On Bisimulation and Model-Checking for Concurrent Systems with Partial Order Semantics

Julian Gutierrez

Doctor of Philosophy
Laboratory for Foundations of Computer Science
School of Informatics
University of Edinburgh
2011

Abstract

In concurrency theory—the branch of (theoretical) computer science that studies the logical and mathematical foundations of parallel computation—there are two main formal ways of modelling the behaviour of systems where multiple actions or events can happen independently and at the same time: either with *interleaving* or with *partial order* semantics.

On the one hand, the interleaving semantics approach proposes to reduce concurrency to the nondeterministic, sequential computation of the events the system can perform independently. On the other hand, partial order semantics represent concurrency explicitly by means of an independence relation on the set of events that the system can execute in parallel; following this approach, the so-called 'true concurrency' approach, independence or concurrency is a *primitive* notion rather than a *derived* concept as in the interleaving framework.

Using interleaving or partial order semantics is, however, more than a matter of taste. In fact, choosing one kind of semantics over the other can have important implications—both from theoretical and practical viewpoints—as making such a choice can raise different issues, some of which we investigate here. More specifically, this thesis studies concurrent systems with partial order semantics and focuses on their *bisimulation* and *model-checking* problems; the theories and techniques herein apply, in a uniform way, to different classes of Petri nets, event structures, and transition system with independence (TSI) models.

Some results of this work are: a number of *mu-calculi* (in this case, fixpoint extensions of modal logic) that, in certain classes of systems, induce exactly the same identifications as some of the standard bisimulation equivalences used in concurrency. Secondly, the introduction of (infinite) *higher-order logic games* for bisimulation and for model-checking, where the players of the games are given (local) monadic second-order power on the sets of elements they are allowed to play. And, finally, the formalization of a new *order-theoretic concurrent game* model that provides a uniform approach to bisimulation and model-checking and bridges some mathematical concepts in order theory with the more operational world of games.

In particular, we show that in all cases the logic games for bisimulation and model-checking developed in this thesis are *sound* and *complete*, and therefore, also determined—even when considering models of *infinite* state systems; moreover, these logic games are *decidable* in the finite case and underpin novel decision procedures for systems verification.

Since the mu-calculi and (infinite) logic games studied here *generalise* well-known fixpoint modal logics as well as game-theoretic decision procedures for analysing concurrent systems with interleaving semantics, this thesis provides some of the groundwork for the design of a logic-based, game-theoretic framework for studying, in a *uniform* manner, several concurrent systems regardless of whether they have an interleaving or a partial order semantics.

Acknowledgements

Foremost, I thank my main supervisor, Julian Bradfield, for his guidance and support in the past three years and for having introduced me to both mathematical logic and true concurrency. I also thank my co-supervisor, Ian Stark, and Colin Stirling for their academic advice as well as for having encouraged me to explore my own ideas. I am indebted to Colin and Julian for providing some of the logical and mathematical tools for the development of this PhD thesis. Indeed, most of the results herein build upon their work on logic, games, and concurrency.

I have also been benefited, and in many cases drawn inspiration, from discussions and exchanges of ideas with various people inside and outside LFCS, in particular, with Sibylle Fröschle, Martin Lange, Robin Milner, Luke Ong, and Glynn Winskel; I thank Luke and Glynn also for their hospitality during my visits to, respectively, Oxford and Cambridge, where we discussed parts of this thesis. The final version of this monograph contains a number of improvements suggested by my two PhD examiners, Richard Mayr and Mogens Nielsen.

I also thank the very long list of people here in Scotland and abroad who, at any point, helped me polish and finish this manuscript. I am especially grateful to those who commented on preliminary versions of the papers and chapters associated with this thesis or provided me with useful feedback on different aspects of this work. I extend my gratitude to the anonymous referees who reviewed the papers in FOSSACS, CONCUR, WoLLIC, and the Information and Computation journal, where preliminary versions of parts of this thesis have been presented.

Special thanks also go to those who, a few years ago, supported my PhD application to the University of Edinburgh: Andres Jaramillo, Camilo Rueda, and Eugenio Tamura. Many thanks to the members of the Avispa research group as well, especially to Alejandro Arbelaez, Hugo Lopez, Jorge Perez, and Frank Valencia who have openly shown a positive and enthusiastic attitude towards my work, even long before I came to study to the United Kingdom.

Doing this thesis was a much more enjoyable experience thanks to some fellow doctoral students of LFCS. In particular, I would like to mention Lorenzo Clemente, Willem Heijltjes, Ben Kavanagh, Gavin Keighren, Anthony W. Lin, Matteo Mio, and Grant Passmore.

I was fortunate to have had financial support by an Overseas Research Studentship (ORS) Award as well as a School of Informatics PhD Scholarship of the University of Edinburgh; while finishing, I have been a part-time Research Assistant on the EPSRC Research Grant EP/G012962/1 ‘Solving Parity Games and Mu-Calculi’.

Finally, I want to express my deepest gratitude to the three most important women in my life: Cecilia, Paula, and Teresa, without whose love and moral support it would not have been possible for me to produce this thesis. I thank Cecilia particularly for encouraging me, day by day, to do my very best. Lastly, above everything and everyone, I especially thank you, Teresa, for you have always covered “my blind side”.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Julian Gutierrez*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation and Context

In the theory of sequential computation various mathematical models have been proposed in the last 70 years, e.g., Turing machines, the λ-calculus, etc. A hallmark of this theory is that all these formal models of sequential computation are equivalent in the sense that the observable behaviour given by the input-output functions they compute is exactly the same. For concurrent computation, however, an agreement on how to formally represent the observable behaviour of a concurrent or distributed system does not exist yet, and consequently, *the* model of concurrency is still to be developed—should a unifying theory of concurrency be desired.

Nevertheless, some partial uniformity can be found across different theories of concurrency as most concurrent systems have either an 'interleaving' or a 'partial order' semantics. This difference, in turn, separates off the set of models for concurrency in two families. Examples of *interleaving* models include Kripke structures, labelled transitions systems, infinite trees, Hoare languages, Moore and Mealy machines, and many more. On the other hand, examples of *partial order* models include Petri nets, event structures, asynchronous transition systems, Mazurkiewicz trace languages, Chu spaces, transition systems with independence (TSI), amongst others. A good introduction to various models for concurrency can be found in [72].

The difference between interleaving and partial order semantics is conceptually simple. On the one hand, the interleaving approach proposes to represent concurrency *implicitly* by reducing it to the nondeterministic, sequential computation of the events that the system can perform independently. On the other hand, partial order semantics represent concurrency *explicitly* by means of an 'independence relation' on the set of events that the system can execute in parallel; following this approach, the so-called 'true concurrency' approach, independence or concurrency is a *primitive* notion rather than a *derived* concept as in the interleaving framework.

In this sense, partial order models provide a semantically more faithful representation of concurrency and, therefore, allows one to study concurrency at a more fundamental semantic

level. Partial order semantics are, however, far more complex than interleaving ones. In fact, any nondeterministic sequential process or concurrent system with an interleaving semantics can be seen, trivially, as a partial order model with an *empty* independence relation. Then, one may expect to pay a price for having such a greater amount of information, and indeed, several decision problems that are tractable (to some degree) in the interleaving world can become rather difficult, or even algorithmically unsolvable, in a partial order setting.

In contrast, having the additional information about independence that comes with any partial order model of concurrency has also its advantages. Just to mention a few examples: with regards to verification, partial order models can be less sensitive to the 'state space explosion' problem [95] when analysing concurrent and distributed systems, and are the natural input of 'partial order reduction' methods [36] and 'unfolding' techniques [25] for performing both software and hardware verification. In fact, in practice, these features of partial order models can help build computationally better verification tools based on these techniques.

On the other hand, although all usual 'temporal properties' [58] can be verified over interleaving models of concurrency, more complex temporal properties involving *parallelism* and *causality*—which are natural to concurrent and distributed systems—can no longer be verified in the interleaving interpretations, cf., [6, 56, 78]. Besides this, regarding logic and semantics issues, it has been shown that partial order models, unlike interleaving ones, can be used to give very natural 'denotational semantics' to logics with concurrent features, e.g., [2, 44]; and quite interestingly, even to define 'games semantics' that generalize traditional, sequential games models of sequential processes, e.g., as presented in [64].

It is therefore fair to say that the study of theories and techniques for partial order models of concurrency continues to deserve much attention, since they can help alleviate some of the limitations related to the use of interleaving semantics in some particular contexts. By doing so, we expect to acquire a better understanding of the *semantic foundations* of concurrent computations. Roughly, this is the main motivation of this work. In particular, this thesis studies concurrent systems with partial order semantics and focuses on their 'bisimulation' and 'model-checking' problems. The theories and techniques developed in this thesis apply, in a *uniform* way, to different classes of Petri nets, event structures, and TSI models. Let us now give a brief informal introduction to these two decision problems.

The study of bisimulation and model-checking problems started in the late 1970's and early 1980's, respectively, from independent works in mathematical logic, concurrency theory, and program verification. In the bisimulation case, it started with the work of Milner [65] and Park [74] in concurrency, and independently of van Benthem [8] in mathematical logic. Whereas Milner and Park wanted to study behavioural equivalences between different concurrent systems, van Benthem was interested in finding a relationship between modal logic and first-order (FO) logic; in fact, he showed that modal logic is the 'bisimulation invariant' fragment of FO

logic. On the other hand, the study of model-checking problems started with the work of Clarke and Emerson [16] in concurrency and verification as an alternative—fully automatic—way of proving the correctness of programs. Nowadays, both problems are major research topics in theoretical and applied informatics as well as in mathematical logic.

Bisimulation and model-checking problems are, technically, quite different. On the one hand, a *bisimulation* problem takes two (concurrent) systems and a bisimulation equivalence relation as input—say, two systems $\mathbb{F}$ and $\mathbb{G}$ and an equivalence relation $\sim_b$—and asks the question 'is the model of $\mathbb{F}$ equivalent to the model of $\mathbb{G}$ with respect to $\sim_b$?', meaning, 'are $\mathbb{F}$ and $\mathbb{G}$ behaviourally or observationally equivalent?'; the answer of this question is not unique, even if $\mathbb{F}$ and $\mathbb{G}$ are fixed, since it depends on the particular equivalence relation $\sim_b$ under consideration. On the other hand, the *model-checking* problem takes a system and a logical formula as input—say, a system $\mathbb{H}$ and a formula ϕ—and asks the question 'does the model of $\mathbb{H}$ satisfy the formula ϕ?', meaning, 'does $\mathbb{H}$ have the property specified by ϕ?'.

Although the bisimulation and model-checking problems have been both studied for about thirty years (cf., [21, 84]), many issues are still unsolved, or have unsatisfactory solutions, especially when considering partial order models of concurrency such as those studied in this thesis. For this reason, here we investigate a number of core issues related to the use of 'mu-calculi' (*fixpoint* extensions of modal logic [14], in this case) and infinite 'logic games' [9] for bisimulation and model-checking. Our results show that *generalizations* to a partial order setting of some of the theories and techniques for interleaving concurrency can be used to address, *uniformly*, the analysis of concurrent systems with either kind of semantics.

The approach we follow uses fixpoint modal logics (for the mathematical specification of system properties) and infinite logic games (as a formal verification methodology) in an order-theoretic setting. The combination of fixpoint operators and infinite games is what gives our approach the power to deal with systems whose behaviour is deemed to be non-terminating, as infinitely long interactions with their environments are expected. Thus, fixpoint logics and infinite games are our main objects of study apart from the partial order models themselves. These choices we have made are not arbitrary as explained in the following paragraphs.

Although fixpoint logics are usually considered hard to understand, it is also well-known that they allow for a mathematically elegant study of finite and infinite behaviours (by the use of *least* and *greatest* fixpoint operators, respectively) and possess a high expressive power. For instance, most temporal logics for verification—such as LTL, CTL, or CTL*—can be translated to the 'modal mu-calculus' ($\mathcal{L}_\mu$ [54]), a rather small and simple fixpoint modal logic. Moreover, modal logics naturally characterise bisimulation equivalences, the *de facto* equivalence relations for concurrent systems, and one of the two main topics of study in this thesis.

Games, on the other hand, also form a great deal of the technical work herein. Infinite two-player games have proved to be useful in different applications [97], such as the synthesis of

reactive controllers, the evaluation of logical formulae, the verification of temporal logic specifications, and the formal description of systems designed to exhibit non-terminating behaviour, e.g., an operating system. These games have a long history in mathematical logic, but only in the last two decades have been given enough *algorithmic* content for efficiently solving several problems in computer science, both from practical and theoretical standpoints.

In this thesis we consider infinite games played by two players, a "Verifier" Eve ($\exists$) and a "Falsifier" Adam ($\forall$), whose interaction provides a formal framework for analysing concurrent systems with possibly non-terminating behaviour. In these games the flow of information between the two players is used to *verify* the truth or falsity of a given property; these kinds of logic games are called 'verification games'. In a verification game the main computational problem is the effective and efficient construction of a 'winning strategy', which is a mathematical representation of the solution of the particular decision problem under consideration, e.g., bisimulation or model-checking in our context. In the games approach, solving a verification problem—either a bisimulation or a model-checking one—reduces to answering the *same* question: 'does Eve have a strategy to win *all* plays in the verification game?'; i.e., checking that a strategy for Eve is indeed a winning strategy (provided that such a strategy exists).

Verification games are also interesting because they are closely related to (and can serve as a bridge between) other verification methods, such as the better-known 'automata' and 'tableau' techniques [38, 55]. Indeed, research in the last decade has started to show that both games and automata techniques, in combination with temporal and fixpoint logics, can be used as a basis of software and hardware verification methods for industrial-scale applications.

However, despite their attractive mathematical properties, both fixpoint logics and infinite games are hard to manipulate in a partial order setting since some bisimulation and model-checking problems are already computationally difficult, or even algorithmically unsolvable, for various families of partial order models. For instance, some bisimulation and model-checking problems can be *undecidable* in relatively simple classes of Petri nets, event structures, and TSI models, even when restricted to finite concurrent systems, e.g., [52, 56]. Another important issue is the 'determinacy' of infinite games, which is a property that guarantees the existence of winning strategies. While most bisimulation and model-checking games are determined on interleaving structures, they can easily be *undetermined* (and hence problematic from an algorithmic viewpoint) when considering partial order models of concurrency.

Therefore, there is a real challenge in developing mathematical theories for the analysis of concurrent systems with partial order semantics. This thesis makes various contributions in that direction, both for bisimulation and for model-checking (especially when traditional logics and game-based verification techniques fall into undecidability or undeterminacy problems). This is achieved by showing that the games developed in this thesis are *sound* and *complete*, and therefore, determined—even when considering *infinite* state systems; moreover, they are

decidable in the finite case and underpin new decision procedures for systems verification.

Since the mu-calculi and logic games studied here *generalise* well-known fixpoint modal logics and game-based decision procedures for concurrent systems with interleaving semantics, this PhD thesis gives some of the mathematical groundwork for the design of a logic-based, game-theoretic framework for studying, in a *uniform* manner, several concurrent systems both with interleaving and with partial order semantics.

1.2 Synopsis and Contributions

In summary, the main contributions of this PhD thesis are:

1. The development of various *mu-calculi with partial order semantics* which give a novel logical characterisation of concurrent behaviour whereas, in some classes of systems, induce the same identifications as some of the standard bisimilarities in concurrency.
2. The introduction of new classes of *infinite higher-order logic games* for bisimulation as well as for model-checking, where the two players of the games are given local monadic second-order (LMSO) power on the sets of elements they are allowed to play.
3. The definition of an *order-theoretic concurrent logic game model* which, on the one hand, reduces reasoning on partially ordered structures and provides a uniform approach to bisimulation and model-checking and, on the other hand, builds a bridge between some mathematical concepts in order theory and the more operational world of games.

The structure of the rest of this document and a brief description of some of the particular results of this work, which expands the summary just given, is outlined next:

Chapter 2. This chapter gives an introduction to the theoretical background that is used throughout the thesis. We discuss specific topics related to models of *concurrency*, fixpoint modal *logics*, and infinite logic *games* for bisimulation and model-checking. The chapter starts with a formal description of the three partial order models of concurrency we consider here: Petri nets, event structures, and TSI models. Then, a uniform presentation of the three models is given by using the TSI model. Next, we discuss the Hennessy–Milner logic (HML), a simple logical language for interleaving concurrency, as well as its extension with fixpoint operators—which is known as the modal mu-calculus ($\mathcal{L}_\mu$). After that, we give an introduction to different logic games for bisimulation and model-checking. Firstly, we present some games for bisimulation and, through them, the three bisimulation equivalences we consider here: strong bisimilarity (sb), history-preserving bisimilarity (hpb), and hereditary history-preserving bisimilarity (hhpb). Then, we discuss the (local) model-checking game for $\mathcal{L}_\mu$ as defined by Stirling on interleaving systems. The chapter finishes with some remarks on the decidability and determinacy of both of these logic games for verification.

Chapter 3. This chapter starts by studying a mathematical axiomatization for true concurrency and discusses how the concurrent, partial order behaviour of Petri nets, event structures, and TSI models can be captured in a uniform way by two simple and general *dualities* of local behaviour. Then, we study a logical characterization of those dualities of concurrent behaviour and show that natural *fixpoint modal logics* can be extracted from it. Also, with the aid of a few examples, some applications to concurrency of these new fixpoint logics are presented.

Then, the chapter is devoted to the study of the *logical equivalences* induced by such modal logics. As a result, it is shown that—when restricted to certain classes of systems—some of these logics induce the *same* identifications as some of the standard *bisimulation equivalences* used in interleaving and in true concurrency. In this part of the chapter we pay particular attention to the interplay between concurrency, causality, and conflict/choice. We do so by restricting these three kinds of behaviour *syntactically* (through the logics) and *semantically* (by considering special classes of systems). In particular, this study reveals that the *global* notion of causality induced by hpb can be captured by a simpler *local* notion given by the logical equivalence induced by a causal mu-calculus (which is defined in this chapter along with other mu-calculi). This result holds when restricted to a class of systems, named Ξ systems, for which a phenomenon called confusion (roughly, a situation where both concurrency and conflict happen simultaneously on a set of events) appears only in a deterministic form.

Chapter 4. This chapter introduces a new form of infinite logic games where the players are given (local) monadic second-order power on the sets of elements they are allowed to play. These *higher-order logic games* are defined, independently, for bisimulation and for model-checking. In both cases it is shown that when restricted to interleaving models of concurrency, such games provide, essentially, first-order power. In fact, it is shown that in an interleaving context they are equivalent to well-known logic games for interleaving concurrency, namely to the game for Milner and Park's strong bisimilarity (in the bisimulation case) and to the game for local $\mathcal{L}_\mu$ model-checking of Stirling (in the model-checking case). Moreover, we show that these higher-order games can be solved using so-called *memoryless* or history-free winning strategies (strategies that tell a player how to make a move—i.e., how to play—depending only on his/her current position in the game board), despite their higher-order power.

Moreover, with respect to the bisimulation case, we also show that, for the class of Ξ systems, the logical equivalences induced by the most expressive modal logics introduced in Chapter 3 are *decidable* as well as strictly *stronger* than hpb and strictly *weaker* than hhpb. Then, based on the study of the expressivity of the modal logics considered in this thesis, a hierarchy of logics and games for concurrency is defined. Next, we study the model-checking case and show that the higher-order games for model-checking defined in this chapter underpin a new game-based decision procedure for temporal verification. The game is then used to

model-check a class of regular, infinite event structures which has an undecidable monadic second-order (MSO) theory. As a result, we improve previous work in the literature in terms of temporal expressive power. On the logical side, this is obtained by allowing a free interplay of *fixpoint operators* and *local monadic second-order power* on the sets of elements that can be described within the logics studied in the previous chapter.

Chapter 5. This chapter studies an *order-theoretic concurrent game model* and some of its mathematical properties and algorithmic applications. The model provides a game-theoretic approach to system and property verification which applies *uniformly* to different decision problems (which include bisimulation and $\mathcal{L}_\mu$ model-checking) and models of concurrency, in particular to various classes of Petri nets, event structures, and TSI models.

This new games framework uses partially ordered sets (*posets*) to give a uniform mathematical representation of concurrent systems, logical specifications, and problem descriptions. Moreover, the games model comes with generic *metatheorems* for soundness, completeness, and determinacy, and reduces reasoning on partially ordered structures by focusing on simple *local* correctness conditions—which make considerably easier the formulation or design of concrete game-based decision procedures for bisimulation and for model-checking.

In this new logic game the players are allowed to make *asynchronous* and *independent* local moves in the board where they play. Moreover, each player follows a set of (locally defined) strategies, which are globally scheduled at the beginning of the game. Such strategies are given in the form of *closure operators* on a possibly infinite poset; other elements of the game are also formalised in order-theoretic terms, e.g., as order ideals or order filters of a poset. As a result, this new model builds a bridge between various mathematical concepts in *order theory* and other (still mathematical but more operationally intuitive) notions in *game theory*.

Chapter 6. This chapter finishes the thesis by presenting some *concluding remarks* as well as most relevant *related work*. Also, some of the present author's personal views on different topics related to concurrency, logic, and games, are put forward for discussion. A number of directions and ideas for *further work* are also given at the end of the chapter.

Main publications

Preliminary versions of parts of this thesis have been already published by Springer as part of the proceedings of FOSSACS [39], CONCUR [42], and WoLLIC [41]. An extended version of [42] appears in a special issue of the Information and Computation journal [43].

About this Ph.D. Thesis

This Ph.D. dissertation builds all of its theory upon the following thesis: that the logical and mathematical study of concurrent systems with *partial order semantics* provides a very *natural* formal framework where different concurrency models and decision problems (which include bisimulation and model-checking) can be studied both at a more *fundamental* semantic level as well as in a *uniform* manner. This is, mainly, because the information about *independent* and *local* partial order behaviour, which is inherent to concurrent and distributed systems, is mathematically *explicit* in concurrency models with such a kind of semantics.

Chapter 2

Concurrency, Logic, and Games

In this chapter we study the models for concurrency of interest in this dissertation, together with background material on the modal logics and games for verification that are relevant to the work presented in subsequent chapters. We also discuss some relationships between the models for concurrency that are studied throughout this document as well as between the bisimulation equivalences induced by the modal logics presented in this chapter and the equivalences for concurrency considered in this and forthcoming chapters.

2.1 Models for Concurrency

In concurrency theory there are two main semantic approaches to modelling concurrent or parallel behaviour, either using interleaving or using partial order models for concurrency. On the one hand, *interleaving* models represent concurrency as the nondeterministic combination of all possible sequential behaviours in the system. On the other hand, *partial order* models represent concurrency explicitly by means of an independence relation on the set of actions, transitions, or events in the system that can be executed concurrently.

We are interested in partial order models of concurrency for various reasons. In particular, because they can be seen as a generalisation of interleaving models as explained later on in this section. This feature allows us to define the (fixpoint modal) logics and (infinite logic) games developed in further chapters in a uniform way for several different models for concurrency, regardless of whether they are used to provide interleaving or partial order semantics.

We now present the models of concurrency that are considered in this PhD thesis, namely Petri nets, transition systems with independence (TSI), and event structures. We also present some basic relationships between them and how they generalise some models for interleaving concurrency. For further information on this topic the reader is referred to [72, 85].

2.1.1 Petri Nets

A 'net' $\mathcal{N}$ is a 5-tuple (P,C,R,θ,Σ), where P is a set of 'places', C is a set of 'actions', $R \subseteq (P \times C) \cup (C \times P)$ is a relation between places and actions, and θ is a labelling function $\theta : C \to \Sigma$ from actions to a set Σ of action labels. Places and actions are called 'nodes'; given a node $n \in P \cup C$, the set ${}^\bullet n = \{x \mid (x,n) \in R\}$ is the 'preset' of n and the set $n^\bullet = \{y \mid (n,y) \in R\}$ is its 'postset'. These elements define the static structure of a net.[1] The notion of computation state in a net (its dynamic part) is that of a 'marking', which is a set or a multiset of places; in the former case such nets are called 'safe'. Hereafter we only consider safe nets; Figure 2.1 shows a safe net together with a particular marking.

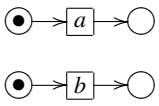


Figure 2.1: Places are depicted as circles and actions as boxes, with the corresponding label inside them. The black dots inside places are called 'tokens', and represent the current marking.

Definition 2.1. A *Petri net* $\mathfrak{N}$ is a tuple $(\mathcal{N}, M_0)$, where $\mathcal{N} = (P,C,R,\theta,\Sigma)$ is a net and $M_0 \subseteq P$ is its initial marking. ◁

As mentioned before, markings define the dynamics of nets. We say that a marking M enables an action t iff ${}^\bullet t \subseteq M$. If t is enabled at M, then t can occur and its occurrence leads to a successor marking M', where $M' = (M \setminus {}^\bullet t) \cup t^\bullet$, written as $M \xrightarrow{t} M'$. Let $\xrightarrow{t}$ be the relation between successor markings and $\longrightarrow^*$ its transitive closure. Given a Petri net $\mathfrak{N} = (\mathcal{N}, M_0)$, the relation $\longrightarrow^*$ defines the set of reachable markings in the system $\mathfrak{N}$; such a set of reachable markings is fixed for any M_0, and can be constructed with the occurrence net unfolding construction defined by Nielsen, Plotkin, and Winskel [71]. Finally, let par be the symmetric independence relation on actions such that t_1 par t_2 iff ${}^\bullet t_1^\bullet \cap {}^\bullet t_2^\bullet = \emptyset$, where ${}^\bullet t^\bullet$ stands for the set ${}^\bullet t \cup t^\bullet$, and there exists a reachable marking M such that both ${}^\bullet t_1 \subseteq M$ and ${}^\bullet t_2 \subseteq M$. Then, if two actions t_1 and t_2 can occur concurrently they must be independent, i.e., $(t_1,t_2) \in$ par.

2.1.2 Transition Systems with Independence

A 'labelled transition system' (LTS) is an edge-labelled graph structure. Formally, an LTS is a tuple (S,T,Σ), where S is a set of vertices called 'states', Σ is a set of labels, and $T \subseteq S \times \Sigma \times S$ is a set of Σ-labelled edges, which are called 'transitions'. A 'rooted LTS' is an LTS with a designated initial state $s_0 \in S$. A 'transition system with independence' (TSI) is a rooted LTS where independent transitions can be explicitly recognised. Formally:

[1] The reader acquainted with net theory may have noticed that we use the word 'action' instead of 'transition', more common in the Petri net literature. We made this choice of notation to avoid confusion later in the document.

Definition 2.2. A *transition system with independence* (TSI) $\mathfrak{T}$ is a tuple (S, s_0, T, I, Σ), where S is a set of states with initial state s_0, $T \subseteq S \times \Sigma \times S$ is a transition relation, Σ is a set of labels, and $I \subseteq T \times T$ is an irreflexive and symmetric relation of independent transitions. The binary relation $\prec$ on transitions defined by

$$(s,a,s_1) \prec (s_2,a,q) \Leftrightarrow \exists b.(s,a,s_1)I(s,b,s_2) \wedge (s,a,s_1)I(s_1,b,q) \wedge (s,b,s_2)I(s_2,a,q)$$

expresses that two transitions are instances of the same action, but in two different interleavings. We let $\sim$ be the least equivalence relation that includes $\prec$, i.e., the reflexive, symmetric, and transitive closure of $\prec$. The equivalence relation $\sim$ is used to group all transitions that are instances of the same action in all its possible interleavings. Additionally, I is subject to the following axioms:

- **A1**. $(s,a,s_1) \sim (s,a,s_2) \Rightarrow s_1 = s_2$
- **A2**. $(s,a,s_1)\ I\ (s,b,s_2) \Rightarrow \exists q.(s,a,s_1)\ I\ (s_1,b,q) \wedge (s,b,s_2)\ I\ (s_2,a,q)$
- **A3**. $(s,a,s_1)\ I\ (s_1,b,q) \Rightarrow \exists s_2.(s,a,s_1)\ I\ (s,b,s_2) \wedge (s,b,s_2)\ I\ (s_2,a,q)$
- **A4**. $(s,a,s_1)(\prec \cup \succ)(s_2,a,q)\ I\ (w,b,w') \Rightarrow (s,a,s_1)\ I\ (w,b,w')$ ◁

Axiom **A1** states that from any state, the execution of a transition leads to a unique state. This is a determinacy condition. Axioms **A2** and **A3** ensure that independent transitions can be executed in either order. Finally, **A4** ensures that the relation I is well defined. More precisely, **A4** says that if two transitions t and t' are independent, then all other transitions in the equivalence class $[t]_\sim$ (i.e., all other transitions that are instances of the same action but in different interleavings) are independent of t' as well, and vice versa. An alternative definition for **A4** can be given. Let $\mathfrak{I}(t)$ be the set $\{t' \mid t\ I\ t'\}$. Then, axiom **A4** is equivalent to this expression: **A4'**. $t \sim t_2 \Rightarrow \mathfrak{I}(t) = \mathfrak{I}(t_2)$.

This axiomatization of concurrent behaviour was defined by Nielsen and Winskel [72], but has its roots in the theory of traces [62], notably developed by Mazurkiewicz for trace languages, one of the simplest partial order models for concurrency. As shown in Figure 2.2, this axiomatization can be used to generate a so-called 'concurrency diamond' for any two independent transitions t and t', say, for $t = (s,a,s_1)$ and $t' = (s,b,s_2)$.

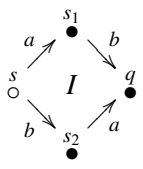


Figure 2.2: The 'concurrency diamond' defined by $t\ I\ t'$, where $t = (s,a,s_1)$ and $t' = (s,b,s_2)$. Concurrency is recognised by the I symbol inside the square. The initial state is marked by $\circ$.

2.1.3 Event Structures

Definition 2.3. A *labelled event structure* $\mathcal{E}$ is a tuple $(E, \preccurlyeq, \sharp, \eta, \Sigma)$, where E is a set of 'events' partially ordered by $\preccurlyeq$, the 'causal' dependency relation; events in an event structure are occurrences of actions in a system. Moreover, $\eta : E \to \Sigma$ is a labelling function from events to a set of labels Σ, and $\sharp \subseteq E \times E$ is an irreflexive and symmetric 'conflict' relation such that: if $e_1, e_2, e_3 \in E$ and $e_1 \sharp e_2 \preccurlyeq e_3$, then $e_1 \sharp e_3$; and, $\forall e \in E$ the set $\{e' \in E \mid e' \preccurlyeq e\}$ is finite. $\triangleleft$

The independence relation co on events is defined with respect to the causality $\preccurlyeq$ and conflict $\sharp$ relations. Two events e_1 and e_2 are said to be concurrent with each other, denoted by e_1 co e_2, if, and only if, $e_1 \not\preccurlyeq e_2$ and $e_2 \not\preccurlyeq e_1$ and $\neg(e_1 \sharp e_2)$. The notion of computation state for event structures is that of a 'configuration'. A configuration C is a conflict-free set of events, i.e., $\forall e_1, e_2 \in C.\ (e_1, e_2) \notin \sharp$, such that if $e \in C$ and $e' \preccurlyeq e$, then $e' \in C$. The initial configuration (or state) is by definition the empty configuration $\{\}$. Finally, a successor configuration C' of a configuration C is given by $C' = C \cup \{e\}$ such that $e \notin C$. Write $C \xrightarrow{e} C'$ for this relation, and let $\longrightarrow^*$ be defined similar to the nets case. Figure 2.3 shows a representation of event structures.

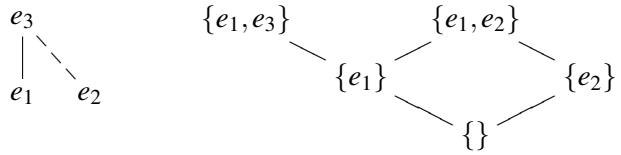


Figure 2.3: Graphical representation of an event structure where e_1 co e_2, $e_1 \preccurlyeq e_3$, and $e_2 \sharp e_3$ (on the left), together with the corresponding partial order of its configurations (on the right). The causal order relation $\preccurlyeq$ is read upwards through the straight lines. The dashed lines represent the conflict relation $\sharp$. In the partial order on the right, configurations are ordered by set inclusion.

2.1.4 A Uniform Representation

Despite being different mathematical structures, the three models just presented have a number of fundamental relationships between them, as well as with some models for interleaving concurrency. More precisely, TSI are noninterleaving transition-based representations of Petri nets, whereas event structures are unfoldings of TSI. This is analogous to the fact that LTS are interleaving transition-based representations of Petri nets while trees are unfoldings of LTS.

There are also relationships between TSI and LTS and between event structures and trees: LTS are those TSI with an empty independence relation I on transitions, and trees are those event structures with and empty relation co on events. In this way, partial order models can generalise the most important interleaving models in concurrency (and in verification), namely LTS, trees, and Kripke structures (which are the vertex-labelled counterparts of LTS models).

Since the results presented in further chapters are valid across all the models of concurrency previously mentioned, it is convenient to fix some notations to refer unambiguously to any of them. To this end, we use the notation coming from the TSI model and present the maps that determine a TSI model based on the primitives of Petri nets and event structures. Also, with no further distinctions we use the word 'system' when referring to any of these (partial order) models or to submodels therefrom, e.g., to LTS, infinite trees, or Kripke structures.

The main reason for this choice of notation is that the basic components of the TSI model can be easily and uniformly recognised in all the other models of concurrency studied here. Thus, the translations are simple and direct. Also, this generic setting allows one to see more clearly that the simple axiomatization of (true) concurrency presented for the TSI model also holds for the other partial order models of concurrency when analysing their local behaviour.

Petri Nets and Event Structures as TSI Models. A Petri net $\mathfrak{N} = (\mathcal{N}, M_0)$, where the 5-tuple $\mathcal{N} = (P, C, R, \theta, \Sigma)$ is a net as defined before and M_0 is its initial marking, can be represented as a TSI $\mathfrak{T} = (S, s_0, T, I, \Sigma)$ in the following way:

$$\begin{array}{lcl} S & = & \{M \subseteq P \mid M_0 \longrightarrow^* M\} \\ T & = & \{(M, a, M') \mid \exists t \in C.\ a = \theta(t), M \xrightarrow{t} M'\} \\ I & = & \{((M_1, a, M_1'), (M_2, b, M_2')) \mid \exists (t_1, t_2) \in \mathsf{par}. \\ & & a = \theta(t_1), b = \theta(t_2), M_1 \xrightarrow{t_1} M_1', M_2 \xrightarrow{t_2} M_2'\} \end{array}$$

where the set of states S of the TSI $\mathfrak{T}$ represents the set of reachable markings of $\mathfrak{N}$, the initial state s_0 is the initial marking M_0, the set of labels Σ remains the same, and T and I have the expected derived interpretations. Similarly, an event structure $\mathcal{E} = (E, \preccurlyeq, \sharp, \eta, \Sigma)$ determines a TSI $\mathfrak{T} = (S, s_0, T, I, \Sigma)$ by means of the following mapping:

$$\begin{array}{lcl} S & = & \{C \subseteq E \mid \{\} \longrightarrow^* C\} \\ T & = & \{(C, a, C') \mid \exists e \in E.\ a = \eta(e), C \xrightarrow{e} C'\} \\ I & = & \{((C_1, a, C_1'), (C_2, b, C_2')) \mid \exists (e_1, e_2) \in \mathsf{co}. \\ & & a = \eta(e_1), b = \eta(e_2), C_1 \xrightarrow{e_1} C_1', C_2 \xrightarrow{e_2} C_2'\} \end{array}$$

where the set of states S is the set of configurations of $\mathfrak{E}$, the initial state s_0 is the initial configuration $\{\}$, and, as before, the set of labels Σ remains the same in both models, and T and I have the expected derived TSI interpretations.

Notice that *actions* in a Petri net, *transitions* in a TSI and *events* in an event structure are all different. As said before, transitions are *instances* of actions, i.e., are actions relative to a particular interleaving. For instance, a Petri net composed of only two independent actions ($a \parallel b$ in CCS notation [66]) is represented by a TSI with four different transitions, since there are two possible interleavings in such a system, namely $a_1.b_2$ and $b_1.a_2$. Therefore each action in the Petri net for $a \parallel b$ becomes two different transitions in the corresponding TSI.

On the other hand, events are *occurrences* of actions, i.e., are actions relative to the causality relation. For instance, the Petri net representing the system defined by $(a+b).c$, where $a+b$ is the nondeterministic choice between actions a and b, and . is the sequential composition of such a choice with the action c, is represented by four events, instead of only three, because there are two different causal lines for the execution of action c, namely $a.c_1$ and $b.c_2$. Therefore, the Petri net action c becomes two different events c_1 and c_2 in the corresponding event structure.

Notation 2.4. Given a transition $t = (s,a,s')$, also written as $s \xrightarrow{a} s'$ or $s \xrightarrow{t} s'$ if no confusion arises, we have that: state s is called the 'source' of t, and write $\sigma(t) = s$; state s' is the 'target' of t, and write $\tau(t) = s'$; and a is the 'label' of t, and write $\delta(t) = a$. ◁

Remark 2.5. The systems we study may be finite or infinite, and this is always explicitly stated. However, they all are 'image-finite' [46], i.e., of finite branching. ◁

2.2 Modal Logics and Fixpoints

Modal logics are formalisms that offer an alternative paradigm of applying logical methods: instead of using the traditional languages of quantification (first-order or higher-order) to describe a structure, they look for a quantifier-free language with additional logical operators (called 'modalities') that represent the phenomenon at hand. In many cases, one ends up with a logical language that is much richer than propositional logic, and yet, unlike several languages with quantification, does not fall under the scope of classical undecidability limitations, thus often providing better decidability and complexity results than its rival first-order (FO) logic.

Modal logics can be extended in very simple ways which may turn out to be extremely expressive. For instance, modal logics can be used to express so-called 'temporal' properties by extending the original modal language with *fixpoint* operators. Notably, the 'modal mu-calculus' ($\mathcal{L}_\mu$ [14, 54], which is described later) is a small, yet expressive, temporal logic with modalities to reason about the actions that can be performed in a system.

Whereas modal logics are rather important in mathematical logic, temporal logics play a major role in informatics, and especially in concurrency as well as in systems verification. Temporal logics are special kinds of modal logics. They have modalities for reasoning about the way in which the truth of an assertion changes over time. In general, temporal logics can be seen as logics with a modality for a next step/time, and at least one operator to perform arbitrarily many sequences of steps. They can, therefore, be used to specify properties of the behaviour of a system in time by describing properties of its execution paths.

Temporal logics come in two varieties: *linear-time* and *branching-time*. In a 'linear-time temporal logic', at each point, there is only one possible future moment. On the other hand, in a 'branching-time temporal logic' the possible future moments of time have a tree-like structure, and so there may well be more than only one (as it is in the linear-time case). The modalities

of a temporal logic usually reflect the character of time assumed in the semantics of the logical language. Thus, in a linear-time temporal logic the modalities describe actions along a single time line. In contrast, in a branching-time temporal logic, the modalities reflect the branching nature of time by allowing quantification over various possible futures or execution paths.

The best known temporal logics in the linear-time and branching-time spectra are, respectively, the LTL and CTL families [48]. They are used for many practical applications, especially for 'model-checking' [17] hardware and software systems. It is difficult (or perhaps impossible) to argue that one kind of temporal logic is better than the other in all possible contexts since this depends on the sort of properties one would like to express and verify.

We have no particular reasons to prefer any kind of temporal properties over the other. Hence, in this thesis we study logics based on the mu-calculus (and the mu-calculus itself) since it can be used to express both linear-time and branching-time properties. But first, we review Hennessy–Milner logic (HML [46]), a precursor modal language to the mu-calculus, which has played a major role in computer science, and especially in the specification of properties of concurrent systems. Then we turn our attention to the mu-calculus simply by adding fixpoint operators to HML. After that, we look at the logical equivalences induced by these logics and how they have been used as equivalences for concurrency. See [14, 91] for further information on modal logics, the mu-calculus, or the equivalences induced by such logics.

2.2.1 Hennessy–Milner Logic

Hennessy–Milner logic (HML [46]) is a modal logic of actions that has its roots in the study of process algebras for concurrent and communicating systems (chiefly, of Milner's CCS). It was intended as an alternative, logical approach to the formalisation of the notion of 'observational equivalence' for concurrent and reactive systems. As usual for modal logics, HML formulae are interpreted over the set of states of a system, e.g., over the set of states of an LTS.

Definition 2.6. *Hennessy–Milner logic* (HML [46]) has formulae ϕ built from a set Σ of labels $a, b, \ldots$ by the following grammar:

$$\phi ::= \mathrm{ff} \mid \mathrm{tt} \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \langle a \rangle \phi_1 \mid [a]\phi_1$$

where ff and tt are the false and true boolean constants, respectively, $\wedge$ and $\vee$ are boolean operators, and $\langle a \rangle \phi_1$ and $[a]\phi_1$ are the only two modalities of the logic. ◁

The meanings of ff, tt, $\wedge$, and $\vee$ are the usual ones as in propositional logic. Besides these, the semantics of the 'diamond' modality $\langle a \rangle \phi_1$ is, informally, that at a given state it is possible to perform an a-labelled action to a state where ϕ_1 holds; and dually for the 'box' modality $[a]\phi_1$. Following [91], we give the denotation of HML formulae inductively using an LTS model $\mathfrak{T} = (S, T, \Sigma)$. Then, the semantics of HML formulae are as follows:

Definition 2.7. An *HML model* $\mathfrak{T}$ of an HML formula ϕ is an LTS (S, T, Σ). Then, the denotation $\|\phi\|^{\mathfrak{T}}$ of a formula ϕ is given as follows (omitting the superscript $\mathfrak{T}$):

$$\begin{aligned}
\|\mathrm{ff}\| &= \emptyset \\
\|\mathrm{tt}\| &= S \\
\|\phi_1 \wedge \phi_2\| &= \|\phi_1\| \cap \|\phi_2\| \\
\|\phi_1 \vee \phi_2\| &= \|\phi_1\| \cup \|\phi_2\| \\
\|\langle a \rangle \phi_1\| &= \{s \in S \mid \exists s'.\ s \xrightarrow{a} s' \wedge s' \in \|\phi_1\|\} \\
\|[a]\,\phi_1\| &= \{s \in S \mid \forall s'.\ s \xrightarrow{a} s' \Rightarrow s' \in \|\phi_1\|\}
\end{aligned}$$

And the 'satisfaction' relation $\models$ is then defined in the obvious way: $s \models \phi$ iff $s \in \|\phi\|$. ◁

One of the most interesting properties of HML is that it characterises 'bisimilarity' [46], the equivalence relation induced by modal logic. A bisimulation equivalence relation between two rooted systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$ with initial states s_0 and q_0, respectively, is an equivalence relation $\sim_b$ such that $s_0 \sim_b q_0$ if, and only if, they satisfy the same set of HML formulae. Then, we say that the two states s_0 and q_0 (or equivalently the two corresponding rooted systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$) are bisimilar iff there is a bisimulation equivalence relation between them.

The importance of this result in concurrency theory is that before Milner and Hennessy introduced HML, bisimilarity—or so-called observational equivalence—was already a cornerstone in the theory of concurrency developed by Milner in the 1970's, which ended up with the definition of CCS as a calculus for concurrent and communicating processes [65]. Almost at the same time, bisimilarity was also studied as an equivalence for concurrency in the work of Park on automata on infinite words [74]. So, we should actually say that the definition of bisimulation equivalences and in particular of bisimilarity as a behavioural equivalence for concurrency is due to both Milner and Park. The interested reader is referred to [84] for a nice tutorial paper on bisimulation and its origins in concurrency, logic, and set theory.

Back to HML, recall that this modal logic was initially defined as an alternative, logical approach to understanding process equivalence in the context of CCS; Milner and Hennessy [46] showed that if two CCS processes are bisimilar, or in their words "observationally equivalent", then they satisfy the same set of HML formulae. They found, therefore, a strong correspondence between the logical equivalence induced by HML and an equivalence for concurrency (bisimilarity or observational equivalence in this case), modulo LTS, the class of models that Milner and Hennessy used for giving the semantics of CCS process expressions.

Even though HML is quite a natural logic for studying process equivalences, it is not so useful as a specification language of system properties as it cannot express many temporal properties. Due to this, more expressive logics have been studied. We now review one of such logics, the modal mu-calculus, which has strong connections to HML and a beautiful theory based on the addition of fixpoint operators to modal logic.

2.2.2 Fixpoints and the Modal Mu-Calculus

Fixpoint logics or mu-calculi [14] are logics that make use of fixpoint operators; in particular, the modal mu-calculus is a simple extension of modal logic with fixpoint operators. The mu-calculus as we use it nowadays was defined by Kozen [54], but it can also be seen as HML with fixpoint operators. The use of fixpoints in program logics was, however, not new by the time the mu-calculus was proposed. It actually dates back to Scott, De Bakker, and Park (amongst others) in the late 1960's and early 1970's in the contexts of program semantics and verification.

In informatics, and especially in concurrency and systems verification, the main motivation for extending a logic with fixpoint operators is the ability to express temporal properties of systems, this is their (possibly infinite) behaviour. In the remainder of this section we describe the modal mu-calculus but, before giving its formal definition, let us first state some results that relate to fixpoints and their ubiquity in ordered structures, particularly in lattice theory.

Fixpoints in Ordered Structures. Before stating one of the results on fixpoints that is relevant to this work (which is a direct consequence of the Knaster–Tarski fixpoint theorem [93]), let us first introduce some of the ordered structures of our interest, namely partially ordered sets, lattices, and complete lattices. A 'partially ordered set' (poset) $(A, \leq_A)$ is a set A together with a reflexive, transitive and anti-symmetric relation $\leq_A$ on its elements. A 'lattice' $\mathfrak{A} = (A, \leq_A)$ is a poset where for every two elements x and y in A, arbitrary meets (written $x \times y$) and joins (written $x + y$) exist. If, moreover, arbitrary meets and joins exist for any subset $B \subseteq A$, then $\mathfrak{A}$ is a 'complete lattice'. Fixpoints are ubiquitous in complete lattices as described next.

Fixpoints can be seen as equilibrium points. Their mathematical definition is simple: given a function f, we say that x is a fixpoint of f iff $x = f(x)$; it is a 'pre-fixpoint' of f if $f(x) \leq_A x$ and a 'post-fixpoint' if $x \leq_A f(x)$. As we shall see, fixpoint theory is rather useful in (mathematical) logic when f is monotonic and its domain is a complete lattice. In particular, we are interested in the Knaster–Tarski fixpoint theorem [93], which says that the set of fixpoints of a monotone function in a complete lattice is also a complete lattice, i.e., that the fixpoints themselves also form a complete lattice. An immediate consequence is the following result:

Theorem 2.8. (Least and greatest fixpoints in a complete lattice [93]) Let $f : A \to A$ be a monotone mapping on a complete lattice $\mathfrak{A} = (A, \leq_A)$. Then f has a least fixpoint x_μ and a greatest fixpoint x_ν determined, respectively, by the pre-fixpoints and post-fixpoints of f:

$$x_\mu = \bigotimes \{x \in A \mid f(x) \leq_A x\}$$
$$x_\nu = \bigoplus \{x \in A \mid x \leq_A f(x)\}$$

where $\bigotimes$ and $\bigoplus$ are the generalisations to arbitrary sets of the operators $\times : A^2 \to A$ and $+ : A^2 \to A$ on pairs of elements as described before. ◁

The Modal Mu-Calculus. With these concepts in mind we are now ready to present the modal mu-calculus in full as well as some properties of mu-formulae.

Definition 2.9. The *modal mu-calculus* ($\mathcal{L}_\mu$ [54]) has formulae ϕ built from a set Var of variables $Y, Z, \ldots$ and a set Σ of labels $a, b, \ldots$ by the following grammar:

$$\phi ::= Z \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \langle a \rangle \phi_1 \mid [a]\phi_1 \mid \mu Z.\phi_1 \mid \nu Z.\phi_1$$

Also, define the boolean constants as $\mathrm{ff} \stackrel{\text{def}}{=} \mu Z.Z$ and $\mathrm{tt} \stackrel{\text{def}}{=} \nu Z.Z$; and assume these abbreviations: $\langle K \rangle \phi_1$ for $\bigvee_{a \in K} \langle a \rangle \phi_1$ and $[K]\phi_1$ for $\bigwedge_{a \in K} [a]\phi_1$, where $K \subseteq \Sigma$, as well as $[-]\phi_1$ for $[\Sigma]\phi_1$ and $[-K]\phi_1$ for $[\Sigma \setminus K]\phi_1$, and similarly for the diamond modality. ◁

The meanings of the boolean and modal operators are as for HML. The two additional operators of $\mathcal{L}_\mu$, namely $\mu Z.\phi$ and $\nu Z.\phi$, are the minimal and maximal fixpoint operators of the logic. The denotations of $\mathcal{L}_\mu$ formulae are given over the set of states of a system as follows:

Definition 2.10. A *mu-calculus model* $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$ is an LTS $\mathfrak{T} = (S, T, \Sigma)$ together with a valuation $\mathcal{V} : \mathrm{Var} \to 2^S$. The denotation $\|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$ of a mu-calculus formula ϕ in the model $\mathfrak{M}$ is a subset of S given by the following rules (omitting the superscript $\mathfrak{T}$):

$$\begin{array}{lcl}
\|Z\|_{\mathcal{V}} & = & \mathcal{V}(Z) \\
\|\phi_1 \wedge \phi_2\|_{\mathcal{V}} & = & \|\phi_1\|_{\mathcal{V}} \cap \|\phi_2\|_{\mathcal{V}} \\
\|\phi_1 \vee \phi_2\|_{\mathcal{V}} & = & \|\phi_1\|_{\mathcal{V}} \cup \|\phi_2\|_{\mathcal{V}} \\
\|\langle a \rangle \phi_1\|_{\mathcal{V}} & = & \{s \in S \mid \exists s' \in S.\ s \xrightarrow{a} s' \wedge s' \in \|\phi_1\|_{\mathcal{V}}\} \\
\|[a]\phi_1\|_{\mathcal{V}} & = & \{s \in S \mid \forall s' \in S.\ s \xrightarrow{a} s' \Rightarrow s' \in \|\phi_1\|_{\mathcal{V}}\} \\
\|\mu Z.\phi\|_{\mathcal{V}} & = & \bigcap \{Q \in 2^S \mid \|\phi\|_{\mathcal{V}[Z:=Q]} \subseteq Q\} \\
\|\nu Z.\phi\|_{\mathcal{V}} & = & \bigcup \{Q \in 2^S \mid Q \subseteq \|\phi\|_{\mathcal{V}[Z:=Q]}\}
\end{array}$$

where $\mathcal{V}[Z := Q]$ is the valuation $\mathcal{V}'$ which agrees with $\mathcal{V}$ save that $\mathcal{V}'(Z) = Q$. ◁

Notice that the denotation of the fixpoint operators is an application of the Knaster–Tarski fixpoint theorem (by using Theorem 2.8) where the function f is the mapping $\|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$, the order relation $\leq_A$ is the subset inclusion relation $\subseteq$, and $\bigotimes$ and $\bigoplus$ are $\bigcap$ and $\bigcup$, respectively.

Finally, let us define the subformulae of a modal mu-calculus formula ϕ; formally, the 'subformula set' $Sub(\phi)$ of an $\mathcal{L}_\mu$ formula ϕ is given by the *Fischer–Ladner closure* of $\mathcal{L}_\mu$ formulae in the following way (we follow the notation for subformula sets used in [55]):

$$\begin{array}{lcl}
Sub(Z) & = & \{Z\} \\
Sub(\phi_1 \wedge \phi_2) & = & \{\phi_1 \wedge \phi_2\} \cup Sub(\phi_1) \cup Sub(\phi_2) \\
Sub(\phi_1 \vee \phi_2) & = & \{\phi_1 \vee \phi_2\} \cup Sub(\phi_1) \cup Sub(\phi_2) \\
Sub(\langle a \rangle \phi_1) & = & \{\langle a \rangle \phi_1\} \cup Sub(\phi_1) \\
Sub([a]\phi_1) & = & \{[a]\phi_1\} \cup Sub(\phi_1) \\
Sub(\mu Z.\phi_1) & = & \{\mu Z.\phi_1\} \cup Sub(\phi_1) \\
Sub(\nu Z.\phi_1) & = & \{\nu Z.\phi_1\} \cup Sub(\phi_1)
\end{array}$$

We finish this presentation of the modal mu-calculus with a note on its expressive power. One of the most interesting features of the mu-calculus is that most interesting temporal logics used for software and hardware verification can be embedded into $\mathcal{L}_\mu$. The translation of CTL is straightforward, e.g., as shown in [55]; other mappings, such as the one for CTL* and thus for LTL as well, are not so simple but still possible, e.g., as presented in [18].

The source of the high expressivity of the mu-calculus comes from the freedom to mix (or alternate) minimal and maximal fixpoint operators arbitrarily. In fact, Bradfield showed that this alternation defines a strict hierarchy [11], one the most remarkable results regarding the expressivity of $\mathcal{L}_\mu$. Perhaps, the other most interesting result on the expressivity of the mu-calculus was presented in [49], where Janin and Walukiewicz showed that, up to bisimulation, $\mathcal{L}_\mu$ is as expressive as monadic second-order (MSO) logic on transition systems. These results, amongst others, have made the mu-calculus one of the most important logics in informatics.

2.3 Logic Games for Verification

A logic game [9] for verification is played by two 'players', a "Verifier" ($\exists$) and a "Falsifier" ($\forall$), in order to show the truth or falsity of a given property. In these games the Verifier tries to show that the property holds, whereas the Falsifier wants to refute such an assertion. Solving these games amounts to answering the question of whether the Verifier has a 'strategy' to win all plays in the game. Usually the 'board' where the game is played is a graph structure in which each position of the board belongs to only one of the players. Due to this, the games are sequential as at any point in time only one of them can play. A game can be of finite or infinite duration, and the winner is determined by a set of winning rules (a winning condition).

There are different questions that can be asked in a verification game. For instance, one could ask whether a logic formula has at least one model (a satisfiability problem), or whether a model satisfies a temporal property (a model-checking problem), or whether two systems are equivalent with respect to some notion of equivalence (an equivalence-checking problem). In this thesis, we are interested in two particular problems in verification: *bisimulation* and *model-checking* for concurrent systems with *partial order semantics*. There are some aspects of the games of our interest that need to be pointed out before moving to their study.

Traditionally, the *players* of a logic game have been given names depending on the kind of verification game that is being played. For instance, in a bisimulation game the Verifier is called Duplicator whereas the Falsifier is called Spoiler. Similarly, in other kinds of games, the Verifier and Falsifier have been called, respectively, Eloise and Abelard, Player $\exists$ and Player $\forall$, Builder and Critic, Player $\Diamond$ and Player $\Box$, Proponent and Opponent, Eve and Adam, or simply Player I and Player II. In order to have a uniform notation in this document, we have chosen to call them "Eve" and "Adam" regardless of the particular sort of logic game they are playing.

The *boards* where the games are played also have different structures depending on the kind of verification problem that one wants to solve. In a bisimulation game the board is made up with the elements of the two systems that are being analysed, e.g., each position in the game board is an element of the Cartesian product of the state sets of the two systems. On the other hand, in a model-checking game the board is composed of pairs of elements where one of the components is an element of the model being checked and the other component relates to the temporal property under consideration. All these game features are formally defined in this and further chapters whenever new bisimulation or model-checking games are presented.

Finally, notice that by playing a logic game the two players jointly define sequences of positions of the game board. Such sequences are called plays of the game. Let Γ be the set of plays of a game and $\mathfrak{B}$ be a game board, i.e., a set of positions in the game. The set of plays is, then, a prefix closed set of finite and infinite sequences of positions. We say that a play is partial if, and only if, it is the prefix of another play or it is complete otherwise. Moreover, a deterministic *strategy* is a function $\lambda : \Gamma \to \mathfrak{B}$ from plays in Γ to positions of the game board, so that such strategies define the next move a player will make.

In some cases, in order for a player to make a move he or she only needs to know their current position. In these cases, their strategies can be defined as functions on the set of positions of the board rather than on Γ. These strategies are called 'history-free' (also known as positional or memoryless); formally, a history-free strategy is a function $\lambda : \mathfrak{B} \to \mathfrak{B}$. Finally, a *winning* strategy is a strategy that guarantees that the player that uses it can win all plays of the game. This thesis only features logic games with history-free winning strategies.

2.3.1 Bisimulation Games

Bisimulation games are formal and interactive characterisations of a family of equivalence relations called bisimulation equivalences. One of the simplest bisimulation equivalences is 'bisimilarity', the equivalence relation induced by modal logic. This bisimulation equivalence was defined, independently, by Johan van Benthem [8] while studying the semantics of modal logic (and its connections with FO logic), and a few years later by Milner and Park [65, 74] while studying the behaviour of concurrent systems with interleaving semantics.

Bisimulation equivalences are not as strong as isomorphism, and this was one of the motivations for studying them in concurrency theory. However, the idea of having an equivalence relation not as strong as isomorphism has been around long before their use in concurrency. About 80 years ago Alfred Tarski formulated the notion of two structures being elementarily equivalent iff they satisfy the same set of FO sentences. Some years later, this idea was recast and developed by logicians Roland Fraïssé and Andrzej Ehrenfeucht in what is now known as the Ehrenfeucht-Fraïssé (EF) games, thus providing a game-theoretic approach to comparing different mathematical structures. In particular, EF games are used in concurrency to char-

acterise bisimulation equivalences, and hence, to *compare* the *behaviour* of different parallel processes. In this case, two interleaving systems are said to be observationally equivalent according to Milner (or "elementary equivalent" in Tarski's words) iff the same modal formulae are true in both structures, i.e., in both models for concurrency.

More precisely, a bisimulation game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$ is a formal representation of a bisimulation equivalence $\sim_{eq}$ between two systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$. Whereas Eve believes that $\mathfrak{T}_1 \sim_{eq} \mathfrak{T}_2$, Adam wants to show that $\mathfrak{T}_1 \not\sim_{eq} \mathfrak{T}_2$. All plays start in the initial position (s_0, q_0) consisting of the initial states of the systems, and the players take alternating turns—although Adam always plays first and chooses where to play. Thus, in every round of the game Adam makes the first move in either system according to a set of rules, and then Eve must make an equivalent move on the other system (depending on $\sim_{eq}$); the game can proceed in this way indefinitely. Thus, the plays of the game can be of finite or infinite length. All plays of infinite length are winning for Eve; in the case of plays of finite length, the player who cannot make a move loses the game. These winning conditions apply to all bisimulation games we study in this thesis.

In concurrency theory, bisimulation games are often used to show that two given concurrent or reactive systems interact equivalently (with respect to $\sim_{eq}$) with an arbitrary environment. Since the exact definition of a particular bisimulation equivalence relation $\sim_{eq}$ can be altered (strengthened or weakened) by the kinds of properties that one wants to analyse, then the set of rules for playing a bisimulation game can be different in each game. Nowadays, the best known bisimulation game for interleaving concurrency is the one that characterises 'strong' bisimilarity (sb [46]), the bisimulation equivalence relation induced by HML.

However, in order to capture properties of partial order models of concurrency rather than of interleaving ones, equivalence relations finer than strong bisimilarity have been defined as well as their associated bisimulation games. Two of the most important bisimulation games for partial order models are the ones that characterise 'history-preserving' bisimilarity (hpb [82]) and 'hereditary history-preserving' bisimilarity (hhpb [51]). Both (history-preserving) bisimulation equivalences, together with a deep study of their applications to concurrency and systems verification, can be found in [35]. Let us now introduce some concepts needed to present the bisimulation games that characterise sb, hpb, and hhpb.

Strong Bisimulation Games. A bisimulation game for strong bisimilarity is played on a board $\mathfrak{B}$ composed of pairs (s,q) of states s and q of two systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$, respectively. Such a pair is a position of the game board $\mathfrak{B}$ and is called a 'configuration' of the game.[2] The position (s_0, q_0), where s_0 and q_0 are the initial states of $\mathfrak{T}_1$ and $\mathfrak{T}_2$, is the initial configuration of the game. Since the strategies of the game are history-free, then a strategy λ is a partial function on $\mathfrak{B} \subseteq S \times Q$, where S and Q are the state sets of the two systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$, respectively.

[2] Do not confuse with the configurations of an event structure, which are the computation states in such a model.

Notation 2.11. Since a system has a uniquely defined initial state, a bisimulation game can be unambiguously presented as either $\mathcal{G}(\mathfrak{T}_1,\mathfrak{T}_2)$ or $\mathcal{G}(s_0,q_0)$ if the two systems are obvious from the context. Also, since bisimulation games are symmetric, we omit the subscript in $\mathfrak{T}$ whenever referring to either system. ◁

Definition 2.12. (Strong bisimulation games) Let (s,q) be a configuration of the game $\mathcal{G}(\mathfrak{T}_1,\mathfrak{T}_2)$. There are two players, Adam and Eve, and Adam always plays first and chooses where to play. R_{sb} is a strong bisimulation relation between $\mathfrak{T}_1$ and $\mathfrak{T}_2$ if:

- (Base case) The initial configuration (s_0,q_0) is in R_{sb}.
- ($\sim_{sb}$ rule) If (s,q) is in R_{sb} and Adam chooses a transition in $\mathfrak{T}$, say a transition $s \xrightarrow{a} s'$ of $\mathfrak{T}_1$, then Eve must choose a transition in the other system (any $q \xrightarrow{a} q'$ of $\mathfrak{T}_2$ in this case), such that the new configuration (s',q') is in R_{sb} as well.

Adam wins the game if eventually Eve cannot make a move; otherwise Eve wins the game. We say that $\mathfrak{T}_1 \sim_{sb} \mathfrak{T}_2$ iff Eve has a winning strategy for the sb game $\mathcal{G}(\mathfrak{T}_1,\mathfrak{T}_2)$. ◁

Clearly, bisimulation games do not capture any information of partial order models that is not already present in their interleaving counterparts. For this reason, games for strong bisimilarity are considered games for interleaving concurrency rather than for partial order concurrency. In order to capture properties of partial order models of concurrency, one has to recognise at the very least when two transitions of a system are independent and hence can be executed in parallel. This feature is captured by the following finer bisimulation games.

History-Preserving Bisimulation Games. A game for history-preserving bisimilarity is a bisimulation game as presented before (i.e., as the one for strong bisimilarity) with a further *synchronisation* requirement on transitions. Such a synchronisation requirement makes the selection of transitions by Eve more restricted. Let us first define this notion of synchronisation on transitions before making a formal presentation of the game.

A possibly empty sequence of transitions $\pi = [t_1,...,t_k]$ is a *run* of a system $\mathfrak{T}$. Let $\Pi_{\mathfrak{T}}$ be the set of runs of $\mathfrak{T}$ and $\rho(\pi)$ be the last transition of π. Define $\varepsilon = \rho([\,]) $ and $s_0 = \sigma(\varepsilon) = \tau(\varepsilon)$, for an empty sequence $[\,]$. Given a run π and a transition t, the sequence $\pi.t$ denotes the run π extended with t. Let $\pi_1 \in \Pi_{\mathfrak{T}_1}$ and $\pi_2 \in \Pi_{\mathfrak{T}_2}$ for two systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$. We say that the pair of runs $(\pi_1.u,\pi_2.v)$ is 'synchronous' iff $(\rho(\pi_1),u) \in I_1 \Leftrightarrow (\rho(\pi_2),v) \in I_2$, where I_1 and I_2 are the independence relations of $\mathfrak{T}_1$ and $\mathfrak{T}_2$, and the posets induced by $\pi_1.u$ with I_1 and $\pi_2.v$ with I_2 are isomorphic.[3] By definition $(\varepsilon,\varepsilon)$ is synchronous. As it is more convenient to define hpb games on pairs of runs rather than of states, a configuration of the game will be a pair of runs.

[3]Given a run π and an independence relation I, there is a poset $(E,\leq_E)$ induced by π with I such that E has as elements the event occurrences associated with the transitions in π and where the partial order relation $\leq_E$ is defined by the event structure unfolding of the system whose independence relation is I.

Definition 2.13. (History-preserving bisimulation games) Let (π_1, π_2) be a configuration of the game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$. The initial configuration of the game is $(\varepsilon, \varepsilon)$. The relation R_{hpb} is a history-preserving (hp) bisimulation between $\mathfrak{T}_1$ and $\mathfrak{T}_2$ iff it is a strong bisimulation relation between $\mathfrak{T}_1$ and $\mathfrak{T}_2$ and:

- (Base case) The initial configuration $(\varepsilon, \varepsilon)$ is in R_{hpb}.
- ($\sim_{hpb}$ rule) If (π_1, π_2) is in R_{hpb} and Adam chooses a transition u in either system, say in $\mathfrak{T}_1$, such that $u = \tau(\rho(\pi_1)) \xrightarrow{a} s'$, then Eve must choose a transition v in the other system such that $v = \tau(\rho(\pi_2)) \xrightarrow{a} q'$ and the new configuration $(\pi_1.u, \pi_2.v)$ is synchronous, i.e., $(\pi_1.u, \pi_2.v)$ is in R_{hpb} as well.

Adam wins the game if eventually Eve cannot make a move; otherwise Eve wins the game. We say that $\mathfrak{T}_1 \sim_{hpb} \mathfrak{T}_2$ iff Eve has a winning strategy for the hpb game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$. ◁

Hereditary History-Preserving Bisimulation Games. A game stronger than an hpb game is a game for hereditary history-preserving bisimilarity (hhpb), which is an hpb game extended with *backtracking* moves. These backtracking moves are restricted to transitions that are said to be 'backwards enabled'. More specifically, let $\pi(i)$ be the i-th transition in π. Given a run $\pi = [t_1, ..., t_k]$, a transition $\pi(i)$ is 'backwards enabled' iff it is independent of all transitions t_j that appear after it in π, i.e., iff for all t_j in $\{\pi(i+1), ..., \pi(k)\}$ we have that $\pi(i)\ I\ t_j$.

This definition captures the fact that backwards enabled transitions are the terminal elements of the partial order induced by the independence relation I on the transitions in π. Now, let $\pi - \pi(i)$ be the sequence of transitions π without its i-th element $\pi(i)$. It should be clear that if $\pi(i)$ is backwards enabled, then the partial order induced by I on those transitions in $\pi - \pi(i)$ is just the same partial order induced by I on π without the terminal element or transition $\pi(i)$. Formally, an hhpb game is defined as follows:

Definition 2.14. (Hereditary history-preserving bisimulation games) Let (π_1, π_2) be a configuration of the game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$. The initial configuration is $(\varepsilon, \varepsilon)$. The relation R_{hhpb} is a hereditary history-preserving (hhp) bisimulation between $\mathfrak{T}_1$ and $\mathfrak{T}_2$ iff it is a history-preserving bisimulation relation between $\mathfrak{T}_1$ and $\mathfrak{T}_2$ and:

- (Base case) The initial configuration $(\varepsilon, \varepsilon)$ is in R_{hhpb}.
- ($\sim_{hhpb}$ rule) If (π_1, π_2) is in R_{hhpb} and Adam deletes, say from π_1, a transition $\pi_1(i)$ that is backwards enabled, then Eve must delete the transition $\pi_2(i)$ from the history of the game in the other system, provided that $\pi_2(i)$ is also backwards enabled and that the new configuration $(\pi_1 - \pi_1(i), \pi_2 - \pi_2(i))$ is in R_{hhpb} as well.

Adam wins the game if eventually Eve cannot make a move; otherwise Eve wins the game. We say that $\mathfrak{T}_1 \sim_{hhpb} \mathfrak{T}_2$ iff Eve has a winning strategy for the hhpb game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$. ◁

Unlike the bisimulation game for strong bisimilarity, $\sim_{sb}$, which is a game for interleaving concurrency, both history-preserving bisimulation games presented here can capture properties of partial order models and differentiate them from their interleaving counterparts. The simplest example is the case of the two CCS process expressions $a \parallel b$ and $a.b + b.a$, which are equivalent from an interleaving point of view, but different when considering any partial order semantics for them, say using Petri nets, event structures, or TSI models.

2.3.2 Model-Checking Games

Model-checking games [37, 97], also called Hintikka evaluation games, are logic games played in a formula ϕ and a mathematical model $\mathfrak{M}$. In a model-checking game $\mathcal{G}(\mathfrak{M},\phi)$ the goal of Eve is to show that $\mathfrak{M} \models \phi$, while Adam believes that $\mathfrak{M} \not\models \phi$.

In concurrency theory and program verification, most usually ϕ is a modal or a temporal formula and $\mathfrak{M}$ is a Kripke structure or an LTS, and the two players play the model-checking game $\mathcal{G}(\mathfrak{M},\phi)$ by picking single elements of $\mathfrak{M}$, according to the game rules defined by ϕ. For now, let us consider model-checking games played on interleaving models for concurrency, say in an LTS, and on formulae given as logical specifications in the mu-calculus.

The game we are about to describe is the local model-checking procedure for mu-calculus verification as defined by Stirling [89]. It is a natural game interpretation of the tableau-based method for mu-calculus model-checking introduced by Stirling and Walker [92]. The game is naturally played on interleaving models of concurrency. However, it can also be used to model-check partial order models of concurrency, such as Petri nets, if one considers the one-step interleaving semantics of such models, e.g., as done by Bradfield and Stirling [13].

Local Model-Checking Games in the Mu-Calculus. A local model-checking game $\mathcal{G}(\mathfrak{M},\phi)$ is played on a mu-calculus model $\mathfrak{M} = (\mathfrak{T},\mathcal{V})$, where $\mathfrak{T} = (S, s_0, T, \Sigma)$ is an interleaving system, and on a mu-calculus formula ϕ. Since the game is local, this is, it answers to the question of whether the initial state s_0 satisfies ϕ, then it can also be presented as $\mathcal{G}_{\mathfrak{M}}(s_0,\phi)$, or even as $\mathcal{G}(s_0,\phi)$ whenever the model $\mathfrak{M}$ is clear from the context. The board in which the game is played has the form $\mathfrak{B} \subseteq S \times \mathit{Sub}(\phi)$, where $\mathit{Sub}(\phi)$ is the set of subformulae of the modal mu-calculus formula ϕ (as defined by its Fischer–Ladner closure).

A play is a possibly infinite sequence of configurations $C_0, C_1, \ldots$; each $C_i = (s,\psi)$ is an element of the board $\mathfrak{B}$, i.e., it is a position of the game. Every play starts in $C_0 = (s_0,\phi)$, and proceeds according to the rules of the game, given below. Two deterministic rules control the unfolding of fixpoint operators. Moreover, given a configuration (s,ψ), the rules for $\vee$ and $\wedge$ make, respectively, Eve and Adam choose a next configuration (s,φ) which is determined by the subformula set of ψ. Similarly, the rules for $\langle\ \rangle$ and $[\ ]$ make, respectively, Eve and Adam choose a next configuration (q,ψ) which is determined by those transitions t such that $s = \sigma(t)$

and $q = \tau(t)$. These conditions can be captured in the following way. Let (s, ϕ) be the current configuration of the game; the next configuration of the game is defined by the following rules:

- if $\phi = \mu Z.\varphi$ (resp. $\phi = \nu Z.\varphi$), then Eve (resp. Adam) replaces $\mu Z.\varphi$ (resp. $\nu Z.\varphi$) by its associated fixpoint variable Z and the next configuration is (s, Z).
- if $\phi = Z$ such that $\psi = \mu Z.\varphi$ (resp. $\psi = \nu Z.\varphi$) for some mu-calculus formula ψ, then Eve (resp. Adam) unfolds the fixpoint and the next configuration is (s, φ).
- if $\phi = \psi_1 \vee \psi_2$ (resp. $\phi = \psi_1 \wedge \psi_2$), then Eve (resp. Adam) chooses some ψ_i, for $i \in \{1, 2\}$, and the next configuration is (s, ψ_i).
- if $\phi = \langle a \rangle \psi$ (resp. $\phi = [a]\psi$), then Eve (resp. Adam) chooses a transition $s \xrightarrow{a} s'$ and the next configuration is (s', ψ).

Finally the following rules are the winning conditions that determine a unique winner for every finite or infinite play $C_0, C_1, ...$ in a game $\mathcal{G}(s_0, \phi)$. Adam wins a finite play $C_0, C_1, ..., C_k$ or an infinite play $C_0, C_1, ...$ iff:

1. $C_k = (s, Z)$ and $s \notin \mathcal{V}(Z)$.
2. $C_k = (s, \langle a \rangle \psi)$ and $\{s' \mid s \xrightarrow{a} s'\} = \emptyset$.
3. The play is of infinite length and there exists a mu-calculus formula Z which is both the least fixpoint of some subformula $\mu Z.\psi$ and the syntactically outermost variable in ϕ that occurs infinitely often in the game.

Dually, Eve wins a finite play $C_0, C_1, ..., C_n$ or an infinite play $C_0, C_1, ...$ iff:

1. $C_k = (s, Z)$ and $s \in \mathcal{V}(Z)$.
2. $C_k = (s, [a]\psi)$ and $\{s' \mid s \xrightarrow{a} s'\} = \emptyset$.
3. The play is of infinite length and there exists a mu-calculus formula Z which is both the greatest fixpoint of some subformula $\nu Z.\psi$ and the syntactically outermost variable in ϕ that occurs infinitely often in the game.

Then $s_0 \models \phi$ iff Eve has a winning strategy in the model-checking game $\mathcal{G}(s_0, \phi)$.

Decidability and Determinacy

There are two properties related to games which are of interest from an algorithmic view point: *decidability* and *determinacy*. A logic game for verification is 'decidable' if, and only if, there exists a decision procedure (i.e., an algorithm that always terminates with a "yes" or "no" answer) that solves all possible instances of a game. Apart from the game for hhpb, the games that have been presented in this chapter, either for bisimulation or for mu-calculus model-checking, are all decidable when the models under consideration are finite.

Whereas the undecidability of hhpb was shown in [52], the decidability of the weaker hpb was shown in [50] with an exponential time complexity; strong bisimilarity is also decidable,

and even solvable in polynomial time [53]. On the other hand, we know that mu-calculus model-checking on finite systems is decidable [23] but the exact complexity is still an open problem; at present, it is widely known that the problem belongs to the NP∩co-NP complexity class [22]. In particular, the local model-checking algorithm of Stirling presented before runs in exponential time; however, since it is local, it has the advantage that one could use it to verify some classes of infinite-state systems as the whole model need not be constructed beforehand.

Determinacy is another important property. A game is 'determined' if, and only if, for every instance of the game it is always the case that one of the two players has a winning strategy. Thus, determinacy is simply a guarantee of the existence of winning strategies. Traditionally, determinacy has been studied with respect to another property of games, namely that of being of 'perfect' or 'imperfect' information. Roughly speaking, a game is of perfect information if both players have complete knowledge of the whole history of the game, so that when they make a move they are fully informed of the previous choices of their opponent. Otherwise, the game is of imperfect information. The games presented in this chapter are all of perfect information, but this thesis features games of perfect and imperfect information in other chapters.

Another property also related to determinacy is the kind of winning conditions under consideration. Such winning conditions define the sets of plays that are winning for each player. These sets are called winning sets, and in the case of the bisimulation and model-checking games just presented they are (topologically) quite simple; in particular they are 'Borel' sets. It is well known that all two-player perfect-information infinite games whose winning conditions define Borel winning sets are determined [59]. From this result follows that the bisimulation and model-checking games we have described in this chapter are determined.

Finally, notice that determinacy is a *mathematical* property of a game (the existence of winning strategies) rather than an *algorithmic* property of a game (the existence of a decision procedure to solve the game). For instance, a class of games may well be undecidable and determined at the same time, e.g., several bisimulation games over arbitrary infinite-state systems. The interested reader is referred to [38] for further information on properties of games and their relationships with various computational issues, especially complexity and expressivity ones.

Chapter 3

Mu-Calculi for True Concurrency

In this chapter we start by studying the underlying mathematical properties of a number of partial order models of concurrency based on transition systems, Petri nets, and event structures, and show that the partial order behaviour represented by these models of (true) concurrency can be captured in a uniform way by two simple and general dualities of local behaviour. We then give logical characterisations to these dualities and find that natural fixpoint modal logics with partial order semantics can be extracted from such characterisations.

The naturality of these modal logics is supported by the logical equivalences they induce, which, in a number of cases, coincide with some of the most important bisimulation equivalences for both interleaving and true concurrency. Such coincidence results suggest a logical approach to defining a notion of equivalence for concurrency tailored to be abstract or model independent. The approach put forward here sets the grounds for a logic-based framework for studying different kinds of models of concurrency uniformly. The main results of this chapter were first presented in [39]; an extended and revised version appears in [40].

3.1 Local Dualities in Partial Order Models

In this section we present two ways in which concurrency can be regarded as a dual concept to 'conflict' and 'causality', respectively. These two ways of observing concurrency will be called 'immediate concurrency' and 'linearised concurrency'. Whereas immediate concurrency is dual to conflict, linearised concurrency is dual to causality.

The intuitions behind these two observations are the following. Consider a concurrent system and any two different transitions t and t' with the same source node, i.e., $\sigma(t) = \sigma(t')$. These two transitions are either immediately concurrent, and therefore independent, i.e., $(t,t') \in I$, or dependent, in which case they must be in conflict. Similarly, consider any two transitions t and t' where $\tau(t) = \sigma(t')$. Again, the pair of transitions (t,t') can either belong to I, in which case the two transitions are concurrent, yet have been linearised, or the pair does not belong to I,

and therefore the two transitions are causally dependent. Notice that, in both cases, the two conditions are mutually exclusive and, more importantly, that there are no other possibilities.

These dualities of concurrent behaviour make sense only in a local setting. If two arbitrary transitions t and t' do not have the property that $\sigma(t) = \sigma(t')$ or $\tau(t) = \sigma(t)$ (or vice versa), then nothing can be said about them doing only this analysis. Nevertheless, this simple notion of behavioural observation introduced here is rather powerful and the basic ingredient for defining several fixpoint modal logics with partial order semantics.

The local dualities just described are formally defined in the following way:

$$\begin{aligned}
\otimes &\stackrel{\text{def}}{=} \{(t,t') \in T \times T \mid \sigma(t) = \sigma(t') \wedge t \; I \; t'\} \\
\# &\stackrel{\text{def}}{=} \{(t,t') \in T \times T \mid \sigma(t) = \sigma(t') \wedge \neg(t \; I \; t')\} \\
\ominus &\stackrel{\text{def}}{=} \{(t,t') \in T \times T \mid \tau(t) = \sigma(t') \wedge t \; I \; t'\} \\
\leq &\stackrel{\text{def}}{=} \{(t,t') \in T \times T \mid \tau(t) = \sigma(t') \wedge \neg(t \; I \; t')\}
\end{aligned}$$

Notice the dual conditions between $\otimes$ and # and between $\ominus$ and $\leq$ with respect to the independence relation, if assuming valid the locality requirement.

Definition 3.1. **(Local dualities)** Let t and t' be two transitions. We say that t and t' are *immediately concurrent* iff $(t,t') \in \otimes$, *in conflict* iff $(t,t') \in \#$, *linearly concurrent* iff $(t,t') \in \ominus$, or *causally dependent* iff $(t,t') \in \leq$. ◁

Sets in a Local Context. The relation $\otimes$ on pairs of transitions can be used to define *sets* where every transition is independent of each other and hence can all be executed concurrently. These sets of transitions are called conflict-free; moreover, the transitions in such sets are said to belong to the same (Mazurkiewicz) 'trace' [62].

Definition 3.2. **(Conflict-free sets)** A *conflict-free* set of transitions P is a set of transitions with the same source node, where $t \otimes t'$ for all t,t' in P. ◁

Notice that by definition empty sets and singleton sets are trivially conflict-free. Given a system $\mathfrak{T}$, all conflict-free sets of transitions at a state s can be defined locally from the 'maximal set' of transitions $\mathfrak{X}(s)$, where $\mathfrak{X}(s)$ is the set of all transitions t such that $\sigma(t) = s$. We simply write $\mathfrak{X}$ when the state s is defined elsewhere or is implicit from the context. Moreover, all *maximal sets* and *conflict-free sets* of transitions are fixed given a particular system $\mathfrak{T}$.

Definition 3.3. **(Support sets)** Given a system $\mathfrak{T}$, a *support* set R in $\mathfrak{T}$ is either a maximal set of transitions or a non-empty conflict-free set of transitions in $\mathfrak{T}$. ◁

Given a system $\mathfrak{T}$, the set of all its support sets is denoted by $\mathfrak{P}$. As can be seen from the definition, support sets can be of two kinds, and one of them provides a way of doing local reasoning. More precisely, local reasoning on sets of independent transitions becomes possible when considering conflict-free sets since they can be decomposed into smaller ones, where every transition is, again, independent of each other.

Definition 3.4. (Complete supsets) Given a support set R, a *complete supset* M of R, denoted by $M \sqsubseteq R$, is a support set $M \subseteq R$ such that $\neg\exists t \in R \setminus M.\ \forall t' \in M.\ t \otimes t'$. ◁

Note that if R is a conflict-free support set, then we have that $M = R$. Otherwise, R is necessarily a maximal set of transitions $\mathfrak{X}(s)$, for some state s, and M must be a proper subset of R. Therefore, if $R = \mathfrak{X}(s)$ then the sets of transitions M such that $M \sqsubseteq \mathfrak{X}(s)$ are the biggest conflict-free support sets that can be recognised in a particular state s of a system $\mathfrak{T}$; we call them 'maximal supsets'. Since both complete and maximal supsets are all support sets, then they are also fixed and effectively computable given a particular finite-state system $\mathfrak{T}$.

3.2 Fixpoint Logics with Partial Order Semantics

The local dualities and sets defined in the previous section can be used to provide partial order semantics of various fixpoint modal logics that capture the concurrent behaviour of partial order models that is not present in interleaving ones. Due to this, such logics are more adequate specification languages for expressing properties of so-called 'true concurrency' models such as those studied in this thesis, namely Petri nets, event structures, and TSI models.

The naturality of these (fixpoint) modal logics is reflected by the logical equivalences they induce, since in various cases they either coincide with well-known bisimilarities for concurrency, e.g., with Milner and Park's strong bisimilarity or with hp bisimilarity, or have better decidability properties than other bisimulation equivalences for true concurrency, e.g., with respect to hhp bisimilarity, which is undecidable even on finite-state systems.

The partial order semantics of the fixpoint modal logics defined here are based on the recognition of what is actually observable, locally, in a partial order model of concurrency. In other words, properties of system executions that are conflict-free. Two main fixpoint logics are introduced, Separation Fixpoint Logic (SFL) and Trace Fixpoint Logic ($\mathbb{L}_\mu$). As defined by their semantics, they capture the duality between concurrency and causality in the same way, by means of a refinement of the usual modal operator of the mu-calculus.

However, the duality between concurrency and conflict is captured in different ways in each logic. In the case of SFL we use a separating operator that behaves as a structural conjunction. This *structural operator* allows one to do local reasoning on conflict-free support sets. On the other hand, in $\mathbb{L}_\mu$ the duality between concurrency and conflict is captured by a second-order modality that recognises maximal supsets in the system. Such a modality enjoys beautiful mathematical properties; in particular, not only it is a monotonic, but also an *idempotent operator*, a property of closure operators which we formalize next.

Definition 3.5. (Closure operators) A function $f : A \to A$ is a *closure operator* on a poset $(A, \leq_A)$ iff it is extensive, monotonic, and idempotent, this is, if f satisfies that for all $a, a' \in A$: $a \leq_A f(a)$; $a \leq_A a'$ implies $f(a) \leq_A f(a')$; and $f(a) = f(f(a))$ ◁

Process Spaces. In order to define the denotational semantics of SFL and $\mathbb{L}_\mu$ we construct an intermediate mathematical structure into which any of the systems we consider here can be mapped. Such a structure determines a 'space of processes', which are simple abstract entities that represent pieces of isolated (i.e., local and independent) behaviour.

Definition 3.6. (Process spaces) Let $\mathfrak{T} = (S, s_0, T, I, \Sigma)$ be a system. A *process space* $\mathbb{S}$ is a subset of $S \times \mathfrak{P} \times \mathfrak{A}$, such that S is the set of states of $\mathfrak{T}$, $\mathfrak{P}$ is the set of support sets of $\mathfrak{T}$, and $\mathfrak{A}$ is the set of transitions $T \cup \{t_\varepsilon\}$, where t_ε is the empty transition such that for all $t \in T$, if $s_0 = \sigma(t)$ then $t_\varepsilon \leq t$. A tuple $(s, R, t) \in \mathbb{S}$ is a *process*; the initial process is $(s_0, \mathfrak{X}(s_0), t_\varepsilon)$. ◁

The empty transition is introduced to formalize the fact that at the beginning of any computation no transitions have been performed. Then, t_ε represents the inactivity of the system before anything happens. On the other hand, regarding process spaces, it is important to note that one does not need to actually consider the whole lattice $S \times \mathfrak{P} \times \mathfrak{A}$ in practice, since support sets are defined with respect to a particular state. Therefore, if one knows the support set component of a process, then it is possible to infer the particular state in $\mathfrak{T}$. If $\mathbb{S}$ is presented as $\mathfrak{P} \times \mathfrak{A}$, then such a process space is called 'stateless'. Also, let $\mathcal{X}$ be the subset of $\mathfrak{P}$ that contains only maximal sets and supsets. Call $\mathfrak{S} = \mathcal{X} \times \mathfrak{A}$ a 'stateless maximal' process space.

Although the denotation of both SFL formulae and $\mathbb{L}_\mu$ formulae can be given using a standard process space, we will present the denotation of $\mathbb{L}_\mu$ formulae using a stateless maximal process space. We do it this way with the purpose of showing how the results presented in the following sections can be obtained using a simplified structure.

Notation 3.7. We use the name 'process space' for both $\mathbb{S} = S \times \mathfrak{P} \times \mathfrak{A}$ and $\mathfrak{S} = \mathcal{X} \times \mathfrak{A}$. The particular kind of mathematical structure we are referring to will always be clear from the context. Similarly, we use the word 'process' for elements both in $\mathbb{S}$ and in $\mathfrak{S}$. ◁

Separation Fixpoint Logic

Definition 3.8. *Separation Fixpoint Logic* (SFL) has formulae ϕ built from a set Var of variables $Y, Z, \ldots$ and a set Σ of labels $a, b, \ldots$ by the following grammar:

$$\phi ::= Z \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \langle a \rangle_c \phi_1 \mid \langle a \rangle_{nc} \phi_1 \mid \phi_1 * \phi_2 \mid \mu Z.\phi_1$$

where $Z \in \text{Var}$ and $\mu Z.\phi_1$ has the restriction that any free occurrence of Z in ϕ_1 must be within the scope of an even number of negations. Dual operators are defined in the usual way:

$$\begin{array}{lcl}
\phi_1 \vee \phi_2 & \stackrel{\text{def}}{=} & \neg(\neg\phi_1 \wedge \neg\phi_2) \\
[a]_c \phi_1 & \stackrel{\text{def}}{=} & \neg\langle a \rangle_c \neg\phi_1 \\
[a]_{nc} \phi_1 & \stackrel{\text{def}}{=} & \neg\langle a \rangle_{nc} \neg\phi_1 \\
\phi_1 \bowtie \phi_2 & \stackrel{\text{def}}{=} & \neg(\neg\phi_1 * \neg\phi_2) \\
\nu Z.\phi_1 & \stackrel{\text{def}}{=} & \neg\mu Z.\neg\phi_1[\neg Z/Z]
\end{array}$$

Moreover, 'plain modalities', i.e., HML modalities, can be represented as follows:

$$\begin{array}{lcl} \langle a\rangle\phi_1 & \stackrel{\text{def}}{=} & \langle a\rangle_c\phi_1 \vee \langle a\rangle_{nc}\phi_1 \\ [a]\,\phi_1 & \stackrel{\text{def}}{=} & [a]_c\,\phi_1 \wedge [a]_{nc}\,\phi_1 \end{array}$$

Boolean constants and other abbreviations are defined as for $\mathcal{L}_\mu$. ◁

Notation 3.9. (Positive forms) We say that a logical formula is in 'positive form' if negations are applied only to variables. Any formula built with the language given in Definition 3.8, together with the dual operators, can be converted into positive form; it is moreover in 'positive normal form' (PNF) if there are no clashes of bound variables. Again, any logical formula can be converted into an equivalent one in positive normal form. Then, without loss of generality, hereafter we only consider formulae in positive normal form. ◁

Definition 3.10. An *SFL model* $\mathfrak{M}$ is a system $\mathfrak{T} = (S, s_0, T, I, \Sigma)$ together with a valuation $\mathcal{V} : \mathrm{Var} \to 2^{\mathbb{S}}$, where $\mathbb{S} = S \times \mathfrak{P} \times \mathfrak{A}$ is the process space associated with $\mathfrak{T}$. The denotation $\|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$ of an SFL formula ϕ in the model $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$ is a subset of $\mathbb{S}$, given by the following rules (omitting the superscript $\mathfrak{T}$):

$$\begin{array}{lcl} \|Z\|_{\mathcal{V}} & = & \mathcal{V}(Z) \\ \|\neg\phi_1\|_{\mathcal{V}} & = & \mathbb{S} - \|\phi_1\|_{\mathcal{V}} \\ \|\phi_1 \wedge \phi_2\|_{\mathcal{V}} & = & \|\phi_1\|_{\mathcal{V}} \cap \|\phi_2\|_{\mathcal{V}} \\ \|\langle a\rangle_c\phi_1\|_{\mathcal{V}} & = & \{(s,R,t) \in \mathbb{S} \mid \exists s' \in S.\ \exists t' \in R. \\ & & t' = s \xrightarrow{a} s' \wedge t \leq t' \wedge (s', \mathfrak{X}(s'), t') \in \|\phi_1\|_{\mathcal{V}}\} \\ \|\langle a\rangle_{nc}\phi_1\|_{\mathcal{V}} & = & \{(s,R,t) \in \mathbb{S} \mid \exists s' \in S.\ \exists t' \in R. \\ & & t' = s \xrightarrow{a} s' \wedge t \ominus t' \wedge (s', \mathfrak{X}(s'), t') \in \|\phi_1\|_{\mathcal{V}}\} \\ \|\phi_1 * \phi_2\|_{\mathcal{V}} & = & \{(s,R,t) \in \mathbb{S} \mid \exists R_1, R_2 \in \mathfrak{P}. \\ & & R_1 \uplus R_2 \sqsubseteq R \wedge (s,R_1,t) \in \|\phi_1\|_{\mathcal{V}} \wedge (s,R_2,t) \in \|\phi_2\|_{\mathcal{V}}\} \end{array}$$

Given the usual restriction on free occurrences of variables, imposed to obtain monotone operators in $\wp(\mathbb{S}) = 2^{\mathbb{S}}$, the powerset lattice of $\mathbb{S}$, it is possible to define the denotation of the fixpoint operator $\mu Z.\phi_1$ in the standard way, according to the Knaster–Tarski fixpoint theorem:

$$\|\mu Z.\phi_1\|_{\mathcal{V}} \quad = \quad \bigcap\{Q \in \wp(\mathbb{S}) \mid \|\phi_1\|_{\mathcal{V}[Z:=Q]} \subseteq Q\}$$

where $\mathcal{V}[Z := Q]$ is the valuation $\mathcal{V}'$ which agrees with $\mathcal{V}$ save that $\mathcal{V}'(Z) = Q$. Since PNF is assumed, the semantics of the dual operators is defined as usual. Finally, let $\models^{\mathfrak{T}}_{\mathcal{V}}$ denote the *satisfaction* relation, i.e., for any process $P \in \mathbb{S}$, we have that $P \models^{\mathfrak{T}}_{\mathcal{V}} \phi$ iff $P \in \|\phi\|_{\mathcal{V}}$. ◁

Informally, the meaning of the basic SFL operators is the following: boolean constants and operators are interpreted as usual; the semantics of the 'causal' diamond modality $\langle a\rangle_c\phi_1$ (resp. of the 'non-causal' diamond modality $\langle a\rangle_{nc}\phi_1$) is that a process (s,R,t) satisfies $\langle a\rangle_c\phi_1$ (resp. $\langle a\rangle_{nc}\phi_1$) if it can perform an a-labelled action t' that causally depends on t (resp. that is

independent of t) and move through t' into a process where ϕ_1 holds; and dually for the causal and non-causal box modalities $[a]_c\phi_1$ and $[a]_{nc}\phi_1$. Moreover, the structural operator $\phi_1 * \phi_2$ specifies that there exists a partition in the support set, i.e., a partition of the transitions in the set to be considered, w.r.t. which both formulae ϕ_1 and ϕ_2 can hold independently. This does not necessarily mean that both formulae hold in parallel everywhere because the operator $*$ has a local meaning. Finally, the fixpoint operators are interpreted in the same way that for $\mathcal{L}_\mu$.

SFL can express several properties of true concurrency systems that cannot be specified using $\mathcal{L}_\mu$. Some examples are given in a forthcoming section. Before that, let us introduce $\mathbb{L}_\mu$, a logic where the duality between concurrency and conflict is captured using a higher-order modality on conflict-free sets. Although the main results in this and the next chapter apply for SFL and for $\mathbb{L}_\mu$, the proofs in the latter case are slightly simpler and, therefore, preferred.

Trace Fixpoint Logic

Definition 3.11. *Trace Fixpoint Logic* ($\mathbb{L}_\mu$) has formulae ϕ built from a set Var of variables $Y, Z, ...$ and a set Σ of labels $a, b, ...$ by the following grammar:

$$\phi ::= Z \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \langle a\rangle_c\phi_1 \mid \langle a\rangle_{nc}\phi_1 \mid \langle\otimes\rangle\phi_1 \mid \mu Z.\phi_1$$

where $Z \in \text{Var}$ and $\mu Z.\phi_1$ has the restriction that any free occurrence of Z in ϕ_1 must be within the scope of an even number of negations. Dual boolean, modal, and fixpoint operators as well as boolean constants and other abbreviations are defined as for SFL; in particular, let $[\otimes]\phi_1$ be the dual of $\langle\otimes\rangle\phi_1$ which is defined as usual for modal operators, i.e., $[\otimes]\phi_1 \stackrel{\text{def}}{=} \neg\langle\otimes\rangle\neg\phi_1$. ◁

Let us now define the semantics of $\mathbb{L}_\mu$. Again, since PNF is assumed, the semantics of the dual boolean, modal, and fixpoint operators are given in the usual way.

Definition 3.12. A $\mathbb{L}_\mu$ *model* $\mathfrak{M}$ is a system $\mathfrak{T} = (S, s_0, T, I, \Sigma)$ together with a valuation $\mathcal{V}$: $\text{Var} \to 2^{\mathfrak{S}}$, where $\mathfrak{S} = \mathcal{X} \times \mathfrak{A}$ is the process space associated with $\mathfrak{T}$. The denotation $\|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$ of a formula ϕ in the model $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$ is a subset of $\mathfrak{S}$, given by the following rules (omitting the superscript $\mathfrak{T}$):

$$\begin{array}{lcl}
\|Z\|_{\mathcal{V}} & = & \mathcal{V}(Z) \\
\|\neg\phi_1\|_{\mathcal{V}} & = & \mathfrak{S} - \|\phi_1\|_{\mathcal{V}} \\
\|\phi_1 \wedge \phi_2\|_{\mathcal{V}} & = & \|\phi_1\|_{\mathcal{V}} \cap \|\phi_2\|_{\mathcal{V}} \\
\|\langle a\rangle_c\phi_1\|_{\mathcal{V}} & = & \{(R,t) \in \mathfrak{S} \mid \exists r \in R.\ \delta(r) = a \wedge t \leq r \wedge (\mathfrak{X}(\tau(r)), r) \in \|\phi_1\|_{\mathcal{V}}\} \\
\|\langle a\rangle_{nc}\phi_1\|_{\mathcal{V}} & = & \{(R,t) \in \mathfrak{S} \mid \exists r \in R.\ \delta(r) = a \wedge t \ominus r \wedge (\mathfrak{X}(\tau(r)), r) \in \|\phi_1\|_{\mathcal{V}}\} \\
\|\langle\otimes\rangle\phi_1\|_{\mathcal{V}} & = & \{(R,t) \in \mathfrak{S} \mid \exists M \in \mathcal{X}.\ M \sqsubseteq R \wedge (M,t) \in \|\phi_1\|_{\mathcal{V}}\} \\
\|\mu Z.\phi_1\|_{\mathcal{V}} & = & \bigcap\{Q \subseteq \mathfrak{S} \mid \|\phi_1\|_{\mathcal{V}[Z:=Q]} \subseteq Q\}
\end{array}$$

As before, $\mathcal{V}[Z := Q]$ is the valuation $\mathcal{V}'$ which agrees with $\mathcal{V}$ save that $\mathcal{V}'(Z) = Q$. Finally, the *satisfaction* relation $\models^{\mathfrak{T}}_{\mathcal{V}}$ is defined as before: for any $P \in \mathfrak{S}$, $P \models^{\mathfrak{T}}_{\mathcal{V}} \phi$ iff $P \in \|\phi\|_{\mathcal{V}}$. ◁

The informal meanings of the $\mathbb{L}_\mu$ operators that are also part of SFL is as in the SFL case. In fact, one can show that their semantics are equivalent. Therefore, the only difference between SFL and $\mathbb{L}_\mu$ is given by the structural operator $*$ of SFL and the modality $\langle\otimes\rangle$ of $\mathbb{L}_\mu$ (and their duals). The modalities $\langle\otimes\rangle$ and $[\otimes]$ also provide second-order power on conflict-free sets of transitions, i.e., on concurrent executions of systems, but they do so in a way that is different from how $*$ and $\bowtie$ do it. The higher-order modal operators of $\mathbb{L}_\mu$ allow one to restrict, locally, the executions of a system to those ones that can actually happen concurrently at a given state. It is easy to show that $\langle\otimes\rangle$ is monotonic; however, more interestingly, it is also idempotent.

Proposition 3.13. *$\langle\otimes\rangle$ is an idempotent operator.*

Proof. Let $H = \|\langle\otimes\rangle\phi\|$ and $G = \|\phi\|$. The set G can be split into two disjoint sets of stateless maximal processes $G^\otimes \uplus G^\#$ (called simply processes in the sequel), where the former is the set of processes in G whose support sets are conflict-free, and the latter those processes whose support sets are not, i.e., $G \setminus G^\otimes$. Similarly, the set H can be represented as the disjoint union of sets of processes $H^\otimes$ and $H^\#$. Notice that $H^\otimes = G^\otimes$ because for any process $P_{H^\otimes} = (R,t)$ in $H^\otimes$ there is a process $P_{G^\otimes} = (R,t)$ in $G^\otimes$, as $R \sqsubseteq R$ for any conflict-free support set R.

However, this equality does not necessarily hold for processes in $G^\#$ and $H^\#$, since there may be a process $P_{G^\#}$ in $G^\#$ (whose support set is necessarily maximal and not conflict-free) such that there is no process $P_{G^\otimes}$ in $G^\otimes$ to which the support set of $P_{G^\#}$ can be related using $\sqsubseteq$.

Therefore, whereas the set $G^\#$ would contain such a process, the set $H^\#$ would not. Similarly, there may be new processes in $H^\#$ whose support sets can be related to support sets of processes in $G^\otimes$ (and of course in $H^\otimes$ as well) using $\sqsubseteq$, but that were not in $G^\#$. Now, let $F = F^\otimes \uplus F^\# = \|\langle\otimes\rangle\langle\otimes\rangle\phi\|$. For the same reason given before, $F^\otimes = H^\otimes$. However, in this case $F^\# = H^\#$ since now for every process in both $F^\#$ and $H^\#$, there must be a process in $H^\otimes$ (and of course in $F^\otimes$) to which their support sets can be related using $\sqsubseteq$. Since applying $\langle\otimes\rangle$ only once always leads to a fixpoint, one can conclude that $\langle\otimes\rangle$ is an idempotent operator. □

Fact 3.14. *$\langle\otimes\rangle$ is not an extensive operator.*

Proof. Let $H = H^\otimes \uplus H^\# = \|\langle\otimes\rangle\phi\|$ and $G = G^\otimes \uplus G^\# = \|\phi\|$, where $H^\otimes$ and $H^\#$ as well as $G^\otimes$ and $G^\#$ are defined as before. As shown in the proof of Proposition 3.13, it is possible that $G^\#$ contains processes that are not in $H^\#$. Then, $G \not\subseteq H$ in general. □

Corollary 3.15. *$\langle\otimes\rangle$ is not a closure operator.*

Proof. $\langle\otimes\rangle$ is monotonic and idempotent, but is not extensive. □

Example 3.16. This example shows that $\langle\otimes\rangle$ is not a closure operator as it is not extensive. Let ϕ be $\langle a\rangle_c\langle b\rangle_c$tt and $\psi = \langle\otimes\rangle\phi$; moreover, let $P = Q + R$, $Q = a.b$, and $R = a.b$ be three CCS processes whose behaviour is represented by the systems $\mathfrak{M}_P$, $\mathfrak{M}_Q$, and $\mathfrak{M}_R$, respectively. Although the three processes satisfy ϕ, only $\mathfrak{M}_Q$ and $\mathfrak{M}_R$ satisfy ψ. Thus, $\|\phi\| \not\subseteq \|\psi\|$. ◁

3.3 Examples and Applications

SFL vs. $\mathbb{L}_\mu$. The first example is aimed at uncovering the subtle difference between SFL and $\mathbb{L}_\mu$, which is related to the difference between $*$ and $\langle\otimes\rangle$. Roughly, it has to do with the fact that, in SFL, the operator $*$ considers all conflict-free support sets rather than only maximal ones (as it is the case for $\langle\otimes\rangle$ in $\mathbb{L}_\mu$). As a consequence, in SFL one can differentiate systems with different patters in the independence relation by looking only at the sizes of the support sets of their associated process spaces, without relying on the labelling of the transitions in such support sets. The following example makes this claim concrete.

Example 3.17. Consider the following three sequential processes $A_1 = a.A_1$, $A_2 = a.A_2$, and $A_3 = a.A_3$, which execute an action with an a-label and return to their previous local states. Notice that the two concurrent systems $A_1 \parallel A_2$ and $A_1 \parallel A_2 \parallel A_3$ can be differentiated by $\langle a\rangle\mathrm{tt} * \langle a\rangle\mathrm{tt} * \langle a\rangle\mathrm{tt}$ but cannot be distinguished by any $\mathbb{L}_\mu$ formula. The reason, as described above, is that SFL can recognize that the process $A_1 \parallel A_2 \parallel A_3$ has a support set of size 3, whereas $A_1 \parallel A_2$ has support sets of size 2 only. From the viewpoint of $\mathbb{L}_\mu$, the difference between the two systems is hidden by the fact that all actions are equally labelled. ◁

Temporal Logics. SFL and $\mathbb{L}_\mu$ can express all usual temporal properties, such as, liveness, safety, and so on. These properties are equally handled in interleaving and partial order models.

Example 3.18. (Causal reachability) Let ϕ be the following reachability logical formula: $\phi = [h]_c \mu Z.(\langle a\rangle_c \mathrm{tt} * \langle b\rangle_c \mathrm{tt}) \vee \langle -\rangle_c Z$. This SFL formula expresses that after executing any initial action h (if any) there exists at least one execution of causally dependent actions such that *eventually* two actions a and b can be executed in parallel (since they must be independent). This temporal specification is better than a similar one given by, e.g., the modal mu-calculus, since in the SFL case unnecessary interleavings are not checked (those not depending causally on h) and hence a combinatorial explosion of the state space to be searched is reduced. ◁

Example 3.19. (Safety of critical regions) Let $\phi = \nu Z.\,[\otimes]\,([wrA]\,\mathrm{ff} \vee [wrB]\,\mathrm{ff}) \wedge [-]\,Z$. This $\mathbb{L}_\mu$ formula says that *always* it is impossible for a system to execute in parallel two actions *wrA* and *wrB*. If, for instance, 'wrA' and 'wrB' refer to actions—of two parallel processes A and B—that modify (write) a particular critical region, then one can be sure that the access to such a critical section is safe if the temporal specification ϕ is satisfied by the system. ◁

Example 3.20. (Secure synchronisation) When using a process algebra like CCS, one would like to specify the property that whenever some action, say a, is executed there exists a parallel action $\bar{a}$ that can also be executed in order for them to synchronise. Then, one can be sure that the temporal property that whenever some component of a system performs a always another component is ready to respond with $\bar{a}$ is satisfied iff $\phi = \nu Z.[a]_c\,\langle\bar{a}\rangle_{nc}\mathrm{tt} \wedge [-]\,Z$ is satisfied. ◁

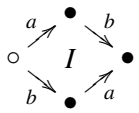
Example 3.21. (Response properties) Another interesting property, which requires the combination of least and greatest fixpoints, is a response to a request of a service. Suppose that whenever a system component executes an a-labelled action, there must exist a sequential system component that eventually executes a b-labelled action, which is meant to be the allocation of the service being requested through a. Such a property is expressed in both SFL and $\mathbb{L}_\mu$ with the following fixpoint formula: $\phi = \nu Z.\,[a]\,(\mu Y.\langle b\rangle_c \mathrm{tt} \wedge \langle -\rangle_c Y) \wedge [-]\,Z$. If, moreover, one wants to specify that b is necessarily reachable in all causal lines, i.e., that it is unavoidable, then ϕ can be modified as follows: $\phi = \nu Z.\,[a]\,(\mu Y.\langle b\rangle_c \mathrm{tt} \wedge [-]_c Y \wedge \langle -\rangle_c \mathrm{tt}) \wedge [-]\,Z$. ◁

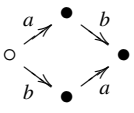
In similar ways, many more temporal (true concurrency) properties can be specified with SFL and $\mathbb{L}_\mu$, especially by allowing a free alternation of least and greatest fixpoint operators.

Model-Independent Logics. SFL and $\mathbb{L}_\mu$ can be used to compare the partial order behaviour of concurrent systems represented with different sorts of models, e.g., Petri nets, TSI, or event structures. This can be done in this unified framework as the semantics of these two fixpoint modal logics are given using a process space, which abstracts away from the particular features of each of the concrete partial order models of concurrency we consider here.

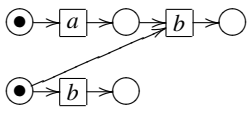
In this way, we say that two concurrent systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$ are behaviourally equivalent with respect to the bisimulation equivalence induced by SFL (resp. $\mathbb{L}_\mu$) if, and only if, the process spaces associated with $\mathfrak{T}_1$ and $\mathfrak{T}_1$ satisfy the same set of SFL (resp. $\mathbb{L}_\mu$) formulae. This notion of partial order behavioural (or observational) equivalence is made precise in the next section. For now, let us use, in the following example, the intuitive definition just given.

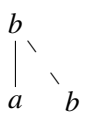
Example 3.22. In Figure 3.1, whereas the two systems at the top are not SFL equivalent, the two systems at the bottom are $\mathbb{L}_\mu$ equivalent. This can be concluded even though $\mathfrak{T}_1$ is a TSI, $\mathfrak{T}_2$ is an LTS, $\mathfrak{T}_3$ is a Petri net, and $\mathfrak{T}_4$ is an event structure. ◁





A TSI $\mathfrak{T}_1$ (on the left) and an LTS $\mathfrak{T}_2$ (on the right)



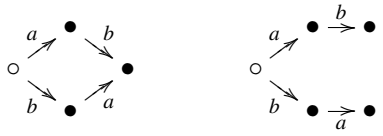


A Petri net $\mathfrak{T}_3$ (on the left) and an event structure $\mathfrak{T}_4$ (on the right)

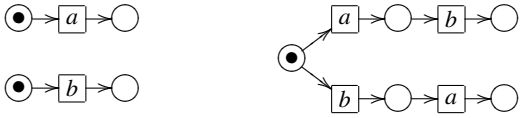
Figure 3.1: Comparing different models of concurrency uniformly using SFL and $\mathbb{L}_\mu$

Logics for True Concurrency. SFL and $\mathbb{L}_\mu$ can differentiate concurrency from nondeterminism using two different local dualities. Consider the following concurrent systems (in CCS notation and with a partial order semantics, e.g., using Petri nets [19] or event structures [99]): $P = a \parallel b$ and $Q = a.b + b.a$; models of P and Q are depicted in Figure 3.2. Processes P and Q are equivalent in an interleaving context (e.g., $P \sim_{sb} Q$), but different from a partial order viewpoint as they are not equated by any equivalence for true concurrency (e.g., $P \not\sim_{hpb} Q$).

Such a difference can be captured with both SFL and $\mathbb{L}_\mu$ in several ways: for instance, using the duality between concurrency and conflict with the SFL formula $\phi = \langle a\rangle \mathrm{tt} * \langle b\rangle \mathrm{tt}$ and the $\mathbb{L}_\mu$ formula $\phi' = \langle\otimes\rangle(\langle a\rangle \mathrm{tt} \wedge \langle b\rangle \mathrm{tt})$, or using the duality between concurrency and causality with the SFL and $\mathbb{L}_\mu$ formula $\psi = \langle a\rangle_c \langle b\rangle_{nc} \mathrm{tt}$, which are all true in P but not in Q.



Interleaving representation of processes P (on the left) and Q (on the right).



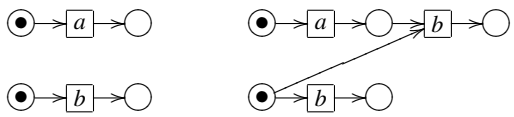
Partial order representation of processes P (on the left) and Q (on the right).

Figure 3.2: Interleaving vs. partial order representations of $P = a \parallel b$ and $Q = a.b + b.a$.

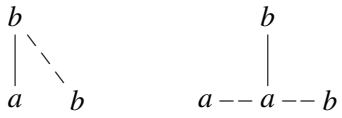
Now, within the true concurrency spectrum there are several bisimilarities (cf. [26, 35]); for instance, step bisimilarity, pomset bisimilarity, or hp bisimilarity, which was already mentioned in Chapter 2. As shown in the following section, the equivalences induced by SFL and $\mathbb{L}_\mu$ are strictly stronger than any of these three bisimilarities in some classes of systems.

Example 3.23. Consider the systems in Figure 3.3. The two Petri nets are told apart by the formula $\langle a\rangle_c(\langle b\rangle_c \mathrm{tt} \wedge \langle b\rangle_{nc} \mathrm{tt})$, the two event structures by the formula $\langle a\rangle_c(\langle b\rangle_c \mathrm{tt} \wedge [-]_{nc} \mathrm{ff})$, and the two TSI models by the $\mathbb{L}_\mu$ formula $\langle\otimes\rangle(\langle a\rangle_c\langle c\rangle_c \mathrm{tt} \wedge \langle b\rangle_c\langle d\rangle_c \mathrm{tt})$ and the SFL formula $\langle a\rangle_c\langle c\rangle_c \mathrm{tt} * \langle b\rangle_c\langle d\rangle_c \mathrm{tt}$, which, in each case, hold only in the system on the right. ◁

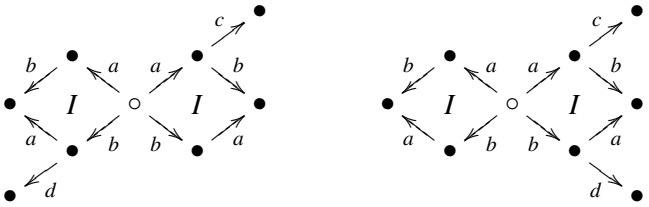
Remark 3.24. The previous example shows the strong distinguishing power of SFL and $\mathbb{L}_\mu$. As shown later, the equivalences they induce are the strongest decidable bisimilarities over the classes of partial order models we consider in the following section and with respect to the best known (bisimulation) equivalences in the literature (see [26]), which make them interesting logics for true concurrency. These logics take full account of the interplay between causality and branching, and recognise subtle differences between partial order behaviours, differences that are hidden behind complex nondeterministic choices of independent local processes. ◁



Two step bisimilar systems that are not pomset bisimilar.



Two pomset bisimilar systems that are not hp bisimilar.



Two hp bisimilar systems that are not hhp bisimilar.

Figure 3.3: True concurrency systems with different partial order behaviour.

Logics for Multi-Agent Systems (MAS). MAS, such as those analysed with logics like ATL [5] and extensions therefrom, can be studied using SFL and $\mathbb{L}_\mu$. In order to model MAS, an explicit notion of an agent must be defined. Since our basic partial order model is based on transitions representing instances of actions in the system, such transitions should belong to uniquely defined agents. Therefore, the following set of agents and corresponding "ownership" mapping on transitions are defined. Let Γ be a finite set of sequential agents and $\mathcal{A} : \mathfrak{A} \to \Gamma$ be a mapping that assigns transitions to agents. In this way, it is possible to know the transitions that an agent can execute. However, since transitions in a partial order model represent instances of actions in a system, rather than actions that an agent can execute, a consistency restriction must be defined so that all transitions that are instances of the same action are performed by the same agent. This is captured by adding the constraint: if $t_1 \sim t_2$ then $\mathcal{A}(t_1) = \mathcal{A}(t_2)$.[1]

Also, since modal logics can make distinctions between transitions with different labels, a consistency relation on labels of transitions should also be defined: if $\mathcal{A}(t_1) \neq \mathcal{A}(t_2)$ then $\delta(t_1) \neq \delta(t_2)$. Imposing these restrictions is equivalent to defining a distributed alphabet Σ over a set of independent agents Γ. A partial order model extended with these definitions and restrictions is called 'consistent' for MAS. Furthermore, a modality $\langle a \rangle^\alpha \phi$ is called 'well-defined' for MAS iff for all transitions $t \in \mathfrak{A}$ if $a = \sigma(t)$ then $\alpha = \mathcal{A}(t)$. Modalities of the form $\langle K \rangle^\alpha \phi$, where $K \subseteq \Sigma$, can be defined by $\langle K \rangle^\alpha \phi = \bigvee_{a \in K} \langle a \rangle^\alpha \phi$ using the standard notation for modalities with sets of labels. We write $\langle - \rangle^\alpha \phi$ to mean the diamond modality on the set of labels restricted to those of α. Assume similar definitions for the other modalities.

[1] Recall that $\sim$ is the equivalence relation on system or TSI transitions as defined in Chapter 2.

Example 3.25. The formula $\psi = [-]^{\beta} \langle - \rangle^{\alpha}_{nc} \mu Z.\phi \vee \langle - \rangle^{\alpha}_{c} Z$ says that there is an agent α (the system) that can satisfy ϕ regardless the behaviour of an adversarial agent β (the environment).[2] Informally, ψ says "whatever you (the environment) do, I (the system) can get to ϕ, though I may have to do some things that do not depend on what you previously did." $\triangleleft$

3.4 Logical and Concurrent Equivalences

We now turn our attention to the formal study of the relationships between the explicit notion of independence in partial order models of concurrency (which we call *model* independence), and the explicit notion of independence in the modal logics we have defined in the previous section (which we call *logical* independence). We do so by relating well-known bisimulation equivalences for interleaving and true concurrency, namely $\sim_{sb}$, $\sim_{hpb}$, and $\sim_{hhpb}$, with the logical equivalences induced by different SFL and $\mathbb{L}_{\mu}$ sublogics where the interplay between concurrency and conflict, and concurrency and causality is *syntactically* restricted.

Definition 3.26. **($\mathfrak{L}$ equivalence $\sim_{\mathfrak{L}}$)** Given a logic $\mathfrak{L}$, two processes P and Q associated with two systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$, respectively, are $\mathfrak{L}$-equivalent, $P \sim_{\mathfrak{L}} Q$, iff for every $\mathfrak{L}$ formula ϕ in $\mathfrak{F}_{\mathfrak{L}}$, $P \models \phi \Leftrightarrow Q \models \phi$, where $\mathfrak{F}_{\mathfrak{L}}$ is the set of all fixpoint-free closed formulae of $\mathfrak{L}$. $\triangleleft$

Remark 3.27. The previous definition delivers a logical, abstract notion of equivalence that can be used across different models of concurrency, i.e., tailored to be model independent. With this logical notion of (bisimulation) equivalence two systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$, possibly of different kinds, are equivalent with respect to some equivalence relation $\sim_{\mathfrak{L}}$ if, and only if, their associated process spaces cannot be differentiated by any $\mathfrak{L}$-logical formula. $\triangleleft$

Recall that in order to obtain an exact match between finitary modal logic and bisimulation, all models considered here are image-finite [46], i.e., of finite branching. Moreover, since the semantics of SFL and $\mathbb{L}_{\mu}$ are based on action labels, we only consider models or systems without 'auto-concurrency' [72], a common restriction when studying either modal logics or bisimulation equivalences for (labelled) partial order models of concurrency.

More precisely, auto-concurrency is the phenomenon by which multiple instances of various concurrent or transitions are equally labelled. In other words, auto-concurrency can be seen as nondeterminism inside a set of independent transitions. In many cases auto-concurrency is regarded as an undesirable situation on partial order models since, firstly, can be easily avoided in practice and, secondly, makes slightly counter-intuitive the analysis of behavioural properties of concurrent processes with partial order semantics.

As a matter of fact, on finite systems, auto-concurrency is formally, but not actually, a further restriction since any bounded branching system that has auto-concurrency can be effectively converted into a system that does not have auto-concurrency by a suitable relabelling

[2] I thank Julian Bradfield for the IFML [10] version of this example.

of auto-concurrent transitions without changing the concurrent behaviour of the model. Notice that no auto-concurrency is a real further restriction for infinite systems as image-finiteness does not imply branching boundedness on infinite models.

Having said that, let us turn to the study of some syntactic fragments of SFL and $\mathbb{L}_\mu$. They are called the *natural* syntactic fragments of SFL and $\mathbb{L}_\mu$ because such sub-logics arise as the languages where the dualities between concurrency and causality as well as concurrency and conflict are syntactically manipulated. As we will see the logical equivalences induced by all such syntactic fragments are decidable and, in some cases, coincide with well-known bisimilarities for interleaving and for partial order models of concurrency.

3.4.1 The Causal-Free Cases

We start this study of logical and concurrent equivalences by analysing the syntactic fragments of SFL and $\mathbb{L}_\mu$ that are oblivious to any causal information in the systems.

The Modal Mu-Calculus. The first sublogic we consider is obtained from both SFL and $\mathbb{L}_\mu$ by disabling the sensitivity of these logics to both dualities. On the one hand, insensitivity to the duality between concurrency and causality can be captured by considering only modalities without subscript, using the abbreviations for modalities given previously in Definition 3.8, which also applies to $\mathbb{L}_\mu$. So, only the following modalities are considered (the ones of HML):

$$\begin{aligned} \langle a\rangle\phi_1 &= \langle a\rangle_c\phi_1 \vee \langle a\rangle_{nc}\phi_1 \\ [a]\,\phi_1 &= [a]_c\,\phi_1 \wedge [a]_{nc}\,\phi_1 \end{aligned}$$

On the other hand, insensitivity to the duality between concurrency and conflict can be captured in SFL (resp. $\mathbb{L}_\mu$) by considering the $*$-free SFL sublanguage (resp. the $\langle\otimes\rangle$-free $\mathbb{L}_\mu$ sublanguage).[3] The resulting logic has the same syntax as the modal mu-calculus. This natural syntactic fragment is the purely-modal $*$-free (resp. $\langle\otimes\rangle$-free) fragment of SFL (resp. of $\mathbb{L}_\mu$).

Proposition 3.28. *The syntactic purely-modal $*$-free (resp. $\langle\otimes\rangle$-free) fragment of SFL (resp. of $\mathbb{L}_\mu$) is semantically equivalent to the modal mu-calculus.*

Proof. Recall the denotational semantics of the logical operators of SFL and $\mathbb{L}_\mu$. Without loss of generality, we can decide to consider only the case of the modal operators, which are the same in both logics. Also, since every support set defines a unique state, then the semantics of $\mathbb{L}_\mu$ can be extended with a state component in order to match the structure of the SFL semantics, and so deliver a single proof. Therefore, the following proof, as well as all other proofs not involving either $*$ or $\langle\otimes\rangle$ (and their duals) apply to both SFL and $\mathbb{L}_\mu$.

[3] By the $*$-free (resp. $\langle\otimes\rangle$-free) sublanguage we mean the sublogic without that operator and its dual.

$$\begin{aligned}\|\langle a\rangle\phi_1\|_{\mathcal{V}} &= \|\langle a\rangle_c\phi_1 \vee \langle a\rangle_{nc}\phi_1\|_{\mathcal{V}} = \|\langle a\rangle_c\phi_1\|_{\mathcal{V}} \cup \|\langle a\rangle_{nc}\phi_1\|_{\mathcal{V}} \\ &= \{(s,R,t) \in \mathbb{S} \mid \exists s' \in S.\ \exists t' = s \xrightarrow{a} s' \in R.\ t \leq t' \wedge (s', \mathfrak{X}(s'), t') \in \|\phi_1\|_{\mathcal{V}}\} \cup \\ &\quad \{(s,R,t) \in \mathbb{S} \mid \exists s' \in S.\ \exists t' = s \xrightarrow{a} s' \in R.\ t \ominus t' \wedge (s', \mathfrak{X}(s'), t') \in \|\phi_1\|_{\mathcal{V}}\}\end{aligned}$$

The first observation to be made is that the $\{*, \langle\otimes\rangle\}$-free fragment of these logics only considers *maximal sets* in the semantics. Therefore if a transition can be performed at a state s then it is always in the support set at s. Hence, support sets can be disregarded and only the state of every support set must be kept. Then, we get the following simplified expression:

$$\begin{aligned}\|\langle a\rangle\phi_1\|_{\mathcal{V}} &= \{(s,t) \in S \times \mathfrak{A} \mid \exists s' \in S.\ t \leq s \xrightarrow{a} s' \wedge (s', s \xrightarrow{a} s') \in \|\phi_1\|_{\mathcal{V}}\} \cup \\ &\quad \{(s,t) \in S \times \mathfrak{A} \mid \exists s' \in S.\ t \ominus s \xrightarrow{a} s' \wedge (s', s \xrightarrow{a} s') \in \|\phi_1\|_{\mathcal{V}}\}\end{aligned}$$

The second observation is that when computing the semantics of the combined operator $\langle a\rangle$, the conditions $t \leq s \xrightarrow{a} s'$ and $t \ominus s \xrightarrow{a} s'$ complement each other and become trivially true (since there are no other possibilities). Therefore, the second component of every pair $(s,t) \in S \times \mathfrak{A}$ can also be disregarded, and the denotation of the diamond modality can be written as follows:

$$\|\langle a\rangle\phi_1\|_{\mathcal{V}} = \{s \in S \mid \exists s' \in S.\ s \xrightarrow{a} s' \wedge s' \in \|\phi_1\|_{\mathcal{V}}\}$$

The case for the box operator $[a]$ is similar. As a consequence, the semantics of all the operators of this sublogic and the modal mu-calculus coincide. □

Remark 3.29. $\mathcal{L}_\mu$ cannot recognise pairs of transitions in I and therefore sees any partial order model as its interleaving counterpart, or what is equivalent, a partial order model with an empty relation I. As a consequence, although using a partial order model, it is possible to retain in these logics all the joys of a logic with an interleaving model, and so, nothing is lost in terms of expressivity with respect to the main interleaving approaches to concurrency. ◁

Regarding logical and concurrent equivalences, which is the main concern of this section, it is now easy to see that Milner and Park's strong bisimilarity, $\sim_{sb}$, the equivalence induced by modal logic (and therefore HML), is captured by the fixpoint-free fragment of this sublogic, which we denote by $\sim_{\mathcal{L}_\mu}$. Hence, the *correspondence* $\sim_{\mathcal{L}_\mu} \equiv \sim_{sb}$ follows from Proposition 3.28 and the fact that modal logic characterises bisimulation on image-finite models.

The Separation Modal Mu-Calculus. The second sublogic we study is the 'separation modal mu-calculus', $\mathcal{L}^*_\mu$. This logic is obtained from SFL by allowing only the recognition of the duality between concurrency and conflict by using its structural operator $*$. The syntax of $\mathcal{L}^*_\mu$ is as follows: $\phi ::= Z \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \langle a\rangle\phi_1 \mid \phi_1 * \phi_2 \mid \mu Z.\phi_1$.

We write $\sim_{\mathcal{L}^*_\mu}$ for the equivalence induced by this SFL sublogic. It is easy to see that $\mathcal{L}^*_\mu$ is more expressive than $\mathcal{L}_\mu$ in partial order models simply because $\mathcal{L}^*_\mu$ includes $\mathcal{L}_\mu$ and can differentiate concurrency from nondeterminism. However, there is a counter-example that shows that $\sim_{\mathcal{L}^*_\mu}$ and $\sim_{hpb}$, in general, do not coincide.

Proposition 3.30. *Neither* $\sim_{hpb} \subseteq \sim_{\mathcal{L}^*_\mu}$ *nor* $\sim_{\mathcal{L}^*_\mu} \subseteq \sim_{hpb}$.

Proof. In Figure 3.3, the two TSI models at the bottom are hp bisimilar and yet can be distinguished by the $\mathcal{L}^*_\mu$ formula $\phi = \langle a\rangle\langle c\rangle \mathrm{tt} * \langle b\rangle\langle d\rangle \mathrm{tt}$. On the other hand, the two Petri net models at the top are not pomset bisimilar (and hence are not hp bisimilar either) and cannot be differentiated by any $\mathcal{L}^*_\mu$ formula. Because of the sizes of the systems, this can be verified by exhaustively checking all semantically different $\mathcal{L}^*_\mu$ formulae of modal depth no greater than 2 (of which there are finitely many) in all the states of these systems. □

The Trace Modal Mu-Calculus. The third sublogic is the 'trace modal mu-calculus', $\mathcal{L}^\otimes_\mu$. This logic is obtained from $\mathbb{L}_\mu$ by allowing, similar to the $\mathcal{L}^*_\mu$ case, only the recognition of the duality between concurrency and conflict. The syntax of $\mathcal{L}^\otimes_\mu$ is given by the following rules: $\phi ::= Z \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \langle a\rangle\phi_1 \mid \langle\otimes\rangle\phi_1 \mid \mu Z.\phi_1$. We write $\sim_{\mathcal{L}^\otimes_\mu}$ for the equivalence induced by this $\mathbb{L}_\mu$ sublogic. As in the $\mathcal{L}^*_\mu$ case, $\mathcal{L}^\otimes_\mu$ is more expressive than $\mathcal{L}_\mu$ in partial order models and the equivalence it induces does not coincide with $\sim_{hpb}$ either.

Proposition 3.31. *Neither* $\sim_{hpb} \subseteq \sim_{\mathcal{L}^\otimes_\mu}$ *nor* $\sim_{\mathcal{L}^\otimes_\mu} \subseteq \sim_{hpb}$.

Proof. Use the same arguments as in the proof of Proposition 3.30 and the $\mathcal{L}^\otimes_\mu$ formula $\phi = \langle\otimes\rangle(\langle a\rangle\langle c\rangle \mathrm{tt} \wedge \langle b\rangle\langle d\rangle \mathrm{tt})$ instead. □

Causal-free Mu-Calculi on Confusion-free Systems. There is a fundamental reason for the mismatch between $\sim_{hpb}$ and $\sim_{\mathcal{L}^*_\mu}$ and between $\sim_{hpb}$ and $\sim_{\mathcal{L}^\otimes_\mu}$. It has to do with a special "sharing" of resources between some of the transitions in the model. This special kind of sharing of resources is characterised by a phenomenon of true concurrency systems called 'confusion', which is a concept in net theory, and thus, it is useful (and actually much easier) to think of it directly on net models. For this reason we will present it using Petri nets.

Confusion can be of two different kinds: symmetric or asymmetric. Roughly speaking, confusion is a phenomenon that arises between at least three different actions, say between t_1, t_2, and t_3. In the symmetric case, two of them are independent, e.g., t_1 par t_2, and at the same time are in conflict with the third action, i.e., ${}^\bullet t_1 \cap {}^\bullet t_3 \neq \emptyset$ and ${}^\bullet t_2 \cap {}^\bullet t_3 \neq \emptyset$. On the other hand, in the asymmetric case, two of the actions are independent, e.g., t_1 par t_2 as before, whereas the third one causally depends on one of the independent actions, say on t_1, and is in conflict with the other, i.e., one has that $t_1^\bullet \cap {}^\bullet t_3 \neq \emptyset$ and ${}^\bullet t_2 \cap {}^\bullet t_3 \neq \emptyset$, respectively.

Confusion is important because, although it is undesirable when analysing the behaviour of a concurrent system, it is also "inherent to any reasonable net model of a mutual exclusion module" [87]. Confusion is also present when modelling race conditions in concurrent and distributed systems with shared memory models. These facts show the ubiquity of this phenomenon when analysing real-life models of communicating concurrent systems. Although

confusion is a natural concept in net theory, it can also be defined for TSI and event structures, though in the TSI case the definition is slightly more complicated because it involves sets of transitions rather than single actions or events as in the Petri net and event structure cases.

Confusion appears in the systems used in the proofs of Propositions 3.30 and 3.31, in both cases in its asymmetric variant. The problem is that $\sim_{hpb}$, $\sim_{\mathcal{L}_\mu^*}$, and $\sim_{\mathcal{L}_\mu^\otimes}$ can recognise some forms of confusion, but not all of them. However, there are classes of systems where confusion never arises, and for which coincidence results between these bisimilarities may be possible.

Definition 3.32. (Confusion-free systems) A system $\mathfrak{T}$ is *confusion-free* if, and only if, for all different transitions t_1, t_2, and t_3 such that $t_1 \# t_2$ and $t_1 \otimes t_3$, there exist transitions r_1 and r_2, where $t_1 \sim r_1$ and $t_2 \sim r_2$, such that $r_1 \# r_2$ and $t_3 \ominus r_1$ and $t_3 \ominus r_2$. ◁

Informally, the previous definition means that a choice, i.e., a conflict in terms of Petri nets or event structures, cannot be globally affected by the execution of a concurrent transition, since equivalent choices are always possible both before and after that[4]. These facts, along with the observations made before, led us to believe that the following statement holds for systems without auto-concurrency, although we have so far not been able to prove it.

Conjecture 3.33. $\sim_{\mathcal{L}_\mu^*} \equiv \sim_{hpb} \equiv \sim_{\mathcal{L}_\mu^\otimes}$ *on confusion-free systems without auto-concurrency.*

Now, let us move to the study of a modal logic that is sensitive to the causal information embodied in partial order systems. In particular, it will be shown that for some classes of (true concurrency) systems the *local* duality between concurrency and causality is good enough to capture the full notion of *global* causality defined by $\sim_{hpb}$ on partial order models.

3.4.2 From Local to Global Causality

In this section we show the first coincidence result of the equivalence induced by one of the sublogics of both SFL and $\mathbb{L}_\mu$ with a bisimilarity for partial order systems. The result holds for a class of systems whose expressive power lies between that of so-called 'free-choice' nets [20] and that of safe nets (cf. Chapter 2), as before with the usual restrictions to systems that are image-finite and have no auto-concurrency.

The coincidence result is with respect to $\sim_{hpb}$. This equivalence is considered to be the standard bisimulation equivalence for causality since it fully captures the interplay between branching and causal behaviour. The interesting feature of this coincidence result is that $\sim_{hpb}$ provides a *global* notion of causality whereas the modal logic we are about to study induces a *local* one, as shown later on. Then, the question we answer here is that of the class of true concurrency systems for which 'local causality' fully captures the standard, and mathematically more complex, notion of 'global causality'. Such an answer is given by the following logic.

[4] Another way to define confusion-free systems, which we use later on, is to define a 'confusion' relation and confusion-free systems as those for which such a relation is empty.

The Causal Modal Mu-Calculus. The fourth sublogic to be considered is the 'causal modal mu-calculus', $\mathcal{L}_\mu^c$. This sublogic is obtained from SFL and $\mathbb{L}_\mu$ by allowing only the recognition of the duality between concurrency and causality throughout the modal operators on transitions of these logics. The syntax of this syntactic fragment is given by the following language: $\phi ::= Z \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \langle a \rangle_c \phi_1 \mid \langle a \rangle_{nc} \phi_1 \mid \mu Z.\phi_1$.

Clearly, $\mathcal{L}_\mu^c$ is also more expressive than $\mathcal{L}_\mu$ because of the same reasons given for $\mathcal{L}_\mu^*$ and $\mathcal{L}_\mu^\otimes$. The naturality of $\mathcal{L}_\mu^c$ for expressing causal properties is demonstrated by the equivalence it induces, written as $\sim_{\mathcal{L}_\mu^c}$, which coincides with $\sim_{hpb}$, the standard bisimilarity for causal systems, when restricted to systems without auto-concurrency where any 3-tuple of transitions (t_1,t_2,t_3) in confusion is in some sense deterministic. Thus, let us define confusion, a ternary relation on transitions as well as its deterministic variant when considering labelled systems.

Definition 3.34. (Confusion) Let cfs be a ternary relation on transitions of a system $\mathfrak{T}$ such that $(t_1,t_2,t_3) \in \mathsf{cfs}$ iff $t_1 \otimes t_2$ and either $t_1 \# t_3$ and $t_2 \# t_3$ (the symmetric case) or $t_1 \leq t_3$ and $\exists r_2.\ t_2 \sim r_2 \wedge r_2 \# t_3$ (the asymmetric case). A tuple $(t_1,t_2,t_3) \in \mathsf{cfs}$ is deterministic iff either the three transitions have different labels or $\delta(t_1) = \delta(t_3)$ and $t_1 \leq t_3$. ◁

There are analogous Petri net and event structure definitions for confusion using the basic elements of such models. Those definitions are better known than the one presented here since confusion is a basic concept in net theory; however, the definition we have given is equivalent. Perhaps due to this is that an easy way of depicting confusion is using nets. Figure 3.4 shows the two simplest nets featuring confusion, both in their symmetric and asymmetric variants.

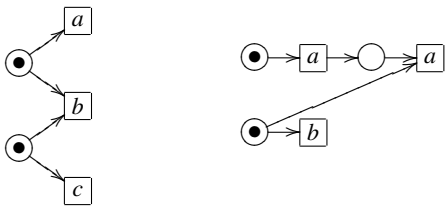


Figure 3.4: Confusion: the Petri net on the left has symmetric confusion and the Petri net on the right has asymmetric confusion. In both cases it is deterministic.

Any Petri net that has confusion must have either of these two nets as a subsystem. The statement equivalently holds for TSI and event structures if considering, respectively, the TSI and event structure models corresponding to such nets. This property allows one to define a class of concurrent systems for which the logical equivalence induced by $\mathcal{L}_\mu^c$ captures $\sim_{hpb}$. Such a class contains all those concurrent systems without auto-concurrency that either are *free-choice* or whose confusion relation has only deterministic tuples. Thus, let us now define the class of free-choice systems. For simplicity, we do so indirectly via the standard definition of free-choice nets, which is well-known in the (Petri net) literature.

Definition 3.35. (Free-choice nets) Let $\mathcal{N}$ be a net. A net is *free-choice* if, and only if, for all $s \in P$ we have that $\mid s^\bullet \mid \leq 1$ or $\forall t \in s^\bullet . \mid {}^\bullet t \mid = 1$. ◁

We extend the previous definition on nets to systems in the following way: a free-choice Petri net is a free-choice net with an initial marking; moreover, a free-choice event structure is an event structure unfolding [71] of a free-choice Petri net and a free-choice TSI is the TSI model obtained from a free-choice Petri net (cf. see the mappings presented in Chapter 2).

Free-choice nets, and hence free-choice systems, have no confusion as other classes of concurrent systems. But what is more important to note about this class of nets is that the confusion-freeness property (which is a behavioural characteristic) comes directly from a simple structural property of the nets in such a class. In particular, any free-choice net, and therefore free-choice system, can be built using the subnets shown in Figure 3.5 (with additional flow arrows after net actions, which can be used unrestrictedly as long as the net stays safe).

Figure 3.5: Free-choice nets: the two subnets from which any free-choice system can be built.

We are almost ready to show that the two bisimulation equivalences $\sim_{\mathcal{L}^c_\mu}$ and $\sim_{hpb}$ coincide for a big family of concurrent systems that we call 'fc-structured', and denote by Ξ.

Definition 3.36. (Fc-structured (Ξ) systems) The family of *fc-structured* (Ξ) systems is the class of systems without auto-concurrency that either are free-choice or whose confusion relation has only deterministic elements. ◁

Remark 3.37. The family of Ξ-systems contains, at least, the following classes of models (without auto-concurrency and with a deterministic confusion relation): Moore and Mealy machines, labelled transition graphs, synchronous and asynchronous products of sequential systems, free-choice systems, and compositions of nondeterministic concurrent systems. As shown at the end of this section, this is a big family of models which can represent several complex systems with interesting concurrent behaviour from a practical viewpoint. ◁

Now, back to the issue of relating $\sim_{hpb}$ and $\sim_{\mathcal{L}^c_\mu}$, the proof that $\sim_{hpb}$ and $\sim_{\mathcal{L}^c_\mu}$ coincide for the class of Ξ-systems goes by showing that the two inclusions $\sim_{hpb} \subseteq \sim_{\mathcal{L}^c_\mu}$ and $\sim_{\mathcal{L}^c_\mu} \subseteq \sim_{hpb}$ hold separately. In fact, the first inclusion holds for any class of systems (a result that shows that $\sim_{hpb}$ is stronger than $\sim_{\mathcal{L}^c_\mu}$) while the second one requires the restriction to the class of Ξ-systems introduced above. In a number of cases, the proof uses, or is driven by, key insights that come directly from the properties of the net representation of Ξ-systems.

In addition, in order to deliver a single proof covering the process space models of both SFL and $\mathbb{L}_\mu$, let us use the model that considers processes in $\mathfrak{S}$ rather than in $\mathbb{S}$. The reduction actually goes in both ways, i.e., from $\mathbb{S}$ to $\mathfrak{S}$ as well as from $\mathfrak{S}$ to $\mathbb{S}$, and can always be used so long as the operators $*$ and $\langle\otimes\rangle$ are not considered since only maximal sets will be needed. Notice that any $(\mathfrak{X}(\tau(t)),t) \in \mathfrak{S}$ induces a unique $(\tau(t),\mathfrak{X}(\tau(t)),t) \in \mathbb{S}$, and vice versa.

Lemma 3.38. (Logical soundness) $\sim_{hpb} \subseteq \sim_{\mathcal{L}^c_\mu}$.

Proof. This inclusion can be shown by induction on $\mathcal{L}^c_\mu$ formulae, which we denote by $\mathfrak{F}_{\mathcal{L}^c_\mu}$. Let $\mathfrak{T}_1$ and $\mathfrak{T}_0$ be two systems and $P \in \mathfrak{S}_1$ and $Q \in \mathfrak{S}_0$ two processes that belong to the process spaces $\mathfrak{S}_1$ and $\mathfrak{S}_0$ associated with $\mathfrak{T}_1$ and $\mathfrak{T}_0$, respectively. If $P \sim_{hpb} Q$ then for all $\phi \in \mathfrak{F}_{\mathcal{L}^c_\mu}$ we have that $P \models^{\mathfrak{T}_1}_{\mathcal{V}_1} \phi \Leftrightarrow Q \models^{\mathfrak{T}_0}_{\mathcal{V}_0} \phi$ given two models $\mathfrak{M}_1 = (\mathfrak{T}_1, \mathcal{V}_1)$ and $\mathfrak{M}_0 = (\mathfrak{T}_0, \mathcal{V}_0)$. Since $\mathcal{L}^c_\mu$ only considers maximal sets, the process $P = (p,t)$ (resp. the process $Q = (q,t)$) is actually a tuple in $S_1 \times \mathfrak{A}_1$ (resp. in $S_0 \times \mathfrak{A}_0$) rather than a tuple in $\mathcal{X}_1 \times \mathfrak{A}_1$ (resp. in $\mathcal{X}_0 \times \mathfrak{A}_0$). Henceforth, let us write $\models$ instead of $\models^{\mathfrak{T}_i}_{\mathcal{V}_i}$, for $i \in \{0,1\}$, since the models will be clear from the context.

The base case of the induction is when $\phi = \mathrm{tt}$ or when $\phi = \mathrm{ff}$ which is trivial since the logical formulae tt and ff are always true and false, respectively. Now, consider the cases for the boolean operators $\wedge$ and $\vee$; first suppose that ϕ is the conjunction $\psi_1 \wedge \psi_2$ and assume that the result holds for both formulae ψ_1 and ψ_2. By the definition of the satisfaction relation $P \models \phi$ iff $P \models \psi_1$ and $P \models \psi_2$ iff by the inductive hypothesis $Q \models \psi_1$ and $Q \models \psi_2$, and hence, by the definition of the satisfaction relation $Q \models \phi$. The case for the boolean operator $\vee$ is similar.

Now, consider the cases for the four modalities. First, suppose $\phi = [a]_{nc}\psi$ and $P \models \phi$. Therefore, for any $P' = (p',t')$, such that $a = \delta(t')$ and $P \xrightarrow{a} P'$ and $t \ominus t'$, it follows that $P' \models \psi$. Now, let $Q \xrightarrow{a} Q'$ such that $a = \delta(t')$ and $t \ominus t'$ since the bisimulation must remain synchronous. Just to recall, synchrony in an hp bisimulation means that the last transition chosen in $\mathfrak{T}_1$ (resp. in $\mathfrak{T}_0$) is concurrent with the former transition also chosen in $\mathfrak{T}_1$ (resp. in $\mathfrak{T}_0$) iff the same pattern holds in the last two transitions chosen in $\mathfrak{T}_0$ (resp. in $\mathfrak{T}_1$), and moreover the two sequences of transitions (i.e., runs) that are generated in this way are the linearisations of isomorphic posets. So, as we know that for some P' there is a $P \xrightarrow{a} P'$, where $t \ominus t'$, and by the inductive hypothesis $P' \sim_{hpb} Q'$, then $Q' \models \psi$, where $t \ominus t'$, and so by the definition of the satisfaction relation $Q \models \phi$. The case when Q satisfies ϕ is symmetric, and the case when $\phi = [a]_c\psi$ is similar (only changing $\ominus$ for $\leq$). The cases for $\langle a\rangle_c$ and $\langle a\rangle_{nc}$ are analogous. □

In order to show the second inclusion ($\sim_{\mathcal{L}^c_\mu} \subseteq \sim_{hpb}$) we first require some lemmas that characterise the set of runs that can be identified by $\mathcal{L}^c_\mu$ in a partial order system. More specifically, a proof that if two systems $\mathfrak{T}_0$ and $\mathfrak{T}_1$ are $\mathcal{L}^c_\mu$-equivalent, then for each run of one of the systems there exists a 'locally synchronous' run (which is defined below) in the other system. Then, one can use this result to show that for any two Ξ-systems $\mathfrak{T}_0$ and $\mathfrak{T}_1$ such that $\mathfrak{T}_0 \sim_{\mathcal{L}^c_\mu} \mathfrak{T}_1$, each pair of locally synchronous runs is moreover induced by two isomorphic posets, and hence, the

two systems must be $\sim_{hpb}$ too, as in such a case the pair of runs is synchronous.

Recall the definition of runs and of synchronous runs from Chapter 2, and let $\pi_0 \in \Pi_{\mathfrak{T}_0}$ and $\pi_1 \in \Pi_{\mathfrak{T}_1}$ be two runs of two systems $\mathfrak{T}_0$ and $\mathfrak{T}_1$, and u, v two transitions. A pair of runs $(\pi_0.u, \pi_1.v)$ is inductively defined as 'locally synchronous' iff (π_0, π_1) is locally synchronous and $(\rho(\pi_0), u) \in I_0 \Leftrightarrow (\rho(\pi_1), v) \in I_1$, where I_0 and I_1 are the independence relations of $\mathfrak{T}_0$ and $\mathfrak{T}_1$. By definition, $(\varepsilon, \varepsilon)$ is locally synchronous. Note that the definitions of locally synchronous runs and synchronous runs are quite similar; the only difference is that synchronous runs must be the linearization of isomorphic posets whereas locally synchronous runs need not be.

Lemma 3.39. *Let $\mathfrak{T}_0$ and $\mathfrak{T}_1$ be two systems and $\Pi_{\mathfrak{T}_0}$ and $\Pi_{\mathfrak{T}_1}$ their sets of runs. If $\mathfrak{T}_0 \sim_{\mathcal{L}_\mu^c} \mathfrak{T}_1$ then for each $\pi_0 \in \Pi_{\mathfrak{T}_0}$ (resp. $\pi_1 \in \Pi_{\mathfrak{T}_1}$) there exists a run $\pi_1 \in \Pi_{\mathfrak{T}_1}$ (resp. $\pi_0 \in \Pi_{\mathfrak{T}_0}$) such that the pair of runs (π_0, π_1) is locally synchronous.*

Proof. The proof goes by a contradiction argument. Suppose that for all ϕ in $\mathfrak{F}_{\mathcal{L}_\mu^c}$ we have that $P \models \phi \Leftrightarrow Q \models \phi$ and there exists a run in one of the systems that is not locally synchronous to any of the runs in the other system. The case where P and Q are the initial processes of $\mathfrak{T}_0$ and $\mathfrak{T}_1$, respectively, is trivially false since, by definition, the pair $(\varepsilon, \varepsilon)$ is locally synchronous.

Then, suppose now that (π_0, π_1) is locally synchronous and that P and Q are two processes reached, respectively, in $\mathfrak{T}_0$ and in $\mathfrak{T}_1$ after following π_0 and π_1 in each system (starting from their initial processes). Additionally, suppose that there exists a transition u in one of the systems, say in $\mathfrak{T}_0$, such that there is no transition v in the other system for which the pair of runs $(\pi_0.u, \pi_1.v)$ is locally synchronous. Note that P and Q are strongly bisimilar, since $\mathcal{L}_\mu^c$ includes $\mathcal{L}_\mu$, and thus, the case in which a processes can perform a transition (regardless of its label) and the other cannot do so is impossible as this contradicts the hypothesis that $P \sim_{sb} Q$.

So, suppose that for some transition u with label a, $P = (p, \rho(\pi_0)) \xrightarrow{a} P' = (p', u)$ and $\rho(\pi_0) \ominus u$ (resp. $\rho(\pi_0) \leq u$), but for all transitions v such that $a = \delta(v)$ it holds that $Q = (q, \rho(\pi_1)) \xrightarrow{a} Q' = (q', v)$ and $\rho(\pi_1) \leq v$ (resp. $\rho(\pi_1) \ominus v$) only. However, we know that, by hypothesis, $P \sim_{\mathcal{L}_\mu^c} Q$ and so, it must be true that if $P \models \langle a \rangle_{nc} \phi$ (resp. if $P \models \langle a \rangle_c \phi$) then $Q \models \langle a \rangle_{nc} \phi$ (resp. if $Q \models \langle a \rangle_c \phi$), which is a contradiction. Thus, one must be able to match pairs of independent transitions in one of the systems whenever the same happens in the other system for all pairs of processes P and Q satisfying that $P \sim_{\mathcal{L}_\mu^c} Q$. □

Lemma 3.39 says that if two systems satisfy the same set of $\mathcal{L}_\mu^c$ formulae, then, locally, they have the same causal behaviour. However, to show that globally they also have the same causal behaviour, one needs some additional information, which is given by the following lemma.

Lemma 3.40. *Let $\mathfrak{T}$ be a Ξ-system whose conflict relation is* cfs *and let $\pi \in \Pi_{\mathfrak{T}}$. If after executing the run π in $\mathfrak{T}$ there are two different enabled transitions u and v such that $\delta(u) = \delta(v)$, then the following two statements hold:*

1. $u \# v$.
2. *There is at most one transition* t *in* π *such that* $\tau(t) = \sigma(u') = \sigma(v')$ *for which* $t \leq u'$ *and* $t \leq v'$ *and* $u \sim u'$ *and* $v \sim v'$.

Proof. In the same order as in the statement of the lemma:

1. Because there is no auto-concurrency.
2. Since the confusion relation is deterministic there is no $c \in \mathsf{cfs}$ such that both u and v belong to c; in particular neither transition can be an instance of an action e (at the net level) for which $|\, {}^\bullet e \,| > 1$. Instead, such transitions must be instances of two different actions e_1 and e_2 for which $|\, {}^\bullet e_1 \,| = 1 = |\, {}^\bullet e_2 \,|$. □

Finally, the following lemma ensures that for the class of Ξ-systems, the notion of locally synchronous runs (associated with local causality) is good enough—strong enough or sufficient—to capture the stronger, and more complex, notion of synchronous runs (associated with global causality), provided that the two systems satisfy the same set of $\mathcal{L}_\mu^c$ formulae.

Lemma 3.41. *Let* $\mathfrak{T}_0$ *and* $\mathfrak{T}_1$ *be* Ξ*-systems whose sets of runs are* $\Pi_{\mathfrak{T}_0}$ *and* $\Pi_{\mathfrak{T}_1}$*. If for each* $\pi_0 \in \Pi_{\mathfrak{T}_0}$ *(resp.* $\pi_1 \in \Pi_{\mathfrak{T}_1}$*) there exists some* $\pi_1 \in \Pi_{\mathfrak{T}_1}$ *(resp.* $\pi_0 \in \Pi_{\mathfrak{T}_0}$*) such that* (π_0, π_1) *is locally synchronous, then* (π_0, π_1) *is, moreover, synchronous.*

Proof. The proof is based on the fact that if (π_0, π_1) is a locally synchronous pair, then the posets induced by such locally synchronous runs induce isomorphic posets if the systems are fc-structured, and hence, the pair of runs is also globally synchronous. We proceed by induction on the length of runs. The base case, i.e., when the pair of runs is the pair of empty runs $(\pi_0, \pi_1) = (\varepsilon, \varepsilon)$, is trivial since in this case the two posets are empty.

For the induction step, suppose that there is a non-empty run π_0 of size k that is locally synchronous to some run π_1; moreover, suppose that π_0 and π_1 induce isomorphic posets. We show that there is not a run $\pi_0.u$ which induces a poset that is not isomorphic to any of the posets induced by those runs $\pi_1.v$ for which the pairs of extended runs $(\pi_0.u, \pi_1.v)$ are locally synchronous. Then, we have to analyse the way in which u and v extend the runs π_0 and π_1.

Due to the definition of Ξ-systems, one can consider the following three cases: (1) the transition u is the instance of a net action e such that $|\, {}^\bullet e \,| > 1$ and u is not in the conflict relation of $\mathfrak{T}_0$; (2) the transition u is the instance of a net action e such that for some net place s we have that $e \in s^\bullet$ and $\forall e \in s^\bullet . |\, {}^\bullet e \,| = 1$ and u is not in the conflict relation of $\mathfrak{T}_0$; or (3) the transition u is an instance of a net action of either type and is in the conflict relation of $\mathfrak{T}_0$.

For the first case, let π_0 be any run such that $\rho(\pi_0) \leq u$. By hypothesis we have that the posets induced by π_0 and π_1 are isomorphic, that u depends only on one transition (namely, on $\rho(\pi_0)$), and that $\rho(\pi_1) \leq v$ as well. Then, the only possibility for this case to fail is if v, unlike u, causally depends on more than only one transition (since it already depends on

$\rho(\pi_1)$). Suppose this could happen; then, there is at least one transition e_j in π_1 on which v also causally depends and that is independent of $\rho(\pi_1)$. Then there must exist a run π_1^- of length $k-1$ that do not contain e_j and where $\rho(\pi_1^-) \leq v'$ for some v' such that $\delta(v') = \delta(v)$. Since v and v' cannot be two instances of the same net action, then they must be in conflict (because there is no auto-concurrency) and moreover belong to some tuple c of the confusion relation cfs of $\mathfrak{T}_1$, which is impossible since Ξ-systems have a deterministic confusion relation.

As a consequence any transition u of this kind can be matched only by a transition v that is the instance of a net action e for which $|\,{}^\bullet e\,| = 1$, and due to Lemma 3.40, such kind of transitions extend a unique transition of any run, keeping the two extended runs $\pi_0.u$ and $\pi_1.v$ not only locally synchronous but also globally synchronous.

For the second case, suppose that u depends on a set of elements $\{e_0^i, ..., e_0^k, ..., e_0^m\}$ of the poset induced by π_0, i.e., $\forall e \in \{e_0^i, ..., e_0^k, ..., e_0^m\}.(e,u) \notin I_0$, and there is at least one e_0^k that was related to some e_1^k of π_1 while constructing the two locally synchronous runs, i.e., $e_0^k = \pi_0(k)$ and $e_1^k = \pi_1(k)$ for some natural number k, but that is not extended in π_0 with respect to u as e_1^k is extended in π_1 with respect to v, i.e., which makes the two induced posets not isomorphic because $(e_0^k, u) \notin I_0$ whereas $(e_1^k, v) \in I_1$.

For the same reasons given in the first case, v cannot depend on only one transition in π_1. On the contrary it must depend on at least two transitions, one of which must have the same label as e_0^k and e_1^k; let e_1^n be such a transition. As in the first case, w.l.o.g. the other transition can be $\rho(\pi_1)$. Then, we have that v causally depends on e_1^n and is independent of e_1^k, which is independent of e_1^n. But this is impossible since $\delta(e_1^n) = \delta(e_1^k)$ and there is no auto-concurrency. Therefore, both runs must be extended in a synchronous way in this case as well.

Finally, for the third case notice that the arguments given before apply here as well, regardless of the kind of transition under consideration since the two properties in the former cases still hold: on the one hand, any two transitions equally labelled are always in conflict and causally depend (locally) on only one transition of any run; and, on the other hand, whenever is enabled a transition that is an instance of a net action whose preset is not a singleton, then that transition is the only one enabled with such a label.

Then, v must extend the poset induced by π_1 as u extends the poset induced by π_0, i.e., for all k in $\{1, ..., |\,\pi_0\,|\}$ one has that $(\pi_0(k), u) \notin I_0$ iff $(\pi_1(k), v) \notin I_1$, making the two posets isomorphic in all cases and for all pairs (π_0, π_1) of locally synchronous runs of any length.[5] □

Informally, one can say that the arguments in the proof just given go through because any "extra-concurrency" in one of the systems with respect to the other can be recognised since there is no auto-concurrency, and any "extra-causality" can be recognised since, for models in the class of Ξ-systems, any two transitions that are enabled at the same time and are equally labelled must be in conflict and causally depend on one transition in any run.

[5] I thank Sibylle Fröschle for discussions and comments on this result and its associated proof.

Corollary 3.42. (Logical completeness) $\sim_{\mathcal{L}^c_\mu} \subseteq \sim_{hpb}$ *on* Ξ*-systems.*

Proof. From Lemmas 3.39 and 3.41. □

Theorem 3.43. (Full logical definability) $\sim_{\mathcal{L}^c_\mu} \equiv \sim_{hpb}$ *on* Ξ*-systems.*

Proof. Immediate from Lemma 3.38 and Corollary 3.42. □

Corollary 3.44. $\sim_{\mathcal{L}^c_\mu}$ *is decidable on* Ξ*-systems.*

Proof. Follows from Theorem 3.43 and the fact that $\sim_{hpb}$ is decidable [96]. □

The previous theorem shows that for the class of Ξ-systems the notion of *local causality* defined by $\mathcal{L}^c_\mu$ captures the stronger notion of *global causality*, which is captured by $\sim_{hpb}$ in arbitrary classes of models of true concurrency. The result does not immediately carry over to all finite systems because in such a case $\sim_{\mathcal{L}^c_\mu} \not\equiv \sim_{hpb}$. A simple counter-example (which uses auto-concurrency) is the following: consider the three processes $A_1 = a.A_1$, $A_2 = a.A_2$, and $A_3 = a.A_3$ of Example 3.17; the two concurrent systems $A_1 \parallel A_2$ and $A_1 \parallel A_2 \parallel A_3$ are not hp bisimilar and yet cannot be differentiated by $\sim_{\mathcal{L}^c_\mu}$. However, the counter-example does not rule out, by any means, the possibility that $\sim_{\mathcal{L}^c_\mu}$ is decidable on general systems. In fact, since $\sim_{\mathcal{L}^c_\mu}$ is not as strong as $\sim_{hpb}$ on all finite systems, we believe the following holds:

Conjecture 3.45. $\sim_{\mathcal{L}^c_\mu}$ *is decidable on all finite systems.*

Theorem 3.44 may have interesting practical applications. For instance, the complexity of deciding whether two *arbitrary* concurrent systems are hp bisimilar, i.e., that they possess the same causal properties, is EXPTIME-complete [50] (a result obtained for 1-safe nets); however, since Ξ-systems belong to a *subclass* of general (1-safe) Petri nets, checking $\sim_{hpb}$, and therefore checking $\sim_{\mathcal{L}^c_\mu}$, on the class of Ξ-systems may be computationally easier. Then, computationally easier simply means that it cannot be harder than the EXPTIME-complete complexity bound given by $\sim_{hpb}$ for general systems.

Then, the question is 'how expressive are Ξ-systems?'; and the answer may be given by analysing which systems can be modelled by Ξ-systems and not by free-choice systems without auto-concurrency. Two such systems are mutual exclusion mechanisms and some models of communicating systems. Due to this, the picture looks interesting from a practical viewpoint as well. A more detailed discussion on this topic is presented in Chapter 6. For now, let us present the following examples of two systems that belong to the Ξ family, but that are not free-choice.

Example 3.46. (A net-based mutual exclusion protocol) The concurrent system shown in Figure 3.6 is a Petri net representation of a mutual exclusion protocol, which cannot be represented using a free-choice system, but can be modelled using a Ξ-system. In the figure, the actions r_a and r_b are requests for entering a critical region denoted by γ; moreover, i_a and i_b

(resp. o_a and o_b) are actions for entering (resp. leaving) γ. The unlabelled action abstracts away from the interaction of either subsystem—SysA or SysB—with the rest of the system once permission to access the critical region γ has been granted. ◁

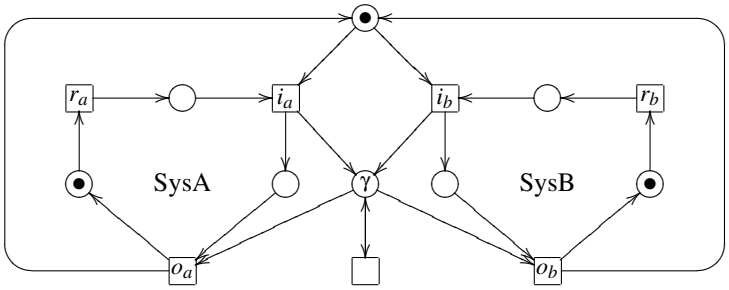


Figure 3.6: A model of a mutual exclusion protocol for two (sub)systems SysA and SysB.

Example 3.47. (A synchronisation model) The concurrent system in Figure 3.7 is the Petri net representation of the CCS process $P = a \parallel \bar{a}$. As before, such a communicating system cannot be represented with a free-choice model as confusion is required. ◁

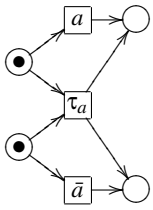


Figure 3.7: A model of the CCS process $P = a \parallel \bar{a}$. The 'synchronization algebra' of P allows a and $\bar{a}$ either to synchronise and produce a joint action τ_a or to be executed independently.

3.4.3 Concurrency Beyond Causality

Some studies on (bisimulation) equivalences for partial order models of concurrency, e.g., [3, 29, 30], suggest that whereas $\sim_{hpb}$ is an equivalence relation only for causality, $\sim_{hhpb}$ is an equivalence for true concurrency; I am neither against nor in favour of this idea. There are also categorical and algebraic studies [27, 35, 51] that support the claim that $\sim_{hhpb}$ is indeed a natural equivalence notion for true concurrency. What is clear is that a bisimulation equivalence relation for concurrency stronger than $\sim_{hpb}$ can capture (partial order) behaviours that go beyond causality, and, as we show in the remainder of this section as well as at the beginning of the next chapter, SFL and $\mathbb{L}_\mu$ can capture some of such kinds of concurrent behaviour.

A Partial Result on the Logical Equivalences Induced by SFL and $\mathbb{L}_\mu$. Although the bisimulation equivalences induced by SFL and $\mathbb{L}_\mu$ are fully analysed in the following chapter using game-theoretical techniques, we first present a simple preliminary result that relates both $\sim_{SFL}$ and $\sim_{\mathbb{L}_\mu}$ with $\sim_{hhpb}$, without using any game-theoretical machinery.

Consider the counter-example given by Fröschle [27] using Petri nets, which provides evidence of the non-coincidence between $\sim_{hpb}$ and $\sim_{hhpb}$ in free-choice systems. Although the systems presented there in Figure 4.8 (page 132) and here in Figure 3.8 are not hhp bisimilar, they cannot be distinguished by any SFL or $\mathbb{L}_\mu$ formula. This result shows that in general the equivalence relation $\sim_{hhpb}$ coincide neither with $\sim_{SFL}$ nor with $\sim_{\mathbb{L}_\mu}$.

Proposition 3.48. *$\sim_{SFL} \not\subseteq \sim_{hhpb}$ and $\sim_{\mathbb{L}_\mu} \not\subseteq \sim_{hhpb}$.*

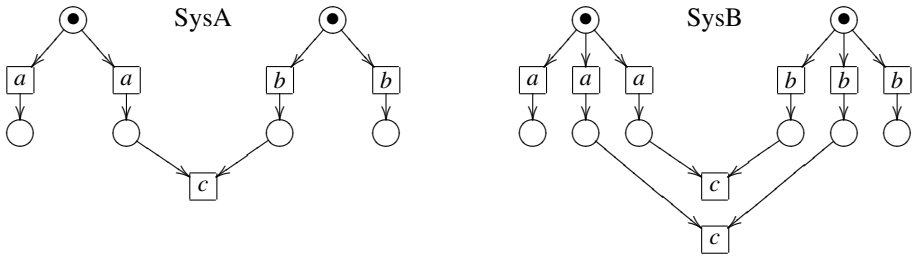


Figure 3.8: Not inclusion of $\sim_{SFL}$ and $\sim_{\mathbb{L}_\mu}$ in $\sim_{hhpb}$. Follows from the fact that the following relations hold for the two systems above: SysA $\{\sim_{SFL}, \sim_{\mathbb{L}_\mu}, \not\sim_{hhpb}\}$ SysB.

A Note on Logical Characterisations of Equivalences for Concurrency. An interesting and challenging problem is that of having (modal) logics capturing bisimulation equivalences for concurrency. A hierarchy of so-called 'true concurrent equivalences' can be found in [26]. Such a hierarchy includes some of the most important equivalence relations for concurrency: strong, step, pomset, hp, and hhp bisimilarity amongst others.[6]

The results presented so far show that a number of bisimilarities in such a hierarchy are captured by (the fixpoint-free fragment of) some of the natural sublogics of SFL and $\mathbb{L}_\mu$; more precisely, two sublogics capture the weakest ($\sim_{sb}$) and the strongest ($\sim_{hpb}$) decidable bisimilarities in such a hierarchy, in the latter case when considering the family of Ξ-systems.

In the first part of the following chapter we continue this study of logical and concurrent equivalences and show that the bisimilarities induced by SFL and $\mathbb{L}_\mu$ lay strictly between $\sim_{hpb}$ and $\sim_{hhpb}$ for the class of Ξ-systems, but this time using a new form of game for bisimulation.

[6] For a description of true concurrency equivalences the reader is referred to [35].

3.5 Summary

In this chapter we have studied the underlying mathematical properties of various partial order models of concurrency based on transition systems, Petri nets, and event structures, and showed that the concurrent behaviour of these systems can be captured in a uniform way by two simple and general dualities of local behaviour.

Such dualities are used to provide partial order semantics of new fixpoint modal logics, some of which induce the same identifications as some of the bisimilarities used in concurrency when considering a number of particular classes of systems.

Moreover, we defined a logical notion of equivalence tailored to be model independent which makes use of a mathematical structure called a process space—a structure that is used as a common bridge between different models of concurrency. Using this approach, two partial order models, possibly of different kinds, can be compared within the same framework by comparing logically their associated process spaces.

Chapter 4

Higher-Order Logic Games

The logic games for verification presented in Chapter 2 provide a *first-order* power on the elements of the board that are picked when playing the game. In order to be able to analyse true-concurrency properties of concurrent systems with partial order semantics, in this chapter we introduce bisimulation and model-checking games that give the players *higher-order* power on the sets of elements of the game board they are allowed to play.

Since such games may be rather powerful without any restrictions, we consider higher-order games where the capabilities of a player are restricted to handle simple characteristic sets of transitions in the boards. Moreover, as these games are intended to be used in the analysis of properties expressible with the modal logics defined in the previous chapter, then such a higher-order power is also restricted to a local setting but freely mixed with fixpoint specifications, which in turn allow for the verification of very complex possibly infinite behaviours.

The main results in this chapter are twofold: on the one hand, by giving game-theoretical characterizations to the logical equivalences induced by SFL and $\mathbb{L}_\mu$—in the form of two game abstraction theorems—it is shown that the bisimilarities that SFL and $\mathbb{L}_\mu$ induce are strictly stronger than hpb and strictly weaker than hhpb when restricted to those systems for which the logical equivalence induced by $\mathcal{L}_\mu^c$ captures hpb, this is, for the class of Ξ-systems.

On the other hand, we also define a new form of sound and complete model-checking games which can verify, in concurrent systems with partial order semantics, several properties not expressible with $\mathcal{L}_\mu$. In particular, such games underpin a novel decision procedure for model-checking all temporal properties of a class of infinite and regular event structures, thus improving, in terms of temporal expressive power, previous results in the literature.

The model-checking games presented in this chapter are defined only for SFL. The $\mathbb{L}_\mu$ version of these games follows exactly the same ideas and therefore is omitted in order to improve the readability of the chapter. Such model-checking games can be found in [40].

4.1 Higher-Order Games for Bisimulation

This section studies higher-order games for bisimulation that help understand the equivalences induced by SFL and $\mathbb{L}_\mu$, and how these equivalences relate to the best known hp bisimilarities for concurrency. To this end, we consider games with monadic second-order power on conflict-free sets of transitions, and show that such games capture the equivalences induced by SFL and $\mathbb{L}_\mu$ and, moreover, are easily related to the logic games that characterise hpb and hhpb.

4.1.1 Model Correspondence

Based on some of the results contained in the previous chapter, we now give a game-theoretical characterisation of the equivalences that SFL and $\mathbb{L}_\mu$ induce by defining bisimulation games for them. The games presented here conservatively extend the hp bisimulation game, and therefore usual games for modal logics, i.e., classical bisimulation. The games we are about to define are called the 'independence hp bisimulation' (ihpb) games and 'trace hp bisimulation' (thpb) games, which characterise, respectively, the logical equivalences induced by SFL and $\mathbb{L}_\mu$.

Since the ihpb (resp. thpb) game induces an equivalence relation that identifies exactly the same set of models as it does SFL (resp. $\mathbb{L}_\mu$), then we say that there is a *model correspondence* between the equivalence relations given by ihpb and SFL (resp. by thpb and $\mathbb{L}_\mu$), which is mathematically captured by a game abstraction theorem for such an equivalence.

We also want to remark that there are some features of SFL and $\mathbb{L}_\mu$ that make them interesting logics for true concurrency; in particular, that, as shown later on, the bisimilarities they induce are decidable and capture behaviours of concurrent systems that go strictly beyond causality. Before presenting the games for SFL and $\mathbb{L}_\mu$, let us make a definition that helps understand the role of support sets as locally identifiable sets of concurrent transitions.

Definition 4.1. Let two sets of transitions R_1 and R_2 be *history-preserving isomorphic* with respect to a pair of transitions (t_m, t_n) if, and only if, there exists a bijection $\mathcal{B}$ between them such that for every $(t_1, t_2) \in \mathcal{B}$, if $t_m \leq t_1$ (resp. $t_m \ominus t_1$) then $t_n \leq t_2$ (resp. $t_n \ominus t_2$). $\triangleleft$

Notice that any infinite play of an hpb game where Eve wins always induces a sequence of history-preserving isomorphic sets, where each set is a singleton. If this was not the case then Adam could win simply by choosing a transition in R_1 or R_2 such that the hp bisimulation would not be synchronous. Let us now define the two new bisimulation games.

Definition 4.2. (Independence history-preserving bisimulation games) An independence history-preserving bisimulation game is a bisimulation game between two players, Eve and Adam, in a pair of systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$ with initial states/processes P and Q, respectively. A configuration of a play in the game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$ is a pair (π_1, π_2), where $\pi_1 \in \Pi_{\mathfrak{T}_1}$ and $\pi_2 \in \Pi_{\mathfrak{T}_2}$. The equivalence relation R_{ihpb} is an independence history-preserving (ihp) bisimulation, $\sim_{ihpb}$, between $\mathfrak{T}_1$ and $\mathfrak{T}_2$ iff it is a history-preserving bisimulation between $\mathfrak{T}_1$ and $\mathfrak{T}_2$, and:

- (Base case) The initial configuration $(\varepsilon,\varepsilon)$ is in R_{ihpb}.
- ($\sim_{ihpb}$ rule) Before Adam chooses a transition t_1 (t_2) from the set of enabled ones at π_1 (π_2), he can also choose a non-empty conflict-free subset of them to be the new set of enabled transitions R_1 (R_2). Then, Eve must respond by choosing a history-preserving isomorphic set R_2 (R_1) with respect to $(\rho(\pi_1),\rho(\pi_2))$ in the opposite structure $\mathfrak{T}_2$ ($\mathfrak{T}_1$).

We say that $\mathfrak{T}_1 \sim_{ihpb} \mathfrak{T}_2$ iff Eve has a winning strategy for the ihpb game $\mathcal{G}(\mathfrak{T}_1,\mathfrak{T}_2)$. ◁

Recall that as usual for bisimulation games, if the play continues forever or Adam cannot make a move, then Eve wins the game. Otherwise Adam wins the game.

Lemma 4.3. *If Eve has a winning strategy for every play in the independence history-preserving bisimulation game $\mathcal{G}(\mathfrak{T}_1,\mathfrak{T}_2)$, then $\mathfrak{T}_1 \sim_{SFL} \mathfrak{T}_2$.*

Proof. By contradiction suppose that Eve has a winning strategy in the ihpb game $\mathcal{G}(\mathfrak{T}_1,\mathfrak{T}_2)$ and that $P \not\sim_{SFL} Q$, where P and Q are the initial processes of $\mathfrak{T}_1$ and $\mathfrak{T}_2$, respectively. There are two cases to analyse. Suppose that at some point Adam cannot make a move. This means that both $P \models [-]\mathrm{ff}$ and $Q \models [-]\mathrm{ff}$ only, which is a contradiction.[1] The other case is when Eve wins in an infinite play. For the same reasons given previously, without any loss of generality, it is possible to consider only the case when Adam uses the $\sim_{ihpb}$ rule of the ihpb game.

Let $P \models \phi_1 * \phi_2$ for some formula $\phi = \phi_1 * \phi_2$ that, by hypothesis, is not satisfied by Q. By the satisfaction relation we have that $P_1 \models \phi_1$ and $P_2 \models \phi_2$ and $Q_1 \not\models \phi_1$ or $Q_2 \not\models \phi_2$, where $P_i = (p, R_i, t)$ for $i \in \{1,2\}$ and $R = R_1 \uplus R_2$ for some set $R \sqsubseteq \mathfrak{X}(p)$, and similarly for the other processes. Then, there are two cases. First, the support set for Q_i cannot be constructed. But this leads to a contradiction since Eve can always do so by hypothesis. The second case is that the support set can be constructed but a synchronous transition in it cannot be found. But this also leads to a contradiction because the support sets that Eve chooses are, additionally, history-preserving isomorphic to the ones that Adam chooses while playing the game.

Therefore all properties that include $*$ must be satisfied at this stage and the the game has to proceed to the next round. However, since the play will continue forever, this holds for all reachable processes, and therefore, all formulae containing $*$ that are satisfied in P must also be satisfied in Q, which is again a contradiction. The case when $P \models \phi_1 \bowtie \phi_2$ is similar. □

Corollary 4.4. (Soundness) *If $\mathfrak{T}_1 \not\sim_{SFL} \mathfrak{T}_2$, then Adam has a winning strategy for every play in the independence history-preserving bisimulation game $\mathcal{G}(\mathfrak{T}_1,\mathfrak{T}_2)$.*

Lemma 4.5. (Completeness) *If $\mathfrak{T}_1 \sim_{SFL} \mathfrak{T}_2$, then Eve has a winning strategy for every play in the independence history-preserving bisimulation game $\mathcal{G}(\mathfrak{T}_1,\mathfrak{T}_2)$.*

[1] Notice that as the models are clear from the context, the decorations in the relation $\models$ are omitted.

Proof. By constructing a winning strategy for Eve based on the fact that $\mathfrak{T}_1 \sim_{SFL} \mathfrak{T}_2$. Since $\mathcal{L}_\mu^c$ induces an hp bisimilarity on Ξ-systems and the ihpb game conservatively extends the hpb game, w.l.o.g. we can consider only the case when Adam plays the $\sim_{ihpb}$ rule of the ihpb game.

So, suppose that Adam is able to choose a conflict-free set of transitions M of size k enabled at $P = (p, M, t)$, where P is a process in the process space associated with $\mathfrak{T}_1$. This implies that $P \models \phi$, where $\phi = \phi_1 * ... * \phi_k$ for some formula ϕ with support set M of size k. By the hypothesis, for some process $Q = (q, N, r)$ that is ihp bisimilar to P, it must be true that $Q \models \phi$ as well, and therefore Eve can choose a conflict-free set N which is the support set for ϕ in $Q = (q, N, r)$. Since $P \sim_{SFL} Q$ then M and N must be history-preserving isomorphic sets with respect to (t, r); otherwise, there would be a simple modal formula differentiating them.

Then Adam must choose an element of either set of transitions using the $\sim_{hpb}$ rule, say a transition $t_m \in M$. However, since for every $i : 1 \leq i \leq k$, $(p, M_i, t) \models \phi_i$ and $(q, N_i, r) \models \phi_i$, where $M = \bigcup_{1 \leq i \leq k} M_i$ and $N = \bigcup_{1 \leq i \leq k} N_i$, and for every $P_i = (p, M_i, t)$ and $Q_i = (q, N_i, r)$, $P_i \sim_{SFL} Q_i$, then it is always possible for Eve to find a transition $r_n \in N$ that synchronises as $t_m \in M$, and proceed to a next round. The play, therefore, must either go on forever or stop because Adam cannot make a move. In either case Eve will win the game. The dual case, for a formula $\phi = \phi_1 \bowtie ... \bowtie \phi_k$, is similar since Adam can always choose to play in either structure. □

Corollary 4.4 (soundness) and Lemma 4.5 (completeness) give a full game-theoretical characterisation to the logical equivalence induced by SFL.

Theorem 4.6. (Full game abstraction) *$\mathfrak{T}_1 \sim_{SFL} \mathfrak{T}_2$ iff Eve has a winning strategy for the ihp bisimulation game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$; conversely, $\mathfrak{T}_1 \not\sim_{SFL} \mathfrak{T}_2$ iff Adam has a winning strategy for the ihp bisimulation game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$.*

Corollary 4.7. (Full logical definability) $\sim_{SFL} \equiv \sim_{ihpb}$.

Now, we turn out attention to the definition of the characteristic game for $\mathbb{L}_\mu$. Following very similar arguments, one can make a few adjustments to the proofs just presented to show that given two systems $\mathfrak{T}_1$ and $\mathfrak{T}_2$, Eve has a winning strategy for every play in the thpb game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$ iff $\mathfrak{T}_1 \sim_{\mathbb{L}_\mu} \mathfrak{T}_2$, i.e., that $\mathfrak{T}_1 \sim_{thpb} \mathfrak{T}_2 \Leftrightarrow \mathfrak{T}_1 \sim_{\mathbb{L}_\mu} \mathfrak{T}_2$, or equivalently that $\sim_{thpb} \equiv \sim_{\mathbb{L}_\mu}$.

Definition 4.8. (Trace history-preserving bisimulation games) Let the pair (π_1, π_2) be a configuration of the game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$. The initial configuration of the game is $(\varepsilon, \varepsilon)$. There are two players, Eve and Adam, and Adam always plays first and chooses where to play before using any rule of the game. The equivalence relation R_{thpb} is a trace history-preserving (thp) bisimulation, $\sim_{thpb}$, between $\mathfrak{T}_1$ and $\mathfrak{T}_2$ iff it is an hp bisimulation between $\mathfrak{T}_1$ and $\mathfrak{T}_2$ and:

- (Base case) The initial configuration $(\varepsilon, \varepsilon)$ is in R_{thpb}.
- ($\sim_{thpb}$ rule). Before Adam chooses a transition using the $\sim_{hpb}$ rule, he can also restrict the set of available transitions by choosing either in π_1 or π_2 a maximal supset to be

the new set of available choices. Then, Eve must choose a maximal set in the other component of the configuration.

We say that $\mathfrak{T}_1 \sim_{thpb} \mathfrak{T}_2$ iff Eve has a winning strategy for the thpb game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$. ◁

Lemma 4.9. *If Eve has a winning strategy for every play in the trace history-preserving bisimulation game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$, then $\mathfrak{T}_1 \sim_{\mathbb{L}_\mu} \mathfrak{T}_2$.*

Proof. By contradiction suppose that Eve has a winning strategy in the thpb game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$ and that $P \not\sim_{\mathbb{L}_\mu} Q$, where $P = (M,t)$ and $Q = (N,r)$ are two processes of $\mathfrak{T}_1$ and $\mathfrak{T}_2$, respectively. As in the SFL case, there are two cases to consider: the first one is when Adam cannot make a move, which leads to a contradiction; and the second one is when Eve wins in an infinite play, for which, as before, we can consider only the case when the rule $\sim_{thpb}$ is necessarily played as the ihpb game also conservatively extends the hpb game on Ξ-systems.

Then, let $P \models \langle\otimes\rangle\phi_1$ that, by hypothesis, is not satisfied by Q. By the satisfaction relation either M is already a maximal supset or there is a maximal supset M' such that $M' \sqsubseteq M$. In addition, such a maximal supset cannot be recognised from the set of transitions N. However, this is not possible since, by hypothesis, Eve can always find such a support set.

Thus, the only other possibility is that the support set can be constructed but a synchronous transition in it cannot be found. But this also leads to a contradiction because the support sets that Eve chooses are, additionally, history-preserving isomorphic to the ones that Adam chooses. Therefore all properties that include $\langle\otimes\rangle$ must be satisfied at this stage and the game has to proceed to the next round. However, since the play will continue forever, this holds for all reachable processes, and therefore, all formulae containing $\langle\otimes\rangle$ that are satisfied in P must also be satisfied in Q, which is again a contradiction. The case when $P \models [\otimes]\,\phi_1$ is similar. □

Corollary 4.10. (Soundness). *If $\mathfrak{T}_1 \not\sim_{\mathbb{L}_\mu} \mathfrak{T}_2$, then Adam has a winning strategy for every play of the trace history-preserving bisimulation game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$.*

Lemma 4.11. (Completeness). *If $\mathfrak{T}_1 \sim_{\mathbb{L}_\mu} \mathfrak{T}_2$, then Eve has a winning strategy for every play of the trace history-preserving bisimulation game $\mathcal{G}(\mathfrak{T}_1, \mathfrak{T}_2)$.*

Proof. By constructing a winning strategy for Eve based on the fact that $\mathfrak{T}_1 \sim_{\mathbb{L}_\mu} \mathfrak{T}_2$. For the same reasons given previously, w.l.o.g., it is possible to consider only the case when Adam uses the $\sim_{thpb}$ rule. So, suppose that Adam is able to choose a maximal set M enabled at $P = (M,t)$, where P is a process in the stateless maximal process space $\mathfrak{S}$ associated with $\mathfrak{T}_1$. This implies that $P \models \phi$, where $\phi = \langle\otimes\rangle\phi_1$ for some formula ϕ with support set M. By the hypothesis, for some process $Q = (N,r)$ that is thp bisimilar to P, it must be true that $Q \models \phi$ as well, and therefore Eve can choose a maximal set N which is the support set for ϕ in $Q = (N,r)$. Since $P \sim_{\mathbb{L}_\mu} Q$ then M and N must be history-preserving isomorphic sets with respect to (t,r); otherwise, there would be a simple modal formula differentiating them.

Then Adam must choose an element of either set of transitions using the $\sim_{hpb}$ rule, say a transition $t' \in M$. But since M and N are history-preserving isomorphic sets with respect to (t,r), then it is always possible for Eve to find a transition $r' \in N$ that synchronises as t', forcing the game to proceed to a next round. Therefore, the play must go on forever or stop because Adam cannot make a move. In either case Eve wins the game. The dual case is similar since Adam can always choose where to play before applying any rule of the game. □

The soundness and completeness results give a full game-theoretical characterisation to the equivalence induced by $\mathbb{L}_\mu$.

Theorem 4.12. (Full game abstraction) $\mathfrak{T}_1 \sim_{\mathbb{L}_\mu} \mathfrak{T}_2$ *iff Eve has a winning strategy for the thp bisimulation game* $\mathcal{G}(\mathfrak{T}_1,\mathfrak{T}_2)$*; conversely,* $\mathfrak{T}_1 \not\sim_{\mathbb{L}_\mu} \mathfrak{T}_2$ *iff Adam has a winning strategy for the thp bisimulation game* $\mathcal{G}(\mathfrak{T}_1,\mathfrak{T}_2)$.

Corollary 4.13. (Full logical definability) $\sim_{\mathbb{L}_\mu} \equiv \sim_{thpb}$.

The previous results let us relate $\sim_{hhpb}$ with both $\sim_{SFL}$ and $\sim_{\mathbb{L}_\mu}$ using game-theoretical arguments. Since all their associated games, namely the one for hhpb, the one for thpb, and the one for ihpb, are conservative extensions of the hpb game, they can be compared just by looking at their additional rules with respect to the hpb game.

Then, consider the game-theoretical definition of hhpb as presented before. Now, only by showing that the additional rule for the hhpb game is at least as powerful as the additional rules for the ihpb and thpb games, and taking into account that, by Proposition 3.48, the equivalences $\sim_{hhpb}$ and $\sim_{SFL}$ as well as $\sim_{hhpb}$ and $\sim_{\mathbb{L}_\mu}$ do not coincide in the general case, then we have:

Theorem 4.14. *Both* $\sim_{hhpb} \subset \sim_{\mathbb{L}_\mu}$ *and* $\sim_{hhpb} \subset \sim_{SFL}$.

Proof. Let us prove this theorem by showing that the contrapositive argument holds, this is: if two systems are not thp (resp. ihp) bisimilar, then they are not hhp bisimilar either. Without loss of generality let us suppose that although the two systems are not thp (resp. ihp) bisimilar, yet they are hp bisimilar; otherwise one would not even need to use the $\{h,i,t\}$hpb rule in order to win the corresponding bisimulation game.

Let $(\pi_1,\pi_2) \in \sim_{thpb}$ but $(\pi_1.r,\pi_2.t) \notin \sim_{thpb}$ for at least one pair of transitions (r,t) that makes the bisimulation equivalence relation fail;[2] moreover, let P and Q be the processes that are reached after executing the runs π_1 and π_2, respectively. Since $(\pi_1.r,\pi_2.t) \notin \sim_{thpb}$, then there exists a conflict-free set of transitions M, where $r \in M$, for which there is no conflict-free set N such that $t \in N$ and $(\pi_1.r,\pi_2.t) \in \sim_{thpb}$, or vice versa. Without loss of generality suppose the former case holds and let i be an index on the elements of M and j an index on the conflict-free sets N^j that can be constructed at Q and for which these two conditions hold: $\mid M \mid = \mid N^j \mid$ and $\forall r \in M.\exists t \in N.\delta(r) = \delta(t)$. Then, since $(\pi_1.r,\pi_2.t) \notin \sim_{ihpb}$, one can conclude that:

[2] For reasons given at the end of the proof, the arguments used hereafter apply to the $\sim_{ihpb}$ case too.

$$\forall i \in \{1,...,k\}.\exists j \in \{1,...,n\}.P \xrightarrow{r_i} P_i \wedge Q \xrightarrow{t_i} Q_i^j \wedge (\pi_1.r_i, \pi_2.t_i) \notin \sim_{thpb}$$

where $k = \mid M \mid$, n is the maximum number of conflict-free sets satisfying the two conditions above, P_i is the process reached after executing the transition r_i from P, and Q_i^j is the process reached after executing the transition $t_i \in N^j$ from Q. We also have that $\delta(r_i) = \delta(t_i)$ as otherwise the last part of the statement would be trivially true.[3]

Given this information, let us show that there is a systematic way of playing the hhpb game that exposes the same mismatch between the concurrent behaviour of the two systems under consideration, implying that the hhpb rule is at least as powerful as the $\{i,t\}$hpb rules. The strategy is simple and has only two stages, which are described next.

Firstly, Adam has to play all actions corresponding to the transitions in M; so, the following sequence is generated: $P \xrightarrow{r_1} P_1 \xrightarrow{r_2^\sim} ... \xrightarrow{r_i^\sim} P_i ... \xrightarrow{r_k^\sim} P_k$, where $r_i \sim r_i^\sim$ for all $i \in \{2,...,k\}$; and Eve will generate a corresponding sequence with her choices: $Q \xrightarrow{t_1} Q_1 \xrightarrow{t_2^\sim} ... \xrightarrow{t_i^\sim} Q_i ... \xrightarrow{t_k^\sim} Q_k$. This way of playing the hhp bisimulation game exposes a set N^j such that $t_1 \in N^j$ and for all i in the set $\{2,...,k\}$ one has that $t_i \sim t_i^\sim$ and $t_i \in N^j$. But now, given this j we know the i for which $P \xrightarrow{r_i} P_i$, $Q \xrightarrow{t_i} Q_i^j$, and $(\pi_1.r_i, \pi_2.t_i) \notin \sim_{thpb}$, i.e, for which $P_i \not\sim_{thpb} Q_i^j$.

Then, using the hhpb rule, Adam has to delete all transitions chosen after reaching P, but the one for which the bisimulation equivalence fails—and let $r_i^\sim$ be such a transition. This can clearly be done because at this point all transitions chosen after P are backwards enabled. Eve must respond by deleting her previous choices accordingly in the only possible way she can. Since $r_i \sim r_i^\sim$ and $t_i \sim t_i^\sim$, after playing in this way the game is in a state as if they had played r_i and t_i at (P,Q), as desired. This concludes the proof for the thpb case.

Finally, regarding the ihpb case, notice that the strategy just described to play the hhpb game can also be used if we had supposed that we were playing the ihpb game. The reason is that we are assuming that at least two conflict-free sets M and N can be constructed with the properties mentioned before (same cardinality and where no two transitions are equally labelled). Therefore, subsequent selections of subsets of M and N are not needed (which can be done in the ihpb game but not in the thpb one). We can simplify the problem in this way as we are assuming that the two systems are hp bisimilar and, moreover, know that there is no auto-concurrency. Otherwise we would not need to use the $\{i,t\}$hpb rules to win the game. □

Remark 4.15. (Decidability of hp bisimilarities) In order to capture the distinguishing power of the $\{i,t\}$hpb rules with the hhpb rule we needed k backtracking moves. This means that a k-hhp bisimilarity as defined by Fröschle *et al.* [29], which is *decidable*, would have been powerful enough to achieve this goal. But note that, in our setting, k is computable on finite systems since it is the bound of their 'concurrency degree', i.e., the maximum number of concurrent transitions in the systems (the cardinality of their biggest conflict-free sets). ◁

[3] Notice that this statement holds for both the ihpb and the thpb games.

The ihpb Game vs. the thpb Game. Let us finish this section by presenting an example that illustrates the main difference between the ihpb game and the thpb game. Such a difference has to do with that in SFL and $\mathbb{L}_\mu$, already discussed in the previous chapter.

Example 4.16. (Linear power in the ihpb game) The main difference between the thpb and ihpb games is that in the ihpb game the players have the power to decompose support sets into smaller ones until they are singletons. This 'linear' power in the ihpb game allows Adam to differentiate systems that cannot be distinguished in the thpb game, e.g., those in Figure 4.1.

The two systems in Figure 4.1 feature *auto-concurrency*, but that is not a problem for Adam to win in the ihp bisimulation game. Interestingly, he can do so even if we had considered *unlabelled* concurrent systems. Let us look at how the two games are played. On the one hand, in the ihpb game Adam can choose to play the only support set available in SysB at that point in the game (whose size is 3). Then, Eve has to choose the corresponding support set available at that point in the game in SysA; such a support set is of size 2. Then, Adam can play the ihpb rule two more times so that he produces a support set that is a singleton set. However, Eve cannot do the same as the support set she chose is of size 2. Then, Adam wins the game without even having to look at the labels of the actions in the Petri net.

On the other hand, in the thpb game Adam will start by doing the same, i.e., choosing a support set in SysB and Eve will respond as in the ihpb game. However, in this case Adam cannot do anything else but choosing an action, whose label is a. Eve can do so in the other system as well. Then, after such a first round the only that both Adam and Eve can do is to choose a support set and afterwards either a causally or a non-causally dependent action which is labelled with an a. Either way, Eve can always do the same Adam does (in any system). Therefore, an infinitely long play is generated where, necessarily, Eve wins. ◁

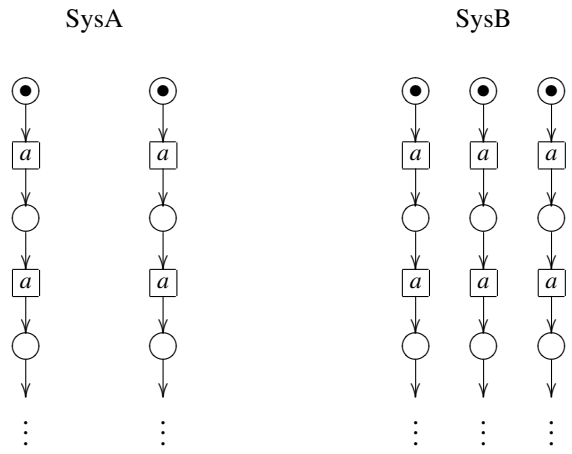


Figure 4.1: Two concurrent systems that are not ihpb equivalent.

4.1.2 Determinacy and Decidability

From results in the previous section we can draw some conclusions about the decidability and determinacy of the games for bisimulation studied in this chapter. In this section, we give further arguments that provide evidence of the decidability and determinacy of these games.

Corollary 4.17. (Determinacy) *Both the ihpb and the thpb games are determined.*

Proof. This follows from the fact that the ihpb game and thpb game are both sound and complete, and therefore also determined. Alternatively, it also follows from some set-theoretic properties of the games. Notice that, as all other bisimulation games presented in this thesis, the ihpb and thpb games are two-player zero-sum perfect-information (infinite) games whose winning conditions define 'Borel' sets, thus due to Martin [59] they are determined. □

Determinacy of the games implies that if Eve does not win a play, then Adam must win it, and vice versa. But, since Eve only wins when the two systems are either SFL-equivalent or $\mathbb{L}_\mu$-equivalent, then Adam must win whenever the two systems are not equivalent. Furthermore, the soundness and completeness properties of the games, i.e., determinacy, can be used (and will be) to show that computing the winner of such games is a decidable problem.

Indeed, decidability on finite systems can be shown by ensuring that the number of elements of a process space (e.g., sets of states, of transitions, or of support sets) that must be computed when playing the game is finite and computable from (i.e., bounded by the size of) the initial specification. In other words, we need to show that the number of different configurations of the games is finite, even for plays of infinite length. For ihpb and thpb games this is the case.

This follows from the fact that whenever Adam cannot make a move or a configuration previously seen is reached again, then such a play can be given as winning for Eve and all previous Eve's moves, which are finite, can be used to construct the winning strategy for Eve. Otherwise, the play is given as winning for Adam and the information of the play (Adam's choices in this case) is used to construct the winning strategy for Adam. Recall that Adam can only win plays of finite length and therefore he always wins after a finite number of rounds.

Theorem 4.18. (Decidability) *Both $\sim_{ihpb}$ and $\sim_{thpb}$ are decidable on finite systems.*

Proof. Since plays of finite length are can all be effectively decided because the game is determined, let us focus on plays of infinite length. We can ensure that any play of infinite length, which is winning for Eve, must visit a previous configuration as follows. Note that an infinite play is possible only if the two systems have cycles (because the systems are finite). Therefore, there must be at least one configuration that is visited infinitely often. By looking at the soundness and completeness proofs of the games it is clear that in order to define a winning strategy for Eve one only needs to analyse the locality of the process space where Eve is playing, rather than the whole history of the game. In the case of the ihpb and thpb games this feature can

be effectively verified because given a state of a partial order model there is always only a finite number of processes with such state in the first component, and similarly, a finite number of support sets relative to such a state. The finiteness of the elements in the third component also follows from these facts—recall that the systems are image-finite and therefore of finite-branching; as a consequence there are only finitely many transitions going to a particular state. Clearly, these arguments also apply for the (stateless maximal) process spaces of $\mathbb{L}_\mu$, since they are strictly smaller than the process spaces of the models for SFL formulae.

More precisely, any finite system produces a finite process space where the number of states (whose size is denoted by m) is bounded by the number of states of the biggest of the two systems being compared. The number of support sets is also bounded by $m.2^n$ (where n is the number of transitions in the systems).[4] Finally, the third component is bounded by the number $n+1$ (because of the additional empty transition in any process space). Therefore, a process space, which is a subset of $S \times \mathfrak{P} \times \mathfrak{A}$ (in the case of SFL and even smaller in the case of $\mathbb{L}_\mu$ because S is not explicitly considered and $\mathcal{X}$ is a subset of $\mathfrak{A}$) has up to $m.(2^n-1).n+1$ processes (because of the additional initial state). Note that the bound of the size of a process space is not $m^2.(2^n-1).n+1$ because a particular support set is defined with respect to a unique state. Since a board is a subset of the Cartesian product of the two process spaces under consideration, such a board has finite size as well. Finally, since one can stop playing a particular play when a configuration is visited again (because according to the soundness and completeness results the games have memoryless winning strategies), then length of any play is bounded by the size of the Cartesian product of the two process spaces just described.

Now, when constructing a winning strategy for Eve it is important to note that only a finite number of processes must be analysed given a particular state of a partial order model. Firstly, notice that a process space embeds the immediate history of a play in the transition component of a process (its last component), and so such information is available locally by exploring a finite number of processes (no more than $2.m.(2^n-1).n$) given a particular element in the process space. Secondly, the support sets that a player can choose given a particular process is also finite (no more than $2.n.2^n-1$ at each round) and can be explored simply by checking all support sets relative to the same either state or support set of the process in the last configuration of the game, i.e., all those in the same neighbourhood. This analysis must be done for all states of the partial order models being compared, but again these sets of states are also finite.

Finally, since Eve wins when Adam cannot make a move (a finite play easily decided) or when a finite set of repeated configurations is visited infinitely often (for infinite plays, which are won only by Eve), then it is always possible to compute the positional winning strategies for Eve, and therefore decidability of these bisimulation games follows. Notice that the proof uniformly applies for both kinds of process spaces, i.e., the one for SFL and the one for $\mathbb{L}_\mu$. □

[4] The bound is actually $m.(2^n-1)$ since support sets cannot be empty.

Due to the correspondence between $\sim_{SFL}$ and $\sim_{ihpb}$ as well as between $\sim_{\mathbb{L}_\mu}$ and $\sim_{thpb}$ over the class of Ξ-systems, the following result immediately holds:

Corollary 4.19. *Both $\sim_{SFL}$ and $\sim_{\mathbb{L}_\mu}$ are decidable over Ξ-systems.*

Proof. Follows from Theorem 4.18 and Corollaries 4.7 and 4.13. □

Nevertheless, no complexity results are currently known for any class of systems. Such a kind of questions should be addressed in the future. As mentioned in Chapter 2 no complexity issues are investigated in this thesis, partly because they are trivially known to be no better than those for interleaving structures, which are already exponential in the case of bisimulation equivalences for general systems. We believe that better complexity results could be achieved only if restricting to simple classes of systems—much simpler than free-choice ones.

Finally, the logical definability results in this section allow one to define a hierarchy of logics and (bisimulation) games for partial order models which is based on the hierarchy of equivalences for true-concurrency studied by Fecher [26]; and moreover, they provide a decidability border with respect to the bisimilarities for true-concurrency in [26] (for Ξ-systems).

4.1.3 A Hierarchy of Logics and Games

An interesting problem in concurrency theory is that of having (modal) logics and games capturing standard bisimilarities for concurrency. A hierarchy of so-called 'true concurrent equivalences' can be found in [26]. The results presented so far define a hierarchy of equivalences for concurrent systems with partial order semantics, where the bisimilarities induced by SFL and $\mathbb{L}_\mu$ rank at the top of the decidable equivalences in such a hierarchy when restricted to Ξ-systems. As such equivalences are related to some logics (through their characteristic games), then there is also an induced hierarchy, in terms of expressivity, for such logics and games.

Prior to this work, we had that on systems without auto-concurrency $\sim_{hhpb}$ was captured by the Path Logic (PL) studied by Nielsen *et al.* [70], as well as the result by Milner and Hennessy [46] that on image-finite systems $\sim_{sb}$ is captured by HML; also, we had that $\sim_{hpb}$ is captured by the logics L_P and L_T studied by De Nicola and Ferrari [68]. We have studied logics that add to results of this kind, in the context of fc-structured systems, since in Chapter 3 it was shown that $\mathcal{L}^c_\mu$ also captures $\sim_{hpb}$. However, $\mathcal{L}^c_\mu$ does so by following both a *forward* and a *local* style of reasoning as opposed to what is done in other settings where causality is captured by means of either past tense operators or global reasoning on infinitely large sets of events.[5]

Moreover, in this chapter, two new equivalences have been introduced, namely $\sim_{ihpb}$ and $\sim_{thpb}$, which have been shown to be decidable and strictly between $\sim_{hpb}$ and $\sim_{hhpb}$ in terms of discriminating power. These results are summarised in Figure 4.2, where $\sim_{PL}$ represents

[5] A more detailed description of other logics and related work is given in Chapter 6.

the bisimilarity induced by PL and $\sim_{L_{\{P,T\}}}$ the one induced by both L_P and L_T; moreover, $\sim_{eq}$ refers to several other equivalences for concurrency, which are not studied in this thesis, e.g., step or pomset bisimulation equivalences. The original hierarchy can be found in [26].

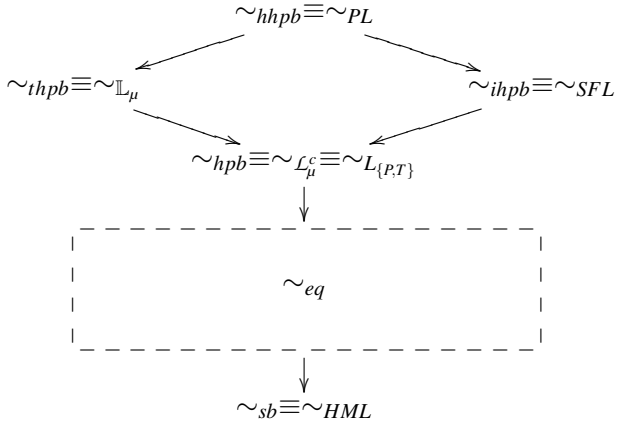


Figure 4.2: A hierarchy of equivalences for concurrency. The arrow $\rightarrow$ means inclusion $\subset$.

Only recently Baldan and Crafa [7] developed a logic that captures many of the bisimilarities in [26]. Their work is partly inspired by the results in this PhD thesis. A description of their work together with some ideas for a possible "only forward" game characterisation of the equivalence induced by their logic, which captures $\sim_{hhpb}$ on event structures is discussed in the final chapter of this thesis since it is actually seen as a potential avenue for future work.

4.2 Higher-Order Games for Model-Checking

In this section we introduce logic games for model-checking that allow *local* monadic second-order power on sets of independent transitions in the underlying partial order models of concurrency where the games are played. Since the interleaving semantics of such models is not considered, some problems that may arise when using interleaving representations are avoided and new decidability results for classes of partial order models are achieved [42, 43].

This kind of logic games for verification is *sound* and *complete*, and therefore determined. While in the interleaving case they coincide with the local model-checking games for $\mathcal{L}_\mu$, in a partial order setting they can be used to verify temporal true-concurrency properties which are not expressible with $\mathcal{L}_\mu$ over concurrent systems with partial order semantics.

As said before, similar to the case of higher-order logic games for bisimulation, the two players in this new game are given local monadic second-order power on conflict-free sets of transitions. The technical details behind the construction of (and proofs for) this game follow seminal ideas on local model-checking games for $\mathcal{L}_\mu$ as presented by Stirling [89].

4.2.1 LMSO Model-Checking Games

Trace local monadic second-order (LMSO) model-checking games $\mathcal{G}(\mathfrak{M},\phi)$ are played on a model $\mathfrak{M} = (\mathfrak{T},\mathcal{V})$, where $\mathfrak{T} = (S,s_0,T,I,\Sigma)$ is a system, and on an SFL formula ϕ. This logic game can also be presented as $\mathcal{G}_{\mathfrak{M}}(H_0,\phi)$, or even as $\mathcal{G}_{\mathfrak{M}}(s_0,\phi)$, where $H_0 = (s_0,\mathfrak{X}_0(s_0),t_\varepsilon)$ is the initial process of $\mathbb{S}$. We write $\mathcal{G}$ instead of $\mathcal{G}_{\mathfrak{M}}$ since $\mathfrak{M}$ is usually clear from the context. The game board in which the game is played has the form $\mathfrak{B} \subseteq \mathbb{S} \times Sub(\phi)$, where $\mathbb{S}$ is the process space $S \times \mathfrak{P} \times \mathfrak{A}$ associated with $\mathfrak{T}$ and $Sub(\phi)$ is the subformula set of ϕ, which is formally defined by the Fischer–Ladner closure of SFL formulae in the following way:

Definition 4.20. (Fischer–Ladner closure of SFL formulae) The 'subformulae' or subformula set $Sub(\phi)$ of an SFL formula ϕ is given in the following way:

$$\begin{aligned}
Sub(Z) &= \{Z\}\\
Sub(\phi_1 \vee \phi_2) &= \{\phi_1 \vee \phi_2\} \cup Sub(\phi_1) \cup Sub(\phi_2)\\
Sub(\langle a\rangle_c \phi_1) &= \{\langle a\rangle_c \phi_1\} \cup Sub(\phi_1)\\
Sub(\langle a\rangle_{nc} \phi_1) &= \{\langle a\rangle_{nc} \phi_1\} \cup Sub(\phi_1)\\
Sub(\phi_1 * \phi_2) &= \{\phi_1 * \phi_2\} \cup Sub(\phi_1) \cup Sub(\phi_2)\\
Sub(\mu Z.\phi_1) &= \{\mu Z.\phi_1\} \cup Sub(\phi_1)
\end{aligned}$$

and similarly for the dual operators $\vee$, $[a]_c$, $[a]_{nc}$, $\bowtie$, and ν. ◁

A play in a trace LMSO model-checking game is a possibly infinite sequence of game configurations $C_0, C_1, \ldots$, written as $(s,R,t) \vdash \phi$ or $H \vdash \phi$ whenever possible; each C_i is an element of the board $\mathfrak{B}$. Every play starts in the configuration $C_0 = H_0 \vdash \phi$, and proceeds according to the rules of the game given in Figure 4.3. As usual for model-checking games, Eve tries to prove that $H_0 \models \phi$ whereas Adam tries to show, instead, that $H_0 \not\models \phi$.

The rules (FP) and (VAR) control the unfolding of fixpoint operators. Their correctness is based on the fact that $\eta Z.\phi \equiv \phi\left[\eta Z.\phi/Z\right]$, where $\eta \in \{\mu,\nu\}$, according to the semantics of the logic. Rules $(\vee)$ and $(\wedge)$ have the same meaning as the disjunction and conjunction rules, respectively, in a Hintikka game for propositional logic. Rules $(\langle\,\rangle_c)$, $(\langle\,\rangle_{nc})$, $([\,]_c)$ and $([\,]_{nc})$ are like the rules for quantifiers in a standard Hintikka game semantics for first-order (FO) logic, provided that the box and diamond operators behave, respectively, as restricted universal and existential quantifiers sensitive to the causal information in the partial order model.

Finally, the most interesting rules are $(*)$ and $(\bowtie)$. Local monadic second-order moves are used to recognize conflict-free sets of transitions in $\mathfrak{M}$, i.e., those in the same *Mazurkiewicz trace*. Such moves, which restrict the second-order power (locally) to Mazurkiewicz traces, give the name to this game. The use of the rules $(*)$ and $(\bowtie)$ requires both players to make a choice in the same round: whereas the player who moves first must look for two conflict-free sets of transitions R_0 and R_1, the player that moves afterwards has to select a formula ϕ_i whose support set will be the corresponding R_i, for $i \in \{0,1\}$, just chosen by the other player.

$$\text{(FP)}\ \frac{H \vdash \overset{\nu}{\mu} Z.\phi}{H \vdash Z} \qquad \text{(VAR)}\ \frac{H \vdash Z}{H \vdash \phi}\ \text{for some } \overset{\nu}{\mu} Z.\phi$$

$$(\vee)\ \frac{H \vdash \phi_0 \vee \phi_1}{H \vdash \phi_i}\ [\exists] i:\ i \in \{0,1\} \qquad (\wedge)\ \frac{H \vdash \phi_0 \wedge \phi_1}{H \vdash \phi_i}\ [\forall] i:\ i \in \{0,1\}$$

$$(\langle\,\rangle_c)\ \frac{(s,R,t) \vdash \langle a \rangle_c \phi}{(s',\mathfrak{X}(s'),t') \vdash \phi}\ [\exists] a:\ t' = s \xrightarrow{a} s',\ t' \in R,\ t \leq t'$$

$$(\langle\,\rangle_{nc})\ \frac{(s,R,t) \vdash \langle a \rangle_{nc} \phi}{(s',\mathfrak{X}(s'),t') \vdash \phi}\ [\exists] a:\ t' = s \xrightarrow{a} s',\ t' \in R,\ t \ominus t'$$

$$([\,]_c)\ \frac{(s,R,t) \vdash [a]_c \phi}{(s',\mathfrak{X}(s'),t') \vdash \phi}\ [\forall] a:\ t' = s \xrightarrow{a} s',\ t' \in R,\ t \leq t'$$

$$([\,]_{nc})\ \frac{(s,R,t) \vdash [a]_{nc} \phi}{(s',\mathfrak{X}(s'),t') \vdash \phi}\ [\forall] a:\ t' = s \xrightarrow{a} s',\ t' \in R,\ t \ominus t'$$

$$(*)\ \frac{(s,R,t) \vdash \phi_0 * \phi_1}{(s,R_i,t) \vdash \phi_i}\ [\exists] R_0,R_1; [\forall] i:\ R_0 \uplus R_1 \sqsubseteq R, i \in \{0,1\}$$

$$(\bowtie)\ \frac{(s,R,t) \vdash \phi_0 \bowtie \phi_1}{(s,R_i,t) \vdash \phi_i}\ [\forall] R_0,R_1; [\exists] i:\ R_0 \uplus R_1 \sqsubseteq R, i \in \{0,1\}$$

Figure 4.3: Trace LMSO Model-Checking Game Rules of SFL. Whereas the notation $[\forall]$ denotes a choice made by Adam, the notation $[\exists]$ denotes a choice made by Eve.

Guided by the semantics of $*$ (resp. $\bowtie$), it is defined that Eve (resp. Adam) must look for a pair of non-empty conflict-free sets R_0 and R_1 to be assigned to each formula ϕ_i as their support sets. This is equivalent to playing a supset for each subformula in the configuration. Then Adam (resp. Eve) chooses one of the two subformulae, with full knowledge of the sets that have been given by Eve (resp. Adam). Note that $*$ can be regarded as a kind of conjunction and $\bowtie$ of disjunction. Indeed, they are a structural conjunction and disjunction, respectively.

Remark 4.21. A similar setting can be defined for $\mathbb{L}_\mu$ (as done in [40]). The main difference between the two games is in the rules for $\langle\otimes\rangle$ and $[\otimes]$, which also allow LMSO moves to recognize conflict-free sets. The use of $\langle\otimes\rangle$ and $[\otimes]$ requires a player to make a choice, locally, on a set of transitions rather than on a singleton set as in the traditional games for model-checking. Guided by the semantics of $\langle\otimes\rangle$ (resp. $[\otimes]$), it is defined that Eve (resp. Adam) must look for a maximal set to be assigned to a particular $\mathbb{L}_\mu$ formula as its support set. $\triangleleft$

Definition 4.22. (Winning conditions) The following rules are the *winning conditions* that determine a unique winner for every finite or infinite play $C_0, C_1, ...$ in a game $\mathcal{G}(H_0, \phi)$. As defined before, $\mathfrak{X}(s')$ is the maximal set at a state s'.

Adam wins a finite play $C_0, C_1, ..., C_n$ or an infinite play $C_0, C_1, ...$ iff:

1. $C_n = H \vdash Z$ and $H \notin \mathcal{V}(Z)$.
2. $C_n = (s,R,t) \vdash \langle a \rangle_c \psi$ and $\{(s', \mathfrak{X}(s'), t') : t \leq t' = s \xrightarrow{a} s' \in R\} = \emptyset$.
3. $C_n = (s,R,t) \vdash \langle a \rangle_{nc} \psi$ and $\{(s', \mathfrak{X}(s'), t') : t \ominus t' = s \xrightarrow{a} s' \in R\} = \emptyset$.
4. $C_n = (s,R,t) \vdash \phi_0 * \phi_1$ and $\{(s, R_0 \cup R_1, t) : R_0 \uplus R_1 \sqsubseteq R\} = \emptyset$.
5. The play is infinite and there are infinitely many configurations where Z appears, such that Z is the least fixpoint of some subformula $\mu Z.\psi$ and the syntactically outermost variable in ϕ that occurs infinitely often.

Eve wins a finite play $C_0, C_1, ..., C_n$ or an infinite play $C_0, C_1, ...$ iff:

1. $C_n = H \vdash Z$ and $H \in \mathcal{V}(Z)$.
2. $C_n = (s,R,t) \vdash [a]_c \psi$ and $\{(s', \mathfrak{X}(s'), t') : t \leq t' = s \xrightarrow{a} s' \in R\} = \emptyset$.
3. $C_n = (s,R,t) \vdash [a]_{nc} \psi$ and $\{(s', \mathfrak{X}(s'), t') : t \ominus t' = s \xrightarrow{a} s' \in R\} = \emptyset$.
4. $C_n = (s,R,t) \vdash \phi_0 \bowtie \phi_1$ and $\{(s, R_0 \cup R_1, t) : R_0 \uplus R_1 \sqsubseteq R\} = \emptyset$.
5. The play is infinite and there are infinitely many configurations where Z appears, such that Z is the greatest fixpoint of some subformula $\nu Z.\psi$ and the syntactically outermost variable in ϕ that occurs infinitely often.

We are now ready to show that trace LMSO model-checking games are sound, complete, and determined—despite their higher-order power. An important step in proving so is to show the effective construction of the strategies that Eve and Adam follow while playing the game.

4.2.2 Soundness and Completeness

Let us first give some intermediate results. The statements in this section are all either standard modal mu-calculus mathematical statements, or standard statements where additional cases for the new operators of SFL need to be checked. Let $\mathfrak{T}$ be a system and $C = (s,R,t) \vdash \psi$ a configuration in the game $\mathcal{G}(H_0, \phi)$, as defined before. As usual, the denotation $\|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$ of an SFL formula ϕ in the model $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$ is a subset of the process space $\mathbb{S}$. Let a game configuration C of $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$ be *true* iff $(s,R,t) \in \|\psi\|^{\mathfrak{T}}_{\mathcal{V}}$ holds and be *false* otherwise.

Fact 4.23. *SFL is closed under negation.*

Lemma 4.24. *A game $\mathcal{G}(H_0, \phi)$, where Eve has a winning strategy, has a dual game $\mathcal{G}(H_0, \neg\phi)$ where Adam has a winning strategy, and conversely.*

Proof. Note that as SFL is closed under negation, for every rule that requires a player to make a choice on a formula ψ there is a dual rule in which the other player makes a choice on the negated formula $\neg\psi$. Also, note that for every winning condition for one of the players in a formula ψ there is a dual winning condition for the other player in $\neg\psi$. Now, suppose Eve has a winning strategy π in $\mathcal{G}(H_0,\phi)$. Adam can use π in the dual game $\mathcal{G}(H_0,\neg\phi)$ since whenever he has to make a choice, by duality, there is a rule that requires $\exists$ to make a choice in $\mathcal{G}(H_0,\phi)$. In this way, regardless of the choices that Eve makes, Adam can enforce a winning play for himself. The case when Adam has a winning strategy in the game $\mathcal{G}(H_0,\phi)$ is dual. □

Lemma 4.25. *Eve preserves falsity and can preserve truth with her choices. Hence, she cannot choose true configurations when playing in a false configuration. Dually, Adam preserves truth and can preserve falsity with his choices. Then, he cannot choose false configurations when playing in a true configuration.*

Proof. The cases for the rules $(\wedge)$ and $(\vee)$ are just as for the Hintikka evaluation games for FO logic. Thus, let us go on to check the rules for the other operators. Firstly, consider the rule $(\langle\,\rangle_c)$ and a configuration $C = (s,R,t) \vdash \langle a\rangle_c\psi$, and suppose that C is false. In this case there is no a such that $t \leq t' = s \xrightarrow{a} s' \in R$, and $(s',\mathfrak{X}(s'),t') \in \|\psi\|^{\mathfrak{T}}_{\mathcal{V}}$. Hence, the following configurations will be false as well. Contrarily, if C is true, then Eve can make the next configuration $(s',\mathfrak{X}(s'),t') \vdash \psi$ true by choosing a transition $t' = s \xrightarrow{a} s' \in R$ such that $t \leq t'$. The case for $(\langle\,\rangle_{nc})$ is similar (simply change $\leq$ for $\ominus$), and the cases for $([\,]_c)$ and $([\,]_{nc})$ are dual.

Now, consider the rule $(*)$ and a configuration $C = (s,R,t) \vdash \psi_0 * \psi_1$, and suppose that C is false. In this case there is no pair of sets R_0 and R_1 such that $R_0 \uplus R_1 \sqsubseteq R$ and both $(s,R_0,t) \in \|\psi_0\|^{\mathfrak{T}}_{\mathcal{V}}$ and $(s,R_1,t) \in \|\psi_1\|^{\mathfrak{T}}_{\mathcal{V}}$ to be chosen by Eve. Hence, Adam can preserve falsity by choosing the $i \in \{0,1\}$ where $(s,R_i,t) \notin \|\psi_i\|^{\mathfrak{T}}_{\mathcal{V}}$, and the next configuration $(s,R_i,t) \vdash \psi_i$ will be false too. On the other hand, suppose that C is true. In this case, regardless of which i Adam chooses, Eve has previously fixed two support sets R_0 and R_1 such that for every $i \in \{0,1\}$, one has that $(s,R_i,t) \in \|\psi_i\|^{\mathfrak{T}}_{\mathcal{V}}$. Thus, the next configuration $(s,R_i,t) \vdash \psi_i$ will be true as well.

Finally, the deterministic rules (FP) and (VAR) preserve both truth and falsity because of the semantics of fixpoint operators. Recall that for any process H, if $H \in \|^{\nu}_{\mu}Z.\psi\|$ then $H \in \|\psi\|_{Z:=\|^{\nu}_{\mu}Z.\psi\|}$ for all free variables Z in ψ. □

Lemma 4.26. *In any infinite play of a game $\mathcal{G}(H_0,\phi)$ there is a unique syntactically outermost variable that occurs infinitely often.*

Proof. By contradiction, assume that the statement is false. Without loss of generality, suppose that there are two variables Z and Y that are syntactically outermost and appear infinitely often. The only possibility for this to happen is that they are at the same level in ϕ. However, if this is the case Z and Y cannot occur infinitely often unless there is another variable X that also

occurs infinitely often and whose unfolding contains Z and Y. But this means that Z and Y are syntactically beneath X, and hence neither Z nor Y is outermost, which is a contradiction. □

Fact 4.27. *Only rule* (VAR) *can increase the size of a formula in a configuration. All other rules decrease the size of formulae in configurations.*

Lemma 4.28. *Every play of a game $\mathcal{G}(H_0, \phi)$ has a uniquely determined winner.*

Proof. Suppose the play is of finite length. Then, the winner of the game is uniquely determined by one of the winning conditions one to four (Definition 4.22) of either Eve or Adam since such winning rules cover all possible cases and, moreover, are mutually exclusive. Now, suppose that the play is of infinite length. Due to Fact 4.27, rule (VAR) must be used infinitely often in the game, and thus, there is at least one fixpoint variable that is replaced by its defining fixpoint formula each time it occurs. Therefore, winning condition five of one of the players can be used to uniquely determine the winner of the game since, due to Lemma 4.26, there is a unique syntactically outermost fixpoint variable that occurs infinitely often. □

Definition 4.29. (Approximants) Let Z be the least fixpoint of some formula ϕ and let $\alpha, \lambda \in \mathbb{O}\mathrm{rd}$ be two ordinals, where λ is a limit ordinal. Then:

$$Z^0 := \mathrm{ff}, \qquad Z^{\alpha+1} = \phi[Z^\alpha/Z], \qquad Z^\lambda = \bigvee_{\alpha<\lambda} Z^\alpha$$

For greatest fixpoints the approximants are defined dually. Let Z be the greatest fixpoint of some formula ϕ and, as before, let $\alpha, \lambda \in \mathbb{O}\mathrm{rd}$ be two ordinals, where λ is a limit ordinal. Then:

$$Z^0 := \mathrm{tt}, \qquad Z^{\alpha+1} = \phi[Z^\alpha/Z], \qquad Z^\lambda = \bigwedge_{\alpha<\lambda} Z^\alpha \qquad \triangleleft$$

It is now possible to show that the analysis for fixpoint modal logics [14] can be extended to this scenario. The proof of soundness uses similar arguments to that in $\mathcal{L}_\mu$ case, but it is presented here in full because it is the basis of the decision procedure for SFL model-checking.

Theorem 4.30. (Soundness) *Let $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$ be a model of a formula ϕ in the game $\mathcal{G}(H_0, \phi)$. If $H_0 \notin \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$ then Adam wins $H_0 \vdash \phi$.*

Proof. Suppose $H_0 \notin \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$. We construct a possibly infinite game tree that starts in $H_0 \vdash \phi$, for Adam. We do so by preserving falsity according to Lemma 4.25, i.e., whenever a rule requires Adam to make a choice then the tree will contain the successor configuration that preserves falsity. All other choices that are available for Eve are included in the game tree.

First, consider only finite plays. Since Eve only wins finite plays that end in true configurations, then she cannot win any finite play by using her winning conditions one to four. Hence, Adam wins each finite play in this game tree. Now, consider infinite plays. The only chance for Eve to win is to use her winning condition five. So, let the configuration $H \vdash \nu Z.\phi$ be reached

such that Z is the syntactically outermost variable that appears infinitely often in the play according to Lemma 4.26. In the next configuration $H \vdash Z$, variable Z is interpreted as the least approximant Z^α such that $H \notin \|Z^\alpha\|^{\mathfrak{T}}_{\mathcal{V}}$ and $H \in \|Z^{\alpha-1}\|^{\mathfrak{T}}_{\mathcal{V}}$, by the principle of fixpoint induction. As a matter of fact, by monotonicity and due to the definition of fixpoint approximants it must also be true that $H \in \|Z^\beta\|^{\mathfrak{T}}_{\mathcal{V}}$ for all ordinals β such that $\beta < \alpha$. Note that, also due to the definition of fixpoint approximants, α cannot be a limit ordinal λ because this would mean that $H \notin \|Z^\lambda = \bigwedge_{\beta<\lambda} Z^\beta\|^{\mathfrak{T}}_{\mathcal{V}}$ and $H \in \|Z^\beta\|^{\mathfrak{T}}_{\mathcal{V}}$ for all $\beta < \lambda$, which is impossible.

Since Z is the outermost fixpoint variable that occurs infinitely often and the game rules follow the syntactic structure of SFL formulae, the next time that a configuration $C' = H' \vdash Z$ is reached, Z can be interpreted as $Z^{\alpha-1}$ in order to make C' false as well. And again, if $\alpha - 1$ is a limit ordinal λ, then there must be a $\gamma < \lambda$ such that both $H' \notin \|Z^\gamma\|^{\mathfrak{T}}_{\mathcal{V}}$ and $H' \in \|Z^{\gamma-1}\|^{\mathfrak{T}}_{\mathcal{V}}$. One can repeat this process even until $\lambda = \omega$.

But, since ordinals are well-founded the play must eventually reach a false configuration $C'' = H'' \vdash Z$ where Z is interpreted as Z^0. And, according to Definition 4.29, $Z^0 := \mathrm{tt}$, which leads to a contradiction since the configuration $C'' = H'' \vdash \mathrm{tt}$ should be false, i.e., $H'' \in \|\mathrm{tt}\|^{\mathfrak{T}}_{\mathcal{V}}$ should be false, which is impossible. In other words, if H had failed a maximal fixpoint, then there must have been a descending chain of failures, but, as can be seen, there is not.

As a consequence, there is no such least α that makes the configuration $H \vdash Z^\alpha$ false, and hence, the configuration $H \vdash \nu Z.\phi$ could not have been false either. Therefore, Eve cannot win any infinite play with her winning condition 5 either. Since Eve can win neither finite plays nor infinite ones whenever $H_0 \notin \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$, then Adam must win all plays of $\mathcal{G}(H_0, \phi)$. □

Remark 4.31. If only finite state systems are considered $\mathbb{O}\mathrm{rd}$, the set of ordinals, can be replaced by $\mathbb{N}$, the set of natural numbers. ◁

Notice that, in our setting, the previous remark is particularly important when the system $\mathfrak{T}$ in a model $\mathfrak{M}$ is the TSI representation of an event structure, since any concurrent system featuring recursive behaviour would be represented by an infinite event structure, and hence, by an infinite-state TSI model, if one uses the mapping from event structures to TSI given previously. Therefore, in this setting, we have to consider the possibility of dealing with infinite-state systems in order for the results of this section to apply to all the partial order models we presented in Chapter 2, as well as to the interleaving models they generalize.

Theorem 4.32. (Completeness) *Let $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$ be a model of a formula ϕ in the game $\mathcal{G}(H_0, \phi)$. If $H_0 \in \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$ then Eve wins $H_0 \vdash \phi$.*

Proof. Suppose that $H_0 \in \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$. Due to Fact 4.23 it is also true that $H_0 \notin \|\neg\phi\|^{\mathfrak{T}}_{\mathcal{V}}$. According to Theorem 4.30 (soundness), Adam wins $H_0 \vdash \neg\phi$, i.e., has a winning strategy in the game $\mathcal{G}(H_0, \neg\phi)$. And, due to Lemma 4.24, Eve has a winning strategy in the dual game $\mathcal{G}(H_0, \phi)$. Therefore, Eve wins $H_0 \vdash \phi$ if $H_0 \in \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$, as desired. □

Theorems 4.30 (soundness) and 4.32 (completeness) imply that the game is also determined. Determinacy and perfect information make the notion of truth defined by this Hintikka game semantics coincide with its Tarskian denotational counterpart.

Corollary 4.33. (Determinacy) *Adam wins the game $\mathcal{G}(H_0, \phi)$ iff Eve does not win it.*

4.2.3 Local Properties and Decidability

It has been shown that trace LMSO model-checking games are still sound and complete even when players are allowed to manipulate sets of independent transitions. Importantly, the power of these model-checking games, and also of SFL, is that such a second-order quantification is kept both *local* and restricted to transitions in the same *Mazurkiewicz trace*.

We now show that trace LMSO model-checking games enjoy several local properties that in turn make them *decidable* in the finite case. Such a decidability result is used later on to extend the decidability border of model-checking a category of partial order models of concurrency.

Proposition 4.34. (Winning strategies) *The winning strategies for the trace LMSO model-checking games of Separation Fixpoint Logic are history-free.*

Proof. Consider a winning strategy π for Eve. According to Lemma 4.25 and Theorem 4.32 such a strategy consists of preserving truth with her choices and annotating variables with their approximant indices. But neither of these two tasks depends on the history of a play. Instead they only depend on the current configuration of the game. In particular notice that, of course, this is also the case for the structural operators since the second-order quantification has only a local scope. Similar arguments apply for the winning strategies of Adam. □

Remark 4.35. Corollary 4.33 and Proposition 4.34 follow also from the fact that the trace LMSO model-checking games for SFL are a form of parity games with perfect information, a game for which history-free winning strategies are known. This kind of winning strategies are desirable, from an algorithmic viewpoint, since they are easier to synthesize. ◁

This result is key to achieve *decidability* of these games in the presence of the local second-order quantification on the Mazurkiewicz traces of the partial order models considered here.

Theorem 4.36. *The model-checking game for finite systems against Separation Fixpoint Logic specifications is decidable.*

Proof. A game is decidable if one can tell in all possible cases which of the players has a winning strategy. As the game is determined, finite plays are decided by winning conditions one to four of either player. Now, consider the case of infinite plays; since the winning strategies of both players are history-free, we only need to look at the set of different configurations in the game, which is finite even for plays of infinite length. Now, in a finite system an infinite

play can be possible only if the model is cyclic. But, since the model has a finite number of states, there is an upper bound on the number of fixpoint approximants that must be calculated (as well as on the number of configurations of the game board that must be checked) in order to ensure that either a greatest fixpoint is satisfied or a least fixpoint has failed. As a consequence, all possible history-free winning strategies for a play of infinite length can be computed, so that the game can be decided using winning condition five of one of the players. □

Remark 4.37. (Complexity) The complexity of model-checking is in principle substantially worse than for $\mathcal{L}_\mu$, but in practice not. The change in complexity from plain L_μ arises from the local second-order quantification in the $*$ operator – in principle, this could involve choosing a partition of a set of the order of the size of the state space, making the $*$ operation NP in the state space; hence the complexity for a formula of length k and alternation depth d on a system of size n is $O(kn.2^{nd})$ with the simple algorithms (or $O(kn.2^{nd/2})$ using the Browne *et al.* optimization). This maximal complexity occurs in highly concurrent systems, where it is the inevitable manifestation of state explosion. For typical systems encountered in reality, where the concurrency is small compared to the overall size, the support sets will be much smaller than the size of the system. Hence for practical purposes, the complexity is unlikely to be significantly worse than that of $\mathcal{L}_\mu$.[6] ◁

The Interleaving Case. Local properties of trace LMSO model-checking games can also be found in the interleaving case, namely, they coincide with the local model-checking games for the modal mu-calculus as defined by Stirling [89]. As shown in the previous chapter interleaving systems can be cast using SFL by both syntactic and semantic means. The importance of this feature of SFL is that even having constructs for independence and a partial order model, nothing is lost with respect to the main approaches to interleaving concurrency. Recall that $\mathcal{L}_\mu$ can be obtained from SFL by considering the $*$-free language and using only the following derived operators: $\langle a\rangle\phi = \langle a\rangle_c\phi \vee \langle a\rangle_{nc}\phi$ and $[a]\,\phi = [a]_c\,\phi \wedge [a]_{nc}\,\phi$.

Proposition 4.38. *If either a model with an empty independence relation or the syntactic $\mathcal{L}_\mu$ fragment of SFL is considered, then the trace LMSO model-checking games for SFL degenerate to the local model-checking games for $\mathcal{L}_\mu$.*

Proof. Let us consider the case when the syntactic $\mathcal{L}_\mu$ fragment of SFL is considered. The first observation to be made is that the $*$-free fragment of SFL only considers *maximal sets*. Hence if a transition can be performed at s then it is always in the support set at s. Therefore, support sets in $\mathfrak{P}$ can be disregarded. Also, without loss of generality, consider only the case of the modal operators since the $\mathcal{L}_\mu$ and SFL boolean and fixpoint operators have the same denotation.

$$\begin{aligned}\|\langle a\rangle\phi_1\|^{\mathfrak{T}}_{\mathcal{V}} \quad = \quad & \{(s,t) \in S \times \mathfrak{A} \mid \exists s' \in S.\ t \leq t' = s \xrightarrow{a} s' \wedge (s',t') \in \|\phi_1\|^{\mathfrak{T}}_{\mathcal{V}}\} \\ & \cup \{(s,t) \in S \times \mathfrak{A} \mid \exists s' \in S.\ t \ominus t' = s \xrightarrow{a} s' \wedge (s',t') \in \|\phi_1\|^{\mathfrak{T}}_{\mathcal{V}}\}\end{aligned}$$

[6] This remark is entirely due to Julian Bradfield. I thank him for it.

The second observation is that when computing the semantics of the combined modal operator $\langle a \rangle$, the conditions $t \leq t'$, i.e., $(t,t') \notin I$, and $t \ominus t'$, i.e., $(t,t') \in I$, complement each other and become always true (since there are no other possibilities). Therefore, the second component of every pair in the structure $S \times \mathfrak{A}$ can also be disregarded.

$$\|\langle a \rangle \phi_1\|^{\mathfrak{T}}_{\mathcal{V}} \quad = \quad \{s \in S \mid \exists s' \in S.\ s \xrightarrow{a} s' \wedge s' \in \|\phi_1\|^{\mathfrak{T}}_{\mathcal{V}}\}$$

The case for the box operator $[a]$ is similar. Now, notice that the new game rules and winning conditions enforced by these restrictions coincide with the ones defined by Stirling for the local model-checking games of $\mathcal{L}_\mu$ over interleaving models. In particular, the new game rules and winning conditions for the modalities are as follows.

In a finite play $C_0, C_1, ..., C_n$ of $\mathcal{G}(H_0, \phi)$, where C_n has a modality as a formula component, Adam wins if, and only if, $C_n = s \vdash \langle a \rangle \psi$ and $\{s' : s \xrightarrow{a} s'\} = \emptyset$, and Eve wins iff $C_n = s \vdash [a]\psi$ and $\{s' : s \xrightarrow{a} s'\} = \emptyset$. Since winning conditions for infinite plays do not depend on modalities, they remain the same. Furthermore, the game rules for modal operators reduce to:

$$(\langle\,\rangle) \quad \frac{s \vdash \langle a \rangle \phi}{s' \vdash \phi} \ [\exists] a : \ s \xrightarrow{a} s' \qquad\qquad ([\,]) \quad \frac{s \vdash [a]\,\phi}{s' \vdash \phi} \ [\forall] a : \ s \xrightarrow{a} s'$$

Clearly, the model-checking games just defined are equivalent to the ones presented in [89]. As can be easily verified, the reason for this coincidence is that when a plain modality $\langle a \rangle \phi$ (resp. $[a]\,\phi$) is encountered while playing the game, only Eve (resp. Adam) gets to choose both the next subformula and the transition used to verify (resp. falsify) the truth value of ϕ.

Now, let us look at the case when a model with an empty independence relation is considered. In such a case the rules $([\,]_{nc})$ and $(\bowtie)$ become trivially true and $(\langle\,\rangle_{nc})$ and $(*)$ trivially false since in an interleaving model all pairs of transitions are in $\leq$. For these reasons the elements that belong to the sets $\mathfrak{P}$ and $\mathfrak{A}$ need no longer be considered and the rules $([\,]_c)$ and $(\langle\,\rangle_c)$ become $([\,])$ and $(\langle\,\rangle)$, respectively. The other rules remain the same. □

4.2.4 Model-Checking Infinite Posets

In this section we use trace LMSO model-checking games to push forward the decidability border of the local model-checking problem of a particular class of partial order models, namely, of a class of infinite and regular event structures [72, 94]. More precisely, we improve previous results in the literature [56, 77] in terms of temporal expressive power.

SFL on Regular Trace Event Structures. As shown in previous sections, trace LMSO model-checking games can be played in either finite or infinite state systems (with finite branching). However, decidability for the games was proved only for finite systems. Therefore, if the system at hand has recursive behaviour and, moreover, is represented by an event structure, then the TSI representation of it may be infinite, and decidability is not guaranteed.

We now analyse the decidability of trace LMSO model-checking games for a special class of infinite, but regular, event structures called 'regular trace event structures'. This class of systems was introduced by Thiagarajan [94] in order to give a canonical representation to the set of Mazurkiewicz traces modelling the behaviour of a finite concurrent system. The model-checking problem for this class of models has been studied independently by Madhusudan [56] and by Penczek [77], and shown to be rather difficult. In the reminder of this section it is shown that model-checking SFL properties of this kind of systems is also decidable.

The first step to do so is to restrict ourselves to concurrent systems which are labelled with so-called Mazurkiewicz 'trace alphabets'. Such alphabets are usual sets of labels with a built in independence relation on their elements. Let us now introduce Mazurkiewicz trace alphabets as well as some classes of (concurrent) systems with such a sort of set of labels. The original definitions can all be found in [94], in some cases, with a slightly different notation.

Definition 4.39. (Trace alphabets) A Mazurkiewicz *trace alphabet* is a pair (Σ_{DR}, I_{DR}) where Σ_{DR} is a finite alphabet set and $I_{DR} \subseteq \Sigma_{DR} \times \Sigma_{DR}$ is an irreflexive and symmetric relation. $\triangleleft$

Now, recall the definition of event structures given in Chapter 2 and based on that, in the remainder of this section, consider the following notations and abbreviations:

Notation 4.40. (Configuration states and minimal relations) The 'configuration states' of a (labelled prime) event structure $\mathfrak{E}_{ES} = (E, \preccurlyeq, \sharp, \eta, \Sigma)$ will be denoted by $Conf_{ES}$. Moreover, for two events $e, e' \in E$, write $e \sharp_{\mathrm{m}} e'$ iff $((\{v \in E \mid v \preccurlyeq e\} \times \{u \in E \mid u \preccurlyeq e'\}) \cap \sharp) = \{(e, e')\}$, and call $\sharp_{\mathrm{m}}$ the 'minimal conflict' relation; and, write $e \lessdot e'$ iff both $e \preccurlyeq e'$ and for all $e'' \in E$ such that $e \preccurlyeq e'' \preccurlyeq e'$ either $e = e''$ or $e' = e''$, and call $\lessdot$ the 'minimal conflict' relation. $\triangleleft$

Based on the definition of trace alphabets one can define classes of systems (event structures in this case) where the independence relation of the trace alphabet can be associated with the independence relation of the systems themselves. As noted by Thiagarajan this can be done through the use of the minimal conflict and causality relations in the following way:

Definition 4.41. (Trace event structures) A *trace event structure* over a Mazurkiewicz trace alphabet (Σ_{DR}, I_{DR}) is an event structure $\mathfrak{E}_{ES} = (E, \preccurlyeq, \sharp, \eta, \Sigma_{DR})$ that satisfies the following:

1. If $e \sharp_{\mathrm{m}} e'$, then $\eta(e) \neq \eta(e')$.
2. If $e \lessdot e'$ or $e \sharp_{\mathrm{m}} e'$, then $(\eta(e), \eta(e')) \notin I_{DR}$.
3. If $(\eta(e), \eta(e')) \notin I_{DR}$, then $e \preccurlyeq e'$ or $e' \preccurlyeq e$ or $e \sharp e'$ $\triangleleft$

Then, a trace event structure is an event structure in which the pattern of labelling, which is given by η respects the independence relation I_{DR}. In fact such patterns of labelling must respect the independence relation co of the underlying (labelled prime) event structure.

Definition 4.42. (Subtrace event structures) Let $\mathfrak{E}_{ES} = (E, \preccurlyeq, \sharp, \eta, \Sigma_{DR})$ be a trace event structure over a trace alphabet (Σ_{DR}, I_{DR}), $C \in \mathit{Conf}_{ES}$, and $\natural(C) = \{e' \in E \mid \exists e \in C.e \sharp e'\}$. The *subtrace event structure* $\mathfrak{E}_{ES} \Uparrow C$ is the trace event structure $\mathfrak{E}'_{ES} = (E', \preccurlyeq', \sharp', \eta', \Sigma'_{DR})$ where:

- $E' = E \setminus (\{C\} \cup \natural(C))$.
- $\preccurlyeq'$ and $\sharp'$ are, respectively, $\preccurlyeq$ and $\sharp$ both restricted to $E' \times E'$.
- η' and Σ'_{DR} are, respectively, η and Σ_{DR} both restricted to E'. ◁

Since a subtrace event structure is also a trace event structure [94], then hereafter we will not make any distinctions between subtrace and trace event structures. Informally, the reader may think of a trace event structure $\mathfrak{E}_{ES} \Uparrow C$ as the trace event structure built from $\mathfrak{E}_{ES}$ and rooted at the configuration state $C \in \mathit{Conf}_{ES}$.

Definition 4.43. (Σ-labelled trace event structures) A Σ-labelled trace event structure is a pair $\mathfrak{E}_{LES} = (\mathfrak{E}_{ES}, \eta_T)$ where $\mathfrak{E}_{ES} = (E, \preccurlyeq, \sharp, \eta, \Sigma_{DR})$ is a trace event structure over a Mazurkiewicz trace alphabet (Σ_{DR}, I_{DR}) and $\eta_T : E \to \Sigma$ is a labelling function. ◁

Note that a Σ-labelled trace event structure has two labelling functions, an "internal" one (η) which must respect the independence relation I_{DR} of the trace alphabet and an "external" one (η_T) which could be used in an unrestricted way. However, in our setting both labelling functions are restricted in the same way: to Mazurkiewicz trace alphabets—a restriction that is good enough for us to achieve the result we are looking for, this is, decidability of the trace LMSO model-checking problem for SFL specifications over regular trace event structures.

Definition 4.44. (Regular trace event structures) Let $\mathfrak{E}_{LES} = (\mathfrak{E}_{ES}, \eta_T)$ be a Σ-labelled trace event structure for which $\mathfrak{E}_{ES} = (E, \preccurlyeq, \sharp, \eta, \Sigma_{DR})$ is a trace event structure over a Mazurkiewicz trace alphabet (Σ_{DR}, I_{DR}); moreover, suppose that $C \in \mathit{Conf}_{ES}$. Then:

1. $\mathfrak{E}_{LES} \Uparrow C = (\mathfrak{E}'_{ES}, \eta'_T)$ where $\mathfrak{E}'_{ES} = \mathfrak{E}_{ES} \Uparrow C$ and η'_T is η_T restricted to E'.
2. The equivalence relation $\sim_{LES} \subseteq \mathit{Conf}_{ES} \times \mathit{Conf}_{ES}$ is given by:
   $C \sim_{LES} C'$ iff $\mathfrak{E}_{LES} \Uparrow C \equiv \mathfrak{E}_{LES} \Uparrow C'$, i.e., iff $\mathfrak{E}_{LES} \Uparrow C$ and $\mathfrak{E}_{LES} \Uparrow C'$ are isomorphic.
3. $\mathfrak{E}_{LES}$ is *regular* iff $\sim_{LES}$ is of finite index. ◁

In [94] Thiagarajan showed that a regular trace event structure can be given a finite net representation in the form of a safe Petri net (when restricted to Mazurkiewicz trace alphabets). Using such a construction, and the canonical map from Petri nets to TSI models presented in Chapter 2, one can effectively construct a finite TSI model and therefore a finite board where a trace LMSO model-checking game can be played. The previous simple observation leads us to the following result, which is an immediate consequence of Theorem 4.36:[7]

Corollary 4.45. *Model-checking regular trace event structures against Separation Fixpoint Logic specifications is decidable.*

[7] I thank the reviewers of this PhD thesis for suggesting the use of Thiagarajan's construction.

4.3 Summary

In this chapter we have studied (infinite) higher-order logic games for bisimulation and for model-checking, where the players of the games are given (local) monadic second-order power on the sets of elements they are allowed to play. In both cases, these logic games were shown to be sound and complete (and therefore determined) and, even, to admit history-free winning strategies despite their higher-order power.

From a more practical standpoint, the games were shown to be decidable in the finite case and to underpin novel decision procedures for bisimulation and model-checking. These logic games, and associated decision procedures, were used to extend, respectively, the decidability border of a hierarchy of bisimulation equivalences for true concurrency as well as the temporal verification capabilities over a class of infinite and regular event structures.

These higher-order logic games were also studied when restricted to an interleaving setting (the first-order case) and two coincidence results were found. In the case of bisimulation, they are equivalent to the game for Milner and Park's strong bisimilarity, whereas in the case of model-checking, they become the local model-checking games of Stirling for $\mathcal{L}_\mu$. These results show, once again, that the techniques presented here extend conservatively, to a partial order setting, those for interleaving concurrency.

Chapter 5

Concurrent Logic Games

Apart from the applications to verification (as extensively discussed in this thesis so far), logic games have been used for semantic purposes as well. In particular, in the last 20 years, these so-called 'semantic games' have been used for giving 'fully complete' models of logic systems as well as 'fully abstract' denotational semantics of various programming languages. Due to their mathematical properties, *semantic* games are regarded as very precise models of interaction.

Interestingly, in the last decade, a number of works have shown that some semantic games can be used to define logic games for verification. One of such lines of work was initiated and has been further developed by Abramsky and Ong, together with collaborators within the game semantics community, as a semantic approach to systems verification. This line of work has provided a very powerful transfer of technology from semantic to verification games.

Albeit new, this approach to verification has already given very positive results, mainly due to the compositionality property that comes with the denotational (game) models on which the technique is based. For instance, (sequential) semantic games have been used to define game-based verification techniques for both imperative and functional sequential programming languages as well as for concurrent programs and systems with *interleaving* semantics.

However, as already discussed in previous chapters, it is a well-known fact in systems verification that interleaving semantics of concurrent systems are highly combinatorial and therefore tend to produce *non-local*, *monolithic*, *large* models which may be difficult to check in practice. This fact poses a serious problem both for systems verification itself as well as for the use of (sequential) semantic games in the verification of concurrent systems.

But, as described earlier in Chapter 1, due to this combinatorial problem when using interleaving representations of concurrent systems, *partial order* semantics have been considered rather than interleaving ones in the context of systems verification. It is then natural to wonder whether *concurrent* semantic games played on partial order models can be used for defining *concurrent* verification games as has already been successfully done for concurrent systems with interleaving semantics. This is the main motivation and starting point of this chapter.

The main problem when trying to define a concurrent verification game from a semantic one in a partial order framework is that concurrent verification games played directly on concurrent systems with partial order semantics (such as Petri nets or event structures) are not known to be *determined* in the general case since they may well be of *imperfect* information, chiefly, due to the information about *locality* and *independence* in such partial order models.

However, we have found that the process of defining sound and complete concurrent games (which must be determined) played on different partial order models of concurrency can be significantly simplified if one moves to a partial order setting in which the players of the game are allowed to make choices *concurrently* and *asynchronously* only in some suitably chosen independent localities of the partially ordered structures where the game is played.

This observation allows us to develop a very general theory intended to be uniformly applicable to a wide range of problems and systems. Since games in this chapter are played on partial orders the technique we put forward here lends itself particularly useful for the verification of concurrent systems with partial order semantics. Nonetheless, games on some interleaving models appear here as particular cases. Indeed, our results generalize those presented in [90], namely the (sequential) games for bisimulation and mu-calculus model-checking of Stirling.

Our framework builds upon two main ideas: firstly, the use of posets to give a *uniform* representation of concurrent systems, logical specifications, and problem descriptions; and secondly, the restriction to games with a *semantic* condition that reduces reasoning on different models and decision problems to the analysis of simpler *local* correctness conditions. These features make considerably easier the analysis of different decision problems and concurrent systems by allowing one to abstract away from particularities of the concrete classes of systems and problems. Formally, this is achieved by a number of metatheorems that can be parameterized and reused in order to provide "off-the-shelf" solutions to different problems.

The proof method is realised by a new 'concurrent logic game' (CLG) which is shown to be determined—even though it is, locally, of imperfect information. Moreover, the elements of the game are all formalised in order-theoretic terms; as a result, this new model builds a bridge between some concepts in order theory and the more operational world of games. To the best of my knowledge, such an order-theoretic characterisation has not been previously investigated for verification games. The formal definition of the CLG model together with the metatheorems for soundness and completeness comprises the first part of this chapter.

Then, in the second half, two algorithmic applications are shown to be cast within this unified approach: *bisimulation* and *model-checking*. The reductions from bisimulation and model-checking to a CLG problem make use of posets extracted from the partially ordered structures obtained when using McMillan's *unfolding* method [63]. But, as the inputs of the decision procedures are posets rather than the concrete systems themselves, different models of concurrency can be treated in the same manner, even some interleaving ones in a trivial way.

Then, the main contribution of this chapter is the formalization of a concurrent logic game model that generalises the results in [90] to a partial order setting—this is, the games of Stirling for bisimulation and model-checking on interleaving structures (and therefore also related tableau-based techniques). The CLG model is inspired by a concurrent semantic game model (of a fragment of Linear Logic [34]) studied by Abramsky and Melliès [1]. However, the mathematics of the original semantic game model have been drastically reformulated (in the quest towards to answer of algorithmic questions), and only a few technical features were kept.

In particular, the results in this chapter provide answers for the following two questions: 'is this concurrent logic game *determined*?'; and if so, 'is the winner of such a game *computable* (and under which conditions)?'. As we are looking for winning strategies with finite poset representations, such strategies may be considerably smaller than those for games on interleaving structures, and hence attractive from a synthesis viewpoint. Let us now introduce some concepts and notations before presenting the mathematical formalization of the CLG model.

Preliminaries on Posets and Closure Operators Revisited

A '$\bot_{\mathcal{A}}$-bounded poset' $\mathfrak{A} = (\mathcal{A}, \leq_{\mathcal{A}})$ is a poset with a bottom element $\bot_{\mathcal{A}}$ such that for all $a \in \mathcal{A}$ we have that $\bot_{\mathcal{A}} \leq_{\mathcal{A}} a$; we may omit the subscript in $\bot$ whenever clear from the context. For any $a \in \mathcal{A}$, an 'immediate successor' of a (hereafter simply called a successor of a) is an element a' such that $a <_{\mathcal{A}} a'$ and for all b if $a \leq_{\mathcal{A}} b$ and $b \leq_{\mathcal{A}} a'$ then either $a = b$ or $b = a'$. Write $a \rightarrow a'$ iff a' is a successor of a and call a a 'terminal element' of $\mathfrak{A}$ iff $a \not\rightarrow$. A 'chain' B of $\mathfrak{A}$ is a totally ordered subset of $\mathcal{A}$ such that if $a, b \in B$ and $c \in \mathcal{A}$ and $a < c < b$, then $c \in B$.

Given a, a '(principal) ideal' $\downarrow a$ is the downward-closed set $\{b \in \mathcal{A} \mid b \leq_{\mathcal{A}} a\}$; dually, a '(principal) filter' $\uparrow a$ of $\mathcal{A}$ is the upward-closed $\{b \in \mathcal{A} \mid a \leq_{\mathcal{A}} b\}$. Also, for any set $A \subseteq \mathcal{A}$, write $\downarrow A$ for the set $\bigcup_{a \in A}\{b \mid b \in \downarrow a\}$, and likewise, $\uparrow A$ for $\bigcup_{a \in A}\{b \mid b \in \uparrow a\}$; call $\downarrow A$ a 'lower subset' and $\uparrow A$ an 'upper subset'. We write $\downarrow a$ for the induced poset $(\downarrow a, \leq_{\mathcal{A}})$, and similarly for $\uparrow a$, $\downarrow A$, and $\uparrow A$. Clearly the posets $\downarrow a$ and $\uparrow a$ are $\bot$-bounded if $\mathfrak{A}$ is $\bot$-bounded, since $\bot_{\downarrow a} = \bot_{\mathcal{A}}$ in the former case and $\bot_{\uparrow a} = a$ in the latter. Finally, recall (from Chapter 3) that a function $f : \mathcal{A} \rightarrow \mathcal{A}$ is a 'closure operator' iff it is extensive, monotonic, and idempotent—i.e., iff f satisfies that for all $a, a' \in \mathcal{A}$: $a \leq_{\mathcal{A}} f(a)$; $a \leq_{\mathcal{A}} a' \Rightarrow f(a) \leq_{\mathcal{A}} f(a')$; and $f(a) = f(f(a))$.

5.1 Concurrent Games on Partial Orders

As presented in previous chapters a logic game for verification is played by two players, the Verifier called Eve ($\exists$) and the Falsifier called Adam ($\forall$), in order to show the truth or falsity of a given property. In these games Eve tries to show that the property holds, whereas Adam wants to refute such an assertion. In traditional settings, the game is played sequentially in a board represented by a graph structure, where each node belongs to one of the players.

As briefly mentioned at the beginning of this chapter, we have found that by enriching a logic game with the explicit information about local and independent behaviour that comes with any partial order model, the sequential setting for logic games can be turned into a concurrent one on a partial order. This framework is simple and general enough to embody different verification problems uniformly—for several partial order models. In the remainder of this section we define the *structure* and *dynamics* of a 'concurrent logic game' (CLG) played on a partially ordered structure, and present some algorithmically useful properties of it.

5.1.1 Structure and Dynamics

Game Boards. A board in a CLG is a $\bot$-bounded poset $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ which is well-founded. A lower (resp. an upper) sub-board $\mathfrak{B}$ of $\mathfrak{D}$ is a poset $(\mathcal{B}, \leq_{\mathcal{D}})$ such that $\mathcal{B}$ is a lower (resp. an upper) subset of $\mathcal{D}$. Then, a lower sub-board is always a $\bot$-bounded poset, whereas an upper sub-board is a union of possibly infinitely large $\bot$-bounded posets. In particular, since $\mathfrak{D}$ is well-founded, then all lower sub-boards are also well-founded. We only consider posets where every chain has a maximal element. Moreover, a 'global position' in $\mathfrak{D}$ is an anti-chain $D \subseteq \mathcal{D}$; the initial global position is $\{\bot\}$. Given a global position D, call any $d \in D$ a 'local position'.

Notation 5.1. Given any $d \in \mathcal{D}$, write $d^{\leftarrow}$ for the set of local positions $\{e \mid e \rightarrow d\}$ and $d^{\rightarrow}$ for the set $\{d' \mid d \rightarrow d'\}$. The sets $d^{\leftarrow}$ and $d^{\rightarrow}$ are the 'preset' and 'postset' of local positions of d. Moreover, let $\mathtt{SP}(d)$ be the predicate that evaluates to true if, and only if, d is a 'synchronization point', which formally means iff $|d^{\leftarrow}| > 1$, or evaluates to false otherwise. $\triangleleft$

Now, let $\nabla : \mathcal{D} \rightarrow \Upsilon$ be a partial function that assigns players in $\Upsilon = \{\exists, \forall\}$ to local positions. More precisely, ∇ is a total function on the set $\mathcal{B} \subseteq \mathcal{D}$ that contains all elements which are not synchronization points—i.e., $\mathcal{B} = \{d \in \mathcal{D} \mid \neg\mathtt{SP}(d)\}$; call the pair $(\mathfrak{D}, \nabla)$ a 'polarised board'.[1] In the following we only consider polarised boards whose synchronization points have the following uniqueness property: if $\mathtt{SP}(d)$ then $|d^{\rightarrow}| = 1$ and $\forall e \in d^{\leftarrow}.|e^{\rightarrow}| = 1$.

This property will induce a correctness condition when playing the game. It ensures: firstly, that there are no choices to make in synchronization points (because they are not assigned by ∇ to any player); and secondly, that a synchronization point does not share any element of its present with any other local position, and therefore, concurrent and local behaviour in the game—which is defined later on—can be made truly independent.

The distinction between local and global positions will be used to define, respectively, local and global strategies for the game, which in turn will allow that in a global position both players make independent local choices (in local positions), leading to a joint global move of the game; thus, globally, one may think of the players as acting simultaneously on $\mathfrak{D}$.

[1] Clearly, by definition, for any local position d there exists a global position $\{d\}$. Yet, it is important to make clear that they are different since any local position that is not a synchronization point has a 'polarity', which can be either $\exists$ or $\forall$, but this is in general not true for global positions.

Strategies. In a CLG a strategy can be local or global. A 'local strategy' $\lambda : \mathcal{D} \to \mathcal{D}$ is a closure operator partially defined on a board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$. Being partially defined means that the properties of closure operators are restricted to elements where the closure operator is defined. In particular, $a \leq_{\mathcal{D}} a'$ implies $\lambda(a) \leq_{\mathcal{D}} \lambda(a')$ holds iff λ is defined both in a and in a'. Let $\mathtt{dfn}(\lambda, d)$ be the predicate that holds iff $\lambda(d)$ is defined or evaluates to false otherwise. The reader may think of a local strategy as a function that tells a player how to make a move at some local positions, independently of the behaviour in other local positions.

The predicate dfn can be defined from a board, for any local strategy, by means of three rules that realise local strategies $\lambda_\forall$ and $\lambda_\exists$ for Adam and Eve, respectively.

Definition 5.2. (Local strategies) Given a board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$, a local strategy $\lambda_\forall$ for Adam (resp. $\lambda_\exists$ for Eve) is a closure operator defined only in those elements of $\mathcal{D}$ given by the following rules:

1. The local strategy $\lambda_\forall$ (resp. $\lambda_\exists$) is defined in the bottom element $\bot_{\mathcal{D}}$.
2. If a local strategy $\lambda_\forall$ (resp. $\lambda_\exists$) is defined in a local position $d \in \mathcal{D}$, and either $\nabla(d) = \exists$ (resp. $\nabla(d) = \forall$) or $\mathtt{SP}(d)$ or $d \not\to$ or $d \to e \wedge \mathtt{SP}(e)$, then for all $d' \in d^{\rightarrow}$ we also have that $\lambda_\forall$ (resp. $\lambda_\exists$) is defined in d'.
3. If a local strategy $\lambda_\forall$ (resp. $\lambda_\exists$) is defined in $d \in \mathcal{D}$, and both $\nabla(d) = \forall$ (resp. $\nabla(d) = \exists$) and $|d^{\rightarrow}| \geq 1$, then there exists only one $d' \in d^{\rightarrow}$ in which $\lambda_\forall$ (resp. $\lambda_\exists$) is defined.

And let $\mathtt{dfn}(\lambda_\forall, d)$ be the predicate that holds whenever $\lambda_\forall(d)$ is defined, and similarly for $\lambda_\exists$. Moreover, the closed elements, i.e., the fixpoints, of $\lambda_\forall$ and $\lambda_\exists$ are as follows:

$$\lambda_\forall(d) = d \quad \text{iff} \quad \mathtt{dfn}(\lambda_\forall, d) \text{ and } (\ \nabla(d) = \exists, \text{ or } \mathtt{SP}(d), \text{ or } d \not\to, \text{ or } (d \to e \wedge \mathtt{SP}(e))\ )$$
$$\lambda_\exists(d) = d \quad \text{iff} \quad \mathtt{dfn}(\lambda_\exists, d) \text{ and } (\ \nabla(d) = \forall, \text{ or } \mathtt{SP}(d), \text{ or } d \not\to, \text{ or } (d \to e \wedge \mathtt{SP}(e))\ )$$

Moreover, let $\lambda^1_\forall$ and $\lambda^1_\exists$ be the identity local strategies of Adam and Eve, respectively, which are defined everywhere in $\mathfrak{D}$; thus, formally: $\lambda^1_\forall(d) = \lambda^1_\exists(d) = d$, for all $d \in \mathcal{D}$. ◁

Let $\Lambda_{\mathfrak{D}}$ be the set of local strategies on $\mathfrak{D}$. Since a logic game is played by two players, then the set of local strategies can be split into two sets of local strategies. Let $\Lambda_{\mathfrak{D}} = \Lambda^\exists_{\mathfrak{D}} \uplus \Lambda^\forall_{\mathfrak{D}}$, where $\Lambda^\exists_{\mathfrak{D}}$ is the set of local strategies of Eve and $\Lambda^\forall_{\mathfrak{D}}$ is the one of Adam. Due to the rules for realising local strategies given before, we can assume that for each chain of $\mathfrak{D}$ there is at least one local strategy, for each player, that is defined in all elements of such a chain.

Notation 5.3. Write $\mathtt{fix}_\forall$ and $\mathtt{fix}_\exists$ for the predicates characterising the fixpoints of the local strategies for Adam and Eve, respectively, in the following way:

$$\mathtt{fix}_\forall(\lambda_\forall, d) \stackrel{\text{def}}{=} \mathtt{dfn}(\lambda_\forall, d) \text{ and } (\ \nabla(d) = \exists, \text{ or } \mathtt{SP}(d), \text{ or } d \not\to, \text{ or } d \to e \wedge \mathtt{SP}(e)\ )$$
$$\mathtt{fix}_\exists(\lambda_\exists, d) \stackrel{\text{def}}{=} \mathtt{dfn}(\lambda_\exists, d) \text{ and } (\ \nabla(d) = \forall, \text{ or } \mathtt{SP}(d), \text{ or } d \not\to, \text{ or } d \to e \wedge \mathtt{SP}(e)\ )$$

where $d \in \mathcal{D}$, $\lambda_\forall \in \Lambda^\forall_{\mathfrak{D}}$, and $\lambda_\exists \in \Lambda^\exists_{\mathfrak{D}}$ in a game board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$. ◁

Informally, this means that, provided that a local strategy is defined at d, all local positions where Eve is to make a move are *fixpoints* of the local strategies of Adam defined at those positions as well, and vice versa, i.e., when is the other player's turn to play. Terminal elements and synchronization points of the game board also are fixpoints of the local strategies of both players since in such cases, respectively, there is no next local position to move to and it is up to the "environment"—whose behaviour is determined by the dynamics of the game[2]—to make such a move. In addition, for the same last reason just given, all the elements in the preset of a synchronization point are also fixpoints of all local strategies for both players.

Let us now move closer to the definition of the dynamics of the game. When playing a game, Eve and Adam will use a set of local strategies $\Lambda^{\exists}_{\mathfrak{O}} \subseteq \Lambda^{\exists}_{\mathfrak{D}}$ and $\Lambda^{\forall}_{\mathfrak{O}} \subseteq \Lambda^{\forall}_{\mathfrak{D}}$, whose elements (the local strategies) can be indexed by the elements i and j of the two sets $\mathbb{K}_{\exists} = \{1, ..., |\Lambda^{\exists}_{\mathfrak{D}}|\}$ and $\mathbb{K}_{\forall} = \{1, ..., |\Lambda^{\forall}_{\mathfrak{D}}|\}$, respectively; by definition, the identity local strategies are always indexed with $i = 1$ and $j = 1$, as already presented in the definition of local strategies.

Now, suppose that at the beginning of the game Eve and Adam choose, independently and at the same time, two sets of indices $\mathrm{K}_{\exists} \subseteq \mathbb{K}_{\exists}$ and $\mathrm{K}_{\forall} \subseteq \mathbb{K}_{\forall}$, and consequently the two sets of local strategies $\Lambda^{\exists}_{\mathfrak{O}} \subseteq \Lambda^{\exists}_{\mathfrak{D}}$ and $\Lambda^{\forall}_{\mathfrak{O}} \subseteq \Lambda^{\forall}_{\mathfrak{D}}$ they are going to use to play the game; this means that both $i \in \mathrm{K}_{\exists}$ iff $\lambda^{i}_{\exists} \in \Lambda^{\exists}_{\mathfrak{O}}$ and $j \in \mathrm{K}_{\forall}$ iff $\lambda^{j}_{\forall} \in \Lambda^{\forall}_{\mathfrak{O}}$; by definition, the identity local strategies $\lambda^{1}_{\exists}$ and $\lambda^{1}_{\forall}$ are always included in $\Lambda^{\exists}_{\mathfrak{O}}$ and $\Lambda^{\forall}_{\mathfrak{O}}$. Based on this initial selection of local strategies one can define the global strategies and global moves for a concurrent logic game, as well as the set of reachable global positions in a particular game.

But, first, let us define the ordered structure where global strategies are interpreted, namely the poset induced by the subset order inclusion on the lower sets of $\mathfrak{D}$, or more precisely, on the lower sets characterised by the anti-chains of $\mathfrak{D}$. The reason why this is the structure where global strategies are interpreted is that, by definition, a global position is an anti-chain of $\mathfrak{D}$. We will define $\mathbb{A} = (\mathcal{A}, \leq_{\mathcal{A}})$ to be such a suitable poset (a space of anti-chains) and call it the 'arena of global positions' of $\mathfrak{D}$, which is formally defined as follows:

Definition 5.4. (Arena of global positions) Given a board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$, let the poset $\mathbb{A} = (\mathcal{A}, \leq_{\mathcal{A}})$ be its arena of global positions, where $\mathcal{A}$ is the set of anti-chains of $\mathfrak{D}$ and $E \leq_{\mathcal{A}} D$ iff $\downarrow E \subseteq \downarrow D$, for all anti-chains of $\mathfrak{D}$. ◁

And, moreover, write `max` for the 'maximal elements' set operation, which is defined as usual: given a poset $\mathbb{P} = (\mathcal{P}, \leq_{\mathcal{P}})$, the set $\mathtt{max}\ \mathbb{P}$ is the anti-chain of maximal elements of $\mathbb{P}$, i.e., $\mathtt{max}\ \mathbb{P} = \{m \in \mathcal{P} \mid \neg\exists n \in \mathcal{P}. m <_{\mathcal{P}} n\}$. Then, finally we have:

Definition 5.5. (Global strategies) Let $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ be a game board. Given two subsets of indices $\mathrm{K}_{\forall}$ of $\mathbb{K}_{\forall}$ and $\mathrm{K}_{\exists}$ of $\mathbb{K}_{\exists}$, and hence, two sets of local strategies $\Lambda^{\forall}_{\mathfrak{O}}$ and $\Lambda^{\exists}_{\mathfrak{O}}$ for Adam

[2]The behaviour of the "environment", which is deterministic, and hence the behaviour in local positions where $\mathtt{SP}(d)$ holds is formally defined later on when the dynamics of the game is presented.

and Eve, let the closure operators $\partial_\forall : \mathcal{A} \to \mathcal{A}$ and $\partial_\exists : \mathcal{A} \to \mathcal{A}$ on the poset $\mathbb{A} = (\mathcal{A}, \leq_{\mathcal{A}})$, where $\mathbb{A}$ is the arena of global positions associated with $\mathfrak{D}$, be the global strategies for Adam and Eve defined as follows:

$$\begin{array}{rcl} \partial_\forall(D) & \stackrel{\text{def}}{=} & \texttt{max} \bigcup_{d \in D, j \in \mathrm{K}_\forall} \{\lambda^j_\forall(d) \mid \texttt{dfn}(\lambda^j_\forall, d)\} \\ \partial_\exists(D) & \stackrel{\text{def}}{=} & \texttt{max} \bigcup_{d \in D, i \in \mathrm{K}_\exists} \{\lambda^i_\exists(d) \mid \texttt{dfn}(\lambda^i_\exists, d)\} \end{array}$$

where $D \subseteq \mathcal{D}$ is a global position of $\mathfrak{D}$, and therefore an element of $\mathcal{A}$. ◁

It should be easy to see that $\partial_\forall$ and $\partial_\exists$ are closure operators indeed. Firstly, they are *extensive* because the identity local strategies $\lambda^1_\forall$ and $\lambda^1_\exists$ ensure that both $\partial_\forall(D)$ and $\partial_\exists(D)$ are at least D; secondly, they are *monotonic* because all $\lambda^j_\forall$ and $\lambda^i_\exists$ are monotonic as well; and finally, they are *idempotent* because `max` chooses only maximal elements, which are already the fixpoints of idempotent functions—the local strategies that characterise the global strategies $\partial_\forall$ and $\partial_\exists$. Nonetheless, notice that whereas a local strategy is partially defined on a board $\mathfrak{D}$, a global strategy is totally defined in the arena of global positions $\mathbb{A}$ associated with $\mathfrak{D}$.

Remark 5.6. (Strategies as closure operators) The intuitions as to why a closure operator captures the behaviour in a CLG follow those in [1]. Note that as boards are posets, then the game is played on *acyclic* structures and therefore there is no reason for a player to make a move to a previous position since this will result in playing again part of the game that has been already played. Then, strategies should be extensive functions. They should also be monotonic so as to preserve the *causality* of the moves or choices made by the players in the game. Finally, it is also desirable for a strategy to be idempotent since this avoids the need for *unnecessary* sequential steps and alternations between the two players, which obscure the dynamics of the game, and hide the really interesting points of interaction between the two players. ◁

Now, the dynamics of a game is captured by the interaction between the two players (together with an external, deterministic environment). This interaction is given by how the strategies $\partial_\forall$ and $\partial_\exists$ are played against each other. The most natural way to do so is by defining such an interaction as their (functional) composition.

Definition 5.7. (Rounds and composition of strategies) Let a $(\exists \circ \forall)$-*round* be a global step of the game such that if $D \subseteq \mathcal{D}$ is the current global position of the game, $\partial_\exists$ is the strategy of Eve, and $\partial_\forall$ is the strategy of Adam, then the game proceeds first to an intermediate global position $D_{\exists\circ\forall}$ such that:

$$\begin{array}{rcl} D_{\exists\circ\forall} & = & (\partial_\exists \circ \partial_\forall)(D) \\ & = & \texttt{max} \bigcup_{d \in D, i \in \mathrm{K}_\exists, j \in \mathrm{K}_\forall} \{(\lambda^i_\exists \circ \lambda^j_\forall)(d) \mid \texttt{dfn}(\lambda^j_\forall, d) \wedge \texttt{dfn}(\lambda^i_\exists, \lambda^j_\forall(d))\} \end{array}$$

and then to the *next* global position D' given by the interference of the environment:

$$D' = (D_{\exists\circ\forall} \setminus e^{\leftarrow}_{\mathtt{SP}}) \bigcup e^{\rightarrow}_{\mathtt{SP}}$$

where

$$e^{\leftarrow}_{\mathtt{SP}} = \bigcup_{e \in D^{\rightarrow}_{\exists \circ \forall}} \{u \in e^{\leftarrow} \mid \mathtt{SP}(e) \wedge e^{\leftarrow} \subseteq D_{\exists \circ \forall}\}$$
$$e^{\rightarrow}_{\mathtt{SP}} = \bigcup_{e \in D^{\rightarrow}_{\exists \circ \forall}} \{v \in e^{\rightarrow} \mid \mathtt{SP}(e) \wedge e^{\leftarrow} \subseteq D_{\exists \circ \forall}\}$$

and call the transition from the global position D to D' a $\supset$-*round* of the game. $\triangleleft$

This definition of interaction of the strategies of Eve and Adam follows the intuition that in a verification game, Eve must respond to any possible move of Adam; moreover, if the game happens to be concurrent, she has to do so in every local position where Adam plays. The reader acquainted with net theory may have noticed that the transition $D \rightarrow D'$ between global positions is similar to that of markings—the so-called 'token game'—in Petri nets.

It is also worth saying that, logically, this behaviour is similar to that of the Henkin quantifier [10, 45], e.g., of the (game for a) Henkin modal formula $\begin{smallmatrix}[\,]\langle\,\rangle \\ [\,]\langle\,\rangle\end{smallmatrix}\phi$, and different from the behaviour of the (game for an) ATL quantifier [5, 10], say of the ATL formula $\begin{smallmatrix}[\,][\,] \\ \langle\,\rangle\langle\,\rangle\end{smallmatrix}\phi$ (this notation for ATL formulae follows that for modal quantifiers in [10]).

Plays. The interaction between the strategies of Eve and Adam define a sequence of global positions $\{\bot\}, D_1, ..., D_k, ...$, and hence, a sequence of posets given by the union of the order ideals (the lower subsets) determined by each D_k. As usual, the set of plays of a game is determined once the board where the game is played is given and the strategies are defined.

A play is any finite or infinite union of the elements of such posets. Formally, a play $\hbar = (\mathcal{H}, \leq_{\mathcal{D}})$ on a board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ is a (possibly infinitely large) poset such that $\mathcal{H}$ is a downward-closed subset of $\mathcal{D}$. We say that a play can be finite or infinite, and closed or open; more precisely, a play is:

1. *finite* iff all chains of $\hbar$ are finite;
2. *infinite* iff $\hbar$ has at least one infinite chain;
3. *closed* iff at least one of the terminal elements of $\mathfrak{D}$ is in $\mathcal{H}$;
4. *open* iff none of the terminal elements of $\mathfrak{D}$ is in $\mathcal{H}$;

this classification of plays is used in a further section to define in a concrete way what the 'winning sets' of a game are (which are abstractly defined below).[3]

As said before, a play can be of finite or infinite length, where the length of the play is the size of $\mathcal{H}$. Therefore, unlike games on interleaving structures, such as trees or graphs, the length of a play is an upper bound on the maximum number of rounds that has been played so far rather than the number of times that the players have made a move. This is a direct consequence of both the plays being partially ordered sets (because different chains can be

[3]Note that a play being finite or infinite is independent of being closed or open, since an infinitely long chain can either have or not a top element, which must necessarily be terminal.

independent and have different lengths) and the strategies being closure operators (as they allow for several sequential moves to be made at once).

Since for any play $\{\bot\}, D_1, ..., D_k, ...$ the lower subset defined by a global position D_k always includes the lower subsets of all other global positions D_j such that $j < k$, then in a partial order setting any global position D determines a play $\hbar_D = (\mathcal{H}, \leq_{\mathcal{D}})$ on a game board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ as follows:

$$\mathcal{H} = \bigcup\{e \in \downarrow d \mid d \in D\} = \bigcup_{d \in D}\{e \in \mathcal{D} \mid e \leq_{\mathcal{D}} d\};$$

finally, let Γ be the set of plays of a game.

Winning Sets and Strategies. Another element of a logic game is the set of *winning* conditions, which define the 'winning sets' for each player. The winning conditions are the rules that determine when a player has won a play. Let $\mathcal{W} : \Gamma \to \Upsilon$ be a *partial* function that assigns a winner $\exists, \forall \in \Upsilon$ to a play $\hbar \in \Gamma$, and call it the winning conditions of a game. The winning sets are determined by those plays containing at least one terminal element of the board as well as those representing infinite behaviour. Therefore, due to the classification of plays given before, one knows that $\mathcal{W}$ is not defined in plays that are both finite and open.

As will be made concrete in a further section, the particular kind of winning conditions, and therefore of winning sets that are formally defined for each logic game, determines the kind of verification game to be played, e.g., a (partial order) parity, reachability, safety, or bisimulation game, just to mention a few simple examples.

On the other hand, once strategies and winning conditions have been defined, a notion of a winning strategy can also be established. As for verification games on interleaving structures, a *winning* strategy is a global strategy which, if followed, guarantees that the player using that strategy will win *all* plays of the game. Given these elements, one finally has the following formulation of the elements which a concurrent logic game for verification is made of:

Definition 5.8. (Concurrent logic games) A tuple $\Im = (\Upsilon, \mathfrak{D}, \Lambda_{\mathfrak{D}}, \nabla, \mathcal{W}, \Gamma)$ is a concurrent logic game (CLG) model, where $\Upsilon = \{\exists, \forall\}$ is the set of players, the $\bot$-bounded poset $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ is a board, $\Lambda_{\mathfrak{D}} = \Lambda^{\exists}_{\mathfrak{D}} \uplus \Lambda^{\forall}_{\mathfrak{D}}$ are two disjoint sets of local strategies, $\nabla : \mathcal{D} \to \Upsilon$ is a partial function that assigns players to local positions, and $\mathcal{W} : \Gamma \to \Upsilon$ is a function defined by the winning conditions of $\Im$ over its set of plays Γ. $\lhd$

Once the elements of a CLG $\Im$ are defined, such a game is played in the following way: the two players, Eve and Adam, start by choosing, independently and at the same time, a set of local strategies, i.e., a global strategy for each of them. As mentioned before, the choice of local strategies is done indirectly by choosing the sets of indices $\mathbb{K}_{\forall}$ and $\mathbb{K}_{\exists}$, and recall that the identity local strategies $\lambda^{1}_{\forall}$ and $\lambda^{1}_{\exists}$ are always chosen.

The only other restriction when choosing the local strategies to be used in the game is, informally, that if the resulting global strategy ∂, for either player, plays some global position that contains a local position a and such a local position belongs to a chain m which eventually synchronizes with another chain n, then the global strategy ∂ must play an element b of the chain n as well. Put another way, this means that if two different chains "cooperate" (by synchronising with each other) and either Eve or Adam wants to play in one such chains, then he/she must play in both chains necessarily. Formally, the local strategies in $\Lambda^{\exists}_{\eth} \subseteq \Lambda^{\exists}_{\mathfrak{D}}$ and $\Lambda^{\forall}_{\eth} \subseteq \Lambda^{\forall}_{\mathfrak{D}}$ selected by Eve and Adam must ensure the following property (for Adam):

$$\forall d \in (\uparrow D \cap \downarrow \partial_{\forall}(D)), \text{ if } \mathtt{BP}(d) \wedge \nabla(d) = \forall \text{ then}$$
$$\forall a, b \in d^{\rightarrow}.\ \mathtt{sync}(a,b) \text{ implies } a, b \in \downarrow \partial_{\forall}(D).$$

and similarly for Eve—by changing $\partial_{\forall}$ for $\partial_{\exists}$ and $\nabla(d) = \forall$ for $\nabla(d) = \exists$. Also, the predicates BP and sync characterise, respectively, the 'branching points' of a poset and pairs of elements that belong to chains that synchronize, i.e., different chains that join at some point in the poset; these predicates are formally defined as follows:

$$\begin{array}{lll} \mathtt{BP}(d) & \text{iff} & |\, d^{\rightarrow} \,| > 1 \\ \mathtt{sync}(a,b) & \text{iff} & \uparrow a \cap \uparrow b \neq U \text{ for } U \in \{\emptyset, \uparrow a, \uparrow b\} \end{array}$$

After this stage of selection of local strategies, Eve and Adam play their global strategies against each other (possibly forever) until $\mathcal{W}$ determines that the play that has been generated by such an interaction is winning, if at all, for one of the players. There is no reason at this point to ensure that every play will always have a winner—or even that winning strategies always exist, i.e., that the game is determined; we postpone the study of this issue for a little longer.

5.1.2 Closure Properties

A concurrent logic game has some closure properties that are both mathematically and algorithmically useful when analysing its structure and dynamics: closure under dual properties (called closure under dual games), closure under lower sub-boards (called closure under ideals), and its order dual, closure under upper sub-boards (called closure under filters).

Firstly, as for some games on trees, CLG can be composed of subgames. The main difference is that since in a CLG two independent subgames may synchronise, then that case must be taken into account. Then, given a CLG $\eth$ played on a board $\mathfrak{D}$, let $\eth \Downarrow_{\mathfrak{B}}$ be the CLG defined from $\eth$ where $\mathfrak{B}$ is a sub-board of $\mathfrak{D}$ and the other components in $\eth$ are restricted to $\mathfrak{B}$.

But first, let us give a simple, though rather useful, technical lemma, which helps one ensure that in some special sub-boards a number of functionals (such as those defined by the strategies and winning conditions of the game) are preserved.

Lemma 5.9. (Unique poset prefixes) *Let D be a global position of a board $\mathfrak{D}$. There is a unique poset representing all plays containing D up to such position.*

Proof. By contradiction suppose that the global position D is reached and that there are two posets representing two different plays $\hbar_1$ and $\hbar_2$ with which this could be true. Due to the definition of global strategies this also means that there is at least one local position $d \in D$ for which there is more than one poset representing the way d was reached in the game. But due to the definition of local strategies and plays such a poset must be the order ideal $\downarrow d$, which is unique for any d; since this property holds for any $d \in D$, then the poset $\downarrow D$ is unique as well, and hence $\hbar_1$ and $\hbar_2$ must be the same. Hence we get a contradition. □

This result facilitates reasoning on games on posets. It implies that regardless of which strategies the players are using, if a position $D \subseteq \mathcal{D}$ appears in different plays, then the 'poset prefixes' of all such plays, up to the global position D, are isomorphic. This lemma is a direct consequence of the restriction imposed on games which states that if a synchronization point is played by the environment, then all previous elements in the poset (i.e., all smaller elements with respect to $\leq_{\mathcal{D}}$) must have been played already. So, this lemma is, technically, what justifies such a condition. Similarly, this lemma ensures that if $\mathcal{W}$ is a functional operator on the set of plays in the board $\mathfrak{D}$, then it is also a functional operator in the set of plays restricted to the new board $\mathfrak{B}$. So, now we can move on to studying some of the closure properties a CLG enjoys.

Lemma 5.10. (Closure under filters) *Let $\Im$ be a CLG and D a global position of the board $\mathfrak{D}$ associated with $\Im$. The structure $\Im \Downarrow_{\mathfrak{B}} = (\Upsilon, \bot \oplus \mathfrak{B}, \Lambda_{\mathfrak{B}}, \nabla \Downarrow_{\mathfrak{B}}, \mathcal{W} \Downarrow_{\mathfrak{B}}, \Gamma \Downarrow_{\mathfrak{B}})$ is also a CLG where $\mathfrak{B}$ is the upper sub-board of $\mathfrak{D}$ defined by D.*

Proof. The proof is by showing a correct construction of the concurrent logic game $\Im \Downarrow_{\mathfrak{B}} = (\Upsilon, \bot \oplus \mathfrak{B}, \Lambda_{\mathfrak{B}}, \nabla \Downarrow_{\mathfrak{B}}, \mathcal{W} \Downarrow_{\mathfrak{B}}, \Gamma \Downarrow_{\mathfrak{B}})$, i.e., that indeed $\Im \Downarrow_{\mathfrak{B}}$ defines a CLG. By definition, D is an anti-chain of local positions, and thus, we can construct the poset given by the union of filters $\uparrow d_i$ of all elements $d_i \in D$. A bottom element $\bot_{\uparrow D}$ of $\mathfrak{B}$ can be adjoined to all chains starting in every d_i using the linear sum operator $\oplus$ so as to construct the poset $\bot_{\uparrow D} \oplus \mathfrak{B}$; then, $\bot = \bot_{\uparrow D}$, and such a structure is a $\bot$-bounded board. Also, let $\nabla \Downarrow_{\mathfrak{B}}$ be as in $\Im$ and define $\nabla(\bot_{\uparrow D}) = \nabla(\bot_{\mathcal{D}})$, so as to preserve the polarity of the bottom element, and hence, who starts playing the game. Notice that the restriction on $\mathfrak{D}$ which states that if $\mathtt{SP}(d)$ then $\mid d^{\rightarrow} \mid = 1$ and $\forall e \in d^{\leftarrow}.\mid e^{\rightarrow} \mid = 1$, for all local positions d, is preserved. Also note that due to Lemma 5.9 if two chains containing two different d_i synchronise in $\mathfrak{D}$, then they also synchronise in $\bot \oplus \mathfrak{B}$, clearly, provided that such a synchronization point does not belong to $\downarrow D$.

Now, the new local strategies in $\Lambda_{\mathfrak{B}}$ are defined from the new $\bot$-bounded, polarised board $(\bot \oplus \mathfrak{B}, \nabla \Downarrow_{\mathfrak{B}})$ according to Definition 5.2; and likewise, the definition of global strategies follows from the local strategies in $\Lambda_{\mathfrak{B}}$ according to Definition 5.5. Furthermore, let $\Gamma \Downarrow_{\mathfrak{B}}$ be the set of plays given by the lower posets in $\mathfrak{B}$ defined by the composition of strategies in $\Lambda_{\mathfrak{B}}$. Finally, let $\mathcal{W} \Downarrow_{\mathfrak{B}}$ be restricted in the natural way to the new set of plays $\Gamma \Downarrow_{\mathfrak{B}}$ just defined;

this means, $\mathcal{W}\Downarrow_{\mathfrak{B}}(\hbar_{\mathfrak{B}}) = \mathcal{W}(\hbar_{\mathfrak{D}})$ iff $(\downarrow D \cup (\hbar_{\mathfrak{B}} \setminus \{\bot_{\mathfrak{B}}\})) = \hbar_{\mathfrak{D}}$, for all plays $\hbar_{\mathfrak{B}} \in \Gamma\Downarrow_{\mathfrak{B}}$.[4] This restriction for winning conditions is possible (i.e., it still defines a function) only because due to Lemma 5.9 the poset given by the union of the order ideals defined by D (which is not part of the new board $\bot \oplus \mathfrak{B}$ but existed in $\mathfrak{D}$) is unique and therefore can be disregarded when defining the winning conditions for the new game. □

The order dual of this result is a closure property under countable unions of ideals.

Lemma 5.11. (Closure under ideals) *Let $\mathfrak{I}$ be a CLG and D a global position of the board $\mathfrak{D}$ associated with $\mathfrak{I}$. The structure $\mathfrak{I}\Downarrow_{\mathfrak{B}} = (\Upsilon, \mathfrak{B}, \Lambda_{\mathfrak{B}}, \nabla\Downarrow_{\mathfrak{B}}, \mathcal{W}\Downarrow_{\mathfrak{B}}, \Gamma\Downarrow_{\mathfrak{B}})$ is also a CLG where $\mathfrak{B}$ is the lower sub-board of $\mathfrak{D}$ defined by D.*

Proof. The elements of the new game $\mathfrak{I}\Downarrow_{\mathfrak{B}}$ are constructed as follows: the poset $\mathfrak{B}$ is already a game board, which can be polarised using the partial function $\nabla\Downarrow_{\mathfrak{B}} = \nabla$. Using Definitions 5.2 and 5.5 one can construct the new sets $\Lambda_{\mathfrak{B}}$ and $\Gamma\Downarrow_{\mathfrak{B}}$ of strategies and plays of the game. Finally, $\mathcal{W}\Downarrow_{\mathfrak{B}}$ is defined to be $\mathcal{W}$ restricted to those plays in $\Gamma\Downarrow_{\mathfrak{B}}$,i.e., $\mathcal{W}\Downarrow_{\mathfrak{B}}(\hbar_{\mathfrak{B}}) = \mathcal{W}(\hbar)$ iff $\hbar_{\mathfrak{B}} = \hbar$, for any $\hbar_{\mathfrak{B}} \in \Gamma\Downarrow_{\mathfrak{B}}$ and $\hbar \in \Gamma$, or undefined otherwise – since $\mathcal{W}\Downarrow_{\mathfrak{B}}$ is a partial function as well. □

The previous statements show that the filters of the board $\mathfrak{D}$ of $\mathfrak{I}$ define the set of 'sub-games' of $\mathfrak{I}$; similarly, the order ideals in $\mathfrak{D}$ can define a subset of the set of plays of Γ. In particular, notice that it surely defines a subset of the plays in Γ if synchronisation points are preserved, which is formalised as follows; a board $\mathfrak{B} = (\mathcal{B}, \leq_{\mathcal{B}})$ preserves the synchronization points of a board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ iff:

$$\text{for all } a, b \in \mathcal{D}, \text{ if } \mathtt{sync}_{\mathfrak{D}}(a,b) \text{ and } a \in \mathcal{B} \vee b \in \mathcal{B}, \text{ then } \mathtt{sync}_{\mathfrak{B}}(a,b)$$

where $\mathtt{sync}_{\mathfrak{D}}$ and $\mathtt{sync}_{\mathfrak{B}}$ are the `sync` predicate over the elements of $\mathfrak{D}$ and $\mathfrak{B}$.

Also, notice that games on trees can be trivially reduced to the particular case when D is always a singleton set and where two chains never synchronise. However, a CLG on a poset $\mathfrak{D}$, even if having a tree-like shape, does not behave exactly as a game on a tree or a graph unfolding since the definition of strategies as closure operators still allows the players to make several moves in a single block. The final outcome of the two games is, nevertheless, the same.

Since the CLG models will be used for verification, another useful feature is that of having a game closed under *dual* games, this is, a game used to check the dual of a given property over the same board—i.e., for the same system(s). Formally:

Definition 5.12. (Dual games) Let $\mathfrak{I} = (\Upsilon, \mathfrak{D}, \Lambda_{\mathfrak{D}}, \nabla, \mathcal{W}, \Gamma)$ be a CLG. The dual game $\mathfrak{I}^{op}$ of $\mathfrak{I}$ is the tuple: $(\Upsilon, \mathfrak{D}, \Lambda_{\mathfrak{D}}, \nabla^{op}, \mathcal{W}^{op}, \Gamma)$, such that for all $d \in \mathcal{D}$ and $\hbar \in \Gamma$:

[4] In order to be clear with our notation, here we use $\cup$ as set union on posets: e.g., given two posets $\mathbb{R} = (\mathcal{R}, \leq_{\mathcal{R}})$ and $\mathbb{S} = (\mathcal{S}, \leq_{\mathcal{S}})$, we say that $\mathbb{R} \cup \mathbb{S}$ iff $\mathcal{R} \cup \mathcal{S}$; and similarly for set difference "\".

- if $\nabla(d) = \exists$ (resp. $\forall$) then $\nabla^{op}(d) = \forall$ (resp. $\exists$), and
- if $\mathcal{W}(\hbar) = \exists$ (resp. $\forall$) then $\mathcal{W}^{op}(\hbar) = \forall$ (resp. $\exists$).

Moreover, let $\mathfrak{J}$ be a class of CLG where for all $\mathcal{G} \in \mathfrak{J}$ there is a dual game $\mathcal{G}^{op} \in \mathfrak{J}$. Then, we say that $\mathfrak{J}$ is closed under dual games. ◁

This definition is based on the fact that changing ∇ for ∇^{op}, i.e., changing each polarity for its 'dual polarity' on every element of $\mathfrak{D}$, leaves unmodified the set of strategies and plays of the dual game, though intuitively each play in $\mathcal{G}$ that is winning for one of the players should be winning for the other player in the dual game $\mathcal{G}^{op}$.

Lemma 5.13. (Closure under dual games) *Let $\mathfrak{J}$ be a class of CLG closed under dual games. If Eve (resp. Adam) has a winning strategy in $\mathcal{G} \in \mathfrak{J}$, then Adam (resp. Eve) has a winning strategy in the dual game $\mathcal{G}^{op} \in \mathfrak{J}$.*

Proof. Suppose that Eve has a winning strategy ∂_W in $\mathcal{G}$. Since for all global positions in the game $\mathcal{G}$ one has that the next global position is initially defined by $\partial_W \circ \partial_\forall$, then whenever Adam has to make a move in $\mathcal{G}^{op}$ he can use the winning strategy ∂_W of Eve because for all $d \in D$, if $\nabla(d) = \exists$ then we have that $\nabla(d)^{op} = \forall$. However, notice that in each local position of the game board Adam must always "play first" both in $\mathcal{G}$ and in $\mathcal{G}^{op}$ because the global evolution of the game, which is determined by the rounds being played, is always defined by pairs of local strategies $\lambda^i_\exists$ and $\lambda^j_\forall$ such that $\lambda^i_\exists \circ \lambda^j_\forall(d)$, for any local position d, regardless of whether we are playing $\mathcal{G}$ or $\mathcal{G}^{op}$.

So, there are actually two cases: firstly, consider those $d \in D$, for any global position D, such that $\nabla(d) = \exists$. In this case $\nabla^{op}(d) = \forall$ and then in $\mathcal{G}^{op}$ Adam can simply play Eve's strategy in $\mathcal{G}$ at position d. The second case is that of those $d \in D$ such that $\nabla(d) = \forall$. In this case, $\nabla^{op}(d) = \exists$ and hence Adam can play d itself, and let Eve decide on the new local position d', for which, by hypothesis, Eve has a winning strategy in $\mathcal{G}$ and the two previous cases apply again, though in a new round of the game; moreover, the behaviour at synchronization points, which are "played" deterministically by the environment, remains as in $\mathcal{G}$. In this way, Adam can enforce in $\mathcal{G}^{op}$ all plays that Eve can enforce in $\mathcal{G}$. Finally, since for all such plays in Γ it was, by hypothesis, Eve who was the winner, i.e., $\mathcal{W}(\hbar) = \exists$ for all $\hbar \in \Gamma$, then Adam is the winner in all plays in $\mathcal{G}^{op}$ since now for all $\hbar \in \Gamma$, we have that $\mathcal{W}^{op}(\hbar) = \forall$. The case in which Adam has a winning strategy in $\mathcal{G}$ is dual. □

5.1.3 Towards Determinacy

Lemma 5.13 does not imply that concurrent games are determined because the existence of winning strategies has not yet been ensured; let alone the guarantee that finite and open plays where $D' = D$ (for two consecutive global positions D and D') are not possible, since this

immediately implies that the game is undetermined; call *stable* any play containing two global positions D, D' such that $D \to D'$ and $D = D'$.

Also notice that another condition that is necessary, though not sufficient, for a game to be determined is that all plays (that are not finite and open) have a winner. This can be ensured by requiring the winning conditions to be *complete*. Let the winning conditions $\mathcal{W}$ be complete iff $\mathcal{W}$ is a total function on all plays in Γ that are not finite and open. We say that $\mathcal{W}$ is complete because it covers all possible cases where a winning set has been defined, i.e., those plays that are closed as well as those that are both infinite and open.

Lemma 5.14. (Unique winner) *Let $\mathfrak{J}$ be a class of CLG closed under dual games for which plays that are stable, finite and open do not occur. If the $\mathcal{W}$ in $\eth \in \mathfrak{J}$ is complete, then every play in $\eth$ and $\eth^{op}$ has a unique winner.*

Proof. The statement holds immediately for $\eth$. On the other hand, since $\eth$ belongs to a class of CLG that is closed under dual games, then the set of plays in $\eth^{op}$ is the same as in $\eth$, but with the players making dual choices in $\eth^{op}$ as described in the proof of Lemma 5.13. As a consequence, $\mathcal{W}^{op}$ must also be a complete set of winning conditions, and hence, every play $\hbar \in \Gamma$ of $\eth^{op}$ has a unique winner as well. □

The previous statement does not imply determinacy either, but it takes us one step closer to it. Determinacy of CLG will be shown as a direct consequence of two other properties (soundness and completeness), which are more useful from a verification viewpoint; these properties are shown to hold for a big class of game boards with a semantic property that is studied in the next section.

Remark 5.15. If a class of games is closed under dual games and has a complete set of winning conditions, then a constructive proof of determinacy, which does not rely on Martin's theorem [59] or on similar results for infinite games with perfect information can be given—e.g. using the Gale–Stewart theorem [31] for determinacy of two-player infinite games with perfect information—provided that the game is shown to be *sound*. This is because, roughly speaking, the scenario of possible game outcomes becomes symmetric, and disallow any draws. Hence sound games with these characteristics must necessarily be also *complete*, and therefore *determined*. In this way one can get proofs of completeness and determinacy almost for *free*! This property is extremely useful since it considerably reduces reasoning on the games. ◁

Let us finish this section with a simple counter-example that shows that CLG are undetermined even with the restrictions already imposed. The following result motivates the definition of a semantics condition that, when satisfied, provides the mathematical properties for the construction of a sound and complete game, which is therefore also determined.

Proposition 5.16. *CLG are undetermined in the general case.*

Proof. Neither player can have a winning strategy in the game presented in Figure 5.1 since Eve and Adam can enforce plays for which $\mathcal{W}$ is not defined (a stable, finite and open play). Let us look in detail at how the game is played:

Since Adam loses whenever the local position $d_{10}^{\exists}$ is played, then he must avoid playing the local position $d_4^{\forall}$ as well. That leaves him with only one possible sensible choice to make, which is to play the local position $d_3^{\forall}$ once he reaches the local position $d_1^{\forall}$. Therefore, the best strategy for Adam tells him to play at once local positions $d_0^{\forall}$, $d_1^{\forall}$, and $d_3^{\forall}$. As $d_3^{\forall}$ is a fixpoint of the local strategies of Adam, then he stops there (in such a chain). In addition, since Adam's choices must preserve synchronization points, then he is forced to play $d_2^{\exists}$ as well because:

$$\forall d \in (\uparrow\{d_0^{\forall}\} \cap \downarrow\partial_{\forall}(\{d_0^{\forall}\})), \text{ if } \mathtt{BP}(d) \wedge \nabla(d) = \forall \text{ then}$$
$$\forall a,b \in d^{\rightarrow}.\ \mathtt{sync}(a,b) \text{ implies } a,b \in \downarrow\partial_{\forall}(\{d_0^{\forall}\}).$$

and we have that $d_0^{\forall} \in (\uparrow\{d_0^{\forall}\} \cap \downarrow\partial_{\forall}(\{d_0^{\forall}\}))$, and $\mathtt{BP}(d_0^{\forall})$ holds, and $\nabla(d_0^{\forall}) = \forall$, and $d_1^{\forall}, d_2^{\exists} \in d_0^{\forall\rightarrow}$, and $\mathtt{sync}(d_1^{\forall}, d_2^{\exists})$ holds, but (so far) $d_2^{\exists} \notin \downarrow\partial_{\forall}(\{d_0^{\forall}\})$; then, $d_2^{\exists}$ must be played as well, i.e., $\partial_{\forall}(\{d_0^{\forall}\}) = \{d_2^{\exists}, d_3^{\forall}\}$. Then, Eve must respond to Adam's choices. All local positions different from $d_2^{\exists}$ are fixpoints of her local strategies, so she agrees with Adam's choices. Then, she must make a non-trivial move only on $d_2^{\exists}$. As Eve wins whenever $d_{10}^{\exists}$ is played, then her best strategy tells her to move to $d_6^{\exists}$ when playing at $d_2^{\exists}$. Note that she is not forced to play $d_5^{\exists}$ because $\mathtt{sync}(d_5^{\exists}, d_6^{\exists})$ does not hold. Thus, the new global position is $\{d_3^{\forall}, d_6^{\exists}\}$. Then, in all further rounds neither player can make a move because both local positions are fixpoints of their local strategies. As a consequence, a stable, finite and open play is generated. □

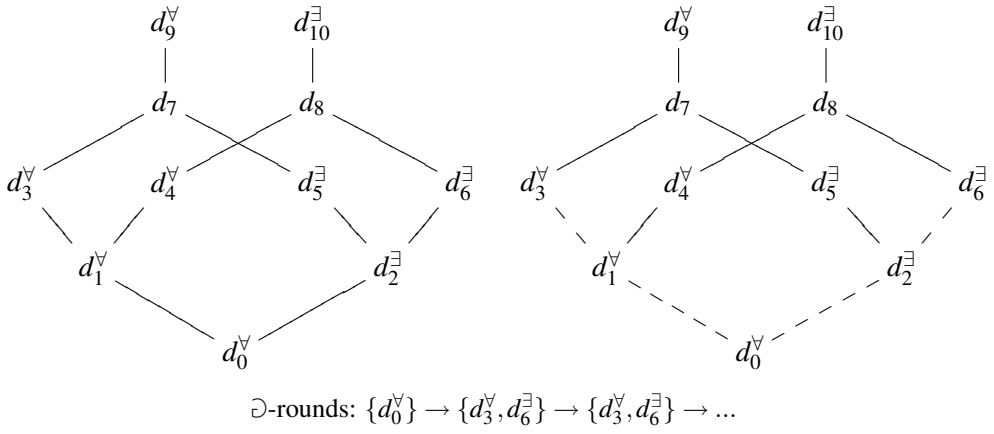


Figure 5.1: Graphical representation of an undetermined game. The poset on the left is the board at the beginning of the game. Nodes in the poset are labelled with their polarities and the dashed lines represent the play so far, i.e., the choices made by the players. Therefore, the poset on the right is a representation of the game after the first ∋-round. In this example $\mathcal{W}(\downarrow D) = \exists$ iff $d_{10}^{\exists} \in D$ and $\mathcal{W}(\downarrow D) = \forall$ otherwise, for all global positions D containing at least one terminal element of the board. Such winning conditions justify the way Eve and Adam play this game. Clearly, the play is stable, finite and open.

5.2 Metatheorems for Verification

We now study a semantic condition that ensures the correctness of a class of CLG models to which different verification problems for various models of concurrency can be reduced in a partial order setting. As explained before a CLG can be seen as a game representation of a verification problem. Then, let $\eth_{\mathrm{P}}$ be the CLG associated with a *decision* problem V(P), for a given problem P, and $\mathfrak{J}$ the class of CLG representing all instances of such a decision problem.

We say that V(P) holds iff such a decision problem has a positive solution (i.e., iff it has a 'yes' answer), and fails to hold iff it has a negative solution; thus, V(P) can be used as a logical predicate. For instance, suppose that P is a bisimulation problem, denoted by $\sim$, for two systems $\mathfrak{M}_1$ and $\mathfrak{M}_2$; then V($\mathfrak{M}_1$,$\mathfrak{M}_2$,$\sim$) holds iff $\mathfrak{M}_1 \sim \mathfrak{M}_2$, and fails to hold otherwise. As usual for logic games, a game $\eth_{\mathrm{P}}$ is correct iff Eve has a winning strategy in $\eth_{\mathrm{P}}$ whenever V(P) holds, and conversely, Adam has a winning strategy in $\eth_{\mathrm{P}}$ whenever V(P) fails to hold.

Now, let a 'local configuration' of a game $\eth_{\mathrm{P}} \in \mathfrak{J}$ be a local position, and similarly, a 'global configuration' be a set of independent local configurations—i.e., an anti-chain of local configurations. As usual for verification games, a 'true configuration' is a configuration, either local or global, from which Eve can win the game; dually, a 'false configuration' is a configuration from which Adam can do so. In a CLG a global configuration is logically interpreted in a conjunctive way; so, we also say that a global configuration is true iff it contains only true local configurations, and it is a false configuration otherwise.

In order to show the correctness of the family of games $\mathfrak{J}$, in this abstract setting, we need to make sure that the CLG $\eth_{\mathrm{P}}$ associated with a particular verification problem V(P) has two properties: ω-Symmetry and local correctness. Before presenting such properties let us provide the definition of some conditions that will be used in the definition of one of such properties:

Definition 5.17. (Parity condition) Let $(\mathcal{A}, \leq_{\mathcal{A}})$ be a poset indexed by a finite subset of $\mathbb{N}$, $\vec{a}$ a sequence of elements of $\mathcal{A}$ whose order respects $\leq_{\mathcal{A}}$ and downward-closure, and $f^{\omega}_{\mathtt{min}} : \mathcal{A}^{\omega} \to \mathbb{N}$ a function that characterizes the minimum index that appears infinitely often in $\vec{a}$. Then, the poset $(\{b \in \mathcal{A} \mid b \in \vec{a}\}, \leq_{\mathcal{A}})$ is definable by a Parity condition iff $f^{\omega}_{\mathtt{min}}(\vec{a})$ is even. ◁

Definition 5.18. (Büchi condition) Let $(\mathcal{A}, \leq_{\mathcal{A}})$ be a poset indexed by a finite subset of $\mathbb{N}$, $B \subseteq \mathbb{N}$, $\vec{a}$ a sequence of elements of $\mathcal{A}$ whose order respects $\leq_{\mathcal{A}}$ and downward-closure , and $f^{\omega} : \mathcal{A}^{\omega} \to \wp(\mathbb{N})$ a function characterizing the indices that appear infinitely often in $\vec{a}$. Then, the poset $(\{b \in \mathcal{A} \mid b \in \vec{a}\}, \leq_{\mathcal{A}})$ is definable by the Büchi condition B iff $(f^{\omega}(\vec{a}) \cap B) \neq \emptyset$. ◁

Definition 5.19. (Rabin condition) Let $(\mathcal{A}, \leq_{\mathcal{A}})$ be a poset indexed by a finite subset of $\mathbb{N}$, R a set of pairs (G,H) such that $G,H \subseteq \mathbb{N}$, $\vec{a}$ a sequence of elements of $\mathcal{A}$ whose order respects $\leq_{\mathcal{A}}$ and downward-closure, and $f^{\omega} : \mathcal{A}^{\omega} \to \wp(\mathbb{N})$ a function characterizing the indices that appear infinitely often in $\vec{a}$. Then, the poset $(\{r \in \mathcal{A} \mid r \in \vec{a}\}, \leq_{\mathcal{A}})$ is definable by the Rabin condition R iff there exists a pair $(G_k, H_k) \in R$ such that $(f^{\omega}(\vec{a}) \cap G_k) = \emptyset$ and $(f^{\omega}(\vec{a}) \cap H_k) \neq \emptyset$. ◁

Dually, one also has co-Parity ($f^{\omega}_{\mathtt{min}}(\vec{a})$ is odd), co-Büchi ($(f^{\omega}(\vec{a}) \cap B) = \emptyset$), and Streett (for all pairs $(G_k, H_k) \in R$, either $(f^{\omega}(\vec{a}) \cap G_k) \neq \emptyset$ or $(f^{\omega}(\vec{a}) \cap H_k) = \emptyset$) conditions, respectively.

Property 5.20. (ω-Symmetry: bi-complete ω-regularity) A family $\mathfrak{J}$ of CLG has Property 5.20 and is said to be bi-complete ω-regular, or ω-*symmetric* for short, iff:

1. $\mathfrak{J}$ is closed under dual games;
2. for all $\eth_{\mathrm{P}} \in \mathfrak{J}$ we have that $\eth_{\mathrm{P}}$ has a complete set of winning conditions;
3. the winning set given by those plays such that $\mathcal{W}(\hbar) = \exists$, i.e., those where Eve wins, is definable by Büchi/Rabin/Parity conditions. ◁

An immediate consequence of the previous property is the following:

Lemma 5.21. *If a CLG $\eth_{\mathrm{P}}$ is ω-symmetric, then it also satisfies that the winning set given by those plays such that $\mathcal{W}(\hbar) = \forall$ is definable by co-Büchi/Streett/co-Parity conditions.*

Proof. Immediate from the fact that since $\eth_{\mathrm{P}}$ is closed under dual games and the set of winning conditions is complete, then the set of plays such that $\mathcal{W}(\hbar) = \forall$ must be definable as its dual, i.e., by co-Büchi/Streett/co-Parity conditions. □

Note that parts 1 and 3 of Property 5.20 (i.e., ω-symmetry) are given by the particular decision problem to be solved. It is well known that several game characterisations of many verification problems have these two properties. On the other hand, part 2 of Property 5.20 is a design issue. It must be ensured when defining the game since it determines, along with part 3, the particular problem being solved. In addition, Property 5.20 and Lemma 5.21 imply that:

Lemma 5.22. *The winning sets of Adam are least fixpoint definable; and dually, the winning sets of Eve are greatest fixpoint definable.*

Proof. The Büchi and Rabin conditions can be reduced to a Parity condition (cf. [38]). Moreover, a Parity condition characterises the winning sets (and winning plays) in $\mathcal{L}_{\mu}$ as follows: infinite plays where the smallest index that appears infinitely often is even (resp. odd) satisfy greatest (resp. least) fixpoints and belong to the winning sets of Eve (resp. Adam). As in our setting plays are posets, the order is the one given by the board. □

Hereafter, we only consider CLG models that are ω-symmetric and, moreover, for which the following semantic condition on game boards holds:

Property 5.23. (Local correctness) Let $\mathfrak{D}$ be the board of a CLG $\eth_{\mathrm{P}}$. If $d \in \mathcal{D}$ is a false configuration, then either $\nabla(d) = \exists$ and all next configurations are false as well or $\nabla(d) = \forall$, i.e., Eve must preserve falsity whereas Adam can preserve it. Dually, if $d \in \mathcal{D}$ is a true configuration, then either $\nabla(d) = \forall$ and all next configurations are true as well or $\nabla(d) = \exists$, i.e., Adam must preserve truth whereas Eve can preserve it. ◁

This local correctness condition implies a global correctness condition:

Corollary 5.24. (Global correctness) *Let $\mathfrak{D}$ be the board of a CLG $\eth_{\mathrm{P}}$. If $D \subseteq \mathcal{D}$ is a false global configuration, then there exists some $d \in D$ such that d is a false local configuration; and dually, if $D \subseteq \mathcal{D}$ is a true global configuration, then for all $d \in D$ we have that d is a true local configuration.*

The game interpretation of Property 5.23 reveals the mathematical property that makes a CLG logically correct (in a Tarskian context rather than in the sense of Hintikka), since such a mathematical property will ensure the existence of winning strategies in all cases, and therefore every play will always have a winner—i.e., stable, finite and open plays will be avoided. Specifically, Property 5.23 implies that not only the local positions that belong to a player must be either true or false local configurations, but also those that belong to the environment, i.e., the joins of $\mathfrak{D}$. Then, truth and falsity must be transferred to those local positions as well, so that the statements "Eve must preserve falsity" and "Adam must preserve truth" hold. Formally, one needs to ensure that the following restriction (which we call '$\eth$-progress') holds:

$$\bigsqcup D \neq \emptyset \Rightarrow \bigsqcup_{d \in D} \partial_{\forall}(\{d\}) \neq \emptyset$$
$$\bigsqcup D \neq \emptyset \Rightarrow \bigsqcup_{d \in D} \partial_{\exists}(\{d\}) \neq \emptyset$$

where D is a global position and $\bigsqcup$ is the 'join operator'. Call 'live' a play that is not stable, finite and open, as well as games whose strategies only generate live plays; $\eth$-progress guarantees that only live plays and games—where truth and falsity are preserved—are generated.

Remark 5.25. Note that in the game presented in Figure 5.1 the strategy of Eve does not satisfy this condition: on the one hand $\bigsqcup\{d_3^{\forall}, d_2^{\exists}\} = \{d_7\}$, and on the other, $\bigsqcup_{d \in \{d_3^{\forall}, d_2^{\exists}\}} \partial_{\exists}(\{d\}) = \partial_{\exists}(\{d_3^{\forall}\}) \sqcup \partial_{\exists}(\{d_2^{\exists}\}) = \{d_3^{\forall}\} \sqcup \{d_6^{\exists}\} = \emptyset$. ◁

With the restriction to strategies that preserve the existence of joins, the game in Figure 5.1 becomes the game in Figure 5.2, for which Adam has a winning strategy. Let us look in detail at how the game is played this time, and how Adam can enforce a winning play for him.

As before, since Adam loses whenever the local position $d_{10}^{\exists}$ is played, then he must avoid playing the local position $d_4^{\forall}$ as well. He, then, plays as in the game in Figure 5.1, this is, he plays at once the local positions $d_0^{\forall}$, $d_1^{\forall}$, and $d_3^{\forall}$, together with the local position $d_2^{\exists}$, which he is forced to play because of the condition that states that $\forall d \in (\uparrow\{d_0^{\forall}\} \cap \downarrow\partial_{\forall}(\{d_0^{\forall}\}))$, if $\mathtt{BP}(d) \wedge \nabla(d) = \forall$, then $\forall a, b \in d^{\rightarrow}$. $\mathtt{sync}(a,b)$ implies $a, b \in \downarrow\partial_{\forall}(\{d_0^{\forall}\})$.

Next, Eve must respond to Adam's choices and since all local positions different from $d_2^{\exists}$ are fixpoints of her local strategies, then she agrees with Adam's choices again. After that, she must make a non-trivial move only on $d_2^{\exists}$. As Eve wins whenever $d_{10}^{\exists}$ is played, then her best strategy tells her to move to $d_6^{\exists}$ when playing at $d_2^{\exists}$. However, unlike the game previously shown in Figure 5.1, Eve is this time forced to play $d_5^{\exists}$ because of the following

reason: since $\bigsqcup\{d_3^\forall, d_2^\exists\} = \{d_7\} \neq \emptyset$, then it must hold that $\bigsqcup_{d\in\{d_3^\forall,d_2^\exists\}} \partial_\exists(\{d\}) \neq \emptyset$ as well, but $\bigsqcup_{d\in\{d_3^\forall,d_2^\exists\}} \partial_\exists(\{d\}) = \partial_\exists(\{d_3^\forall\}) \sqcup \partial_\exists(\{d_2^\exists\}) = \{d_3^\forall\} \sqcup \{d_6^\exists\} = \emptyset$. Therefore, the global strategy of Eve must be changed when playing at $d_2^\exists$ (from $\{d_6^\exists\}$ to $\{d_5^\exists, d_6^\exists\}$) so that $\partial_\exists(\{d_3^\forall\}) \sqcup \partial_\exists(\{d_2^\exists\})$ is $\{d_3^\forall\} \sqcup \{d_5^\exists, d_6^\exists\} = \{d_7\} \neq \emptyset$, as desired.

Thus, the new (intermediate) global position is $\{d_3^\forall, d_5^\exists, d_6^\exists\}$. In this case, the environment can make a deterministic move because $\mathtt{SP}(d_7)$ holds and $d_7^{\leftarrow} = \{d_3^\forall, d_5^\exists\} \subseteq \{d_3^\forall, d_5^\exists, d_6^\exists\}$. Then, the global position after this round is $\{d_6^\exists, d_9^\forall\}$. In all further rounds neither player can make a move because both local positions, i.e., $d_6^\exists$ and $d_9^\forall$, are fixpoints of their local strategies. This play is, however, winning for Adam since $\mathcal{W}(\downarrow\{d_6^\exists, d_9^\forall\}) = \forall$.

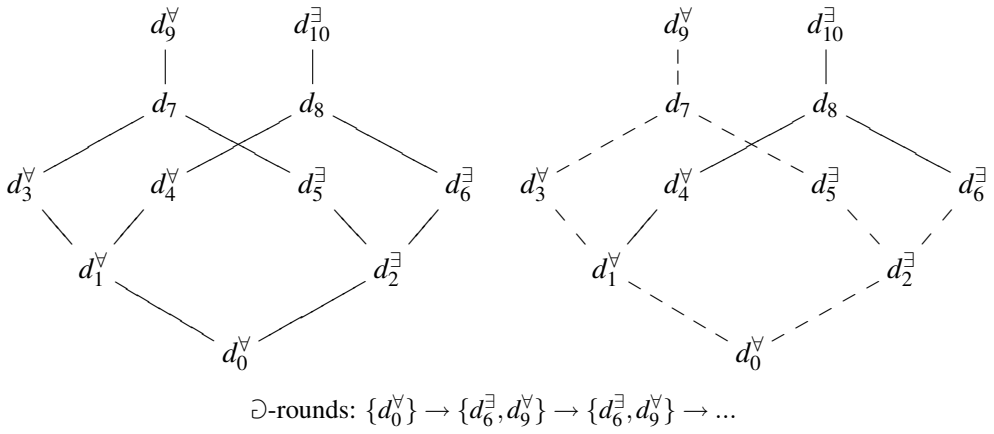


Figure 5.2: A CLG where Adam has a winning strategy.

Remark 5.26. Note that if the board of the game is a join-semilattice, then the restriction on strategies is automatically satisfied, and then all usual strategies can be considered. Moreover, if the strategies preserve the existence of joins, then any play will always have a winner (provided that the set of winning conditions is complete) since either a terminal element is reached or an infinite and open plays is generated—i.e., a deadlock in the game never happens. ◁

Given this new restriction on strategies we can move towards showing the correctness of this kind of CLG. A simple technical lemma is still needed: a direct application of Lemma 5.14 using Property 5.20 gives us Lemma 5.27, which allows one to show the soundness of this concurrent game using only *pure* (deterministic and concurrent) winning strategies.

Lemma 5.27. *Every play of a live,* ω*-symmetric* $\eth_{\mathrm{P}}$ *has a uniquely determined winner.*

Proof. A direct application of Lemma 5.14 using Property 5.20 (ω-symmetry) when restricted to strategies which preserve the existence of joins (i.e., to live plays). □

In the remainder of this chapter we only consider live, ω-symmetric games.

Theorem 5.28. (Soundness) *If* V(P) *fails to hold, then Adam can always win* $\supset_{\mathrm{P}}$.

Proof. We show that Adam can win all plays of $\supset_{\mathrm{P}}$ if V(P) fails to hold by providing a winning strategy for him. The proof has two parts: first, we provide a board where Adam can always win and show how to construct a game on that board, in particular, the local strategies in the game—and hence, a strategy for Adam; then, we show that in such a game Adam can always win by checking that his global strategy is indeed a winning strategy for the game.

Firstly, let $\supset_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$ be a CLG on a poset $\mathfrak{B} = (\mathcal{B}, \leq_{\mathcal{D}})$, which is a subset of $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$, the initial board of the game. Let the set $\mathcal{B}$ be a downward-closed subset of $\mathcal{D}$ with respect to $\leq_{\mathcal{D}}$; the bottom element $\bot_{\mathcal{B}} = \bot_{\mathcal{D}}$ (where the game starts) is, by hypothesis, a false configuration.

The construction of the board is as follows: $\mathfrak{B}$ contains only the winning choices for Adam, which preserve falsity as defined by the local correctness semantic property 5.23. After those elements of the poset have been selected, adjoin to them all possible responses or moves available to Eve that appear in $\mathfrak{D}$. Do this, starting from $\bot$, either infinitely often for infinite chains or until a terminal element is reached in finite chains. This construction clearly ensures that $\mathcal{B}$ is a downward-closed set with respect to $\leq_{\mathcal{D}}$. In addition, as in the proofs of Lemmas 5.10 and 5.11 (closure properties), the polarity function ∇ for $\mathfrak{B}$ is as in $\mathfrak{D}$.

Using the constructions given in the proof of Lemma 5.10, one can define all other elements of $\supset_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$. In particular, the local strategies for Eve and Adam will be 'stable' closure operators; based on Definition 5.2 such stable closure operators are completely defined once one has determined what the 'output' functions will be (since the fixpoints are completely determined already in Definition 5.2). Then, each local strategy $\lambda^j_\forall$ for Adam and each local strategy $\lambda^i_\exists$ for Eve – where $j \in \mathrm{K}_\forall \subseteq \mathbb{K}_\forall$ and $i \in \mathrm{K}_\exists \subseteq \mathbb{K}_\exists$, respectively – is defined as follows:[5]

$$\lambda^i_\exists(d) = d \vee f^i_\exists(d)$$
$$\lambda^j_\forall(d) = d \vee g^j_\forall(d)$$

where:

$$\begin{array}{lcll} \lambda^i_\exists(d) & = & d & \text{, if } \mathtt{fix}_\exists(\lambda^i_\exists, d) \\ \lambda^i_\exists(d) & = & f^i_\exists(d) & \text{, otherwise} \\ \lambda^j_\forall(d) & = & d & \text{, if } \mathtt{fix}_\forall(\lambda^j_\forall, d) \\ \lambda^j_\forall(d) & = & g^j_\forall(d) & \text{, otherwise} \end{array}$$

where each 'output' function $g^j_\forall$ necessarily preserves falsity and each output function $f^i_\exists$ must preserve truth (because $\mathfrak{B}$ was constructed taking into account Property 5.23). However, since in $\mathfrak{B}$ all choices available to Eve were preserved, then the set of local strategies for Eve (i.e., $\Lambda^\exists_{\mathfrak{B}}$) can be safely chosen to be the same set of local strategies in $\mathfrak{D}$ (i.e., $\Lambda^\exists_{\mathfrak{D}}$); therefore, $\Lambda^\exists_{\mathfrak{B}} = \Lambda^\exists_{\mathfrak{D}}$ and $\Lambda^\forall_{\mathfrak{B}} \subseteq \Lambda^\forall_{\mathfrak{D}}$; moreover, the definition of global strategies immediately follows from this specification of local strategies as given by Definition 5.5 – of course, subject the

[5] Recall that $i, j > 1$ since $\lambda^1_\forall$ and $\lambda^1_\exists$ are the identity local strategies for Adam and Eve, respectively.

restriction that any such global strategy must preserve the existence of joins in $\mathfrak{B}$. Finally, the sets of plays and winning conditions are defined from $\mathfrak{B}$ and the new sets of strategies as done in the proof of Lemma 5.10.

For the second part of this proof, let us show that the game $\eth_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$ is winning for Adam, i.e., that his strategy in such a game is indeed a winning strategy. Then, let us analyse the outcome of plays to certify that he indeed wins all plays in such a game. First consider finite plays, which must be closed because all valid strategies must preserve the existence of joins. All such plays have a global position D_f which contains at least one local position that is a terminal element of $\mathfrak{B}$. Due to Property 5.20 (part 2), all those plays are effectively recognised as winning for one of the players, in this case for Adam: since $\bot$ is a false configuration, Eve must preserve falsity, and Adam is only playing strategies that also preserve falsity, then D_f contains at least one local position d_f which also is a false configuration, and therefore D_f is a false configuration as well since it is interpreted conjunctively. As a consequence all finite plays are winning for Adam. The same argument also applies for infinite, closed plays. The final case, is that of open, infinite plays.

The correctness of this case is shown by a transfinite induction on a well-founded poset of sub-boards of $\mathfrak{B}$; this technique generalizes the analysis of approximants of fixpoints on interleaving structures (i.e., on total orders) to a partial ordered setting. So, let $(O, \leq_O)$ be the following partial order on sub-boards (i.e., posets):

$$O = \{\, \eth \Downarrow_{\uparrow D} \mid D \text{ is a global position of } \mathfrak{B}\}$$
$$\eth \Downarrow_{\uparrow D} \leq_O \eth \Downarrow_{\uparrow D'} \text{ iff } \uparrow D' \subseteq \uparrow D$$

The relation $\leq_O$ is clearly well-founded because all finite and infinite chains in the poset $(O, \leq_O)$ have $\bot_O = \eth \Downarrow_{\uparrow \bot_{\mathcal{B}}} = \eth \Downarrow_{\mathfrak{B}}$ as their bottom element. Since any particular play in the game corresponds to a chain of $(O, \leq_O)$, then let us also define a valuation $[\![\cdot]\!] : O \to \{\mathbf{true}, \mathbf{false}\}$ and a total order on the sub-boards (i.e., posets), and therefore subgames, associated with $\mathfrak{B}$. Let $\hbar$ be any open, infinite play (an infinite chain of $(O, \leq_O)$) and let $\alpha, \varpi \in \mathbb{O}\mathrm{rd}$ be two ordinals, where ϖ is a limit ordinal. Then:

$$\begin{array}{lcll} [\![\hbar^0]\!] & = & [\![\bot_O]\!] & \text{(the base case)} \\ [\![\hbar^{\alpha+1}]\!] & = & [\![\to_O (\hbar^\alpha)]\!] & \text{(the induction step)} \\ [\![\hbar^{\varpi}]\!] & = & [\![\bigcup_{\alpha<\varpi}(\hbar^\alpha)]\!] & \text{(because } \varpi \text{ is a limit ordinal)} \end{array}$$

where $\to_O$ is the accessibility relation of $\leq_O$ restricted to the elements of the chain $\hbar$. Then, for Adam, we have the following:

$$\begin{array}{lcll} [\![\bot_O]\!] & = & \mathbf{false} & \text{(by hypothesis, } \bot_O \text{ is a false configuration)} \\ [\![\to_O (\hbar^\alpha)]\!] & = & [\![\hbar^\alpha]\!] & \text{(due to Property 5.23, } \to_O \text{ preserves falsity)} \\ [\![\bigcup_{\alpha<\varpi}(\hbar^\alpha)]\!] & = & \bigvee_{\alpha<\varpi} [\![\hbar^\alpha]\!] & \text{(because due to Lemma 5.22,} \\ & & & \text{Adam's winning sets are least fixpoint definable)} \end{array}$$

Due to the principle of (transfinite) fixpoint induction, the result holds for all ordinals, and therefore for all global positions of any open, infinite play. Note that we can actually repeat this analysis for all ordinals $\beta < \alpha$ (and thus for all global positions), due to Property 5.20 (part 3), since winning configurations, and hence winning sets, are fixpoint definable. But, since the ordinal numbers are well-founded such a process of checking subgames and open, infinite plays always terminates regardless of which α one chooses. Hence, there can be neither a global position D nor a game $\Game_{\mathrm{P}} \Downarrow_{\perp_{\mathscr{B}} \oplus \uparrow D}$ where Eve wins.

As she cannot win any play in $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$, and due to Lemma 5.27 all plays have a unique winner, Adam's strategy is indeed a winning strategy in $\Game_{\mathrm{P}}$; in fact, it is a *pure* and *deterministic* winning strategy. Therefore, If V(P) fails to hold then Adam can always win $\Game_{\mathrm{P}}$. □

The last part of the previous proof shows that if, on the contrary, one supposes that Eve could win from a global position D, i.e., an element $\hbar^{\alpha}$ for some ordinal α, then one immediately would get a contradiction: α cannot be a limit ordinal because in that case $[\![\hbar^{\varpi}]\!] = \bigwedge_{\alpha<\varpi} [\![\hbar^{\alpha}]\!]$ (as due to Lemma 5.22, Eve's winning sets are greatest fixpoint definable); and since ordinals are well-founded, regardless of which α one chooses, there cannot be a descending chain of global positions that can be used to either satisfy a greatest fixpoint or fail to satisfy a least fixpoint. Therefore, Eve cannot win from any such global positions/configurations.

A similar proof can be given to show the completeness of the game. Nevertheless, due to the properties of the game, we can get the proof of completeness almost for free. And, moreover, determinacy with *pure* winning strategies—a property not obvious for concurrent games—follows immediately from the soundness and completeness results.

Theorem 5.29. (Completeness) *If* V(P) *holds, then Eve can always win* $\Game_{\mathrm{P}}$.

Proof. Due to Property 5.20 (part 1) there exists a dual CLG $\Game^{op}_{\mathrm{P}}$ for the dual verification problem V(P^{op}) of V(P) such that V(P^{op}) does not hold. And, due to Theorem 5.28 Adam has a winning strategy in the game $\Game^{op}_{\mathrm{P}}$ for the dual problem P^{op}. Therefore, due to Lemma 5.13 and Lemma 5.27, Eve can use the local strategies of Adam in $\Game^{op}_{\mathrm{P}}$ to be the unique winner of all plays $\hbar \in \Gamma$ of $\Game_{\mathrm{P}}$, and hence the existence of a winning strategy for Eve in $\Game_{\mathrm{P}}$ follows. □

Corollary 5.30. (Determinacy) *Eve has a winning strategy in* $\Game_{\mathrm{P}}$ *iff Adam does not have it, and vice versa.*

Proof. Follows immediately from Theorems 5.28 and 5.29. □

As said before determinacy is not a common feature for concurrent games with pure (i.e., not randomised) strategies as it is the case of CLG. Determinacy is mainly enforced by the semantic property of the game boards we have considered, which in turn makes oneself to restrict to join-preserving strategies for Adam. In fact, that semantic condition is necessary if one is looking for a determined class of games with imperfect information.

For instance, the game of 'Scissors-Paper-Stone', much simpler than the CLG defined here, is already undetermined, because its board does not have the local correctness property. That game has Property 5.20 because the two players can interchange roles and every play has a unique winner provided that draws are defined as winning for one of the two players. However the board where the game is played does not have the local correctness semantic property that is needed to provide a winning strategy for Eve or Adam. That very simple game is therefore undetermined. This shows that this semantic property is very natural and perhaps the least one can hope for when designing a concurrent game for verification with *pure* winning strategies.

Decidability on Finite Posets. Solving a CLG $\Game_{\mathrm{P}}$ using the approach we have presented here requires the construction of a winning game $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$ (and with it a winning strategy) for either Eve or Adam, according to Theorems 5.28 and 5.29. This is in general an undecidable problem because the board $\mathfrak{D}$ can be infinitely large. However, in many practical cases $\mathfrak{D}$ can admit a finite representation of it where all information needed to solve the verification problem is contained. Before showing particular instances where $\mathfrak{D}$ can be given a finite representation (in the following section), let us state the following result:

Theorem 5.31. (Decidability) *The winner of any CLG $\Game_{\mathrm{P}}$ can be decided in finite time if the board $\mathfrak{D}$ in $\Game_{\mathrm{P}}$ has finite size.*

Proof. Since $\mathfrak{D}$, by hypothesis, has finite size, then there are only finitely many possible subboards $\mathfrak{B}$, and consequently, finitely many subgames $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$ that must be checked before constructing a winning one for either player. Moreover, constructing a particular game $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$ either for Eve or Adam as described in the proofs of Theorems 5.28 and 5.29 can be effectively done, also due to the fact that $\mathfrak{D}$ is finite as follows.

Firstly, since $\mathfrak{B}$ is finite there are finitely many different strategies for Eve and Adam. Moreover, since those strategies are closure operators in a finite structure (i.e., order-preserving maps on a finite structure), then their sets of closed elements eventually stabilize. As a consequence, there are finitely many different plays (and game configurations), whose winner can always be checked—because the game is determined and its set of winning conditions is complete. Therefore, a winning strategy can be chosen from the set of strategies of the game by exhaustively searching such a set, simply by comparing it against all possible strategies of the other player. As we assume that Properties 5.20 and 5.23 hold, they need not be verified. □

Although decidability on finite systems is not a surprising result, what is interesting is that several partial order models of concurrency can be given a finite poset representation which, in a number of cases, can be *smaller* than their interleaving counter-part. Therefore, the previous decidability result can have important practical applications since it opens up the possibility of defining new concurrent decision procedures for different verification problems.

5.3 Expressivity

In this section we use the CLG model just defined to represent two different verification problems, namely bisimulation and model-checking. Although the two decision problems are different, using this generic technique they can be solved in a uniform way. Recall the structure of a CLG $\Im$. From its six components, one needs to determine $\mathfrak{D}$, ∇, and $\mathcal{W}$ in order to make it a concrete game since Υ is fixed and $\Lambda_{\mathfrak{D}}$ can be determined by $\mathfrak{D}$, and Γ by $\mathfrak{D}$ and $\Lambda_{\mathfrak{D}}$.

The two reductions rely on a simple observation: both problems can be represented by a binary relation between the elements of two posets $\mathbb{M} = (\mathcal{M}, \leq_{\mathcal{M}})$ and $\mathbb{T} = (\mathcal{T}, \leq_{\mathcal{T}})$. In the bisimulation case $\mathbb{M}$ and $\mathbb{T}$ are the poset representations of two systems $\mathfrak{M}_1$ and $\mathfrak{M}_2$, whereas in the mu-calculus model-checking case $\mathbb{M}$ is the poset representation of the system being checked, say $\mathfrak{M}$, and $\mathbb{T}$ is the poset representation of an $\mathcal{L}_\mu$ formula ϕ. Then, in both cases, the board $\mathfrak{D}$ where the games are played is a subset of the Cartesian product of $\mathbb{M}$ and $\mathbb{T}$.

In Chapter 2 we introduced a reduction from some models of concurrency, namely from Petri nets and event structures, to TSI models and used TSIs to define a uniform representation for all such models of concurrency. Let us now define a further reduction from TSI models, and therefore from Petri nets and event structures as well, to another basic representation that is adequate for playing concurrent games. Such a uniform representation is a poset representation of these systems which can be used to recognize computation traces of infinite behaviour. More importantly, as shown later on, when dealing with finite-state systems this new uniform representation can also be given a finite poset representation, and therefore allows for the development of a number of game-based decision procedures on partial orders.

From Partial Order Models to Posets. Let (S, s_0, T, I, Σ) be a concurrent system $\mathfrak{M}$. Sassone, Nielsen, and Winskel [85] showed that $\mathfrak{M}$ can be unfolded into an event structure $\mathcal{E} = (E, \preccurlyeq, \sharp, \eta, \Sigma)$ (almost) in the same way that a safe Petri net can be unfolded into a prime event structure, as done by Nielsen, Plotkin, and Winskel [71]. Then, such an event structure can be translated into a poset of the configurations of $\mathcal{E}$ (cf. [98]). Such a 'configuration structure' is actually an edge-labelled event structure $\mathfrak{U} = (\mathcal{C}, \preccurlyeq, \sharp, \eta, \Sigma)$, where $\mathcal{C}$ is the set of configurations of $\mathcal{E}$, the causality relation $\preccurlyeq$ is the subset inclusion order $\subseteq$, and $\sharp$ is the conflict relation on configuration states induced by that on events, i.e., forall $e_1, e_2 \in E$ if $e_1 \in q_1 \in \mathcal{C}$ and $e_2 \in q_2 \in \mathcal{C}$ and $e_1 \sharp e_2$, then $q_1 \sharp q_2$. So, the main difference between $\mathcal{E}$ and $\mathfrak{U}$ is that in $\mathfrak{U}$ the events are "occurrences of states" rather than occurrences of events. Accordingly, the domain of the labelling function η of $\mathfrak{U}$ is $\mathcal{C} \times \mathcal{C}$ rather than only E (as in a normal labelled event structure) since, clearly, in this new unfolding construction one has to label the elements of the successor relation given by $\preccurlyeq$ rathen than the elements of $\mathcal{C}$.

The structure we consider here is an extension of $\mathfrak{U}$ that allows to recover the information about the cycles in $\mathfrak{M}$, which is lost when looking only at the unfolded structure $\mathfrak{U}$. Recall that any unfolding construction defines an unfolding map f_u from the elements of the initial

structure (which we call the 'kernel structure') to the elements of the unfolded one (which we call the 'replicated structure') as well as an equivalence relation $\sim_{f_u}$ on the elements of the replicated structure which identifies different elements of it that are the unfolding of the same element in the kernel structure. Then if we consider the unfolding construction from $\mathfrak{M}$ to $\mathfrak{U}$ we have that $f_u : S \to 2^{\mathcal{C}}$ and for all $q_1, q_2 \in \mathcal{C}$ we have that $q_1 \sim_{f_u} q_2$ if, and only if, there exists some $s \in S$ such that $q_1, q_2 \in f_u(s)$, i.e., the elements q_1 and q_2 are two different *occurrences* in the unfolding of the same element s in the kernel structure.

This information can be used to define a *recursion* relation $\circlearrowleft$ that identifies cycles, i.e., recursive behaviour, in the kernel structure. Such a relation is defined as follows: for all $u, v \in \mathcal{C}$ we have that $u \circlearrowleft v$ iff $v \preccurlyeq u$ and $\exists w \in \mathcal{C}.\ u \rightarrow_{\mathcal{C}} w \wedge v \sim_{f_u} w$. Using this new binary relation, let us define a poset structure which provides a uniform poset representation of the models of concurrency we have consider in this thesis: Petri nets, event structures, and TSI models.

Definition 5.32. A *labelled recursive poset structure* $\mathbb{M}$ is a tuple $(\mathfrak{Q}, \eta, \Sigma, \circlearrowleft, \#_{\mathcal{Q}})$, where $\mathfrak{Q} = (\mathcal{Q}, \leq_{\mathcal{Q}})$ is a $\bot_{\mathcal{Q}}$-bounded poset, Σ is a set of labels, $\eta : \mathcal{Q} \times \mathcal{Q} \to \Sigma$ is a labelling function, $\circlearrowleft \subseteq \mathcal{Q} \times \mathcal{Q}$ is a 'recursion' binary relation such that if $q, r \in \mathcal{Q}$ and $q \circlearrowleft r$ then $r \leq_{\mathcal{Q}} q$, and $\#_{\mathcal{Q}} \subseteq \mathcal{Q} \times \mathcal{Q}$ is an irreflexive and symmetric 'conflict' binary relation such that if $q_1, q_2, q_3 \in \mathcal{Q}$ and $q_1 \#_{\mathcal{Q}} q_2 \leq_{\mathcal{Q}} q_3$, then $q_1 \#_{\mathcal{Q}} q_3$. ◁

With the previous definition, the following reduction from concurrent systems to posets can be defined. Let (S, s_0, T, I, Σ) be a system $\mathfrak{M}$ whose unfolding [85] is the (unlabelled) event structure $\mathfrak{U} = (\mathcal{C}, \preccurlyeq, \sharp)$. We say that the labelled recursive poset structure $\mathbb{M} = (\mathfrak{Q}, \eta, \Sigma, \circlearrowleft, \#_{\mathcal{Q}})$ is the *poset representation* of $\mathfrak{M}$ iff $\mathfrak{Q} = (\mathcal{Q}, \leq_{\mathcal{Q}})$ is the poset $(\mathcal{C}, \preccurlyeq)$, the conflict relation $\#_{\mathcal{Q}}$ is $\sharp$, the set of labels Σ is as in $\mathfrak{M}$, and the recursive relation $\circlearrowleft$ as well as the labelling function η are defined with respect to the (state component $f_u : S \to 2^{\mathcal{Q}}$ of the) unfolding map in the following way: we have that $u \circlearrowleft v$ if, and only if, v is the smallest element of $\mathcal{Q}$ which satisfies that $v \leq_{\mathcal{Q}} u$ and $\exists w \in \mathcal{Q}.\ u \rightarrow_{\mathcal{Q}} w \wedge v \sim_{f_u} w$; and for all $(q, r) \in (\rightarrow_{\mathcal{Q}} \cup \circlearrowleft)$ we have that $\eta(q, r) = \delta(t)$ where $t \in T$, and $q \in f_u(\sigma(t))$, and $r \in f_u(\tau(t))$. Taking this reduction into account, let us show how to use the CLG model to represent a bisimulation checking problem.

5.3.1 Bisimulation

The usual presentation of a bisimulation problem is not given by a class of games closed under dual games. However, if we consider the more general problem of equivalence-checking, i.e., being able to ask whether two systems are bisimilar or are not bisimilar explicitly, then the game becomes closed under dual games.

Assume we are dealing with two labelled concurrent systems $\mathfrak{M}_1$ and $\mathfrak{M}_2$, and let $\mathbb{M} = (\mathcal{M}, \leq_{\mathcal{M}}, \eta_{\mathcal{M}}, \Sigma_{\mathcal{M}}, \circlearrowleft_{\mathcal{M}}, \#_{\mathcal{M}})$ and $\mathbb{T} = (\mathcal{T}, \leq_{\mathcal{T}}, \eta_{\mathcal{T}}, \Sigma_{\mathcal{T}}, \circlearrowleft_{\mathcal{T}}, \#_{\mathcal{M}})$ be their labelled recursive poset structures; moreover, extend the definition of the successor relation $\rightarrow$ on posets to its la-

belled version $\xrightarrow{a\in\Sigma}$ in the obvious way. In order to avoid confusion, the relations of different posets are marked with the name of the corresponding poset; for instance, $\xrightarrow{a}_{\mathcal{M}}$ for $\mathbb{M}$ and $\xrightarrow{a}_{\mathcal{T}}$ for $\mathbb{T}$. Also, we write $\text{EQ}(\mathfrak{M}_1,\mathfrak{M}_2,\sim_{sb})$ for a bisimulation checking problem $\text{V}(\text{P})$, and $\text{EQ}(\mathfrak{M}_1,\mathfrak{M}_2,\not\sim_{sb})$ for its dual $\text{V}(\text{P}^{op})$, or simply $\sim$ and $\not\sim$ if the two concurrent systems and equivalences are obvious from the context.

Note that EQ satisfies parts 1 and 3 of Property 5.20 (ω-symmetry) since, on the one hand, the problem—or its game representation—is now, by definition, closed under dual games, and on the other hand, it satisfies part 3 too because bisimilarity is the greatest bisimulation relation between two systems, a property that is mu-calculus definable, and hence parity definable. Moreover, part 2 of Property 5.20 and Property 5.23 (local correctness) are ensured with the constructions, i.e., the three sets of rules, given below:

1. Rules for the construction of the (pre)board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$. Let $\mathcal{D}$ be the least poset, whose bottom element is $\bot_{\mathcal{D}} = (\bot_{\mathcal{M}}, \bot_{\mathcal{T}})$, such that:
   - if $(s,q) \in \mathcal{D}$ & $s \rightarrow_{\mathcal{M}} e$ & $\nabla((s,q)) = \nabla(\bot_{\mathcal{D}})$, then $(e,q) \in \mathcal{D}$ & $((s,q),(e,q)) \in \rightarrow_{\mathcal{D}}$
   - if $(s,q) \in \mathcal{D}$ & $q \rightarrow_{\mathcal{T}} d$ & $\nabla((s,q)) = \nabla(\bot_{\mathcal{D}})$, then $(s,d) \in \mathcal{D}$ & $((s,q),(s,d)) \in \rightarrow_{\mathcal{D}}$
   - if $(s,d) \in \mathcal{D}$ & $s \xrightarrow{a}_{\mathcal{M}} e$ & $\nabla((s,d)) \neq \nabla(\bot_{\mathcal{D}})$ & $\exists q \in \mathcal{T}.\ q \xrightarrow{a}_{\mathcal{T}} d$, then $(e,d) \in \mathcal{D}$ & $((s,d),(e,d)) \in \rightarrow_{\mathcal{D}}$
   - if $(e,q) \in \mathcal{D}$ & $q \xrightarrow{a}_{\mathcal{T}} d$ & $\nabla((e,q)) \neq \nabla(\bot_{\mathcal{D}})$ & $\exists s \in \mathcal{M}.\ s \xrightarrow{a}_{\mathcal{M}} e$, then $(e,d) \in \mathcal{D}$ & $((e,q),(e,d)) \in \rightarrow_{\mathcal{D}}$

   provided that there is no $p \in \mathcal{D}$ where the ideal $\downarrow p$ contains two elements (u,v) and (m,n) for which either $u \#_{\mathcal{M}} m$ or $v \#_{\mathcal{T}} n$, i.e., all chains are conflict-free!
2. Rules for the polarisation function ∇:
   $\nabla(\bot_{\mathcal{D}}) = \forall$ (resp. $\exists$) , if EQ is $\sim$ (resp. $\not\sim$)
   $\nabla(d') = \forall$ (resp. $\exists$) , if $d \rightarrow_{\mathcal{D}} d'$ & $\nabla(d) = \exists$ (resp. $\forall$)
3. Rules for a complete set of winning conditions $\mathcal{W}$:
   $\mathcal{W}(\hbar) = \exists$ (resp. $\forall$) , if $\texttt{inf}(\hbar)$ & EQ is $\sim$ (resp. $\not\sim$)
   $\mathcal{W}(\hbar) = \exists$ (resp. $\forall$) , if $\neg\texttt{inf}(\hbar)$ & EQ is $\sim$ (resp. $\not\sim$) & $\exists d \in \hbar.\ d \not\rightarrow_{\mathcal{D}}$ & $\nabla(d) = \forall$ (resp. $\exists$)

   where $\texttt{inf} : \Gamma \rightarrow \{\textbf{true},\textbf{false}\}$ is a predicate characterising open, infinite plays.

The poset generated with the rules given above is not yet a valid polarised board since it does not satisfy that if $\texttt{SP}(d)$ then $|\, d^{\rightarrow} \,| = 1$ and $\forall e \in d^{\leftarrow}.|\, e^{\rightarrow} \,| = 1$. Thus, in order to play the game we need to add a few elements to $\mathfrak{D}$, so that the condition holds. Graphically, we need to define the two maps depicted in Figure 5.3.

Then, as the reader can see (from Figure 5.3), the following transformations on $\mathfrak{D}$, which we denote by ":=", do the intended job:

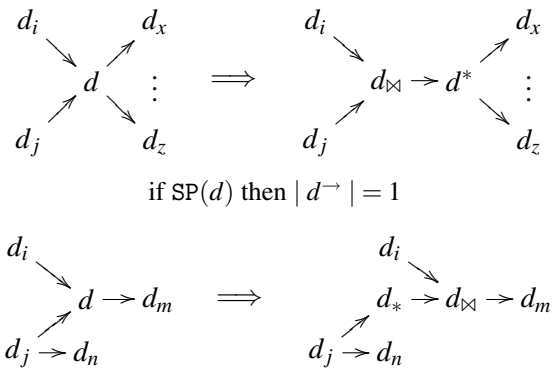


Figure 5.3: Maps which guarantee that if $\mathtt{SP}(d)$ then $|\, d^{\rightarrow} \,| = 1$ and $\forall e \in d^{\leftarrow}.|\, e^{\rightarrow} \,| = 1$.

1. if $\mathtt{SP}(d)$ and $|\, d^{\rightarrow} \,| > 1$, then:
   - $\mathcal{D} := \mathcal{D} \cup \{d_\bowtie, d^*\} \setminus \{d\}$
   - $\rightarrow_{\mathcal{D}} := \rightarrow_{\mathcal{D}} \cup \{(d_\bowtie, d^*)\} \cup \{(u, d_\bowtie) \mid u \rightarrow_{\mathcal{D}} d\} \cup \{(d^*, v) \mid d \rightarrow_{\mathcal{D}} v\}$
     $\setminus \{(u,v) \mid d = u \vee d = v\}$
   - $\nabla(d^*) = \nabla(d)$
2. if $\mathtt{SP}(d)$ and $\exists e \in d^{\leftarrow}.|\, e^{\rightarrow} \,| > 1$, then:
   - $\mathcal{D} := \mathcal{D} \cup \{d_*, d_\bowtie\} \setminus \{d\}$
   - $\rightarrow_{\mathcal{D}} := \rightarrow_{\mathcal{D}} \cup \{(e, d_*), (d_*, d_\bowtie)\} \cup \{(u, d_\bowtie) \mid u \rightarrow_{\mathcal{D}} d \wedge u \neq e\}$
     $\cup \{(d_\bowtie, v) \mid d \rightarrow_{\mathcal{D}} v\} \setminus \{(u,v) \mid d = u \vee d = v\}$
   - $\nabla(d_*) = \nabla(e)$

Now the poset $\mathfrak{D}$ is a valid polarised board, and the following statement holds:

Theorem 5.33. (Correctness) *Let* $\mathrm{V}(\mathrm{P})$ *be a bisimulation checking problem. The CLG* $\Game_{\mathrm{P}}$ *associated with* $\mathrm{V}(\mathrm{P})$ *is sound, complete, and determined.*

Proof. Showing that the game $\Game_{\mathrm{P}}$ is closed under dual games and have a complete set of winning conditions is immediate from the construction of $\mathrm{V}(\mathrm{P})$ given above. So, we only have to show that the rules to construct $\mathfrak{D}$ are locally correct.

First, suppose the two systems being compared are bisimilar. Then, Eve cannot lose in $\bot_{\mathcal{D}}$ because $\nabla(\bot_{\mathcal{D}}) = \forall$; and, by hypothesis, she cannot lose in any local position d such that $d \not\rightarrow_{\mathcal{D}}$, because in all such local positions it must be the case that $\nabla(d) = \forall$ as well. Then, $\bot_{\mathcal{D}}$ and all local positions such that $d \not\rightarrow_{\mathcal{D}}$ are locally correct. Finally, let $\nabla(d) = \exists$; in this case, Eve must respond to a choice just made by Adam (due to the initial strictly alternating construction of the (pre)board), and then, by hypothesis, there exists some d' such that $d \rightarrow_{\mathcal{D}} d'$ that Eve can choose. Thus, she cannot lose in any of those positions either, and hence, all such positions

must be locally correct as well. The case when the systems are not bisimilar is dual. Therefore, the conditions for applying the Theorems 5.28 and 5.29 hold, and the result follows. □

Although this concurrent bisimulation game is sound, complete, and determined, up to this point one cannot effectively decide which of the two players of the game has a winning strategy since a board can be infinitely large. However, as shown by McMillan [63], one can construct a finite, though "complete", representation of a possibly infinitely large poset structure modelling the behaviour of a concurrent system by using the unfolding technique [25].

The unfolding technique introduced by McMillan was initially defined for a class of Petri nets (cf. Chapter 6 of McMillan's PhD thesis [63]), but later extended to many other models of concurrency, chiefly by Esparza and collaborators (see [25] for a good introduction). Such a technique constructs a partially ordered structure which is finite and has the property that it contains all the states and transitions of the 'reachability graph' associated with the initial (folded) system. This partially ordered structure is called a *finite complete prefix*.

The construction of a finite complete prefix (as done in [63]) is based on the recognition of a set of so-called 'cut-off' events/actions/transitions which do not introduce any new behaviour to the prefix already constructed. This set of cut-off elements relate different states of the initial system and are the only sources of infinite behaviour. In our setting, the set of cut-off events is the set of elements that define the recursive relation $\circlearrowleft$, in the obvious way: if $t = u \rightarrow w$ and t is a cut-off event/action/transition, then there exists some state v such that $u \circlearrowleft v$ such that, by the definition of cut-off events, w and v represent the same state in the original system and moreover v belongs to the prefix already constructed.

Therefore, in order to play the concurrent bisimulation game defined before, one only needs to consider a board which contains no more than the pairs of states that correspond to the elements in the Cartesian product of the finite complete prefixes of the two systems $\mathfrak{M}_1$ and $\mathfrak{M}_2$ under consideration. More precisely, let $\mathfrak{D}'$ be the smallest finite board whose terminal elements are either the terminal elements of the original board $\mathfrak{D}$ or 'cut-off local positions' $d = (s,q) \in \mathcal{D}$ such that there exists a local position $e = (m,n) \in \mathcal{D}$ for which $e \leq_{\mathcal{D}} d$ and either $s \circlearrowleft m$ and $q \circlearrowleft n$, or $s = m$ and $q \circlearrowleft n$, or $s \circlearrowleft m$ and $q = n$. And, since in this new board $\mathfrak{D}'$ (arguably, a finite complete prefix of $\mathfrak{D}$) all plays will be of finite length due to the cuts, then let the predicate `inf` be redefined as follows: for any play $\hbar$ in $\mathfrak{D}'$ we have that $\mathtt{inf}(\hbar)$ iff for all terminal elements $d = (s,q)$ of $\hbar$ we have that d is a cut-off local position of $\mathfrak{D}$.

Proposition 5.34. (Termination) *Let* $\mathrm{V}(\mathrm{P})$ *be a bisimulation checking problem for two finite systems* $\mathfrak{M}_1$ *and* $\mathfrak{M}_2$. *The winner of* $\Im_{\mathrm{P}}$ *can be decided in finite time.*

Proof. Follows from Theorem 5.31 since the two systems $\mathfrak{M}_1$ and $\mathfrak{M}_2$ can be given finite poset representations as described before. □

5.3.2 Model-Checking

In order to construct a CLG for $\mathcal{L}_\mu$ model-checking we need to give a poset representation $\mathbb{T}$ of $\mathcal{L}_\mu$ formulae. We do so by representing $\mathcal{L}_\mu$ formulae as event structures. The construction also uses the subformulae relation given by the Fischer–Ladner (FL) closure of $\mathcal{L}_\mu$ formulae (cf. Chapter 2) and the semantic equivalence between the denotations of fixpoint variables and the denotations of their unfoldings.

Thus, we first define a poset representation of a $\mathcal{L}_\mu$ formula, and then show how to construct a concrete CLG for this verification problem. We write MC($\mathfrak{M}$,ϕ,$\models$) for a $\mathcal{L}_\mu$ model-checking problem V(P), and MC($\mathfrak{M}$,ϕ,$\not\models$) for its dual V(P^{op}), which is equivalent to model-checking the negated formula $\neg\phi$; similar to the bisimulation case, we write $\models$ and $\not\models$ if the the system and $\mathcal{L}_\mu$ formula are obvious from the context.

Let $Sub(\phi)$ be the subformula set of a $\mathcal{L}_\mu$ formula ϕ. According to the FL closure of $\mathcal{L}_\mu$ formulae, at the end of the (tree) unfolding of an $\mathcal{L}_\mu$ formula there are only fixpoint variables. Let $\mathcal{F}$ be the set of all subformulae in the infinite unfolding of an $\mathcal{L}_\mu$ formula and $\sqsubseteq_\mu$ the partial order on such a set, such that $\psi \sqsubseteq_\mu \phi$, i.e., ψ is a subformula of ϕ, iff $Sub(\psi) \subseteq Sub(\phi)$. Moreover, let $\#_\mu^\vee$ be a binary relation on $\mathcal{L}_\mu$ formulae, such that $(\psi_1,\psi_2) \in \#_\mu^\vee$ iff there exists ϕ such that $\phi = \psi_1 \vee \psi_2$. Now, define $\#_\mu$ as the irreflexive and symmetric conflict relation on $\mathcal{L}_\mu$ formulae such that, as usual, if $\psi_1,\psi_2,\psi_3 \in \mathcal{F}$ and $\psi_3 \sqsubseteq_\mu \psi_2 \#_\mu^\vee \psi_1$, then $\psi_3 \#_\mu \psi_1$.

Definition 5.35. (Partial order $\mathcal{L}_\mu$ specifications) Let ϕ be an $\mathcal{L}_\mu$ formula. A poset model of ϕ is a poset $\mathfrak{T} = (\mathcal{F}, \sqsubseteq_\mu)$ such that $\mathcal{F}$ is the set of subformula sets in the unfolding of ϕ, and $\sqsubseteq_\mu$ is the subformula set inclusion ordering given by the FL closure of $\mathcal{L}_\mu$ formulae such that any two occurrences Z_1^i and Z_2^i of the same fixpoint variable Z at the same unfolding level i are mapped to the same subformula set $\{Z^i\}$ if, and only if, $(Z_1^i, Z_2^i) \notin \#_\mu$. ◁

Lemma 5.36. ($\mathcal{L}_\mu$ Poset specifications) *For any $\mathcal{L}_\mu$ formula ϕ there is a labelled event structure $\mathcal{E} = (E, \preccurlyeq, \sharp, \eta, \Sigma)$ that represents it. Moreover, to such an event structure a recursion relation $\circlearrowleft \subseteq E \times E$ can be associated. Then, a poset specification of an $\mathcal{L}_\mu$ formula is a node-labelled recursive poset structure.*

Proof. There is a simple reduction from the poset model of an $\mathcal{L}_\mu$ formula ϕ to a labelled event structure, as follows: let E be $\mathcal{F}$, $\preccurlyeq$ be $\sqsubseteq_\mu^{-1}$, and $\sharp = \#_\mu$. Moreover, let Σ be a set of formula labels, and η be a labelling function from elements of the set of subformula sets E to Σ. Then $\mathcal{E}$ is an event structure. Finally, the recursion relation $\circlearrowleft$ is defined with respect to the fixpoint variables since they are the only formulae that increase the size of an $\mathcal{L}_\mu$ formula, and therefore allow for infinitely large poset representations. So, let ψ be a fixpoint variable; we say that $\psi \circlearrowleft \varphi$ iff φ is the smallest (resp. biggest) subformula with respect to $\preccurlyeq$ (resp. $\sqsubseteq_\mu$) such that $\varphi \preccurlyeq \psi$ (resp. $\psi \sqsubseteq_\mu \varphi$) and, moreover, φ corresponds to the $\mathcal{L}_\mu$ formula to which ψ unfolds. □

Parities and Types. Since $\mathcal{L}_\mu$ model-checking is equivalent to solving a parity game, let $\chi : \mathcal{F} \to \Upsilon$ be a function *typing* the elements of $\mathfrak{T} = (\mathcal{F}, \sqsubseteq_\mu)$ as expected: $\exists$ is assigned to disjunctions, diamond modalities, greatest fixpoint operators, and their corresponding fixpoint variables; and $\forall$ is assigned to the dual operators. Using this information, ∇ can be defined according to χ. On the other hand, in order to define $\mathcal{W}$, one has to assign parities to $\mathcal{D}$, which can be done in the usual way through the parities associated with the formulae in $\mathcal{F}$.

Let $\kappa : \mathcal{D} \to \mathbb{N}$ be a function that assigns natural numbers to local positions according to the $\mathcal{L}_\mu$ formula associated with a local position $d \in \mathcal{D}$, this is, even numbers iff $\nabla(d) = \exists$, odd numbers iff $\nabla(d) = \forall$, and the priorities respect the ranks of the fixpoint operators in the usual way (as in a parity game for $\mathcal{L}_\mu$ model-checking). Also, let $\mathtt{rnk} : \Gamma_\infty \to \mathbb{N}$ be a function from the set of infinite plays Γ_∞ to the least priority seen infinitely often in a given play $\hbar$ according to κ. Since $\mathtt{rnk}$ is undefined for finite plays, it is therefore a partial function on Γ.

Finally, a concurrent game for $\mathcal{L}_\mu$ model-checking is easily shown to be closed under dual games since $\mathcal{L}_\mu$ is closed under negation and, moreover, it is equivalent to a parity game, i.e., the set of winning conditions is parity definable. Therefore, parts 1 and 3 of Property 5.20 are satisfied. Let us now show the construction of the other parts of the CLG model for $\mathcal{L}_\mu$ model-checking, namely of $\mathfrak{D}$, ∇, and $\mathcal{W}$. Thus, for a verification problem MC($\mathfrak{M}$,ϕ,$\models$), let $\mathbb{M} = (\mathcal{S}, \leq_\mathcal{S}, \eta_\mathcal{S}, \Sigma_\mathcal{S}, \circlearrowleft_\mathcal{S}, \#_\mathcal{S})$ be the poset representation of $\mathfrak{M}$ and $\mathbb{T} = (\mathcal{T}, \leq_\mathcal{T}, \eta_\mathcal{T}, \Sigma_\mathcal{T}, \circlearrowleft_\mathcal{T}, \#_\mathcal{T})$ be the (node-labelled) poset representation of ϕ:

1. The construction of $\mathfrak{D} = (\mathcal{D}, \leq_\mathcal{D})$, whose bottom element is $\bot_\mathcal{D} = (\bot_\mathcal{S}, \bot^0_\mathcal{T})$, is given by the rules that determine the satisfaction relation of $\mathcal{L}_\mu$ formulae (see Chapter 2 or [14]). As in the bisimulation case, $\mathcal{D}$ is the least poset such that:
   - if $(s, \{Z^i\}) \in \mathcal{D}$ & ${}^{\nu}_{\mu}Z^i.\phi$, then $(s, \{\phi^{i+1}\}) \in \mathcal{D}$ & $((s, \{Z^i\})(s, \{\phi^{i+1}\})) \in \to_\mathcal{D}$
   - if $(s, \{{}^{\nu}_{\mu}Z^i.\phi\}) \in \mathcal{D}$, then $(s, \{\phi^i\}) \in \mathcal{D}$ & $((s, \{{}^{\nu}_{\mu}Z^i.\phi\})(s, \{\phi^i\})) \in \to_\mathcal{D}$
   - if $(s, \{\phi_1 \wedge^i \phi_2\}) \in \mathcal{D}$, then $(s, \{\phi^i_1\}) \in \mathcal{D}$ & $((s, \{\phi_1 \wedge^i \phi_2\}), (s, \{\phi^i_1\})) \in \to_\mathcal{D}$ & $((s, \{\phi_1 \wedge^i \phi_2\}), (s, \{\phi^i_2\})) \in \to_\mathcal{D}$
   - if $(s, \{[a]^i \phi\}) \in \mathcal{D}$ & $s \to^a_\mathcal{S} s'$, then $(s', \{\phi^i\}) \in \mathcal{D}$ & $((s, \{[a]^i \phi\}), (s', \{\phi^i\})) \in \to_\mathcal{D}$

   and likewise for '$\vee$' and '$\langle a \rangle$'; moreover ${}^{\nu}_{\mu} \in \{\mu, \nu\}$
2. Rules for the polarisation function ∇:
   $\nabla((s,q)) = \forall$ (resp. $\exists$) , if MC is $\models$ (resp. $\not\models$) & $\chi(q) = \forall$ (resp. $\exists$)
   $\nabla((s,q)) = \exists$ (resp. $\forall$) , if MC is $\models$ (resp. $\not\models$) & $\chi(q) = \exists$ (resp. $\forall$)
   provided that $|\ (s,q)^{\leftarrow}\ | = 1$, i.e., that (s,q) is not a synchronization point.
3. Rules for the complete set of winning conditions $\mathcal{W}$:

$\mathcal{W}(\hbar) = \exists$ (resp. $\forall$) , if $\texttt{inf}(\hbar)$ & MC is $\models$ (resp. $\not\models$) & $\texttt{rnk}(\hbar)$ is even (resp. odd)

$\mathcal{W}(\hbar) = \exists$ (resp. $\forall$) , if $\neg\texttt{inf}(\hbar)$ & MC is $\models$ (resp. $\not\models$) & $\exists d \in \hbar.\ d \not\rightarrow_{\mathcal{D}}$ & $\nabla(d) = \forall$ (resp. $\exists$)

$\mathcal{W}(\hbar) = \forall$ (resp. $\exists$) , if $\texttt{inf}(\hbar)$ & MC is $\models$ (resp. $\not\models$) & $\texttt{rnk}(\hbar)$ is odd (resp. even)

$\mathcal{W}(\hbar) = \forall$ (resp. $\exists$) , if $\neg\texttt{inf}(\hbar)$ & MC is $\models$ (resp. $\not\models$) & $\exists d \in \hbar.\ d \not\rightarrow_{\mathcal{D}}$ & $\nabla(d) = \exists$ (resp. $\forall$)

And, as in the bisimulation case, the elements of the board must satisfy that there is no $p \in \mathcal{D}$ where the ideal $\downarrow p$ contains two elements (u,v) and (m,n) for which either $u \#_{\mathcal{S}} m$ or $v \#_{\mathcal{T}} n$, i.e., all chains are conflict-free! The only final consideration is that in order for the board $\mathfrak{D}$ to satisfy the condition that if $\texttt{SP}(d)$ then $\mid d^{\rightarrow} \mid = 1$ and $\forall e \in d^{\leftarrow}.\mid e^{\rightarrow} \mid = 1$, we need to consider only $\mathcal{L}_\mu$ formulae in 'guarded form' [6] [54] as well as transformations which are similar to those used in the bisimulation case.

The reasons are that, due to the construction of poset $\mathcal{L}_\mu$ specifications, for all synchronization points $(s,\psi) \in \mathfrak{D}$ we have that ψ is always a fixpoint variable. Then, this ensures that $\forall e \in d^{\leftarrow}.\mid e^{\rightarrow} \mid = 1$, as the formula component of e can never be a boolean operator. Moreover, since a fixpoint variable unfolds to a unique $\mathcal{L}_\mu$ formula, it is always true that $\mid d^{\rightarrow} \mid = 1$. Since the conditions to instantiate metatheorem 5.28 hold, then we have the following result:

Theorem 5.37. (Correctness) *Let* V(P) *be a mu-calculus model-checking problem. The CLG* $\mathfrak{I}_{\mathrm{P}}$ *associated with* V(P) *is sound, complete, and determined.*

Proof. Again, Property 5.20 is satisfied because closure under dual games is given by the fact that $\mathcal{L}_\mu$ is closed under negation, the winning conditions of an $\mathcal{L}_\mu$ model-checking problem are parity definable, and the rules given before ensure that the set of winning conditions given by $\mathcal{W}$ is complete. So, we only need to show that the rules for the construction of the board are locally correct, i.e., that Property 5.23 also holds.

The local correctness condition follows from the fact that the elements in $\rightarrow_{\mathcal{D}}$ that relate different local positions are exactly the rules that define the semantics of $\mathcal{L}_\mu$ formulae, i.e., the one-step rules for the satisfaction relation $\models$ of $\mathcal{L}_\mu$, which are necessarily locally correct. Then, given a local position (s,ψ), we can ensure that the rules to define any (s',ψ') such that $(s,\psi) \rightarrow_{\mathcal{D}} (s',\psi')$ are locally correct by checking all possible cases for ψ following the usual case analysis for $\mathcal{L}_\mu$ formulae [14].

Firstly, regardless of whether a local configuration is false or true, if it corresponds to a fixpoint formula or a fixpoint variable, then it unfolds to a unique $\mathcal{L}_\mu$ formula (in the same

[6] Roughly speaking, an $\mathcal{L}_\mu$ formula ϕ is in guarded form if every occurrence of a fixpoint variable in ϕ is within the immediate scope of a modal operator; since any $\mathcal{L}_\mu$ formula can be translated into an equivalent one in guarded form, considering only formulae in guarded form is by no means a restriction.

state) and therefore true or false is preserved—i.e., neither Adam nor Eve has any influence on the truth value of the new configuration; the same happens if the local configuration is a join in the poset, which, according to our constructions, has to correspond to a local configuration associated with a fixpoint variable.

Now, suppose that a local configuration is true and it is none of the previous cases. Then, if it is assigned to Adam, it must correspond to a conjunction or a box modality (whenever MC is $\models$) and since the local configuration is true then either all successor local positions that eventually synchronize with independent choices of Eve are true as well (because Eve can preserve truth). As a consequence Adam must preserve truth as well. This is the reason why preservation of the existence of join is needed in a concurrent setting. Now, if, on the other hand, such a true local configuration has been assigned to Eve, then it corresponds to a disjunction or a diamond modality and hence she can simply choose the successor that makes the next global configuration true as well; therefore, truth is also preserved in this case.

The case when the local configuration is false is dual (by exchanging values and roles above). Also, if MC is $\not\models$ the arguments are the same but with the roles of Eve and Adam exchanged. As a result, the semantic local correctness condition holds, and Theorems 5.28 and 5.29 and Corollary 5.30 apply here as well. □

As in the bisimulation case, the board $\mathfrak{D}$ to be constructed for this verification problem can be bounded as well provided that $\mathbb{M}$ is the poset representations of a finite system. In this case, any infinitely large poset $\mathfrak{T}$ can be reduced to its finite poset representation which contains only terminal elements of $\mathfrak{D}$ or 'cut-off local positions' as the terminal elements of the new finite board where the game is played. It is well known that such a board is actually bounded by the size of the system $\mathfrak{M}$ or rather of its poset representation in this setting. In more traditional techniques this upper bound can be used to ensure that the number of fixpoints approximants that must be calculated before knowing that either a greatest fixpoint has been satisfied or a least fixpoint has failed has been reached, and therefore that no more computations are needed.

Moreover, as in the bisimulation checking, the recognition of infinite plays is redefined in the same way. And finally, the definition of the rank function $\mathtt{rnk}$, which depends only on plays of infinite length is redefined as follows: let $\hbar$ be a play such that $\mathtt{inf}(\hbar)$ holds. Then, due to the new definition of the function $\mathtt{inf}$ there exist two sets of local positions, namely $\mathtt{dom}(\circlearrowleft)$ and $\mathtt{codom}(\circlearrowleft)$, that characterise, respectively, the terminal elements of the new board $\mathfrak{D}'$ that allow for infinite behaviour and the returning points to previous elements in the new board. Since the priorities between the elements of such sets are the ones that are seen infinitely often, then $\mathtt{rnk}$ has to evaluate to the least number in such a set. More precisely, it has to evaluate to $\mathtt{min}\{\kappa(d) \mid d \in \hbar \cap (\uparrow \mathtt{codom}(\circlearrowleft) \cap \downarrow \mathtt{dom}(\circlearrowleft))\}$. Then, one has the following result:

Proposition 5.38. (Termination) *Let* $\mathrm{V}(\mathrm{P})$ *be a mu-calculus model-checking problem for a finite system* $\mathfrak{M}$*. The winner of* $\ni_{\mathrm{P}}$ *can be decided in finite time.*

Proof. Follows from Theorem 5.31 and the effective construction of the finite poset representations given above. □

Example 5.39. (A CLG model for $\mathcal{L}_\mu$ model-checking) Let us finish this section with an example of a CLG model for $\mathcal{L}_\mu$ model-checking. Let $\mathfrak{M}$ be the Petri net in Figure 5.4 and $\mathbb{T}$ a partial order representation of the $\mathcal{L}_\mu$ formula $\mu Z.\,[-]Z \vee \langle c\rangle \mathrm{tt}$. The poset $\mathfrak{D}$ in Figure 5.5 is the board of the CLG $\Game_{\mathrm{P}}$ associated with $\mathrm{MC}(\mathfrak{M},\phi,\models)$. The type $\{\exists,\forall\}$ given by ∇ is shown as a superscript, whereas the parity given by κ is shown as a subscript in each element of $\mathfrak{D}$.

Notice that Eve and Adam can play concurrently in the (sub)chains that are independent in $\mathfrak{D}$. Eve wins the game, and her winning choices are defined by the poset $\downarrow(\{c\},Y)^{\exists}_{2} \cup \uparrow(\{c\},Y)^{\exists}_{2}$. In Figure 5.5, the superscripts of $\mathcal{L}_\mu$ formulae are omitted. Moreover, we only depict explicitly the board $\mathfrak{D}$ that is generated if one considers the recursive poset structures associated with the poset representations of the Petri net $\mathfrak{M}$ and the $\mathcal{L}_\mu$ formula ϕ. Notice that independent subgames can be analysed independently, and therefore distributed in practice. Moreover, some subgames are joined and, as a consequence, need not be re-evaluated. ◁

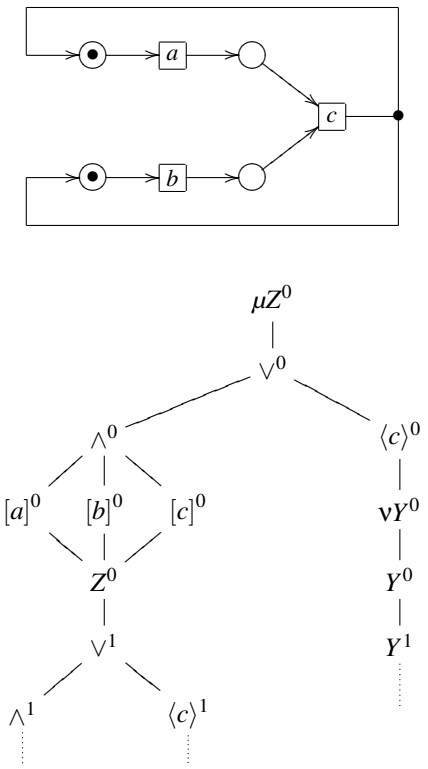


Figure 5.4: A Petri net $\mathfrak{M}$ and the poset representation of the $\mathcal{L}_\mu$ formula $\phi = \mu Z.\,[-]Z \vee \langle c\rangle \mathrm{tt}$. We only depict the main operators in ϕ and omit the conflict relation $\#_\mu$ beneath the logical disjunctions $\vee^i$, for $i \in \mathbb{N}$. In this case, the poset is depicted downwards, i.e., the bottom element of the poset is μZ^0.

Figure 5.5: The board $\mathfrak{D}$ of the CLG $\Game_{\mathsf{P}}$ for $\mathsf{MC}(\mathfrak{M},\phi,\models)$. The states of $\mathfrak{M}$ are represented by sets of ('principal') events in $\mathfrak{M}$.

5.4 Prime Concurrent Games

A powerful feature of the CLG model, which in turn makes it easier to analyse, is that each global position D of a board $\mathcal{D}$ determines the history $\hbar$ of the game up to such a position; this is, the union of all order ideals given by each local position $d \in D$. What is not determined is the set of local strategies, and hence global strategy, that has been used to reach such a position.

This is due to the fact that two different local strategies λ and λ' may behave the same up to some point in the game, say up to a local position d, but operate differently after it. For instance, suppose that two different local strategies λ and λ' are defined at d. Notice that although for both strategies could have been used to reach d, it is also possible that $\lambda(d) = e$ and $\lambda'(d) = e'$, such that $e \neq e'$; otherwise reaching global positions different from singleton sets, i.e., playing concurrently, would not be possible.

This fact, namely that a local position d determines its own 'history' $\downarrow d$, explains why in a partial order setting, unlike for games on graphs, a strategy can be defined from a single position to a next position of the game, rather than from the whole history of a play to a next position, i.e., to a set of nodes in the graph, without really depending on whether the strategy is positional or not in the traditional sense. This feature of games on posets raises a natural question: 'what exactly is a history-free winning strategy in a concurrent game on a poset?'. A useful fact for answering this question is the following result for usual sequential games:

Fact 5.40. (Closure under history-free strategies) *The union of history-free strategies is a history-free strategy.*

Then, in order to answer the question given before, one needs to introduce a notion that formally ensures that a local strategy really is history-free (positional or memoryless). Let us consider an equivalence relation $\sim$ on positions of a game board $\mathfrak{D}$ given by a quotient set $(\mathcal{D}/\sim)$. We say that a local strategy λ is history-free or *memoryless* if, and only if, for all $d, d' \in [d_0]_\sim$, if $\lambda(d) = e$ and $\lambda(d') = e'$, then $e, e' \in [e_0]_\sim$, where d_0 and e_0 are two representative local positions of the equivalence classes under consideration.

In this way, even though $\downarrow d \neq \downarrow d'$, we can ensure that λ behaves uniformly on all elements of the equivalence class $[d_0]_\sim$, i.e., independently of the history of the game. This means that under this restriction and with respect to the quotient set $(\mathcal{D}/\sim)$, a local strategy λ defined at d really depends *only* on d rather than on the whole order ideal $\downarrow d$. It should be easy to see that in our setting a local strategy is memoryless, in the traditional setting, if we let $\sim$ be the equivalence relation on elements of an unfolding, i.e., if $\sim \equiv \sim_{f_u}$.

A poset $\downarrow d$ is sometimes called a prime ideal (instead of an order ideal) and d its 'prime' element. Thus, since a memoryless local strategy λ must behave uniformly in all $d, d' \in [d_0]_\sim$, then the prime elements of a board can be used to characterise the memoryless strategies for the game. Since such strategies are desirable, let us define the following subclass of logic games: a

'prime concurrent game' is the subclass of concurrent games on partial orders where the local strategies in $\Lambda_{\mathfrak{D}}$ are memoryless, i.e., characterised by the prime elements in $\mathfrak{D}$. From a practical viewpoint, further investigations into problems that can be reduced to prime concurrent games are desirable since, by the definition of prime concurrent games, the following holds:

Fact 5.41. *Prime concurrent games have memoryless winning strategies, if any.*

5.5 Summary

We have defined a *sound*, *complete*, and determined concurrent logic game model which can be played directly on *partial orders*, and showed that in such a setting different decision problems—which include bisimulation and model-checking—and concurrent systems can be analysed uniformly. This approach makes easier the design of different logic games for verification by focusing on simple *local* correctness conditions, and is well suited for reasoning about partial order models, such as Petri nets, event structures, or TSI models, since poset representations are considered rather than interleaving graph representations.

This chapter *generalises* to a partial order setting previous work on games both for bisimulation and for $\mathcal{L}_\mu$ model-checking, in a uniform way, while embracing well-known traditional sequential techniques. Moreover, through the use of concurrent games we have provided an implicit method for *reducing* the state space to be analysed since independent subgames (and their sub-boards) are analysed only once; as a consequence, some—though not all—irrelevant interleavings are avoided, and identical subgames are not unnecessarily re-evaluated.

Also, the lift to a partial order setting allowed us to define some metatheorems that can provide, in a number of cases, *reusable* solutions for different problems and systems by using general and powerful order-theoretic techniques, which, to the best of our knowledge, had not been previously investigated in order to formalize a concurrent logic game for verification.

Chapter 6

Conclusions and Further Work

The results of this thesis are related to three connected topics: mu-calculi, bisimulation equivalences, and model-checking problems; then, the conclusions, most relevant related work, and ideas for further research are given accordingly. As we were interested in mu-calculi as fixpoint extensions of modal logic as well as in bisimulation and model-checking problems from a games perspective, our results mainly relate to work on these topics with respect to partial order models. However, as our framework also embraced interleaving systems, at times, pointers to similar work in the interleaving context are given along with our concluding remarks.

6.1 Logics with Partial Order Models

Our work on mu-calculi can be related to modal and temporal logics with partial order models at large, not only to mu-calculi for true concurrency. Work on logics with partial order models dates back to the 1980's soon after the introduction of temporal logics for verification. In the most traditional approaches, formulae of logics with partial order semantics were usually given denotations that considered the one-step interleaving semantics of a particular partial order model. Following this approach no new logical constructions had to be introduced; unfortunately, in this case, the explicit notion of concurrency in the models is completely lost.

As a result, a common solution when defining logics with partial order models was to introduce operators that somehow capture the independence information on the partial order models. In most cases that kind of logical independence is actually a sequential interpretation of concurrency, which is based on the introduction of past operators sensitive to concurrent transitions and a mixture of forwards and backwards reasoning; however, this can lead to undecidability results with respect to the decision problems related to such logics, e.g., with respect to their satisfiability, equivalence, or model-checking problems, cf., [70, 75, 76, 78].

Several logics with the characteristics described above whose semantics are given using partial order models (as well as their related decision problems) can be found in [76, 78], and

the references therein. Other logics with partial order semantics that do not appear there can be found, e.g., in [6, 70], but the literature includes many, many more references. It is worth saying that not all such logics are extensions of modal logic or even other kinds of mu-calculi. In some cases, they are variations of usual temporal logics, such as LTL, CTL, and CTL*.

More recently, partly motivated by preliminary results of this thesis (presented in [39] and contained in Chapters 3 and 4), Baldan and Crafa [7] introduced a logic for true concurrency where independence is captured without the use of backwards modalities. Instead, they introduced of a novel logical characterisation of the concepts of *observation* and *execution* over the elements of an event structure. Their interest was to define a logic to capture several equivalences for true concurrency; they succeeded in different directions as some syntactic fragments of their logic capture various equivalences for (true) concurrency: strong, step, pomset, hp, and hhp bisimilarity. Therefore, the equivalence induced by the full logic is undecidable [52].

At a more philosophical level, our study on logics for true concurrency is also similar to that of Bradfield and Fröschle [10, 12]—a work primarily on mathematical logic using game logics for true concurrency. Bradfield and Fröschle's main goal was to explicitly capture what we call *model independence*, i.e., explicit concurrency in the models, in a logical way with the use of 'Henkin quantifiers' (which are partial order generalisations of the usual quantifiers in classical logic). More precisely, in [10] different properties of a number of fixpoint logics based on Hintikka and Sandu's 'Independence-Friendly' (IF) logic [47] are discussed, and in [12] the bisimilarity induced by one of such logics, namely of 'IF modal logic' (IFML), is thoroughly studied. Their main motivation closely relates to ours, especially because of their interest in the bisimulation equivalences induced by such logics as well as the use of games.

However, the mu-calculi original to this thesis have mathematical foundations different from all the examples above. Here, we have given a logical characterisation to the *dualities* that are found when analysing *locally* the relationships between concurrency and conflict as well as concurrency and causality. This characterisation aims at defining connections between equivalences that take into account the notion of independence when considering partial order semantics, and which can be defined at the level of the models (then, capturing a *model* independence) as well as at the level of the logics (then, capturing a *logical* independence).

A key ingredient of our work on logic is that we allowed a free interplay of *fixpoints* and *local monadic second-order* power in the mu-calculi we have presented. Our results, together with the analysis of some of the related work, suggest that restricting the quantification power to *conflict-free* sets (of transitions) in partial order models may be a sensible/plausible way of retaining decidability while still having a high expressivity. In fact, as our mu-calculi are at least as expressive as $\mathcal{L}_\mu$ in an interleaving context, nothing is lost with respect to the main approaches to logics for concurrency with interleaving semantics. Instead, logics and techniques for interleaving concurrency are extended to a partial order setting in our framework.

6.2 Logics for Local Reasoning

The mu-calculi developed here can be related to logics for local reasoning. In particular, the use of *separation* properties for local reasoning has been investigated elsewhere and successfully applied in many settings. Separation as disjointness of resources was an idea introduced to computer science in the 1970's in order to reason independently about program components. However, the "revolution" came many years later when this idea was recast by Reynolds *et al.* [73, 83] to address the problem of verifying the correctness of programs with pointers.

Since then, due to the success of this approach, a great deal of work has been done in different areas by applying the notion of separation of resources to reason independently about different systems. For instance, to specify safety properties of concurrent programs [15, 44], to verify while programs with no concurrent behaviour [86], to reason about proofs [81], to model resource-sensitive processes [80], etc. At present, the literature includes many more examples.

In [73], some of these works are discussed and several open problems are presented. One of them, which had not been addressed until now, was that of using this kind of local reasoning to specify and verify both linear-time and branching-time temporal properties. In this thesis we do so. The task required a heavy use of a mixture of concepts of true concurrency and fixpoint theory, besides the introduction of a new concept of locality, namely that of *support sets*. Locality and concurrency are therefore connected in our framework in terms of specification.

6.3 On Bisimulation

Bisimulation equivalences have been intensively studied in concurrency in the last thirty years because they are 'observational equivalences', perhaps the most natural behavioural equivalence notions for concurrent systems. In this thesis we have argued that the problem of observing concurrency and nondeterminism, as initially studied by Milner and Hennessy [46] on interleaving models, can be refined to a problem of observing concurrency, causality, and conflict in a partial order context. Consequently, we studied bisimilarities that were introduced to reason about concurrent systems with partial order semantics. In particular, we focused our attention on the two strongest bisimilarities in [26, 35], namely on hpb [82] and hhpb [51].

Closely related to our results is the work of Joyal, Nielsen, and Winskel [51] on bisimulation from 'open maps'. Whereas in [51] they proposed a *categorical* approach to defining an abstract or model independent notion of bisimulation equivalence for several concurrent systems, here we have proposed a *logical* one, following the way of reasoning used in [46, 69], but in a partial order setting instead of in an interleaving one.

This was done by defining the semantics of our mu-calculi by an intermediate mathematical structure—namely through a *process space*—which was intended to be used as a common bridge between the particular models of concurrency under consideration. Then, two partial

order models, possibly of different kinds, can be compared within the same framework by comparing logically their associated process spaces. Thus, following this approach one can study different models of concurrency *uniformly*, even interleaving ones.

Moreover, the strongest bisimilarities introduced in this thesis are decidable, and thereby their associated logical and game characterizations. Nielsen and Clausen [70] have also studied logics and games for other bisimilarities for concurrency, namely of hhpb, a concretization of the abstract notion of bisimulation defined in [51]. Since hhpb is undecidable, computing the winner in such games is also an undecidable problem. However, our result holds over Ξ systems, whereas Nielsen and Clausen's work considers arbitrary systems without auto-concurrency. Presently, we do not know whether hhpb is decidable on the class of Ξ systems.

As mentioned before, Bradfield and Fröschle [12] also studied the equivalence induced by IFML using game-theoretic techniques. They followed a logical approach, which is in spirit quite close to our work. Unfortunately, albeit being a very interesting work, the bisimilarity induced by IFML did not coincide, in most cases, with the standard bisimilarities for partial order models, not even for classes of systems with very restricted concurrent behaviour.

Finally, although our results apply to classes of Petri nets, event structures, and TSI models, we believe that they also apply to other models of concurrency provided that in such models the local dualities we study here can be defined too. For instance, interleaving equivalences, say, for transition systems (and their unfoldings) appear as particular cases in our framework. Indeed, strong bisimilarity—the standard bisimulation equivalence for interleaving concurrency—is captured by both syntactic and semantics means in our framework. Therefore, nothing is lost with respect to the main approaches to bisimulation for interleaving concurrency.

6.4 On Model-Checking

In the past three decades model-checking has emerged as a remarkably powerful technique for verification which has had a big impact in practical applications. Not surprisingly, Clarke, Emerson, and Sifakis won in 2007 the 'Alan M. Turing award' (the "Nobel prize" in computer science) for their seminal work and contributions to the development of model-checking, recognizing in this way how important this verification technique has become nowadays.

Indeed, several techniques (not only game-theoretic ones) for model-checking concurrent systems both with interleaving and with partial order semantics have been studied elsewhere; see [17, 25, 32, 36, 37, 55, 63, 89] for many different examples. As our main motivation was to develop games (and induced decision procedures whenever decidability followed) to analyse concurrent systems with partial order semantics, only the techniques considering these kinds of systems directly relate to our work, though, as mentioned before, not all such techniques are game-theoretic. Thus, let us talk a little more about games approaches to model-checking.

As shown in [37, 97], model-checking games have been studied for both theoretical and practical reasons in the last few years. For instance, in order to formally pin down their logical and mathematical properties [38, 55, 90] or to construct tools for verification, e.g., [33, 88]. Most approaches based on games have considered either interleaving models or the one-step interleaving semantics of partial order models.

Our work differs from these approaches in that we deal with games played on partial order models rather than on interleaving structures. This difference makes the landscape algorithmically harder since in a partial order setting most model-checking problems are computationally more complex [24]—some of which even undecidable over finite concurrent systems [56]. The games developed in this thesis are all *decidable* in the finite case; moreover, although we did not study complexity issues, our games are clearly exponential and so not better than the best decision procedures for interleaving systems from a theoretical complexity point of view.

Regarding the temporal verification of event structures, previous studies have been done on restricted classes. Closer to our work in Chapter 4 is [56, 77]. Indeed, model-checking regular trace event structures has turned out to be rather difficult and previous work has shown that verifying MSO logic properties on these structures is already undecidable. For this reason weaker (classical, modal, and temporal) logics have been studied. Unfortunately, although very interesting results have been achieved, especially in [56] where CTL* temporal properties can be verified, previous approaches have not managed to define decidable theories for a logic with enough expressive power to describe all usual temporal properties as can be done with $\mathcal{L}_\mu$ in the interleaving case, and hence with SFL and $\mathbb{L}_\mu$ when considering partial order models.

Recall that one of the reasons why $\mathcal{L}_\mu$ is more expressive than CTL* is that $\mathcal{L}_\mu$ can express properties about "moments" in computation paths, whereas CTL* in general cannot do so. Similarly, one can think of simple properties that talk about moments in traces (i.e., posets) of partial order models. Those kinds of properties are not expressible in logics whose expressive power equals that of CTL* on interleaving models, e.g., [67]. This means that there are temporal true concurrency properties that are not definable with logics (over partial order models) whose temporal expressive power on traces equals that of CTL* on (infinite) trees or graphs.

The main difference between the logics and decision procedures in [56] and the approach we followed here is that in [56] a *global* second-order quantification on conflict-free sets in the partial order model is permitted, whereas only a *local* second-order quantification in the same kind of sets is defined here, but such a second-order power can be embedded into fixpoint specifications, which in turn allows one to express more temporal properties. In this way we were able to improve, in terms of temporal expressive power, previous results on model-checking infinite, regular trace event structures against a branching-time logic for true concurrency.

Finally, our results on model-checking (along with those on bisimulation) suggest that there is hope in developing a very general approach to verification, as we have defined a *unified*

framework, for both model-checking and bisimulation, which applies to several kinds of concurrent systems, not only those with partial order semantics—an approach that is investigated in more detail in Chapter 5. Indeed, this claim is also supported by the fact that in different cases we got almost for *free* various decision procedures for interleaving concurrency.

6.5 Sequential & Concurrent Games

Most games have been defined to be played sequentially, i.e., it is not the case that two players make a move at the same time because either they take turns or alternate their actions in the game. This setting has changed in the last years so as to address some problems in concurrency.

For instance, Abramsky and Melliès [1, 64] defined various classes of concurrent and asynchronous games in which players can have strategies where several moves can be made independently and at the same time. Such games are, however, not for verification; their main application is in the area of formal programming language semantics. Also, Henzinger *et al.* [4, 5] have studied concurrent games. In this case, such games are played on so-called 'concurrent game structures', which are graphs where several players can interact synchronously in order to model the behaviour of reactive systems regarded as 'open' ones. These concurrent games, and also variants of them, have been used to study ω-regular properties of reactive systems [4] as well as to give semantics to a number of logics for multi-agent systems, e.g., [5].

This thesis also studies concurrent games and, in particular, provides alternative, order-theoretic foundations of the mathematics of concurrent games with respect to interleaving approaches—e.g., with respect to the work presented in [4] and related concurrent games played on interleaving structures. Whereas concurrent games on graphs provide a natural framework for reasoning about systems with interleaving semantics, the CLG model developed here was especially designed so that it could be used directly in a partial order setting.

In fact, when considering a CLG on (the unfolding of) a graph structure or an infinite tree, the game no longer needs to be concurrent. However, in this case, the CLG model is a generalization of sequential games played on such structures, e.g., it generalises Stirling games for bisimulation and $\mathcal{L}_\mu$ model-checking [90] (and therefore also related tableau-based techniques). Roughly, the generalization is similar to the way that infinite partially ordered structures generalize infinite trees (as tableaux and sequential games can be seen as potentially infinite trees). As in most parts of this thesis, this was done by keeping the information about *independence* and *locality* in the partial order models of concurrency that we considered here.

It is also worth saying that the CLG model is inspired by the semantic concurrent games model for Linear Logic [34] of Abramsky and Melliès [1] above mentioned. However, the mathematics of the model presented in [1] have been drastically reformulated (in the quest towards the answer of *algorithmic* questions), and only a few technical features were kept.

6.6 Further Work

There are still many questions to be asked and answered for concurrent systems with partial order semantics. Some of them already existed before this work and some others have been risen more recently due to the development of this thesis.

Logic. We believe that the problem of looking for *the* logic for concurrency is perhaps as difficult as looking for *the* model of concurrency. And, indeed, several authors regard some particular logic as natural for concurrency whenever such a logic fits (mathematically) well with a particular model of concurrency that is considered as natural already. Thus, we believe, the answer for the question as to whether there is a natural logic for concurrency will be open until some agreement can be reached with respect to what the model of concurrency should be. However, some work can still be done with the aim to find a natural logic for concurrency. One direction could be to look for a logic with as many of the following properties as possible:

- size: small in the number of basic constructs, but highly expressive;
- syntax: with operators for locality and independence (i.e., for true concurrency);
- semantics: provided with models that allow for local and modular reasoning;
- equivalence: preserved by composition and decomposition of sub-systems;
- decidability: achievable in as many models of concurrency as possible;
- complexity: hopefully not too difficult to check in real applications (though the theoretical complexity of the usual decision problems is expected to be high);
- usability: intuitive for humans, so either not a fixpoint logic as, in general, humans find mu-calculi difficult to understand or with a suitable "syntactic sugar".

In this thesis we developed logics which try to capture some of the properties above mentioned, but much work remains to be done. New and better logical formalisms can be studied based on the experience we have gain with the development of this work—especially through the use of support sets but in other logical contexts.

Another interesting idea is the study of *independence logics*, such as the family of IF logics of Hintikka and Sandu in restricted settings. Bradfield and Fröschle [12] showed that in some classes of systems IFML captures a very strong bisimulation equivalence for true concurrency. It would be nice to see what happens when the very permissive notion of independence of IFML is semantically restricted in order to capture only the information about concurrency that a suitable support set of IFML formulae gives. Some inspiration can also be drawn from the recent work of Baldan and Crafa [7] on a logic for true concurrency—where several equivalence relations for concurrency are given a logical characterization.

A final idea is to look for a *logic for posets* as an indirect strategy. In this case, the logic should allow for reasoning about fundamental elements of posets, e.g., about meets and joins

of particular kinds. To the best of our knowledge this idea has not been investigated yet as posets have been thoroughly studied from an algebraic point of view, but not from a logical one. This third avenue of further research seems to be a reasonable way to get some insights into what the logic for posets should be like, and thereby, the logic for partial order models.

Bisimulation. A lesson to learn from the development of this work is that a potentially interesting bisimulation equivalence for concurrency *must* be composable and decomposable with respect to "prime" components of concurrent systems, i.e., a congruence with respect to a suitable set of operators for concurrency, causality, and conflict. There are already several equivalences for concurrency that are preserved by refinement (causality and some forms of concurrency) and conflict (nondeterministic choices or branching behaviour), e.g., history-preserving bisimilarities, but it is far more difficult to find equivalences which are congruences with respect to operators for arbitrary forms of communication (i.e., for interdependent concurrency).

There are two main reasons for this: firstly, that communication is defined in very different ways in different formalisms for concurrency; in fact, the way a particular language or model of concurrency defines its communication mechanisms drastically affects the way a natural equivalence for such a model or language is defined (e.g., bisimilarity for Milner's CCS processes [66]). Due to this, we wonder what the most natural notion of equivalence, say, for Petri nets or for event structures is; some experts in the field believe that such an equivalence could be hhpb, though the author's personal view is that the issue is still open. One of the reasons to believe this is that a decidable notion, e.g., n-hhpb [29] or a similar bisimilarity, is algorithmically more adequate and hence computationally more attractive.

The second reason is that there is not a definite answer for the question of what a "prime" concurrent module should be. Fröschle [28] has already addressed this problem and given a initial answer; her work found interesting theoretical results as well as applications to the analysis of bisimilarities for true concurrency [28, 30]. However, the present author believes that further work in this direction should be done as research on this topic can produce very positive results. Again, the notion of support sets we introduced may shed light on new mathematical formulations of the concept of primality for concurrent components.

Another possibility for future work is the study of bisimulation equivalences for true concurrency with the concurrent logic games model introduced in Chapter 5. Time restrictions allowed us to make only a preliminary study of bisimilarities up to interleaving concurrency. Logic games for stronger bisimulation equivalences should be naturally definable in such a partial order setting, but this is yet work to be done.

A final issue is that of complexity features. Neither for bisimulation nor for model-checking complexity bounds were investigated, and this could be done in the years to come—though, as mentioned previously, such bounds are not expected to be low, and thus perhaps not very

interesting from a complexity viewpoint (unless restricted to "easy" classes of systems). In fact, the author's view is that for true concurrency complexity issues should be addressed mainly on restricted classes of systems of practical usability. An initial idea for such a study is to analyse the class of concurrent systems studied in Chapter 3 for which the global notion of causality is fully captured by a simpler local one. This class of systems can be of practical usability since it can model synchronization mechanisms as well as mutual exclusion protocols, arguably, the two most important kinds of control "devices" when studying communicating systems.

Model-Checking. As in the bisimulation case, one obvious avenue of further research is that of complexity issues, especially since new decision procedures have been found for special classes of partial order models. However, even more interesting, is to consider some problems related to the expressivity and decidability of, respectively, logics and their associated model-checking problems for particular classes of concurrent systems. Two initial ideas are the following: on the one hand, regarding expressivity issues, an interesting problem is to look for logics for true concurrency stronger than SFL and $\mathbb{L}_\mu$ but still with a decidable model-checking problem on hard families of partial order models. On the other hand, regarding decidability issues, another interesting problem is to look for easier (but still interesting) classes of systems for logics that are already known to be undecidable on certain partial order models of concurrency. In this direction Madhusudan *et al.* [57] have already done some work which can serve as a borderline for comparisons or as a different starting point for further investigations.

Another idea for future work is to analyse model-checking problems for concurrent systems with partial order semantics in a much simpler setting. In this thesis we focused on properties expressible with fixpoint modal logics that extend the mu-calculus, and therefore, rather powerful winning objectives could be defined since the very beginning. However, one could think of simpler winning conditions, such as Büchi, reachability, or safety, which, despite their simplicity, are quite interesting from a practical point of view. In these cases, even more complex classes of partial order models can be considered, e.g., 'parallel pushdown systems' [60] or 'higher-order dimensional automata' [79] (just to mention a few examples) which due to their richer expressivity are easy to have undecidable model-checking problems, cf., see [61] for a survey of various decidability results with respect to several models of concurrency.

A third avenue of further research is in adapting ideas for the definition of partial order methods and unfolding techniques for verification. These investigations should be carried out, especially, over the CLG model as in such a setting the games are played in posets. Perhaps more importantly, where we see better prospectus for further research, in terms of model-checking, is in adapting ideas of compositional reasoning that were originally studied by the games semantics community in a sequential context. Indeed, the CLG model was an attempt to do so, and some improvements have been foreseen already. Let us elaborate on this idea next.

From Semantic to Verification Games in Concurrency. Let us start by giving a brief explanation of what 'semantic games' are and how their underlying mathematics can be used as the basis for the definition of verification games, e.g., for bisimulation or model-checking. A semantic game is also played by two players; in a semantic game a computation is represented by the interaction, i.e., exchange of moves, between the two players. In this kind of games one of the players represents a program or a system and the other player represents its environment. Semantic games have been successfully used, in their sequential form, to provide highly precise models for a wide range of languages and programming features of interest to the semantics community. This approach is general enough to embrace different programming paradigms, including functional, imperative, and concurrent. This remarkable feature of semantic games distinguishes them from other approaches, which lack a comparable robustness.

Semantic games have also been used to define decision procedures for verifying temporal properties of systems, and in the last few years yielded algorithms for program analysis and verification. In particular, the semantic games approach focuses on the *observable* patterns of behaviour exhibited by a system when interacting with its environment, rather than in the explicit representation of the program's state space. As a consequence, program components intended to live within a concurrent and distributed environment can be modelled as simple 'open systems' and verified following a compositional approach. This feature suggests that well-known techniques for the verification of concurrent systems (especially with partial order semantics) can be enhanced by the use of ideas initially developed with semantic purposes.

This final idea for further work is, therefore, about the development of decision procedures for a number of key questions about concurrent games, perhaps building on the work presented in Chapter 5. Such questions relate to the existence and computability of winning strategies: e.g., 'is the game determined?'; if so, 'are winning strategies computable?'; and if so, 'how can we compute such winning strategies?'. As one would be looking for winning strategies with finite partial order representations, such strategies may be considerably smaller than those for traditional sequential games, and hence very attractive from a synthesis point of view.

6.7 Epilogue

In this Ph.D. thesis we have argued for the naturalness of *partial order* semantics in the unified study of different models and problems in concurrency. We believe that the results herein help support the following idea: that a key to deeper our understanding of the *semantic* foundations of concurrency is the formal description of the notions of *locality* and *independence* in concurrent systems. We also think that a mathematical and logical study of these two notions can eventually provide the tools for the development of a *unified* theory of concurrency.

Bibliography

[1] Abramsky, S., Melliès, P.A.: Concurrent games and full completeness. In: LICS. 431–442, IEEE Computer Society (1999)

[2] Abramsky, S.: Sequentiality vs. concurrency in games and logic. Mathematical Structures in Computer Science **13**(4), 531–565 (2003)

[3] Aceto, L.: History preserving, causal and mixed-ordering equivalence over stable event structures. Fundamenta Informaticae **17**(4), 319–331 (1992)

[4] Alfaro, L.D., Henzinger, T.A.: Concurrent omega-regular games. In: LICS. 141–54, IEEE Computer Society (2000)

[5] Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM **49**(5), 672–713 (2002)

[6] Alur, R., Peled, D., Penczek, W.: Model-checking of causality properties. In: LICS. 90–100, IEEE Computer Society (1995)

[7] Baldan, P., Crafa, S.: A logic for true concurrency. In: CONCUR. LNCS **6269**, 147–161, Springer (2010)

[8] Benthem, J.V.: Modal Correspondence Theory. Ph.D. thesis, University of Amsterdam (1977)

[9] Benthem, J.V.: Logic games, from tools to models of interaction. In: Logic at the Crossroads. 283–317, Allied Publishers (2007)

[10] Bradfield, J.: Independence: logics and concurrency. Acta Philosophica Fennica **78**, 47–70 (2006)

[11] Bradfield, J.C.: The modal μ-calculus alternation hierarchy is strict. Theoretical Computer Science **195**(2), 133–153 (1998)

[12] Bradfield, J.C., Fröschle, S.B.: Independence-friendly modal logic and true concurrency. Nordic Journal of Computing **9**(1), 102–117 (2002)

[13] Bradfield, J.C., Stirling, C.: Local model checking for infinite state spaces. Theoretical Computer Science **96**(1), 157–174 (1992)

[14] Bradfield, J.C., Stirling, C.: Modal mu-calculi. In: Handbook of Modal Logic. 721–756, Elsevier (2007)

[15] Brookes, S.D.: A semantics for concurrent separation logic. In: CONCUR. LNCS **3170**, 16–34, Springer (2004)

[16] Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching-time temporal logic. LNCS **131**, 52–71, Springer (1981)

[17] Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press (2000)

[18] Dam, M.: CTL* and ECTL* as fragments of the modal mu-calculus. Theoretical Computer Science **126**(1), 77–96 (1994)

[19] Degano, P., Nicola, R.D., Montanari, U.: A distributed operational semantics for CCS based on condition/event systems. Acta Informatica **26**(1/2), 59–91 (1988)

[20] Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge Tracts in Theoretical Computer Science **40**, Cambridge University Press (1995)

[21] Emerson, E.A.: The beginning of model checking: A personal perspective. In: 25 Years of Model Checking. LNCS **5000**, 27–45, Springer (2008)

[22] Emerson, E.A., Jutla, C.S., Sistla, A.P.: On model checking for the μ-calculus and its fragments. Theoretical Computer Science **258**(1-2), 491–522 (2001)

[23] Emerson, E.A., Lei, C.L.: Efficient model checking in fragments of the propositional mu-calculus. In: LICS. 267–278, IEEE Computer Society (1986)

[24] Esparza, J.: Decidability and complexity of petri net problems - An introduction. In: Petri Nets. LNCS **1491**, 374–428, Springer (1998)

[25] Esparza, J., Heljanko, K.: Unfoldings - A Partial-Order Approach to Model Checking. EATCS Monographs in Theoretical Computer Science, Springer (2008)

[26] Fecher, H.: A completed hierarchy of true concurrent equivalences. Information Processing Letters **89**(5), 261–265 (2004)

[27] Fröschle, S.B.: Decidability and Coincidence of Equivalences for Concurrency. Ph.D. thesis, Univeristy of Edinburgh (2004)

[28] Fröschle, S.B.: Composition and decomposition in true-concurrency. In: FOSSACS. LNCS **3441**, 333–347, Springer (2005)

[29] Fröschle, S.B., Hildebrandt, T.T.: On plain and hereditary history-preserving bisimulation. In: MFCS. LNCS **1672**, 354–365, Springer (1999)

[30] Fröschle, S.B., Lasota, S.: Causality versus true-concurrency. Electronic Notes in Theoretical Computer Science **154**(3), 3–18 (2006)

[31] Gale, D., Stewart, F.M.: Infinite games with perfect information. Annals of Mathematics Studies **28**, 245–266 (1953)

[32] Gerth, R., Kuiper, R., Peled, D., Penczek, W.: A partial order approach to branching time logic model checking. Information and Computation **150**(2), 132–152 (1999)

[33] Ghica, D.R.: Applications of game semantics: From program analysis to hardware synthesis. In: LICS. 17–26, IEEE Computer Society (2009)

[34] Girard, J.Y.: Linear logic. Theoretical Computer Science **50**, 1–102 (1987)

[35] Glabbeek, R.J.V., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Informatica **37**(4/5), 229–327 (2001)

[36] Godefroid, P.: Partial-Order Methods for the Verification of Concurrent Systems - An Approach to the State-Explosion Problem. LNCS **1032**, Springer (1996)

[37] Grädel, E.: Model checking games. Electronic Notes in Theoretical Computer Science **67**:15–34 (2002)

[38] Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games. LNCS **2500**, Springer (2002)

[39] Gutierrez, J.: Logics and bisimulation games for concurrency, causality and conflict. In: FOSSACS. LNCS **5504**, 48–62, Springer (2009)

[40] Gutierrez, J.: Logics and games for true concurrency. Informatics Report Series, Technical Report EDI-INF-RR-1393, University of Edinburgh (2010)

[41] Gutierrez, J.: Concurrent logic games on partial orders. In: WoLLIC. LNCS **6642**, 146–160, Springer (2011)

[42] Gutierrez, J., Bradfield, J.C.: Model-checking games for fixpoint logics with partial order models. In: CONCUR. LNCS **5710**, 354–368, Springer (2009)

[43] Gutierrez, J., Bradfield, J.C.: Model-checking games for fixpoint logics with partial order models. Information and Computation **209**(5), 766–781 (2011)

[44] Hayman, J., Winskel, G.: Independence and concurrent separation logic. Logical Methods in Computer Science **4**(1) (2008)

[45] Henkin, L.: Some remarks on infinitely long formulas. In: Infinitistic Methods. 167–183, Pergamon Press (1961)

[46] Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. Journal of the ACM **32**(1), 137–161 (1985)

[47] Hintikka, J., Sandu, G.: A revolution in logic? Nordic Journal of Philosophical Logic **1**(2), 169–183 (1996)

[48] Huth, M., Ryan, M.: Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press (2004)

[49] Janin, D., Walukiewicz, I.: On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In: CONCUR. LNCS **1119**, 263–277, Springer (1996)

[50] Jategaonkar, L., Meyer, A.R.: Deciding true concurrency equivalences on safe, finite nets. Theoretical Computer Science **154**(1), 107–143 (1996)

[51] Joyal, A., Nielsen, M., Winskel, G.: Bisimulation from open maps. Information and Computation **127**(2), 164–185 (1996)

[52] Jurdzinski, M., Nielsen, M., Srba, J.: Undecidability of domino games and hhp-bisimilarity. Information and Computation **184**(2), 343–368 (2003)

[53] Kanellakis, P.C., Smolka, S.A.: CCS expressions, finite state processes, and three problems of equivalence. Information and Computation **86**(1), 43–68 (1990)

[54] Kozen, D.: Results on the propositional mu-calculus. Theoretical Computer Science **27**, 333–354 (1983)

[55] Lange, M.: Games for Modal and Temporal Logics. Ph.D. thesis, Univeristy of Edinburgh (2002)

[56] Madhusudan, P.: Model-checking trace event structures. In: LICS. 371–380, IEEE Computer Society (2003)

[57] Madhusudan, P., Thiagarajan, P.S., Yang, S.: The MSO theory of connectedly communicating processes. In: FSTTCS. LNCS **3821**, 201–212, Springer (2005)

[58] Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer (1992)

[59] Martin, D.A.: Borel determinacy. Annals of Mathematics **102**(2), 363–371 (1975)

[60] Mayr, R.: Process rewrite systems. Information and Computation **156**(1-2), 264–286 (2000)

[61] Mayr, R.: Decidability of model checking with the temporal logic EF. Theoretical Computer Science **256**(1-2), 31–62 (2001)

[62] Mazurkiewicz, A.W.: Introduction to trace theory. In: The Book of Traces. 3–42, World Scientific (1995)

[63] McMillan, K.: Symbolic Model Checking. Kluwer Academic Publishers (1993)

[64] Melliès, P.A.: Asynchronous games 2: The true concurrency of innocence. Theoretical Computer Science **358**(2-3), 200–228 (2006)

[65] Milner, R.: A Calculus of Communicating Systems. LNCS **92**, Springer (1980)

[66] Milner, R.: Communication and Concurrency. Prentice-Hall (1989)

[67] Moller, F., Rabinovich, A.M.: On the expressive power of CTL*. In: LICS. 360–369, IEEE Computer Society (1999)

[68] Nicola, R.D., Ferrari, G.L.: Observational logics and concurrency models. In: FSTTCS. LNCS **472**, 301–315, Springer (1990)

[69] Nicola, R.D., Vaandrager, F.W.: Three logics for branching bisimulation. Journal of the ACM **42**(2), 458–487 (1995)

[70] Nielsen, M., Clausen, C.: Games and logics for a noninterleaving bisimulation. Nordic Journal of Computing **2**(2), 221–249 (1995)

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

[72] Nielsen, M., Winskel, G.: Models for concurrency. In: Handbook of Logic in Computer Science. 1–148, Oxford University Press (1995)

[73] O'Hearn, P.W.: Resources, concurrency, and local reasoning. Theoretical Computer Science **375**(1-3), 271–307 (2007)

[74] Park, D.M.R.: Concurrency and automata on infinite sequences. LNCS **104**, 167–183, Springer (1981)

[75] Penczek, W.: On undecidability of propositional temporal logics on trace systems. Information Processing Letters **43**(3), 147–153 (1992)

[76] Penczek, W.: Branching time and partial order in temporal logics. In: Time and Logic: A Computational Approach. 179–228, UCL Press (1995)

[77] Penczek, W.: Model-checking for a subclass of event structures. In: TACAS. LNCS **1217**, 145–164, Springer (1997)

[78] Penczek, W., Kuiper, R.: Traces and logic. In: The Book of Traces. 307–390, World Scientific (1995)

[79] Pratt, V.R.: Modeling concurrency with geometry. In: POPL. 311–322, ACM (1991)

[80] Pym, D.J., Tofts, C.M.N.: A calculus and logic of resources and processes. Formal Aspects of Computing **18**(4), 495–517 (2006)

[81] Pym, D.J., O'Hearn, P.W., Yang, H.: Possible worlds and resources: the semantics of BI. Theoretical Computer Science **315**(1), 257–305 (2004)

[82] Rabinovich, A.M., Trakhtenbrot, B.A.: Behavior structures and nets. Fundamenta Informaticae **11**, 357–403 (1988)

[83] Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: LICS. 55–74, IEEE Computer Society (2002)

[84] Sangiorgi, D.: On the origins of bisimulation and coinduction. ACM Transactions on Programming Languages and Systems **31**(4) (2009)

[85] Sassone, V., Nielsen, M., Winskel, G.: Models for concurrency: Towards a classification. Theoretical Computer Science **170**(1-2), 297–348 (1996)

[86] Sims, É.J.: Extending separation logic with fixpoints and postponed substitution. Theoretical Computer Science **351**(2), 258–275 (2006)

[87] Smith, E.: On the border of causality: Contact and confusion. Theoretical Computer Science **153**(1&2), 245–270 (1996)

[88] Stevens, P., Stirling, C.: Practical model-checking using games. In: TACAS. LNCS **1384**, 85–101, Springer (1998)

[89] Stirling, C.: Local model checking games. In: CONCUR. LNCS **962**, 1–11, Springer (1995)

[90] Stirling, C.: Bisimulation, modal logic and model checking games. Logic Journal of the IGPL **7**(1), 103–124 (1999)

[91] Stirling, C.: Modal and Temporal Properties of Processes. Texts in Computer Science, Springer (2001)

[92] Stirling, C., Walker, D.: Local model checking in the modal mu-calculus. Theoretical Computer Science **89**(1), 161–177 (1991)

[93] Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics **5**(2), 285–309 (1955)

[94] Thiagarajan, P.S.: Regular trace event structures. BRICS Report Series, Technical Report RS-96-32, University of Aarhus (1996)

[95] Valmari, A.: The state explosion problem. In: Petri Nets. LNCS **1491**, 429–528, Springer (1998)

[96] Vogler, W.: Deciding history preserving bisimilarity. In: ICALP. LNCS **510**, 495–505, Springer (1991)

[97] Walukiewicz, I.: A landscape with games in the background. In: LICS. 356–366, IEEE Computer Society (2004)

[98] Winskel, G.: Events in Computation. Ph.D. thesis, University of Edinburgh (1980)

[99] Winskel, G.: Event structure semantics for CCS and related languages. In: ICALP. LNCS **140**, 561–576, Springer (1982)

List of Notations

	Sets/Models/Concurrency
$(A, \leq_A), \bot_A$	A poset and its bottom element
$\downarrow, \uparrow, \Downarrow, \circ$	Order ideal, order filter, restriction, composition
$[a]_\sim, (A/\sim)$	Equivalence class, quotient set ($\sim$ is an equivalence relation and $a \in A$)
$\mathcal{A} \to \mathcal{B}, A \to B$	Map between categories $\mathcal{A}$ and $\mathcal{B}$, function between sets A and B
$\oplus, \uplus, \wp, \bigsqcup$	Linear sum, disjoint union, powerset, and join operators
$\mathtt{min}, \mathtt{max}$	Minimal and maximal elements set operators
$\mathtt{dom}, \mathtt{codom}$	Domain and codomain
${}^\bullet n, n^\bullet, d^{\leftarrow}, d^{\rightarrow}$	Preset and postset (of a node n of a net and of an element d of a poset)
$\xrightarrow{t}, \xrightarrow{a}, \xrightarrow{e}$	Transition relations (Petri net actions, TSI transitions, and events)
$\longrightarrow^*$	Transitive closure of a transition relation (either kind)
$\mathsf{par}, I, \mathsf{co}$	Independence relations (Petri nets, TSI models, and event structures)
$\sim, \preccurlyeq, \sharp, \mathsf{cfs}, \circlearrowleft$	Equivalence, causal, conflict, confusion, and recursion relations
θ, η	Labelling functions on actions and events
$\sigma(t), \tau(t), \delta(t)$	Source, target, and label of a transition t
$\sim_{xb}$	Bisimilarity of type x (e.g., strong, hp, hhp, ihp, thp)
$\pi, \rho(\pi)$	Run (or sequence of transitions π) and last transition of π
$\otimes, \#, \ominus, \leq$	Local duality relations
$\mathcal{X}, \mathfrak{P}$	Sets of maximal supsets and support sets
$\sqsubseteq$	Inclusion ordering for complete supsets and support sets
$\mathfrak{X}(s)$	Maximal set at a state s
$\mathbb{S}, \mathfrak{S}$	A process space and a stateless maximal process space
	Logic
$\langle\,\rangle, [\,], \mu, \nu$	Logical operators (diamond,box, and least and greatest fixpoints)
$\langle\,\rangle_c, [\,]_c, \langle\,\rangle_{nc}, [\,]_{nc}$	Causal (c) and non-causal (nc) diamond and box modalities
$*, \bowtie, \langle\otimes\rangle, [\otimes]$	Structural operators and idempotent diamond and box modalities
$Sub(\phi), \sqsubseteq_\mu$	Subformula set of a fixpoint formula ϕ and its inclusion ordering
$\mathrm{Var}, \mathcal{V}$	Set of fixpoint variables and its valuation function
$\mathcal{V}[Z := Q]$	Updated valuation of a fixpoint variable $Z \in \mathrm{Var}$
Z^α	Fixpoint approximant indexed by an ordinal number α
$\phi[Z^\alpha/Z]$	Syntactic substitution
$\models$	Semantic satisfaction relation
$\sim_{\mathfrak{L}}$	Logical equivalence induced by a logic $\mathfrak{L}$
	Games
$\mathtt{SP}, \mathtt{BP}, \mathtt{sync}$	Predicates on elements of a game board
$\mathtt{dfn}, \mathtt{fix}$	Predicates on local strategies (defined, fixpoint)
$\mathtt{inf}, \mathtt{rnk}$	Predicates on plays (infinite,rank)
$\lambda^i_\exists, \lambda^j_\forall, \partial_\exists, \partial_\forall$	Local and global strategies for Eve ($\exists$) and Adam ($\forall$) in a CLG