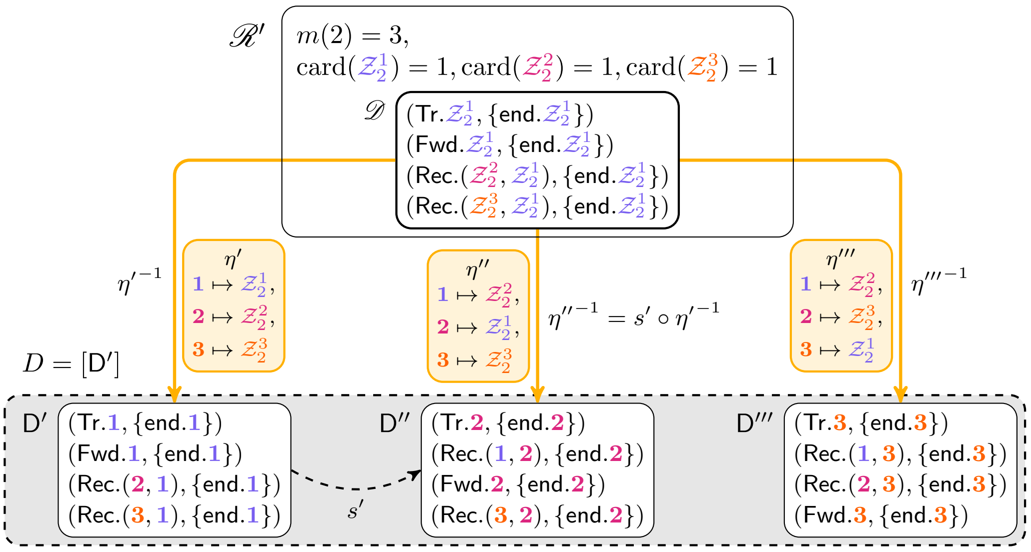
Proof. For every symbolic decision set there is a "trivial" encoding as a representation that we now demonstrate. Let $\mathcal{P} = (m, \mathrm{stat}, \mathrm{card})$ be the dynamic partition of basic color classes such that for every $i \in I$, we have $m(i) = |\mathscr{C}_i|$, and for every $1 \leq j \leq m(i)$ we have $\mathrm{card}(\mathcal{Z}_i^j) = 1$. This means $\mathscr{Z}_i = \{\mathcal{Z}_i^1, \dots, \mathcal{Z}_i^{|\mathscr{C}_i|}\}$, and every valid assignment partitions the basic color classes $\mathscr{C}_i$ into singletons – each color is represented by another dynamic subclass. For the values of stat, we first rename the basic colors such that we have $\mathscr{C}_i = \{c_i^1, \dots, c_i^{|\mathscr{C}_i|}\}$ for every basic color class $\mathscr{C}_i$. In the case of an ordered class $\mathscr{C}_i$

(i.e. $i > \mathbf{u}$), we additionally assure that $c_i^{\mathrm{suc}(j)} = \mathrm{suc}(c_i^j)$ for every $1 \leq j \leq |\mathscr{C}_i|$. We then set $\mathrm{stat}(\mathcal{Z}_i^j) = q \Leftrightarrow c_i^j \in \mathscr{C}_{i,q}$.

Let $D = [\mathsf{D}]$ be a symbolic decision set and $\overline{\mathsf{D}}$ an arbitrary representative. We build $\mathscr{D}$ from $\overline{\mathsf{D}}$ as follows. The dynamic decision set $\mathscr{D}$ then is the result from replacing every appearing color c_i^j in D by the dynamic subclass $\mathcal{Z}_i^j$. We therefore have $\mathscr{D} \subseteq \mathscr{P} \times (\mathbb{P}(\mathscr{T}) \cup \{\dagger\})$.

We obviously have for $\eta = (\eta_i)_{i \in I}$ with $\eta_i = \{c_i^j \mapsto \mathcal{Z}_i^j \mid 1 \leq j \leq |\mathscr{C}_i|\}$ that $\overline{\mathsf{D}} = \eta^{-1}(\mathscr{D})$. Using now Prop. 3.45, we get that all decision sets of the form $\widetilde{\eta}^{-1}(\mathscr{D})$, where $\widetilde{\eta}$ is valid assignment, are symmetric to $\overline{\mathsf{D}}$, since there is a symmetry s such that $\widetilde{\eta} = \eta \circ s$. On the other hand, all decision sets symmetric to $\overline{\mathsf{D}}$ can be reached by such an inverse of an assignment. Therefore, the representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ represents the symbolic decision set $D = [\overline{\mathsf{D}}]$. □

We call the dynamic partition as in the proof above *trivial*. A representation with the trivial partition is also called *trivial*. We present such a trivial representation for the symbolic decision set from Fig. 3.18.



$\mathscr{R}'$
$m(2) = 3,$
$\mathrm{card}(\mathcal{Z}_2^1) = 1, \mathrm{card}(\mathcal{Z}_2^2) = 1, \mathrm{card}(\mathcal{Z}_2^3) = 1$
$\mathscr{D}$
$(\mathsf{Tr}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Fwd}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Rec}.(\mathcal{Z}_2^2, \mathcal{Z}_2^1), \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Rec}.(\mathcal{Z}_2^3, \mathcal{Z}_2^1), \{\mathsf{end}.\mathcal{Z}_2^1\})$
η'^{-1}
η' $\mathbf{1} \mapsto \mathcal{Z}_2^1, \mathbf{2} \mapsto \mathcal{Z}_2^2, \mathbf{3} \mapsto \mathcal{Z}_2^3$
η'' $\mathbf{1} \mapsto \mathcal{Z}_2^2, \mathbf{2} \mapsto \mathcal{Z}_2^1, \mathbf{3} \mapsto \mathcal{Z}_2^3$
$\eta''^{-1} = s' \circ \eta'^{-1}$
η''' $\mathbf{1} \mapsto \mathcal{Z}_2^2, \mathbf{2} \mapsto \mathcal{Z}_2^3, \mathbf{3} \mapsto \mathcal{Z}_2^1$
η'''^{-1}
$D = [\mathsf{D}']$
D' $(\mathsf{Tr}.\mathbf{1}, \{\mathsf{end}.\mathbf{1}\})$ $(\mathsf{Fwd}.\mathbf{1}, \{\mathsf{end}.\mathbf{1}\})$ $(\mathsf{Rec}.(\mathbf{2}, \mathbf{1}), \{\mathsf{end}.\mathbf{1}\})$ $(\mathsf{Rec}.(\mathbf{3}, \mathbf{1}), \{\mathsf{end}.\mathbf{1}\})$
s'
D'' $(\mathsf{Tr}.\mathbf{2}, \{\mathsf{end}.\mathbf{2}\})$ $(\mathsf{Rec}.(\mathbf{1}, \mathbf{2}), \{\mathsf{end}.\mathbf{2}\})$ $(\mathsf{Fwd}.\mathbf{2}, \{\mathsf{end}.\mathbf{2}\})$ $(\mathsf{Rec}.(\mathbf{3}, \mathbf{2}), \{\mathsf{end}.\mathbf{2}\})$
D''' $(\mathsf{Tr}.\mathbf{3}, \{\mathsf{end}.\mathbf{3}\})$ $(\mathsf{Rec}.(\mathbf{1}, \mathbf{3}), \{\mathsf{end}.\mathbf{3}\})$ $(\mathsf{Rec}.(\mathbf{2}, \mathbf{3}), \{\mathsf{end}.\mathbf{3}\})$ $(\mathsf{Fwd}.\mathbf{3}, \{\mathsf{end}.\mathbf{3}\})$

Figure 3.19: A trivial dynamic representations of the symbolic decision set from Fig. 3.18.

Consider again the symbolic decision set D from Fig. 3.18. The result of constructing a trivial dynamic representation (as described in the proof above) of D is shown in Fig. 3.19. We assumed $\mathsf{D}' = \overline{\mathsf{D}'}$, and built $\mathscr{D}$ by replacing each of the colors $\mathbf{1}, \mathbf{2}, \mathbf{3}$ in D by a dynamic subclass $\mathcal{Z}_2^1, \mathcal{Z}_2^2, \mathcal{Z}_2^3$, respectively. Every $\mathcal{Z}_2^j$ has cardinality 1. An occurring color $\mathbf{0}$ would have been replaced by $\mathcal{Z}_1^1$. As in Fig. 3.18, we omitted the trivial elements $m(1) = 1$ and $\mathrm{card}(\mathcal{Z}_1^1) = 1$, and $\forall i, j : \mathrm{stat}(\mathcal{Z}_i^j) = 1$ in the figure.

The valid assignment η' maps D' to $\mathscr{D}$ and, since η' is bijective, we get $\mathsf{D}' = (\eta')^{-1}(\mathscr{D})$. Consider now the symmetry $s' = \{\mathbf{1} \mapsto \mathbf{2}, \mathbf{2} \mapsto \mathbf{1}, \mathbf{3} \mapsto \mathbf{3}\} \in S_G$ (where G is the symmetric

Petri game from Fig. 3.14), mapping D' to D''. We see that $\eta'' = \eta' \circ (s')^{-1}$. In general, we have for all η, and s that $s(\eta^{-1}(\mathscr{D})) = (\eta \circ s^{-1})^{-1}(\mathscr{D})$.

We have now observed two distinct dynamic representations of the same symbolic decision set D. In a general, there might exist numerous dynamic representations for the same symbolic decision set. The central objective of this section, as implied by its title, is to identify a representation that achieves both uniqueness and a certain "minimality". This representation, once determined, will be referred to as the *canonical* representation. We move forward by formalizing the concept of minimality and describing the process for constructing a minimal dynamic representation for a given symbolic decision set.

Minimality

When we compare the two dynamic representations from Fig. 3.18 and Fig. 3.19, we see that the non-trivial one from Fig. 3.18 is "smaller" in the sense that it uses less dynamic subclasses. So the first step towards a canonical representation will be to find one that has a minimal number of dynamic subclasses. Later we will see that minimal representations are unique up to a permutation (i.e., renaming) of the dynamic subclasses. Constructing a minimal representation is achieved by merging subclasses from non-minimal representations. This is now informally explained on the running example.

In the trivial representation from Fig. 3.19, the two dynamic subclasses $\mathcal{Z}_2^2$ and $\mathcal{Z}_2^3$ appear "in the same context" in $\mathscr{D}$. Both of them occur exactly at the position of ∇ in a tuple $(Rec.(\nabla, \mathcal{Z}_2^1), \{end.\mathcal{Z}_2^1\})$. We will see in Lemma 3.53 that this means the two dynamic subclasses can be "merged", or rather be replaced by a dynamic subclass of cardinality 2 (since $\mathrm{card}(\mathcal{Z}_2^2)+\mathrm{card}(\mathcal{Z}_2^3) = 1+1 = 2$). Doing this yields the representation from Fig. 3.18. Since there no two dynamic subclasses appear in the same context, this representation will be called "minimal".

We begin with the definition of merging two dynamic subclasses.

Definition 3.49 (Merging dynamic subclasses.). Let $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ be a dynamic representation of a symbolic decision set with such that the dynamic partition defines two dynamic subclasses $\mathcal{Z}_{\hat{\imath}}^j, \mathcal{Z}_{\hat{\imath}}^k$ with $j \neq k$. Then the dynamic representation $\mathscr{R}[\mathcal{Z}_{\hat{\imath}}^j \curlywedge \mathcal{Z}_{\hat{\imath}}^k] = (\mathcal{P}^*, \mathscr{D}^*)$ results from replacing every occurrence of $\mathcal{Z}_{\hat{\imath}}^j$ and $\mathcal{Z}_{\hat{\imath}}^k$ by a new dynamic subclass $\mathcal{Z}_{\hat{\imath}}^*$ with $\mathrm{card}^*(\mathcal{Z}_{\hat{\imath}}^*) = \mathrm{card}(\mathcal{Z}_{\hat{\imath}}^j) + \mathrm{card}(\mathcal{Z}_{\hat{\imath}}^k)$ and $\mathrm{stat}^*(\mathcal{Z}_{\hat{\imath}}^*) = \mathrm{stat}(\mathcal{Z}_{\hat{\imath}}^j)$. $\lhd$

Notice that in the description of $\mathscr{R}[\mathcal{Z}_{\hat{\imath}}^j \curlywedge \mathcal{Z}_{\hat{\imath}}^k]$ and in particular $\mathcal{P}^* = (m^*, \mathrm{stat}^*, \mathrm{card}^*)$ we do not use the notation from Definition 3.47. When we have from $\mathcal{P}$ the symbolic color classes $\mathscr{Z} = \{\mathscr{Z}_1, \dots, \mathscr{Z}_{\hat{\imath}-1}, \mathscr{Z}_{\hat{\imath}}, \mathscr{Z}_{\hat{\imath}+1}, \dots, \mathscr{Z}_n\}$, then we have from $\mathcal{P}^*$ the symbolic color classes $\mathscr{Z}^* = \{\mathscr{Z}_1, \dots, \mathscr{Z}_{\hat{\imath}-1}, \mathscr{Z}_{\hat{\imath}}^*, \mathscr{Z}_{\hat{\imath}+1}, \dots, \mathscr{Z}_n\}$, with $\mathscr{Z}_{\hat{\imath}}^* = (\mathscr{Z}_{\hat{\imath}} \setminus \{\mathcal{Z}_{\hat{\imath}}^j, \mathcal{Z}_{\hat{\imath}}^k\}) \cup \{\mathcal{Z}_{\hat{\imath}}^*\}$. The values of m^*, stat^*, card^* not specified in the lemma then are the same as of m, stat, card, but with $m^*(\hat{\imath}) = m(\hat{\imath}) - 1$, since two dynamic subclasses have been replaced by one. Regarding the successor function suc on the superscripts of dynamic subclasses in the case $\hat{\imath} > \mathbf{u}$, $*$ takes the place of j and suc skips over k.

We already informally discussed the following example above.

Example 3.50. Consider the trivial representation from Fig. 3.19. When merging the dynamic subclasses $\mathscr{Z}_2^2$ and $\mathscr{Z}_2^3$, we arrive at the dynamic representation from Fig. 3.18. Here, we renamed the new dynamic subclass $\mathscr{Z}_2^*$ to $\mathscr{Z}_2^2$, to adhere to the notation in Definition 3.47. $\lhd$

The next step is to define a criterion on the subclasses that we merge, guaranteeing that the new dynamic representation, resulting from the merging, represents the same symbolic decision set. A crucial part of such a criterion is the notion of *context* of a dynamic subclass:

Definition 3.51 (Context con**).** Let $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ be a representation of a symbolic decision set. The *context* $\mathrm{con}_{\mathscr{R}}(\mathscr{Z}_i^j)$ of a dynamic subclass $\mathscr{Z}_i^j$ is defined as the set of tuples in $\mathscr{D}$ where exactly one appearance of $\mathscr{Z}_i^j$ is replaced by an "place holder symbol" $\triangledown$. $\lhd$

We consider again our running example.

Example 3.52. For the representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ from Fig. 3.19, this definition yields

$$\mathrm{con}_{\mathscr{R}}(\mathscr{Z}_1^1) = \{\ \} \tag{3.1}$$

$$\begin{aligned}\mathrm{con}_{\mathscr{R}}(\mathscr{Z}_2^1) = \{\ & (\mathit{Tr}.\triangledown, \{\mathit{end}.\mathscr{Z}_2^1\}), & & (\mathit{Tr}.\mathscr{Z}_2^1, \{\mathit{end}.\triangledown\}), \\ & (\mathit{Fwd}.\triangledown, \{\mathit{end}.\mathscr{Z}_2^1\}), & & (\mathit{Fwd}.\mathscr{Z}_2^1, \{\mathit{end}.\triangledown\}), \\ & (\mathit{Rec}.(\mathscr{Z}_2^2, \triangledown), \{\mathit{end}.\mathscr{Z}_2^1\}), & & (\mathit{Rec}.(\mathscr{Z}_2^2, \mathscr{Z}_2^1), \{\mathit{end}.\triangledown\}) \\ & (\mathit{Rec}.(\mathscr{Z}_2^3, \triangledown), \{\mathit{end}.\mathscr{Z}_2^1\}), & & (\mathit{Rec}.(\mathscr{Z}_2^3, \mathscr{Z}_2^1), \{\mathit{end}.\triangledown\})\ \}\end{aligned} \tag{3.2}$$

$$\mathrm{con}_{\mathscr{R}}(\mathscr{Z}_2^2) = \{\ (\mathit{Rec}.(\triangledown, \mathscr{Z}_2^1), \{\mathit{end}.\mathscr{Z}_2^1\})\ \} \tag{3.3}$$

$$\mathrm{con}_{\mathscr{R}}(\mathscr{Z}_2^3) = \{\ (\mathit{Rec}.(\triangledown, \mathscr{Z}_2^1), \{\mathit{end}.\mathscr{Z}_2^1\})\ \}. \tag{3.4}$$

Since $\mathscr{Z}_1^1$ does not appear in any tuple in $\mathscr{D}$, we get in (3.1) that $\mathrm{con}_{\mathscr{R}}(\mathscr{Z}_1^1)$ is empty. The context $\mathrm{con}_{\mathscr{R}}(\mathscr{Z}_2^1)$ in (3.2) is the biggest set in this example. The reason for this is that $\mathscr{D}$ contains four elements, and $\mathscr{Z}_2^1$ occurs twice in each of these. This yields eight occurrences of $\mathscr{Z}_2^1$ in $\mathscr{D}$ and therefore eight elements in $\mathrm{con}_{\mathscr{R}}(\mathscr{Z}_2^1)$. Finally, in (3.3) and (3.4), the only occurrence of $\mathscr{Z}_2^2$ resp. $\mathscr{Z}_2^3$ in $\mathscr{D}$ has been replaced by $\triangledown$, yielding the same context for the two dynamic subclasses. $\lhd$

The following Lemma states that two dynamic subclasses with the same context (as $\mathscr{Z}_2^2$ and $\mathscr{Z}_2^3$ above) can be merged in a representation, while preserving the represented symbolic decision set.

Lemma 3.53. *Let $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ be a representation containing two dynamic subclasses $\mathscr{Z}_{\hat{\imath}}^j, \mathscr{Z}_{\hat{\imath}}^k$, $j \neq k$, satisfying*

$$\mathrm{con}_{\mathscr{R}}(\mathscr{Z}_{\hat{\imath}}^j) = \mathrm{con}_{\mathscr{R}}(\mathscr{Z}_{\hat{\imath}}^k)\ \wedge\ \mathrm{stat}(\mathscr{Z}_{\hat{\imath}}^j) = \mathrm{stat}(\mathscr{Z}_{\hat{\imath}}^k)\ \wedge\ (\hat{\imath} \leq \mathbf{u} \vee (\hat{\imath} > \mathbf{u} \wedge k = \mathrm{suc}(j))). \tag{3.5}$$

Then $\mathscr{R}$ and $\mathscr{R}[\mathscr{Z}_{\hat{\imath}}^j \curlywedge \mathscr{Z}_{\hat{\imath}}^k]$ represent the same symbolic decision set.

Before we give the proof, we inspect and explain the condition (3.5), and view the lemma in the context of Fig. 3.19 and Fig. 3.18. The purpose of condition (3.5) is to state that two respective dynamic subclasses can be merged without changing the represented symbolic decision set. We now see how this is ensured. The first part of the conjunction deals with the just defined context of the two dynamic subclasses under consideration. When the two are replaced by the new dynamic subclass, the latter has to take the role of *both*. In particular, it must be ensured that, when in the original representation there is more than one occurrence of these subclasses in the same tuple in the dynamic decision set, then it must contain a tuple with *every possible combination* of the subclasses.

Consider the following two (generic) examples of representations:

$$\mathscr{R}_1\ \boxed{\begin{array}{l} \mathrm{stat}(\mathcal{Z}_1^1) = \mathrm{stat}(\mathcal{Z}_1^2) \\ \mathscr{D}_1\ \boxed{\begin{array}{l} (p.(\mathcal{Z}_1^1, \mathcal{Z}_1^2), \dagger) \\ (p.(\mathcal{Z}_1^2, \mathcal{Z}_1^1), \dagger) \end{array}} \end{array}} \qquad \mathscr{R}_2\ \boxed{\begin{array}{l} \mathrm{stat}(\mathcal{Z}_1^1) = \mathrm{stat}(\mathcal{Z}_1^2) \\ \mathscr{D}_2\ \boxed{\begin{array}{l} (p.(\mathcal{Z}_1^1, \mathcal{Z}_1^2), \dagger) \\ (p.(\mathcal{Z}_1^2, \mathcal{Z}_1^1), \dagger) \\ (p.(\mathcal{Z}_1^1, \mathcal{Z}_1^1), \dagger) \\ (p.(\mathcal{Z}_1^2, \mathcal{Z}_1^2), \dagger) \end{array}} \end{array}}.$$

While one could naively think that in $\mathscr{R}_1$ the two dynamic subclasses $\mathcal{Z}_1^1$ and $\mathcal{Z}_1^2$ can be merged since that appear "symmetrically" in $\mathscr{D}_1$, this is not the case since we have

$$\begin{aligned} \mathrm{con}_{\mathscr{R}_1}(\mathcal{Z}_1^1) &= \{(p.(\nabla, \mathcal{Z}_1^2), \dagger), (p.(\mathcal{Z}_1^2, \nabla, \dagger))\} \\ &\neq \{(p.(\nabla, \mathcal{Z}_1^1), \dagger), (p.(\mathcal{Z}_1^1, \nabla, \dagger))\} = \mathrm{con}_{\mathscr{R}_1}(\mathcal{Z}_1^2). \end{aligned}$$

For the second representation $\mathscr{R}_2$, however, we have

$$\mathrm{con}_{\mathscr{R}_2}(\mathcal{Z}_1^1) = \{(p.(\nabla, \mathcal{Z}_1^1), \dagger), (p.(\mathcal{Z}_1^1, \nabla, \dagger)), (p.(\nabla, \mathcal{Z}_1^2), \dagger), (p.(\mathcal{Z}_1^2, \nabla, \dagger))\} = \mathrm{con}_{\mathscr{R}_2}(\mathcal{Z}_1^2).$$

This means in $\mathscr{R}_2$, the dynamic subclasses $\mathcal{Z}_1^1$ and $\mathcal{Z}_1^2$ satisfy (3.5). Merging the two subclasses (i.e., replacing them by a dynamic subclass $\mathcal{Z}_1^*$) would result in a dynamic decision set $\mathscr{D}^* = \{(p.(\mathcal{Z}_1^*, \mathcal{Z}_1^*), \dagger)\}$. The result would be the same if we merged the two subclasses from $\mathscr{R}_1$, where we are not allowed to do that. This negative example is in a similar spirit as the one on p. 104, where we explained that the decision set D^- is not represented by the dynamic representation in Fig. 3.18.

The second part of the conjunction (3.5) states that the two dynamic subclasses represent colors from the same static subclass $\mathscr{C}_{\boldsymbol{i},q}$ with $q = \mathrm{stat}(\mathcal{Z}_{\boldsymbol{i}}^j) = \mathrm{stat}(\mathcal{Z}_{\boldsymbol{i}}^k)$. This is necessary since every dynamic subclass represents colors from only one static subclass, so merging the two should result in a dynamic subclass also representing colors from $\mathscr{C}_{\boldsymbol{i},q}$. The third part of the conjunction ensures that, in the case of an ordered basic color class $\mathscr{C}_i$, we deal with two successive dynamic subclasses. This is important since for valid assignments, to every dynamic subclass, a set of successive colors is assigned.

Before we prove Lemma 3.53, we consider again the running example from Fig. 3.19.

Example 3.54. As we have already seen in (3.3) and (3.4), we have $\mathrm{con}_{\mathscr{R}}(\mathcal{Z}_2^2) = \mathrm{con}_{\mathscr{R}}(\mathcal{Z}_2^3)$. Since we have $\mathrm{stat}(\mathcal{Z}_2^2) = \mathrm{stat}(\mathcal{Z}_2^3) = 1$, and $\mathscr{C}_2$ is unordered (i.e., $2 \leq 2 = \mathbf{u}$), condition (3.5) is satisfied for these two subclasses and they can be merged. The result of this is the representation from Fig. 3.18 (after renaming the new subclass $\mathcal{Z}_2^*$ to $\mathcal{Z}_2^2$). $\lhd$

Having now established some intuition about condition the (3.5), we finally give the proof of Lemma 3.53.

Proof of Lemma 3.53. Let D be the symbolic decision set represented by $\mathscr{R}$ and $\mathsf{D} \in D$. Then there is a valid assignment $\eta = (\eta_i)_{i=1}^n$ with $\eta_i : \mathscr{C}_i \to \mathscr{Z}_i$ such that $\mathsf{D} = \eta^{-1}(\mathscr{D})$. From (3.5) it follows that the family of functions $\eta^* = (\eta_i^*)_{i=1}^n$ with $\eta_i' : \mathscr{C}_i \to \mathscr{Z}_i^*$, given by

$$\eta_i^* = \begin{cases} \eta_i & , i \neq \hat{\imath} \\ \{\mathcal{Z}_{\hat{\imath}}^j \mapsto \mathcal{Z}_{\hat{\imath}}^*, \mathcal{Z}_{\hat{\imath}}^k \mapsto \mathcal{Z}_{\hat{\imath}}^*\} \circ \eta_i & , i = \hat{\imath} \end{cases}$$

is a valid assignment, and we have $(\eta')^{-1}(\mathscr{D}') = \mathsf{D}$. Thus, we have that D is in the symbolic decision set represented by $\mathscr{R}^*$. The other direction works analogously. $\square$

A dynamic representation is called *minimal* if no two dynamic subclasses in it can be merged:

Definition 3.55 (Minimal representation). A representation $\mathscr{R}$ is called *minimal* if it does not contain any two dynamic subclasses $\mathcal{Z}_{\hat{\imath}}^j, \mathcal{Z}_{\hat{\imath}}^k$, $j \neq k$, satisfying (3.5). $\lhd$

Given a dynamic representation, it is algorithmically simple to construct a minimal representation of the same symbolic decision set by iteratively merging all pairs of dynamic subclasses satisfying (3.5). Still, minimality is not enough to obtain a unique canonical representation, since we can permute the indices j of the dynamic subclasses $\mathcal{Z}_i^j$.

Permutations on the dynamic subclasses are defined analogously to symmetries on basic color classes. This fits into the concept of dynamic subclasses replacing the colors.

Definition 3.56 (Permutations on dynamic partitions). For a given a dynamic partition $\mathcal{P}$, a *permutation* of the dynamic subclasses is a tuple $\hat{s} = (\hat{s}_1, \ldots, \hat{s}_n)$ of permutations $\hat{s}_i$ on $\mathscr{Z}_i = \{\mathcal{Z}_i^j \mid 1 \leq j \leq m(i)\}$ s.t. $\forall i, q : \hat{s}_i(\mathscr{Z}_{i,q}) = \mathscr{Z}_{i,q}$ (i.e., $\forall j : \mathrm{stat}(\hat{s}(\mathcal{Z}_i^j)) = \mathrm{stat}(\mathcal{Z}_i^j)$). In the case of an ordered class $\mathscr{C}_i$ we again only consider rotations $\hat{s}_i$ w.r.t. suc. We denote the set of all these permutations by $\hat{S}_{\mathcal{P}}$, where we omit the subscript when no confusion arises. $\lhd$

A permutation $\hat{s} \in \hat{S}_{\mathcal{P}}$ can be applied to the whole partition $\mathcal{P} = (m, \mathrm{stat}, \mathrm{card})$, yielding the dynamic partition $\hat{s}(\mathcal{P}) = (m, \mathrm{stat}, \mathrm{card} \circ \hat{s})$. This means that the number of dynamic subclasses of a color class does not change. By definition, the function stat is invariant under permutations. The only part that is different in $\mathcal{P}$ and $\hat{s}(\mathcal{P})$ is the cardinality function, since in $\hat{s}(\mathcal{P})$ a dynamic subclass $\mathcal{Z}_i^j$ now has the cardinality of $\hat{s}^{-1}(\mathcal{Z}_i^j)$ in $\mathcal{P}$.

Since in the trivial dynamic partition of a family of basic color classes we have $\forall \mathcal{Z}_i^j : \mathrm{card}(\mathcal{Z}_i^j) = 1$, we directly get the following property:

Property 3.57. *For the trivial dynamic partition $\mathcal{P}$, we have $\forall \hat{s} \in \hat{S}_{\mathcal{P}} : \hat{s}(\mathcal{P}) = \mathcal{P}$. The permutations $\hat{S}_{\mathcal{P}}$ are isomorphic to the symmetries S_G of the symmetric Petri game under consideration.*

A permutation $\hat{s} \in \hat{S}_{\mathcal{P}}$ can be applied to a representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$, yielding the dynamic representation $\hat{s}(\mathscr{R}) := (\hat{s}(\mathcal{P}), \hat{s}(\mathscr{D}))$, where $\hat{s}(\mathscr{D})$ is the result of replacing every occurrence of $\mathcal{Z}_i^j$ in $\mathscr{D}$ by $\hat{s}_i(\mathcal{Z}_i^j)$.

Example 3.58. Consider again the minimal representation from Fig. 3.18. In Fig. 3.20, we see the result of a permutation $\hat{s}$ (switching $\mathcal{Z}_2^1$ and $\mathcal{Z}_2^2$) applied to this representation. We see that m remains unaffected. Since the values of stat are trivial, they are again not depicted, but they also coincide. The cardinalities, however, are switched, as well as the occurences in $\mathscr{D}$. ◁

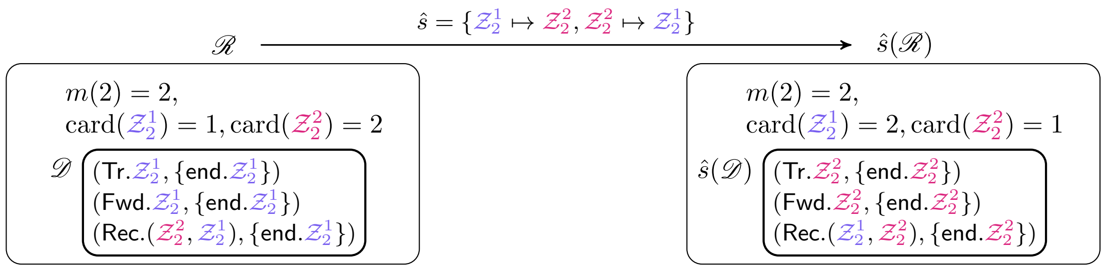

$\hat{s} = \{\mathcal{Z}_2^1 \mapsto \mathcal{Z}_2^2, \mathcal{Z}_2^2 \mapsto \mathcal{Z}_2^1\}$
$\mathscr{R}$
$\hat{s}(\mathscr{R})$
$m(2) = 2,$
$\text{card}(\mathcal{Z}_2^1) = 1, \text{card}(\mathcal{Z}_2^2) = 2$
$\mathscr{D}$
$(\mathsf{Tr}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Fwd}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Rec}.(\mathcal{Z}_2^2, \mathcal{Z}_2^1), \{\mathsf{end}.\mathcal{Z}_2^1\})$
$m(2) = 2,$
$\text{card}(\mathcal{Z}_2^1) = 2, \text{card}(\mathcal{Z}_2^2) = 1$
$\hat{s}(\mathscr{D})$
$(\mathsf{Tr}.\mathcal{Z}_2^2, \{\mathsf{end}.\mathcal{Z}_2^2\})$
$(\mathsf{Fwd}.\mathcal{Z}_2^2, \{\mathsf{end}.\mathcal{Z}_2^2\})$
$(\mathsf{Rec}.(\mathcal{Z}_2^1, \mathcal{Z}_2^2), \{\mathsf{end}.\mathcal{Z}_2^2\})$

Figure 3.20: The application of a permutation $\hat{s} \in \hat{S}_{\mathcal{P}}$ to the dynamic representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ from Fig. 3.18.

All trivial representations of the same symbolic decision set have the same (trivial) dynamic partition. Compared to the explicit decision sets in the represented equivalence class, each dynamic subclass represents exactly one color. Since all elements in a symbolic decision set are symmetric, we get from Property 3.57 the following corollary:

Corollary 3.59. *For any two trivial representations $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ and $\mathscr{R}' = (\mathcal{P}, \mathscr{D}')$ of the same symbolic decision set, there is a permutation $\hat{s} \in \hat{S}_{\mathcal{P}}$ such that $\hat{s}(\mathscr{R}) = \mathscr{R}'$.*

We now prove that minimal representations are unique up to permutation of the dynamic subclasses. This result is obtained using the observation that every minimal representation (as in Fig 3.18) can be reached from a trivial representation, i.e., containing only dynamic subclasses of cardinality 1 (as in Fig 3.19), by merging dynamic subclasses satisfying (3.5).

Lemma 3.60. *The minimal representations of a symbolic decision set are unique up to permutations of the dynamic subclasses.*

Proof. First, we observe that the minimal representation reached by merging dynamic subclasses is unique. Assume a dynamic representation $\mathscr{R}$ such that there are $\mathcal{Z}_{\boldsymbol{i}}^j, \mathcal{Z}_{\boldsymbol{i}}^k, \mathcal{Z}_{\boldsymbol{i}}^\ell$ with $j \neq k$, $k \neq \ell$ and $j \neq \ell$, such that $\mathcal{Z}_{\boldsymbol{i}}^j$ and $\mathcal{Z}_{\boldsymbol{i}}^k$ satisfy (3.5), and $\mathcal{Z}_{\boldsymbol{i}}^k$ and $\mathcal{Z}_{\boldsymbol{i}}^\ell$ satisfy

(3.5) as well. Then in the dynamic representation $\mathscr{R}^* = \mathscr{R}[\mathcal{Z}^j_{\hat{\boldsymbol{\imath}}} \curlywedge \mathcal{Z}^k_{\hat{\boldsymbol{\imath}}}]$ where $\mathcal{Z}^j_{\hat{\boldsymbol{\imath}}}$ and $\mathcal{Z}^k_{\hat{\boldsymbol{\imath}}}$ are replaced by $\mathcal{Z}^*_{\hat{\boldsymbol{\imath}}}$ as in Lemma 3.53, the new dynamic subclass $\mathcal{Z}^*_{\hat{\boldsymbol{\imath}}}$ and $\mathcal{Z}^\ell_{\hat{\boldsymbol{\imath}}}$ satisfy (3.5). This holds since $\mathrm{con}_{\mathscr{R}^*}(\mathcal{Z}^*_{\hat{\boldsymbol{\imath}}}) = \mathrm{con}_{\mathscr{R}}(\mathcal{Z}^k_{\hat{\boldsymbol{\imath}}})[\mathcal{Z}^j_{\hat{\boldsymbol{\imath}}} \leftarrow \mathcal{Z}^*_{\hat{\boldsymbol{\imath}}}, \mathcal{Z}^k_{\hat{\boldsymbol{\imath}}} \leftarrow \mathcal{Z}^*_{\hat{\boldsymbol{\imath}}}] = \mathrm{con}_{\mathscr{R}}(\mathcal{Z}^\ell_{\hat{\boldsymbol{\imath}}})[\mathcal{Z}^j_{\hat{\boldsymbol{\imath}}} \leftarrow \mathcal{Z}^*_{\hat{\boldsymbol{\imath}}}, \mathcal{Z}^k_{\hat{\boldsymbol{\imath}}} \leftarrow \mathcal{Z}^*_{\hat{\boldsymbol{\imath}}}] = \mathrm{con}_{\mathscr{R}^*}(\mathcal{Z}^\ell_{\hat{\boldsymbol{\imath}}})$. The dynamic subclasses $\mathcal{Z}^*_{\hat{\boldsymbol{\imath}}}$ and $\mathcal{Z}^\ell_{\hat{\boldsymbol{\imath}}}$ can therefore be merged again. This result shows that the order in which we merge dynamic subclasses does not matter: given a dynamic representation, the minimal representation reached by merging dynamic subclasses is unique.

Second, we observe that the existence of a permutation between two dynamic representations is preserved by merging dynamic subclasses. Let $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ and $\mathscr{R}' = (\mathcal{P}', \mathscr{D}')$ be two dynamic representations such that there exists a permutation $\hat{s} = (\hat{s}_i)_{i=1}^n \in \hat{S}_{\mathcal{P}}$ with $\hat{s}(\mathscr{R}) = \mathscr{R}'$. If there are $\mathcal{Z}^j_{\hat{\boldsymbol{\imath}}}$ and $\mathcal{Z}^k_{\hat{\boldsymbol{\imath}}}$ satisfying (3.5) in $\mathscr{R}$. Then $\hat{s}_{\hat{\boldsymbol{\imath}}}(\mathcal{Z}^j_{\hat{\boldsymbol{\imath}}})$ and $\hat{s}_{\hat{\boldsymbol{\imath}}}(\mathcal{Z}^k_{\hat{\boldsymbol{\imath}}})$ satisfy (3.5) in $\mathscr{R}'$. Let $\mathscr{R}^* = (\mathcal{P}^*, \mathscr{D}^*)$ (resp. $\mathscr{R}'^* = (\mathcal{P}'^*, \mathscr{D}'^*)$) be the two dynamic representations resulting from replacing $\mathcal{Z}^j_{\hat{\boldsymbol{\imath}}}$ and $\mathcal{Z}^k_{\hat{\boldsymbol{\imath}}}$ by $\mathcal{Z}^*_{\hat{\boldsymbol{\imath}}}$ (resp. replacing $\hat{s}_{\hat{\boldsymbol{\imath}}}(\mathcal{Z}^j_{\hat{\boldsymbol{\imath}}})$ and $\hat{s}_{\hat{\boldsymbol{\imath}}}(\mathcal{Z}^k_{\hat{\boldsymbol{\imath}}})$ by $\mathcal{Z}'^*_{\hat{\boldsymbol{\imath}}}$) as in Lemma 3.53. Then $\hat{s}^* = (\hat{s}^*_i)_{i=1}^n$ given by

$$\hat{s}^*_i = \begin{cases} \hat{s}_i & , i \neq \hat{\boldsymbol{\imath}} \\ (\hat{s}_i|_{\mathscr{X}_i \setminus \{\mathcal{Z}^j_{\hat{\boldsymbol{\imath}}}, \mathcal{Z}^k_{\hat{\boldsymbol{\imath}}}\}}) \cup \{\mathcal{Z}^*_{\hat{\boldsymbol{\imath}}} \mapsto \mathcal{Z}'^*_{\hat{\boldsymbol{\imath}}}\} & , i = \hat{\boldsymbol{\imath}} \end{cases}$$

is a permutation $\hat{s}^* \in \hat{S}_{\mathcal{P}^*}$ and $\hat{s}^*(\mathscr{R}^*) = \mathscr{R}'^*$.

From these two observations follows the statement of Lemma 3.60: Let $\mathscr{R}_{min}$ and $\mathscr{R}'_{min}$ be two minimal representations. Then there are two trivial representations $\mathscr{R}_{triv}$ and $\mathscr{R}'_{triv}$ such that $\mathscr{R}_{min}$ (resp. $\mathscr{R}'_{min}$) is the result of iteratively merging all pairs of dynamic subclasses (resp. $\mathscr{R}'_{triv}$) satisfying (3.5), starting from $\mathscr{R}_{triv}$. By Cor. 3.59, there is a permutation between $\mathscr{R}_{triv}$ and $\mathscr{R}'_{triv}$. This means, by the second observation, that after every merge of two subclasses in the process of reaching $\mathscr{R}_{min}$ from $\mathscr{R}_{triv}$, there is a permutation from the current representation to one that can be reached by merging subclasses starting from $\mathscr{R}'_{triv}$. Thus, there is also a permutation between the $\mathscr{R}_{min}$ and the respective representation $\widetilde{\mathscr{R}}'$ reached from $\mathscr{R}'_{triv}$. This representation must be minimal since else $\mathscr{R}_{min}$ would not be minimal. By the first observation, $\widetilde{\mathscr{R}}'$ must therefore be equal to $\mathscr{R}'_{min}$, which concludes the proof. □

Ordering

We finally get to the concept of *ordered* representations. This, together with minimality, will lead to canonical representations. We have seen above that

- it is easy to calculate a minimal representation of a given symbolic decision set,
- the minimal representations of a given symbolic decision set are unique up to a permutation of the dynamic subclasses.

Furthermore, from the definition of minimality it directly follows that applying a permutation to a minimal representation yields another minimal representation. This means, for *each* minimal representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$, the set of *all* minimal representations is given by $\{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}_{\mathcal{P}}\}$.

The goal of this Section 3.3.2 is to define for every symbolic decision set one *canonical* representation. Considering the set of of minimal representations will turn out to be the first step in this direction. In the following, we define what it means for a representation to be *ordered*. We then proceed and prove that for every symbolic decision set there is exactly one minimal and ordered representation. This representation then constitutes the canonical representation of this symbolic decision set.

For a given representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$, the question whether it is ordered is answered by an order on the set $\{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}_{\mathcal{P}}\}$. This order, in its turn uses a special presentation of the dynamic decision sets $\hat{s}(\mathscr{D})$ as matrices $\mathrm{mat}(\hat{s}(\mathscr{D}))$. To define, for a given dynamic representation, the matrix presentation of its dynamic decision set is therefore the first step to define what it means for a dynamic representation to be ordered.

The rows of the matrix $\mathrm{mat}(\mathscr{D})$ correspond to a set $\mathscr{Z}[\mathrm{pmax}]$, such that, for every place p in the symmetric Petri game under consideration, the symbolic color domain $\mathscr{Z}(p)$ can be embedded into $\mathscr{Z}[\mathrm{pmax}]$. On the other hand, every element in $\mathscr{Z}[\mathrm{pmax}]$ can be projected to an element in $\mathscr{Z}(p)$ by a function $\mathrm{proj}_p : \mathscr{Z}[\mathrm{pmax}] \to \mathscr{Z}(p)$. The formal definition proceeds as follows:

Given a dynamic partition $\mathcal{P}$, let $\mathscr{Z}[\mathrm{pmax}] := \mathscr{Z}_1^{\mathrm{pmax}(1)} \times \cdots \times \mathscr{Z}_n^{\mathrm{pmax}(n)}$, where $\forall 1 \leq i \leq n : \mathrm{pmax}(i) := \max\{J(p,i) \mid p \in P\}$. By this definition, we have that for all places p, the symbolic color domain $\mathscr{Z}(p) = \mathscr{Z}_1^{J(p,1)} \times \cdots \times \mathscr{Z}_n^{J(p,n)}$ can be embedded into $\mathscr{Z}[\mathrm{pmax}]$. For every place $p \in P$, let $\mathrm{proj}_p : \mathscr{Z}[\mathrm{pmax}] \to \mathscr{Z}(p)$ be the projection from $\mathscr{Z}[\mathrm{pmax}]$ to the symbolic color class of p, given by

$$\begin{aligned}\mathrm{proj}_p(&\mathcal{Z}_1^{(1)}, \dots, \mathcal{Z}_1^{(\mathrm{pmax}(1))}, \dots, \mathcal{Z}_n^{(1)}, \dots, \mathcal{Z}_n^{(\mathrm{pmax}(n))}) \\ &:= (\mathcal{Z}_1^{(1)}, \dots, \mathcal{Z}_1^{(J(p,1))}, \dots, \mathcal{Z}_n^{(1)}, \dots, \mathcal{Z}_n^{(J(p,n))}).\end{aligned}$$

Analogously, the columns of the matrix correspond to $\mathscr{Z}[\mathrm{tmax}] := \mathscr{Z}_1^{\mathrm{tmax}(1)} \times \cdots \times \mathscr{Z}_n^{\mathrm{tmax}(n)}$, where $\mathrm{tmax}(i) := \max\{J(t,i) \mid t \in T\}$. For every transition t, the symbolic color domain $\mathscr{Z}(t) = \mathscr{Z}_1^{J(t,1)} \times \cdots \times \mathscr{Z}_n^{J(t,n)}$ can be embedded into $\mathscr{Z}[\mathrm{tmax}]$, and for every transition $t \in T$, let $\mathrm{proj}_t : \mathscr{Z}[\mathrm{tmax}] \to \mathscr{Z}(t)$ be the projection from $\mathscr{Z}[\mathrm{tmax}]$ to the symbolic color class of t.

Example 3.61. Consider again the running example Signal Sending Satellites from Fig. 3.14, and the dynamic partition from the dynamic representation from Fig. 3.18, with $\mathscr{Z}_1 = \{\mathcal{Z}_1^1\}$ and $\mathscr{Z}_2 = \{\mathcal{Z}_2^1, \mathcal{Z}_2^2\}$.

The maximal number of occurrences of $\mathscr{C}_1$ in the type of a place in Fig. 3.14 is 1, at place *Env* with $J(\mathit{Env}, 1) = 1$. We therefore have $\mathrm{pmax}(1) = 1$ All other places p have $\mathscr{C}(p) = \mathscr{C}_2$, except for *Rec* with $\mathscr{C}(\mathit{Rec}) = \mathscr{C}_2 \times \mathscr{C}_2$, meaning $J(\mathit{Rec}, 2) = 2$. This leads to $\mathrm{pmax}(2) = 2$. We therefore have $\mathscr{Z}[\mathrm{pmax}] = \mathscr{Z}_1 \times \mathscr{Z}_2 \times \mathscr{Z}_2$.

Consider now the place *Fwd* with $\mathscr{C}(\mathit{Fwd}) = \mathscr{C}_2$ and therefore $\mathscr{Z}(\mathit{Fwd}) = \mathscr{Z}_2$. For the tuple $\mathcal{Z} = (\mathcal{Z}_1^1, \mathcal{Z}_2^2, \mathcal{Z}_2^1) \in \mathscr{Z}[\mathrm{pmax}]$ we have $\mathrm{proj}_{\mathit{Fwd}}(\mathcal{Z}) = (\mathcal{Z}_2^2)$, since $\mathcal{Z}_2^2$ is the first appearance of a dynamic subclass from $\mathscr{Z}_2$ in $\mathcal{Z}$. However, for place *Rec* with $\mathscr{C}(\mathit{Rec}) = \mathscr{C}_2 \times \mathscr{C}_2$ and therefore $\mathscr{Z}(\mathit{Rec}) = \mathscr{Z}_2 \times \mathscr{Z}_2$, we have $\mathrm{proj}_{\mathit{Rec}}(\mathcal{Z}) = (\mathcal{Z}_2^2, \mathcal{Z}_2^1)$

For the transitions, we have that $\mathscr{C}_1$ appears in no color domain, i.e., we have for all transitions t that $J(t, 1) = 0$, leading to $\mathrm{tmax}(1) = 0$. For $\mathscr{C}_2$ we have, however,

$\mathrm{tmax}(2) = J(\mathit{rec}, 2) = 2$. This leads to $\mathrm{tmax}(2) = 2$ and $\mathscr{Z}[\mathrm{tmax}] = \mathscr{Z}_2 \times \mathscr{Z}_2$ Consider the transition *end* with $\mathscr{C}(\mathit{end}) = \mathscr{C}_2$, leading to $\mathscr{C}(\mathit{end}) = \mathscr{Z}_2$, and the tuple $\mathcal{Z}' = (\mathcal{Z}_2^1, \mathcal{Z}_2^1)$. Then $\mathrm{proj}_{\mathit{end}}(\mathcal{Z}') = (\mathcal{Z}_2^1)$. $\lhd$

The elements in $\mathscr{Z}[\mathrm{pmax}]$ resp. $\mathscr{Z}[\mathrm{tmax}]$ are tuples of dynamic subclasses $\mathcal{Z}_i^j$. We therefore have a natural order over these sets, induced by the superscripts j. For a given dynamic representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$, the matrix $\mathrm{mat}(\mathscr{R})$ encodes the dynamic decision set $\mathscr{D}$. The rows of $\mathrm{mat}(\mathscr{R})$ correspond to the elements in $\mathscr{Z}[\mathrm{pmax}]$, and the rows to the elements in $\mathscr{Z}[\mathrm{tmax}]$, with the natural induced by the superscripts.

The elements in $\mathrm{mat}(\mathscr{R})$ are *lists* of tuples in $P \times (T \cup \{\dagger, \emptyset\})$. We have the lexicographic order on $P \times (T \cup \{\dagger, \emptyset\})$, given by the names of places and transitions, assuming $\dagger < \emptyset < t$ for all $t \in T$. Thus, we know what it means for such a list to be lexicographically ordered. With these discussions in mind we finally arrive at the definition of $\mathrm{mat}(\mathscr{R})$.

Definition 3.62 (Representation matrix mat**).** Let $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ be a dynamic representation. Then $\mathrm{mat}(\mathscr{R})$ is a 2-dimensional matrix, where the rows correspond to $\mathscr{Z}[\mathrm{pmax}]$ and the columns to $\mathscr{Z}[\mathrm{tmax}]$. Let $\mathcal{Z} \in \mathscr{Z}[\mathrm{pmax}]$ and $\mathcal{Z}' \in \mathscr{Z}[\mathrm{tmax}]$. Then the entry of $\mathrm{mat}(\mathscr{R})$ at position $(\mathcal{Z}, \mathcal{Z}')$ is the repetition free, lexicographically ordered list with elements in $P \times (T \cup \{\dagger, \emptyset\})$, given by

$$\begin{aligned}
(p, \dagger) \in \mathrm{mat}(\mathscr{R})(\mathcal{Z}, \mathcal{Z}') \quad &\Leftrightarrow \quad (p.\,\mathrm{proj}_p(\mathcal{Z}), \dagger) \in \mathscr{D} \\
(p, \emptyset) \in \mathrm{mat}(\mathscr{R})(\mathcal{Z}, \mathcal{Z}') \quad &\Leftrightarrow \quad (p.\,\mathrm{proj}_p(\mathcal{Z}), \emptyset) \in \mathscr{D} \\
(p, t) \in \mathrm{mat}(\mathscr{R})(\mathcal{Z}, \mathcal{Z}') \quad &\Leftrightarrow \exists \widehat{\mathcal{T}} \subseteq \mathscr{T} : (p.\,\mathrm{proj}_p(\mathcal{Z}), \widehat{\mathcal{T}}) \in \mathscr{D} \wedge t.\,\mathrm{proj}_t(\mathcal{Z}') \in \widehat{\mathcal{T}} \quad \lhd
\end{aligned}$$

We illustrate this definition by continuing the examples from above.

Example 3.63. We consider again the representation $\mathscr{R}$ from Fig. 3.18. Recall that the dynamic decision set of this representation is given by

$$\mathscr{D}\ \boxed{\begin{array}{l} (\mathsf{Tr}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\}) \\ (\mathsf{Fwd}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\}) \\ (\mathsf{Rec}.(\mathcal{Z}_2^2, \mathcal{Z}_2^1), \{\mathsf{end}.\mathcal{Z}_2^1\}) \end{array}}\ ,$$

with $\mathscr{Z}_1 = \{\mathcal{Z}_1^1\}$ and $\mathscr{Z}_2 = \{\mathcal{Z}_2^1, \mathcal{Z}_2^2\}$. We have already discussed in Example 3.61 that $\mathscr{Z}[\mathrm{pmax}] = \mathscr{Z}_1 \times \mathscr{Z}_2 \times \mathscr{Z}_2$ and $\mathscr{Z}[\mathrm{tmax}] = \mathscr{Z}_2 \times \mathscr{Z}_2$. In the matrix $\mathrm{mat}(\mathscr{R})$, we see the rows and columns corresponding to $\mathscr{Z}[\mathrm{pmax}]$ and $\mathscr{Z}[\mathrm{tmax}]$, respectively. They are ordered by the natural order on the superscripts of dynamic subclasses. From Def. 3.62 the matrix $\mathrm{mat}(\mathscr{R})$ is given by

$$\begin{array}{c|cccc|}
 & (\mathcal{Z}_2^1, \mathcal{Z}_2^1) & (\mathcal{Z}_2^1, \mathcal{Z}_2^2) & (\mathcal{Z}_2^2, \mathcal{Z}_2^1) & (\mathcal{Z}_2^2, \mathcal{Z}_2^2) \\
(\mathcal{Z}_1^1, \mathcal{Z}_2^1, \mathcal{Z}_2^1) & [(\mathit{Fwd}, \mathit{end}), (\mathit{Tr}, \mathit{end})] & [(\mathit{Fwd}, \mathit{end}), (\mathit{Tr}, \mathit{end})] & [\,] & [\,] \\
(\mathcal{Z}_1^1, \mathcal{Z}_2^1, \mathcal{Z}_2^2) & [(\mathit{Fwd}, \mathit{end}), (\mathit{Tr}, \mathit{end})] & [(\mathit{Fwd}, \mathit{end}), (\mathit{Tr}, \mathit{end})] & [\,] & [\,] \\
(\mathcal{Z}_1^1, \mathcal{Z}_2^2, \mathcal{Z}_2^1) & [(\mathit{Rec}, \mathit{end})] & [(\mathit{Rec}, \mathit{end})] & [\,] & [\,] \\
(\mathcal{Z}_1^1, \mathcal{Z}_2^2, \mathcal{Z}_2^2) & [\,] & [\,] & [\,] & [\,]
\end{array}\ .$$

We exemplarily discuss two entries in this matrix in detail. Consider the first entry, highlighted green, at position $\big((\mathcal{Z}_1^1, \mathcal{Z}_2^1, \mathcal{Z}_2^1), (\mathcal{Z}_2^1, \mathcal{Z}_2^1)\big)$. We have seen in Example 3.61 that $\mathrm{proj}_{Fwd}(\mathcal{Z}_1^1, \mathcal{Z}_2^1, \mathcal{Z}_2^1) = \mathcal{Z}_2^1$ and $\mathrm{proj}_{end}(\mathcal{Z}_2^1, \mathcal{Z}_2^1) = \mathcal{Z}_2^1$. Since we have that $end.\mathcal{Z}_2^1 \in \{end.\mathcal{Z}_2^1\}$ and $(Fwd.\mathcal{Z}_2^1, \{end.\mathcal{Z}_2^1\}) \in \mathscr{D}$, we get that (Fwd, end) is an entry in the list at the green position $\big((\mathcal{Z}_1^1, \mathcal{Z}_2^1, \mathcal{Z}_2^1), (\mathcal{Z}_2^1, \mathcal{Z}_2^1)\big)$. With the same arguments, we get that (Tr, end) is an element in the list at this position. The list at position $\big((\mathcal{Z}_1^1, \mathcal{Z}_2^1, \mathcal{Z}_2^1), (\mathcal{Z}_2^1, \mathcal{Z}_2^1)\big)$ is then given by $[(Fwd, end), (Tr, end)]$. The order inside the list results from the fact that Fwd is lexicographically smaller than Tr.

Consider now the entry at position $\big((\mathcal{Z}_1^1, \mathcal{Z}_2^2, \mathcal{Z}_2^1), (\mathcal{Z}_2^1, \mathcal{Z}_2^2)\big)$, highlighted red. In Example 3.61, we discussed $\mathrm{proj}_{Rec}(\mathcal{Z}_1^1, \mathcal{Z}_2^2, \mathcal{Z}_2^1) = (\mathcal{Z}_2^2, \mathcal{Z}_2^1)$, and we have $\mathrm{proj}_{end}(\mathcal{Z}_2^1, \mathcal{Z}_2^2) = \mathcal{Z}_2^1$. As for the green entry, we have $(Rec.(\mathcal{Z}_2^2, \mathcal{Z}_2^1), \{end.\mathcal{Z}_2^1\}) \in \mathscr{D}$, implying that (Rec, end) is in the list at position $\big((\mathcal{Z}_1^1, \mathcal{Z}_2^2, \mathcal{Z}_2^1), (\mathcal{Z}_2^1, \mathcal{Z}_2^2)\big)$. This is the only place transition combination for which this is true, leading to the red highlighted list containing a single element. $\lhd$

Notice that one entry in the dynamic decision set can lead to multiple appearances in the matrix representation. This is caused by the projection of multiple elements in $\mathscr{X}[\mathrm{pmax}]$ resp. $\mathscr{X}[\mathrm{tmax}]$ to the same element in $\mathscr{X}(p)$ resp. $\mathscr{X}(t)$ for a place p resp. transition t. When, for example, for a transition t, two tuples $\mathcal{Z}, \mathcal{Z}' \in \mathscr{X}[\mathrm{tmax}]$ are mapped to the same tuple in $\mathscr{X}(t)$, then every entry (p, t) in a list in column $\mathcal{Z}$ appears at the same place in column $\mathcal{Z}'$. An instance of that can be seen in the third row with the entry (Rec, end) in column $(\mathcal{Z}_2^1, \mathcal{Z}_2^1)$ as well as in column $(\mathcal{Z}_2^1, \mathcal{Z}_2^2)$. Here, we have $\mathrm{proj}_{end}(\mathcal{Z}_2^1, \mathcal{Z}_2^1) = \mathrm{proj}_{end}(\mathcal{Z}_2^1, \mathcal{Z}_2^2)$.

Recall that the motivation of defining this matrix $\mathrm{mat}(\mathscr{R})$ for a given representation $\mathscr{R}$ was to define an order on $\{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}\}$. When applying a permutation $\hat{s}$ to $\mathscr{R}$, the new corresponding matrix $\mathrm{mat}(\hat{s}(\mathscr{R}))$ can be calculated easily from the $\mathrm{mat}(\mathscr{R})$: in $\mathrm{mat}(\hat{s}(\mathscr{R}))$, the list in position $(\mathcal{Z}, \mathcal{Z}')$ is the list from position $(\hat{s}^{-1}(\mathcal{Z}), \hat{s}^{-1}(\mathcal{Z}'))$ in $\mathrm{mat}(\mathscr{R})$. In other words, the matrix $\mathrm{mat}(\hat{s}(\mathscr{R}))$ can be build from $\mathrm{mat}(\mathscr{R})$ by traversing $\mathrm{mat}(\mathscr{R})$ and writing the list at position $(\mathcal{Z}, \mathcal{Z}')$ into $\mathrm{mat}(\hat{s}(\mathscr{R}))$ at position $(\hat{s}(\mathcal{Z}), \hat{s}(\mathcal{Z}'))$. We demonstrate this on the example from above.

Example 3.64. Consider again Example 3.58, where we applied the permutation $\hat{s} = \{\mathcal{Z}_2^1 \mapsto \mathcal{Z}_2^2, \mathcal{Z}_2^2 \mapsto \mathcal{Z}_2^1\}$ to the representation $\mathscr{R}$, as we illustrated in Fig. 3.20. The matrix $\mathrm{mat}(\mathscr{R})$ is shown in Example 3.63 above. Traversing this matrix, and writing the list at every position $(\mathcal{Z}, \mathcal{Z}')$ into $\mathrm{mat}(\hat{s}(\mathscr{R}))$ at position $(\hat{s}(\mathcal{Z}), \hat{s}(\mathcal{Z}'))$ arrives at the following matrix $\mathrm{mat}(\hat{s}(\mathscr{R}))$:

$$\begin{array}{c|cccc|}
 & (\mathcal{Z}_2^1, \mathcal{Z}_2^1) & (\mathcal{Z}_2^1, \mathcal{Z}_2^2) & (\mathcal{Z}_2^2, \mathcal{Z}_2^1) & (\mathcal{Z}_2^2, \mathcal{Z}_2^2) \\
(\mathcal{Z}_1^1, \mathcal{Z}_2^1, \mathcal{Z}_2^1) & [\,] & [\,] & [\,] & [\,] \\
(\mathcal{Z}_1^1, \mathcal{Z}_2^1, \mathcal{Z}_2^2) & [\,] & [\,] & [(Rec, end)] & [(Rec, end)] \\
(\mathcal{Z}_1^1, \mathcal{Z}_2^2, \mathcal{Z}_2^1) & [\,] & [\,] & [(Fwd, end), (Tr, end)] & [(Fwd, end), (Tr, end)] \\
(\mathcal{Z}_1^1, \mathcal{Z}_2^2, \mathcal{Z}_2^2) & [\,] & [\,] & [(Fwd, end), (Tr, end)] & [(Fwd, end), (Tr, end)]
\end{array}.$$

We see, for example, that the green entry from position $\big((\mathcal{Z}_1^1, \mathcal{Z}_2^1, \mathcal{Z}_2^1), (\mathcal{Z}_2^1, \mathcal{Z}_2^1)\big)$ in $\mathrm{mat}(\mathscr{R})$ is now at position $\big((\mathcal{Z}_1^1, \mathcal{Z}_2^2, \mathcal{Z}_2^2), (\mathcal{Z}_2^2, \mathcal{Z}_2^2)\big)$, and the red entry from position

$\big((\mathcal{Z}_1^1, \mathcal{Z}_2^2, \mathcal{Z}_2^1), (\mathcal{Z}_2^1, \mathcal{Z}_2^2)\big)$ in $\text{mat}(\mathscr{R})$ can be found at position $\big((\mathcal{Z}_1^1, \mathcal{Z}_2^1, \mathcal{Z}_2^2), (\mathcal{Z}_2^2, \mathcal{Z}_2^1)\big)$ in $\text{mat}(\hat{s}(\mathscr{R}))$. $\lhd$

With this, we can finally define what it means for a representation to be ordered. The goal is to find a unique representation of a given symbolic decision set. By considering only minimal representations, Lemma 3.60 gives uniqueness up to permutations $\hat{s}$ of the dynamic subclasses. We now use the matrix $\text{mat}(\mathscr{R})$ of a representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ to determine a unique element in $\{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}_{\mathcal{P}}\}$, i.e., the set of all minimal representations. This element is found by defining an order $<_{\text{mat}}$ on $\{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}_{\mathcal{P}}\}$, which then will lead to a total order $<_{\mathscr{R}}$ on $\{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}_{\mathcal{P}}\}$. With this total order we can find for every symbolic decision set a unique minimal representation, which will be called *canonical*.

Order $<_{\text{mat}}$. We compare, for a representation $\mathscr{R}$, all matrices of representations resulting from applying a permutation $\hat{s}$ to $\mathscr{R}$. The comparison is made with respect to the lexicographic order. Here, we assume for two lists of different length that the longer list is bigger. Two lists of the same length are compared entry-wise. For two representations $\mathscr{R}_1, \mathscr{R}_2 \in \{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}_{\mathcal{P}}\}$, we say $\mathscr{R}_1 <_{\text{mat}} \mathscr{R}_2$ if $\text{mat}(\mathscr{R}_1)$ is lexicographically smaller than $\text{mat}(\mathscr{R}_2)$.

Order $<_{\text{card}}$. Let $\mathscr{R}_1, \mathscr{R}_2 \in \{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}_{\mathcal{P}}\}$. Then for $\mathscr{R}_1 = (m_1, \text{stat}_1, \text{card}_1, \mathscr{D}_1)$ and $\mathscr{R}_2 = (m_2, \text{stat}_2, \text{card}_2, \mathscr{D}_2)$, there is an $\hat{s} \in \hat{S}_{\mathcal{P}_1}$ such that $\hat{s}(\mathscr{R}_1) = \mathscr{R}_2$, and we know by the remark after Definition 3.56 that $m_1 = m_2$ and $\text{stat}_1 = \text{stat}_2$. The only difference in $\mathcal{P}_1 = (m_1, \text{stat}_1, \text{card}_1)$ and $\mathcal{P}_2 = (m_2, \text{stat}_2, \text{card}_2)$ is in the cardinality function. We can now compare $\mathcal{P}_1$ and $\mathcal{P}_2$ w.r.t. the cardinality by iterating through all $\mathcal{Z}_i^j$ and look for the first difference. If the first difference is in $\mathcal{Z}_{\boldsymbol{\imath}}^{\boldsymbol{\jmath}}$, and $\text{card}_1(\mathcal{Z}_{\boldsymbol{\imath}}^{\boldsymbol{\jmath}}) < \text{card}_2(\hat{s}(\mathcal{Z}_{\boldsymbol{\imath}}^{\boldsymbol{\jmath}})))$, we say $\mathscr{R}_1 <_{\text{card}} \mathscr{R}_2$. Formally, $\mathscr{R}_1 <_{\text{card}} \mathscr{R}_2$ iff

$$\begin{aligned}
\exists \boldsymbol{\imath} \in I : \quad & (\forall i < \boldsymbol{\imath}\ \forall 1 \le j \le m(i) : \text{card}_1(\mathcal{Z}_i^j) = \text{card}_2(\hat{s}(\mathcal{Z}_i^j)) \\
\wedge \quad & \exists 1 \le \boldsymbol{\jmath} \le m(\boldsymbol{\imath}) : (\forall 1 \le j \le \boldsymbol{\jmath} : \text{card}_1(\mathcal{Z}_{\boldsymbol{\imath}}^j) = \text{card}_2(\hat{s}(\mathcal{Z}_{\boldsymbol{\imath}}^j))) \\
& \qquad \wedge \text{card}_1(\mathcal{Z}_{\boldsymbol{\imath}}^{\boldsymbol{\jmath}}) < \text{card}_2(\hat{s}(\mathcal{Z}_{\boldsymbol{\imath}}^{\boldsymbol{\jmath}}))).
\end{aligned}$$

Order $<_{\mathscr{R}}$. Let $\mathscr{R}_1, \mathscr{R}_2 \in \{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}_{\mathcal{P}}\}$. We say $\mathscr{R}_1 <_{\mathscr{R}} \mathscr{R}_2$ if

$$\begin{aligned}
& \mathscr{R}_1 >_{\text{mat}} \mathscr{R}_2 \\
\vee \quad & (\text{mat}(\mathscr{R}_1) = \text{mat}(\mathscr{R}_2) \wedge \mathscr{R}_1 >_{\text{card}} \mathscr{R}_2)
\end{aligned}$$

At first glance unintuitively, we thus have $\mathscr{R}_1 <_{\mathscr{R}} \mathscr{R}_2$ if $\mathscr{R}_1 >_{\text{mat}} \mathscr{R}_2$. The reason for this is that we want to call a representation ordered iff it is *minimal* w.r.t. $<_{\mathscr{R}}$. The term $\mathscr{R}_1 >_{\text{mat}} \mathscr{R}_2$ means that the dynamic subclasses $\mathcal{Z}_i^j$ with a smaller superscript j have bigger lists in the matrix. This, in its turn, describes that these subclasses appear more often in the dynamic decision set. When the matrices of two minimal representations are equal, the one smaller w.r.t. $<_{\mathscr{R}}$ is the one in which the dynamic subclasses with smaller superscript have a bigger cardinality.

By this, we have defined for every representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$, an order $<_{\mathscr{R}}$ on the set $\{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}_{\mathcal{P}}\}$.

Definition 3.65 (Ordered representation). A representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ is called *ordered* if $\mathscr{R}$ is minimal w.r.t. $<_{\mathscr{R}}$ in the set $\{\hat{s}(\mathscr{R}) \mid \hat{s} \in \hat{S}_{\mathcal{P}}\}$. $\lhd$

Example 3.66. In our running example $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ from Fig. 3.18, we have $\hat{S} = \{id, \hat{s}\}$ with $id = \{\mathcal{Z}_2^1 \mapsto \mathcal{Z}_2^1, \mathcal{Z}_2^2 \mapsto \mathcal{Z}_2^2\}$ and $\hat{s} = \{\mathcal{Z}_2^1 \mapsto \mathcal{Z}_2^2, \mathcal{Z}_2^2 \mapsto \mathcal{Z}_2^1\}$. We saw the matrix $\text{mat}(\mathscr{R}) = \text{mat}(id(\mathscr{R}))$ in Example 3.63, and the matrix $\text{mat}(\hat{s}(\mathscr{R}))$ in Example 3.64. We directly see that $\text{mat}(\mathscr{R})$ is bigger, since the first entry is lexicographically bigger (the empty list is the smallest element). This means the representation $\mathscr{R}$ from Fig. 3.18 is a minimal and ordered representation of the represented symbolic decision set. We will see in a moment that such a representation is unique for every symbolic decision set, making it *canonical.* $\lhd$

We have defined what it means for a dynamic representation to be minimal and to be ordered. To the aim of showing that this leads to canonical representations of symbolic decision sets, we formulate the following lemma, making a statement of two ordered dynamic decision sets that can be reached from each other by a permutation.

Lemma 3.67. *Let $\mathscr{R}_1 = (\mathcal{P}_1, \mathscr{D}_1)$ and $\mathscr{R}_2 = (\mathcal{P}_2, \mathscr{D}_2)$ be two ordered representations of the same symbolic decision set, and $\hat{s} \in \mathcal{P}_1$ be a permutation such that $\mathscr{R}_2 = \hat{s}(\mathscr{R}_1)$. Then $\mathscr{D}_1 = \mathscr{D}_2$.*

Proof. Since $\mathscr{R}_1$ is ordered, $\hat{s}$ cannot transform $\text{mat}(\mathscr{R}_1)$ into a (lexicographically) bigger matrix. Aiming a contradiction, assume that $\hat{s}$ transforms $\text{mat}(\mathscr{R}_1)$ into a smaller matrix $\text{mat}(\hat{s}(\mathscr{R}))$. Since $\text{mat}(\hat{s}(\mathscr{R}_1)) = \text{mat}(\mathscr{R}_2)$, this would mean that $\hat{s}^{-1}$ transforms $\text{mat}(\mathscr{R}_2)$ into a bigger matrix, implying that $\mathscr{R}_2$ is not ordered. Contradiction. Therefore, $\text{mat}(\mathscr{R}_1) = \text{mat}(\hat{s}(\mathscr{R}_1)) = \text{mat}(\mathscr{R}_2)$. Since $\text{mat}(\mathscr{R}_i)$ is just a presentation of $\mathscr{D}_i$, we follow $\mathscr{D}_1 = \mathscr{D}_2$. $\square$

Now we are equipped with all tools to prove that there is exactly one minimal and ordered representation for each symbolic decision set. We call this representation the *canonical* (dynamic) representation.

Theorem 3.68. *For every symbolic decision set there is exactly one minimal and ordered dynamic representation.*

Proof. Let $\mathscr{R}_1 = (\mathcal{P}_1, \mathscr{D}_1)$ and $\mathscr{R}_2 = (\mathcal{P}_2, \mathscr{D}_2)$ be two minimal and ordered representations of the same symbolic decision set. Lemma 3.60 gives us that there is a permutation $\hat{s} \in \hat{S}_{\mathcal{P}_1}$ such that $\mathscr{R}_2 = \hat{s}(\mathscr{R}_1)$. Then we have by Lemma 3.67 that $\mathscr{D}_1 = \mathscr{D}_2$. We therefore have $\text{mat}(\mathscr{D}_1) = \text{mat}(\mathscr{D}_2)$. Assume now $\mathcal{P}_1 \neq \mathcal{P}_2$. Then w.l.o.g. $\mathcal{P}_1 <_{\text{card}} \mathcal{P}_2$, meaning $\mathscr{R}_1 <_{\mathscr{R}_1} \mathscr{R}_2$, implying that $\mathscr{R}_1$ is not ordered. Contradiction. We therefore have $\mathcal{P}_1 = \mathcal{P}_2$ and thereby $\mathscr{R}_1 = \mathscr{R}_2$. $\square$

Definition 3.69 (Canonical representation). The *canonical representation* of a symbolic decision set is its unique ordered and minimal dynamic representation. $\lhd$

We can algorithmically order a minimal representation by calculating all symmetric representations and finding the one with the lexicographically biggest matrix. If there is more than one such representations then we compare the cardinality functions in the dynamic partition.

Let now a symbolic decision set D be given. We then can construct a minimal representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ of D (by first constructing a trivial representation and then merging subclasses until it is not possible anymore) in $O(1)$ time. We can then order this minimal representation in $O(|\hat{S}_{\mathcal{P}}|)$ time, since we must find the smallest (w.r.t. $<_{\mathscr{R}}$) of $|\hat{S}_{\mathcal{P}}|$ many matrices. Since we have for all partitions $\mathcal{P}$ that $|\hat{S}_{\mathcal{P}}| \leq |S_G|$ for the given high-level Petri game G, we conclude that we get the following Corollary.

Corollary 3.70. *For a given symbolic decision set, we can construct the canonical representation in $O(|S_G|)$.*

We therefore have a fixed cost for generating canonical representations. However, due to minimality of the constructed representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$, we often have $|\hat{S}_{\mathcal{P}}| \leq |S_G|$ since there are less dynamic subclasses than colors. We later go more into detail with comparing the cost of building the symbolic two-player game with arbitrary representatives vs. canonical representations.

3.3.3 Relations between Canonical Representations

In the preceding section, we presented the process of constructing the canonical representation for a given symbolic decision set. These canonical representations are set to substitute the explicit symbolic decision sets as vertices in the symbolic two-player game. However, the edges in the symbolic two-player game are defined based on relations between symbolic decision sets. Consequently, in this section we define corresponding relations between the canonical representations in order to accurately define the so-called canonical two-player game, with canonical representations as vertices.

In [CDFH91a; CDFH93], the authors define the symbolic firing of transitions between canonical representations of equivalence classes of *markings*. This leads to a symbolic reachability graph with canonical representations as vertices. In [GW21a], we lifted these results to decision sets, defining symbolic transition firing and symbolic †-resolution between canonical representations of symbolic decision sets. With this, we defined the canonical two-player game.

The concept revolves around adapting the guards and arc expressions surrounding a transition to handle dynamic subclasses rather than individual colors. Additionally, for each transition, a set of "symbolic modes" is defined. In this context, a mode encompasses not only a tuple of dynamic subclasses but also accounts for distinct instances of the same dynamic subclass. To fire a transition in a symbolic mode, a canonical representation must first be "split" into a finer representation, depending on the mode. Following this, a firing takes place, after which the new canonical representation is constructed.

However, this approach to defining symbolic relations presents two drawbacks concerning this thesis. Firstly, the definitions involve a substantial amount of notation. While this on its own is not a significant drawback, they also lack a novel concept. In essence, the outcome aligns with the symbolic relations detailed in Sec. 3.2.2. Therefore, we choose an alternative method for defining the symbolic relations instead of the one in [CDFH91a] and [GW21a]. We revert to the definitions of relations between (arbitrary representatives) of symbolic decision sets from Def. 3.20. This ensures that we maintain the focus on the core concept of this section, namely, the definition and construction of canonical representations. Introducing new notation, without a corresponding increase in clarity regarding the used concepts, is thus avoided.

Recall that in Def. 3.20, we defined the relations $\overline{\mathsf{D}}[\![t.\alpha_{\overline{\mathsf{D}}}(\sigma)\rangle\!\rangle\overline{\mathsf{D}'}$ and $\overline{\mathsf{D}}[\![\dagger\rangle\!\rangle\overline{\mathsf{D}'}$ between the (arbitrary) representatives $\overline{\mathsf{D}}$ and $\overline{\mathsf{D}'}$ of symbolic decision sets $[\mathsf{D}]$ and $[\mathsf{D}']$. The definition resulted from the ordinary relations $\overline{\mathsf{D}}[t.\alpha_{\overline{\mathsf{D}}}(\sigma)\rangle\mathsf{D}'$ and $\overline{\mathsf{D}}[\dagger\rangle\mathsf{D}'$ of transition firing and $\dagger$-resolution. The idea for relations between canonical representations $\mathscr{R}$ and $\mathscr{R}'$ is to define a *canonical representatives* $\mathsf{D}(\mathscr{R})$. Then, the ordinary relations $\mathsf{D}(\mathscr{R})[t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)\rangle\mathsf{D}'$ and $\mathsf{D}(\mathscr{R})[\dagger\rangle\mathsf{D}'$ of transition firing and $\dagger$-resolution will imply the symbolic relations $\mathscr{R}[\![t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)\rangle\!\rangle\mathscr{R}'$ and $\mathscr{R}[\![\dagger\rangle\!\rangle\mathscr{R}'$, just as for symbolic decision sets.

Thus, before we define the symbolic relations, we have to define for a canonical representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ of a symbolic decision set D, the canonical representative $\mathsf{D}(\mathscr{R}) \in D$. The decision set $\mathsf{D}(\mathscr{R})$ is obtained by defining a *canonical valid assignment* $\eta_{\mathscr{R}}$ which defines $\mathsf{D}(\mathscr{R}) = \eta_{\mathscr{R}}^{-1}(\mathscr{D})$. This assignment in turn is obtained by traversing, for a fixed order of the colors, both the set of colors and dynamic subclasses simultaneously. Before we formally define this procedure, we demonstrate it on our running example Signal Sending Satellites.

Example 3.71 (Canonical Representative). We consider again the example from Fig. 3.18, with the symbolic decision set D containing the three elements $\mathsf{D}', \mathsf{D}'', \mathsf{D}'''$, and its representation $\mathscr{R}$. They are depicted again in Fig. 3.21.

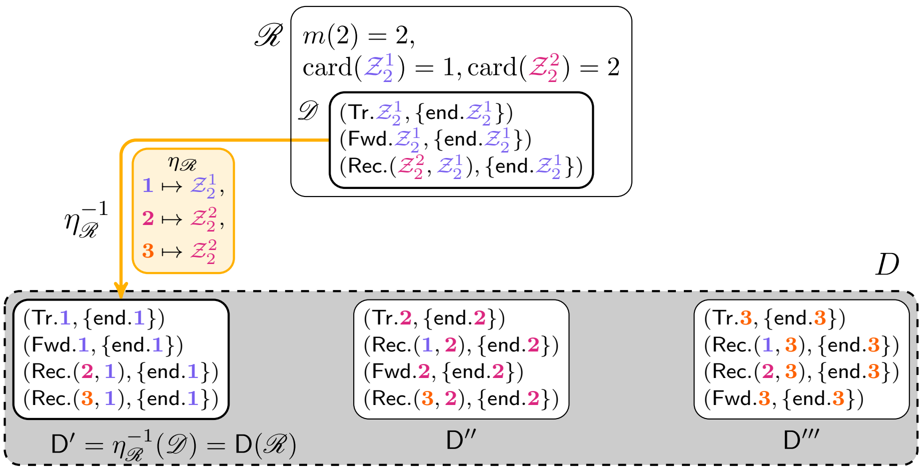

$\mathscr{R}$
$m(2) = 2$,
$\mathrm{card}(\mathcal{Z}_2^1) = 1, \mathrm{card}(\mathcal{Z}_2^2) = 2$
$\mathscr{D}$
$(\mathsf{Tr}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Fwd}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Rec}.(\mathcal{Z}_2^2, \mathcal{Z}_2^1), \{\mathsf{end}.\mathcal{Z}_2^1\})$
$\eta_{\mathscr{R}}$
$\mathbf{1} \mapsto \mathcal{Z}_2^1$,
$\mathbf{2} \mapsto \mathcal{Z}_2^2$,
$\mathbf{3} \mapsto \mathcal{Z}_2^2$
$\eta_{\mathscr{R}}^{-1}$
D
$(\mathsf{Tr}.\mathbf{1}, \{\mathsf{end}.\mathbf{1}\})$
$(\mathsf{Fwd}.\mathbf{1}, \{\mathsf{end}.\mathbf{1}\})$
$(\mathsf{Rec}.(\mathbf{2}, \mathbf{1}), \{\mathsf{end}.\mathbf{1}\})$
$(\mathsf{Rec}.(\mathbf{3}, \mathbf{1}), \{\mathsf{end}.\mathbf{1}\})$
$(\mathsf{Tr}.\mathbf{2}, \{\mathsf{end}.\mathbf{2}\})$
$(\mathsf{Rec}.(\mathbf{1}, \mathbf{2}), \{\mathsf{end}.\mathbf{2}\})$
$(\mathsf{Fwd}.\mathbf{2}, \{\mathsf{end}.\mathbf{2}\})$
$(\mathsf{Rec}.(\mathbf{3}, \mathbf{2}), \{\mathsf{end}.\mathbf{2}\})$
$(\mathsf{Tr}.\mathbf{3}, \{\mathsf{end}.\mathbf{3}\})$
$(\mathsf{Rec}.(\mathbf{1}, \mathbf{3}), \{\mathsf{end}.\mathbf{3}\})$
$(\mathsf{Rec}.(\mathbf{2}, \mathbf{3}), \{\mathsf{end}.\mathbf{3}\})$
$(\mathsf{Fwd}.\mathbf{3}, \{\mathsf{end}.\mathbf{3}\})$
$\mathsf{D}' = \eta_{\mathscr{R}}^{-1}(\mathscr{D}) = \mathsf{D}(\mathscr{R})$
D''
D'''

Figure 3.21: The canonical valid assignment and the canonical representative of a canonical representation.

Recall that the partition describes two dynamic subclasses $\mathcal{Z}_2^1$ and $\mathcal{Z}_2^2$, respectively representing $\mathrm{card}(\mathcal{Z}_2^1) = 1$ and $\mathrm{card}(\mathcal{Z}_2^2) = 2$ colors from $\mathscr{C}_2 = \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$, such that $\mathscr{Z}_2 = \{\mathcal{Z}_2^1, \mathcal{Z}_2^2\}$. Since $\mathscr{C}_1 = \{\mathbf{0}\}$ contains only one color, we trivially have $\mathscr{Z}_1 = \{\mathcal{Z}_1^1\}$ with $\mathrm{card}(\mathcal{Z}_1^1)$ in every representation.

We now construct the canonical valid assignment $\eta_{\mathscr{R}} = (\eta_1, \eta_2)$, where $\eta_i : \mathscr{C}_i \to \mathscr{Z}_i$. The idea of the canonical valid assignment is to define, for the symmetric Petri game in question, a fixed order on the colors of each basic color class. We then iterate, for every i, over $\mathscr{C}_i$ and $\mathscr{Z}_i$ in parallel, mapping the first $\mathrm{card}(\mathcal{Z}_i^1)$ colors in $\mathscr{C}_i$ to $\mathcal{Z}_i^1$, the next $\mathrm{card}(\mathcal{Z}_i^2)$ colors in $\mathscr{C}_i$ to $\mathcal{Z}_i^2$, and so on.

We trivially have $\eta_1 = [\mathbf{0} \mapsto \mathcal{Z}_1^1]$, so we demonstrate this procedure on η_2. We have an obvious order on $\mathscr{C}_2$, namely $\mathbf{1}, \mathbf{2}, \mathbf{3}$. Iterating over this list, η_2 now maps the first $\mathrm{card}(\mathcal{Z}_2^1) = 1$ colors (i.e., $\mathbf{1}$) to $\mathcal{Z}_2^1$, and the next $\mathrm{card}(\mathcal{Z}_2^2) = 2$ colors (i.e., $\mathbf{2}$ and $\mathbf{3}$) to $\mathcal{Z}_2^2$. We therefore have $\eta_2 = [\mathbf{1} \mapsto \mathcal{Z}_2^1, \mathbf{2} \mapsto \mathcal{Z}_2^2, \mathbf{3} \mapsto \mathcal{Z}_2^2]$. With $\eta_{\mathscr{R}} = (\eta_1, \eta_2)$ we finally get $\mathsf{D}(\mathscr{R}) = \eta_{\mathscr{R}}^{-1}(D) = \mathsf{D}'$, as indicated in the figure. $\lhd$

In our running example, we have a relatively simple scenario with two basic color classes that are neither ordered nor partitioned into static subclasses. The formal definition of a canonical valid assignment and canonical representative of course has to consider these possibilities for a basic color class $\mathscr{C}_i$. Formalizing the intuitive idea of fixing an order of the basic colors and iteratively assigning them to dynamic subclasses with increasing superscript, thus involves some more notation:

Definition 3.72 (Canonical valid assignment). We define, for a given symmetric Petri game G with basic color classes $\mathscr{C}_1, \dots, \mathscr{C}_n$, and a canonical representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ of a symbolic decision set in G, with $\mathcal{P}$ describing the symbolic color classes $\mathscr{Z}_i = \{\mathcal{Z}_i^j \mid 1 \leq j \leq m(i)\}$, for every i the assignment $\eta_i : \mathscr{C}_i \to \mathscr{Z}_i$ from $\eta_{\mathscr{R}} := (\eta_i)_{i=1}^n$. We distinguish between three cases of the structure of $\mathscr{C}_i$. In each case, we fix a renaming of the colors in $\mathscr{C}_i$ for the game, independently of the representation.

- If $\mathscr{C}_i$ is *unordered* (i.e., $i \leq \mathbf{u}$), and possibly *partitioned* into static subclasses (i.e., $\mathscr{C}_i = \bigsqcup_{q=1}^{n_i} \mathscr{C}_{i,q}$) then we rename the colors such that for all q we have $\mathscr{C}_{i,q} = \{c_{i,q}^j \mid 1 \leq j \leq |\mathscr{C}_{i,q}|\}$. For each $\mathscr{Z}_{i,q}$, we now denote its elements $\mathscr{Z}_{i,q} = \{\mathcal{Z}_{i,q}^j \mid 1 \leq j \leq |\mathscr{Z}_{i,q}|\}$, where j increases with the superscripts of their original names above. η_i is then defined to map the first $\mathrm{card}(\mathcal{Z}_{i,q}^1)$ colors in $\mathscr{C}_{i,q}$ to $\mathcal{Z}_{i,q}^1$, the next $\mathrm{card}(\mathcal{Z}_{i,q}^2)$ colors in $\mathscr{C}_{i,q}$ to $\mathcal{Z}_{i,q}^2$, and so on.
- If $\mathscr{C}_i$ is *ordered* (i.e., $i > \mathbf{u}$), and *not* partitioned into static subclasses (i.e., $\mathscr{C}_i = \mathscr{C}_{i,1}$) then we rename the colors such that we have $\mathscr{C}_i = \{c_i^j \mid 1 \leq j \leq |\mathscr{C}_i|\}$, where we assure that $\mathrm{suc}(c_i^j) = c_i^{\mathrm{suc}(j)}$, where $\mathrm{suc}(j) = (j \mod |\mathscr{C}_i|) + 1$ η_i is then defined to map the first $\mathrm{card}(\mathcal{Z}_i^1)$ colors in $\mathscr{C}_i$ to $\mathcal{Z}_i^1$, the next $\mathrm{card}(\mathcal{Z}_i^2)$ colors in $\mathscr{C}_i$ to $\mathcal{Z}_i^2$, and so on.
- If $\mathscr{C}_i$ is *ordered* (i.e., $i > \mathbf{u}$), *and* partitioned into static subclasses (i.e., $\mathscr{C}_i = \bigsqcup_{q=1}^{n_i} \mathscr{C}_{i,q}$) then we rename the colors as in the second case. Since a valid assignment must be "compatible" with suc (cp. Def. 3.44) there are only $|\mathscr{C}_i|$ possibilities for a corresponding mapping $\eta_i : \mathscr{C}_i \to \mathscr{Z}_i$. All these are instances of $\widetilde{\eta}_i \circ s^k$, with $\widetilde{\eta}_i$ the mapping from the second case, $s = \{c_i^j \mapsto c_i^{\mathrm{suc}(j)} \mid 1 \leq j \leq |\mathscr{C}_i|\}$, and $0 \leq k \leq |\mathscr{C}_i|$. We define η_i to be the instance with the minimal k such that for all $1 \leq j \leq |\mathscr{C}_i|$ we have $c_i^j \in \mathscr{C}_{i,q}$ with $q = \mathrm{stat}(\eta_i(c_i^j))$. $\lhd$

With this formal definition following a simple idea for a canonical valid assignment $\eta_{\mathscr{R}}$, we can now easily define the canonical representative $\mathsf{D}(\mathscr{R})$ of a representation $\mathscr{R}$:

Definition 3.73 (Canonical representative). Let $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ be a canonical representation of a symbolic decision set D. Then $\mathsf{D}(\mathscr{R}) = \eta_{\mathscr{R}}^{-1}(\mathscr{D}) \in D$ is the *canonical representative* of $\mathscr{R}$. $\triangleleft$

We have already discussed an example for this in Example 3.71 above. We can now define the symbolic relations $\mathscr{R}[\![t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)\rangle\!\rangle\mathscr{R}'$ and $\mathscr{R}[\![\dagger\rangle\!\rangle\mathscr{R}'$ based on the relations $\mathsf{D}(\mathscr{R})[t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)\rangle\mathsf{D}'$ and $\mathsf{D}(\mathscr{R})[\dagger\rangle\mathsf{D}'$ just as in Def. 3.20.

Recall from Sec. 3.2.2 that the so-called isotropy group $S_{\mathsf{D}} = \{s \in S \mid s(\mathsf{D}) = \mathsf{D}\}$ of a decision set D is the group of all admissible symmetries that preserve D. For a transition $t \in T$, we denoted by $\Sigma(t)_{\mathsf{D}} = \Sigma(t)/S_{\mathsf{D}}$ the set containing the equivalence classes of all modes of t, with respect to the isotropy group S_{D}. For each class in $\Sigma(t)_{\mathsf{D}}$ we arbitrarily chose a representative mode $\overline{\sigma}$, and defined α_{D} as the function mapping each $\sigma \in \Sigma(t)$ to its representative $\alpha_{\mathsf{D}}(\sigma)$.

Definition 3.74 (Symbolic relations between canonical representations). We say a transition t can *fire symbolically* from a canonical representation $\mathscr{R}$ in mode $\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)$ representing σ in $\Sigma(t)_{\mathsf{D}(\mathscr{R})}$, denoted by $\mathscr{R}[\![t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)\rangle\!\rangle$, iff $\mathsf{D}(\mathscr{R})[t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)\rangle$. The canonical representation $\mathscr{R}'$ obtained after the symbolic firing is determined as follows:

$$\mathscr{R}[\![t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)\rangle\!\rangle\mathscr{R}' \Leftrightarrow \exists \mathsf{D}' \in [\mathsf{D}(\mathscr{R}')] : \mathsf{D}(\mathscr{R})[t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)\rangle\mathsf{D}'.$$

We say a $\dagger$ can be *symbolically resolved* in a canonical representation $\mathscr{R}$, denoted by $\mathscr{R}[\![\dagger\rangle\!\rangle$, iff $\mathsf{D}(\mathscr{R})[\dagger\rangle$. The canonical representations $\mathscr{R}'$ sets obtained after the symbolic $\dagger$-resolution are canonical representations of the decision sets D' satisfying $\mathsf{D}(\mathscr{R})[\dagger\rangle\mathsf{D}'$:

$$\mathscr{R}[\![\dagger\rangle\!\rangle\mathscr{R}' \Leftrightarrow \exists \mathsf{D}'' \in [\mathsf{D}(\mathscr{R}')] : \mathsf{D}(\mathscr{R})[\dagger\rangle\mathsf{D}''.$$ $\triangleleft$

Before we demonstrate the symbolic relations on an example, we illustrate them in Fig. 3.22. In the top left, we see a canonical representation $\mathscr{R}$ with dynamic partition $\mathcal{P}$

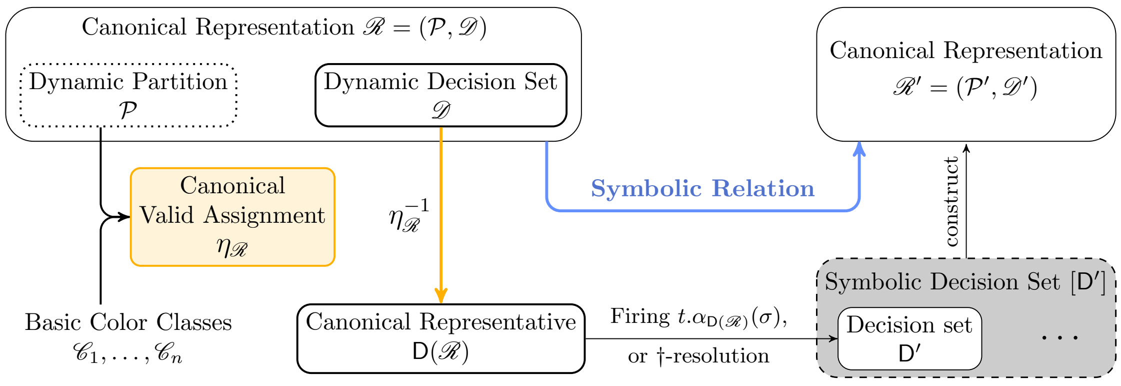

Canonical Representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$
Dynamic Partition $\mathcal{P}$
Dynamic Decision Set $\mathscr{D}$
Canonical Representation $\mathscr{R}' = (\mathcal{P}', \mathscr{D}')$
Canonical Valid Assignment $\eta_{\mathscr{R}}$
$\eta_{\mathscr{R}}^{-1}$
Symbolic Relation
construct
Symbolic Decision Set $[\mathsf{D}']$
Basic Color Classes $\mathscr{C}_1, \ldots, \mathscr{C}_n$
Canonical Representative $\mathsf{D}(\mathscr{R})$
Firing $t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)$, or $\dagger$-resolution
Decision set D'
...

Figure 3.22: Illustration of symbolic relations between canonical representations.

and dynamic decision set $\mathscr{D}$. From the dynamic partition we get the canonical valid assignment $\eta_{\mathscr{R}}$ mapping colors in the basic color classes to dynamic subclasses, as described in Def. 3.72. This leads to the canonical representative $\mathsf{D}(\mathscr{R})$ of $\mathscr{R}$, defined by $\mathsf{D}(\mathscr{R}) = \eta_{\mathscr{R}}^{-1}(\mathscr{D})$ (cp. Def. 3.73). From there, we fire a transition instance $t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)$ or resolve a †, arriving at a decision set D'. We then construct the canonical representation $\mathscr{R}'$ of the corresponding symbolic decision set $[\mathsf{D}']$, as described in Sec. 3.3.2. This yields a symbolic relation between the canonical representations $\mathscr{R}$ and $\mathscr{R}'$.

We now consider again the running example Signal Sending Satellites and demonstrate the symbolic relations.

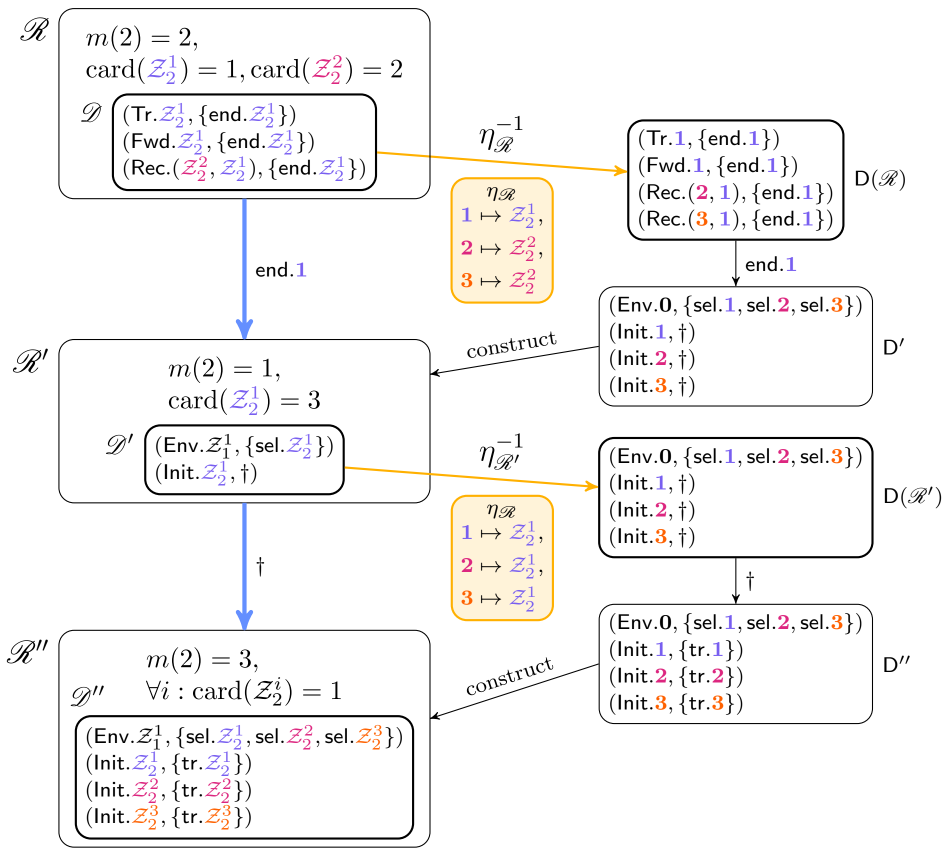

$\mathscr{R}$
$m(2) = 2$,
$\mathrm{card}(\mathcal{Z}_2^1) = 1, \mathrm{card}(\mathcal{Z}_2^2) = 2$
$\mathscr{D}$
$(\mathsf{Tr}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Fwd}.\mathcal{Z}_2^1, \{\mathsf{end}.\mathcal{Z}_2^1\})$
$(\mathsf{Rec}.(\mathcal{Z}_2^2, \mathcal{Z}_2^1), \{\mathsf{end}.\mathcal{Z}_2^1\})$
$\eta_{\mathscr{R}}^{-1}$
$\eta_{\mathscr{R}}$
$1 \mapsto \mathcal{Z}_2^1$,
$2 \mapsto \mathcal{Z}_2^2$,
$3 \mapsto \mathcal{Z}_2^2$
$(\mathsf{Tr}.1, \{\mathsf{end}.1\})$
$(\mathsf{Fwd}.1, \{\mathsf{end}.1\})$
$(\mathsf{Rec}.(2, 1), \{\mathsf{end}.1\})$
$(\mathsf{Rec}.(3, 1), \{\mathsf{end}.1\})$
$\mathsf{D}(\mathscr{R})$
end.1
end.1
$(\mathsf{Env}.\mathbf{0}, \{\mathsf{sel}.1, \mathsf{sel}.2, \mathsf{sel}.3\})$
$(\mathsf{Init}.1, \dagger)$
$(\mathsf{Init}.2, \dagger)$
$(\mathsf{Init}.3, \dagger)$
D'
construct
$\mathscr{R}'$
$m(2) = 1$,
$\mathrm{card}(\mathcal{Z}_2^1) = 3$
$\mathscr{D}'$
$(\mathsf{Env}.\mathcal{Z}_1^1, \{\mathsf{sel}.\mathcal{Z}_2^1\})$
$(\mathsf{Init}.\mathcal{Z}_2^1, \dagger)$
$\eta_{\mathscr{R}'}^{-1}$
$\eta_{\mathscr{R}}$
$1 \mapsto \mathcal{Z}_2^1$,
$2 \mapsto \mathcal{Z}_2^1$,
$3 \mapsto \mathcal{Z}_2^1$
$(\mathsf{Env}.\mathbf{0}, \{\mathsf{sel}.1, \mathsf{sel}.2, \mathsf{sel}.3\})$
$(\mathsf{Init}.1, \dagger)$
$(\mathsf{Init}.2, \dagger)$
$(\mathsf{Init}.3, \dagger)$
$\mathsf{D}(\mathscr{R}')$
†
†
$(\mathsf{Env}.\mathbf{0}, \{\mathsf{sel}.1, \mathsf{sel}.2, \mathsf{sel}.3\})$
$(\mathsf{Init}.1, \{\mathsf{tr}.1\})$
$(\mathsf{Init}.2, \{\mathsf{tr}.2\})$
$(\mathsf{Init}.3, \{\mathsf{tr}.3\})$
D''
construct
$\mathscr{R}''$
$m(2) = 3$,
$\mathscr{D}''$
$\forall i : \mathrm{card}(\mathcal{Z}_2^i) = 1$
$(\mathsf{Env}.\mathcal{Z}_1^1, \{\mathsf{sel}.\mathcal{Z}_2^1, \mathsf{sel}.\mathcal{Z}_2^2, \mathsf{sel}.\mathcal{Z}_2^3\})$
$(\mathsf{Init}.\mathcal{Z}_2^1, \{\mathsf{tr}.\mathcal{Z}_2^1\})$
$(\mathsf{Init}.\mathcal{Z}_2^2, \{\mathsf{tr}.\mathcal{Z}_2^2\})$
$(\mathsf{Init}.\mathcal{Z}_2^3, \{\mathsf{tr}.\mathcal{Z}_2^3\})$

Figure 3.23: The symbolic relations from Fig. 3.10 between the corresponding canonical representations.

Example 3.75. Consider once more the canonical representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ from Fig. 3.21. This representation is illustrated in the top left of Fig. 3.23. We have demonstrated the construction of the canonical valid assignment $\eta_{\mathscr{R}}$ and the canonical representation $\mathsf{D}(\mathscr{R}) = \eta_{\mathscr{R}}^{-1}(\mathscr{D})$ in Example 3.71. From this decision set, only the transition *end*.1 can fire. Doing so leads to the decision set D' depicted below $\mathsf{D}(\mathscr{R})$. Left of this decision set is the canonical representation $\mathscr{R}'$ of the corresponding symbolic decision set $[\mathsf{D}']$. This can be built as described in Sec. 3.3.2. This implies, according to Def. 3.74, that there exists a symbolic relation $\mathscr{R}[\![end.1\rangle\!\rangle\mathscr{R}'$, represented as the blue edge from $\mathscr{R}$ to $\mathscr{R}'$ in Fig. 3.23.

As the symbolic decision set represented by $\mathscr{R}'$ only contains one element, namely D', in this special case we trivially have $\mathsf{D}(\mathscr{R}') = \mathsf{D}'$. From D', we can resolve a †, arriving at D''. The canonical representation of the symbolic decision set $[\mathsf{D}'']$ is $\mathscr{R}''$. This leads, once again by Def. 3.74 and analogously to the above, to a relation $\mathscr{R}'[\![\dagger\rangle\!\rangle\mathscr{R}''$, also depicted as a blue edge in Fig. 3.23. $\lhd$

We just formulated symbolic relations for canonical representations, aligning closely with those defined for symbolic decision sets (cp. Def. 3.20). Consequently, the following lemma, relating the two concepts to each other, arises as a natural consequence. It states that for every symbolic relation between canonical representations there is a corresponding symbolic relation between symbolic decision sets, and vice versa.

Lemma 3.76. *Let $\overline{\mathsf{D}}$ and $\overline{\mathsf{D}'}$ be the arbitrary representatives and $\mathscr{R}$ and $\mathscr{R}'$ be the canonical representations of the two symbolic decision sets $[\overline{\mathsf{D}}]$ and $[\overline{\mathsf{D}'}]$, respectively. Then the symbolic relations correspond to each other as follows:*

1. $\overline{\mathsf{D}}[\![t.\alpha_{\overline{\mathsf{D}}}(\sigma)\rangle\!\rangle\overline{\mathsf{D}'} \Rightarrow \exists s \in S_G : \mathscr{R}[\![t.\alpha_{\mathsf{D}(\mathscr{R})}(s(\sigma))\rangle\!\rangle\mathscr{R}'$ *and* $\overline{\mathsf{D}}[\![\dagger\rangle\!\rangle\overline{\mathsf{D}'} \Rightarrow \mathscr{R}[\![\dagger\rangle\!\rangle\mathscr{R}'$.
2. $\mathscr{R}[\![t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)\rangle\!\rangle\mathscr{R}' \Rightarrow \exists s \in S_G : \overline{\mathsf{D}}[\![t.\alpha_{\overline{\mathsf{D}}}(s(\sigma))\rangle\!\rangle\overline{\mathsf{D}'}$ *and* $\mathscr{R}[\![\dagger\rangle\!\rangle\mathscr{R}' \Rightarrow \overline{\mathsf{D}}[\![\dagger\rangle\!\rangle\overline{\mathsf{D}'}$.

Proof.

1. Let $\overline{\mathsf{D}}[\![t.\alpha_{\overline{\mathsf{D}}}(\sigma)\rangle\!\rangle\overline{\mathsf{D}'}$. Then, by Def. 3.20, $\exists \mathsf{D}'' \in [\overline{\mathsf{D}'}] : \overline{\mathsf{D}}[t.\alpha_{\overline{\mathsf{D}}}(\sigma)\rangle\mathsf{D}''$. By Prop. 3.18, there is a $s_{\overline{\mathsf{D}}} \in S_{\overline{\mathsf{D}}}$ such that $s_{\overline{\mathsf{D}}}(\sigma) = \alpha_{\overline{\mathsf{D}}}(\sigma)$. Additionally, since $\mathsf{D}(\mathscr{R}) \in [\overline{\mathsf{D}}]$, there is a $s_{\mathsf{D}(\mathscr{R})} \in S_G$ such that $s_{\mathsf{D}(\mathscr{R})}(\overline{\mathsf{D}}) = \mathsf{D}(\mathscr{R})$. Applying $s_{\mathsf{D}(\mathscr{R})}$ to the ordinary firing relation above yields by Lemma 3.14 that $\exists s_{\mathsf{D}(\mathscr{R})}(\mathsf{D}'') \in [\overline{\mathsf{D}'}] : \mathsf{D}(\mathscr{R})[t.s_{\mathsf{D}(\mathscr{R})}(s_{\overline{\mathsf{D}}}(\sigma))\rangle s_{\mathsf{D}(\mathscr{R})}(\mathsf{D}'')$. Finally, again by Prop. 3.18, there is a $s'_{\mathsf{D}(\mathscr{R})} \in S_{\mathsf{D}(\mathscr{R})}$ such that $s'_{\mathsf{D}(\mathscr{R})}(s_{\mathsf{D}(\mathscr{R})}(s_{\overline{\mathsf{D}}}(\sigma))) = \alpha_{\mathsf{D}(\mathscr{R})}(s_{\mathsf{D}(\mathscr{R})}(s_{\overline{\mathsf{D}}}(\sigma)))$. Let now $\mathsf{D}''' = s'_{\mathsf{D}(\mathscr{R})}(s_{\mathsf{D}(\mathscr{R})}(\mathsf{D}''))$ and $s = s_{\mathsf{D}(\mathscr{R})} \circ s_{\overline{\mathsf{D}}}$. Then we have $\mathsf{D}''' \in [\mathsf{D}(\mathscr{R}')]$ since $\mathsf{D}''' \in [\mathsf{D}''] = [\overline{\mathsf{D}'}] = [\mathsf{D}(\mathscr{R}')]$, and therewith
$$\exists \mathsf{D}''' \in [\mathsf{D}(\mathscr{R}')] : \mathsf{D}(\mathscr{R})[t.\alpha_{\mathsf{D}(\mathscr{R})}(s(\sigma))\rangle\mathsf{D}'''.$$
This, by Def. 3.74, implies $\mathscr{R}[\![t.\alpha_{\mathsf{D}(\mathscr{R})}(s(\sigma))\rangle\!\rangle\mathscr{R}'$.

   The case of †-resolution works analogously, albeit easier since there are no transition modes involved.
2. Works analogously to 1. □

With this we close the subsection on symbolic relations between canonical representations. In the following subsection, we will use these relations to define the symbolic two-player game with canonical representations as vertices.

3.3.4 The Canonical Two-Player Game

The goal is to use canonical representations instead of individual decision sets or (arbitrary representatives of) symbolic decision sets as vertices in a two-player game. The edges $(\mathscr{R}, \mathscr{R}')$ in this game are built from relations $\mathscr{R}[\![t.\alpha_{\mathsf{D}(\mathscr{R})}(\sigma)\rangle\!\rangle\mathscr{R}'$ and $\mathscr{R}[\![\dagger\rangle\!\rangle\mathscr{R}'$, depending on the properties of $\mathscr{R}$. For example, if $\mathscr{R}$ describes nondeterministic situations in the Petri game, then the edges from $\mathscr{R}$ are built in such a way that Player 0 (representing the system) cannot win the game from there. This means before we get to the definition of the canonical two-player game, we must define the relevant properties of canonical representations.

Recall that the important properties (cp. p. 51) a decision set can have are:

- being environment-dependent,
- containing a bad place resp. target place,
- being a deadlock,
- being terminating,
- being nondeterministic.

In Lemma 3.16 and Cor. 3.17 we showed that all decision sets in one equivalence class share the same properties. Thus, we defined a symbolic decision set to have one of the above properties iff its individual members (and in particular its arbitrary representative) have the respective property. We now analogously define these properties for canonical representations based on the properties of its canonical representative.

Definition 3.77 (Properties of canonical representation). Let $\mathscr{R}$ be a canonical representation of a symbolic decision set. We then say $\mathscr{R}$ is environment-dependent, containing a bad place resp. target place, a deadlock, terminating, or nondeterministic if and only if $\mathsf{D}(\mathscr{R})$ has the respective property. ◁

From this definition and the preceding recap of properties of symbolic decision sets we immediately get the following corollary.

Corollary 3.78. *Every symbolic decision set shares its properties with its canonical representation.*

In this Section 3.3, we accomplished the following so far:

1. We established definitions for canonical representations of symbolic decision sets.
2. We proceeded to formulate symbolic relations between these representations.
3. We have just defined the properties that a representation can possess.

With these components in place, we are now equipped to define the canonical two-player game for a given proper symmetric Petri game.

The rationale behind introducing canonical representations is to serve as the vertices within this canonical two-player game. Similar to the symbolic two-player game defined in Section 3.2.3, the edges in the canonical two-player game are constructed based on the relations between the canonical representations, contingent on the properties of the source representation.

Now that we have established these concepts, we can proceed to define, for a given proper symmetric Petri game G, the canonical two-player game $\widehat{\mathbb{G}}(G)$. Let a proper symmetric Petri game $G = (\mathscr{C}, P_{\mathrm{S}}, P_{\mathrm{E}}, T, J, F, g, M_0, \mathrm{Obj}, P_{\circledast})$ with expansion $\mathrm{Exp}(G) = (\mathsf{P}_{\mathrm{S}}, \mathsf{P}_{\mathrm{E}}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Cond}, \mathsf{P}_{\circledast})$ be given. The definition of $\widehat{\mathbb{G}}(G)$ is a direct adaptation of Definition 3.26 (for $\overline{\mathbb{G}}(G)$) to the framework of canonical representations:

Definition 3.79 (Canonical two-player game). The *canonical two-player game* over a finite graph $\widehat{\mathbb{G}}(G) = (\widehat{\mathbb{V}}_0, \widehat{\mathbb{V}}_1, \widehat{v}_0, \widehat{\mathbb{E}}, \widehat{\mathrm{Win}})$ is has the following components:

- The 0-vertices $\widehat{\mathbb{V}}_0$ are all canonical representations of symbolic decision sets in G that are *not* environment-dependent.
- The 1-vertices $\widehat{\mathbb{V}}_1$ are all canonical representations of symbolic decision sets in G that *are* environment-dependent.
- The initial vertex is canonical representation $\mathscr{R}_0$, such that

$$\mathsf{D}(\mathscr{R}_0) := \mathsf{D}_0 = \{(p.c, \dagger) \mid p.c \in \mathsf{M}_0 \cap \mathsf{P}_{\mathrm{S}}\} \cup \{(e.d, post(e.d)) \mid e.d \in \mathsf{M}_0 \cap \mathsf{P}_{\mathrm{E}}\}.$$

- the *labeled edge relation* $\widehat{\mathbb{E}} \subseteq \widehat{\mathbb{V}} \times (\mathsf{T} \cup \{\dagger\}) \times \widehat{\mathbb{V}}$ is defined as follows: If $\mathscr{R}$ contains a bad place, is a deadlock, is terminating, or is nondeterministic, there is only a †-labeled self-loop originating from $\mathscr{R}$. Otherwise, consider three disjunct cases for edges originating in $\mathscr{R}$:

  Case $\mathscr{R} \in \widehat{\mathbb{V}}_1$; i.e., all players have decided for a commitment set, but cannot proceed without the environment. Then for all $t \in T$ and $\overline{\sigma} \in \Sigma(t)_{\mathsf{D}(\mathscr{R})}$, $(\mathscr{R}, t.\overline{\sigma}, \mathscr{R}') \in \widehat{\mathbb{E}}$ iff $\mathscr{R}[\![t.\overline{\sigma}\rangle\!\rangle\mathscr{R}'$.

  Case $\mathscr{R} \in \widehat{\mathbb{V}}_0$ and $\mathscr{R}[\![\dagger\rangle\!\rangle$; i.e., at least one system player has yet to decide for a commitment set. Then $(\mathscr{R}, \dagger, \mathscr{R}') \in \widehat{\mathbb{E}}$ iff $\mathscr{R}[\![\dagger\rangle\!\rangle\mathscr{R}'$.

  Case $\mathscr{R} \in \widehat{\mathbb{V}}_0$ and $\neg\mathsf{D}[\![\dagger\rangle\!\rangle$; i.e., all system players made their decisions and can proceed without the environment. Then for all $t \in T$ and $\overline{\sigma} \in \Sigma(t)_{\mathsf{D}(\mathscr{R})}$ with $pre(t.\overline{\sigma}) \cap \mathsf{P}_{\mathrm{E}} = \emptyset$, $(\mathscr{R}, t.\overline{\sigma}, \mathscr{R}') \in \widehat{\mathbb{E}}$ iff $\mathscr{R}[\![t.\overline{\sigma}\rangle\!\rangle\mathscr{R}'$.
- $\widehat{\mathrm{Win}}$ depending on Obj and $P_{\circledast}$ as follows:
  - if Obj = Safety then $\widehat{\mathrm{Win}} := \mathrm{Safety}(\widehat{\mathbb{F}})$, where $\widehat{\mathbb{F}}$ contains all canonical representations that are a deadlock, nondeterministic, or contain a bad place,
  - if Obj = Reach then $\widehat{\mathrm{Win}} := \mathrm{Reach}(\widehat{\mathbb{F}})$, where $\widehat{\mathbb{F}}$ contains all canonical representations that contain a target place. $\lhd$

The following lemma formalizes that $\overline{\mathbb{G}}(G)$ and $\widehat{\mathbb{G}}(G)$ are indeed “the same” by stating that they are *isomorphic*. We say two two-player games are isomorphic iff there is a bijective bisimulation between them.

Lemma 3.80. *Let G be a proper high-level Petri game with an underlying symmetric net. Then $\overline{\mathbb{G}}(G)$ and $\widehat{\mathbb{G}}(G)$ are isomorphic.*

Proof. We prove that the relation $R = \{([\mathsf{D}(\mathscr{R})], \mathscr{R}) \mid \mathscr{R} \in \widehat{\mathbb{V}}\}$ is a bijective bisimulation between $\overline{\mathbb{G}}(G)$ and $\widehat{\mathbb{G}}(G)$. Since all symbolic decision sets have exactly one canonical representation (Thm. 3.68), and all canonical representations by definition represent exactly one symbolic decision set, this relation is a bijection. We have by definition $(\overline{v}_0, \widehat{v}_0) \in R$. From the definition of the symbolic and the canonical two-player game we get that they have the same objective Obj, and Cor. 3.78 implies that $[\mathsf{D}(\mathscr{R})] \in \overline{\mathbb{V}}_1 \Leftrightarrow \mathscr{R} \in \widehat{\mathbb{V}}_1$ and $[\mathsf{D}(\mathscr{R})] \in \overline{\mathbb{F}} \Leftrightarrow \mathscr{R} \in \widehat{\mathbb{F}}$. Finally, Cor. 3.78 together with Lemma 3.76 implies that $([\mathsf{D}(\mathscr{R})], \mathscr{R}) \in R \wedge ([\mathsf{D}(\mathscr{R})], [\mathsf{D}']) \in \overline{\mathbb{E}} \Rightarrow \exists \mathscr{R}' : (\mathscr{R}, \mathscr{R}') \in \widehat{\mathbb{E}} \wedge ([\mathsf{D}'], \mathscr{R}') \in \mathscr{R}$ and $([\mathsf{D}(\mathscr{R})], \mathscr{R}) \in R \wedge (\mathscr{R}, \mathscr{R}') \in \widehat{\mathbb{E}} \Rightarrow ([\mathsf{D}(\mathscr{R})], [\mathsf{D}(\mathscr{R}')]) \in \overline{\mathbb{E}} \wedge (\mathsf{D}'(\mathscr{R}'), \mathscr{R}') \in \mathscr{R}$, yielding that R is a bisimulation, which completes the proof. $\square$

From this lemma, together with Lemma 3.29 and Theorem 3.34 we finally get the soundness of the canonical two-player game.

Theorem 3.81. *Given a proper high-level Petri game G with an underlying symmetric net, there is a winning strategy for the system players in the P/T Petri game* $\mathrm{Exp}(G)$ *if and only if there is a winning strategy for Player 0 in $\widehat{\mathbb{G}}(G)$.*

We compare the two two-player game $\overline{\mathbb{G}}(G)$ and $\widehat{\mathbb{G}}(G)$. The size of $\widehat{\mathbb{G}}(G)$ is the same as of $\overline{\mathbb{G}}(G)$ (exponential in the size of G). This means, using $\widehat{\mathbb{G}}(G)$, the question whether a winning strategy in $\mathsf{G} = \mathrm{Exp}(G)$ exists can still be answered in single exponential time [FO17].

We now compare the construction of $\overline{\mathbb{G}}(G)$ and $\widehat{\mathbb{G}}(G)$. In $\overline{\mathbb{G}}(G)$ we must, for a newly reached vertex D', test if it is equivalent to another, already existing, representative (solving the orbit problem). This means we check for all symmetries $s \in S$ whether $s(\mathsf{D}')$ is already a vertex in the game. In the best case, if we directly find the vertex, this is 1 comparison. In the worst case, at step i with currently $|\mathbb{V}^i|$ vertices, we must make $O(|S_G| \cdot |\mathbb{V}^i|)$ comparisons (when no symmetric vertex is in the game so far).

In the approach using canonical representations in $\widehat{\mathbb{G}}(G)$, to solve the constructive orbit problem, we must calculate the canonical representation of the reached symbolic decision set after each step. In Cor. 3.70 we stated that this can be accomplished in $O(|S_G|)$ time, with maximally $|S_G|$ comparisons. We must then check whether the canonical representation is already a vertex in the game. This is at least 1 comparison in the best case vs. $|\mathbb{V}^i|$ in the i-th step in the worst case.

Using $|S_G|$ to denote the number of comparisons required to construct the canonical representation of a reached symbolic decision set, we can establish the following relations between best and worst-case scenarios for arbitrary representative versus canonical representations in the i-th step:

Best Case **Arbitrary** Representations 1	$<$	**Best** Case **Canonical** Representations $\lvert S_G\rvert + 1$	$<$	**Worst** Case **Canonical** Representations $\lvert S_G\rvert + \lvert\mathbb{V}^i\rvert$	$<$	**Worst** Case **Arbitrary** Representations $\lvert S_G\rvert \cdot \lvert\mathbb{V}^i\rvert$

Nevertheless, it is important to note that the number of comparisons, $|S_G|$, serves as an upper bound in the cases of canonical representations. In practical terms, once a minimal representation $\mathscr{R} = (\mathcal{P}, \mathscr{D})$ of a reached symbolic decision set has been constructed in $O(1)$ time, the actual number of comparisons needed to obtain a canonical representation is $|S_\mathcal{P}|$. It always holds that $|S_\mathcal{P}| \leq |S_G|$. The case $|S_\mathcal{P}| = |S_G|$, only occurs when the partition $\mathcal{P}$ is trivial. However, this circumstance is highly contingent on the specific characteristics of both the underlying symmetric net and the symbolic decision set currently under consideration.

3.3.5 Experimental Results

In this section we briefly investigate the impact of using *canonical representations* for solving the realizability problem of distributed systems modeled with proper high-level Petri games G. The primary work on the implementation and experiments was conducted by Manuel Gieseking in the context of [GW21a].

The prototype discussed in Sec. 3.2.6 was extended and the algorithms presented above were implemented to obtain the same state space reduction by using canonical representations in $\widehat{\mathbb{G}}(G)$. Furthermore, a solving algorithm to exploit the reduced state space for the *realizability problem* (i.e., the question whether a winning strategy for the system players exists) of high-level Petri games was implemented. As a reference, an explicit approach which does not exploit any symmetries of the system was implemented. The algorithms are applied to the benchmark families presented in Sec. 3.2.6 and a new benchmark family *Client/Server* introduced in [GW21a]. The Client/Server benchmark family is similar to the Signal Sending Satellites example, as there are n computers that all must connect to the same computer chosen by the environment.

The complete results are contained in the corresponding artifact [GW21c]. An extract of the results for three of these benchmark families (CS being Client/Server, and DW, CM as in Sec. 3.2.6) are given in Table 3.2. We compare of the run times of the two approach using either canonical representations ($\widehat{\mathbb{G}}(G)$) or checking equivalence ($\overline{\mathbb{G}}(G)$) solving the realizability problem (✓/✗) for the 3 benchmark families. We additionally list the number of states $|\overline{\mathbb{V}}| = |\widehat{\mathbb{V}}|$ and number of symmetries $|S_G|$. A gray number of states $|\mathbb{V}|$ for the *explicit reference approach* indicates a timeout. Results are obtained on an AMD Ryzen™ 7 3700X CPU, 4.4 GHz, 64 GB RAM and a timeout (TO) of 30 minutes. The run times are in seconds.

We can see that for those benchmark families the extra effort of computing the canonical representations is worthwhile for most instances compared to the cost of checking the membership of a decision set in an equivalence class. This is not the case for all benchmark families.

In Fig. 3.24 instances of all benchmark families according to their number of symmetries and states are plotted. The comparison of the performance improvement of the canonical and the membership approach is conducted with respect to the number of states and symmetries of the input problem for the benchmark families *Package Delivery (PD)*, *Alarm System (AS)*, *CM*, *DW*, *DWs*, *CS*. Labels are the parameters of the benchmark. A blue (solid) marker indicates a performance increase when using canonical representa-

Table 3.2: Comparison of the run times of the two approaches using canonical representations or checking equivalence.

CS	$\lvert\mathbb{V}\rvert$	$\models$	$\lvert\overline{\mathbb{V}}\rvert = \lvert\widehat{\mathbb{V}}\rvert$	$\lvert S_G\rvert$	$\overline{\mathbb{G}}(G)$	$\widehat{\mathbb{G}}(G)$
1	21	✓	21	1	.38	**.36**
2	639	✓	326	2	**.63**	.64
3	45042	✓	7738	6	**5.20**	6.05
4	*7.225e6*	✓	3.100e5	24	151.62	**148.08**
5	*3.154e9*	-	-	120	TO	TO

DW	$\lvert\mathbb{V}\rvert$	$\models$	$\lvert\overline{\mathbb{V}}\rvert = \lvert\widehat{\mathbb{V}}\rvert$	$\lvert S_G\rvert$	$\overline{\mathbb{G}}(G)$	$\widehat{\mathbb{G}}(G)$
1	57	✓	57	1	.40	**.39**
2	457	✓	241	2	.67	**.62**
	...	...	...		...	
7	*4.055e6*	✓	5.793e5	7	100.67	**75.24**
8	*2.097e7*	✓	2.621e6	8	986.77	**671.04**
9	*1.053e8*	-	-	9	TO	TO

CM	$\lvert\mathbb{V}\rvert$	$\models$	$\lvert\overline{\mathbb{V}}\rvert = \lvert\widehat{\mathbb{V}}\rvert$	$\lvert S_G\rvert$	$\overline{\mathbb{G}}(G)$	$\widehat{\mathbb{G}}(G)$
2/1	155	✓	79	2	**.49**	.52
2/2	2883	✗	760	4	**1.07**	1.08
2/3	58501	✗	5548	12	**4.38**	5.94
2/4	*1.437e6*	✗	33250	48	15.12	**14.40**
2/5	*3.419e7*	✗	1.701e5	240	296.05	**185.81**
2/6	*8.376e8*	-	-	1440	TO	TO
3/1	702	✓	147	6	.71	**.58**
3/2	45071	✓	4048	12	**4.46**	4.99
3/3	*3.431e6*	✗	91817	36	89.35	**49.90**
3/4	*2.622e8*	-	-	144	TO	TO
4/1	2917	✓	239	24	**1.24**	1.42
4/2	*6.587e5*	✓	16012	48	25.42	**14.09**
4/3	*1.546e8*	-	-	144	TO	TO

tions, while a striped marker indicates a performance decrease. The color of the marker shows the percentaged in- or decrease in performance when using canonical representations while solving high-level Petri games. Blue (solid) indicates a performance gain when using the canonical approach. This shows that the benchmarks in general benefit from the canonical approach for an increasing number of states (the right blue (solid) area). However, the *DWs* benchmark (a simplified version of *DW*) exhibits the opposite behavior. This is most likely explained by the very simple structure, which favors a quick member check.

Additionally, I must note here that it is highly probable that the full advantage of the canonical representations approach may not be apparent, as in the implementation

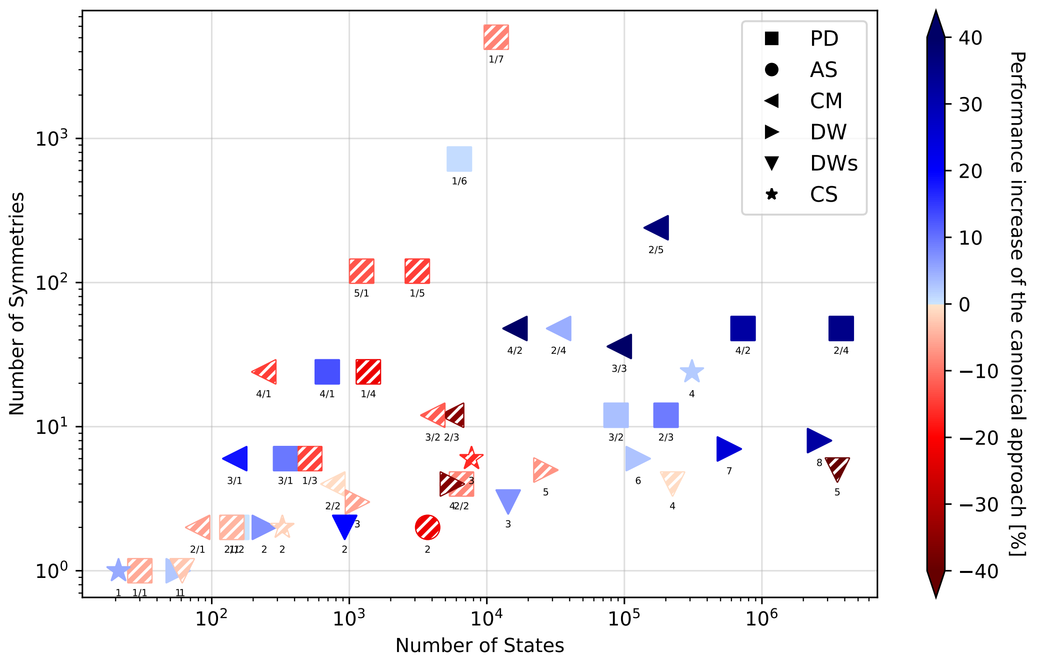


Figure 3.24: Comparing the percentage performance gain of the canonical and the equivalence check approach.

"ordered representatives" we generated. This corresponds to the procedure of considering only trivial representations and ordering them according to the order $<_{\mathscr{R}}$ defined in Sec. 3.3.2. This means, in particular, that the discussion at the end of Sec. 3.3.4 is not applicable, as we are always considering the entire set of symmetries. Therefore, these results merely provide an indication of the potential benefits of canonical representations. This needs to be further investigated in subsequent experiments.

The algorithms are integrated in AdamSYNT [FGO15; FGHO17; Gie20], open source, and available online[5]. Additionally, an artifact with the current version running in a virtual machine for reproducing and checking all experimental data with provided scripts [GW21c] was created.

3.4 Related Work

An active research area is Petri net synthesis [BBD15; BD15; DT22]. Two-player games are studied under the name Petri net supervisory control [BBD15; ABP23], inspired by the work of Ramadge and Wonham on discrete event systems [RW87]. A significant body of work on synthesis and control based on Petri nets is in this area (cf. [Giu92; ZWD95; RSB03; BDLV05]), also for structured Petri nets like modules of signal nets [DHJ+04].

[5]https://github.com/adamtool/highlevel, Accessed: 28 November 2023

However, these approaches solve the single-process synthesis problem, as opposed to the multi-process synthesis problem for concurrent systems considered in this paper.

The most prominent model for distributed synthesis is that of Pnueli and Rosner [PR90]. In their setting, the restricted access to the full system is modeled with processes communicating via single-writer single-reader shared variables with synchronous concurrency. Thus, each process is only *partially informed* of the system's state, since the received information is limited by the input variables of the process. After a series of isolated decidability results, notably for pipelines [PR90] and rings [KV01], *information forks* [FS05] were indentified as the necessary and sufficient criterion for undecidability. For architectures without information forks, the synthesis problem can be solved by an automata-theoretic construction that iteratively eliminates processes from the architecture (in the order of growing informedness). The complexity, however, is nonelementary in the number of processes.

Zielonka's *asynchronous automata* [Zie87; Zie95] have been proposed as an alternative setting for distributed synthesis [GLZ04; GGMW13; MTY05; MW14]. In [Gim22], undecidability of the synthesis problem for six or more processes is shown. There are various decidability results for restricted cases, e.g., concerning the dependencies of actions [GLZ04] or the synchronization behavior [MTY05]. Decidability, albeit again with nonelementary complexity, has also been obtained for acyclic communication structures [GGMW13; MW14].

Petri games based on P/T Petri nets were introduced in [FO14; FO17]. They exploit concurrency and causality in defining a notion of informedness for the players. In [FO17] it is shown that the problem whether the system players have a winning strategy for a safety objective, is undecidable for unbounded Petri games. However, for Petri games with one environment player and a bounded number of system players the problem is EXPTIME-complete. The winning strategy can be obtained in single-exponential time by a reduction to a two-player graph game. In [Gie22] this result is generalized to different local winning conditions for the system players, such as reachability and Büchi conditions. In [Hec21; FGHO22], decidability of Petri games with global winning conditions is shown. In [FG17] it was shown that also for one system player and a bounded number of environment players the synthesis problem is also EXPTIME-complete.

A formal connection between games on asynchronous automata and Petri games is established in [BFH19]. In [Han23], the undecidability result for asynchronous automata from [Gim22] is translated to Petri games with six players, and decidability for Petri games with a global safety condition and up to four players is shown. In [HO22], a decidability result for Petri games with a synchronization condition is presented. In [Fin15] a *bounded synthesis* approach was introduced. It sets a bound for the size of the strategy and constitutes a semi-decision procedure, optimized in finding small implementations. Parameterized high-level Petri games have been introduced in [GO21] with the aim of defining benchmark families of P/T Petri games.

For practical applications, high-level Petri nets in the form of Coloured Petri Nets (CPN) have been introduced [GL81; Jen96; Rei13]. In CPNs, individual data values are represented by coloured tokens to describe concurrent systems succinctly. Boolean conditions on these tokens appear as guards of transitions, and expressions define which of these tokens are moved when a transition fires. In general, multisets of coloured tokens may appear as markings. In [Jen96], a translation from CPNs back into normal P/T Petri nets, here called expansion, is defined.

There is a significant body of work regarding symmetries. For high-level Petri nets the notion of equivalent markings and the idea of exploiting symmetries was originally introduced in [HJJJ84; HJJJ86]. The symbolic two-player game is inspired by the symbolic reachability graph for high-level nets from [CDFH97], and the calculation of canonical representatives [CDFH91a] from [CDFH93]. For obtaining the symmetries of the system efficiently, several approaches on different subclasses of high-level Petri nets had been introduced, e.g., in [DH89; CDFH91a; CDFH93; Lin91; Sch95]. There are also several for efficiency improvements for systems with different degrees of symmetrical behavior [HITZ95; BHI04; BC04]. In [CDFH97] the idea of using equivalent transitions in addition to the equivalent marking for the creation of the symbolic reachability graph is lifted to CPNs. For low-level Petri nets the reduction ideas are introduced in [Sta91]. From then on lots of work has been done following that direction, e.g., [Sch00a; Sch00b; Wol15]. Using symmetries for the alleviation of state-explosion problems are also common in model checking [CEJS98; CJEF96; CFJ93]. The complications that arise when using BDDs for the symmetric state space evaluation in this context is elaborated in [CJEF96].

3.A Appendix: Correctness of the Low-Level Reduction

We prove Theorem 3.12. The structure of this proof is taken directly from [Gie22], but adapted to *proper* P/T Petri games. Restricting ourselves to this subclass of Petri games allowed us to significantly simplify the definition of the corresponding two-player game, which also facilitates proving correctness of this reduction.

Before we start with the proof, we briefly introduce the notion of *strategy trees* for two-player games. Given a strategy $f : \mathbb{V}^*\mathbb{V}_0 \to \mathbb{V}$ for Player 0 in a two-player game $\mathbb{G} = (\mathbb{V}_0, \mathbb{V}_1, \mathbb{E}, v_0, \mathrm{Win})$, we can inductively build the (infinite) corresponding strategy tree $\mathbb{T}_f = (\mathbb{T}, \mathbb{E}_\mathbb{T}, l, r)$, as a special case of directed, acyclic graphs: $\mathbb{T}$ is the set of vertices, $\mathbb{E}_\mathbb{T} \subseteq \mathbb{T} \times \mathbb{T}$ is the set of directed edges, $l : \mathbb{T} \to \mathbb{V}$ is the labeling function, and r is the root of $\mathbb{T}_f$. We start with $l(r) = v_0$.

Then, in breadth-first order, we go through and and extend the tree. Let $v_\mathbb{T}$ be the currently considered vertex. If $l(v_\mathbb{T}) \in \mathbb{V}_1$, then we add, for every v' with $(l(v_\mathbb{T}), v') \in \mathbb{E}$, a corresponding vertex $v'_\mathbb{T}$ with $l(v'_\mathbb{T}) = v$ to $\mathbb{T}$, and extend $\mathbb{E}_\mathbb{T}$ by $(v_\mathbb{T}, v'_\mathbb{T})$. If $l(v_\mathbb{T}) \in \mathbb{V}_0$, then let $rv^1_\mathbb{T} \cdots v^n_\mathbb{T} v_\mathbb{T}$ be the unique path from r to $v_\mathbb{T}$ in $\mathbb{T}_f$. We add a single vertex $v'_\mathbb{T}$ with $l(v'_\mathbb{T}) = f(l(r)l(v^1_\mathbb{T}) \cdots l(v^n_\mathbb{T})l(v_\mathbb{T}))$ to $\mathbb{T}$, and extend $\mathbb{E}_\mathbb{T}$ by $(v_\mathbb{T}, v'_\mathbb{T})$.

Construction 3.82 (From two-player game strategy to Petri game strategy). Let f be a winning strategy for Player 0 in the two-player game $\mathbb{G}(\mathsf{G})$ corresponding to a proper Petri game $\mathsf{G} \in \mathrm{Exp}(\mathbf{G})$. Let $\mathbb{T}_f = (\mathbb{T}, \mathbb{E}_\mathbb{T}, l, r)$ be the strategy tree corresponding to f.

By traversing $\mathbb{T}_f$ in breadth-first order, we step-wise create an occurrence net $\mathsf{O}^\xi = (\mathsf{B}^\xi, \mathsf{E}^\xi, \mathsf{H}^\xi, \mathsf{K}^\xi_0)$, an initial homomorphism π^ξ from O^ξ to $\mathsf{N}(\mathsf{G})$, and a function $k : \mathbb{T} \to$

$\mathcal{R}(\mathsf{O}^\xi)$ associating tree vertices to cuts. This will yield a winning strategy $\xi = (\mathsf{O}^\xi, \pi^\xi)$ for the system players in G.

(IA) For the root $r \in \mathbb{T}$ we create a new condition $\mathsf{b} \in \mathsf{B}^\xi$ in O^ξ for each $\mathsf{p} \in \mathsf{M}(l(r)) \subseteq \mathsf{P}$. Every condition is mapped by π^ξ to the respective place. The thereby created cut is K_0^ξ and we set $k(r) = \mathsf{K}_0^\xi$.

(IS) Consider an edge $(v_\mathbb{T}, \tau, v'_\mathbb{T}) \in \mathbb{E}_\mathbb{T}$ with $l(v_\mathbb{T}) = \mathsf{D}$ and $l(v'_\mathbb{T}) = \mathsf{D}'$ for which we already considered the vertex $v_\mathbb{T}$.

If Obj = Safety and $\mathsf{M}(l(v_\mathbb{T}))$ contains a target place, then we do not add anything to the strategy, and stop considering the future of the vertex $v_\mathbb{T} \in \mathbb{T}_f$. Else, we proceed as follows:

Case $\tau = \dagger$. Nothing is added to O^ξ and the cut is copied, i.e., $k(v_\mathbb{T}) = k(v'_\mathbb{T})$.

Case $\tau \in \mathcal{T}$. We add to O^ξ a fresh event e with $\pi^\xi(\mathsf{e}) = \tau$. The preset of e then is in the cut associated to $v_\mathbb{T}$, i.e., $pre(\mathsf{e}) = k(v_\mathbb{T}) \cap \{\mathsf{b} \in \mathsf{B}^\xi \mid \pi^\xi \in pre(\tau)\}$. For each place $\mathsf{p} \in post(\tau)$, we add a corresponding condition b with $\pi^\xi(\mathsf{b}) = \mathsf{p}$ to O^ξ. The set of these conditions constitute the postset of e. By this construction we directly get $\pi^\xi(pre(\mathsf{e})) = pre(\pi^\xi(\mathsf{e}))$ and $\pi^\xi(post(\mathsf{e})) = post(\pi^\xi(\mathsf{e}))$. Finally, we associate to $v'_\mathbb{T}$ the cut $k(v'_\mathbb{T}) = (k(v'_\mathbb{T}) \setminus pre(\mathsf{e})) \cup post(\mathsf{e})$. $\triangleleft$

Lemma 3.83 (Correctness of Construction 3.82). *Let* $\mathsf{G} \in \mathrm{Exp}(\mathbf{G})$*. For a given winning strategy* f *for Player* 0 *in* $\mathbb{G}(\mathsf{G})$*, Construction 3.82 yields a winning strategy* $\xi = (\mathsf{O}^\xi, \pi^\xi)$ *for the system players in* G*.*

Proof. Let the Petri game be given by $\mathsf{G} = (\mathsf{P}_\mathrm{S}, \mathsf{P}_\mathrm{E}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Obj}, \mathsf{P}_\circledast)$, and $\mathbb{G}(\mathsf{G}) = (\mathbb{V}_0, \mathbb{V}_1, \mathbb{E}, v_0, \mathrm{Win})$ the corresponding two-player game. Let $f : \mathbb{V}^*\mathbb{V}_0$ be a winning strategy for Player 0 in $\mathbb{G}$ and $\mathbb{T}_f$ the corresponding strategy tree. We show that $\xi = (\mathsf{O}^\xi, \pi^\xi)$ resulting from Construction 3.82 is a winning strategy for the system players in G.
Subprocess: Firstly, we demonstrate that ξ is a *subprocess* of an unfolding $\mathsf{U} = (\mathsf{O}, \pi)$ of $\mathsf{N} = \mathsf{N}(\mathsf{G})$. The net O^ξ is an occurrence net, which can be easily seen from the construction.

π^ξ obviously only maps conditions into places and events into transitions, and by construction $\pi^\xi(\mathsf{K}_0^\xi) = \{\pi^\xi(\mathsf{b}) \mid \mathsf{b} \in \mathsf{K}_0^\xi\} = \{\pi^\xi(\mathsf{b}) \mid \pi^\xi(\mathsf{b}) \in \mathsf{M}(l(r))\} = \mathsf{M}(\mathsf{D}_0) = \mathsf{M}_0$. Since, as already noted in Construction 3.82, we have for all $\mathsf{e} \in \mathsf{E}^\xi$ that $\pi^\xi(pre(\mathsf{e})) = pre(\pi^\xi(\mathsf{e}))$ and $\pi^\xi(post(\mathsf{e})) = post(\pi^\xi(\mathsf{e}))$, all that is left to show that π^ξ is an initial homomorphism, is that π^ξ is injective on events with the same preset. Aiming a contradiction, assume there are $\mathsf{e}_1, \mathsf{e}_2 \in \mathsf{E}^\xi$ with $\mathsf{e}_1 \neq \mathsf{e}_2$ and $\pi^\xi(\mathsf{e}_1) = \pi^\xi(\mathsf{e}_2)$ and $pre(\mathsf{e}_1) = pre(\mathsf{e}_2)$. Then, these events can only be generated by different branches of the strategy tree $\mathbb{T}_f$ because whenever in a step **(IS)** a new event e is added to O^ξ, the next vertex is associated to the cut reached by firing e, with *fresh conditions* in e's postset. Hence, $pre(\mathsf{e}_1) = pre(\mathsf{e}_2)$ can only be satisfied for events in different branches. But we cannot have two *different* events mapping to the same transition ($\pi^\xi(\mathsf{e}_1) = \pi^\xi(\mathsf{e}_2)$) in different branches while preserving the preset due to the scheduling of system transitions.

This means that ξ is a branching process of N. Thus, the strategy is a subprocess of an unfolding $\mathsf{U} = (\mathsf{O}, \pi)$ (a maximal branching process of N)
Justified refusal: Let $\mathsf{U} = (\mathsf{O}, \pi) = (\mathsf{B}, \mathsf{E}, \mathsf{H}, \mathsf{K}_0, \pi^{\xi})$ be the unfolding of N. To show that the system players only refuse to play transitions of the unfolding which are not violating the game play, i.e., they do not disallow pure environment transitions and do not allow an instance of a transition for which they already refused another instance in the same situation (condition), we have to show that

$$\begin{aligned}\forall \mathsf{e} \in \mathsf{E} :& (\mathit{pre}(\mathsf{e}) \subseteq \mathsf{B}^{\xi} \wedge \mathsf{e} \notin \mathsf{E}^{\xi}) \\ &\quad \Rightarrow (\exists \mathsf{b} \in \mathit{pre}(\mathsf{e}) \cap \mathsf{B}_{\mathrm{S}}\ \forall \mathsf{e}' \in \mathit{post}(\mathsf{b}) : \pi(\mathsf{e}') = \pi(\mathsf{e}) \Rightarrow \mathsf{e}' \notin \mathsf{E}^{\xi}).\end{aligned}$$

Let $\mathsf{e} \in \mathsf{E}$ with $\mathit{pre}(\mathsf{e}) \subseteq \mathsf{B}^{\xi}$ and $\mathsf{e} \notin \mathsf{E}^{\xi}$. From $\mathsf{e} \in \mathsf{E}$ follows that there is a sequence of transitions with $\mathsf{M}_0[\mathsf{t}_0, \dots, \mathsf{t}_n\rangle\mathsf{M}$ with $\mathsf{M}[\pi(\mathsf{e})\rangle$. Due to Construction 3.82 we know that $\mathit{pre}(\mathsf{e}) \subseteq \mathsf{B}^{\xi}$ yields that all $b \in \mathit{pre}(\mathsf{e})$ have been added via the construction. We consider two cases: (i) there is no vertex $v_{\mathbb{T}} \in \mathbb{T}$ with $\mathit{pre}(\mathsf{e}) \subseteq k(v_{\mathbb{T}})$, and (ii) there is such a vertex.

(i) All conditions in $\mathit{pre}(\mathsf{e})$ have been created. Since $\mathsf{e} \in \mathsf{E}$, all conditions must have been created in a single branch in $\mathbb{T}_{\xi}$, since else they would be in conflict. However, since there is no vertex still having all conditions in the associated cut, there must be an event in the strategy that is generated in the same branch and which takes tokens of $\mathit{pre}(\mathsf{e})$ before all conditions $\mathsf{b} \in \mathit{pre}(\mathsf{e})$ are created in the construction. Let $\mathsf{e}^{\star} \in \mathsf{E}^{\xi}$ be such an event such that the edge $(v^{\star}_{\mathbb{T}}, \pi(\mathsf{e}), v'_{\mathbb{T}}) \in \mathbb{E}_{\mathbb{T}}$ by which $\mathsf{e}^{\star}$ is added to O^{ξ} is minimal in $\mathbb{T}_f$. Let $\mathsf{K}^{\star} = k(v^{\star}_{\mathbb{T}})$ be the cut associated to $v^{\star}_{\mathbb{T}}$. We have $\mathit{pre}(\mathsf{e}) \cap \mathit{pre}(\mathsf{e}^{\star}) \neq \emptyset$. We denote the set of conditions in the preset of e created before firing $\mathsf{e}^{\star}$ with $\mathsf{X}^{pre} = \mathit{pre}(\mathsf{e}) \cap \mathsf{K}^{\star}$ and the ones which are later created by $\mathsf{X}^{post} = \mathit{pre}(\mathsf{e}) \backslash \mathsf{K}^{\star}$. We know $\mathsf{X}^{post} \neq \emptyset$ because otherwise $\mathit{pre}(\mathsf{e}) \subseteq k(v^{\star}_{\mathbb{T}})$. All conditions in $\mathsf{X}^{pre} \cup \mathit{pre}(\mathsf{e}^{\star})$ are pairwise concurrent due to $\mathit{pre}(\mathsf{e}^{\star}) \subseteq \mathsf{K}^{\star}$. Also, all conditions in $\mathsf{X}^{post} \cup \mathit{pre}(\mathsf{e}^{\star})$ are concurrent: otherwise, there would be a condition in $\mathsf{b} \in \mathsf{X}^{post}$ that is either causally related to or in conflict with a condition in $\mathsf{b} \in \mathit{pre}(\mathsf{e}^{\star})$. The case $\mathsf{b} < \mathsf{b}^{\star}$ contradicts that the edge generating e was minimal. The case $\mathsf{b}^{\star} < \mathsf{b}$ means that e is causally depending on e, which together with $\mathit{pre}(\mathsf{e}) \cap \mathit{pre}(\mathsf{e}^{\star}) \neq \emptyset$ contradicts $\mathsf{e} \in \mathsf{E}$. Finally, $\mathsf{b}^{\star} \# \mathsf{b}$ would imply that $\mathsf{e}^{\star}$ is generated from a different branch in $\mathbb{T}_f$.

Targeting a contradiction, assume $\exists \mathsf{b}_{\mathrm{E}} \in \mathit{pre}(\mathsf{e}^{\star}) \cap \mathsf{B}_{\mathrm{E}}$. Then, all system tokens must have been maximally progressed and are directly or indirectly depending on $\mathsf{e}^{\star}$. Hence, all conditions in X^{post} are dependent on $\mathsf{e}^{\star}$, but $\mathit{pre}(\mathsf{e}) \cap \mathit{pre}(\mathsf{e}^{\star}) \neq \emptyset$, thus e cannot be fireable and $\mathsf{e} \notin \mathsf{E}$. Contradiction. Therefore, $\mathit{pre}(\mathsf{e}^{\star}) \subseteq \mathsf{B}_{\mathrm{S}}$, and since we have a proper Petri game and thereby no mixed communication, we have $\mathit{pre}(\mathsf{e}) \subseteq \mathsf{B}_{\mathrm{S}}$.

Targeting a contradiction, *assume* the negation of the conclusion in the Justified Refusal property, i.e., assume $\forall \mathsf{b} \in \mathit{pre}(\mathsf{e}) \cap \mathsf{B}_{\mathrm{S}}\ \exists \mathsf{e}' \in \mathit{post}(\mathsf{b}) : \pi(\mathsf{e}') = \pi(\mathsf{e}) \wedge \mathsf{e}' \in \mathsf{E}^{\xi}$. Since $\mathit{pre}(\mathsf{e}) \subseteq \mathsf{B}_{\mathrm{S}}$ every condition in $\mathit{pre}(\mathsf{e})$, and especially every condition in X^{post} has such an event e' in its postset. Consider now such a condition $\mathsf{b} \in \mathsf{X}^{post}$ with an event $\mathsf{e} \in \mathit{post}(\mathsf{b})$ satisfying the above conjunction.

Assume now additionally $\mathsf{e}' \neq \mathsf{e}$. Due to the injectivity property of π, we know $\mathit{pre}(\mathsf{e}') \neq \mathit{pre}(\mathsf{e})$. Since $|\mathit{pre}(\mathsf{e}')| = |\mathit{pre}(\mathsf{e})|$ there must be a condition $\mathsf{b}' \in \mathit{pre}(\mathsf{e}) \setminus \mathit{pre}(\mathsf{e}')$. The condition b' cannot be in on $\mathsf{X}^{\mathit{post}}$ because then b' and b would be concurrent, and $\mathsf{b}' \in \mathit{pre}(\mathsf{e}')$. Thus, $\mathsf{b}' \in \mathsf{X}^{\mathit{pre}}$ an in particular $\mathsf{b}' \in \mathsf{K}^\star$. The condition b' can also not be in $\mathsf{X}^\star$ because then again b' and b would be concurrent and therefore $\mathsf{b}' \in \mathit{pre}(\mathsf{e}')$. So we now have $\mathsf{b}' \in (\mathsf{K}^\star \cap \mathit{pre}(\mathsf{e})) \setminus \mathit{pre}(\mathsf{e}^\star)$. This means b' is already existing in the cut $\mathsf{K}^\star$ and not stolen by $\mathsf{e}^\star$. So either it is still in the cut where the last condition of $\mathsf{X}^{\mathit{post}}$ is created and thus b and b' are concurrent and $\mathsf{b}' \in \mathit{pre}(\mathsf{e}')$. Otherwise, there must be another (minimal w.r.t. generation by $\mathbb{T}_f$) event $\mathsf{e}_2^\star$ taking b' before all tokens of $\mathit{pre}(\mathsf{e})$ are generated. This can only be done finitely often, due to the eventually creation of $\mathit{pre}(\mathsf{e})$. Contradiction.
Thus, $\mathsf{e}' \neq \mathsf{e}$ yields the final *contradiction*, because $\mathsf{e} \notin \mathsf{E}^\xi$.

(ii) Let $v_\mathbb{T} \in \mathbb{T}$ such that $\mathit{pre}(\mathsf{e}) \subseteq k(v_\mathbb{T})$. Let $l(v_\mathbb{T}) = \mathsf{D}$ and therefore $\mathit{pre}(\pi(\mathsf{e})) \subseteq \mathsf{M}(\mathsf{D})$.

Case $\mathit{pre}(\mathsf{e}) \subseteq \mathsf{B}^\xi_\mathrm{E}$. If D contains a target place (in the case Obj = Reach) then the construction stops when considering $v_\mathbb{T}$. Thus, the possibly added conditions when reaching $v_\mathbb{T}$ have *no* events to their postset and can therefore not violate the Justified Refusal property. Therefore, there must be another vertex before $v_\mathbb{T}$ such that its associated cut already contains the preset of e. Consider this vertex instead.
If D contains *no* target place (in the case Obj = Reach), but is either nondeterministic, a deadlock, or contains a bad place (in the case Obj = Safety), then the strategy f cannot be winning. Thus, there must be a successor reached from D by firing a transition or resolving a †-symbol.
If $\mathsf{D} \in \mathbb{V}_1$, then all successors of D must be in f. Since $\mathsf{D}[\pi(\mathsf{e})\rangle\mathsf{D}'$, for some D', we have a corresponding edge in $\mathbb{T}_f$, and e is eventually added to the strategy.
If $\mathsf{D} \in \mathbb{V}_0$ then there exists exactly one edge from $v_\mathbb{T}$ in $\mathbb{T}_f$ which corresponds to firing a transitions with only system places in its preset, or resolving a †-symbol. Neither of these relations moves tokens environment places. Since we consider proper Petri games we have a recurrently interfering environment. Thus, we cannot fire infinitely many transitions involving only system players, and will eventually arrive at a vertex $v'_\mathbb{T}$ with $l(v'_\mathbb{T}) \in \mathbb{V}_1$ and $\mathit{pre}(\mathsf{e}) \subseteq k(v'_\mathbb{T})$.

Case $\exists \mathsf{b} \in \mathit{pre}(\mathsf{e}) \cap \mathsf{B}^\xi_\mathrm{S}$. Remember that we have $\mathsf{e} \notin \mathsf{E}^\xi$ and $\mathit{pre}(\mathsf{e}) \subseteq \mathsf{B}^\xi$. We now have $\mathsf{b} \in \mathit{pre}(\mathsf{e})$, and consider an event $\mathsf{e}' \in \mathit{post}(\mathsf{b})$ such that $\pi(\mathsf{e}') = \pi(\mathsf{e})$. We show that $\mathsf{e}' \notin \mathsf{E}^\xi$. Construction 3.82 we see that $\mathsf{e} \notin \mathsf{E}^\xi$ implies that for *none* of the vertices $v'_\mathbb{T} \in \mathbb{T}$ with $\mathit{pre}(\mathsf{e}) \subseteq k(v'_\mathbb{T})$ there is an outgoing edge $(v'_\mathbb{T}, \pi(\mathsf{e}), v''_\mathbb{T}) \in \mathbb{E}_\mathbb{T}$. We consider the two cases that (a) $\pi(\mathsf{e})$ is *not* chosen in D and (b) $\pi(\mathsf{e})$ *is* chosen in D.

(a) Since $\pi(\mathsf{e})$ is not chosen in D, there exists one condition $\mathsf{b}' \in \mathit{pre}(\mathsf{e}) \cap \mathsf{B}^\xi_\mathrm{S}$ with $(\pi(\mathsf{b}'), \mathcal{T}) \in \mathsf{D}$ and $\pi(\mathsf{e}) \notin \mathcal{T}$. As long as the token in b' is not moved, no new commitment set can be chosen. Therefore, $\pi(\mathsf{b}')$ forbids all instances of $\pi(\mathsf{e})$, and thus, $\mathsf{e}' \notin \mathsf{E}^\xi$.

(b) This means $\mathsf{D}[\pi(\mathsf{e})\rangle$ but the transition is never fired. Thus, either there is an infinite path in ξ that starts in $l(v_\mathbb{T})$ and is not taking any tokens of the preset of

$\pi(\mathsf{e})$, or there is an event $\mathsf{e}^\star$ taking a token from $pre(\mathsf{e})$. In the former case $\pi(\mathsf{e})$ is enabled in every state $v'_{\mathbb{T}} \in \mathbb{T}$ of the infinite path and $pre(\mathsf{e}) \subseteq k(v'_{\mathbb{T}})$ because no token of the preset is taken. Hence, $\mathsf{e}' \notin \mathsf{E}^\xi$ because $\mathsf{b} \in pre(\mathsf{e}) \cap pre(\mathsf{e}')$. In the letter case $\mathsf{e}^\star \neq \mathsf{e}'$ because there cannot be two instances of the same transition $\pi(\mathsf{e})$ enabled in the same cut of an unfolding of a safe Petri net. If $\mathsf{e}^\star$ takes the token of b, we have $\mathsf{e}' \notin \mathsf{E}^\xi$. Otherwise, $\mathsf{e}^\star$ takes a token residing in another condition $\widetilde{\mathsf{b}} \neq \mathsf{b}$ with $\widetilde{\mathsf{b}} \in pre(\mathsf{e})$. If $\widetilde{\mathsf{b}} \in \mathsf{B}^\xi_{\mathrm{S}}$ then D is nondeterministic and f is not winning. Contradiction If $\widetilde{\mathsf{b}} \in \mathsf{B}^\xi_{\mathrm{E}}$, then this tree vertex would correspond to a 1-vertex in $\mathbb{G}(\mathsf{G})$, and all postset edges would be considered in f. With that, we would have $\mathsf{e} \in \mathsf{E}^\xi$. Contradiction.

Determinism: Targeting a contradiction, assume that the constructed strategy ξ is nonderterminic, i.e., there exists a condition $\mathsf{b} \in \mathsf{B}^\xi_{\mathrm{S}}$ a cut $\mathsf{K} \in \mathcal{R}(\mathsf{O}^\xi)$ with $\mathsf{b} \in \mathsf{K}$ such that there are two events $\mathsf{e}_1, \mathsf{e}_2 \in \mathsf{E}^\xi$ with $\mathsf{e}_1 \neq \mathsf{e}_2$ and $\mathsf{b} \in pre(\mathsf{e}_1) \cap pre(\mathsf{e}_2)$ such that $\mathsf{K}[\mathsf{e}_1\rangle$ and $\mathsf{K}[\mathsf{e}_2\rangle$. Due to Construction 3.82, this means there are two vertices $v_{\mathbb{T}1}, v_{\mathbb{T}2} \in \mathbb{T}$ with $pre(\mathsf{e}_1) \subseteq k(v_{\mathbb{T}1})$ and $pre(\mathsf{e}_2) \subseteq k(v_{\mathbb{T}2})$, and edges $(v_{\mathbb{T}1}, \pi(\mathsf{e}_1), v_{\mathbb{T}}{'_1}), (v_{\mathbb{T}2}, \pi(\mathsf{e}_2), v_{\mathbb{T}}{'_2}) \in \mathbb{E}_{\mathbb{T}}$, because e_1 and e_2 have been added to the strategy. Let $l(v_{\mathbb{T}1}) = \mathsf{D}_1$ and $l(v_{\mathbb{T}2}) = \mathsf{D}_2$. We consider the two cases of one of the two edges lying *before* the other in $\mathbb{T}_f$ and the two edges being in different branches of $\mathbb{T}_f$.

CASE W.l.o.g. $(v_{\mathbb{T}1}, \pi(\mathsf{e}_1), v_{\mathbb{T}}{'_1})$ lies before $(v_{\mathbb{T}2}, \pi(\mathsf{e}_2), v_{\mathbb{T}}{'_2})$ in $\mathbb{T}_f$. Since e_1 fired first and $b \in pre(\mathsf{e}_1)$, the token on b is removed and can never be put back because ξ is a branching process. Thus, e_2 cannot fire in any future of $v_{\mathbb{T}}{'_1}$. Thus, $\mathsf{e}_2 \notin \mathsf{E}^\xi$. Contradiction.
The case "$(v_{\mathbb{T}2}, \pi(\mathsf{e}_2), v_{\mathbb{T}}{'_2})$ lies before $(v_{\mathbb{T}1}, \pi(\mathsf{e}_1), v_{\mathbb{T}}{'_1})$ in $\mathbb{T}_f$" works analogously.

CASE $(v_{\mathbb{T}1}, \pi(\mathsf{e}_1), v_{\mathbb{T}}{'_1})$ and$(v_{\mathbb{T}2}, \pi(\mathsf{e}_2), v_{\mathbb{T}}{'_2})$ are in different branches in $\mathbb{T}_f$. Then it must be $v_{\mathbb{T}1} \neq v_{\mathbb{T}2}$ since else we have $\mathsf{D}_1 = \mathsf{D}_2$ and therewith $\mathsf{D}_1[\pi(\mathsf{D}_1)\rangle$ and $\mathsf{D}_1[\pi(\mathsf{D}_2)\rangle$ which would make D_1 nondeterministic and f would not be winning. Notice that D_1 would be nondeterministic since $\pi(\mathsf{D}_1) \neq \pi(\mathsf{D}_2)$, because no two instances of the same transition can ever be enabled at the same cut in the unfolding of a safe Petri net. Let $v_{\mathbb{T}\mathrm{E}}$ be the last common predecessor of $v_{\mathbb{T}1}$ and $v_{\mathbb{T}2}$ with $l(v_{\mathbb{T}\mathrm{E}}) \in \mathbb{V}_1$. Let $\mathsf{e}'_1, \mathsf{e}'_2\mathsf{E}^\xi$ be the environment events that leave $v_{\mathbb{T}\mathrm{E}}$ and lead to $v_{\mathbb{T}1}$ and $v_{\mathbb{T}2}$, respectively. Both these events move an environment token $\mathsf{b}_{\mathrm{E}} \in \mathsf{B}^\xi_{\mathrm{E}} \cap pre(\mathsf{e}'_1) \cap pre(\mathsf{e}'_2)$. Due to the scheduling in $\mathbb{G}(\mathsf{G})$, e_1 and e_2 causally depend on e'_1 and e'_2, respectively. Additionally, since $v_{\mathbb{T}1} \neq v_{\mathbb{T}2}$, we have $\mathsf{e}'_1 \neq \mathsf{e}_1$ and $\mathsf{e}'_2 \neq \mathsf{e}_2$ and therefore $\mathsf{e}'_1 < \mathsf{e}_1$ and $\mathsf{e}'_2 < \mathsf{e}_2$. This in particular means that $\mathsf{e}_1 \# \mathsf{e}_2$. But two events that are in conflict can never be enabled in the same cut unless they are the source of the conflict. Since e_1 and e_2 are *not* the source of the conflict ($\mathsf{e}'_1 < \mathsf{e}_1$ and $\mathsf{e}'_2 < \mathsf{e}_2$ and $\mathsf{e}'_1 \# \mathsf{e}'_2$) this leads to a contradiction.

Winning: We show that the constructed strategy ξ is winning. This depends on the objective for the system players in G.

CASE Obj = Safety. The system players in G have a safety objective with bad places $\mathsf{P}_\spadesuit \subseteq \mathsf{P}_{\mathrm{S}}$. We call ξ if it is deadlock-avoiding, and no run in ξ ever reaches a bad condition $\mathsf{b} \in \mathsf{B}^\xi_\spadesuit$. A bad condition is only added to ξ when in $\mathbb{T}_f$ a vertex $v_{\mathbb{T}}$ with $\mathsf{M}(l(v_{\mathbb{T}})) \cap \mathsf{P}_\spadesuit \neq \emptyset$ is reached. However, this means $l(v_{\mathbb{T}})$ contains a bad place and therefore f cannot be winning. Since f is assumed to be winning, we know that $\mathsf{B}^{\xi}_{\spadesuit} = \emptyset$, and therefore no run in ξ can ever contain a bad condition.

For showing that ξ is *deadlock-avoiding* we have to show that for every cut, if there is an event enabled in the unfolding, there must also be an event enabled in the strategy. Let $\mathsf{K} \in \mathcal{R}(\mathsf{O}^{\xi})$ and $\mathsf{e} \in \mathsf{E}$ with $\mathsf{K}[\mathsf{e}\rangle$. We have to show that there is an event $\mathsf{e}^{\xi} \in \mathsf{E}^{\xi}$ such that $\mathsf{K}[\mathsf{e}^{\xi}\rangle$. Since all conditions in K have been created in Construction 3.82, there either is a vertex $v_{\mathbb{T}} \in \mathbb{T}$ with $k(v_{\mathbb{T}}) = \mathsf{K}$ or there must be an event $\mathsf{e}^{\star}$ that takes a token from a condition $\mathsf{b} \in \mathsf{K}$ before all conditions in K are generated in the scheduling induced by $\mathbb{T}_f$. Since K is a cut, all contained conditions must be generated in one branch of the strategy tree, and $\mathsf{e}^{\star}$ must also be generated from that branch. This means in the latter case we have $\mathit{pre}(\mathsf{e}^{\star}) \subseteq \mathsf{K}$, because all conditions of K are still created and K is a cut, which means the creation of the conditions cannot be causally dependent on $\mathsf{e}^{\star}$. Thus, $\mathsf{e}^{\xi} = \mathsf{e}^{\star}$ satisfies the conditions above. Consider now the former case. In the two-player game $\mathbb{G}(\mathsf{G})$ each vertex has a successor and so has every vertex in the strategy tree $\mathbb{T}_f$. Let $(v_{\mathbb{T}}, \tau, v'_{\mathbb{T}}) \in \mathbb{E}_{\mathbb{T}}$ be the existing edge in $\mathbb{T}_f$ and $(\mathsf{D}, \tau, \mathsf{D}') \in \mathbb{E}$ the corresponding edge in $\mathbb{G}(\mathsf{G})$.
Consider $\tau = \dagger$. If $\mathsf{D} = \mathsf{D}'$ then we know that D is either terminating (which would mean there is no event enabled in the unfolding, contradicting the assumption $\mathsf{K}[\mathsf{e}\rangle$), or D is a deadlock or contains a bad place or is nondeterministic, contradicting that ξ is winning. If $\mathsf{D} \neq \mathsf{D}'$ then this corresponds to a $\dagger$ revolution, and we have $k(v_{\mathbb{T}}) = k(v'_{\mathbb{T}})$. Since D' cannot contain a $\dagger$-symbol, we then consider $v'_{\mathbb{T}}$ instead of $v_{\mathbb{T}}$.
Consider $\tau = \mathsf{t} \in \mathsf{T}$. Then in the construction, a corresponding event $\mathsf{e}^{\star}$ with $\pi(\mathsf{e}^{\star}) = \mathsf{t}$ and $\mathit{pre}(\mathsf{e}^{\star})$ is added to the strategy, satisfying the above conditions.

Case Obj = Reach. The system players in G have a reachability objective with target places $\mathsf{P}_{\heartsuit} \subseteq \mathsf{P}_{\mathrm{S}}$. We have to show that every run in the constructed strategy ξ contains a target transition $\mathsf{B}^{\xi}_{\heartsuit}$. We gather what we showed so far: Firstly, Construction 3.82 yields, for a given winning strategy tree $\mathbb{T}_f$ a strategy ξ for the system players in G. Consider now a part of the strategy tree between two vertices $v_{\mathbb{T}}, v'_{\mathbb{T}}$ such that $l(v_{\mathbb{T}}), l(v_{\mathbb{T}}) \in \mathbb{V}_1$, i.e., between two vertices representing environment dependent vertices in $\mathbb{G}(\mathsf{G})$. Then between these two vertices, the tree does not branch. Correspondingly, in the generated strategy ξ, the cuts $k(v_{\mathbb{T}})$ and $k(v'_{\mathbb{T}})$ are mcuts. In a strategy, there are no conflicts between two mcuts (Justified Refusal). Correspondingly, since every mcut in ξ is generated, and for every mcut K there is a corresponding vertex $v_{\mathbb{T}\mathrm{E}}$ with $k(v_{\mathbb{T}\mathrm{E}}) = \mathsf{K}$, this means that for every part of ξ between two mcuts K_1 and K_2 there is exactly one corresponding part of a branch in $\mathbb{T}_f$ beginning with a vertex $v_{\mathbb{T}\mathrm{E}}^1$ s.t. $k(v_{\mathbb{T}\mathrm{E}}^1) = \mathsf{K}_1$ and ending with a vertex $v_{\mathbb{T}\mathrm{E}}^2$ s.t. $k(v_{\mathbb{T}\mathrm{E}}^2) = \mathsf{K}_2$.
Let now ρ be a run in ξ. Since we have a recurrently interfering environment, we know that there ρ is built from parts between environment-dependent cuts. Considering now the corresponding branch (play) in $\mathbb{T}_f$ guarantees that a vertex containing a target place is reached. Thus, there is a target condition in ρ. $\square$

Construction 3.84 (From Petri game strategy to two-player game strategy). Let $\mathsf{G} \in \mathrm{Exp}(\mathbf{G})$ be a proper Petri game and let $\xi = (\mathsf{O}^{\xi}, \pi^{\xi})$ with occurrence net

$\mathsf{O}^\xi = (\mathsf{B}^\xi, \mathsf{E}^\xi, \mathsf{H}^\xi, \mathsf{K}_0^\xi)$ be a winning strategy for the system players in G. Let further $\mathbb{G}(\mathsf{G}) = (\mathbb{V}_0, \mathbb{V}_1, v_0, \mathbb{E}, \mathrm{Win})$ be the to G corresponding two-player game.

From ξ we define a function $f : \mathbb{V}^*\mathbb{V}_0 \to \mathbb{V}$, that will turn out to be a winning strategy for Player 0 in $\mathbb{G}(\mathsf{G})$. Given a word $\overline{v} = v_1 \cdots v_n v \in \mathbb{V}^*\mathbb{V}_0$, we define the successor $f(\overline{v})$ as follows:
If $\overline{v}$ is not a prefix of a play in $\mathbb{G}(\mathsf{G})$, then we choose an arbitrary successor vertex according to $\mathbb{E}$.
Otherwise, let $\mathsf{D} = v$.
If D is a deadlock, is terminating, or is nondeterministic, by definition there is only a $\dagger$-labeled self-loop originating from D. We therefore set $f(\overline{v}) = \mathsf{D}$.
Otherwise, let $\mathit{fs}(\overline{v})$ be the sequence of transitions that the edges between vertex in $\overline{v}$ are labeled with. It is easy to show that $\mathit{fs}(\overline{v})$ is a fireable transition sequence in G, and therefore, there is a corresponding, fireable event sequence (a configuration C) in G's unfolding.
If this configurations is *not* contained in the strategy ξ, then we choose an arbitrary successor vertex according to $\mathbb{E}$.
Otherwise, let $\mathsf{K} = \mathrm{cut}(\mathsf{C})$ be the corresponding cut. Then for all places $\mathsf{p} \in \mathsf{M}(\mathsf{D})$ there is a condition $\mathsf{p}^\xi \in \mathsf{K}$ such that $\pi(\mathsf{p}^\xi) = \mathsf{p}$. If there exists a $(\mathsf{p}, \dagger) \in \mathsf{D}$, then we define $f(\overline{v}) = \mathsf{D}'$ such that $(\mathsf{D}, \dagger, \mathsf{D}') \in \mathbb{E}$ and for all $(\mathsf{p}, \dagger) \in \mathsf{D}$ there is a $(\mathsf{p}, \mathcal{T}') \in \mathsf{D}'$ such that $\mathcal{T}' = \{\pi^\xi(\mathsf{e}) \mid \mathsf{e} \in \mathsf{E}^\xi \wedge \mathsf{p}^\xi \in \mathit{pre}(\mathsf{e})\}$.
Otherwise, we chose one of the events $\mathsf{e} \in \mathsf{E}^\xi$ with $\mathit{pre}(\mathsf{e}) \subseteq \mathsf{B}_\mathrm{S}^\xi$ and $\mathsf{K}[\mathsf{e}\rangle$. We define $f(\overline{v}) = \mathsf{D}'$, such that $(\mathsf{D}, \pi^\xi(\mathsf{e}), \mathsf{D}') \in \mathbb{E}$. $\lhd$

Lemma 3.85 (Correctness of Construction 3.84). *Let* $\mathsf{G} \in \mathrm{Exp}(\mathbf{G})$*. For a given winning strategy* $\xi = (\mathsf{O}^\xi, \pi^\xi)$ *for the system players in* G*, Construction 3.84 yields a winning strategy* f *for Player* 0 *in* $\mathbb{G}(\mathsf{G})$*.*

Proof. Let $\mathsf{G} = (\mathsf{P}_\mathrm{S}, \mathsf{P}_\mathrm{E}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Obj}, \mathsf{P}_\circledast)$ and $\mathbb{G}(\mathsf{G}) = (\mathbb{V}_0, \mathbb{V}_1, v_0, \mathbb{E}, \mathrm{Win})$. The function f is a *strategy* because it defines successors for $\mathbb{V}_0$ vertices and in all cases the definition either fits into the edges $\mathbb{E}$, or an arbitrary successor is chosen.

We show that f is *winning*. Let $\overline{v} = v_0 v_1 \cdots$ be a play in $\mathbb{G}(\mathsf{G})$. The sequence $\mathit{fs}(\overline{v})$ indeed corresponds to an initial firing sequence in G because all edges either keep the marking, or the successor marking corresponds to the firing relation. Thus, each v_i corresponds to a cut K_i in the unfolding of G. Even more, due to the construction choosing successors according to the firing of an event $\mathsf{e} \in \mathsf{E}^\xi$, each K_i is a cut in the strategy ξ. Since ξ is deterministic and all plays conforming to f choose every possible commitment set according to the events contained E^ξ, no vertex can be a nondeterministic decision set.

Case Obj = Safety. Then $\mathrm{Win} = \mathrm{Safety}(\mathbb{F})$, where $\mathbb{F}$ contains all decision sets that are a deadlock, nondeterministic, or contain a bad place. We have established above that no v_i is a nondeterministic decision set. Since ξ is winning, the play in ξ corresponding to $\mathit{fs}(\overline{v})$ contains no bad condition. Thus, no K_i contains bad condition, and thereby no v_i contains a bad place. Analogously, no K_i is a deadlock, and therefore, no v is a deadlock.

Case Obj = Reach. Then Win = Reach($\mathbb{F}$), where $\mathbb{F}$ contains all decision sets that contain a target place. Since ξ is winning, the play in ξ corresponding to $fs(\overline{v})$ contains at least one target condition. This target condition must be at least in at least one cut K_i, meaning that v_i contains a target place. This means $\overline{v}$ is won by Player 0. $\square$

Chapter 4

Finite and Complete Prefixes of Symbolic Unfoldings

Contents

Since strategies in a Petri game are defined as subprocesses of the Petri game's unfolding, the latter takes a crucial role when investigating Petri games. Specifically, as each subprocess constitutes a *prefix* of the unfolding, we will now explore prefixes in more detail. In particular, we will investigate the notion of complete finite prefixes of the symbolic unfolding of a high-level Petri net. Understanding prefixes of the symbolic unfolding lays groundwork for the definition of symbolic strategies for high-level Petri games in Chapter 5.

A complete prefix of a P/T Petri net's unfolding contains all information to verify, e.g., reachability of markings. In [McM95], McMillan presents an algorithm to compute a complete finite prefix of the unfolding of a given P/T Petri net. In a well-known paper [ERV02], Esparza, Römer, and Vogler improve this algorithm by defining and exploiting a total order on the set of configurations in the unfolding. We call the improved algorithm the "ERV-algorithm". It leads to a comparably small complete finite prefix of the unfolding.

In this chapter, we lift the concepts of complete prefixes and adequate orders to the level of symbolic unfoldings of high-level Petri nets. We consider the class of safe high-level Petri nets that have decidable guards and finitely many reachable markings. This class generalizes safe P/T Petri nets, and we obtain a generalized version of the ERV-algorithm creating a complete finite prefix of the symbolic unfolding of such a given high-level Petri net. Our results are a generalization of [ERV02] in the sense that if a P/T Petri net is viewed as a high-level Petri net (as discussed on p. 24), the new definitions of adequate orders and completeness of prefixes on the symbolic level, as well as the algorithm producing them, all coincide with their P/T counterparts.

We then proceed to identify an even more general class of so-called *symbolically compact* high-level Petri nets, where we drop the assumption of finitely many reachable markings, and instead assume the existence of a bound on the number of steps needed to reach all reachable markings. In such a case, the expansion is possibly not finite, and the original ERV-algorithm from [ERV02] therefore not applicable. We adapt the generalized ERV-algorithm by weakening the cut-off criterion to ensure finiteness of the resulting prefix. In this cut-off criterion we have to compare infinite sets of markings. We overcome this obstacle by symbolically representing these sets, using decidability of the guards to decide cut-offs. Finally, we present four new benchmark families for which we report on the results of applying a prototype implementation of the generalized ERV-algorithm.

We now present the running example for this chapter.

Running Example: Three Times Termination. Let $\mathit{Col} = \{0, \dots, m\}$ for a fixed m, and $\mathit{Var} = \{x, y, z, w\}$ be the given sets of colors and variables. Figure 4.1 shows the running example N for this chapter, called Three Times Termination. The set of initial markings is the singleton $\mathcal{M}_0 = \{M_0\}$ with $M_0 = \{\!\!\{ (a, 0), (b, 0) \}\!\!\}$, which is depicted in the net.

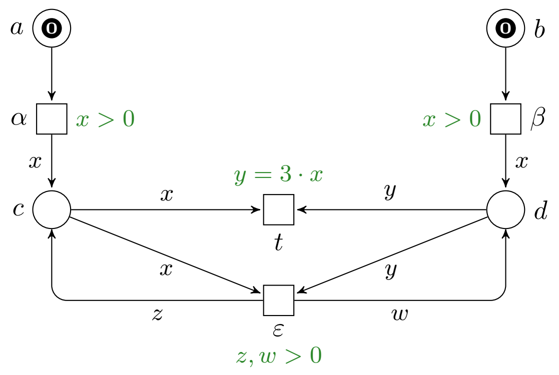


Figure 4.1: The running example N, called “Three Times Termination” with $\mathit{Col} = \{0, \dots, m\}$ for a fixed m, and $\mathit{Var} = \{x, y, z, w\}$.

From M_0, both α and β can fire. α takes the color 0 from place a and places a color $k \in \{1, \dots, m\}$ on place c when firing in mode $\{x \leftarrow k\}$. Analogously, β takes the 0 from b and places a k on place d. The mode $\{x \leftarrow 0\}$ is for both transitions excluded by their respective guard. When both α and β fire, the net arrives at a marking $\{\!|\, (c, k), (d, \ell) \,|\!\}$ with $0 < k, \ell \leq m$. From there, ε can fire arbitrarily often, always replacing the colors k, ℓ currently residing on c, d by any colors k', ℓ' with $0 < k', \ell' \leq m$ by firing in mode $\{x \leftarrow k, y \leftarrow \ell, z \leftarrow k', w \leftarrow \ell'\}$. From every marking $\{\!|\, (c, k), (d, \ell) \,|\!\}$ satisfying $\ell = 3 \cdot k$, transition t can fire in mode $\{x \leftarrow k, y \leftarrow \ell\}$, ending the execution of the net.

4.1 Finite Complete Prefixes in the P/T Case

This chapter serves as a concise overview of the results from [ERV02] for safe safe P/T Petri nets. We present the central concepts of complete prefixes and adequate orders.

For that, we briefly recall the notation from Sec. 2.1, with P/T Petri nets $\mathsf{N} = (\mathsf{P}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0)$: an initial branching process $\beta = (\mathsf{O}, \mathsf{h})$ of N consists of an occurrence net $\mathsf{O} = (\mathsf{B}, \mathsf{E}, \mathsf{H}, \mathsf{K}_0)$ and an initial homomorphism $\mathsf{h} : \mathsf{B} \cup \mathsf{E} \to \mathsf{P} \cup \mathsf{T}$ that is injective on events with same preset, i.e., $\forall \mathsf{e}_1, \mathsf{e}_2 \in \mathsf{E} : (\mathit{pre}(\mathsf{e}_1) = \mathit{pre}(\mathsf{e}_2) \wedge \mathsf{h}(\mathsf{e}_1) = \mathsf{h}(\mathsf{e}_2)) \Rightarrow \mathsf{e}_1 = \mathsf{e}_2$. Configurations C in β are causally closed sets of event that are not in conflict. The cut of C is given by $\mathrm{cut}(\mathsf{C}) = (\mathsf{K}_0 \setminus ({\to}\mathsf{C})) \cup (\mathsf{C}{\to})$ and describes the conditions occupied with tokens after firing C. We now define by $\mathrm{mark}(\mathsf{C}) := \mathsf{h}(\mathrm{cut}(\mathsf{C}))$ the marking in N represented by $\mathrm{cut}(\mathsf{C})$.

The definitions in this section, namely Def. 4.1, Def. 4.2, and Def. 4.3, are directly taken from [ERV02].

Definition 4.1 (Complete branching process (P/T) [ERV02]). A branching process β of a P/T Petri net N is *complete* iff for every reachable marking M in N there is a configuration C in β such that

- $\text{mark}(\mathsf{C}) = \mathsf{M}$ (i.e., M is represented in β), and
- for every transition t enabled by M there exists a configuration $\mathsf{C} \cup \{\mathsf{e}\}$ such that $\mathsf{e} \notin \mathsf{C}$ and e is labeled by t. $\lhd$

Recall (also from Sec. 2.1) that the maximal (w.r.t. the prefix relation) initial branching process of N is the unfolding $\Upsilon(\mathsf{N}) = (\mathsf{U}(\mathsf{N}), \pi^{\mathsf{N}})$, with the occurrence net $\mathsf{U}(\mathsf{N}) = (\mathsf{B}^{\mathsf{N}}, \mathsf{E}^{\mathsf{N}}, \mathsf{H}^{\mathsf{N}}, \mathsf{K}_0^{\mathsf{N}})$.

For a finite configuration C of a branching process $\beta = (\mathsf{O}, \mathsf{h})$ we denote by $\Uparrow\mathsf{C}$ the branching process $(\mathsf{O}', \mathsf{h}')$, where O' is the unique subnet of O whose set of nodes is $\{x \mid x \notin \mathsf{C} \cup \rightarrow \mathsf{C} \wedge \forall y \in \mathsf{C} : \neg(x \sharp y)\}$, and h' is the restriction of h to the net of O'. The branching process $\Uparrow\mathsf{C}$ is called the *future* of C. If for two configurations C_1 and C_2 we have $\text{mark}(\mathsf{C}_1) = \text{mark}(\mathsf{C}_2)$ then $\Uparrow\mathsf{C}_1$ and $\Uparrow\mathsf{C}_2$ are isomorphic, and we denote the by I_1^2 the isomorphism from $\Uparrow\mathsf{C}_1$ to $\Uparrow\mathsf{C}_2$.

Given a configuration C and a set Q of events if $\mathsf{C} \cup \mathsf{Q}$ is a configuration and $\mathsf{C} \cup \mathsf{Q} = \emptyset$ then we denote $\mathsf{C} \cup \mathsf{Q}$ by $\mathsf{C} \oplus \mathsf{Q}$. In this case we call $\mathsf{C} \oplus \mathsf{Q}$ an *extension* of C, and Q a *suffix* of C.

Definition 4.2 (Adequate order (P/T) [ERV02]). A partial order $\prec$ on the configurations of the unfolding of a P/T Petri net is an *adequate order* iff

- $\prec$ is well-founded,
- $\mathsf{C}_1 \subset \mathsf{C}_2$ implies $\mathsf{C}_1 \prec \mathsf{C}_2$, and
- $\prec$ is preserved by finite extensions; if $\mathsf{C}_1 \prec \mathsf{C}_2$ and $\text{mark}(\mathsf{C}_1) = \text{mark}(\mathsf{C}_2)$ then the isomorphism I_1^2 from above satisfies $\mathsf{C}_1 \oplus \mathsf{Q} \prec \mathsf{C}_2 \oplus I_1^2(\mathsf{Q})$ for all finite extensions $\mathsf{C}_1 \oplus \mathsf{Q}$ of C_1. $\lhd$

In [ERV02], three such adequate orders are presented. In particular, the authors present a *total* adequate order. This order uses the so-called *Foata normal form* of configurations.

Equipped with the definition of adequate orders, we get to the definition of *cut-off* events in a branching process.

Definition 4.3 (Cut-off event (P/T) [ERV02]). Let $\prec$ be an adequate order on the configurations of the unfolding of a Petri net. Let β be a prefix of the unfolding containing an event e. The event e is called a *cut-off event* (w.r.t. $\prec$) iff β contains an event e' s.t.

- $\text{mark}([\mathsf{e}]) = \text{mark}([\mathsf{e}'])$, and
- $[\mathsf{e}'] \prec [\mathsf{e}]$. $\lhd$

Using this notion of cut-off events, Esparza et al. present in [ERV02] an algorithm for the generation of a finite complete prefix of the unfolding. The algorithm iteratively extends the prefix under construction by events e (called possible extensions) such that $[\mathsf{e}]$ is minimal w.r.t. a given adequate order $\prec$ on the configurations of the unfolding. After adding an event (and its output conditions), it is checked whether it is a cut-off event. Events that structurally depend on cut-off events are *not* added to the prefix. When there are no more possible extensions, the algorithm terminates.

We call this algorithm the *ERV-algorithm*, named after its authors Esparza, Römer, and Vogler. In [ERV02], they show that when the adequate order $\prec$ used in the ERV-algorithm is *total*, then the resulting prefix has at most n non cut-off events, where n is the number of reachable markings in the considered Petri net. Since there is such a total adequate order (using the Foata normal form, cp. above), they thereby show how to generate a comparably small finite complete prefix of a Petri net's unfolding.

We do not present the algorithm here, in particular because the later presented algorithm, Alg. 1, is a generalization of the ERV-algorithm. Instead, we demonstrate the idea of the algorithm for the expansion of our running example:

Example 4.4. We show the result applying the the ERV-algorithm to $\mathsf{N} = \mathrm{Exp}(N)$, where N is the running example Three Times Termination from Fig. 4.1 for a fixed $m \in \mathbb{N}$. The places and transitions in the P/T net N (restricted to reachable places and fireable transitions) are then given by

$$\begin{aligned}\mathsf{P} =& \{a.0, b.0\} \cup \{c.k, d.k \mid 1 \leq k \leq m\}, \quad \text{and} \\ \mathsf{T} =& \{\alpha.\{x \leftarrow k\}, \beta.\{x \leftarrow k\} \mid 1 \leq k \leq m\} \\ & \cup \{t.\{x \leftarrow k, y \leftarrow \ell\} \mid \ell = 3 \cdot k, 1 \leq \ell \leq m\} \\ & \cup \{\varepsilon.\{x \leftarrow k, y \leftarrow \ell, z \leftarrow k', w \leftarrow \ell'\} \mid 1 \leq k, \ell, k', \ell' \leq m\},\end{aligned}$$

respectively. The expansion therefore consists of $2 + 2m$ places and $2m + \lfloor \frac{m}{3} \rfloor + m^4$ transitions.

The finite complete prefix of $\Upsilon(\mathsf{N})$, generated by the ERV-algorithm, is depicted in Fig. 4.2. The events and conditions are named after their label, with an additional superscript. We abbreviate the modes $\{x \leftarrow k\}$ of α and β by k, the modes $\{x \leftarrow k, y \leftarrow \ell\}$ of t by (k, ℓ), and the modes $\{x \leftarrow k, y \leftarrow \ell, z \leftarrow k', w \leftarrow \ell'\}$ of ε by (k, ℓ, k', ℓ'). Cut-off events and their output conditions are shaded blue, and the blue line indicates the complete finite prefix resulting from the original ERV-algorithm.

The ERV-algorithm initially generates the conditions $a.0'$ and $b.0'$ representing the initial marking $\{\!\!\{ a.0, b.0 \}\!\!\}$. The set of possible extensions is given by $2m$ events with labels $\alpha.k$ and $\beta.k$, where $1 \leq k \leq m$. The order in which events are added to the prefix depends on the used adequate order $\prec$ on the configurations. For this example, we assume the order to generate the events in Fig. 4.2 layer wise, from top to bottom, and inside each layer from left to right.

Thus, the first event that is added is $\alpha.1^0$, together with the condition $c.1^0$, representing $\alpha.1$'s postset $\{\!\!\{ c.1 \}\!\!\}$. For this event we have the cone configuration $[\alpha.1^0] = \{\alpha.1^0\}$ and $\mathit{mark}([\alpha.1^0]) = \{\!\!\{ c.1, b.0 \}\!\!\}$. Since we have not seen this marking before, $\alpha.1'$ is not a cut-off event. Analogously, none of the other events $\alpha.k'$ and $\beta.k'$ are cut-off events.

After adding any event of $\beta.\ell^0$ and the output condition $d.\ell^0$ (due to the assumed order, all α instances then are already part of the prefix) m^3 instances of ε are added to the possible extensions; the events $\varepsilon.(k, \ell, k', \ell')$ for all combinations of k, k', ℓ'. Additionally,

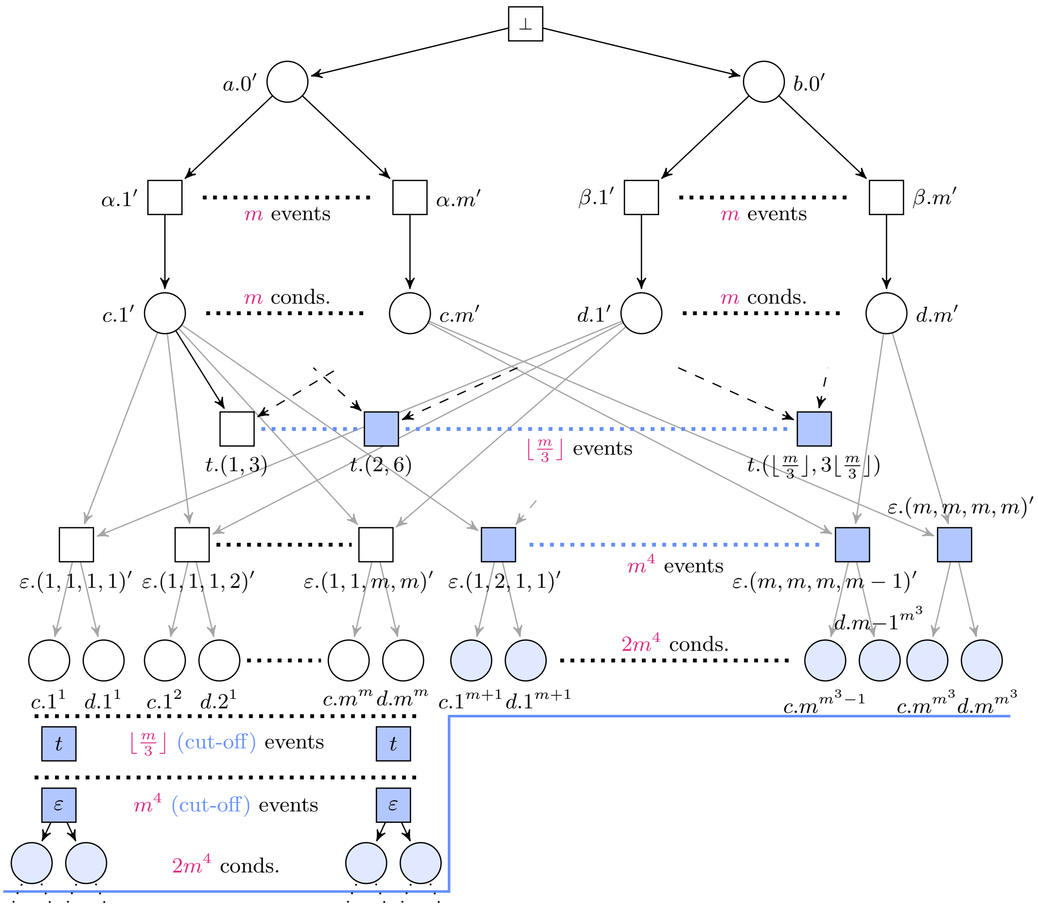


Figure 4.2: The finite complete prefix of the unfolding of the *expanded* running example "Three Times Termination" from Fig. 4.1, generated by the ERV-algorithm from [ERV02]

if there exists a condition $c.k^0$ such that $3 \cdot k = \ell$ then the event $t.(k, \ell)^0$ is added to the possible extensions.

We assume that first the t instances are added to the prefix. For the first one of these events $(t.(1,3)^0)$ we get $[t.(1,3)^0] = \{\alpha.1^0, \beta.3^0, t.(1,3)^0\}$ with $\mathit{mark}([t.(1,3)^0]) = \emptyset$. We have not seen the empty marking before, so $t.(1,3)^0$ is not a cut-off event. For all other k, ℓ, however, we have $[t.(k,\ell)^0] = \{\alpha.k^0, \beta.\ell^0.\ell^0, t.(k,\ell)^0\}$ and $\mathit{mark}([t.(k,\ell)^0]) = \emptyset = \mathit{mark}([t.(1,3)^0])$. Since we assumed $[t.(1,3)^0] \prec [t.(k,\ell)^0]$, for all other k, ℓ, the corresponding events $t.(k,\ell)^0$ are cut-off events.

For the events $\varepsilon.(1,1,k',\ell')'$ we have $\mathit{mark}([\varepsilon.(1,1,k',\ell')']) = \{\!|\, c.k', d.\ell' \,|\!\}$. Although these markings are already represented in the so far constructed prefix by the configuration $\{\alpha.k', \beta.\ell'\}$, they are *not* represented by *cone* configurations $[\mathsf{e}']$, i.e., there does not exist any e' such that $\mathit{mark}([\mathsf{e}']) = \{\alpha.k', \beta.\ell'\}$. This means these first m^2 events are not cut-off events.

For all following $\varepsilon.(k,\ell,k',\ell')'$ (with $k > 1$ or $\ell > 1$), however, $\mathit{mark}([\varepsilon.(k,\ell,k',\ell')']) =$

$\{\!\!\{ c.k', d.\ell' \}\!\!\} = \mathit{mark}([\varepsilon.(1,1,k',\ell')'])$, meaning we have seen the represented marking before, making all of these events cut-off events.

After the non cut-off events, however, we have to repeat the part from above for the ERV-algorithm to terminate. All of the then added events are cut-off events. All in all the complete finite prefix contains $6m^4 + 4m + 2\lfloor \frac{m}{3} \rfloor + 2$ nodes for every fixed m determining the color class $\mathit{Col} = \{0, 1, \dots, m\}$. $\lhd$

4.2 Generalizing Finite and Complete Prefixes

We combine ideas from [ERV02] (computing small finite and complete prefixes of unfoldings) with results from [CJ04] (symbolic unfoldings of high-level Petri nets) to define and construct complete finite prefixes of symbolic unfoldings of high-level Petri nets. We generalize the concepts and the ERV-algorithm from [ERV02] for safe P/T Petri nets to a class of safe high-level Petri nets, and compare this generalization to the original. We will see that for P/T nets interpreted as high-level nets, all generalized concepts (i.e., complete prefixes, adequate orders, cut-off events), and, as a consequence, the result of the generalized ERV-algorithm, all coincide with their P/T counterparts.

We frequently draw comparisons between the presented generalizations and the corresponding concepts and findings outlined in [ERV02], which are concisely presented in Sec. 4.1 above. For more comprehensive information, we direct the reader to the original article.

4.2.1 Properties of the Symbolic Unfolding.

Before we present the generalization of complete finite prefixes, we state three analogues of well-known properties of the unfolding of P/T Petri nets for the symbolic unfolding of high-level nets. These properties are:

(i) The cuts in the symbolic unfolding represent precisely the reachable markings in the high-level Petri net.

(ii) For every transition that can occur in the high-level Petri net, there is an event in the symbolic unfolding with corresponding label (and vice versa).

(iii) For any configuration, the part of the symbolic unfolding that "lies after" that configuration is the symbolic unfolding of the original high-level net structure with the initial markings being the ones represented by the configurations cut.

The properties are stated in Prop. 4.6, Prop. 4.7, and Prop. 4.8, respectively.

We recall notation for high-level Petri nets and symbolic unfoldings from Sec. 2.3.1 and Sec. 2.3.3: (symbolic) configurations are defined as sets of high-level events that are free of structural conflict and color conflict, and causally closed, and the configurations in a symbolic branching process β are collected in the set $\mathcal{C}(\beta)$. For a configuration C, we denoted by $\mathrm{cut}(C) = (B_0 \cup (C \rightarrow)) \setminus (\rightarrow C)$ the high-level conditions that are occupied after any concurrent execution of C. We established that $\mathrm{cut}(C)$ is a co-set, and that $\emptyset$ is a configuration with $\mathrm{cut}(\emptyset) = B_0$.

Let $e \in E$ be a high-level event. We define the so-called *cone configuration* $[e] := \{e' \in E \mid e' \leq e\}$. Additionally, we define the sets $\mathit{Var}_e := \{v_e \mid v \in \mathit{Var}(e)\}$ and $\mathit{Var}_\perp := \{v^b_\perp \mid b \in B_0\}$ of indexed variables, and for a set $E' \subseteq E \cup \{\perp\}$ we denote $\mathit{Var}_{E'} := \bigcup_{e \in E'} \mathit{Var}_e$. Note that, for every event e, $\mathit{pred}(e)$ (defined in Sec. 2.3) is a predicate over the variables $\mathit{Var}_{[e]\cup\{\perp\}}$. Recall that $\mathit{pred}(e)$ evaluates to true under an assignment $\theta : \mathit{Var}_{[e]\cup\{\perp\}} \to \mathit{Col}$ iff the events in $[e]$ can all fire in a single execution of the net in the modes represented by θ.

Let $\beta = (O, h)$ be a symbolic branching process with $O = (B, E, H, g, \mathcal{K}_0)$. To express the three properties, we introduce the notion of *instantiations* of configurations $C \in \mathcal{C}(\beta)$, choosing a mode for every event in C without creating color conflicts. This is realized by assigning to each variable $v_e \in \mathit{Var}_{C\cup\{\perp\}}$ a value in Col, such that all predicates $\mathit{pred}(e)$ with $e \in C$ evaluate to *true*. For each $e \in C$, the assignment of values to the indexed variables in Var_e corresponds to a mode of e.

Definition 4.5 (Instantiation of Symbolic Configuration). For a given symbolic configuration $C \in \mathcal{C}(\beta)$, an *instantiation of* C is a function $\theta : \mathit{Var}_{C\cup\{\perp\}} \to \mathit{Col}$, such that $\forall e \in C \cup \{\perp\} : \mathit{pred}(e)[\theta] \equiv \mathit{true}$, i.e., it satisfies predicate of every event in the configuration. The set of instantiations of C is denoted by $\Theta(C)$. $\lhd$

Note that it follows directly from the definition, that every symbolic configuration C has an instantiation θ. We introduce some useful notations. We denote by

$$\mathrm{cut}(C, \theta) := \{(b, c) \mid b \in \mathrm{cut}(C) \wedge \theta(\mathbf{v_e}(b)) = c\} \subseteq B \times \mathit{Col}$$

the *cut* of an "instantiated configuration", and by

$$\mathrm{mark}(C, \theta) := \{\!\!\{\, (h(b), c) \mid (b, c) \in \mathrm{cut}(C, \theta) \,\}\!\!\}$$

its *marking*. We collect both of these in

$$\mathrm{Cuts}(C) := \{\mathrm{cut}(C, \theta) \mid \theta \in \Theta(C)\}$$

and

$$\mathrm{Marks}(C) := \{\mathrm{mark}(C, \theta) \mid \theta \in \Theta(C)\}.$$

Note that in this notation, for the empty configuration $\emptyset$ we have $\mathrm{Cuts}(\emptyset) = \mathcal{K}_0$ and $\mathrm{Marks}(\emptyset) = \mathcal{M}_0$.

We now show in Prop. 4.6 and Prop 4.7 the properties (i) and (ii) described above.

Proposition 4.6. *Let N be a high-level Petri net and Υ its symbolic unfolding. Then* $\mathcal{R}(N) = \{\mathrm{mark}(C, \theta) \mid C \in \mathcal{C}(\Upsilon), \theta \in \Theta(C)\} \quad (= \bigcup_{C \in \mathcal{C}(\Upsilon)} \mathrm{Marks}(C))$.

Proof. The proof is an easy induction over the number n of transitions/events needed to reach a respective marking/cut. The induction anchor $n = 0$ is proved by using that π is an initial homomorphism which gives $\mathcal{M}_0 = \{\{\!\!\{\, (\pi(b), c) \mid (b, c) \in K_0 \,\}\!\!\} \mid K_0 \in \mathcal{K}_0\} = \{\{\!\!\{\, (\pi(b), c) \mid (b, c) \in K \,\}\!\!\} \mid K \in \mathrm{Cuts}(\emptyset)\} = \{\mathrm{mark}(\emptyset.\theta) \mid \theta \in \Theta(\emptyset)\}$. The induction step is realized by Prop. 4.7. $\square$

Proposition 4.7. *The symbolic unfolding $\Upsilon = (U, \pi)$ with events E of a high-level Petri net $N = (P, T, F, g, \mathcal{M}_0)$ satisfies $\forall C \in \mathcal{C}(\Upsilon)\ \forall \theta \in \Theta(C)\ \forall t \in T\ \forall \sigma \in \Sigma(t)$:*

$$\mathrm{mark}(C, \theta)[t, \sigma\rangle \ \Leftrightarrow\ \exists e \in E : \pi(e) = t \wedge \mathrm{cut}(C, \theta)[e, \sigma\rangle.$$

Proof. Let $U = (B, E, H, g, \mathcal{K}_0)$, and let $C \in \mathcal{C}(\Upsilon), \theta \in \Theta(C), t \in T, \sigma \in \Sigma(t)$.

Let $\mathrm{mark}(C, \theta)[t, \sigma\rangle$, which means

$$\mathit{pre}(t, \sigma) \leq \mathrm{mark}(C, \theta) = \{\!\mid (\pi(b), \theta(\mathbf{v_e}(b))) \mid (b, \theta(\mathbf{v_e}(b))) \in \mathrm{cut}(C, \theta) \mid\!\},$$

Let $B' \subseteq \mathrm{cut}(C)$ be a set of conditions s.t.

$$\mathit{pre}(t, \sigma) = \{\!\mid (\pi(b), \theta(\mathbf{v_e}(b))) \mid b \in B' \mid\!\}.$$

Aiming a contradiction, assume there is *no* $e \in E$ s.t. $\pi(e) = t$ and $\mathrm{cut}(C, \theta)[e, \sigma\rangle$: we extend Υ by such an event. We add to E an event $\widetilde{e}$ with $\pi(\widetilde{e}) = t$ and $g(\widetilde{e}) = g(t)$. Choose for every $b \in B'$ a variable $v^b \in \mathit{Var}$ s.t.

$$\{\!\mid (\pi(b), v^b) \mid b \in B' \mid\!\} = \{\!\mid (p, v) \mid (p, v) \in \mathit{pre}(t) \mid\!\} \quad (= \mathit{pre}(t)).$$

We define $\mathit{pre}(\widetilde{e}) = \{\!\mid (b, v^b) \mid b \in B' \mid\!\}$. Then we have $\{\!\mid (\pi(b), v) \mid (b, v) \in \mathit{pre}(\widetilde{e}) \mid\!\} = \mathit{pre}(t) = \mathit{pre}(\pi(\widetilde{e}))$. For every $(p, v) \in \mathit{post}(t)$, we then add $\mathit{post}(t)(p, v)$ conditions b with $\pi(b) = p$ to B and add $(\widetilde{e}, v, b)$ to H. We thus get $\mathit{post}(\pi(\widetilde{e})) = \{\!\mid (\pi(b), v) \mid (b, v) \in \mathit{post}(\widetilde{e}) \mid\!\}$. We now created a symbolic branching process bigger than Υ, contradicting that Υ is the symbolic unfolding.

Conversely, assume $\exists e \in E : \pi(e) = t \wedge \mathrm{cut}(C, \theta)[e, \sigma\rangle$. Then $\mathit{pre}(e, \sigma) \leq \mathrm{cut}(C, \theta)$, and therefore, $\mathit{pre}(t) = \{\!\mid (\pi(b), v) \mid (b, v) \in \mathit{pre}(e) \mid\!\} \leq \{\!\mid (\pi(b), v) \mid (b, v) \in \mathrm{cut}(C, \theta) \mid\!\} = \mathrm{mark}(C, \theta)$, meaning $\mathrm{mark}(C, \theta)[t, \sigma\rangle$. □

Given a finite configuration C of a symbolic branching process $\beta = (O, h)$, we define $\Uparrow\! C$ as the pair (O', h'), where O' is the unique subnet of O whose set of nodes is $\{x \in B \cup E \mid x \notin (C \cup \to C) \wedge \forall y \in C : \neg(y \sharp x) \wedge (C \cup \{x\}$ is not in color conflict$)\}$ with the set $\mathrm{Cuts}(C)$ of initial cuts, and h' is the restriction of h to the nodes of O'. The branching process $\Uparrow\! C$ is referred to as the *future* of C.

Proposition 4.8 formalizes property (iii) from above.

Proposition 4.8. *If β is a symbolic branching process of $(\mathcal{N}, \mathcal{M}_0)$ and C is a configuration of β, then $\Uparrow\! C$ is a branching process of $(\mathcal{N}, \mathrm{Marks}(C))$. Moreover, if β is the unfolding of $(\mathcal{N}, \mathcal{M}_0)$, then $\Uparrow\! C$ is the unfolding of $(\mathcal{N}, \mathrm{Marks}(C))$.*

Proof. Let $\Uparrow\! C = (O', h')$ with $O' = (B', E', F', g', \mathrm{Cuts}(C))$. To show that O' is an occurrence net, we have to show *i* – *iv* from the definition on p. 28. *i* – *iii* are purely structural properties and follow from the fact that O is an occurrence net. *iv* is satisfied since $\forall b \in \mathrm{cut}(C)\ \forall K \in \mathrm{Cuts}(C) : \sum_{c \in \mathit{Col}} K(b, c) = 1$ and $\forall b \in B' \setminus \mathrm{cut}(C)\ \forall K \in \mathrm{Cuts}(C) : \sum_{c \in \mathit{Col}} K(b, c) = 0$. h' is a homomorphism that is injective on events with the same preset since h is, and that h' is initial follows by Prop. 4.6 and Prop. 4.7.

When β is the symbolic unfolding of $(\mathcal{N}, \mathcal{M}_0)$, then the maximality of $\Uparrow\! C$ follows from the maximality of β, making $\Uparrow\! C$ the symbolic unfolding of $(\mathcal{N}, \mathrm{Marks}(C))$. □

4.2.2 Complete Prefixes

We lift the definition of completeness to the level of symbolic unfoldings. Together with Prop. 4.6 and Prop. 4.7, this can be seen as a direct translation from the low-level case described in Def. 4.1.

Definition 4.9 (Complete symbolic branching process). Let $\beta = (O, h)$ be a symbolic branching process of a high-level Petri net N, with events E'. Then β is called *complete* iff for every reachable marking M in N there exists $C \in \mathcal{C}(\beta)$ and $\theta \in \Theta(C)$ s.t.

i) $M = \text{mark}(C, \theta)$, and

ii) $\forall t \in T \, \forall \sigma \in \Sigma(t) : \; M[t, \sigma\rangle \;\Rightarrow\; \exists e \in E' : h(e) = t \;\wedge\; \text{cut}(C, \theta)[e, \sigma\rangle$. $\lhd$

We now define the class $\mathbf{N_F}$ of high-level Petri nets for which we generalize the construction of finite and complete prefixes of the unfolding of *safe* P/T Petri nets from [ERV02]. We discuss the properties defining this class, and describe how it generalizes safe P/T nets.

Definition 4.10 (Class $\mathbf{N_F}$). The class $\mathbf{N_F}$ contains all finite high-level Petri nets $N = (P, T, F, g, \mathcal{M}_0)$ satisfying the following three properties:

(1) The net is *safe*, i.e., in every reachable marking there lies at most 1 color on every place (formally; $\forall M \in \mathcal{R}(N) \, \forall p \in P : \sum_{c \in Col} M(p, c) \leq 1$).

(2) Guards are written in a decidable theory with the set Col as its domain of discourse.

(3) The net has **f**initely many reachable markings (formally; $|\mathcal{R}(N)| < \infty$). $\lhd$

We require the safety property **(1)** for two reasons; on the one hand, to avoid adding to the already heavy notation. On the other hand, while we think that a generalization to bounded high-level Petri nets is possible, it comes with all the difficulties known from going from safe to k-bounded in the P/T case in [ERV02], plus the problems arising from the expressive power of the high-level formalism. Note that, under the safety condition, we can w.l.o.g. assume $\mathcal{N}$ to be ordinary (i.e., $\forall x, y \in P \cup T : \sum_{v \in Var} F(x, v, y) \leq 1$), since transitions violating this property could never fire. The finiteness of $\mathcal{N}$ additionally implies that we can assume Var to be finite.

While property **(2)** seems very strong, the goal is an algorithm that generates a complete finite prefix of the symbolic unfolding of a given high-level Petri net. The definition of the symbolic unfolding requires the predicate of every event added to the prefix to be satisfiable, and the predicates are build from the guards in the given net. Thus, satisfiability checks in the generation of the prefix seem for now inevitable. An example for such a theory is Presburger arithmetic [Pre30], which is a first-order theory of the natural numbers with addition. The guards in the example from Figure 4.1 are expressible in Presburger arithmetic.

We need property **(3)** to ensure that the generalized version of the cut-off criterion from [ERV02] yields a finite prefix constructed in the generalized ERV-Algorithm. $|\mathcal{R}(N)| < \infty$ can be ensured by having a finite set Col of colors, i.e., $|Col| < \infty$. In

Sec. 4.4, we identify a class of high-level Petri nets with infinitely many reachable markings for which the algorithm is sufficient when adapting the cut-off criterion.

Under these three assumptions we generalize the finite safe P/T Petri nets considered in [ERV02]: every such P/T net can be seen as a high-level Petri net with $Col = \{\bullet\}$ and all guards being *true*, and thus satisfying the three properties above. Replacing the safety property **(1)** by a respective "k-bounded property" would result in a generalization of k-bounded P/T nets. In Sec. 4.3, we compare the result of the generalized ERV-algorithm, Alg. 1, applied to a high-level net to the result of the original ERV-algorithm from [ERV02] applied to the high-level net's expansion.

For the rest of the section, let $N = (P, T, F, g, \mathcal{M}_0) \in \mathbf{N_F}$ with symbolic unfolding $\Upsilon = (U, \pi) = (B, E, H, g, \mathcal{K}_0, \pi)$.

Since we consider safe high-level Petri nets, we can relate two cuts of configurations that represent the same set of places in the following way:

Definition 4.11. Let $C_1, C_2 \in \mathcal{C}(\Upsilon)$ with $\pi(\text{cut}(C_1)) = \pi(\text{cut}(C_2))$. Then there is a unique bijection $\phi : \text{cut}(C_1) \to \text{cut}(C_2)$ preserving π. We call this mapping $\phi_{C_1}^{C_2}$. $\lhd$

4.2.3 Generalizing Adequate Orders and Cut-Off Events

We lift the concept of adequate orders on the configurations of an occurrence net to the level of symbolic unfoldings. A main property of adequate orders is the preservation by finite *extensions*, which in the high-level case are defined as for P/T-nets (cp. Sec. 4.1):

For a (high-level) configuration C, if $C \cup Q$ is a configuration and $C \cap Q = \emptyset$ then we denote $C \cup Q$ by $C \oplus Q$. We say that $C \oplus Q$ is an *extension* of C, and that Q is a *suffix* of C. Obviously, for a configuration C', if $C \subsetneq C'$ then there is a nonempty suffix Q of C such that $C \oplus Q = C'$. For a configuration $C \oplus Q$, denote by $O(C|Q) = (\text{cut}(C) \cup {\to}Q \cup Q{\to}, Q, H', \text{Cuts}(C))$ the occurrence net around Q from $\text{cut}(C)$, where H' is the restriction of H to the nodes of $O(C|Q)$. Note that for every finite configuration C with an extension $C \oplus Q$, we have that Q is a configuration of $\Uparrow C$.

For better readability, we introduce for a marking M the following notation:

$$C[\![M]\!]Q \quad :\Leftrightarrow \quad \exists \theta \in \Theta(C \oplus Q) : \text{mark}(C, \theta|_{Var_{C \cup \{\perp\}}}) = M. \tag{4.1}$$

Thus, $C[\![M]\!]Q$ means that we can fire the events in $C \oplus Q$ in modes (formalized by the instantiation θ) such that after firing only the events in C we arrive at a cut representing M.

In [ERV02], the authors infer from the low-level version of Prop. 4.8 that if the cuts of two low-level configurations represent the same marking in the low-level net, then their futures are isomorphic. A respective (unique) isomorphism I_1^2 (cp. Sec. 4.1) maps the suffixes of one configuration to the suffixes of the other.

This holds also in the high-level case: if two cuts represent the same *set* of markings $(\text{Marks}(C_1) = \text{Marks}(C_2))$ then there exists an isomorphism I_1^2 between the two futures C_1 and C_2. This isomorphism maps suffixes of C_1 to suffixes of C_2.

The following Prop. 4.12 is a "weak" version of this argument. It is "weaker" in the sense that we do not assume that two high-level configurations represent the *same* set of markings. We only assume that there is a marking represented by both configurations ($M \in \text{Marks}(C_1) \cap \text{Marks}(C_2)$) from which we can fire a fixed suffix Q of C_1 (notation $C_1[\![M]\!]Q$ from (4.1)). Weakening the assumption leads not to an isomorphism between the two futures but only a monomorphism $\varphi^2_{1,Q}$, which in addition is dependent on the suffix Q under consideration.

Proposition 4.12. *Let C_1 and C_2 be two finite configurations in Υ, and let Q be a suffix of C_1. If there is a marking $M \in \text{Marks}(C_1) \cap \text{Marks}(C_2)$ s.t. $C_1[\![M]\!]Q$, then there is a unique monomorphism $\varphi^2_{1,Q} : O(C_1|Q) \to \Uparrow C_2$ that satisfies $\varphi^2_{1,Q}(\text{cut}(C_1)) = \text{cut}(C_2)$ and preserves the labeling π.*
For this monomorphism we have that $\varphi^2_{1,Q}(Q)$ is a suffix of C_2.

Notation. For functions $f : X \to Y$ and $f' : X' \to Y$ with $X \cap X' = \emptyset$ we define $f \sqcup f' : X \sqcup X' \to Y$ by mapping x to $f(x)$ if $x \in X$ and to $f'(x)$ if $x \in X'$.

Proof. By induction over the size $k = |Q|$ of the suffix Q.

Base case $k = 0$. This means $Q = \emptyset$. Then $O(C_1|Q) = (\text{cut}(C_1), \emptyset, \emptyset, \text{Cuts}(C_1))$. Since $M \in \text{Marks}(C_1) \cap \text{Marks}(C_2)$, we know that $\pi(\text{cut}(C_1)) = \pi(\text{cut}(C_2))$. Since we only consider safe nets, $\varphi^2_{1,Q}$ is uniquely realized by $\phi^{C_2}_{C_1} : \text{cut}(C_1) \to \text{cut}(C_2)$ from Def. 4.11.

Induction step. Let $k > 0$. Let $\theta \in \Theta(C_1 \oplus Q)$ s.t. $\text{mark}(C_1, \theta|_{\mathit{Var}_{C_1 \cup \{\perp\}}}) = M$. Let $e \in \mathit{Min}(Q)$. Then for $\sigma = [v \leftarrow \theta(v_e)]_{v \in \mathit{Var}(e)}$ we have $M[\pi(e), \sigma\rangle$. Thus, by Prop. 4.7, $\exists e' \in E : \pi(e') = \pi(e) \wedge C_2 \oplus \{e'\} \in \mathcal{C}(\Upsilon)$. This means $\to e' \subseteq (B_0 \cup (C_2 \to)) \setminus (\to C_2)$; else, $C_2 \oplus \{e'\}$ would not be a configuration. Thus, e' is an event in $\Uparrow C_2$. Since $\pi(e) = \pi(e')$, we get by definition of homomorphisms that $\{(\pi(b), v) \mid (b, v) \in \mathit{post}(e)\} = \{(\pi(b), v) \mid (b, v) \in \mathit{post}(e')\}$. The net N is safe, therefore we can define the bijection $\phi_1 : (e \to) \to (e' \to)$ by $\phi_1(b) = b' \Leftrightarrow \pi(b) = \pi(b')$. We now define $\varphi_1 : O(C_1|\{e\}) \to \Uparrow C_2$ by $\varphi_1 = \phi^{C_2}_{C_1} \sqcup \{e \mapsto e'\} \sqcup \phi_1$, which is a homomorphism satisfying the claimed conditions.

Let now $C'_1 = C_1 \cup \{e\}$, $C'_2 = C_2 \cup \{\varphi_1(e)\}$ and $Q' = Q \setminus \{e\}$. We then have for M' given by $M[\pi(e), \sigma\rangle M'$ that $C'_1[\![M']\!]Q'$, $M' \in \text{Marks}(C'_1) \cap \text{Marks}(C'_2)$, and $|Q'| < k$. Thus, by the induction hypothesis, we get that there is a unique monomorphism $\varphi_2 : O(C'_1|Q') \to \Uparrow C'_2$ satisfying the conditions above. Since φ_1 and φ_2 coincide on $\text{cut}(C'_1)$, we can now define $\varphi^2_{1,Q}$ by "gluing together" φ_1 and φ_2 at $\text{cut}(C'_1)$.

This proves the claim for finite extensions. For an infinite extension, every node also contained in a finite extension. Due to uniqueness of the homomorphisms, we can define the $\varphi^2_{1,Q}$ in the case of an infinite Q as the union of all homomorphisms of smaller finite extensions. □

Equipped with Prop. 4.12, we now lift the concept of adequate order to the level of symbolic branching processes. Compared to Def. 4.2, the monomorphism $\varphi^2_{1,Q}$ defined above replaces the isomorphism I^2_1 between $\Uparrow \mathsf{C}_1$ and $\Uparrow \mathsf{C}_2$ for two low-level configurations $\mathsf{C}_1, \mathsf{C}_2$ representing the same marking.

Definition 4.13 (Adequate order, high-level). A partial order $\prec$ on the finite configurations of the symbolic unfolding of a set of high-level Petri net is an *adequate order* if:

i) $\prec$ is well-founded,

ii) $C_1 \subset C_2$ implies $C_1 \prec C_2$, and

iii) $\prec$ is preserved by finite extensions in the following way: if C_1, C_2 are two finite configurations, and $C_1 \oplus Q$ is a finite extension of C_1 such that there is a marking $M \in \text{Marks}(C_1) \cap \text{Marks}(C_2)$ satisfying $C_1[\![M]\!]Q$, then the monomorphism $\varphi^2_{1,Q}$ from above satisfies $C_1 \prec C_2 \Rightarrow C_1 \oplus Q \prec C_2 \oplus \varphi^2_{1,Q}(Q)$. $\lhd$

In the case of a P/T net interpreted as a high-level net, we have $|\,\text{Marks}(C)| = 1$ for every configuration C, and therefore, Def. 4.13 coincides with its P/T version Def. 4.2.

We could alternatively generalize the P/T case by replacing '$\exists M \in \text{Marks}(C_1) \cap \text{Marks}(C_2)$ s.t. $C_1[\![M]\!]Q$' by '$\text{Marks}(C_1) = \text{Marks}(C_2)$'. Then there exists, as in the P/T case, the isomorphism I^2_1 between $\Uparrow C_1$ and $\Uparrow C_2$, which we could use to define preservation by finite extension. However, in the upcoming generalization of the ERV-algorithm from [ERV02], the generalized cut-off criterion exploits property iii) of adequate orders. Using '$\text{Marks}(C_1) = \text{Marks}(C_2)$' would produce an exponential blowup of the generated prefix's size. This is circumvented by using '$\exists M \in \text{Marks}(C_1) \cap \text{Marks}(C_2)$ s.t. $C_1[\![M]\!]Q$', which however leads to obtaining merely a monomorphism $\varphi^2_{1,Q}$ that depends on the considered suffix Q, instead of an isomorphism between the futures. We now show that this monomorphism sufficient.

The proof that the generalized ERV-algorithm is complete will be structurally analogous to the respective proof in [ERV02]. It uses that, under the conditions of Def. 4.13 iii), we also have $C_2 \prec C_1 \Rightarrow C_2 \oplus \varphi^2_{1,Q}(Q) \prec C_1 \oplus Q$. This result would directly be obtained if $\varphi^2_{1,Q}$ was an isomorphism, as I^2_1 is. However, a monomorphism is an isomorphism when its codomain is restricted to its range. This idea is used in the proof of the following proposition, which states that $\varphi^2_{1,Q}$ indeed satisfies the above property.

Proposition 4.14. *Let $\prec$ be an adequate order. Under the conditions of Def. 4.13 iii) the monomorphism $\varphi^2_{1,Q}$ also satisfies $C_2 \prec C_1 \Rightarrow C_2 \oplus \varphi^2_{1,Q}(Q) \prec C_1 \oplus Q$.*

Proof. Let $Q' = \varphi^2_{1,Q}(Q)$. We first show that $\varphi^1_{2,Q'}(Q') = Q$.

Let $\varphi_1 : O(C_1|Q) \to \varphi^2_{1,Q}(O(C_1|Q))$ be the isomorphism that acts on $O(C_1|Q)$ as $\varphi^2_{1,Q}$ does, and let $\varphi_2 : O(C_2|Q') \to \varphi^1_{2,Q'}(O(C_2|Q'))$ be the isomorphism that acts on $O(C_2|Q')$ as $\varphi^1_{2,Q'}$ does. Since $\varphi_1^{-1} : \varphi^2_{1,Q}(O(C_1|Q)) \to O(C_1|Q)$ and $O(C_1|Q) \subset \Uparrow C_1$, and $\varphi_1^{-1}(\varphi^2_{1,Q}(Q)) = Q$ is a suffix of C_1, we get by Prop. 4.12 that $\varphi_1^{-1} = \varphi_2$, which means $\varphi^1_{2,Q'}(Q') = Q$.

Assume now $C_2 \prec C_1$. From the proof of Prop. 4.12 we see that $C_2[\![M]\!]\varphi^2_{1,Q}(Q)$. Thus, we get by the definition of adequate order and the result above that $C_2 \oplus \varphi^2_{1,Q}(Q) \prec C_1 \oplus \varphi^1_{2,\varphi^2_{1,Q}(Q)}(\varphi^2_{1,Q}(Q)) = C_1 \oplus Q$ □

In [ERV02], three adequate orders on the configurations of the low-level unfolding are discussed. In particular, the authors present a *total* adequate order that uses the *Foata normal form* of configurations. Using such a total order in the algorithm limits the size of the resulting finite and complete prefix; It contains at most $|\mathcal{R}(N)|$ non cut-off events. All three adequate orders presented in [ERV02] can be directly lifted to the configurations of the symbolic unfolding by exchanging every low-level term by its high-level counterpart. The lifted order using the Foata normal form is still a total order. We include these discussions in Appendix 4.A.1.

We now define cut-off events in a symbolic unfolding. In the low-level case Def. 4.3, e is a cut-off event iff there is another event e' satisfying $[\mathsf{e}'] \prec [\mathsf{e}]$ and $\mathrm{mark}([\mathsf{e}]) = \mathrm{mark}([\mathsf{e}'])$, which ensures that the future of e needs not be considered further. In the high-level case, we generalize these conditions to high-level events e. However, we do not require the existence of *one* other high-level event e' with $[e'] \prec [e]$ and $\mathrm{Marks}([e]) = \mathrm{Marks}([e'])$. While this would still be a valid cut-off criterion and would lead to finite and complete prefixes, the upper bound on the size of such a prefix would be exponential in the number of markings in the original net, since we must argue on the power set of N's reachable markings.

Instead, we check whether $\mathrm{Marks}([e])$ is contained in the union of *all* $\mathrm{Marks}([e'])$ with $[e'] \prec [e]$. This criterion expresses that we have already seen every marking in $\mathrm{Marks}([e])$ in the prefix β under construction, and therefore need not consider the future of e any further. By this, we obtain the same upper bounds on the size of the produced prefix as in [ERV02], as discussed later.

Definition 4.15 (Cut-off event, high-level). Let $\prec$ be an adequate order on the configurations of the symbolic unfolding of a high-level Petri net. Let β be a prefix of the symbolic unfolding containing a high-level event e. The high-level event e is a *cut-off* event in β (w.r.t. $\prec$) if $\mathrm{Marks}([e]) \subseteq \bigcup_{[e'] \prec [e]} \mathrm{Marks}([e'])$. $\lhd$

When interpreting P/T nets as high-level nets, this definition corresponds to the cut-off events defined in Def. 4.3, since then $|\mathrm{Marks}([e])| = 1$ for all events e, and a singleton is contained in a union of other singletons iff its element is equal to one of the other's.

4.2.4 The Generalized ERV-Algorithm

We present the algorithm for constructing a finite and complete prefix of the symbolic unfolding of a given high-level Petri net. It is a generalization of the ERV-algorithm from [ERV02], and is structurally equal (and therefore looks very similar). However, the algorithm is contingent upon the previous section's work of generalizing adequate orders and cut-off events, which ultimately enables us to adopt this structure.

A crucial concept of the ERV-algorithm is the notion of "possible extensions", i.e., the set of individual events that extend a given prefix of the unfolding. In Def. 4.16, we lift this concept to the high-level formalism. We do so by isolating the procedure of adding high-level events in the algorithm from [CJ04] which generates the whole symbolic unfolding of a given high-level Petri net (but does not terminate if the symbolic unfolding is infinite).

We define the data structures similarly to [ERV02]. There, an event is given by a tuple $\mathsf{e} = (\mathsf{t}, \mathsf{B}')$ with $\mathsf{h}(\mathsf{e}) = \mathsf{t} \in \mathsf{T}$ and $\mathit{pre}(\mathsf{e}) = \mathsf{B}' \subseteq \mathsf{B}$, and a condition given by a tuple

$\mathsf{b} = (\mathsf{p}, \mathsf{e})$ with $\mathsf{h}(\mathsf{b}) = \mathsf{p} \in \mathsf{P}$ and $pre(\mathsf{b}) = \{\mathsf{e}\} \subseteq \mathsf{E}$. The finite and complete prefix is a set of such events and transitions.

In the high-level case, we need more information inside the tuples. A high-level event is given by a tuple $e = (t, X, pred)$ described by $h(e) = t$, $pre(e) = X \subseteq B \times Var$, and $pred(e) = pred$. Analogously, a high-level condition is given by a tuple $b = (p, (e, v), pred)$, where $h(b) = p$, $pre(b) = (e, v) \in (E \times Var) \cup (\{\perp\} \times \{v^b \mid b \in B_0\})$, and $pred(\mathbf{e}(b)) = pred$.

Definition 4.16 (Possible extensions). Let $\beta = (O, h)$ be a branching process of a high-level Petri net N. The *possible extensions* $PE(\beta)$ are the set of tuples $e = (t, X, pred)$ where t is a transition of N, and $X \subseteq B \times Var$ satisfying

- $\{b \mid (b, v) \in X\}$ is a co-set, and $pre(t) = \{(h(b), v) \mid (b, v) \in X\}$,
- $pred = loc\text{-}pred \wedge \big(\bigwedge_{(b,v)\in X} pred(\mathbf{e}(b))\big)$ is satisfiable, where $loc\text{-}pred = g(t)[v \leftarrow v_e]_{v \in Var(t)} \wedge \big(\bigwedge_{(b,v)\in X} v_e = \mathbf{v_e}(b)\big)$,
- β does not contain $(t, X, pred)$. $\lhd$

Since this is the isolated procedure of adding high-level events in the algorithm from [CJ04] which generates the complete symbolic unfolding of a given high-level Petri net, we know that the set $PE(\beta)$ precisely contains the events that can be added to the prefix.

The notion of co-set in high-level occurrence nets is achieved by the direct translation from low-level occurrence nets plus the "color conflict freedom". Thus, possible extensions in a prefix β can be found by searching first for sets of conditions that are not in structural conflict as in the low-level case, and then checking whether these sets are in color conflict.

Alg. 1 is a generalization of the ERV-Algorithm in [ERV02] for complete finite prefixes of the low-level unfolding. The structure is taken from there, with the only difference being the special event $\perp$. The algorithm takes as input a high-level Petri net $N \in \mathbf{N_F}$ and assumes a given adequate order $\prec$.

Example 4.17. Consider the running example Three Times Termination from Figure 4.1. Alg. 1 produces the complete finite prefix marked by the blue lines in Fig. 4.3. Cut-off events are again shaded blue. We demonstrate this in detail:

Starting with the initial conditions a' and b', the possible extensions are α' and β'. assuming $[\alpha'] \prec [\beta']$, we first add α' together with a condition c' corresponding to the output place c of α, and then analogously add β' and the condition d'.

For α' we have $\text{Marks}([\alpha']) = \{\{\!|\, (c, k), (b, 0) \,|\!\} \mid k \in \{1, \ldots, m\}\}$ and analogously, for β' we have $\text{Marks}([\beta']) = \{\{\!|\, (a, 0), (d, k) \,|\!\} \mid k \in \{1, \ldots, m\}\}$. Since we have not seen these markings before, neither α' nor β' are cut-off events. Thus, we have the possible extensions t' and ε'. For t' we have $\text{Marks}([t']) = \{\{\!|\,|\!\}\}$, since no tokens are in the net after firing t. However, we have not seen the empty marking $\{\!|\,|\!\}$ before, so formally, t' is not a cut-off event.

For ε' we have $\text{Marks}([\varepsilon']) = \{\{\!|\, (c, k), (d, \ell) \,|\!\} \mid k, \ell \in \{1, \ldots, m\}\}$. Corresponding cuts can be reached in the prefix constructed so far by concurrently firing α' and β'. However, no marking $\{\!|\, (c, k), (d, \ell) \,|\!\}$ is represented by a *cone* configuration $[e']$ before

Algorithm 1: Generalization of the ERV-Algorithm from [ERV02] for complete finite prefixes.

```
Data: High-level Petri net $N = (P, T, F, g, \mathcal{M}_0) \in \mathbf{N_F}$.
Result: A complete finite prefix $Fin$ of the symbolic unfolding of $N$.
$Fin := \{\bot\}$;
$pred(\bot) := \bigvee_{M_0 \in \mathcal{M}_0} \bigwedge_{(p,c) \in M_0} v_\bot^{b_p} = c$;
foreach $p \in P_0$ do
    Create a fresh condition $b_p := (p, (\bot, v^{b_p}), pred(\bot))$;
    $Fin := Fin \cup \{b_p\}$;
$pe := PE(Fin)$;
$cut\text{-}off := \emptyset$;
while $pe \neq \emptyset$ do
    Pick $e = (t, X, pred)$ from $pe$ such that $[e]$ is minimal w.r.t. $\prec$;
    if $[e] \cap cut\text{-}off = \emptyset$ then
        $Fin := Fin \cup \{e\}$;
        foreach $(p, v) \in post(t)$ do
            Create a fresh condition $b := (p, (e, v), pred)$;
            $Fin := Fin \cup \{b\}$;
        $pe := PE(Fin)$;
        if $e$ is a cut-off event of $Fin$ then
            $cut\text{-}off := cut\text{-}off \cup \{e\}$;
    else
        $pe := pe \setminus \{e\}$
```

$[\varepsilon']$, and thus ε' does *not* satisfy $\text{Marks}([\varepsilon']) \subseteq \bigcup_{[e'] \prec [\varepsilon]} \text{Marks}([e'])$. This means ε' is not a cut-off event and we have to proceed with the possible extensions t'' and ε''.

Since $\text{Marks}([t'']) = \{\{\!\!\{ \}\!\!\}\} = \text{Marks}([t'])$ with $[t'] \prec [t'']$, have that t'' is a cut-off event. This, however, has no impact on the prefix since we cannot continue after t'' anyway. For ε'' we have $\text{Marks}([\varepsilon'']) = \{\{\!\!\{ (c, k), (d, \ell) \}\!\!\} \mid k, \ell \in \{1, \dots, m\}\} = \text{Marks}([\varepsilon'])$ with $[\varepsilon'] \prec [\varepsilon'']$. This makes ε'' also a cut-off event. We therefore have no more possible extensions, and the algorithm terminates. In the figure, this is indicated by the blue lines.

Note that in this special case, for both cut-off events ($e = \varepsilon''$ or $e = t''$), we had that there was an event e' with $[e'] \prec [e]$ and $\text{Marks}([e]) = \text{Marks}([e'])$. The definition of cut-off event from Def. 4.15 is more general. $\lhd$

We now prove correctness of Alg. 1 analogously to [ERV02], by stating two propositions – one each to show that the output of the algorithm, i.e., the prefix *Fin*, is finite and complete, respectively. The proof structure is also as in [ERV02], but adapted to the setting of high-level Petri nets and symbolic unfoldings.

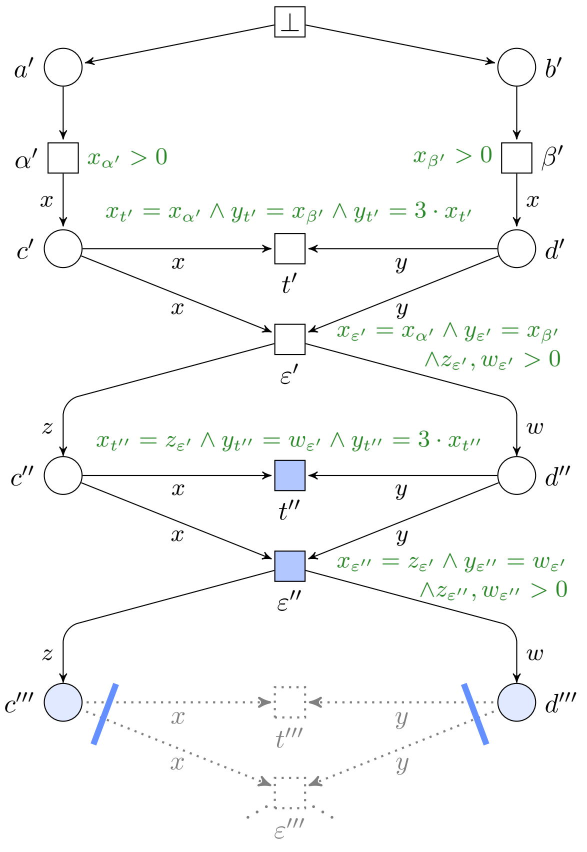

⊥
a'
b'
α'
$x_{\alpha'} > 0$
$x_{\beta'} > 0$
β'
x
$x_{t'} = x_{\alpha'} \land y_{t'} = x_{\beta'} \land y_{t'} = 3 \cdot x_{t'}$
c'
d'
t'
y
$x_{\varepsilon'} = x_{\alpha'} \land y_{\varepsilon'} = x_{\beta'}$
$\land z_{\varepsilon'}, w_{\varepsilon'} > 0$
ε'
z
w
$x_{t''} = z_{\varepsilon'} \land y_{t''} = w_{\varepsilon'} \land y_{t''} = 3 \cdot x_{t''}$
c''
d''
t''
$x_{\varepsilon''} = z_{\varepsilon'} \land y_{\varepsilon''} = w_{\varepsilon'}$
$\land z_{\varepsilon''}, w_{\varepsilon''} > 0$
ε''
c'''
d'''
t'''
ε'''

Figure 4.3: The symbolic unfolding $\Upsilon(N)$ of the net N in Fig. 4.1

Proposition 4.18. *Fin is finite.*

Given an event e, define the *depth* of e as the length of the longest chain of events $e_1 < e_2 < \cdots < e$; the depth of e is denoted by $d(e)$.

Proof. As in [ERV02], we prove the following results (1) – (3):

(1) For every event e of *Fin*, $d(e) \leq |\mathcal{R}(N)| + 1$,

(2) For every event e of *Fin*, the sets *pre*(e) and *post*(e) are finite, and

(3) For every $k \geq 0$, *Fin* contains only finitely many events e such that $d(e) \leq k$.

This works exactly as in [ERV02], with minor adaptations to the generalization of cut-offs in the symbolic unfolding in (1):

(1) Let $n = |\mathcal{R}(N)|$. Every chain of events $e_1 < e_2 < \cdots < e_n < e_{n+1}$ in the unfolding contains an event e_i, $i > 1$, s.t. $\text{Marks}([e_i]) \subseteq \bigcup_{j=1}^{i-1} \text{Marks}([e_j])$, since, if every $\text{Marks}([e_j])$, $j = 1, \ldots, n$, contains a marking not contained in $\bigcup_{k=1}^{j-1} \text{Marks}([e_k])$, then finally $\bigcup_{j=1}^{n} \text{Marks}([e_j])$ contains all n markings. This makes e_{n+1} a cut-off event.

(2) By the construction in the algorithm we see that there is a bijection between $\mathit{post}(e)$ and $\mathit{post}(h(e))$, and similarly for $\mathit{pre}(e)$ and $\mathit{pre}(h(e))$. The result then follows from the finiteness of N.

(3) By complete induction on k. The base case, $k = 0$, is trivial. Let E_k be the set of events of depth at most k. We prove that if E_k is finite then E_{k+1} is finite. By (2) and the induction hypothesis, $\mathit{post}(E_k)$ is finite. Since $\{b \mid \exists v \in \mathit{Var} : (b, v) \in \mathit{pre}(E_{k+1})\} \subseteq \{b \mid \exists v \in \mathit{Var} : (b, v) \in \mathit{post}(E_k)\}$, we get by property *iv* in the definition of occurrence nets that E_{k+1} is finite. □

Proposition 4.19. *Fin is complete.*

The proof of this proposition also has the same general structure as the respective proof in [ERV02]. However here we use the generalizations of adequate order, possible extensions, and the cut-off criterion to symbolic branching processes.

Proof. We first show that for every reachable marking in N there exists a configuration in Υ satisfying a) from the definition of complete prefixes, and then show that one of these configurations (a minimal one) also satisfies b).

(1) Let M be an arbitrary reachable marking in N. Then by Prop. 4.6, we have that there is a $C_1 \in \mathcal{C}(\Upsilon)$ s.t. $M \in \text{Marks}(C_1)$. Let $\theta_1 \in \Theta(C_1)$ s.t. $M = \text{mark}(C_1.\theta_1)$. If C is not a configuration in *Fin*, then it contains a cut-off event e_1, and so $C_1 = [e_1]\oplus Q$ for some set Q of events. Let $M_1 = \text{mark}([e_1].\theta_1|_{\mathit{Var}_{[e_1]\cup\{\perp\}}}) \in \text{Marks}([e_1])$. By the definition of cut-off event, there exists an event e_2 with $[e_2] \prec [e_1]$ and $M_1 \in \text{Marks}([e_2])$. Since we have $C_1[\![M_1]\!]Q$, we get by Prop. 4.12 that the monomorphism $\varphi_1 := \varphi^{[e_2]}_{[e_1],Q} : O([e_1]|Q) \to \Uparrow[e_2]$ exists and that $\varphi_1(Q)$ is a suffix of $[e_2]$. By Prop. 4.14 we know

$$C_2 := [e_2]\oplus\varphi_1(Q) \prec [e_1]\oplus Q = C_1.$$

Let $\theta'_2 \in \Theta([e_2])$ s.t. $M_1 = \text{mark}([e_2], \theta'_2)$. Define now $\theta_2 \in \Theta(C_2)$ by $\theta_2 = \theta'_2 \sqcup \theta''_2$, where $\theta''_2 : \mathit{Var}_{\varphi_1(Q)} \to \mathit{Col}$ is given by $\theta''_2(v_{\varphi_1(e)}) = \theta_1(v_e)$. By this construction we get $M = \text{mark}(C_2, \theta_2) \in \text{Marks}(C_2)$.

If C_2 is not a configuration of *Fin*, then we can iterate the procedure and find a configuration C_3 such that $C_3 \prec C_2$ and $M \in \text{Marks}(C_3)$. The procedure cannot be iterated infinitely often because $\prec$ is well-founded. Therefore, it terminates in a configuration of *Fin*.

(2) Let now C be a minimal configuration w.r.t. $\prec$ s.t. $M \in \text{Marks}(C)$, and let $t \in T$, $\sigma \in \Sigma(t)$ s.t. $M[t, \sigma\rangle$. If C contains some cut-off event, then we can apply the arguments of a) to conclude that *Fin* contains a configuration $C' \prec C$ such that $M \in \text{Marks}(C')$. This contradicts the minimality of C. So C contains no cut-off events. Let $\theta \in \Theta(C)$ s.t. $M = \text{mark}(C, \theta)$. Since $\mathit{pre}(t.\sigma) \subseteq M$, we have that there is a co-set $B_{t,\sigma} \subseteq \text{cut}(C)$ s.t. $\mathit{pre}(t, \sigma) = \{(h(b), \theta(\mathbf{v_e}(b))) \mid b \in B_{t,\sigma}\}$. Let now $X := \{(b, v) \mid b \in B_{t,\sigma}, (h(b), v) \in \mathit{pre}(t)\}$. We then have $\forall(b, v) \in X : \sigma(v) = \theta(\mathbf{v_e}(b))$.

We now show that

$$pred := g(t)[v \leftarrow v_e]_{v \in Var(e)} \wedge \Big(\bigwedge_{(b,v) \in X} v_e = \mathbf{v_e}(b) \Big) \wedge \bigwedge_{(b,v) \in X} pred(\mathbf{e}(b))$$

is satisfiable. Let $\theta' := \theta \sqcup (\sigma \circ \{v_e \mapsto v \mid v \in Var(e)\})$. Then

- $g(t)[v \leftarrow v_e]_{v \in Var(e)}[\theta'] \equiv g(t)[\sigma] \equiv true$, and
- $\big(\bigwedge_{(b,v) \in X} v_e = \mathbf{v_e}(b)\big)[\theta'] \equiv \big(\bigwedge_{(b,v) \in X} \sigma(v) = \theta(\mathbf{v_e}(b))\big) \equiv true$, and
- $\bigwedge_{(b,v) \in X} pred(\mathbf{e}(b))[\theta'] \equiv \bigwedge_{(b,v) \in X} pred(\mathbf{e}(b))[\theta] \equiv true$, since $\theta \in \Theta(C)$.

Thus, $pred[\theta'] \equiv true$. Therefore, $e = (t, X, pred)$ is a possible extension and added in the execution of the algorithm. Then we directly have $e \notin C$, $h(e) = t$, and with the same arguments as in a), we get $C \cup \{e\} \in \mathcal{C}(Fin)$ and $\theta \sqcup (\sigma \circ \{v_e \mapsto v \mid v \in Var(e)\}) \in \Theta(C \cup \{e\})$, which means $\mathrm{cut}(C, \theta)[e, \sigma\rangle$. Since we chose θ independently of t and σ, this concludes the proof. $\square$

Notice that by this construction, as described in [ERV02], we get that if $\prec$ is a total order, then *Fin* contains at most $|\mathcal{R}(N)|$ non cut-off events. As mentioned in Sec. 4.2.3, the total adequate order presented in [ERV02] can be lifted to the configurations in the symbolic unfolding, where it again is total (cp. Appendix 4.A.1). Thus, we generalized the possibility to construct such a small complete finite prefix by application of Alg. 1 with $\prec$ being a total adequate order.

4.3 High-Level versus P/T Expansion

In this section we state in Lemma 4.21 that the expansion of a finite complete prefix of the unfolding of a high-level Petri net is a finite and complete prefix of the unfolding of the expanded high-level Petri net. This means the generalization of complete prefixes is "canonical", and compatible with the established low-level concepts. We then compare for our running example the results of

- applying the generalized ERV-algorithm Alg. 1 to obtain a complete finite prefix of the symbolic unfolding of a given high-level Petri net, and
- first expanding a given high-level Petri net and then applying the ERV-algorithm from [ERV02] for a complete finite prefix of the (P/T) unfolding.

Recall the notation for P/T Petri nets, branching processes, and unfoldings from Sec. 2.1, as well as expansions from Sec. 2.3.1. With this, we can define for a high-level occurrence net O the P/T occurrence net $\mathrm{Exp}_{\mathrm{O}}(O) := \mathsf{U}(\mathrm{Exp}(O))$, i.e., the occurrence net from the unfolding $\Upsilon(\mathrm{Exp}(O))$, which is given by $(\mathsf{U}(\mathrm{Exp}(O)), \pi^{\mathrm{Exp}(O)})$. We abbreviate $\pi^{\mathrm{Exp}(O)}$ by π^{O}. The operator $\mathrm{Exp}_{\mathrm{O}}$ therefore maps high-level occurrence nets to occurrence nets (cf. [CF10]). Let now $\beta = (O, h)$ be a symbolic branching process of N. Then we can define the *expanded symbolic branching process* $\mathrm{Exp}_{\mathrm{O}}(\beta) := (\mathrm{Exp}_{\mathrm{O}}(O), \mathsf{h})$ of $\mathrm{Exp}(N)$ with the homomorphism $\mathsf{h} : \mathrm{Exp}_{\mathrm{O}}(O) \to \mathrm{Exp}(N)$, defined by $\mathsf{h}(\mathsf{e}) = t.\sigma \overset{df}{\Leftrightarrow}$

$\pi^O(\mathsf{e}) = e.\sigma \wedge h(e) = t$ and $\mathsf{h}(\mathsf{b}) = p.c \overset{df}{\Leftrightarrow} \pi^O(\mathsf{b}) = b.c \wedge h(b) = p$ for events e resp. conditions b in $\mathrm{Exp}_\mathrm{O}(O)$. The following diagram serves as an overview:

$$\begin{tikzcd}
N \arrow[dd, mapsto, "\mathrm{Exp}"'] & & O \arrow[ll, "h"'] \arrow[dl, mapsto, "\mathrm{Exp}"] \arrow[dd, mapsto, "\mathrm{Exp}_\mathrm{O}"] & & \beta = (O,h) \arrow[dd, mapsto, "\mathrm{Exp}_\mathrm{O}"] \\
& \mathrm{Exp}(O) & & \rightsquigarrow & \\
\mathrm{Exp}(N) & & \mathrm{Exp}_\mathrm{O}(O) = \mathsf{U}(\mathrm{Exp}(O)) \arrow[ll, "\mathsf{h}"] \arrow[ul, "\pi^O"'] & & \mathrm{Exp}_\mathrm{O}(\beta) = (\mathrm{Exp}_\mathrm{O}(O), \mathsf{h})
\end{tikzcd}$$

The following result is shown in [CF10]. It states that, for a high-level Petri net N, the unfolding of N's expansion is isomorphic to the expanded symbolic unfolding of N.

Lemma 4.20 ([CF10], Sec. 4.1). $\Upsilon(\mathrm{Exp}(N)) \simeq \mathrm{Exp}_\mathrm{O}(\Upsilon(N))$.

With this result, we state the following:

Lemma 4.21. *Let N be a high-level Petri net and β be a prefix of $\Upsilon(N)$. Then β is finite and complete if and only if* $\mathrm{Exp}_\mathrm{O}(\beta)$ *is a finite and complete prefix of* $\Upsilon(\mathrm{Exp}(N))$.

The proof uses the results from Prop. 4.6 and Prop. 4.7, since the definition of completeness on the symbolic level is a direct translation from its P/T analogue.

Proof. Let $\beta = (O, h)$ be finite and complete. From Lemma 4.20 we already know that $\mathrm{Exp}_\mathrm{O}(O) \subseteq \mathsf{U}(\mathrm{Exp}(N))$. Since $\mathrm{Exp}_\mathrm{O}(\beta)$ is a branching process of $\mathrm{Exp}(N)$, we see that is a prefix of the unfolding of $\mathrm{Exp}(N)$. Also, $\mathrm{Exp}_\mathrm{O}(\beta)$ is obviously finite since O is a finite high-level occurrence net.

We now prove that $\mathrm{Exp}_\mathrm{O}(\beta) = (\mathrm{Exp}_\mathrm{O}(O), \mathsf{h})$ is complete. Let M be a reachable marking in $\mathrm{Exp}(N)$. Then the high-level marking M defined by $M(p,c) = \mathsf{M}(p.c)$ is reachable in N. Thus, since β is complete, there is a configuration $C \in \mathcal{C}(\beta)$ and an instantiation $\theta \in \Theta(C)$ satisfying a) and b) from Def. 4.9. This means there is a firing sequence $K_0[e_1, \sigma_1\rangle K_1 \ldots [e_n, \sigma_n\rangle K_n$ with $\{e_1, \ldots, e_n\} = C$, $\sigma_i = \theta \circ [v \mapsto v_{e_i}]_{v \in Var(e_i)}$, and $K_n = \mathrm{cut}(C, \theta)$ (meaning $M = \mathrm{mark}(C, \theta) = \{\!|\, (h(b), c) \mid (b, c) \in K_n \,|\!\}$). Then, in $\mathrm{Exp}(O)$, the marking $\{\!|\, b.c \mid (b, c) \in K_n \,|\!\}$ is reachable from the initial marking $\{\!|\, b.c \mid (b, c) \in K_0 \,|\!\}$ by the firing sequence $(e_1.\sigma_1, \ldots, e_n.\sigma_n)$. Thus, there is a configuration $\mathsf{C} = \{\mathsf{e}_1, \ldots, \mathsf{e}_n\}$ in $\mathsf{U}(\mathrm{Exp}(O)) = \mathrm{Exp}_\mathrm{O}(O)$ with $\forall i : \pi^\mathrm{O}(\mathsf{e}_i) = e_i.\sigma_i$. Then, by the definition of h, we get $\mathrm{mark}^\mathsf{h}(\mathsf{C}) := \{\!|\, \mathsf{h}(\mathsf{b}) \mid \mathsf{b} \in \mathrm{cut}(\mathsf{C}) \,|\!\} = \{\!|\, h(b).c \mid (b, c) \in K_n \,|\!\} = \mathsf{M}$.

Let now $t.\sigma \in \mathsf{T}$ s.t. $\mathsf{M}[t.\sigma\rangle$. Then $M[t, \sigma\rangle$. Since C, θ satisfy property b) from Def. 4.9, we know that $\exists e \in E$ s.t. $e \notin C$, $h(e) = t$, and $C, \theta[e, \sigma\rangle$. This means, in $\mathrm{Exp}(\beta)$, we have $\{\!|\, b.c \mid (b, c) \in K_n \,|\!\}[e.\sigma\rangle$. Thus, there exists an e in $\mathsf{U}(\mathrm{Exp}(\beta))$ such that $\mathsf{C}[\mathsf{e}.\sigma\rangle$ and $\pi^\mathrm{O}(\mathsf{e}) = e.\sigma$, which again means that $\mathsf{h}(\mathsf{e}) = h(e).\sigma = t.\sigma$. This proves that C and θ satisfy a) and b) from Def. 4.1, and therefore that $\mathrm{Exp}_\mathrm{O}(\beta)$ is complete.

The other direction works analogously. $\square$

We can now compare the two complete finite prefixes resulting from the original ERV-algorithm from [ERV02] applied to $\mathrm{Exp}(N)$ and the generalized ERV-algorithm Alg. 1 applied to $N \in \mathbf{N_F}$. From the definition of the generalized cut-off criterion we get that both these prefixes have the same depth. However, due to the high-level representation, the breadth of the symbolic prefix can be substantially smaller. This is the case for our running example Three Times Termination:

Example 4.22. Consider again $N \in \mathbf{N_F}$ from Figure 4.1 with $Col = \{0, 1, \dots, m\}$ for a fixed $m > 0$. The complete finite prefix contains $6m^4 + 4m + 2\lfloor \frac{m}{3} \rfloor + 2$ nodes for every fixed m in the color class $Col = \{0, 1, \dots, m\}$. The complete finite prefix of the *symbolic* unfolding $\Upsilon(N)$ that is shown in Figure 4.3, on the other hand has the same number of nodes for *every* m.

As we stated above, both prefixes have the same depth. However, it is noteworthy that, would we expand the symbolic prefix then it would be bigger than the P/T prefix. The reason is that some cut-off events in the P/T prefix correspond to *non* cut-off events in the symbolic prefix. $\triangleleft$

Generalizing this example to a family of nets gives the following proposition:

Proposition 4.23. *For every $n \in \mathbb{N}$, there is a family $(N_m^n)_{m \in \mathbb{N}^{>0}}$ of high-level nets $N_m^n \in \mathbf{N_F}$ such that every N_m^n has the set of colors $Col = \{0, \dots, m\}$, and the family satisfies that*

- *the complete finite prefix of $\Upsilon(N_m^n)$ obtained by Alg. 1 has the same number of nodes for every m,*
- *the number of nodes in the low-level prefix of $\Upsilon(\mathrm{Exp}(N_m^n))$ obtained by the original ERV-algorithm is greater than m^n.*

In particular, the benchmark family Fork And Join, which will be presented in Sec. 4.6.2 satisfies this property.

4.4 Handling Infinitely Many Reachable Markings

When applying the generalized ERV-algorithm, Alg. 1, to high-level Petri nets with infinitely many reachable markings (therefore violating **(3)** from the definition of $\mathbf{N_F}$), the proof for finiteness of the resulting prefix does not hold anymore: the proof of Prop. 4.18, step (1), is a generalization of the proof of the respective claim in [ERV02]. This proof, in its turn, uses the pigeonhole principle: the argument is that we cannot have $|\mathcal{R}(N)| + 1$ consecutive events s.t. their cone configurations each generate a marking in the net not seen before, and we thus have a cut-off event. When we deal with infinitely many markings, this argument cannot be made.

In this section, we introduce a class $\mathbf{N_{SC}}$ of safe high-level nets, called symbolically compact, that have possibly infinitely many reachable markings (and therefore an infinite expansion), generalizing the class $\mathbf{N_F}$. We then proceed to make adaptions to Alg. 1

(specifically, to the used cut-off criterion), so that it generates a finite and complete prefix of the symbolic unfolding for any $N \in \mathbf{N_{SC}}$.

The following Lemma precisely describes the finite high-level Petri nets for which a finite and complete prefix of the symbolic unfolding exists. They are characterized by having a bound on the number of steps needed to arrive at every reachable marking. For the proof we argue that in the case of such a bound, the symbolic unfolding up to depth $n+1$ is a finite and complete prefix, and that in the absence of such a bound no depth of a prefix suffices for it to be complete.

Lemma 4.24. *For a finite high-level Petri net $N = (\mathcal{N}, \mathcal{M}_0)$ there exists a finite and complete prefix of $\Upsilon(N)$ if and only if there exists a bound $n \in \mathbb{N}$ such that every marking in $\mathcal{R}(N)$ is reachable from a marking in $\mathcal{M}_0$ by firing at most n transitions.*

Proof. From Prop. 4.6 and Prop. 4.7 we see that for a finite high-level Petri net with such a bound n, the prefix of the symbolic unfolding containing exactly the events e with $d(e) \leq n+1$ is complete. Finiteness of this prefix follows from the finiteness of the original net and the definition of homomorphism.

Assume now that no such bound exists, and, for the purpose of contradiction, assume that there is a finite and complete prefix β of $\Upsilon(N)$. Denote $\widetilde{n} = \max\{|C| \mid C \in \mathcal{C}(\beta)\} < \infty$. Then there exists a marking $M \in \mathcal{R}(N)$ for which we have to fire at least $\widetilde{n}+1$ transitions to reach it. Again from Prop. 4.6 and Prop. 4.7 it follows that a configuration C with $M \in \mathrm{Marks}(C)$ must contain at least $\widetilde{n}+1$ events, contradicting that β is complete. □

4.4.1 Symbolically Compact High-Level Petri Nets

We use the result of Lemma 4.24 to define the class $\mathbf{N_{SC}}$ of high-level nets for which we adapt the algorithm for constructing finite and complete prefixes of the symbolic unfolding.

Definition 4.25 (Class $\mathbf{N_{SC}}$). A finite high-level Petri net N is called *symbolically compact* if it satisfies **(1)** and **(2)** from Def. 4.10, and

(3*) There is a bound $n \in \mathbb{N}$ on the number of transition firings needed to reach all markings in $\mathcal{R}(N)$.

We denote the class containing all symbolically compact high-level Petri nets by $\mathbf{N_{SC}}$. ◁

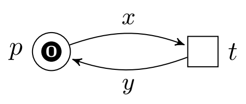
Note that in the case of a (finite, safe) P/T net, property **(3*)** is equivalent to **(3)** (i.e., $|\mathcal{R}(N)| < \infty$). However, this is *not* true for all high-level nets N: while $|\mathcal{R}(N)| < \infty$ still implies **(3*)** (meaning $\mathbf{N_F} \subseteq \mathbf{N_{SC}}$), the reverse implication does not hold, as our running example from Figure 4.1 demonstrates when we change the set of colors to $Col = \mathbb{N}$: it still satisfies **(1)** and **(2)**, with $\mathcal{R}(N) = \{\{\!\!\{ (a,0),(b,0) \}\!\!\}, \{\!\!\{\ \}\!\!\}\} \cup \{\{\!\!\{ (c,k),(b,0) \}\!\!\}, \{\!\!\{ (a,0),(d,k) \}\!\!\}, \{\!\!\{ (c,k),(d,\ell) \}\!\!\} \mid k,\ell \in \mathbb{N}^+\}$. So we have infinitely many markings that can all be reached by firing at most two transitions, meaning the net satisfies **(3*)** and is therefore symbolically compact. The symbolic unfolding in this case is the same as for $Col = \{0, \ldots, m\}$, except that the color class is also replaced. The complete finite prefix from Fig. 4.3 is also complete in this case.

Lemma 4.24 implies that the class $\mathbf{N_{SC}}$ of symbolically compact nets contains exactly all high-level Petri nets satisfying **(1)** and **(2)** for which a finite and complete prefix of the symbolic unfolding exists (independently of the number of reachable markings). Since the reachable markings of a high-level Petri net and its expansion correspond to each other, this observation leads to an interesting subclass $\mathbf{N_{SC}} \setminus \mathbf{N_F}$ of symbolically compact high-level Petri nets that have infinitely many reachable markings. For every net N in this subclass

- there exists a finite and complete prefix of $\Upsilon(N)$, but
- there does *not* exist a finite and complete prefix of $\Upsilon(\mathrm{Exp}(N))$.

In particular, the original ERV-algorithm cannot be applied to $\mathrm{Exp}(N)$, since this expansion is an infinite net.

An example for such a net is our running example from Figure 4.3 when we replace the color class $Col = \{0, 1, \ldots, m\}$ by $Col = \mathbb{N}$, as described above. Much simpler is the following net, also with $Col = \mathbb{N}$:



(4.2)

Obviously, every reachable marking $\{\!|\, (p, n) \,|\!\}$ with $n \in \mathbb{N}$ can be reached by firing t one time in mode $\{x \leftarrow 0, y \leftarrow n\}$, so the net is symbolically compact. The expansion of this net however is infinite, and the original ERV-algorithm does not terminate when applied to it.

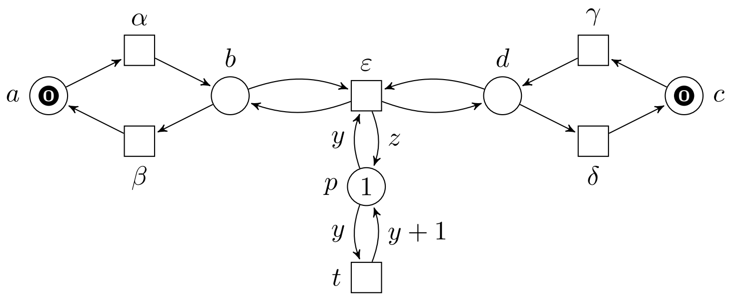
4.4.2 Insufficiency of the Cut-off Criterion for $\mathbf{N_{SC}}$

Naturally, the question arises whether the generalized ERV-Algorithm, Alg. 1, also yields a finite and complete prefix of a symbolically compact input net. For many examples (like the simple one in (4.2) above) this is the case. However, there are symbolically compact high-level Petri nets for which Alg. 1 does *not* terminate.

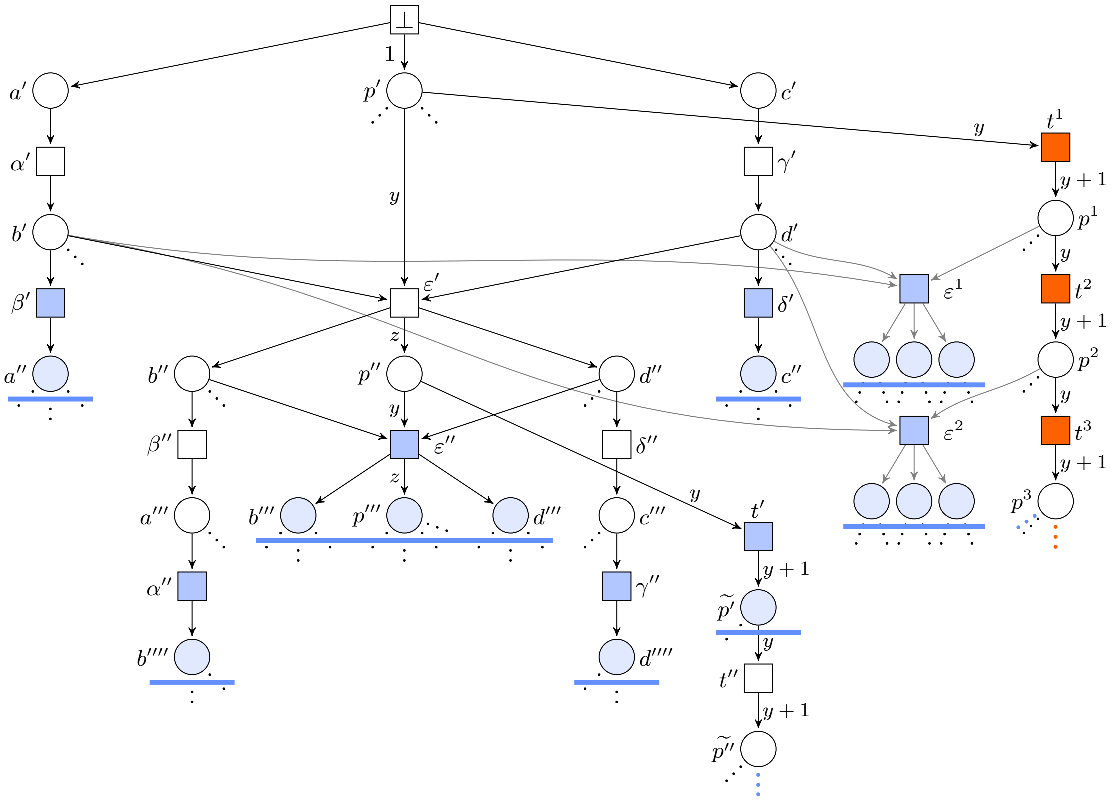
The criterion for nontermination of Alg. 1 is that in the symbolic unfolding Υ of the net, there is an infinite sequence of cone configurations $[e_1], [e_2], \ldots$ with $[e_1] \prec [e_2] \prec \ldots$ such that

- $\forall i \in \mathbb{N}_{>0} : \{e_1, \ldots, e_i\} \in \mathcal{C}(\Upsilon)$, i.e., $(e_1, e_2, \ldots)$ is a fireable sequence in the symbolic unfolding Υ, and
- $\forall i \in \mathbb{N}_{>0} : \mathrm{Marks}([e_i]) \not\subseteq \bigcup_{[e'] \prec [e_i]} \mathrm{Marks}([e'])$, i.e., no event e_i is a cut-off event.

Note that in the second condition, the e' in the union are abitrary events in the unfolding, and not restricted to the sequence $(e_1, e_2, \ldots)$. We give an example for a net satisfying this criterion:



(a) A symbolically compact net with $Col = \mathbb{N}$.



(b) A prefix of the symbolic unfolding of the net in (a).

Figure 4.4: A symbolically compact net in (a) where Alg. 1, trying to build a complete finite prefix of the symbolic unfolding shown in (b), does not terminate.

Example 4.26 (Non-termination of Alg. 1 for a symbolically compact net). Consider the high-level Petri net in Figure 4.4a. The set of colors is given by $Col = \mathbb{N}$. Initially there is a token 0 in each of the places a and c. The token on a can cycle between a and b by transitions α and β. Analogously, the other token can cycle between c and d by γ and δ. Additionally, in the initial marking, there is a color 1 on place p. This number can be increased by 1 by firing t. Thus, every number n can be placed on p by $n-1$ firings of t. When, however, the two cycling tokens of color 0 are on places b and d, an arbitrary number n can be placed directly on p by firing ε. The net therefore is symbolically compact.

Examine now the unfolding in Figure 4.4b. Cut-off events and their output conditions are again shaded blue. For a cleaner presentation we do not write the local predicate next to each event. For the event ε' we have $\text{Marks}([\varepsilon']) = \{\{\!| (b,0),(d,0),(p,n) |\!\} \mid n \in \mathbb{N}\}$. This means, by firing no event, only β'', only δ'', or both β'' and δ'' from the corresponding cut, we can represent every reachable marking in the net.

The sequence $[t^1], [t^2], \ldots$ of cone configurations, with the corresponding events shaded orange, now satisfies the criterion from above: The condition $\forall i : \text{cut}([t^i])[t^{i+1}\rangle$ is obviously satisfied. The sequence of events corresponds to firing t infinitely often, always increasing the number on p by 1. The cones $[t^i]$ are the only cone configurations where the cuts represent a markings with *no tokens* on the two places b and d. For all other cones in the unfolding, there is a 0 on b and/or a 0 on d. Thus, no event t^i is a cut-off event. This means if Alg. 1 is applied to the net in Figure 4.4a it does not terminate, building a prefix containing every t^i with $i \in \mathbb{N}^+$. $\lhd$

4.4.3 The Finite Prefix Algorithm for Symbolically Compact Nets

As previously discussed, the argument that states the existence of one event in a chain of $|\mathcal{R}(N)|+1$ consecutive events, such that every marking represented by its cone configuration is contained in the union of all markings represented by previous cone configurations, cannot be applied in the case of an infinite number of reachable markings. Consequently, Alg. 1 may not terminate when applied to a net in $\mathbf{N_{SC}} \setminus \mathbf{N_F}$. However, condition **(3*)** guarantees that every marking reached by a cone configuration $[e]$ with depth $> n$ can be reached by a configuration C containing no more than n events.

For the algorithm to terminate, we need to adjust the cut-off criterion since we do not know whether C is also a cone configuration, as demanded in Def. 4.15. Therefore, we define *cut-off* events*, that generalize cut-off events. They only require that every marking in $\text{Marks}([e])$ has been observed in a set $\text{Marks}(C)$ for *any* configuration $C \prec [e]$, rather than just considering cone configurations.

Definition 4.27 (Cut-off* event). Under the assumptions of Def. 4.15, the high-level event e is a *cut-off** event (w.r.t. $\prec$) if $\text{Marks}([e]) \subseteq \bigcup_{C \prec [e]} \text{Marks}(C)$. $\lhd$

We additionally assume that the used adequate order satisfies $|C_1| < |C_2| \Rightarrow C_1 \prec C_2$, so that every event with depth $> n$ will be a cut-off event. Since all adequate orders discussed in [ERV02] satisfy this this property (cp. Appendix 4.A.1), this is a reasonable requirement. This adaption and assumption now lead to:

Theorem 4.28. *Assume a given adequate order $\prec$ to satisfy $|C_1| < |C_2| \Rightarrow C_1 \prec C_2$. When replacing in Alg. 1 the term "cut-off event" by "cut-off* event", it terminates for any input net $N \in \mathbf{N_{SC}}$, and generates a complete finite prefix of $\Upsilon(N)$.*

Proof. The properties of symbolic unfoldings that we stated in Sec. 4.2.1 are independent on the class of high-level nets. Def. 4.11 only uses that the considered net is safe, and so do Prop. 4.12 and Prop. 4.14. We therefore only have to check that the correctness proof for the algorithm still holds. In the proof of Prop. 4.18 (*Fin* is finite), the steps (2) and (3) are independent of the used cut-off criterion. In step (1), however, it is shown that the depth of events never exceeds $|\mathcal{R}(N)|+1$. This is not applicable when $|\mathcal{R}(N)| = \infty$, as argued above. Instead we show:

(1*) For every event e of *Fin*, $d(e) \leq n+1$, where n is the bound on the number of transitions needed to reach all markings in $\mathcal{R}(N)$.

In the proof of Prop. 4.19, the cut-off criterion is used to show (by an infinite descent approach), for any marking $M \in \mathcal{R}(N)$ the existence of a minimal configuration $C \in \mathit{Fin}$ with $M \in \mathrm{Marks}(C)$. Due to the similarity of cut-off and cut-off*, this proof can easily be adapted to work as before:

Assume that at some point during the algorithm, we reach a state $(B', E', H', g', \mathcal{K}_0')$ of the prefix under construction, such that there occurs a chain of events $e_1 < e_2 < \cdots < e_{n+1}$. We prove that e_{n+1} must be a cut-off* event. Let $M \in \mathrm{Marks}([e_{n+1}])$. Then, by definition of $\mathbf{N_{SC}}$, M can be reached by firing at most n transitions. Accordingly, from Prop. 4.7, we get that there is a configuration $C \in \Upsilon$ containing at most n events such that $M \in \mathrm{Marks}(C)$. As in the proof of Prop. 4.19, we can now follow that there is a configuration $\widetilde{C} \in \mathcal{C}(\Upsilon)$ such that $M \in \mathrm{Marks}(\widetilde{C})$ and $\widetilde{C} \prec C$, that contains no cut-off event and is therefore in *Fin*. Since $|C| \leq n < n+1 \leq |[e_{n+1}]|$, we follow $\widetilde{C} \prec [e_{n+1}]$. So we have that $\forall M \in \mathrm{Marks}([e_{n+1}])\, \exists \widetilde{C} \prec [e_{n+1}]$, which means that e_{n+1} is a cut-off* event. This proves that *Fin* is finite.

It remains to show termination. In the case of nets in $\mathbf{N_F}$, every object is finite, which, together with Prop. 4.18, leads to termination of the algorithm. For nets in $\mathbf{N_{SC}} \setminus \mathbf{N_F}$, however, there is at least one event e in *Fin* s.t. $|\,\mathrm{Marks}([e])| = \infty$. Thus, we have to show that we can check the cut-off* criterion in finite time. This follows from Cor. 4.33 in the next section, which is dedicated to symbolically representing markings generated by configurations. □

4.4.4 Feasibility of Symbolically Compact Nets and Cut-Off*

To check the cut-off* criterion of an event added to a prefix of the unfolding, we have to compare the set of markings represented by the cut of the event's cone configuration to *all* markings represented by cuts of smaller configurations. This means that we possibly have to store the whole state space.

This realization gives rise to two questions. Firstly, how do we manage the storage of an infinite number of markings? This query is addressed in Sec. 4.5, where we demonstrate how to symbolically represent the markings represented by a configuration's cut and how to check the cut-off* criterion within finite time. The prototype implementation outlined in Sec. 4.6.1 utilizes these methods for the $\mathbf{N_F}$-case.

The second question that arises asks how the complete finite prefix resulting from the generalized ERV-algorithm with the cut-off* criterion relates to a reachability graph

– both in terms of size and computation time. However, as symbolically compact nets possibly have an infinite expansion, the reachability graph can be infinitely broad. Thus, at present this method provides a more viable solution compared to calculating the infinite reachability graph. However, we give an outlook on how a (finite) symbolic reachability tree of a symbolically compact net could possibly be constructed.

Outlook: Symbolic Reachability Trees of Symbolically Compact Nets. The idea of a symbolic reachability tree has been realized for algebraic Petri nets in [Sch95] by Karsten Wolf. However, in contrast to this work, we think that for the class of symbolically compact nets we can build a symbolic reachability tree that is complete.

The idea is to gradually extend for every subset P' of places a formula $\mathcal{R}_{P'}$ that symbolically describes all reachable markings that we have seen so far and have colors on exactly all places in P'. Initially, all formulae are *false*, except for $\mathcal{R}_{P_0}$, where P_0 are the initially marked places. $\mathcal{R}_{P_0}$ symbolically represents the set of initial markings.

The symbolic reachability tree is then constructed by starting with a root n_0 labeled with $\kappa_{n_0} = \mathcal{R}_{P_0}$ representing the set of initial markings. For every transition t, we can determine whether t can fire in any mode from any marking represented by f_{n_0} by a satisfiability check. If t can fire, we add a new vertex n', and label it with a formula κ' that symbolically represents all markings reached from firing t in any mode from any marking in $\mathcal{M}_0$. In all these markings, there are colors on the same set of places P'. If $\kappa' \Rightarrow \mathcal{R}_{P'}$ then we end this branch since this means we have seen every marking represented by κ' before. We then extend $\mathcal{R}_{P'}$ to $\mathcal{R}_{P'} \vee \kappa'$.

By repeating this procedure in breadth-first-order, we build a tree that symbolically represents all reachable markings. This tree should correspond to the complete finite prefix of the symbolic unfolding of the net to which you added a shared resource (in form of a new place) that every transition consumes and recreates. We give here only the idea, and not a formal definition. In future work we want to further investigate on such a tree. We can then compare the complete finite prefix of the symbolic unfolding to the symbolic reachability tree.

4.5 Checking Cut-offs Symbolically

We show how to symbolically check whether a high-level event e is a cut-off* event in finite time. By definition, this means checking whether $\mathrm{Marks}([e]) \subseteq \bigcup_{C \prec [e]} \mathrm{Marks}(C)$. However, since the cut of a configuration can represent infinitely many markings, when applying the adapted algorithm we cannot simply store the set $\mathrm{Marks}(C)$ for every $C \in \mathcal{C}(Fin)$. Instead, we now define constraints that symbolically describe the markings represented by a configuration's cut. Checking the inclusion above then reduces to checking an implication of these constraints. Since we consider high-level Petri nets with guards written in a decidable theory, such implications can be checked in finite time.

At the end we see that this method can be easily adapted to symbolically check whether, in a prefix of the symbolic unfolding of a net $N \in \mathbf{N_F}$, an event e is a cut-off event in the sense of Def. 4.15. This method is also used in the implementation described in Sec. 4.6.1.

For the rest of this section, let $N = (P, T, F, g, \mathcal{M}_0) \in \mathbf{N_{SC}}$ with symbolic unfolding $\Upsilon(N) = (U, \pi) = (B, E, H, g, \mathcal{K}_0, \pi)$.

We first define for every condition b a new predicate $\mathit{pred}^{\odot}(b)$ by

$$\mathit{pred}^{\odot}(b) := \mathit{pred}(\mathbf{e}(b)) \wedge (\pi(b) = \mathbf{v_e}(b)).$$

This predicate now has (in an abuse of notation) an extra variable, called $\pi(b)$. The remaining variables in $\mathit{pred}(\mathbf{e}(b))$ are $\mathit{Var}_{[\mathbf{e}(b)]\cup\{\perp\}}$. As we know, $\mathit{pred}(\mathbf{e}(b))$ evaluates to *true* under an assignment $\theta : \mathit{Var}_{[\mathbf{e}(b)]\cup\{\perp\}} \to \mathit{Col}$ if and only if a concurrent execution of $[\mathbf{e}(b)]$ with the assigned modes is possible (i.e., under every instantiation of $[\mathbf{e}(b)]$). In such an execution, $\theta(\mathbf{v_e}(b)) \in \mathit{Col}$ is placed on b. Due to the equality in the second part of $\mathit{pred}^{\odot}(b)$, the predicate $\mathit{pred}^{\odot}(b)$ therefore can only be true if $\pi(b)$ is assigned a color that can be placed on b.

We now define for a co-set $B' \subseteq B$ of high-level conditions the constraint on B' as an expression with free variables $\pi(B') = \{\pi(b) \mid b \in B'\}$ describing which color combinations can lie on the places represented by the high-level conditions. We build the conjunction over all predicates $\mathit{pred}^{\odot}(b)$ for $b \in B'$ and quantify over all appearing variables v_e.

For that we use the following notation: for a set $X = \{x_1, \ldots, x_n\}$ of arbitrary variables, and a predicate P such that X is a subset of the variables in P, we abbreviate $\exists x_1, \ldots, x_n : P$ by $\exists[X] : P$.

The *constraint on* B' is defined by

$$\kappa(B') = \exists\Big[\bigcup_{b\in B'} \mathit{Var}_{[\mathbf{e}(b)]\cup\{\perp\}}\Big] : \bigwedge_{b\in B'} \mathit{pred}^{\odot}(b), \tag{4.3}$$

With this notation that we just introduce, we quantify in (4.3) over all free variables in $\bigwedge_{b\in B'} \mathit{pred}^{\odot}(b)$ except for the variables in $\pi(B')$.

We denote by $\Xi(B')$ the set of variable assignments $\vartheta : \pi(B') \to \mathit{Col}$ that satisfy $\kappa(B')[\vartheta] \equiv \mathit{true}$.

Example 4.29. Consider the symbolic unfolding of the running example Three Times Termination from Fig. 4.3. We described in Sec. 4.4.1 that for $\mathit{Col} = \mathbb{N}$, the running example from Fig. 4.1 is symbolically compact, and the symbolic unfolding does not change (apart from the color class).

For condition a' we have

$$\begin{aligned}\mathit{pred}^{\odot}(a') &= \mathit{pred}(\mathbf{e}(a')) \wedge (\pi(a') = \mathbf{v_e}(a')) \\ &= \mathit{pred}(\perp) \wedge (a = v_{\perp}^{a'}) \\ &= (v_{\perp}^{a'} = 0) \wedge (v_{\perp}^{b'} = 0) \wedge (a = v_{\perp}^{a'}).\end{aligned}$$

For condition c' we have

$$\begin{aligned}
\mathit{pred}^{\odot}(c') &= \mathit{pred}(\mathbf{e}(a')) \wedge (\pi(c') = \mathbf{v_e}(c')) \\
&= \mathit{pred}(\beta) \wedge (c = x_\beta) \\
&= \mathit{pred}(\bot) \wedge \mathit{loc\text{-}pred}(\beta') \wedge (c = x_{\beta'}) \\
&= (v_\bot^{a'} = 0) \wedge (v_\bot^{b'} = 0) \wedge (v_{\beta'}^0 = v_\bot^{b'}) \wedge (v_{\beta'}^0 = 0) \wedge (x_{\beta'} > 0) \wedge (c = x_{\beta'}).
\end{aligned}$$

Since $\{a', c'\}$ is a co-set, $\kappa(\{a', c'\})$ is defined and given by

$$\begin{aligned}
\kappa(\{a', c'\}) &= \exists \Big[\bigcup_{b \in \{a',c'\}} \mathit{Var}_{[\mathbf{e}(b)] \cup \{\bot\}} \Big] : \bigwedge_{b \in \{a',c'\}} \mathit{pred}^{\odot}(b) \\
&= \exists v_\bot^{a'}\, \exists v_\bot^{b'}\, \exists v_{\beta'}^0\, \exists x_{\beta'} : \\
&\quad (v_\bot^{a'} = 0) \wedge (v_\bot^{b'} = 0) \wedge (a = v_\bot^{a'}) \quad \wedge \\
&\quad (v_\bot^{a'} = 0) \wedge (v_\bot^{b'} = 0) \wedge (v_{\beta'}^0 = v_\bot^{b'}) \wedge (v_{\beta'}^0 = 0) \wedge (x_{\beta'} > 0) \wedge (c = x_{\beta'}).
\end{aligned}$$

This can be simplified such that

$$\kappa(\{a', c'\}) \equiv \exists v_\bot^{a'}\, \exists x_{\beta'} : (v_\bot^{a'} = 0) \wedge (x_{\beta'} > 0) \wedge (a = v_\bot^{a'}) \wedge (c = x_{\beta'}),$$

and even further to

$$\kappa(\{a', c'\}) \equiv (a = 0) \wedge (c > 0).$$

We see that, as described above, the free variables in $\kappa(\{a', c'\})$ are $\{a, c\} = \{\pi(a'), \pi(c')\}$. The set $\Xi(\{a, c\})$ contains all variable assignments $\vartheta : \{a, c\} \to \mathbb{N}^+$ satisfying $(\vartheta(a) = 0) \wedge (\vartheta(c) > 0)$. $\lhd$

For a configuration C, we have that $B' = \mathrm{cut}(C)$ is a co-set, and $\pi(B') = \pi(\mathrm{cut}(C))$ describes the set of places occupied in every marking in Marks(C). Note that in this case, we have $\bigcup_{b \in \mathrm{cut}(C)} \mathit{Var}_{[\mathbf{e}(b)]} = \mathit{Var}_C$, i.e., the bounded variables in $\kappa(\mathrm{cut}(C))$ are exactly the variables appearing in predicates in C. For every instantiation θ of C we define a variable assignment $\vartheta_\theta : \pi(\mathrm{cut}(C)) \to \mathit{Col}$ by setting $\forall \pi(b) \in \pi(\mathrm{cut}(C)) : \vartheta_\theta(b) = \theta(\mathbf{v_e}(b))$. Instantiations of a configuration and the constraint on its cut are related as follows.

Lemma 4.30. *Let $C \in \mathcal{C}(\Upsilon(N))$. Then $\Xi(\mathrm{cut}(C)) = \{\vartheta_\theta \mid \theta \in \Theta(C)\}$.*

Proof. The proof follows by construction of $\mathit{pred}^{\odot}$ and ϑ_θ: Let $\vartheta \in \Xi(\mathrm{cut}(C))$. Then $\mathit{true} \equiv \kappa(\mathrm{cut}(C))[\vartheta]$. Thus, there exists $\theta : \mathit{Var}_{C \cup \{\bot\}} \to \mathit{Col}$ s.t.

$$\Big(\bigwedge_{b \in \mathrm{cut}(C)} \mathit{pred}^{\odot}(b) \Big)[\vartheta][\theta] \equiv \mathit{true}$$

and therefore

$$\Big(\bigwedge_{b \in \mathrm{cut}(C)} \mathit{pred}(\mathbf{e}(b))[\theta] \Big) \wedge \Big(\bigwedge_{b \in \mathrm{cut}(C)} \vartheta(\pi(b)) = \theta(\mathbf{v_e}(b)) \Big) \equiv \mathit{true}. \tag{4.4}$$

From the inductive definition of *pred* then follows that $\forall e \in C \cup \{\bot\} : \mathit{pred}(e)[\theta] \equiv \mathit{true}$. Thus, θ is an instantiation of C, and $\vartheta_\theta = \vartheta$, as shown by the posterior conjunction in (4.4).

Let on the other hand $\theta \in \Theta(\mathrm{cut}(C))$. Then directly, by the definition of $\mathit{pred}^{\odot}(b)$ and ϑ_θ, we get $\big(\bigwedge_{b\in\mathrm{cut}(C)} \mathit{pred}^{\odot}(b)\big)[\vartheta_\theta][\theta] \equiv \mathit{true}$ and by the definition of $\kappa(\mathrm{cut}(C))$ that $\kappa(\mathrm{cut}(C))[\vartheta_\theta] \equiv \mathit{true}$, i.e., $\vartheta_\theta \in \Xi(\mathrm{cut}(C))$. □

From the definition of $\mathrm{Cuts}(C)$ and $\mathrm{Marks}(C)$ we get:

Corollary 4.31. *Let* $C \in \mathcal{C}(\Upsilon(N))$. *Then* $\mathrm{Cuts}(C) = \{\{(b, \vartheta(\pi(b))) \mid b \in \mathrm{cut}(C)\} \mid \vartheta \in \Xi(\mathrm{cut}(C))\}$ *and* $\mathrm{Marks}(C) = \{\{\!\!\{ (\pi(b), \vartheta(\pi(b))) \mid b \in \mathrm{cut}(C) \}\!\!\} \mid \vartheta \in \Xi(\mathrm{cut}(C))\}$.

Example 4.29 (continued). For the configuration $[\beta'] = \{\beta\}$ we have $\mathrm{cut}([\beta']) = \{a', c'\}$. Since a' is initially occupied by a color 0, and β' can only place a color greater than 0 on c', we get $\mathrm{Cuts}(\{\beta'\}) = \{\{(a', 0), (c', k)\} \mid k > 0\}$. And indeed we see that

$$\begin{aligned}
&\{\{(b, \vartheta(\pi(b))) \mid b \in \{a', c'\}\} \mid \vartheta \in \Xi(\{a', c'\})\} \\
&= \{\{(a', \vartheta(a)), (c', \vartheta(c))\} \mid \vartheta \in \Xi(\{a', c'\})\} \\
&= \{\{(a', 0), (c', k)\} \mid k > 0\} = \mathrm{Cuts}([\beta']).
\end{aligned}$$

From this we can also directly verify

$$\begin{aligned}
\mathrm{Marks}([\beta']) &= \{\{\!\!\{ (a, 0), (c, k) \}\!\!\} \mid k > 0\} \\
&= \{\{\!\!\{ (\pi(b), \vartheta(\pi(b))) \mid b \in \mathrm{cut}(\beta') \}\!\!\} \mid \vartheta \in \Xi(\mathrm{cut}([\beta]))\}.
\end{aligned}$$

Thus, the constraint $\kappa(\mathrm{cut}([\beta])) \equiv (a = 0) \wedge (c > 0)$ calculated above indeed precisely described the markings represented by the cut of $[\beta]$. ◁

We now show how to check whether an event is a cut-off* event via the constraints defined above. For that, we first look at general configurations in Theorem 4.32, and then explicitly apply this result to cone configurations $[e]$ in Cor. 4.33.

Theorem 4.32. *Let* $C, C_1, \ldots, C_n$ *be finite configurations in the symbolic unfolding of a safe high-level Petri net s.t.* $\forall 1 \le i \le n : \pi(\mathrm{cut}(C)) = \pi(\mathrm{cut}(C_i))$. *Then*

$$\mathrm{Marks}(C) \subseteq \bigcup_{i=1}^{n} \mathrm{Marks}(C_i) \quad \textit{if and only if} \quad \kappa(\mathrm{cut}(C)) \Rightarrow \bigvee_{i=1}^{n} \kappa(\mathrm{cut}(C_i)).$$

Proof. Assume $\mathrm{Marks}(C) \subseteq \bigcup_{i=1}^{n} \mathrm{Marks}(C_i)$ and let $\vartheta \in \Xi(\mathrm{cut}(C))$. We have that $M_\vartheta := \{(\pi(b), \vartheta(\pi(b))) \mid b \in \mathrm{cut}(C)\} \in \mathrm{Marks}(C)$ by Cor. 4.31. Thus, $\exists 1 \le i \le n : M_\vartheta \in \mathrm{Marks}(C_i)$. This, again by Cor. 4.31, means $\exists \vartheta_i \in \Xi(\mathrm{cut}(C_i)) :$

$$M_\vartheta = \{(\pi(b'), \vartheta_i(\pi(b'))) \mid b' \in \mathrm{cut}(C_i)\}$$

This shows that $\vartheta = \vartheta_i$. Thus, $\kappa(\mathrm{cut}(C_i))[\vartheta] \equiv \mathit{true}$, which proves the implication.

Assume on the other hand $\kappa(\mathrm{cut}(C)) \Rightarrow \bigvee_{i=1}^{n} \kappa(\mathrm{cut}(C_i))$. Let $M \in \mathrm{Marks}(C)$. Then $\exists \vartheta \in \Xi(\mathrm{cut}(C)) : M = \{(\pi(b), \vartheta(\pi(b))) \mid b \in \mathrm{cut}(C)\}$. Thus, $\exists 1 \le i \le n : \kappa(\mathrm{cut}(C_i))[\vartheta] \equiv \mathit{true}$. Let $\vartheta_i = \vartheta$. Then $\vartheta_i \in \Xi(\mathrm{cut}(C_i))$, and $M = \{(\pi(b'), \vartheta_i(\pi(b'))) \mid b' \in \mathrm{cut}(C_i)\} \in \mathrm{Marks}(C_i)$, which completes the proof. □

The following Corollary now gives us a characterization of cut-off* events in a symbolic branching process. It follows from Theorem 4.32 together with the facts that $\mathrm{Marks}(C_1) \cap \mathrm{Marks}(C_2) \neq \emptyset \Rightarrow \pi(\mathrm{cut}(C_1)) = \pi(\mathrm{cut}(C_2))$, and that $\prec[e]$ is finite.

Corollary 4.33. *Let $\beta = (O, h)$ be a symbolic branching process and e an event in β. Then e is a cut-off* event in β if and only if*

$$\kappa(\mathrm{cut}([e])) \Rightarrow \bigvee_{\substack{C \prec [e] \\ h(\mathrm{cut}(C)) = h(\mathrm{cut}([e]))}} \kappa(\mathrm{cut}(C)).$$

Example 4.34. Consider again the symbolic unfolding of the running example Three Times Termination from Fig. 4.3 for $Col = \mathbb{N}$. For the event ε' we have $\pi(\mathrm{cut}([\varepsilon'])) = \{c, d\}$. We have only one configuration C satisfying $C \prec [\varepsilon']$ and $\pi(\mathrm{cut}(C)) = \{c, d\}$, namely $C = \{\alpha', \beta'\}$. Since $\kappa(\mathrm{cut}([\varepsilon'])) \equiv (c > 0) \wedge (d > 0) \equiv \kappa(\mathrm{cut}(C))$ we have in particular $\kappa(\mathrm{cut}([\varepsilon'])) \Rightarrow \kappa(\mathrm{cut}(C))$, making ε' a cut-off* event. This means ε' is a cut-off* event but no cut-off event. $\lhd$

We showed how to decide for any event e added to a prefix of the unfolding whether it is a cut-off* event, namely, by checking the above implication in Cor. 4.33. Note that we can also check whether e is a *cut-off* event (w.r.t. Def. 4.15) by the implication in Cor. 4.33 when we replace all occurrences of "C" by "$[e']$". This strategy of symbolically checking cut-offs was in particular used in the implementation ColorUnfolder [Pan], as described in the next section.

4.6 Implementation and Experimental Results

This section, we delve into the implementation details of the generalized ERV-algorithm and discuss the results of our experiments. We give a concise overview of the technical decisions made during implementation and provide an evaluation of its performance across four benchmark families. We identify a property called "mode-determinism" that offers an indicator for whether (a complete finite prefix of) the symbolic unfolding is expected to be faster to construct than (a complete finite prefix of) the low-level unfolding. The primary work on the implementation (described in Sec. 4.6.1) and experiments (described in Sec. 4.6.4) was conducted by Lukas Panneke as part of his master thesis [Pan23].

4.6.1 Implementation Specifics

Other tools designed for generating (prefixes of) P/T Petri net unfoldings include Mole [Sch], Cunf [Rod; RS13], and Punf [Kho]. However, as these tools are specifically optimized for their intended purpose and do not cater to high-level Petri nets, the new algorithms were not integrated into any of these frameworks. Furthermore, we refrain from conducting a speed comparison between our implementation and the aforementioned tools. The objective of Section 4.6 is to provide a comparison between two approaches: calculating a respective complete finite prefix of the low-level or the symbolic unfolding.

The prototype implementation is called ColorUnfolder [Pan] and written in the Java programming language. It serves a dual purpose as an implementation of the low-level approach as a base for comparisons, and the novel symbolic approach. It can calculate a finite complete prefix of the low-level unfolding for a given high-level Petri net, combining the concepts from [ERV02] (complete finite prefixes) and [KK03] (generating the low-level unfolding without expansion). Additionally, it is capable of computing a complete finite prefix of the net's symbolic unfolding, utilizing a modified version of Alg. 1. Since the goal was to compare the low-level with the high-level case, the considered nets were restricted to the class $\mathbf{N_F}$ to guarantee that the low-level unfolding exists.

Both, the generalized (Alg. 1) and the original ([ERV02]) ERV-algorithm create possible extensions that are structurally dependent cut-off events, whereas in the implementation a cut-off event never triggers the calculation of possible extension. With the same idea, conditions in the postset of cut-off events are never considered for finding co-sets. This leaves the finite complete prefix unmodified, as it only eliminates unnecessary work.

More importantly, the tool operates on a modified unfolding since an implementation using the predicates defined here turned out to be very slow. It rewrites the predicates in the unfolding and modifies arc labels to drastically reduce the number of variables. After a finite complete prefix of the modified unfolding is found, the result is transformed into the expected result with barely any overhead. The underlying idea is to reuse the same variable in the unfolding for as long as the tokens represented by it in firing modes have the same color. For example, in the unfolding presented in Figure 2.9b, ColorUnfolder replaces the four variables by a single variable. For more details, cf. [Pan23].

This optimization yields a significant speed up. However, when working on the symbolic unfolding, in the experiments still more than 99 percent of the time is spent evaluating the satisfiability of predicates to identify cut-off events using Cor. 4.33, and to detect when to discard event candidates because of a color conflict. For this task, the *cvc5* SMT solver [BBB+22] was chosen. It performed best in the relevant category (non-linear arithmetic with equality and quantifiers) of the Satisfiability Modulo Theories Competition (SMT-COMP 2023)[1].

4.6.2 Benchmark Families

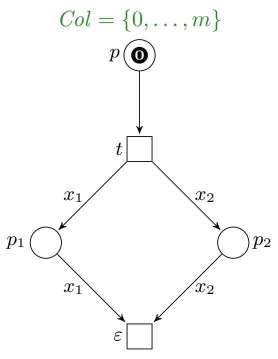
In this section, we present the four benchmark families on which the calculation of (resp. verification on) the symbolic unfolding was tested and compared to the calculation of (resp. verification on) the low-level unfolding. These benchmark were developed together with Lukas Panneke in the context of his master thesis [Pan23] and are introduced in [WCHP23].

Fork And Join

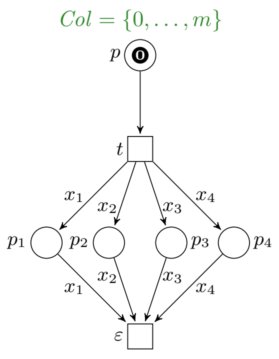
The simplest of our benchmark families is called *Fork And Join*. In the initial marking, a token resides on place p_0. A transition t takes this token from p and places an arbi-

[1] `https://smt-comp.github.io/2023/results/equality-nonlineararith-single-query`

trary color on each of its output places. A transition ε then takes these colors from all places, ending the nets execution. We have two parameters: the first parameter, $m \in \mathbb{N}$, determines the set of colors $Col = \{0, \ldots, m\}$. The second parameter, $n \in \mathbb{N}$, determines the number of output places of t. Fig. 4.5 shows Fork And Join for $n = 2$ in (a) and for $n = 4$ in (b).



(a) $n = 2$.



(b) $n = 4$.

Figure 4.5: Fork And Join for $n = 2$ in (a) and $n = 4$ in (b).

The symbolic unfolding of a Fork And Join has $n+3$ nodes as it is structurally equal to the net itself. The low-level unfolding of the expansion has $(n+2)(m+1)^n + 1$ nodes (since t is fireable in $(m+1)^n$ modes). The P/T unfolding is presented in Appendix 4.A.2.

The Water Pouring Puzzle

This benchmark family generalizes the following logic puzzle (cf., e.g., [AP76]):

> "You have an infinite supply of water and two buckets. One holds 5 liters, the other holds 3 liters. Measure exactly 4 liters of water in one bucket."

In our generalization we have two parameters. The first parameter is a finite list $n = [n_1, \ldots, n_k]$ of natural numbers. Each entry n_i represents an available bucket i holding n_i liters. The second parameter, $m \in \mathbb{N}$ is the amount of water that should be measured. Fig. 4.6 shows the high-level Petri net corresponding to the parameters $n = [3, 5]$ and $m = 4$, corresponding to the puzzle above. Independently of the parameters, we have $Col = \mathbb{N}$. The current fill level of each bucket i is represented by a place *bucket-i*, with an initial color 0, and two attached transitions *fill-i* and *empty-i*, that, when fired, replace the color on *bucket-i* by n_i or 0, respectively. Additionally, for each pair of buckets i, j, there are two transitions *i-transfer-j* and *j-transfer-i* that transfer as much water as possible from one bucket to the other without overflowing it. When at least one bucket contains m liters, the transition *goal* can fire. We include a prefix the symbolic unfolding of the net from Fig. 4.6 in Appendix 4.A.2.

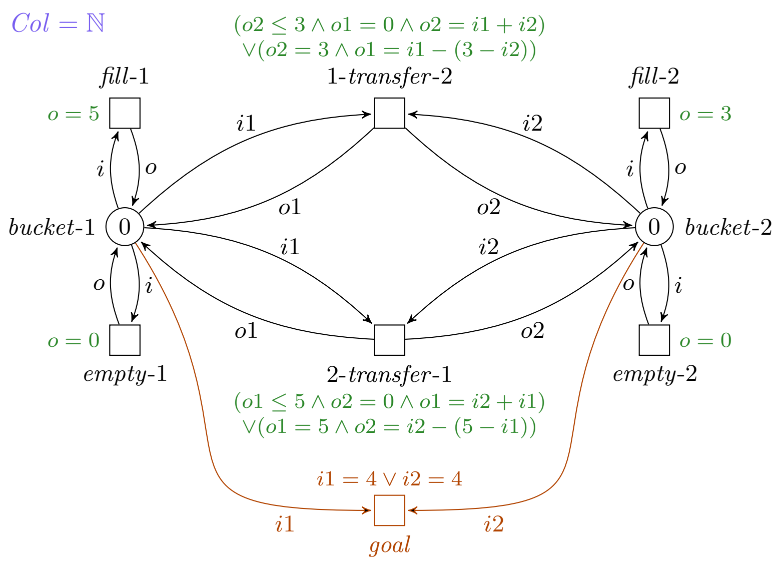


Figure 4.6: The Water Pouring Puzzle with 2 buckets, where one can hold 5 liters and the other can hold 3 liters.

Hobbits And Orcs

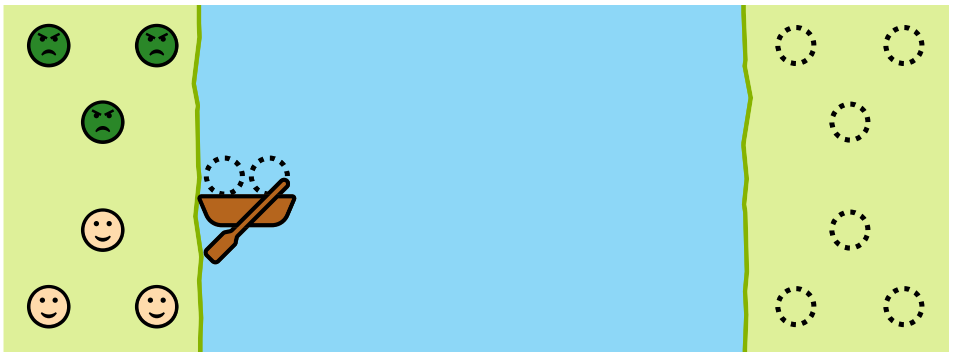
The *Hobbits And Orcs* problem[2] is another logic puzzle (in particular one of many "river crossing problems", cf., e.g., [JPRA77; PS89]):

> "Three Hobbits and three Orcs must cross a river using a boat which can carry at most two passengers. For both river banks, if there are Hobbits present on the bank, they cannot be outnumbered by Orcs (if they were, the Orcs would attack the Hobbits). The boat cannot cross the river by itself with no one on board."

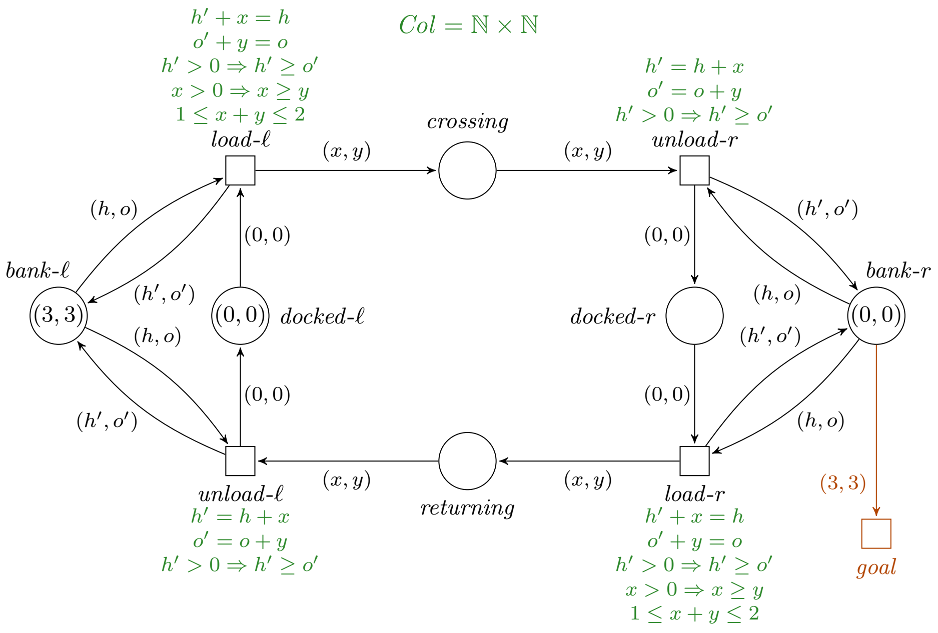
We generalize this problem by introducing two parameters. The first parameter, $m \in \mathbb{N}$ is the number of both Hobbits and Orcs. We always have equally many of the two parties. The second parameter, $n \in \mathbb{N}$ is the number of passengers the boat can carry. We additionally assume that also on the boat, if there are Hobbits present, they cannot be outnumbered by Orcs.

Figure 4.7a shows an illustration of the original puzzle presented above, with three of both, Hobbits and Orcs, and a boat fitting two passengers. Figure 4.7b shows the corresponding high-level Petri net. The colors are given by $Col = \mathbb{N} \times \mathbb{N}$, where a tuples describes the number of Hobbits and Orcs at a location – the left bank, the boat, or the right bank. We start with three Hobbits and three Orcs on the left bank, indicated by the tuple $(3, 3)$ on the place *bank*-ℓ. The four center places describe the current state of the boat, being either empty and docked on a bank, or loaded and on the river. Initially,

[2]The problem is also known as "Missionaries and Cannibals", and is a variation of the "Jealous Husbands" problem.



(a) An illustration of the “Hobbits And Orcs” problem.[3]



(b) The problem modeled as a high-level Petri net.

Figure 4.7: The Hobbits And Orcs problem with 3 Hobbits, 3 Orcs, and a boat fitting 2 passengers.

there is a tuple $(0, 0)$ on *docked-ℓ*, indicating an empty boat the left bank. Via the transitions *load* and *unload* left and right, the boat can be loaded or unloaded, with the guards ensuring all the conditions from the riddle regarding the number of Hobbits and Orcs on both banks and on the boat. When all creatures are on the right bank, the transition *goal* can fire, ending the net’s execution.

[3]The "Boat" icon used in Figure 4.7a is by DinosoftLabs from Noun Project, `https://thenounproject.com/dinosoftlab/`.

Mastermind

The last benchmark family models a generalization of the classic code-breaking game *Mastermind* developed in the early Seventies and completely solved in 1993 [Knu77; KL93]. The game is played between two players. The *code maker* secretly chooses an ordered, four digit color code, with six available colors. The *code breaker* then guesses the code. The code maker evaluates the guess by a number of *red pins* indicating how many colors in the guess are in the correct position, and a number of *white pins* indicating how many colors in the guess appear at a different position in the code. Using this knowledge, the code breaker makes the next guess, with up to twelve attempts.

In our generalization we have three parameters. The first parameter, $m \in \mathbb{N}$, describes the number of available colors. The second parameter, $n \in \mathbb{N}$, describes the length of the code. The third parameter, $k \in \mathbb{N}$, describes the number of guesses the code breaker can make. To simplify the net, we restricted the allowed codes to not contain any color twice.

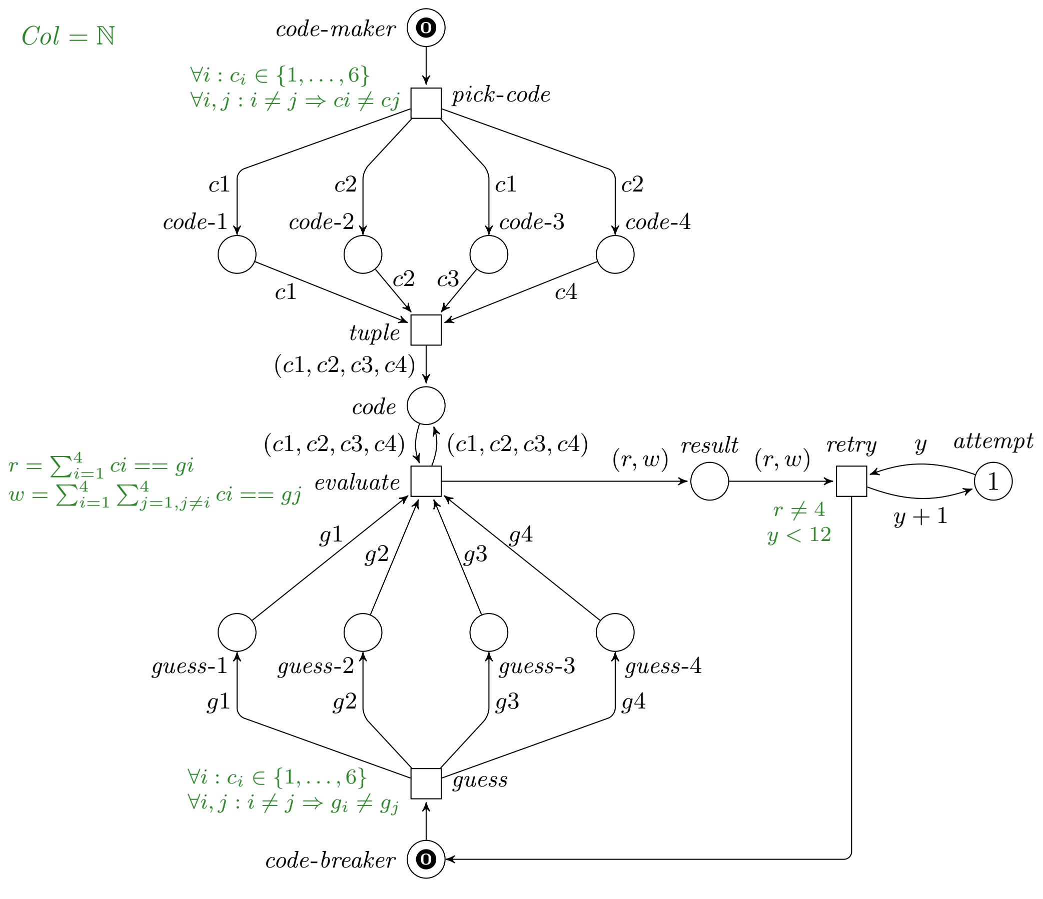


Figure 4.8: The Mastermind game with a code of length 4, 6 possible colors, and 12 attempts.

Fig. 4.8 shows the high-level Petri net with $m = 6$ available colors, a code of length $n = 4$, and $k = 12$ possible attempts, i.e., the scenario described above. The code maker places one color on each of the four places *code-i* by firing *pick-code*. Transition *tuple* puts these colors into a tuple on place *code*. The purpose of this (for the model unimportant) transition is to make the symbolic unfolding, presented in Appendix 4.A.2, resemble an actual game board of Mastermind. The code breaker, analogously to the code maker, concurrently guesses a code via transition *guess*. Transition *evaluate* compares this guess to the code and places the result, i.e., the corresponding number of red and white pins, on *result*. From there, the code breaker either wins if the guessed code was correct, or resets with transition *retry*, provided that he has another attempt left.

4.6.3 Mode-Deterministic High-Level Petri Nets

In the experiments presented in the next section we identified an important indicator for whether using the symbolic approach for a finite complete prefix presented in this chapter is expected to be faster than to calculate a complete finite prefix of the low-level unfolding, combining the concepts of [ERV02] and [KK03].

The identified net property is that in every reachable marking of N, every transition can fire in at most one mode. We call a high-level Petri net with this property *mode-deterministic*, formally:

Definition 4.35. A high-level Petri net N with transitions T is called *mode-deterministic* iff

$$\forall M \in \mathcal{R}(N)\, \forall t \in T\, \exists^{\leq 1} \sigma \in \Sigma(t) : M[t, \sigma\rangle. \qquad \lhd$$

In the case of a mode-deterministic net N, the *skeleton* of N's symbolic unfolding (essentially, the core structure of the high-level occurrence net, devoid of arc labels and guards, and interpreted as a P/T Petri net) is equivalent in structure to the low-level unfolding of N's expansion. This implies that the high-level abstraction does not offer any computational advantage in the unfolding process. To the best of my knowledge, this property has not been studied elsewhere.

We borrow terminology from "regular" determinism and say that a high-level Petri net is *mode-nondeterministic* if it does not satisfy the above property, and implicitly describe by a high or low "degree" of mode-nondeterminism if there are many or few transition-mode combinations making the net mode-nondeterministic.

An illustrative example of a family of mode-deterministic nets is the benchmark family Water Pouring Puzzle introduced in Sec. 4.6.2. Although this property may not be immediately apparent, it is true that in every state of the system, all transitions can fire in at most one mode. Specifically, transitions *fill-i* and *fill-i* replace the color on *bucket-i* with a predetermined one. Every transition *i-transfer-j* emulates the transfer of *all* water from bucket i that can fit into bucket j.

A condition that guarantees N to be a mode-deterministic net is as follows: "For any conceivable assignment of colors to variables on input arcs, there exists at most one possible mode that completes this assignment to all variables." The nets in the

Water Pouring Puzzle family fulfill this criterion, since all output variables (o) are either predetermined or derived from the input variables (i).

An example for nets that have a high degree of mode-nondeterministism is the benchmark family Fork And Join from Sec. 4.6.2. There are only two transitions in the net, but from the initial marking, t can fire in *every* of its m^n modes. This structural pattern of Fork And Join can also be observed in the Mastermind benchmark family (Sec. 4.6.2), making these nets also mode-nondeterministic.

The Hobbits and Orcs family (Sec. 4.6.2), on the other hand, falls in between and is highly contingent on the parameters involved. When there are only two seats on the boat, there are merely five "potential modes" of *load*-ℓ from the initial marking (reflecting the possible ways to occupy one or both seats with two types of creatures). However, in the scenario where there are n seats on the boat, along with m Hobbits and Orcs, and $m > n$, the total number of possibilities is $\sum_{i=1}^{n} i + 1 = \frac{1}{2}n(n+3)$. It's worth noting that approximately half of these possibilities would result in more orcs than hobbits on the boat, rendering them non-modes. Hence, given the condition $m > n$, the "degree" of mode-determinism remains unaffected by an increase in m, but solely varies with the parameter n.

In the subsequent section, we empirically validate the hypothesis that the degree of mode-nondeterminism is proportional to the benefit gained from employing symbolic unfolding compared to low-level unfolding.

4.6.4 Experiments and Results

We now present the experimental results of applying ColorUnfolder to the four benchmark families described above. These experiments were set up by Lukas Panneke as part of his master thesis [Pan23]. He then supplied the results in the form of Fig. 4.9, Table 4.1, Fig. 4.10, and Table 4.2. These results are also presented in [WCHP23].

ColorUnfolder can check the reachability of a (set of) marking(s) by adding a respective transition to the net. It then executes Alg. 1 but stops when an instance of this transition is added to the prefix under construction, which means that the respective (set of) marking(s) is/are reachable. When the algorithm terminates without such an instance, the completeness of the prefix implies that the marking is not reachable.

For the nets from the Fork And Join benchmark family, the complete unfolding (being its own smallest finite and complete prefix) is calculated. For the benchmark families Water Pouring Puzzle and Hobbits And Orcs, the reachability of the goal state of the riddle is checked. This is done by adding the respective *goal* transition that is depicted in each of the two figures Fig. 4.6 and Fig. 4.7. This transition is not part of the input net. Finally, for nets in the Mastermind benchmark family, reachability of a marking with the result of $n-1$ red pins and 1 white pin is checked. Such a marking is never reachable, so always the complete (but finite) unfolding is calculated.

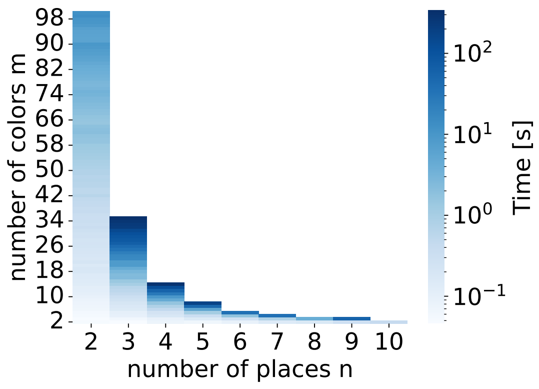
The tasks described above are executed twice, once using the symbolic unfolding and once using the low-level unfolding as described in Sec. 4.6.1. The experiments are calculated on an otherwise idle system with an Intel i7-6700K CPU at 4.0 GHz and 16 GB RAM.

Fork & Join. The results for Fork & Join are shown in Fig. 4.9.

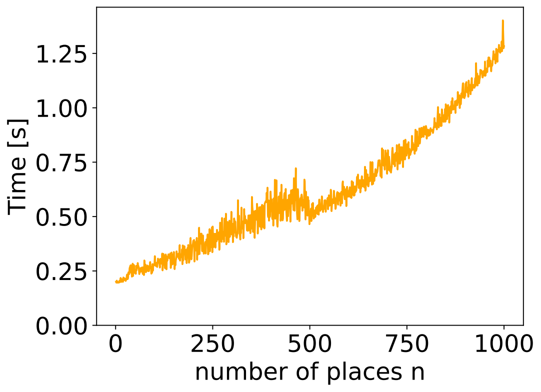
In Fig. 4.9a, the elapsed time for calculating the low-level unfolding for the instance with n places (x-axis) and m colors (y-axis) is indicated by the heatmap intensity of the respective cell. Empty cells exceeded a 5-minute timeout. We see that the low-level approach is only viable if at least one parameter is very small. It takes exponentially more time with a growing color class and with growing number of places.

Since the symbolic approach outperforms the low-level approach by a wide margin, it is presented in its own figure, Fig. 4.9b. For all the cases from Fig. 4.9a, the symbolic approach takes around 200 ms to complete the symbolic unfolding. This is independent of the color class $\{0, \ldots, m\}$. This corresponds to the fact that the structure of the symbolic unfolding of Fork & Join is, for a fixed number of places n, independent of the color class $Col = \{0, \ldots, m\}$. This even does not change for $Col = \mathbb{N}$. Since cvc5 (the tool used for checking satisfiability, cp. Sec. 4.6.1) can handle such infinite color domains, $Col = \mathbb{N}$ is fixed, and Fig. 4.9b shows the elapsed time (y-axis) for calculating the symbolic unfolding of the Fork & Join instance with n places (x-axis). This approach is very fast (but not linear in the number of places).

The symbolic approach is faster for all choices of the parameters. Only for the smallest choices are both approaches equally fast within the margin of error. This behavior was expected since the Petri nets are highly mode-nondeterministic, as explained above, and the low-level unfolding is much broader than the symbolic unfolding.



(a) Low-level, for growing number of places on x-axis, color class on y-axis and time as heatmap intensity.



(b) Symbolic, for growing number of places on x-axis and time on y-axis. Behavior is independent of the color class. This graph is for $Col = \mathbb{N}$.

Figure 4.9: Time needed to calculate unfolding (a) and symbolic unfolding (b) of Fork & Join.

Water Pouring Puzzle. For the Water Pouring Puzzle we cannot get interesting results by varying one parameter while holing the others fixed, because the complexity of the solution is highly volatile and dependent on the combination of all parameters.

Since the nets are mode-deterministic, the low-level and symbolic unfolding, as well as their prefixes, are isomorphic.

Table 4.1: Results of the Water Pouring Puzzle benchmark.

buckets n	target m	solvable	$\|B\|$	$\|E\|$	time low-level	time symbolic
$3, 5$	4	yes, 6 steps	90	75	**95 ms**	1.2 s
$15, 17$	10	yes, 18 steps	258	195	**220 ms**	36 s
$57, 73$	51	yes, 92 steps	1294	935	**1.7 s**	> 1 h
$9, 12$	4	no	106	74	**130 ms**	1.8 s
$10, 16$	5	no	190	134	**140 ms**	11 s
$8, 14, 17$	2	yes, 4 steps	411	642	**300 ms**	26 s
$14, 26, 27$	14	yes, 15 steps	21635	15279	**7.7 s**	> 1 h
$12, 15, 18$	10	no	2391	1442	**550 ms**	20 min
$12, 21, 27$	8	no	4029	2444	**1.0 s**	88 min

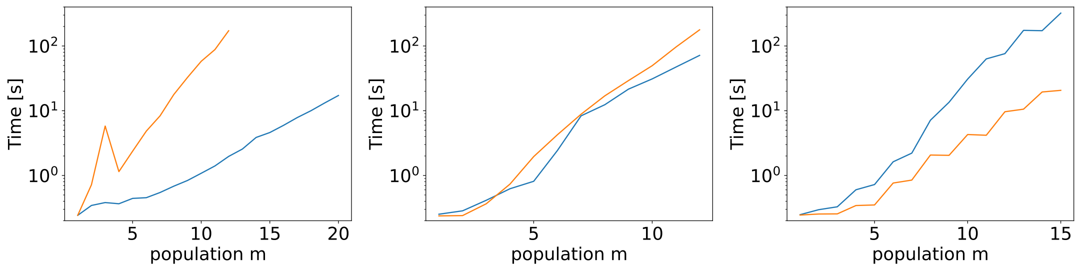
Table 4.1 presents the results for this benchmark. The original puzzle with $n = [3, 5]$ and $m = 4$ is included in the first line. Additionally, eight parameter sets were randomly selected yielding four scenarios involving 2 buckets and four scenarios involving 3 buckets.

For each scenario it is indicated whether the puzzle is solvable, and in the positive case, how many steps a minimal solution has. The times to check the reachability of a goal state (fireability of the *goal* transition, cp. above) of the low-level and symbolic approach are compared. The faster approach is highlighted with bold font. Additionally, the number of conditions ($|B|$) and events ($|E|$) of the generated prefix is indicated. Each of these two numbers coincides between the two approaches.

When the parameters are picked at random the low-level approach generally is much faster. However,examples in which the symbolic approach outspeeds the low-level approach can be constructed. In these cases we have a large color domain but can reach the goal quite quickly, e.g., with two buckets of capacity 10^6 and $10^6 + 1$, and a target of $m = 1$.

Hobbits And Orcs. For the Hobbits And Orcs benchmark the capacity n of the boat is fixed in Figures 4.10a, 4.10b, 4.10c, and the number of each Hobbits and Orcs, m, is plotted on the x-axis. The time needed by the low-level approach and by the symbolic approach is indicated in blue and in orange, respectively, with a timeout of 3 minutes.

The low-level approach performs better when the boat capacity is smaller. For $n = 2$, the symbolic approach gets a timeout at $m = 13$ while the low-level approach can calculate the prefix up to $m = 33$. We expected this behavior, because a smaller the boat capacity yields a higher degree of mode-determinism in the net. With the capacity n also the degree of mode-nondeterminism increases. For $n \geq 6$ the symbolic approach is faster for all population sizes. Independent of the boat size n, both approaches take exponentially more time with growing population size.



(a) $n = 2$ (b) $n = 4$ (c) $n = 6$

Figure 4.10: Results for Hobbits And Orcs problem for a fixed boat capacity n in (a), (b), (c), and population with m of each Hobbits and Orcs on the x-axis. The orange line shows the time needed by the symbolic approach and the blue line shows the low-level approach.

Table 4.2: Results of the Mastermind benchmark.

			Low-level			Symbolic		
k	n	m	time	|B|	|E|	time	$\|B\|$	$\|E\|$
12	4	6	> 2 min	-	-	**1 min**	149	36
1	3	3	86 ms	219	48	**62 ms**	14	3
1	3	5	1.8 s	18363	3720	**54 ms**	14	3
1	3	6	18 s	72723	14640	**61 ms**	14	3
1	3	7	> 2 min	-	-	**54 ms**	14	3
1	3	1000	> 2 min	-	-	**53 ms**	14	3
1	4	4	1 s	3651	624	**60 ms**	17	3
1	4	5	> 2 min	-	-	**67 ms**	17	3
1	5	1000	> 2 min	-	-	**77 ms**	20	3
1	28	1000	> 2 min	-	-	**1 s**	89	3
2	3	3	215 ms	1719	438	**206 ms**	24	6
2	3	4	3.3 s	110115	27672	**120 ms**	24	6
2	3	5	111 s	1726443	432060	**124 ms**	24	6
2	3	6	> 2 min	-	-	**119 ms**	24	6
2	3	1000	> 2 min	-	-	**127 ms**	29	6
2	4	4	5.5 s	137235	27672	**1.3 s**	29	6
2	4	5	> 2 min	-	-	**116 ms**	29	6
2	4	1000	> 2 min	-	-	**162 ms**	29	6
2	10	1000	> 2 min	-	-	**1 s**	59	6
3	4	4	> 2 min	-	-	**35 s**	41	9
4	3	3	2.6 s	46719	12138	**803 ms**	44	12
5	3	3	39 s	234219	60888	**1.3 s**	54	15
6	3	3	> 2 min	-	-	**1.8 s**	64	18

Mastermind. Some results for the Mastermind benchmark are presented in Table 4.2. The time needed by the low-level and symbolic approach to generate the unfolding is compared, with a timeout of two minutes. The table also shows the number of nodes (conditions $|\mathsf{B}|$ resp. $|B|$, and events $|\mathsf{E}|$ resp. $|E|$) in the unfolding. In each row the faster time is highlighted in bold font.

The first line shows the original problem with $k = 12$ attempts, a code length of $n = 4$, and $m = 6$ colors. We observe that in the low-level approach, time and size grow exponentially with respect to the number m of colors, whereas the high-level approach remains constant. When the parameter n controlling the length of the code is increased, both approaches grow exponentially in time. Interestingly the high-level approach only grows linear in *size* (number of nodes). This is due to the size of the guard of the *evaluate* transition increasing exponentially (n choose 2).

Overall, the symbolic approach is the clear winner of the Mastermind benchmark. The low-level approach can only barely compete for the smallest instances of the problem.

We also observe that, in comparison to other parameters, the performance of the high-level approach drops when $n = m$. There currently is no obvious explanation for this. This could be a quirk of the SMT solvers, or alternatively, the formulas may be inherently more challenging to solve in this particular case. The low-level approach is not impacted negatively by this case, but still much slower than the symbolic approach, as seen in lines with parameters $(1, 3, 3)$, $(1, 4, 4)$, $(2, 3, 3)$, $(2, 4, 4)$, and in the last four lines.

4.7 Related Work

Unfoldings of P/T Petri nets are first introduced by Nielsen et al. in [NPW81]. Engelfriet generalizes this concept in [Eng91] by introducing the notion of *branching processes*, and shows that the unfolding of a net is its maximal branching process. A comprehensive overview of unfoldings is given in [EH08]. In [McM95], McMillan presents an algorithm to compute a complete finite prefix of the unfolding of a given P/T Petri net. The ERV-algorithm, leading to comparably small complete finite prefixes of the unfolding, is presented in [ERV02]. This algorithm is generalized in [KKV03], where canonical prefixes of unfoldings are introduced.

In [KK03], Khomenko and Koutny describe how to construct the unfolding of the expansion of a high-level Petri net (given in the formalism of "M-nets" [BFF+95]) without first expanding it. In [EHP+02], Ehrig et al. define processes of high-level Petri nets. This work is continued in [EHGP08] for results about composition, equivalence and independence of high-level net processes. In [CJ04], Chatain and Jard define *symbolic branching processes* and *unfoldings* of high-level Petri nets. The work on the latter is built upon in [CF10] by Chatain and Fabre, where they consider so-called "puzzle nets". Based on the construction of a symbolic unfolding, in [CJ06; Cha06], complete finite prefixes of safe time Petri nets are constructed.

Unfoldings of unbounded P/T Petri nets (i.e., with infinitely many markings) have been investigated in [AIN00; DJN04], and in [HST07] concurrent well-structured transition systems with infinite state space are unfolded.

4.A Appendix

4.A.1 Examples of Adequate Orders

We show that the adequate order used in [McM95], as well as the orders $\prec_E$ and $\prec_F$ treated in [ERV02], when lifted to the symbolic unfolding, are still adequate orders. In particular we show that $\prec_F$ is a total adequate order on the symbolic unfolding, limiting the size of the later constructed finite prefix. The definition of these orders does not change, so we take most of the following notation and formulations directly from [ERV02].

The orders $\prec_M$ and $\prec_E$. The order $\prec_M$ used in [McM95] is defined by $C_1 \prec_M C_2 :\Leftrightarrow |C_1| < |C_2|$. It is trivial to see that $\prec_M$ satisfies i) and ii) from Def. 4.13. Since $\varphi^2_{1,Q}$ is a injective, we have $|\varphi^2_{1,Q}(Q)| = |Q|$, which yields iii).

For a high-level Petri net N, let $\ll$ be an arbitrary total order on the transitions of N. Given a set E' of events in the unfolding of N, let $\mathbf{p}(E')$ be that sequence of transitions which is ordered according to $\ll$, and contains each transition t as often as there are events in E' with label t. We say $\mathbf{p}(E_1) \ll \mathbf{p}(E_2)$ if $\mathbf{p}(E_1)$ is lexicographically smaller than $\mathbf{p}(E_2)$ with respect to the order $\ll$.

The order $\prec_E$ is then defined as follows: let C_1, C_2 be two configurations of the symbolic unfoldings of a high-level Petri net. $C_1 \prec_E C_2$ holds if either $|C_1| < |C_2|$, or $|C_1| = |C_2|$ and $\mathbf{p}(C_1) \ll \mathbf{p}(C_2)$. The proof that $\prec_E$ is an adequate order works exactly as in [ERV02]:

It is easy to show that $\prec_E$ is a well-founded partial order implied by inclusion. We now show that $\prec_E$ is preserved by finite extensions. As already mentioned above, $|Q| = |\varphi^2_{1,Q}(Q)|$. Additionally, we have $\mathbf{p}(Q) = \mathbf{p}(\varphi^2_{1,Q}(Q))$, since $\varphi^2_{1,Q}$ preserves the labeling of events.

Assume $C_1 \prec_E C_2$. If $|C_1| < |C_2|$, then $|C_1 \oplus Q| < |C_2 \oplus \varphi^2_{1,Q}(Q)|$. If $|C_1| = |C_2|$ and $\mathbf{p}(C_1) \ll \mathbf{p}(C_2)$, then $|C_1 \oplus Q| = |C_2 \oplus \varphi^2_{1,Q}(Q)|$ and, by the properties of the lexicographic order, $\mathbf{p}(C_1 \oplus Q) \ll \mathbf{p}(C_2 \oplus \varphi^2_{1,Q}(Q))$.

The Total Adequate Order $\prec_F$. The *Foata normal form FC* of a configuration C is obtained by starting with FC empty, and iteratively deleting the set $Min(C)$ from C and appending it to FC, until C is empty.

Given two configurations C_1, C_2, we can compare their Foata normal forms $FC_1 = C_{11} \dots C_{1n_1}$ and $FC_2 = C_{21} \dots C_{2n_2}$ with respect to the order $\ll$ by saying $FC_1 \ll FC_2$ if there exists $i \leq i \leq n_1$ such that $\mathbf{p}(C_{1j}) = \mathbf{p}(C_{2j})$ for every $1 \leq j < i$, and $\mathbf{p}(C_{1i}) \ll \mathbf{p}(C_{2i})$.

Definition 4.36 (Order $\prec_F$). let C_1 and C_2 be two configurations of the symbolic unfolding of a high-level Petri net. $C_1 \prec_F C_2$ holds if

- $|C_1| < |C_2|$, or
- $|C_1| = |C_2|$ and $\mathbf{p}(C_1) \ll \mathbf{p}(C_2)$, or
- $\mathbf{p}(C_1) = \mathbf{p}(C_2)$ and $FC_1 \ll FC_2$.

$\triangleleft$

We prove that $\prec_F$ is a total adequate order. In the proof, (a) – (c) are taken directly from [ERV02], with small adaptations due to the high-level formalism. While the ideas from (d) also come directly from [ERV02], we have work with the monomorphism $\varphi^2_{1,Q}$ instead of the isomorphism I^2_1, and the new definition of adequate order. This is where the only deviation from [ERV02] happens.

Let $\beta = (O, h)$ be the symbolic unfolding of $N = (\mathcal{N}, \mathcal{M}_0)$.

(a) $\prec_F$ is a well-founded partial order.

This follows immediately from the fact that $\prec_E$ is a well-founded partial order as is the lexicographic order on transition sequences of some fixed length.

(b) $C_1 \subset C_2$ implies $C_1 \prec_F C_2$.

This is obvious, since $C_1 \subset C_2$ implies $|C_1| < |C_2|$.

(c) $\prec_F$ is total.

Assume that C_1 and C_2 are two incomparable configurations under $\prec_F$, i.e., $|C_1| = |C_2|$, $\mathbf{p}(C_1) = \mathbf{p}(C_2)$, and $\mathbf{p}(FC_1) = \mathbf{p}(FC_2)$. We prove $C_1 = C_2$ by induction on the common size $k = |C_1| = |C_2|$.

The base case $k = 0$ gives $C_1 = C_2 = \emptyset$, so assume $k > 0$.

We first prove $Min(C_1) = Min(C_2)$. Aiming a contradiction, assume w.l.o.g. that $e_1 \in Min(C_1) \setminus Min(C_2)$. Since $\mathbf{p}(Min(C_1)) = \mathbf{p}(Min(C_2))$, $Min(C_2)$ contains an event e_2 s.t. $h(e_1) = h(e_2)$. Since $\rightarrow Min(C_1)$ and $\rightarrow Min(C_2)$ are subsets of B_0, and all conditions of B_0 carry different labels, we have $\rightarrow e_1 = \rightarrow e_2$, and thus, $pre(e_1) = pre(e_2)$. This contradicts the definition of symbolic branching processes.

Since $Min(C_1) = Min(C_2)$, both $C_1 \setminus Min(C_1)$ and $C_2 \setminus Min(C_1)$ are configurations of the branching process $\Uparrow Min(C_1)$ of $(\mathcal{N}, \text{Marks}(Min(C_1)))$, and they are incomparable under $\prec_F$ by construction. Since the common size of $C_1 \setminus Min(C_1)$ and $C_2 \setminus Min(C_1)$ is strictly smaller than k, we can apply the induction hypothesis and conclude $C_1 = C_2$.

(d) $\prec_F$ is preserved by finite extensions.

Take two finite configurations C_1 and C_2, let Q be a finite suffix of C_1, and let $M \in \text{Marks}(C_1) \cap \text{Marks}(C_2)$ such that $C_1[\![M]\!]Q$. We have to show that $C_1 \prec C_2$ implies $C_1 \oplus Q \prec C_2 \oplus \varphi^2_{1,Q}(Q)$.

First, notice that we can assume $Q = \{e\}$: For $e \in Min(Q)$ we have from $C_1[\![M]\!]Q$ that $\exists \theta \in \Theta(C_1 \oplus Q) : \text{mark}(C_1.\theta|_{Var_{C_1 \cup \{\perp\}}}) = M$. Thus, for M' s.t. $M[h(e).\sigma\rangle M'$ with $\sigma = \theta \circ [v \mapsto v_e]_{v \in Var(e)}$, we have that $M' \in \text{Marks}(C_1 \oplus \{e\}) \cap \text{Marks}(C_2 \oplus \{\varphi^2_{1,Q}(e)\})$ and $(C_1 \oplus \{e\})[\![M']\!](Q \setminus \{e\})$.

Second, the cases $|C_1| < |C_2|$ and $C_1 \prec_E C_2$ in (i), (and the respective cases $|C_2| < |C_1|$ and $C_2 \prec_E C_1$ in (ii)) are easy (shown above). Hence, assume $|C_1| = |C_2|$ and $\mathbf{p}(C_1) = \mathbf{p}(C_2)$.

Third, we show that under these two assumptions e is a minimal event of $C'_1 := C_1 \cup \{e\}$ if and only if $\varphi^2_{1,Q}(e)$ is a minimal event of $C'_2 := C_2 \cup \{\varphi^2_{1,Q}(e)\}$. Let e be minimal in C'_1, i.e., the transition $h(e)$ can be fired in a mode in one initial marking. Let $p \in {\to}h(e)$; then no condition in ${\to}C \cup C{\to}$ is labeled p, since these conditions would be concurrent to the p-labeled condition in ${\to}e$, contradicting that $(\mathcal{N}, \mathcal{M}_0)$ is safe. Thus, C_1 contains no event e' with $p \in {\to}h(e')$, and the same holds for C_2, since $\mathbf{p}(C_1) = \mathbf{p}(C_2)$. Therefore, the conditions in $\mathrm{cut}(C_2)$ with label in ${\to}h(e)$ are minimal conditions of β, and $\varphi^2_{1,Q}(e) = e$ is minimal in C'_2. The reverse implication holds analogously, since about C_1 and C_2 we have only used the hypothesis $\mathbf{p}(C_1) = \mathbf{p}(C_2)$.

With this knowledge, we now show the implication. Assume $C_1 \prec_F C_2$. We show $C'_1 \prec_F C'_2$.

If $\mathit{Min}(C_1) \prec_E \mathit{Min}(C_2)$, then we now see $\mathit{Min}(C'_1) \prec_E \mathit{Min}(C'_2)$, hence $\mathbf{p}(FC'_1) \ll \mathbf{p}(FC'_2)$ and so we are done. If $\mathbf{p}(\mathit{Min}(C_1)) = \mathbf{p}(\mathit{Min}(C_2))$ and $e \in \mathit{Min}(C'_1)$, then

$$C'_1 \setminus \mathit{Min}(C'_1) = C_1 \setminus \mathit{Min}(C_1) \prec_F C_2 \setminus \mathit{Min}(C_2) = C'_2 \setminus \mathit{Min}(C'_2),$$

hence $C'_1 \prec_F C'_2$. Finally, if $\mathbf{p}(\mathit{Min}(C_1)) = \mathbf{p}(\mathit{Min}(C_2))$ and $e \notin \mathit{Min}(C'_1)$, we again argue that $\mathit{Min}(C_1) = \mathit{Min}(C_2)$ and that, hence, $C \setminus \mathit{Min}(C_1)$ and $C_2 \setminus \mathit{Min}(C_1)$ are configurations of the branching process ${\Uparrow}\mathit{Min}(C_1)$ of $(\mathcal{N}, \mathrm{Marks}(\mathit{Min}(C_1)))$. With an inductive argument we get $C'_1 \setminus \mathit{Min}(C'_1) \prec_F C'_2 \setminus \mathit{Min}(C'_2)$ and are also done in this case.

4.A.2 More Symbolic and Low-Level Unfoldings

In this appendix we present unfoldings omitted in the main body of the paper.

Low-level Unfolding of Fork And Join. Since the symbolic unfolding of any Fork And Join is structurally equal to the net itself we do not display it here. Fig. 4.11 shows the low-level unfolding with $(n+2)(m+1)^n + 1$ nodes for a Fork And Join.

Symbolic Unfolding of Water Pouring Puzzle. Figure 4.12 shows the complete finite prefix of the symbolic unfolding of the Water Pouring Puzzle from Fig. 4.6 with $n = [5, 3]$ and $m = 4$, calculated by our implementation ColorUnfolder [Pan] of the generalized ERV-algorithm, Alg. 1. The figure is automatically produced from the output of ColorUnfolder. Cut-off events are marked by a red border. Additionally marked by a red border are the events representing the added *goal* transition that checks the target states. Since the net is mode-deterministic, the symbolic unfolding's skeleton coincides with the low-level unfolding. Thus, the skeleton of the complete finite prefix in Figure 4.12 coincides with the complete finite prefix of the low-level unfolding generated by the original ERV-algorithm from [ERV02].

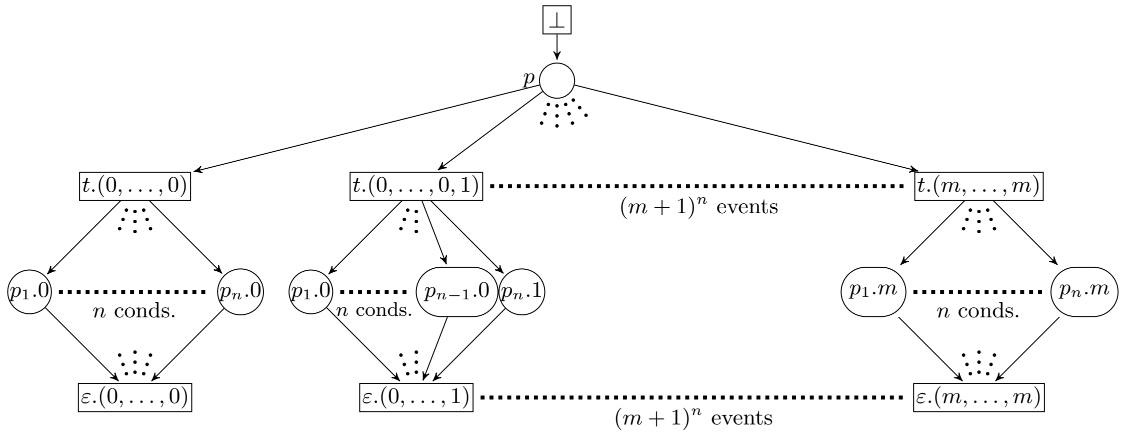


Figure 4.11: The symbolic unfolding of a Fork And Join, depending on the parameters m and n.

Low-Level Unfolding of Mastermind. Figure 4.13 shows a prefix of the symbolic unfolding of the Mastermind net from Fig. 4.8 with 6 available colors, a code of length 4, and 12 possible attempts. Combinatorial arguments give that the low-level unfolding has more than 10^{37} nodes, so we do not present it here.

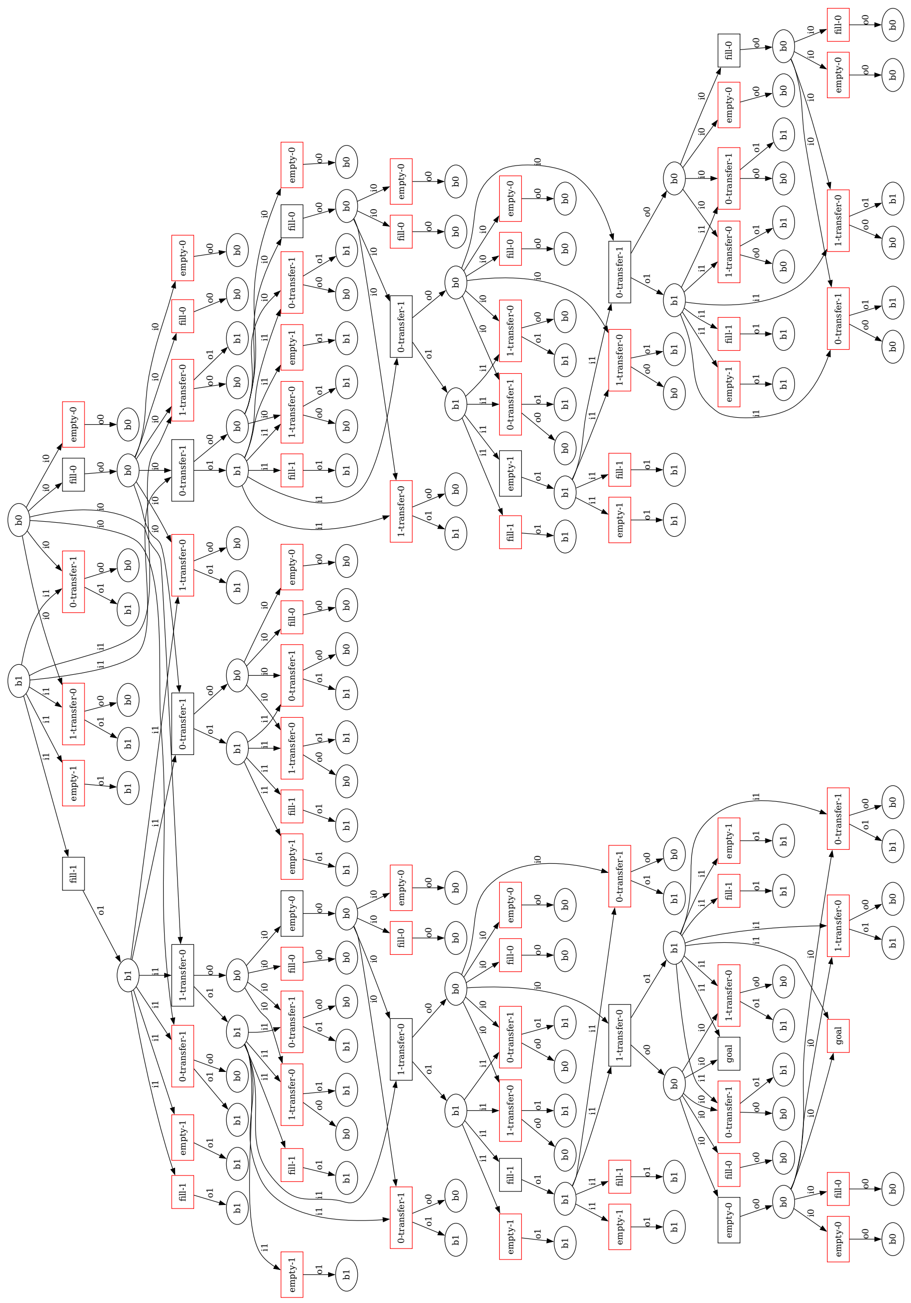


Figure 4.12: A prefix of the symbolic unfolding of the Water Pouring Puzzle for $n = [3, 5]$ and $m = 4$, produced by ColorUnfolder.

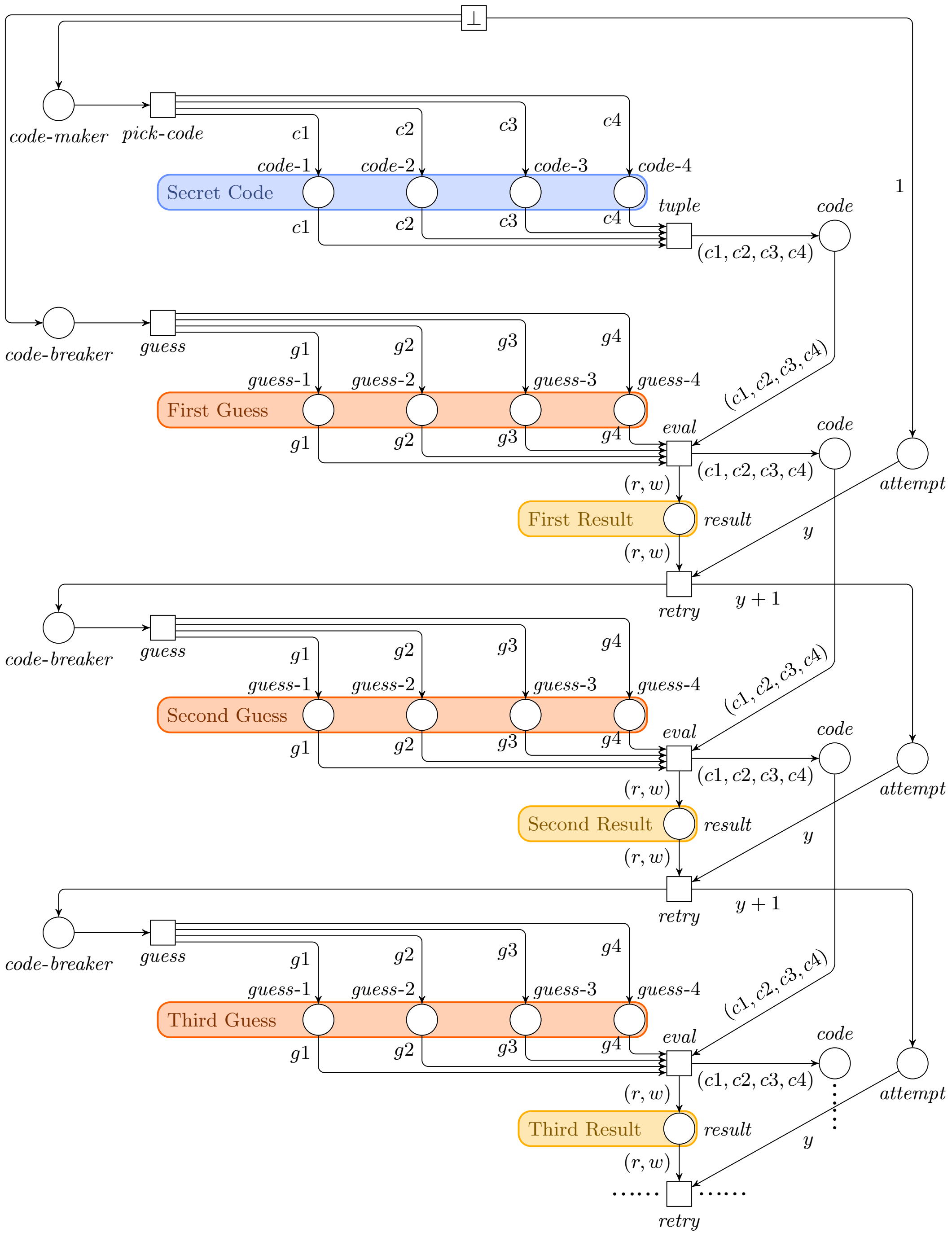


Figure 4.13: A prefix of the symbolic unfolding of the Mastermind net from Fig. 4.8.

Chapter 5

Outlook: Symbolic Strategies for High-Level Petri Games

Contents

In this thesis, we have now investigated

- a reduction of proper high-level Petri games to a symbolic two-player game in Chapter 3, and
- complete finite prefixes of the symbolic unfolding of safe high-level Petri nets in Chapter 4.

However, a direct connection between the two concepts is still missing. In this chapter we give a brief outlook on how to combine (complete finite prefixes of) symbolic unfoldings with high-level Petri games. We start with a proposal for a definition of symbolic strategies and then give a glimpse into the possibilities of synthesizing such strategies. The work in this chapter is unpublished.

The envisioned benefit of symbolic strategies over low-level counterparts is the ability to abstractly describe the correct behavior of system players in a high-level Petri game. Notably, we aspire to articulate the significance of individual events in a system player's causal history for its decisions regarding the permission or prohibition of specific transitions, respectively their modes.

5.1 Defining Symbolic Strategies

In Sec. 2.2, Petri game strategies are defined as subprocesses of the unfolding satisfying certain conditions. We show that a direct translation into the symbolic setting, i.e., trying to define symbolic strategies as subprocesses (prefixes) of the symbolic unfolding, does not yield the desired result. We instead give an alternative, equivalent definition of low-level strategies that can be lifted to the level of symbolic unfoldings more easily.

We define the symbolic unfolding of a high-level Petri game G as the unfolding $\Upsilon(N(G))$ of the game's underlying high-level Petri net and, as in the P/T case, denote by B_{S}, B_{E}, $B_{\circledast}$ the system-, environment-, and special conditions.

5.1.1 Insufficiency of Prefixes

Recall that in Sec. 2.2 strategies of P/T Petri games are defined as subprocesses (and thus, prefixes) of the (low-level) unfolding. So a first idea is to define symbolic strategies as prefixes of the symbolic unfolding. However, additionally to being a subprocess of the unfolding, a low-level strategy must satisfy certain properties, namely determinism and justified refusal. In case of a Petri game with safety objective, a winning strategy must additionally be deadlock-free. These properties, however, are hard to achieve with the definition of prefixes of the symbolic unfolding from Sec. 2.3.1. In particular, prefixes of the symbolic unfolding inherently often do not satisfy the determinism property of Petri game strategies.

We investigate this on the example Communicating Copycats from Fig. 2.6a. We call this game G and continue to use it as a running example for this section. The symbolic unfolding $\Upsilon(G)$ of Communicating Copycats is presented in Fig. 2.8. Recall that for this example a(n informally described) winning strategy for the system players is to communicate once after being generated and, equipped with the gained knowledge, choose the correct values for x resp. y in the firing of mim_{A} resp. mim_{B} that enable the firing of end. For $m = 2$ (determining $Col = \{0, 1, 2\}$) the winning low-level strategy of the game's expansion (the P/T Petri game $\mathrm{Exp}(G)$ in Fig. 2.4) is presented in Fig. 2.5.

The minimal prefix $\xi = (O^{\xi}, \pi^{\xi})$ of the symbolic unfolding that contains this strategy is presented in Fig. 5.1. "Contained" is meant in the sense that the strategy from Fig. 5.1 is a prefix of the expanded branching process $\mathrm{Exp}_{\mathrm{O}}(\xi)$ (cp. Sec. 4.3), and "minimal" is meant w.r.t. the prefix relation. This prefix ξ however, is not deterministic: recall the determinism condition b) in the P/T case from p. 17:

$$\forall \mathsf{K} \in \mathcal{R}(\mathsf{O}^{\xi})\ \forall \mathsf{b} \in \mathsf{K} \cap \mathsf{B}_{\mathrm{S}}\ \exists^{\leq 1} \mathsf{e} \in \mathsf{E}^{\xi} : \mathsf{b} \in pre(\mathsf{e}) \wedge \mathsf{K}[\mathsf{e}\rangle.$$

When we translate translate this into the language of symbolic unfoldings using Prop. 4.6 and Prop. 4.7, we arrive at

$$\forall K \in \mathcal{R}(O^{\xi})\ \forall (b, c) \in K \cap (B_{\mathrm{S}} \times Col)\ \exists^{\leq 1} (e, \sigma) : \begin{cases} e \in E^{\xi} \wedge \sigma \in \Sigma(e) \wedge \\ (b, c) \in pre(e, \sigma) \wedge K[e, \sigma\rangle \end{cases} . \quad (5.1)$$

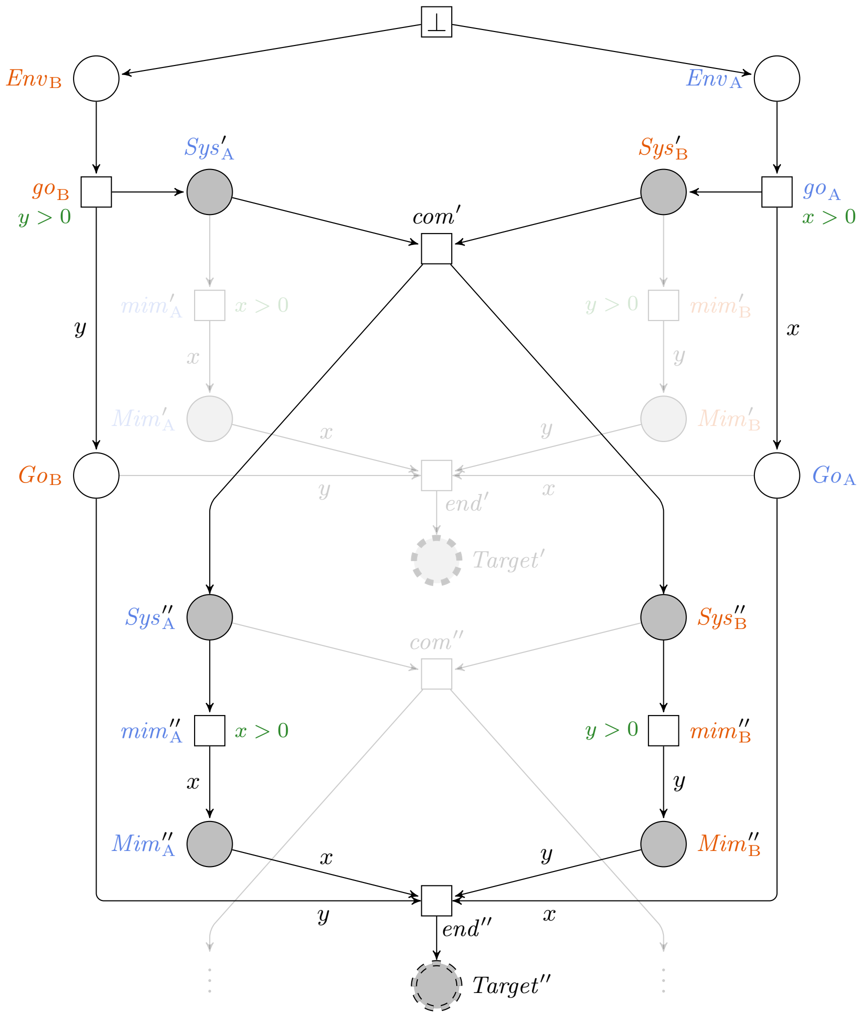


Figure 5.1: A minimal prefix ξ of the symbolic unfolding "containing" the winning strategy from Fig. 2.5.

The prefix ξ from Fig. 5.1 does *not* satisfy this property: it is violated by every cut K containing a $(Sys''_A, 0)$ or $(Sys''_B, 0)$. For such cuts K we have

$$K[mim''_A, \{x \leftarrow 1\}\rangle \wedge K[mim''_A, \{x \leftarrow 2\}\rangle, \text{ resp. } K[mim''_B, \{y \leftarrow 1\}\rangle \wedge K[mim''_B, \{y \leftarrow 2\}\rangle.$$

An example for such a cut is $\{(Sys''_A, 0), (Sys''_B, 0), (Go_A, 1), (Go_B, 2)\}$, which is reached by the firing sequence $((go_B, \{y \leftarrow 2\})(go_A, \{x \leftarrow 1\})(com', \{\}))$

The arguments above are analogous to saying that ξ cannot be a strategy because $\mathrm{Expo}_O(\xi)$ is not deterministic and therefore not a (low-level) Petri game strategy. $\mathrm{Expo}_O(\xi)$ contains all instances of both mim''_A and mim''_B. In a prefix of the symbolic unfolding we cannot express that the "allowed" modes of mim''_A depend on the mode in which go_A fired, and analogously for mim''_B and go_B.

5.1.2 A Different View at Low-Level Strategies

Before we get to a proposal of how high-level strategies can be defined, we look at an alternative but equivalent definition of low-level Petri game strategies. The idea behind Petri game strategies is that every system player, depending on its causal history and place, allows a set of transitions in the net that depend on the player's participation, i.e., transitions with the respective place in their preset. The conditions in the unfolding precisely represent all place-history combinations. Thus, in the low-level case, Petri game strategies can concisely be defined as subprocesses of the unfolding.

An alternative formal definition of low-level Petri game strategies would be a function ζ mapping every condition $\mathsf{b} \in \mathsf{B}$ in the unfolding to a set $\zeta(\mathsf{b}) \subseteq \{\mathsf{t} \in \mathsf{T} \mid \pi(\mathsf{b}) \in \mathit{pre}(\mathsf{t})\}$ of transitions that the player with the corresponding causal history allows. For environment conditions $\mathsf{b} \in \mathsf{B}_{\mathrm{E}}$, we would always have $\zeta(\mathsf{b}) = \{\mathsf{t} \in \mathsf{T} \mid \pi(\mathsf{b}) \in \mathit{pre}(\mathsf{t})\}$, representing that a strategy for the system players cannot restrict the environments behavior. By such a definition, "justified refusal", i.e., the requirement that players only allow or forbid transitions in the net, and only depending on their history, would directly be satisfied. For any co-set B' in the unfolding (and in particular, cuts), we can then define the set of allowed transitions by $\zeta(\mathsf{B}') = \bigcap_{\mathsf{b} \in \mathsf{B}'} \zeta(\mathsf{b})$. For determinism, we would need that in no cut of the unfolding, there are two fireable events that share a system condition from the cut in their preset and their labels are both allowed, formally

$$\forall \mathsf{K} \in \mathcal{R}(\mathsf{U})\, \forall \mathsf{b} \in \mathsf{K} \cap \mathsf{B}_{\mathrm{S}}\, \exists^{\leq 1} \mathsf{e} \in \mathsf{E} : \mathsf{K}[\mathsf{e}\rangle \wedge \mathsf{b} \in \mathit{pre}(\mathsf{e}) \wedge \pi(\mathsf{e}) \in \zeta(\mathsf{K}). \tag{5.2}$$

To facilitate referencing this alternative definition, we call such functions *strategy functions*:

Definition 5.1 (Petri Game Strategy Function). Let G be a P/T Petri game with transitions T and unfolding $\Upsilon(\mathsf{G}) = (\mathsf{U}, \pi)$ having the conditions B events E. A *strategy function* in G is a function $\zeta : \mathsf{B} \to \mathbb{P}(\mathsf{T})$ satisfying

- $\forall \mathsf{b} \in \mathsf{B}_{\mathrm{S}} : \zeta(\mathsf{b}) \subseteq \{\mathsf{t} \in \mathsf{T} \mid \pi(\mathsf{b}) \in \mathit{pre}(\mathsf{t})\}$,
- $\forall \mathsf{b} \in \mathsf{B}_{\mathrm{E}} : \zeta(\mathsf{b}) = \{\mathsf{t} \in \mathsf{T} \mid \pi(\mathsf{b}) \in \mathit{pre}(\mathsf{t})\}$, and
- Condition (5.2). $\lhd$

Example 5.2. We show how the strategy ξ from Fig. 2.5 (defined as a subprocess) would be defined as a strategy function ζ on B_{S}, where B_{S} are the system conditions in the unfolding. In Fig. 2.5 we see that the "first layer" of system conditions, i.e., $\mathit{Sys}^1_{\mathrm{A}}$, $\mathit{Sys}^2_{\mathrm{A}}$, $\mathit{Sys}^1_{\mathrm{B}}$, and $\mathit{Sys}^2_{\mathrm{B}}$, all only participate in instances of the transition *com*. As a function, this would read as

$$\zeta(\mathit{Sys}^1_{\mathrm{A}}) = \zeta(\mathit{Sys}^2_{\mathrm{A}}) = \zeta(\mathit{Sys}^1_{\mathrm{B}}) = \zeta(\mathit{Sys}^2_{\mathrm{B}}) = \{\mathit{com}\}.$$

The output conditions of the *com* events are $\mathit{Sys}^{ij}_{\mathrm{X}}$ with $\mathrm{X} \in \{\mathrm{A}, \mathrm{B}\}$ and $i, j \in \{1, 2\}$. Recall that in this example we chose the names of conditions such that their superscripts indicate the players knowledge (cp. p. 21): the condition $\mathit{Sys}^{ij}_{\mathrm{X}}$ represents that the player on place Sys_X knows that Environment A fired $go_{\mathrm{A}i}$ and Environment B fired $go_{\mathrm{B}j}$. To reach the target place, the player on Sys_A must mimic Environment A and the player on Sys_B must mimic Environment B. Thus, as a function ζ, the strategy reads as

$$\forall i, j \in \{1,2\} : \quad \zeta(Sys^{ij}_\mathrm{A}) = \{mim_{\mathrm{A}i}\} \quad \wedge \quad \zeta(Sys^{ij}_\mathrm{B}) = \{mim_{\mathrm{B}j}\}.$$

For all other system conditions b in the unfolding we can define $\zeta(\mathsf{b}) = \emptyset$, since they are never reached when only events representing allowed transitions are fired. $\lhd$

Definition 5.1 is *equivalent* to the definition of Petri game strategies from Sec. 2.2 in the sense that

- Petri game strategies in the form of subprocesses can be transformed into strategy functions by a mapping `subprocess2function`,
- strategy functions can be transformed into Petri game strategies in the form of subprocesses by a mapping `function2subprocess`, and
- for all Petri game strategies ξ in the form of a subprocess we have

  $$\texttt{function2subprocess}(\texttt{subprocess2function}(\xi)) = \xi,$$

  and for all strategy functions ζ we have

  $$\texttt{subprocess2function}(\texttt{function2subprocess}(\zeta)) = \zeta$$

  on the conditions that can be reached by firing only events representing allowed transitions.

The formal proof is not given in this outlook chapter and left to the reader. The mapping `subprocess2function` was implicitly presented in Example 5.2. The mapping `function2subprocess` can be defined such that it iteratively constructs a subprocess. It starts with K_0, and repeatedly adds all possible events $\mathsf{e} = (\mathsf{t}, \mathsf{X})$ such that $\forall b \in \mathsf{X} : \mathsf{t} \in \zeta(\mathsf{b})$, and corresponding output conditions.

5.1.3 Proposing a Definition of Symbolic Strategies

We now use a similar intuition to define symbolic strategies, by lifting Def. 5.1 to the formalism of symbolic unfoldings.

A high-level condition b represents a system player on place $\pi(b)$ knowing that all events in $[\mathbf{e}(b)]$ fired, and also *in which mode* they fired. This means the system players knows for all events $e \in [\mathbf{e}(b)]$, for all variables $v \in \mathit{Var}(e)$, what value v was assigned by the mode in which e fired. Additionally, the system player knows the initial marking. This corresponds to knowing the mode in which the special event $\perp$ fired. We can therefore say the knowledge of the system player corresponds exactly to knowing the values of $\mathit{Var}_{[\mathbf{e}(b)]\cup\{\perp\}} = \{v_e \mid e \in [\mathbf{e}(b)] \cup \{\perp\}, v \in \mathit{Var}(e)\}$.

We now discuss how a system player on b can allow or forbid transitions in *different modes*. It can describe in which modes it allows transitions t that take a color from $\pi(b)$

to fire. The idea is now that, for every condition b in the unfolding, the system player gives constraints on the set $\mathit{Var}(t)$ of variables for every transition t with $\pi(b) \to t$, depending on its knowledge, i.e., thevalues of $\mathit{Var}_{[\mathsf{e}(b)]\cup\{\perp\}}$.

This is expressed via a *predicate* $\zeta(b)_t$ over all those variables $\mathit{Var}_{[\mathsf{e}(b)]\cup\{\perp\}} \cup \mathit{Var}(t)$ for every such transition t. If this predicate is unsatisfiable then this corresponds to no mode being enabled ever for this transition. Similarly, if the predicate $\zeta(b)_t$ is equivalent to *true* then the condition allows all modes of this transition, independently of its knowledge. The remaining case is that the predicate $\zeta(b)_t$ is true for some but not all assignments of colors to the variables from $\mathit{Var}_{[\mathsf{e}(b)]\cup\{\perp\}} \cup \mathit{Var}(t)$. In this case the predicate explicitly relates the allowed modes of transitions to the values of variables in the condition's causal history. The predicate can also be seen as a function that maps every variable assignment on $\mathit{Var}_{[\mathsf{e}(b)]\cup\{\perp\}}$ (i.e., every possible history) to a set of variable assignments on $\mathit{Var}(t)$ (i.e., the allowed modes of t) such that the predicate evaluates to true under the corresponding combinations of variable assignments.

When interpreting P/T Petri nets as high-level nets, the alternate definition of low-level strategies from Sec. 5.1.2 corresponds to the predicate $\zeta(b)_t$ of a transition t being *true* if t is in the set of allowed transitions, and *false* if t is not in the set of allowed transitions for a given condition b.

Definition 5.3 (Symbolic Petri Game Strategy). Let G be a high-level Petri game with transitions T with unfolding $\Upsilon = (U, \pi)$ that has the system conditions B and events E. A symbolic strategy for the system players in G is a function ζ mapping each $b \in B$ to a family of predicates $\zeta(b) = \{\zeta(b)_t \mid \pi(b) \to t\}$ such that every $\zeta(b)_t$ is a predicate over the variables $\mathit{Var}_{[\mathsf{e}(b)]\cup\{\perp\}} \cup \mathit{Var}(t)$, satisfying $\forall b \in B_{\mathrm{E}}\, \forall t \in T : \pi(b) \to t \Rightarrow \zeta(b)_t = \mathit{true}$ and

$$\forall C \in \mathcal{C}(\Upsilon)\, \forall \theta \in \Theta(C)\, \forall b \in B_{\mathrm{S}}\, \exists^{\leq 1}(e, \sigma) : \begin{cases} e \in E \wedge \sigma \in \Sigma(e) \wedge \mathrm{cut}(C, \theta)[e, \sigma\rangle \wedge b \to e \\ \wedge\, \zeta(b)_{\pi(e)}[\theta][\sigma] \equiv \mathit{true}. \end{cases}$$

$\lhd$

The condition at the end describes determinism of the strategy: the quantified C and θ describe all reachable cuts together with all possible histories of variable assignments (analogous to K from (5.2)). As in (5.2), the quantified b describes all system conditions, and (e, σ) are the event-mode combinations. The formula describes that no two such fireable event-mode combinations that share a system condition in their preset are both allowed in the same cut with the same history. Since modes are either allowed or forbidden for every fixed history, Justified Refusal is trivially satisfied.

We demonstrate for our running example Communicating Copycats a symbolic strategy corresponding to the low-level strategy from Fig. 2.5:

Example 5.4. Consider again the symbolic unfolding (already presented in Fig. 2.8 and recalled as base for Fig. 5.1) of the high-level Petri game Communicating Copycats from Fig. 2.6a. By Def. 5.3, a symbolic strategy for the system players in this high-level Petri game equips every condition in the symbolic unfolding with a family of predicates. Figure 5.2 illustrates the symbolic strategy describing the procedure of "first communicate, then mimic the environment" that was informally described on p. 20 and formally presented for the low-level case in Fig. 2.5. For every system condition $b \in B_S$, the family of predicates $\zeta(b) = \{\zeta(b)_t \mid \pi(b) \to t\}$ is denoted in red next to the condition in the form of listing "$t\colon \zeta(b)_t$" for the t such that $\pi(b) \to t$.

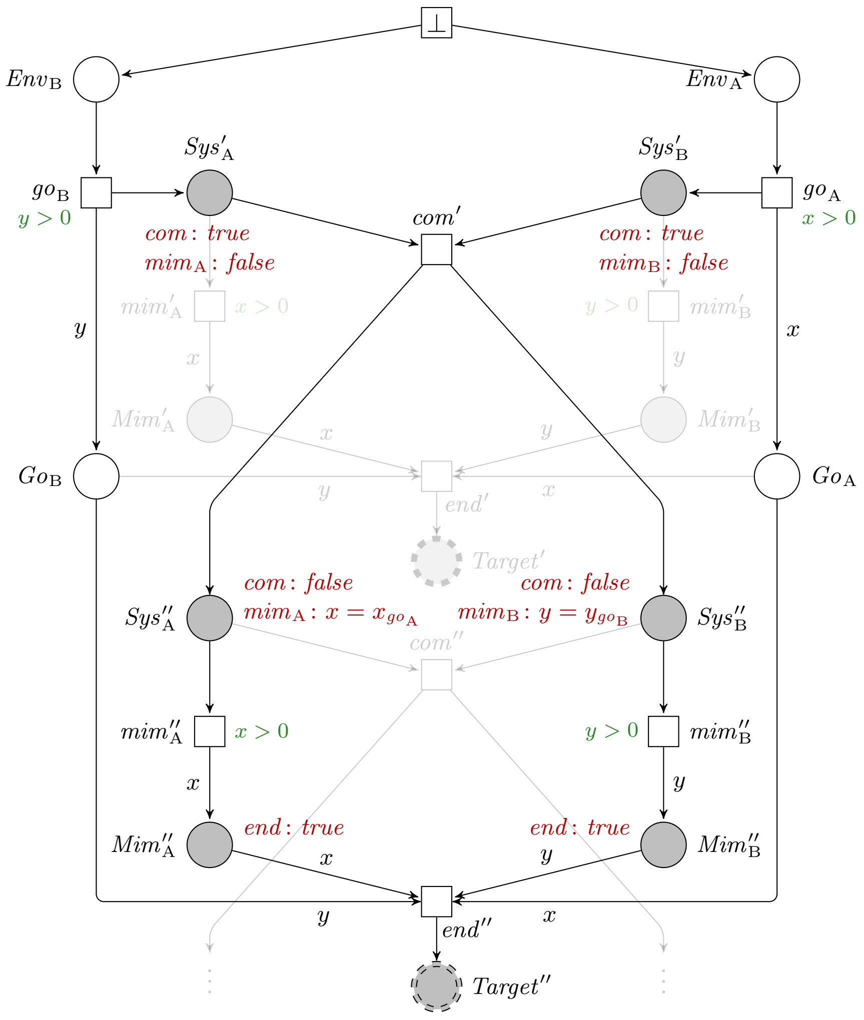


Figure 5.2: An illustration of a symbolic strategy corresponding to the low-level strategy in Fig. 2.5.

Consider condition Sys'_A. The predicates of the strategy are $\zeta(Sys'_A)_{com} = true$ and $\zeta(Sys'_A)_{mim_A} = false$. By the above discussion this means that a system player on this condition, independently of the history, allows transition *com* to fire in all modes of *com* (in this case, the only mode), and forbids transition mim_A to fire in any of its modes. Analogously, a system player on condition Sys'_B allows *com* to fire but forbids mim_B to fire in any mode.

The events in the unfolding which are π-labeled by *com*, mim_A or mim_B and involve

Sys'_A or Sys'_B are com', mim'_A, and mim'_B. Since Sys'_A resp. Sys'_B forbid mim_A resp. mim_B to fire in any mode, the events mim'_A and mim'_B may never fire in the described strategy. we gray out these events in the figure, as well as every node causally dependent on them.

The event com, however, *can* fire, and after this we arrive at the conditions $\mathit{Sys}''_\mathrm{A}$ and $\mathit{Sys}''_\mathrm{B}$. Independent of the history (i.e., the values of y and x in the modes of firing go_A and go_B), the two conditions forbid com to fire by $\zeta(\mathit{Sys}''_\mathrm{A})_{\mathit{com}} = \mathit{false}$ and $\zeta(\mathit{Sys}''_\mathrm{B})_{\mathit{com}} = \mathit{false}$. We again gray out the event com'' and all nodes that are structurally dependent on it.

Now comes the crucial part of the strategy: it is $\zeta(\mathit{Sys}''_\mathrm{A})_{\mathit{mim}_\mathrm{A}} = (x = x_{\mathit{go}_\mathrm{A}})$. This means that a system player on $\mathit{Sys}''_\mathrm{A}$ allows mim_A to fire only in the modes where x is assigned the same value that the mode of go_A assigned to x. This corresponds to "mimic the environment", and is not expressible by a only prefix of the symbolic unfolding. Analogously, a system player on $\mathit{Sys}''_\mathrm{B}$ allows mim_B to fire only in the modes where y is assigned the same value that the mode of go_B assigned to y.

The corresponding events are $\mathit{mim}''_\mathrm{A}$ and $\mathit{mim}''_\mathrm{B}$ with respective output conditions $\mathit{Mim}''_\mathrm{A}$ and $\mathit{Mim}''_\mathrm{B}$. Players on these conditions allow end to fire in any mode. The only corresponding event is end'' with output condition Target''. The set $\{t \in T \mid \pi(\mathit{Target}'') \to t\}$ is empty, so we trivially have $\zeta(\mathit{Target}'') = \{\}$ and therefore list nothing next to the condition.

By deleting from the symbolic unfolding the branches that are dependent on events which are (independently of the variables) forbidden, we can depict the strategy as a structure that could be described as a "refined prefix" of the symbolic unfolding, as seen in Fig. 5.2. We now demonstrate exemplarily for two configurations C why this strategy is deterministic, i.e., satisfying the formula from Def. 5.3.

The first example is the configuration $C = \{\mathit{go}_\mathrm{B}, \mathit{go}_\mathrm{A}\}$. Consider the instantiation $\theta = \{y_{\mathit{go}_\mathrm{B}} \leftarrow 1, x_{\mathit{go}_\mathrm{A}} \leftarrow 2\} \in \Theta(C)$. The corresponding cut is $\mathrm{cut}(C, \theta) = \{(\mathit{Go}_\mathrm{B}, 1), (\mathit{Go}_\mathrm{A}, 2), (\mathit{Sys}'_\mathrm{A}, 0), (\mathit{Sys}'_\mathrm{B}, 0)\}$. The event-mode combinations (e, σ) that can fire from this cut are

$$(\mathit{com}', \{\}),\ (\mathit{mim}'_\mathrm{A}, \{x \leftarrow 1\}),\ (\mathit{mim}'_\mathrm{A}, \{x \leftarrow 2\}),\ (\mathit{mim}'_\mathrm{B}, \{y \leftarrow 1\}),\ (\mathit{mim}'_\mathrm{B}, \{y \leftarrow 2\}).$$

Consider now the system condition $b = \mathit{Sys}'_\mathrm{A}$. We verify that at most one of the event-mode combinations (e, σ) from above satisfies $\mathit{Sys}'_\mathrm{A} \to e$ and $\zeta(b)_{\pi(e)}[\theta][\sigma] \equiv \mathit{true}$. We have $\mathit{Sys}'_\mathrm{A} \to \mathit{com}'$ and $\mathit{Sys}'_\mathrm{A} \to \mathit{mim}'_\mathrm{A}$.

- For $(e, \sigma) = (\mathit{com}', \{\})$ we have $\zeta(\mathit{Sys}'_\mathrm{A})_{\pi(e)}[\theta][\sigma] = \zeta(\mathit{Sys}'_\mathrm{A})_{\mathit{com}}[\theta][\sigma] \equiv \mathit{true}[\theta][\sigma] \equiv \mathit{true}$.
- For $(e, \sigma) = (\mathit{mim}'_\mathrm{A}, \{x \leftarrow 1\})$ we have $\zeta(\mathit{Sys}'_\mathrm{A})_{\pi(e)}[\theta][\sigma] = \zeta(\mathit{Sys}'_\mathrm{A})_{\mathit{mim}_\mathrm{A}}[\theta][\sigma] \equiv \mathit{false}[\theta][\sigma] \equiv \mathit{false}$.
- For $(e, \sigma) = (\mathit{mim}'_\mathrm{A}, \{x \leftarrow 2\})$ we have $\zeta(\mathit{Sys}'_\mathrm{A})_{\pi(e)}[\theta][\sigma] = \zeta(\mathit{Sys}'_\mathrm{A})_{\mathit{mim}_\mathrm{A}}[\theta][\sigma] \equiv \mathit{false}[\theta][\sigma] \equiv \mathit{false}$.

So only $(e, \sigma) = (\mathit{com}, \{\})$ satisfies $\mathit{Sys}'_\mathrm{A} \to e$ and $\zeta(\mathit{Sys}'_\mathrm{B})_{\pi(e)}[\theta][\sigma] \equiv \mathit{true}$. Analogously, $(e, \sigma) = (\mathit{com}, \{\})$ is also the only event-mode combination satisfying $\mathit{Sys}'_\mathrm{B} \to e$ and

$\zeta(Sys'_{\mathrm{B}})_{\pi(e)}[\theta][\sigma] \equiv \mathit{true}$. These arguments hold independently of θ. Thus, we showed that ζ satisfies the determinism condition from Def. 5.3 for the fixed $C = \{go_{\mathrm{B}}, go_{\mathrm{A}}\}$.

The second example is the configuration $C = \{go_{\mathrm{B}}, go_{\mathrm{A}}, com'\}$. Consider again the instantiation $\theta = \{y_{go_{\mathrm{B}}} \leftarrow 1, x_{go_{\mathrm{A}}} \leftarrow 2\} \in \Theta(C)$. The corresponding cut is $\mathrm{cut}(C, \theta) = \{(Go_{\mathrm{B}}, 1), (Go_{\mathrm{A}}, 2), (Sys''_{\mathrm{A}}, 0), (Sys''_{\mathrm{B}}, 0)\}$. The event-mode combinations (e, σ) that can fire from this cut are

$$(com'', \{\}),\ (mim''_{\mathrm{A}}, \{x \leftarrow 1\}),\ (mim''_{\mathrm{A}}, \{x \leftarrow 2\}),\ (mim''_{\mathrm{B}}, \{y \leftarrow 1\}),\ (mim''_{\mathrm{B}}, \{y \leftarrow 2\}).$$

Consider now the system condition $b = Sys''_{\mathrm{A}}$. We verify that at most one of the event-mode combinations (e, σ) from above satisfies $Sys''_{\mathrm{A}} \to e$ and $\zeta(Sys''_{\mathrm{A}})_{\pi(e)}[\theta][\sigma] \equiv \mathit{true}$. We have $Sys''_{\mathrm{A}} \to com''$ and $Sys'_{\mathrm{A}} \to mim''_{\mathrm{A}}$.

- For $(e, \sigma) = (com'', \{\})$ we have $\zeta(Sys''_{\mathrm{A}})_{\pi(e)}[\theta][\sigma] = \zeta(Sys'_{\mathrm{A}})_{com}[\theta][\sigma] \equiv \mathit{false}[\theta][\sigma]$ $\equiv \mathit{false}$.
- For $(e, \sigma) = (mim''_{\mathrm{A}}, \{x \leftarrow 1\})$ we have
$$\begin{aligned}\zeta(Sys''_{\mathrm{A}})_{\pi(e)}[\theta][\sigma] &= \zeta(Sys''_{\mathrm{A}})_{mim_{\mathrm{A}}}[\theta][\sigma]\\ &\equiv (x = x_{go_{\mathrm{A}}})[y_{go_{\mathrm{B}}} \leftarrow 1, x_{go_{\mathrm{A}}} \leftarrow 2][x \leftarrow 1]\\ &\equiv (1 = 2)\\ &\equiv \mathit{false}.\end{aligned}$$
- For $(e, \sigma) = (mim''_{\mathrm{A}}, \{x \leftarrow 2\})$ we have
$$\begin{aligned}\zeta(Sys''_{\mathrm{A}})_{\pi(e)}[\theta][\sigma] &= \zeta(Sys''_{\mathrm{A}})_{mim_{\mathrm{A}}}[\theta][\sigma]\\ &\equiv (x = x_{go_{\mathrm{A}}})[y_{go_{\mathrm{B}}} \leftarrow 1, x_{go_{\mathrm{A}}} \leftarrow 2][x \leftarrow 2]\\ &\equiv (2 = 2)\\ &\equiv \mathit{true}.\end{aligned}$$

So only $(e, \sigma) = (mim''_{\mathrm{A}}, \{x \leftarrow 2\})$ satisfies $Sys''_{\mathrm{A}} \to e$ and $\zeta(Sys''_{\mathrm{A}})_{\pi(e)}[\theta][\sigma] \equiv \mathit{true}$. Analogously, only $(e, \sigma) = (mim''_{\mathrm{B}}, \{y \leftarrow 1\})$ satisfies $Sys''_{\mathrm{B}} \to e$ and $\zeta(Sys''_{\mathrm{B}})_{\pi(e)}[\theta][\sigma] \equiv \mathit{true}$. For all other instantiations $\theta \in \Theta(C)$, the arguments are analogous. Thus, we showed that ζ satisfies the determinism condition from Def. 5.3 for the fixed $C = \{go_{\mathrm{B}}, go_{\mathrm{A}}, com'\}$. $\lhd$

5.2 A Glimpse into Synthesis of Symbolic Strategies

After providing a definition of symbolic strategies for high-level Petri games, naturally the question arises whether we can transfer the results of synthesizing strategies of P/T Petri games to the symbolic level. In this section, we initiate a preliminary discussion on the generalization of two established synthesis approaches. The first approach is the one presented in [FO17; Gie22], where the Petri game is reduced to a two-player game. The second approach, presented in [Fin15], is called *bounded synthesis* and searches for finite representations of possibly infinite strategy in a progressively increased prefix of the unfolding.

Synthesis via reduction to a two-player game. In Sec. 3.1.3, we recalled the approach from [FO17; Gie22] of reducing a (in our case, proper) Petri game to a two-player game on a finite graph. In Sec. 3.2 we reduced the size of this two-player game by considering equivalence classes of nodes with respect to symmetries in the net. The resulting two-player game was called the symbolic two-player game.

I now first briefly discuss the question whether the reduction w.r.t. symmetries is compatible with the notion of symbolic unfolding, i.e., whether strategies in the symbolic two-player game can be translated into symbolic strategies. I will elaborate on my perspective on why I think that, even if this was possible, a symmetry-based approach is suboptimal. While this concepts shares a similar foundational idea to symbolic unfoldings, they differ in its execution.

Following this discussion, I will briefly explain my viewpoint on why, in general, the approach for the two-player game presented in [FO17; Gie22] would have to undergo a major revision to generate symbolic strategies.

One reason why I think that a symmetry-based approach is suboptimal is that the reduction by symmetries, is often not as strong as the one in the symbolic unfolding. In some instances, there might be no reduction at all. This means one possibly has to consider multiple vertices in the symbolic two-player game that all correspond to the same cut in the symbolic unfolding. Every time the same condition is encoutered, the strategy function would have to be modified to include the setting of each corresponding vertex.

Another reason for incompatibility of the two approaches is that the (low-level) strategies generated by the approach with the symbolic two-player game are always "symmetric" in the sense that, in any two states that are symmetric to each other, the decisions of the system players in one state are symmetric to the decisions in the other state. Symbolic strategies as defined above, however, can encode different/asymmetric behavior in symmetric situations. This means losing a property from the low-level case described in App. 3.A: for a given P/T Petri game G, we can represent any strategy for the system players in G by a strategy for player 0 in the two-player game $\mathbb{G}(\mathsf{G})$. In case of a high-level Petri game G, however, "asymmetric" symbolic strategies for the system players cannot be represented by strategies for Player 0 in the symbolic two-player game $\overline{\mathbb{G}}(G)$.

The second reason can be extended into an argument convincing me that the approach presented in [FO17; Gie22] for the two-player game reduction of P/T Petri games cannot be generalized without significant modifications to high-level Petri games. In the proposed definition of symbolic strategies, the set of modes of a transition that a player on a system condition allows (encoded by the predicate) can explicitly depend on all values of variables around events in the condition's past.

In the low-level two-player game on the other hand, each vertex in the two-player game corresponds to an enriched marking in the P/T Petri game. One could think of constructing a two-player game where vertices represent enriched "symbolic markings". However, not in the sense of symmetries, as it involves variables replacing colors on places and in modes of transitions. Just like, in the (low-level) two-player game, we do not remember the whole causal history but only the occupied places, one would now not remember *all* used variables but only represent the values of variables on arcs to the occupied places.

Such an approach would naturally lead to equivalence classes. While resembling the concept of dynamic decision sets from Sec. 3.3, it differs by having a distinct variable on each place and being independent of symmetries. The symmetries in Sec. 3.3, however, ensured compatibility of equivalence classes with the firing relation. Without relying on symmetries, maintaining this compatibility necessitates storing information about relations between variables. The specifics of what must be stored depend on the logic used for expressing guards, but it would probably lead to an increase of the size of the two-player game that is exponential in the number of variables. Additionally, a way must be found to build the predicates constituting the symbolic strategy only on the variables the corresponding condition knows.

This brief exploration indicates a significant modification of the two-player game approach is needed even with preliminary considerations. However, it presents an intriguing avenue for future research.

Bounded Synthesis. Finally, I briefly consider the potential of bounded synthesis for the aforementioned symbolic strategies in high-level Petri games. For low-level Petri games with a safety objective, a bounded synthesis approach has been introduced in [Fin15]. This approach involves seeking winning strategies for system players within a prefix of the Petri game's unfolding, progressively increasing two bounds that restrict the width and depth of the prefix. It is noteworthy that, in these prefixes, the search for *infinite* strategies is conducted, leading to certain arcs being implicitly looped back into the prefix.

The result of the search then is a finite representation of a possibly infinite strategy. By considering prefixes of the symbolic unfolding, these methodologies may potentially extend to high-level Petri games. Techniques [CHJ+14; AHP+22], where a progressively growing complete prefix of the unfolding of a P/T Petri net is employed, may prove valuable in this context. However, since in the proposed definition, symbolic strategies are families of predicates over variables in the unfolding, verifying the existence of a strategy essentially involves *synthesizing functions* [KMPS10; FG19]. In general, this problem is undecidable [FW21]. Nevertheless, under certain constraints on the structure of the considered high-level Petri games and the logic of guards, there might be feasible scenarios.

Chapter 6

Conclusions

We introduced a symmetry-exploiting solving algorithm for a subclass of expansion safe high-level Petri games with a single recurrently interfering source of external information. The main part of the algorithm is a reduction of the high-level Petri game to a two-player game which vertices consist of enriched equivalence classes of the Petri game's behavior. The key idea of the reduction is borrowed from the reduction of a P/T Petri game with a single external source of information to a two-player game presented in [FO17; Gie22]. We proved the correctness of the new reduction by defining a bisimulation between the new game and the game obtained by converting the high-level Petri game to a P/T Petri game and applying the reduction of [FO17; Gie22]. Experimental results show that utilizing the symmetries of the system enabled us to reduce the state space needed for resolving the synthesis problem in the presented benchmark families by up to three orders of magnitude. Next, we presented the definition and computation of unique, canonical representations for the equivalence classes forming the vertices in the symbolic two-player game. Although there is an associated fixed cost for computing these representations, the worst-case scenario is an improvement compared to the absence of canonical representations. Experiments show that employing the canonical representations as vertices in the symbolic two-player game results in a performance gain for larger state space to number of symmetries ratio.

We introduced the notion of complete finite prefixes of symbolic unfoldings of high-level Petri nets. We identified a class of safe high-level Petri nets generalizing safe P/T Petri nets, for which we generalized a well-known algorithm by Esparza, Römer, Vogler to compute such a finite and complete prefix. This constitutes a consolidation and generalization of the concepts of [ERV02; Cha06; CF10; CJ04]. While the resulting symbolic prefix has the same depth as a finite and complete prefix of the unfolding of the represented P/T Petri net, it can be significantly smaller due to less branching. In the case of infinitely many reachable markings (where the original algorithm is not applicable) we identified the class of so-called *symbolically compact* nets for which an adapted version of the generalized algorithm can be applied. For that, we showed how to check an adapted cut-off criterion by symbolically describing sets of markings. An implementation of the generalized algorithm, which was tested against four benchmark families, confirmed that a high degree of mode-nondeterminism in a high-level Petri net favors the symbolic approach, surpassing the performance of the low-level approach.

Future Work. Several improvements regarding symmetries in high-level Petri nets exist in the literature. For example, the papers [HITZ95; BHI04; BC04; BC05; Cap05] introduce efficiency improvements for systems with a mixture of symmetric and asymmetric behaviors, or, in [TDM03] the symmetries of entirely symmetric models are deduced from the system itself, i.e., the color classes of a symmetric net can be partitioned automatically. It seems interesting to investigate to what extent distributed synthesis via high-level Petri games could profit from these results. Additionally, applying symmetries to decidability procedures for Petri games with other restrictions on the number of players or net structure [FG17; HO22] or other objectives for the system players [Hec21; FGHO22; Han23] could potentially yield significant state-space reductions in these cases.

Another intriguing goal is the construction of a symbolic reachability tree for symbolically compact nets, and a comparison with the complete finite prefix, as outlined in Sec. 4.4.4.

Moreover, it seems feasible to extend the definition and computation of complete finite prefixes of the symbolic unfolding to k-bounded high-level Petri nets. This extension can probably leverage the generalizations proposed in [ERV02] for k-bounded P/T Petri nets.

Ultimately, achieving the synthesis of symbolic strategies for high-level Petri games, emerges as a crucial component to finalize the comprehensive framework of high-level Petri games. An outlook on this topic was given in Chapter 5. In particular, the results for finite complete prefixes could be employed in a bounded synthesis approach, with techniques similar to [Fin15] or [CHJ+14; AHP+22], with a stepwise increasing bound on the size of the symbolic unfolding.

Bibliography

[ABP23] Federica Adobbati, Luca Bernardinello, and Lucia Pomello. “Solving a Safety Game on the Unfolding of Safe Petri Nets”. In: *Proceedings of the 2023 International Workshop on Petri Nets and Software Engineering (PNSE 2023) co-located with the 44th International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2023), June 27, 2023, Lisbon, Portugal.* Vol. 3430. CEUR Workshop Proceedings. CEUR-WS.org, 2023, pp. 53–69. URL: `https://ceur-ws.org/Vol-3430/paper4.pdf`.

[AHP+22] Giann Karlo Aguirre-Samboní, Stefan Haar, Loïc Paulevé, Stefan Schwoon, and Nick Würdemann. “Avoid One’s Doom: Finding Cliff-Edge Configurations in Petri Nets”. In: *Proceedings of the 13th International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2022, Madrid, Spain, September 21-23, 2022.* Vol. 370. EPTCS. 2022, pp. 178–193. URL: `https://doi.org/10.4204/EPTCS.370.12`.

[AIN00] Parosh Aziz Abdulla, S. Purushothaman Iyer, and Aletta Nylén. “Unfoldings of Unbounded Petri Nets”. In: *Proc. CAV 2000.* Vol. 1855. Lecture Notes in Computer Science. Springer, 2000, pp. 495–507. URL: `https://doi.org/10.1007/10722167%5C_37`.

[AP76] Michael E. Atwood and Peter G. Polson. “A process model for water jug problems”. In: *Cognitive Psychology* 8.2 (1976), pp. 191–216. ISSN: 0010-0285. URL: `https://doi.org/10.1016/0010-0285(76)90023-2`.

[BBB+22] Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. “cvc5: A Versatile and Industrial-Strength SMT Solver”. In: *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I.* Vol. 13243. Lecture Notes in Computer Science. Springer, 2022, pp. 415–442. URL: `https://doi.org/10.1007/978-3-030-99524-9%5C_24`.

[BBD15] Éric Badouel, Luca Bernardinello, and Philippe Darondeau. *Petri Net Synthesis*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2015. ISBN: 978-3-662-47966-7. URL: https://doi.org/10.1007/978-3-662-47967-4.

[BC04] Carlo Bellettini and Lorenzo Capra. "A Quotient Graph for Asymmetric Distributed Systems". In: *12th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2004), 4-8 October 2004, Vollendam, The Netherlands*. IEEE Computer Society, 2004, pp. 560–568. URL: https://doi.org/10.1109/MASCOT.2004.1348313.

[BC05] Carlo Bellettini and Lorenzo Capra. "Quotient Graphs for the Analysis of Asymmetric Distributed Systems: Surveying Two Alternative Approaches". In: *19th International Parallel and Distributed Processing Symposium (IPDPS 2005), CD-ROM / Abstracts Proceedings, 4-8 April 2005, Denver, CO, USA*. IEEE Computer Society, 2005. URL: https://doi.org/10.1109/IPDPS.2005.371.

[BCJ18] Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann. "Graph Games and Reactive Synthesis". In: *Handbook of Model Checking*. Ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. Springer, 2018, pp. 921–962. URL: https://doi.org/10.1007/978-3-319-10575-8%5C_27.

[BD15] Eike Best and Raymond R. Devillers. "Synthesis and reengineering of persistent systems". In: *Acta Informatica* 52.1 (2015), pp. 35–60. URL: https://doi.org/10.1007/s00236-014-0209-7.

[BDLV05] Ugo A. Buy, Houshang Darabi, Mihai Lehene, and Vikram Venepally. "Supervisory Control of Time Petri Nets Using Net Unfolding". In: *29th Annual International Computer Software and Applications Conference, COMPSAC 2005, Edinburgh, Scotland, UK, July 25-28, 2005. Volume 2*. Vol. 2. IEEE Computer Society, 2005, pp. 97–100. URL: https://doi.org/10.1109/COMPSAC.2005.148.

[BFF+95] Eike Best, Hans Fleischhack, Wojciech Fraczak, Richard P. Hopkins, Hanna Klaudel, and Elisabeth Pelz. "A Class of Composable High Level Petri Nets with an Application to the Semantics of B(PN)^2". In: *Application and Theory of Petri Nets 1995, 16th International Conference, Turin, Italy, June 26-30, 1995, Proceedings*. Vol. 935. Lecture Notes in Computer Science. Springer, 1995, pp. 103–120. URL: https://doi.org/10.1007/3-540-60029-9%5C_36.

[BFH19] Raven Beutner, Bernd Finkbeiner, and Jesko Hecking-Harbusch. "Translating Asynchronous Games for Distributed Synthesis". In: *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*. Vol. 140. LIPIcs. Schloss Dagstuhl - Leibniz-

Zentrum für Informatik, 2019, 26:1–26:16. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2019.26.

[BGJ+07] Roderick Bloem, Stefan J. Galler, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Martin Weiglhofer. "Interactive presentation: Automatic hardware synthesis from specifications: a case study". In: *2007 Design, Automation and Test in Europe Conference and Exposition, DATE 2007, Nice, France, April 16-20, 2007*. EDA Consortium, San Jose, CA, USA, 2007, pp. 1188–1193. URL: https://dl.acm.org/citation.cfm?id=1266622.

[BHI04] Soheib Baarir, Serge Haddad, and Jean-michel Ilié. "Exploiting partial symmetries in well-formed nets for the reachability and the linear time model checking problems". In: *IFAC Proceedings Volumes* 37.18 (2004). 7th International Workshop on Discrete Event Systems (WODES'04), Reims, France, September 22-24, 2004, pp. 219–224. ISSN: 1474–6670. URL: https://doi.org/10.1016/S1474-6670(17)30749-8.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. ISBN: 978-0-262-02649-9.

[BL69] J. Richard Büchi and Lawrence H. Landweber. "Solving sequential conditions by finite-state strategies". In: *Transactions of the American Mathematical Society* 138 (1969), pp. 295–311. URL: https://api.semanticscholar.org/CorpusID:4568478.

[Cap05] Lorenzo Capra. "Colored Petri Nets State-Space Reduction via Symbolic Execution". In: *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005), 25-29 September 2005, Timisoara, Romania*. IEEE Computer Society, 2005, pp. 231–238. URL: https://doi.org/10.1109/SYNASC.2005.26.

[CDFH91a] Giovanni Chiola, Claude Dutheillet, Giuliana Franceschinis, and Serge Haddad. "On Well-Formed Coloured Nets and Their Symbolic Reachability Graph". In: *High-level Petri Nets: Theory and Application*. Ed. by Kurt Jensen and Grzegorz Rozenberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 373–396. ISBN: 978-3-642-84524-6. URL: https://doi.org/10.1007/978-3-642-84524-6_13.

[CDFH91b] Giovanni Chiola, Claude Dutheillet, Giuliana Franceschinis, and Serge Haddad. "Stochastic Well-Formed Coloured Nets and Multiprocessor Modelling Applications". In: *High-level Petri Nets: Theory and Application*. Ed. by Kurt Jensen and Grzegorz Rozenberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 504–530. ISBN: 978-3-642-84524-6. URL: https://doi.org/10.1007/978-3-642-84524-6_19.

[CDFH93] Giovanni Chiola, Claude Dutheillet, Giuliana Franceschinis, and Serge Haddad. "Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications". In: *IEEE Trans. Computers* 42.11 (1993), pp. 1343–1360. URL: https://doi.org/10.1109/12.247838.

[CDFH97] Giovanni Chiola, Claude Dutheillet, Giuliana Franceschinis, and Serge Haddad. “A Symbolic Reachability Graph for Coloured Petri Nets”. In: *Theor. Comput. Sci.* 176.1-2 (1997), pp. 39–65. URL: https://doi.org/10.1016/S0304-3975(96)00010-2.

[CEJS98] Edmund M. Clarke, E. Allen Emerson, Somesh Jha, and A. Prasad Sistla. “Symmetry Reductions in Model Checking”. In: *Computer Aided Verification, 10th International Conference, CAV ’98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings*. Vol. 1427. Lecture Notes in Computer Science. Springer, 1998, pp. 147–158. URL: https://doi.org/10.1007/BFb0028741.

[CF10] Thomas Chatain and Eric Fabre. “Factorization Properties of Symbolic Unfoldings of Colored Petri Nets”. In: *Applications and Theory of Petri Nets, 31st International Conference, PETRI NETS 2010, Braga, Portugal, June 21-25, 2010. Proceedings*. Vol. 6128. Lecture Notes in Computer Science. Springer, 2010, pp. 165–184. URL: https://doi.org/10.1007/978-3-642-13675-7_11.

[CFJ93] Edmund M. Clarke, Thomas Filkorn, and Somesh Jha. “Exploiting Symmetry In Temporal Logic Model Checking”. In: *Computer Aided Verification, 5th International Conference, CAV ’93, Elounda, Greece, June 28 - July 1, 1993, Proceedings*. Vol. 697. Lecture Notes in Computer Science. Springer, 1993, pp. 450–462. URL: https://doi.org/10.1007/3-540-56922-7_37.

[Cha06] Thomas Chatain. “Symbolic Unfoldings of High-Level Petri Nets and Application to Supervision of Distributed Systems”. PhD thesis. Universit é de Rennes, 2006. URL: https://www.sudoc.fr/246936924.

[CHJ+14] Thomas Chatain, Stefan Haar, Loïg Jezequel, Loïc Paulevé, and Stefan Schwoon. “Characterization of Reachable Attractors Using Petri Net Unfoldings”. In: *Computational Methods in Systems Biology - 12th International Conference, CMSB 2014, Manchester, UK, November 17-19, 2014, Proceedings*. Vol. 8859. Lecture Notes in Computer Science. Springer, 2014, pp. 129–142. URL: https://doi.org/10.1007/978-3-319-12982-2_10.

[Chu57] A. Church. “Applications of recursive arithmetic to the problem of circuit synthesis”. In: *Summaries of the Summer Institute of Symbolic Logic*. Vol. 1. Cornell Univ., Ithaca, NY, 1957, pp. 3–50.

[Chu62] Alonzo Church. “Logic, arithmetic, and automata”. In: 1962. URL: https://api.semanticscholar.org/CorpusID:6464830.

[CJ04] Thomas Chatain and Claude Jard. “Symbolic Diagnosis of Partially Observable Concurrent Systems”. In: *Formal Techniques for Networked and Distributed Systems - FORTE 2004, 24th IFIP WG 6.1 International Conference, Madrid Spain, September 27-30, 2004, Proceedings*. Vol. 3235. Lecture Notes in Computer Science. Springer, 2004, pp. 326–342. URL: https://doi.org/10.1007/978-3-540-30232-2%5C_21.

[CJ06] Thomas Chatain and Claude Jard. “Complete Finite Prefixes of Symbolic Unfoldings of Safe Time Petri Nets”. In: *Petri Nets and Other Models of Concurrency - ICATPN 2006, 27th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, Turku, Finland, June 26-30, 2006, Proceedings*. Vol. 4024. Lecture Notes in Computer Science. Springer, 2006, pp. 125–145. URL: https://doi.org/10.1007/11767589_8.

[CJEF96] Edmund M. Clarke, Somesh Jha, Reinhard Enders, and Thomas Filkorn. “Exploiting Symmetry in Temporal Logic Model Checking”. In: *Formal Methods Syst. Des.* 9.1/2 (1996), pp. 77–104. URL: https://doi.org/10.1007/BF00625969.

[DH89] Claude Dutheillet and Serge Haddad. “Regular stochastic Petri nets”. In: *Advances in Petri Nets 1990 [10th International Conference on Applications and Theory of Petri Nets, Bonn, Germany, June 1989, Proceedings]*. Vol. 483. Lecture Notes in Computer Science. Springer, 1989, pp. 186–209. URL: https://doi.org/10.1007/3-540-53863-1_26.

[DHJ+04] Jörg Desel, Hans-Michael Hanisch, Gabriel Juhás, Robert Lorenz, and Christian Neumair. “A Guide to Modelling and Control with Modules of Signal Nets”. In: *Integration of Software Specification Techniques for Applications in Engineering, Priority Program SoftSpez of the German Research Foundation (DFG), Final Report*. Vol. 3147. Lecture Notes in Computer Science. Springer, 2004, pp. 270–300. URL: https://doi.org/10.1007/978-3-540-27863-4_16.

[DJN04] Jörg Desel, Gabriel Juhás, and Christian Neumair. “Finite Unfoldings of Unbounded Petri Nets”. In: *Applications and Theory of Petri Nets 2004, 25th International Conference, ICATPN 2004, Bologna, Italy, June 21-25, 2004, Proceedings*. Vol. 3099. Lecture Notes in Computer Science. Springer, 2004, pp. 157–176. URL: https://doi.org/10.1007/978-3-540-27793-4_10.

[DT22] Raymond R. Devillers and Ronny Tredup. “Synthesis of Inhibitor-Reset Petri Nets: Algorithmic and Complexity Issues”. In: *Application and Theory of Petri Nets and Concurrency - 43rd International Conference, PETRI NETS 2022, Bergen, Norway, June 19-24, 2022, Proceedings*. Vol. 13288. Lecture Notes in Computer Science. Springer, 2022, pp. 213–235. URL: https://doi.org/10.1007/978-3-031-06653-5%5C_12.

[EH08] Javier Esparza and Keijo Heljanko. *Unfoldings - A Partial-Order Approach to Model Checking*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2008. ISBN: 978-3-540-77425-9. URL: https://doi.org/10.1007/978-3-540-77426-6.

[EHGP08] Hartmut Ehrig, Kathrin Hoffmann, Karsten Gabriel, and Julia Padberg. "Composition and Independence of High-Level Net Processes". In: *Proceedings of the First Workshop on Formal Methods for Wireless Systems, FMWS@CONCUR 2008, Toronto, ON, Canada, August 23, 2008*. Vol. 242. Electronic Notes in Theoretical Computer Science 2. Elsevier, 2008, pp. 59–71. URL: https://doi.org/10.1016/j.entcs.2009.06.023.

[EHP+02] Hartmut Ehrig, Kathrin Hoffmann, Julia Padberg, Paolo Baldan, and Reiko Heckel. "High-Level Net Processes". In: *Formal and Natural Computing - Essays Dedicated to Grzegorz Rozenberg [on occasion of his 60th birthday, March 14, 2002]*. Vol. 2300. Lecture Notes in Computer Science. Springer, 2002, pp. 191–219. URL: https://doi.org/10.1007/3-540-45711-9_12.

[Eng91] Joost Engelfriet. "Branching Processes of Petri Nets". In: *Acta Informatica* 28.6 (1991), pp. 575–591. URL: https://doi.org/10.1007/BF01463946.

[ERV02] Javier Esparza, Stefan Römer, and Walter Vogler. "An Improvement of McMillan's Unfolding Algorithm". In: *Formal Methods Syst. Des.* 20.3 (2002), pp. 285–310. URL: https://doi.org/10.1023/A:1014746130920.

[FG17] Bernd Finkbeiner and Paul Gölz. "Synthesis in Distributed Environments". In: *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*. Vol. 93. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 28:1–28:14. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2017.28.

[FG19] Grigory Fedyukovich and Aarti Gupta. "Functional Synthesis with Examples". In: *Principles and Practice of Constraint Programming - 25th International Conference, CP 2019, Stamford, CT, USA, September 30 - October 4, 2019, Proceedings*. Vol. 11802. Lecture Notes in Computer Science. Springer, 2019, pp. 547–564. URL: https://doi.org/10.1007/978-3-030-30048-7_32.

[FGHO17] Bernd Finkbeiner, Manuel Gieseking, Jesko Hecking-Harbusch, and Ernst-Rüdiger Olderog. "Symbolic vs. Bounded Synthesis for Petri Games". In: *Proceedings Sixth Workshop on Synthesis, SYNT@CAV 2017, Heidelberg, Germany, 22nd July 2017*. Vol. 260. EPTCS. 2017, pp. 23–43. URL: https://doi.org/10.4204/EPTCS.260.5.

[FGHO22] Bernd Finkbeiner, Manuel Gieseking, Jesko Hecking-Harbusch, and Ernst-Rüdiger Olderog. "Global Winning Conditions in Synthesis of Distributed Systems with Causal Memory". In: *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*. Vol. 216. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 20:1–20:19. URL: https://doi.org/10.4230/LIPIcs.CSL.2022.20.

[FGO15] Bernd Finkbeiner, Manuel Gieseking, and Ernst-Rüdiger Olderog. "Adam: Causality-Based Synthesis of Distributed Systems". In: *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*. Vol. 9206. Lecture Notes in Computer Science. Springer, 2015, pp. 433–439. URL: https://doi.org/10.1007/978-3-319-21690-4_25.

[Fin15] Bernd Finkbeiner. "Bounded Synthesis for Petri Games". In: *Correct System Design - Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday, Oldenburg, Germany, September 8-9, 2015. Proceedings*. Vol. 9360. Lecture Notes in Computer Science. Springer, 2015, pp. 223–237. URL: https://doi.org/10.1007/978-3-319-23506-6_15.

[Fin16] Bernd Finkbeiner. "Synthesis of Reactive Systems". In: *Dependable Software Systems Engineering*. Ed. by Javier Esparza, Orna Grumberg, and Salomon Sickert. Vol. 45. NATO Science for Peace and Security Series - D: Information and Communication Security. IOS Press, 2016, pp. 72–98. URL: https://doi.org/10.3233/978-1-61499-627-9-72.

[FO14] Bernd Finkbeiner and Ernst-Rüdiger Olderog. "Petri Games: Synthesis of Distributed Systems with Causal Memory". In: *Proceedings Fifth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2014, Verona, Italy, September 10-12, 2014*. Vol. 161. EPTCS. 2014, pp. 217–230. URL: https://doi.org/10.4204/EPTCS.161.19.

[FO17] Bernd Finkbeiner and Ernst-Rüdiger Olderog. "Petri games: Synthesis of distributed systems with causal memory". In: *Inf. Comput.* 253 (2017), pp. 181–203. URL: https://doi.org/10.1016/j.ic.2016.07.006.

[FS05] Bernd Finkbeiner and Sven Schewe. "Uniform Distributed Synthesis". In: *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*. IEEE Computer Society, 2005, pp. 321–330. URL: https://doi.org/10.1109/LICS.2005.53.

[FW21] Emmanuel Filiot and Sarah Winter. "Synthesizing Computable Functions from Rational Specifications over Infinite Words". In: *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*. Vol. 213. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 43:1–43:16. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2021.43.

[GGMW13] Blaise Genest, Hugo Gimbert, Anca Muscholl, and Igor Walukiewicz. "Asynchronous Games over Tree Architectures". In: *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*. Vol. 7966. Lecture Notes in Computer Science. Springer, 2013, pp. 275–286. URL: https://doi.org/10.1007/978-3-642-39212-2_26.

[Gie20] Manuel Gieseking. AdamSYNT. https://github.com/adamtool/adamsynt. Accessed 28 November 2023. 2020.

[Gie22] Manuel Gieseking. “Correctness of Data Flows in Asynchronous Distributed Systems – Model Checking and Synthesis”. PhD thesis. Carl von Ossietzky Universität Oldenburg, Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften, Department für Informatik, 2022. URL: https://uol.de/f/2/dept/informatik/download/Promotionen/Gieseking_Diss.pdf.

[Gim22] Hugo Gimbert. “Distributed Asynchronous Games With Causal Memory are Undecidable”. In: *Log. Methods Comput. Sci.* 18.3 (2022). URL: https://doi.org/10.46298/lmcs-18(3:30)2022.

[Giu92] Alessandro Giua. “Petri Nets as Discrete Event Models for Supervisory Control”. PhD thesis. Rensselaer Polytechnic Institute, 1992. URL: https://www.alessandro-giua.it/UNICA/PAPERS/92phd.pdf.

[GL81] Hartmann J. Genrich and Kurt Lautenbach. “System Modelling with High-Level Petri Nets”. In: *Theor. Comput. Sci.* 13 (1981), pp. 109–136. URL: https://doi.org/10.1016/0304-3975(81)90113-4.

[GLZ04] Paul Gastin, Benjamin Lerman, and Marc Zeitoun. “Distributed Games with Causal Memory Are Decidable for Series-Parallel Systems”. In: *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science, 24th International Conference, Chennai, India, December 16-18, 2004, Proceedings*. Vol. 3328. Lecture Notes in Computer Science. Springer, 2004, pp. 275–286. URL: https://doi.org/10.1007/978-3-540-30538-5_23.

[GO21] Manuel Gieseking and Ernst-Rüdiger Olderog. “High-Level Representation of Benchmark Families for Petri Games”. In: *Model Checking, Synthesis, and Learning - Essays Dedicated to Bengt Jonsson on The Occasion of His 60th Birthday*. Vol. 13030. Lecture Notes in Computer Science. Springer, 2021, pp. 115–137. URL: https://doi.org/10.1007/978-3-030-91384-7_7.

[GOW20] Manuel Gieseking, Ernst-Rüdiger Olderog, and Nick Würdemann. “Solving high-level Petri games”. In: *Acta Informatica* 57.3-5 (2020), pp. 591–626. URL: https://doi.org/10.1007/s00236-020-00368-5.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, eds. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*. Vol. 2500. Lecture Notes in Computer Science. Springer, 2002. ISBN: 3-540-00388-6. URL: https://doi.org/10.1007/3-540-36387-4.

[GW21a] Manuel Gieseking and Nick Würdemann. “Canonical Representations for Direct Generation of Strategies in High-Level Petri Games”. In: *Application and Theory of Petri Nets and Concurrency - 42nd International Conference, PETRI NETS 2021, Virtual Event, June 23-25, 2021, Proceedings.* Vol. 12734. Lecture Notes in Computer Science. Springer, 2021, pp. 95–117. URL: https://doi.org/10.1007/978-3-030-76983-3_6.

[GW21b] Manuel Gieseking and Nick Würdemann. “Canonical Representations for Direct Generation of Strategies in High-level Petri Games (Full Version)”. In: *CoRR* abs/2103.10207 (2021). arXiv: 2103.10207. URL: https://arxiv.org/abs/2103.10207.

[GW21c] Manuel Gieseking and Nick Würdemann. *Artifact: Canonical Representations for Direct Generation of Strategies in High-level Petri Games.* Nov. 2021. URL: https://doi.org/10.6084/m9.figshare.13697845.

[Han23] Paul Hannibal. “(Un)Decidability Bounds of the Synthesis Problem for Petri Games”. In: *Proceedings of the Fourteenth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2023, Udine, Italy, 18-20th September 2023.* Vol. 390. EPTCS. 2023, pp. 115–131. URL: https://doi.org/10.4204/EPTCS.390.8.

[Hec21] Jesko Hecking-Harbusch. “Synthesis of asynchronous distributed systems from global specifications”. PhD thesis. Saarland University, Saarbrücken, Germany, 2021. URL: https://publikationen.sulb.uni-saarland.de/handle/20.500.11880/32108.

[HITZ95] Serge Haddad, Jean-Michel Ilié, Mohamed Taghelit, and Belhassen Zouari. “Symbolic Reachability Graph and Partial Symmetries”. In: *Application and Theory of Petri Nets 1995, 16th International Conference, Turin, Italy, June 26-30, 1995, Proceedings.* Vol. 935. Lecture Notes in Computer Science. Springer, 1995, pp. 238–257. URL: https://doi.org/10.1007/3-540-60029-9_43.

[HJJJ84] Peter Huber, Arne M. Jensen, Leif O. Jepsen, and Kurt Jensen. “Towards reachability trees for high-level Petri nets”. In: *Advances in Petri Nets 1984, European Workshop on Applications and Theory in Petri Nets, covers the last two years which include the workshop 1983 in Toulouse and the workshop 1984 in Aarhus, selected papers.* Vol. 188. Lecture Notes in Computer Science. Springer, 1984, pp. 215–233. URL: https://doi.org/10.1007/3-540-15204-0_13.

[HJJJ86] Peter Huber, Arne M. Jensen, Leif O. Jepsen, and Kurt Jensen. “Reachability Trees for High-level Petri Nets”. In: *Theor. Comput. Sci.* 45.3 (1986), pp. 261–292. URL: https://doi.org/10.1016/0304-3975(86)90046-0.

[HKPT06] Lom Hillah, Fabrice Kordon, Laure Petrucci-Dauchy, and Nicolas Trèves. “PN Standardisation: A Survey”. In: *Formal Techniques for Networked and Distributed Systems - FORTE 2006, 26th IFIP WG 6.1 International Conference, Paris, France, September 26-29, 2006*. Vol. 4229. Lecture Notes in Computer Science. Springer, 2006, pp. 307–322. URL: https://doi.org/10.1007/11888116_23.

[HO22] Paul Hannibal and Ernst-Rüdiger Olderog. “The Synthesis Problem for Repeatedly Communicating Petri Games”. In: *Application and Theory of Petri Nets and Concurrency - 43rd International Conference, PETRI NETS 2022, Bergen, Norway, June 19-24, 2022, Proceedings*. Vol. 13288. Lecture Notes in Computer Science. Springer, 2022, pp. 236–257. URL: https://doi.org/10.1007/978-3-031-06653-5%5C_13.

[HST07] Frédéric Herbreteau, Grégoire Sutre, and The Quang Tran. “Unfolding Concurrent Well-Structured Transition Systems”. In: *Tools and Algorithms for the Construction and Analysis of Systems, 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24 - April 1, 2007, Proceedings*. Vol. 4424. Lecture Notes in Computer Science. Springer, 2007, pp. 706–720. URL: https://doi.org/10.1007/978-3-540-71209-1_55.

[Jen96] Kurt Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use - Volume 1, Second Edition*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 1996. ISBN: 978-3-642-08243-6. URL: https://doi.org/10.1007/978-3-662-03241-1.

[JPRA77] Robin Jeffries, Peter G. Polson, Lydia Razran, and Michael E. Atwood. “A process model for Missionaries-Cannibals and other river-crossing problems”. In: *Cognitive Psychology* 9.4 (1977), pp. 412–440. ISSN: 0010-0285. URL: https://doi.org/10.1016/0010-0285(77)90015-9.

[JR91] Kurt Jensen and Grzegorz Rozenberg, eds. *High-level Petri Nets: Theory and Application*. Springer Berlin, Heidelberg, 1991. ISBN: 978-3540541257. URL: https://doi.org/10.1007/978-3-642-84524-6.

[KFP09] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. “Temporal-Logic-Based Reactive Mission and Motion Planning”. In: *IEEE Trans. Robotics* 25.6 (2009), pp. 1370–1381. URL: https://doi.org/10.1109/TRO.2009.2030225.

[Kho] Victor Khomenko. *Punf*. Accessed 28 November 2023. URL: homepages.cs.ncl.ac.uk/victor.khomenko/tools/punf/.

[KK03] Victor Khomenko and Maciej Koutny. “Branching Processes of High-Level Petri Nets”. In: *Tools and Algorithms for the Construction and Analysis of Systems, 9th International Conference, TACAS 2003, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS*

2003, Warsaw, Poland, April 7-11, 2003, Proceedings. Vol. 2619. Lecture Notes in Computer Science. Springer, 2003, pp. 458–472. URL: https://doi.org/10.1007/3-540-36577-X_34.

[KKV03] Victor Khomenko, Maciej Koutny, and Walter Vogler. “Canonical prefixes of Petri net unfoldings”. In: *Acta Informatica* 40.2 (2003), pp. 95–118. URL: https://doi.org/10.1007/s00236-003-0122-y.

[KL93] Kenji Koyama and Tony W. Lai. “An optimal Mastermind Strategy”. In: *J. Recreational Mathematics* 25.4 (1993), pp. 251–256.

[KLP06] Fabrice Kordon, Alban Linard, and Emmanuel Paviot-Adet. “Optimized Colored Nets Unfolding”. In: *Formal Techniques for Networked and Distributed Systems - FORTE 2006, 26th IFIP WG 6.1 International Conference, Paris, France, September 26-29, 2006*. Vol. 4229. Lecture Notes in Computer Science. Springer, 2006, pp. 339–355. URL: https://doi.org/10.1007/11888116_25.

[KMPS10] Viktor Kuncak, Mikaël Mayer, Ruzica Piskac, and Philippe Suter. “Comfusy: A Tool for Complete Functional Synthesis”. In: *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*. Vol. 6174. Lecture Notes in Computer Science. Springer, 2010, pp. 430–433. URL: https://doi.org/10.1007/978-3-642-14295-6_38.

[Knu77] Donald E. Knuth. “The Computer as Master Mind”. In: *J. Recreational Mathematics* 9.1 (1977), pp. 1–6.

[KV01] Orna Kupferman and Moshe Y. Vardi. “Synthesizing Distributed Systems”. In: *16th Annual IEEE Symposium on Logic in Computer Science, Boston, Massachusetts, USA, June 16-19, 2001, Proceedings*. IEEE Computer Society, 2001, pp. 389–398. URL: https://doi.org/10.1109/LICS.2001.932514.

[LHY12] Fei Liu, Monika Heiner, and Ming Yang. “An efficient method for unfolding colored Petri nets”. In: *Winter Simulation Conference, WSC ’12, Berlin, Germany, December 9-12, 2012*. WSC, 2012, 295:1–295:12. URL: https://doi.org/10.1109/WSC.2012.6465203.

[Lin91] Markus Lindquist. “Parameterized Reachability Trees for Predicate/Transition Nets”. In: *Advances in Petri Nets 1993, Papers from the 12th International Conference on Applications and Theory of Petri Nets, Gjern, Denmark, June 1991*. Vol. 674. Lecture Notes in Computer Science. Springer, 1991, pp. 301–324. URL: https://doi.org/10.1007/3-540-56689-9_49.

[Maz01] René Mazala. “Infinite Games”. In: *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*. Vol. 2500. Lecture Notes in Computer Science. Springer, 2001, pp. 23–42. URL: https://doi.org/10.1007/3-540-36387-4_2.

[McM95] Kenneth L. McMillan. “A Technique of State Space Search Based on Unfolding”. In: *Formal Methods Syst. Des.* 6.1 (1995), pp. 45–65. URL: https://doi.org/10.1007/BF01384314.

[MH08] Dirk Missal and Hans-Michael Hanisch. “A Modular Synthesis Approach for Distributed Safety Controllers, Part A: Modelling and Specification”. In: *IFAC Proceedings Volumes* 41.2 (2008). 17th IFAC World Congress, pp. 14473–14478. ISSN: 1474-6670. URL: https://doi.org/10.3182/20080706-5-KR-1001.02452.

[Mis12] Dirk Missal. “Formal synthesis of safety controller code for distributed controllers”. PhD thesis. Martin Luther University of Halle-Wittenberg, 2012. ISBN: 978-3-8325-3147-8. URL: http://d-nb.info/1021582786.

[MT01] P. Madhusudan and P. S. Thiagarajan. “Distributed Controller Synthesis for Local Specifications”. In: *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*. Vol. 2076. Lecture Notes in Computer Science. Springer, 2001, pp. 396–407. URL: https://doi.org/10.1007/3-540-48224-5_33.

[MTY05] P. Madhusudan, P. S. Thiagarajan, and Shaofa Yang. “The MSO Theory of Connectedly Communicating Processes”. In: *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*. Vol. 3821. Lecture Notes in Computer Science. Springer, 2005, pp. 201–212. URL: https://doi.org/10.1007/11590156_16.

[MW14] Anca Muscholl and Igor Walukiewicz. “Distributed Synthesis for Acyclic Architectures”. In: *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*. Vol. 29. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014, pp. 639–651. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2014.639.

[NPW81] Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. “Petri Nets, Event Structures and Domains, Part I”. In: *Theor. Comput. Sci.* 13 (1981), pp. 85–108. URL: https://doi.org/10.1016/0304-3975(81)90112-2.

[ÖW21] Okan Özkan and Nick Würdemann. “Resilience of Well-structured Graph Transformation Systems”. In: *Proceedings Twelfth International Workshop on Graph Computational Models, GCM@STAF 2021, Online, 22nd June 2021*. Vol. 350. EPTCS. 2021, pp. 69–88. URL: https://doi.org/10.4204/EPTCS.350.5.

[Pan] Lukas Panneke. *ColorUnfolder*. Accessed 28 November 2023. URL: https://github.com/Selebrator/ColorUnfolder.

[Pan23] Lukas Panneke. “Implementing Symbolic Unfoldings of High-level Petri Nets”. MA thesis. Carl von Ossietzky Universität Oldenburg, 2023.

[PR90] Amir Pnueli and Roni Rosner. “Distributed Reactive Systems are Hard to Synthesize”. In: *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*. IEEE Computer Society, 1990, pp. 746–757. URL: https://doi.org/10.1109/FSCS.1990.89597.

[Pre30] Mojżesz Presburger. “Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt”. In: *Proc. Comptes-rendus du I Congrés des Mathématiciens des Pays Slaves, Varsovie 1929*. 1930, pp. 92–101.

[PS89] Ian Pressman and David Singmaster. “The jealous husbands and The missionaries and cannibals”. In: *The Mathematical Gazette* 73.464 (1989), pp. 73–81. URL: https://doi.org/10.2307/3619658.

[Rei13] Wolfgang Reisig. *Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies*. Springer, 2013. ISBN: 978-3-642-33277-7. URL: https://doi.org/10.1007/978-3-642-33278-4.

[Rod] César Rodríguez. *Cunf*. Accessed 28 November 2023. URL: https://github.com/cesaro/cunf.

[Roz16] Kristin Yvonne Rozier. “Specification: The Biggest Bottleneck in Formal Methods and Autonomy”. In: *Verified Software. Theories, Tools, and Experiments - 8th International Conference, VSTTE 2016, Toronto, ON, Canada, July 17-18, 2016, Revised Selected Papers*. Vol. 9971. Lecture Notes in Computer Science. 2016, pp. 8–26. URL: https://doi.org/10.1007/978-3-319-48869-1_2.

[RS13] César Rodríguez and Stefan Schwoon. “Cunf: A Tool for Unfolding and Verifying Petri Nets with Read Arcs”. In: *Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*. Vol. 8172. Lecture Notes in Computer Science. Springer, 2013, pp. 492–495. URL: https://doi.org/10.1007/978-3-319-02444-8_42.

[RSB03] Jean-Francois Raskin, Mathias Samuelides, and Laurent Van Begin. *Petri Games are Monotonicbut Difficult to Decide*. Technical Report. Université Libre De Bruxelles, 2003. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=65e2d540a55c6dfa3d8f572c19af3f8b04e3bc8f.

[RW87] P. J. Ramadge and W. M. Wonham. “Supervisory Control of a Class of Discrete Event Processes”. In: *SIAM Journal on Control and Optimization* 25.1 (1987), pp. 206–230. eprint: https://doi.org/10.1137/0325013. URL: https://doi.org/10.1137/0325013.

[Sch] Stefan Schwoon. *Mole*. Accessed 28 November 2023. URL: http://www.lsv.ens-cachan.fr/~schwoon/tools/mole/.

[Sch00a] Karsten Schmidt. “How to Calculate Symmetries of Petri Nets”. In: *Acta Informatica* 36.7 (2000), pp. 545–590. URL: https://doi.org/10.1007/s002360050002.

[Sch00b] Karsten Schmidt. “Integrating Low Level Symmetries into Reachability Analysis”. In: *Tools and Algorithms for Construction and Analysis of Systems, 6th International Conference, TACAS 2000, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings*. Vol. 1785. Lecture Notes in Computer Science. Springer, 2000, pp. 315–330. URL: https://doi.org/10.1007/3-540-46419-0_22.

[Sch03] Pierre-Yves Schobbens. “Alternating-time logic with imperfect recall”. In: *1st International Workshop on Logic and Communication in Multi-Agent Systems, LCMAS 2003, Eindhoven, The Netherlands, June 29, 2003*. Vol. 85. Electronic Notes in Theoretical Computer Science 2. Elsevier, 2003, pp. 82–93. URL: https://doi.org/10.1016/S1571-0661(05)82604-0.

[Sch95] Karsten Schmidt. “Parameterized Reachability Trees for Algebraic Petri Nets”. In: *Application and Theory of Petri Nets 1995, 16th International Conference, Turin, Italy, June 26-30, 1995, Proceedings*. Vol. 935. Lecture Notes in Computer Science. Springer, 1995, pp. 392–411. URL: https://doi.org/10.1007/3-540-60029-9_51.

[Sta91] Peter H Starke. “Reachability Analysis of Petri Nets Using Symmetries”. In: *Syst. Anal. Model. Simul.* 8.4-5 (1991), pp. 293–303. ISSN: 0232-9298. URL: https://dl.acm.org/doi/10.5555/115220.115224.

[TDM03] Yann Thierry-Mieg, Claude Dutheillet, and Isabelle Mounier. “Automatic Symmetry Detection in Well-Formed Nets”. In: *Applications and Theory of Petri Nets 2003, 24th International Conference, ICATPN 2003, Eindhoven, The Netherlands, June 23-27, 2003, Proceedings*. Vol. 2679. Lecture Notes in Computer Science. Springer, 2003, pp. 82–101. URL: https://doi.org/10.1007/3-540-44919-1_9.

[TS07] Andrew S. Tanenbaum and Maarten van Steen. *Distributed systems - principles and paradigms, 2nd Edition*. Pearson Education, 2007. ISBN: 978-0-13-239227-3.

[WCH23] Nick Würdemann, Thomas Chatain, and Stefan Haar. “Taking Complete Finite Prefixes to High Level, Symbolically”. In: *Application and Theory of Petri Nets and Concurrency - 44th International Conference, PETRI NETS 2023, Lisbon, Portugal, June 25-30, 2023, Proceedings*. Vol. 13929. Lecture Notes in Computer Science. Full version available at https://inria.hal.science/hal-04029490. Springer, 2023, pp. 123–144. URL: https://doi.org/10.1007/978-3-031-33620-1_7.

[WCHP23] Nick Würdemann, Thomas Chatain, Stefan Haar, and Lukas Panneke. *Taking Complete Finite Prefixes To High Level, Symbolically.* Submitted to Fundamenta Informaticae. 2023. arXiv: 2311.11443 [cs.LO].

[Wol15] Karsten Wolf. “The Petri net twist in explicit model checking”. In: *Softw. Syst. Model.* 14.2 (2015), pp. 711–717. URL: https://doi.org/10.1007/s10270-014-0422-4.

[Wür21] Nick Würdemann. “Exploiting symmetries of high-level Petri games in distributed synthesis”. In: *it Inf. Technol.* 63.5-6 (2021), pp. 321–331. URL: https://doi.org/10.1515/itit-2021-0012.

[Zie87] Wieslaw Zielonka. “Notes on Finite Asynchronous Automata”. In: *RAIRO Theor. Informatics Appl.* 21.2 (1987), pp. 99–135. URL: https://doi.org/10.1051/ita/1987210200991.

[Zie95] Wieslaw Zielonka. “Asynchronous Automata”. In: *The Book of Traces*. Ed. by Volker Diekert and Grzegorz Rozenberg. World Scientific, 1995, pp. 205–247. URL: https://doi.org/10.1142/9789814261456_0007.

[ZWD95] Qiong Zhou, Michael Wang, and S. P. Dutta. “Generation of optimal control policy for flexible manufacturing cells: A Petri net approach”. In: *Intern. Journal of Advanced Manufacturing Technology* 10 (1995), pp. 59–65. URL: https://doi.org/10.1007/BF01184279.

Index

Symbol Overview

Petri Nets & Petri games		
P/T Petri net	$\mathsf{N} = (\mathscr{N}, \mathsf{M}_0) = (\mathsf{P}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0)$	12
P/T occurrence net	$\mathsf{O} = (\mathsf{B}, \mathsf{E}, \mathsf{H}, \mathsf{K}_0)$	13
Branching process	$\beta = (\mathsf{O}, \mathsf{h})$	14
Unfolding	$\Upsilon(\mathsf{N}) = (\mathsf{U}(\mathsf{N}), \pi^{\mathsf{N}})$	14
P/T Petri game	$\mathsf{G} = (\mathsf{P}_{\mathrm{S}}, \mathsf{P}_{\mathrm{E}}, \mathsf{T}, \mathsf{F}, \mathsf{M}_0, \mathrm{Obj}, \mathsf{P}_{\circledast}), \quad \circledast \in \{\spadesuit, \heartsuit\}$	15
Strategy	$\xi = (\mathsf{O}^{\xi}, \pi^{\xi})$	17
High-level Petri net	$N = (\mathcal{N}, \mathcal{M}_0) = (\mathit{Col}, \mathit{Var}, P, T, F, g, \mathcal{M}_0)$	23
Symmetric net	$N = (\mathscr{C}, P, T, J, F, g, M_0)$	84
Modes of transition	$\sigma \in \Sigma(t)$ (in symmetric nets)	23 (85)
High-level Petri game	$G = (\mathit{Col}, \mathit{Var}, P_{\mathrm{S}}, P_{\mathrm{E}}, T, F, g, \mathcal{M}_0, \mathrm{Cond}, P_{\circledast})$	23
Symbolic strategy	$\zeta, \quad \zeta(b) = \{\zeta_t \mid \pi(b) \to t\}$	192
High-level occurrence net	$O = (\mathit{Col}, \mathit{Var}, B, E, H, g, \mathcal{K}_0)$	28
Symbolic branching process	$\beta = (O, h)$	30
Symbolic unfolding	$\Upsilon(N) = (U(N), \pi^N)$	30
Petri net homomorphism	h high-level: h	13, 24
Symmetries	$s \in S(N), \quad S(G), \quad s : \mathit{Col} \to \mathit{Col}$	42
in symmetric nets	$s = (s_1, \dots, s_n) \in S_N,\ S_G$	89

Two-Player games		
Two-player game	$\mathbb{G}(\mathsf{G}) = (\mathbb{V}_0, \mathbb{V}_1, \mathbb{E}, v_0, \mathrm{Win}), \quad \mathbb{V} = \mathbb{V}_0 \sqcup \mathbb{V}_1$	52
Decision set	D	50
Symbolic two-player game	$\overline{\mathbb{G}}(\mathsf{G}) = (\overline{\mathbb{V}}_0, \overline{\mathbb{V}}_1, \overline{\mathbb{E}}, \overline{v}_0, \overline{\mathrm{Win}}), \quad \overline{\mathbb{V}} = \overline{\mathbb{V}}_0 \sqcup \overline{\mathbb{V}}_1$	68
Symbolic decision set	$D = [\mathsf{D}]$, representative $\overline{\mathsf{D}}$	60
Related objects	$s_{\mathsf{D}}, \quad S_{\mathsf{D}},\ \Sigma(t)_{\mathsf{D}},\ \alpha_{\mathsf{D}}$	60, 64
Canonical two-player game	$\widehat{\mathbb{G}}(\mathsf{G}) = (\widehat{\mathbb{V}}_0, \widehat{\mathbb{V}}_1, \widehat{\mathbb{E}}, \widehat{v}_0, \widehat{\mathrm{Win}}), \quad \widehat{\mathbb{V}} = \widehat{\mathbb{V}}_0 \sqcup \widehat{\mathbb{V}}_1$	124

Symmetric Petri games and canonical representations		
Basic color classes	$\mathscr{C} = \{\mathscr{C}_1, \dots, \mathscr{C}_n\}$, $\mathbf{u} \in \{0, \dots, n\}$	84
Static subclasses	$\mathscr{C}_i = \bigsqcup_{i=1}^{n_i} \mathscr{C}_{i,q}$	84
Color domain	$\mathscr{C}(r) = \mathscr{C}_1^{J(r,1)} \times \dots \times \mathscr{C}_n^{J(r,n)}$	84
Dynamic partition	$\mathcal{P} = (m, \mathrm{stat}, \mathrm{card})$	98
Dynamic subclasses	$\mathscr{Z} = \{\mathscr{Z}_i \mid i \in I\}$	98
	$\mathscr{Z}_i = \{\mathcal{Z}_i^j \mid 1 \leq j \leq m(i)\}$	
Symbolic color domain	$\mathscr{Z}(r) = \mathscr{Z}_1^{J(r,1)} \times \dots \times \mathscr{Z}_n^{J(r,n)}$	101
Valid assignment	$\eta = (\eta_1, \dots, \eta_n)$, $\eta^{-1}(\mathscr{D})$	100, 103
Dynamic representation	$\mathscr{R} = (\mathcal{P}, \mathscr{D})$, with dyn. decision set $\mathscr{D}$	103
Representation matrix	$\mathrm{mat}(\mathscr{R})$	113

Configurations	C, $[\mathsf{e}]$, symbolic: C, $[e]$	13, 33, 146
Instantiation of C	$\theta \in \Theta(C)$, $\theta : \mathit{Var}_{C \cup \{\bot\}} \to \mathit{Col}$	146
Cuts	$\mathrm{cut}(\mathsf{C})$, $\mathrm{cut}(C)$, $\mathrm{cut}(C, \theta)$, $\mathrm{Cuts}(C)$	13, 33, 146
Markings	$\mathrm{mark}(\mathsf{C})$, $\mathrm{mark}(C, \theta)$, $\mathrm{Marks}(C)$	141, 146
Future	$\Uparrow C$	147
Suffix & extension	Q, $C \oplus Q$, $C[\![M]\!]Q$, $O(C\|Q)$	149
Special homomorphisms	$I_1^2 : \Uparrow C_1 \to \Uparrow C_2$, $\varphi_{1,Q}^2 : \Uparrow O(C_1\|Q) \to \Uparrow C_2$	149, 150
Adequate order	$\prec$, special cases: $\prec_M$, $\prec_E$, $\prec_F$	150, 181

Predicates and Constraints		
Predicates	$\mathit{loc\text{-}pred}(e)$, $\mathit{pred}(e)$, $\mathit{pred}^{\odot}(b)$	29, 166
Indexed variables	$\mathbf{e}(b)$, $\mathbf{v}(b)$, $\mathbf{v}_{\mathbf{e}}(b)$, Var_e, $\mathit{Var}_{E'}$	29, 146
Constraint	$\kappa(B')$	166
Variable assignment	$\vartheta \in \Xi(B')$, $\vartheta : \pi(B) \to \mathit{Col}$	166

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichungen, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Ort, Datum   Unterschrift