Università
degli Studi
di Padova

Head Office: Università degli Studi di Padova
Department of General Psychology

Ph.D. Course in: BRAIN, MIND AND COMPUTER SCIENCE
Curriculum: Computer Science and Innovation for Societal Challenges
XXXII SERIES

Tableaux, Automata and Games for True Concurrency Properties

Coordinator: Prof. Giuseppe Sartori
Supervisor: Prof. Paolo Baldan
Co-Supervisor: Prof. Sara Mondini

Ph.D. student: Tommaso Padoan

September 2019

Abstract

In the formal verification of software systems, model-checking is one of the most studied and applied techniques. Systems represented by mathematical models are checked against properties expressed as logical formulae. When the subjects of the verification are concurrent and distributed systems, models and specification logics capable of faithfully capturing the concurrent features of computations, like causal dependency and independence between actions of the systems, can sometimes be convenient or even essential.

In this thesis we study the foundations of the model-checking problem in a logic for true concurrency, whose formulae predicate over executability of actions in computations and their causality and concurrency relations. The logic represents the logical counterpart of history-preserving bisimilarity, a popular behavioural equivalence in the true concurrent spectrum and one of the strongest known to be decidable. It includes least and greatest fixpoint operators in mu-calculus style, making it able to express properties of infinite computations. The logic is interpreted over event structures, a classical semantic model for true concurrency, that describes the behaviour of systems in terms of the events that can occur in computations and the causal dependencies and conflicts between them. It can be naturally used over any operational model that admits a causal semantic. Of particular interest, here, will be Petri nets. They are a widely adopted operational model allowing for a natural representation of concurrent and distributed systems.

We prove that the model-checking problem is decidable over a class of event structures satisfying a suitable regularity condition, and provide three decision procedures based on three different approaches. Along the lines of some classical work for the modal mu-calculus, we first develop a local model-checker in the form of a tableau system, for which we prove termination, soundness and completeness. The tableau system allows for a clean and intuitive proof of decidability, but a direct implementation of the procedure can be extremely inefficient. On the way to a practical implementation, we then present an automata-based model-checking technique. Given a formula and a model, a parity tree automaton is constructed whose language is non-empty if and only if the model satisfies the formula. The automaton is usually infinite. We discuss how it can be quotiented to a finite automaton preserving its language via a suitable equivalence over its states. We show how the method instantiates on finite safe Petri nets, where such equivalence can be effectively computed. Finally, we devise a general game-theoretic approach to the solution of systems of fixpoint equations over a wide class of lattices. This applies, in principle, to a multitude of verification tasks, which reduce to the computation of fixpoints of monotone functions. Being an instance of those tasks, we show how the method can be instantiated and applied to the model-checking problem of interest in the thesis.

Acknowledgements

To begin with, I want to especially thank my supervisor, Paolo Baldan, who masterfully guided and supported me during my PhD studies and, even before, in the development of my master's thesis. His help has been invaluable in all our projects and, particularly, for this thesis. Indeed, most results herein originate from discussions we had together, which, by the way, were among the most enticing research activities I have done. I would like to thank also my co-supervisor, Sara Mondini, whose academic advice and approval allowed me to follow my own path.

I am grateful to my external reviewers, Roland Meyer and Irek Ulidowski, for their careful reading of the thesis and their useful comments. Their remarks and suggestions have greatly helped to improve the final version.

A pivotal period was the time I spent abroad in Duisburg. I cannot thank enough Barbara König for her kind hospitality and guidance during my visit. I would like to thank also all the other members of her team, Christina Mika-Michalski, Benjamin Cabrera, Harsh Beohar, Dennis Nolte and Richard Eggert, for so many interesting and inspiring discussions on various topics, not limited to computer science. I am especially indebted to Barbara and Christina for their collaboration in the work on games.

I thank all my colleagues at BMCS for the pleasant conversations and mutual support we had. A special mention goes to Mirko Polato and Daniele Ronzani, who welcomed me when I first arrived at the office, and with whom I had the pleasure to share many "pause caffé" (even though I do not drink coffee) and meals.

I want to express my deepest gratitude to my parents whose financial, moral and loving support has allowed me to become what I am. They taught me how to face life one step at a time, never giving up, and encouraged me all along my journey. I would also like to remember my grandfather Giuseppe, who first directed me towards science in general by showing me the functioning of many electrical components and wiring and taught me how to repair (almost) anything.

Lastly, above all, I wish to thank Nadia, who always rooted for me and cheered me up over these demanding years. Especially during the last months, she tenderly took care of me and brought me peace in the stressful moments. Thank you for making my half whole.

Contents

Chapter 1

Introduction

Concurrent and distributed software systems have by this time become essential in our everyday lives. On the one hand, the relevance of parallel computing has been constantly growing in the last years and supposedly this trend will be confirmed in the future. The use of multicore/multiprocessor computers is nowadays the major means to increase computing power, and they are adopted even in small devices like smartphones. On the other hand, the diffusion of the internet, of mobile computing and network infrastructures has brought distributed software, both small applications and large systems, in uncountably many fields, from entertainment to medical devices, from home automation to industrial controls. Most people, even unknowingly, depend on their accuracy and reliability. However, in contrast to their sequential predecessors, concurrent and distributed systems, able to interact with each other and the environment, are notoriously hard to design, develop, and attest. Software testing of such systems often leads just to partial or imprecise knowledge of the system of interest. It is for these reasons that we believe that formal methods can provide a fundamental contribution in this setting, both for their reliability, as mathematically grounded, and their usability, as apt to be automated.

In this thesis we focus on the formal verification of the behaviour of concurrent and distributed systems. From the specification or the implementation of a system it is possible to derive a formal description of its behaviour in the form of a mathematical model. Different modelling languages allow to capture different features of the computation. In the theory of concurrency, models have been defined along two main different currents, distinguished by their approach to concurrent computations. This major dichotomy opposes so-called interleaving to partial order semantics. *Interleaving* models reduce concurrency to the non-deterministic choice amongst all possible sequentializations of the computations. Models following this approach include, for example, labelled transition systems [Kel76] and Kripke structures [Kri63]. Instead, models based on partial order semantics, also called *true concurrent* models, explicitly represent the causal and independence relations between actions performed by the systems, thus allowing to capture more faithfully the concurrent features of computations. Indeed, such models provide a mean for the study of the concur-

rent behaviour of systems at a more fundamental level. As it often happens in computer science, a price has to be paid for the greater expressiveness of true concurrent models. Indeed, with respect to the interleaving approach, many problems may become much harder to solve, if solvable at all, once set in a partial order semantic. Nevertheless, being able to express concurrency as a primitive notion, true concurrent models allow for a more compact representation of concurrent computations, mitigating the problem known as state space explosion (see [Val98]), which instead occurs more severely in the analysis and verification of concurrent systems with the interleaving approach, where the representation of concurrency via all the possible sequentializations can lead to an exponential blow up. In fact, interleaving models treat the different linearisations as entirely different computations, where instead true concurrent models recognise them for what they really are, that is, the same unique concurrent behaviour. In addition to this, the true concurrent approach provides tools for a deeper study of concurrent and distributed systems, which can be useful, if not essential, in some particular contexts, like, for example, security protocols, distributed software verification and system biology. Based on this motivations our work focuses on tools and techniques following the true concurrent approach.

Several true concurrent models have been defined, each one capturing various computational aspects and having slightly different expressiveness. Examples are Petri nets [Pet62], event structures [NPW81], asynchronous transition systems [Bed87, Shi85] and transition systems with independence [WN95]. We will mainly concentrate on event structures, as a reference semantic model, and on Petri nets, as a reference operational model.

Petri nets are a classical discrete formalism for concurrency and distribution, one of the most used in many fields, even beyond computer science. Petri nets describe systems in terms of the actions that they can perform, represented by so-called transitions, and the resources that each action requires in input and produces in output. The independence between actions is represented by the simultaneous presence of all the resources required by those actions, which can then be executed independently in parallel. Furthermore, Petri nets allow also to express conflicts between actions, that is, actions which, from a certain state, the system could perform but not both, as executing one excludes the other. Such actions are represented by transitions requiring the same resources, which can only be consumed by either one.

Event structures are a widely used semantic model for concurrency that describes the semantic behaviour of a system in terms of events. An event represents the occurrence of an action performed by the system, therefore the same action can be associated with several events, representing different occurrences of such action, with different past computations leading to them. Furthermore, because of this, event structures can be infinite even if they arise from finite state systems. The causal dependencies between events are expressed by means of a partial order. Conflicts are also present, expressed by another relation over events. Event structures and Petri nets have been shown to share an intimate relationship (see [NPW81, Thi02]), such that one model can often be translated, in a suitable way, in

the other.

In order to formally study the behaviour of a system, for example to assess its correctness, it is common practice to compare it with some specification, of the system itself, of some of its components, or of some property of its behaviour. A specification can be given as another model, other than that of the system, or as a logical property, characterising some desired specific behaviour. In the first case, behavioural equivalences can be used for checking the system model against the specification, in order to verify that the system behaves as desired. Depending on the context, different computational features of the behaviour may be relevant or not. For this reason several behavioural equivalences have been defined over the years (see, e.g., [vGG01]), which, like models, take into account different features of computations by abstracting the behaviour of systems. For interleaving semantics the most classical and studied behavioural equivalence is bisimilarity [Mil80], which compares system models only on the basis of what actions they can perform, so that two systems are bisimilar when they can match each other's actions, without taking into account the causal dependencies between the performed actions. Instead, true concurrent behavioural equivalences do take into account causal dependencies as well, each one to a different degree or in a different flavour. Two popular equivalences in the true concurrent spectrum are history-preserving bisimilarity [RT88] and hereditary history-preserving bisimilarity [Bed91]. Both take fully into account the causal dependencies between actions in computations, but only hereditary history-preserving bisimilarity precisely captures the interplay between conflicts, causality and concurrency of actions. Indeed, hereditary history-preserving bisimilarity is the finest equivalence in the true concurrent spectrum of [vGG01] and it has been shown to arise as a canonical equivalence across a range of different models for parallel computation [JNW96]. Its greater expressiveness, however, makes it usually unfeasible to work with. In fact, it has been shown that hereditary history-preserving bisimilarity is undecidable even for finite state systems [JNS03]. On the other hand, history-preserving bisimilarity has been shown to be decidable, e.g., for finite state Petri nets [MP97].

System models can also be tested against logical properties of their behaviour. Such properties are expressed by formulae of suitable behavioural logics. Several true concurrent logics have been introduced for defining causal properties of computations (see, e.g., [DNF90, NC95, Pen95, BF02]). Some logics are related, in a suitable sense, to behavioural equivalences. This happens for a logic and a behavioural equivalence when equivalent systems satisfy exactly the same formulae of the logic. In this case such a logic is also called the logical characterisation of the corresponding behavioural equivalence, which, conversely, is said to be the equivalence induced by that logic. Recently, the logical characterisation of true concurrent behavioural equivalences has received a renewed interest and corresponding logics have been introduced [BC14, PU14], interpreted over event structures. The expressiveness of such logics is sufficient to provide a logical characterisation of hereditary history-preserving bisimilarity, and suitably defined fragments correspond to coarser behavioural equivalences. In this thesis we focus on the logic referred to as $\mathcal{L}_{hp}$ in [BC14],

whose induced behavioural equivalence is history-preserving bisimilarity. The logic allows to predicate over executability of actions in computations and their dependency relations (causality and concurrency). Furthermore, the logic is endowed with least and greatest fixpoint operators in order to express interesting properties of infinite computations.

The formal verification of logical properties over models is usually done via so-called model-checking methods. *Model-checking* is the process of deciding whether or not a given system model satisfies a given property, expressed as a logical formula. Although model-checking problems are among the most studied in formal verification of software, unsolved issues still remain, especially in the case of true concurrent models and logics. For interleaving semantics, instead, the picture is more clear. Many methods have been defined for the solution of model-checking problems in expressive logics from the interleaving approach (see, e.g., [Cle90, SW91] for tableaux-based methods, [EJS01] for an automata-based method, and [Sti95] for a game-theoretic one). Model-checking procedures can be divided into global and local ones. *Global* model-checking procedures iterate on the structure of the formula to be verified exploring at each step the whole model, eventually computing the set of all states of the model from which the formula is satisfied. Then the formula is satisfied by the model in a specific state if the state is in such set. On the other hand, *local* model-checking procedures focus on just one state of the model, usually its initial state, and progressively explore only the part of the model necessary to establish whether the formula is satisfied in such state or not.

The aim of this thesis is to develop an extensive and foundational study of the model-checking problem in $\mathcal{L}_{hp}$, proving its decidability, previously unknown, and providing formal methods for its solution over some interesting class of event structures or other formalisms which can be given a causal semantics, like Petri nets. Note that, although history-preserving bisimilarity is known to be decidable, the model-checking problem in the corresponding logic is unrelated and non-trivial, even for finite state systems, since event structure models are typically infinite and the possibility of expressing properties that depends on the past often leads to undecidability [JNS03]. Moreover, for these reasons, we mainly focus on local approaches which are particularly suited in this setting characterised by the infiniteness of models.

1.1 Contributions and structure of the thesis

The main contribution of this work is represented by the development of a range of methods for the model-checking of true concurrency properties, expressed by formulae of the logic $\mathcal{L}_{hp}$, and beyond. The main results can be summarised as follows.

1. A decision procedure, based on the construction of semantic tableaux, for the solution of the model-checking problem in $\mathcal{L}_{hp}$ over a suitable class of event structures, so providing a proof of the decidability of the problem.

2. An automata-theoretic method for the model-checking of $\mathcal{L}_{hp}$, producing, when the model complies with some finitarity constraints, an effective and efficient procedure, applicable to formalisms like finite safe Petri nets.

3. A very general formal technique based on games for the solution of systems of fixpoint equations over a wide class of lattices, like those arising from a multitude of verification tasks, and, among them, the model-checking of $\mathcal{L}_{hp}$.

The thesis consists of two parts. Part I covers the background mathematical notions from the theories of computation and concurrency used throughout the second part, as such it does not contain original contributions, instead, it provides a range of definitions and reviews some results from the literature. Part II presents all the contributions of the thesis, that is, the three formal techniques devised and some extension based on the related work. The two parts are organised in the following manner.

Part I

Chapter 2 fixes the basic mathematical notations and gives the background material from the theory of domains playing a fundamental role in the modelling and verification of software systems. The concepts of partial order, lattice, monotone function and least and greatest fixpoints are therefore introduced, along with some classical results.

Chapter 3 reviews the main models for concurrency adopted in the thesis, namely, event structures and Petri nets. The relation between such models is made evident, leading to the definition of a classic regularity condition. The main behavioural equivalences over these models are also discussed, from both the interleaving and the true concurrent approach.

Chapter 4 introduces some classical modal logics from the literature, which can be seen as the natural progenitors of the history-preserving logic of interest in this work, whose syntax and semantics are also herein presented, along with some examples of logical properties. Notions about the alternation of fixpoints in formulae are then discussed, which are relevant in the design of model-checking procedures.

Part II

Chapter 5 presents the first contribution on the model-checking of $\mathcal{L}_{hp}$: a decision procedure for model-checking formulae of $\mathcal{L}_{hp}$ given in the form of a tableau system. The method is shown to be terminating, sound and complete over a suitable class of event structures, providing a constructive proof of the decidability of the associated model-checking problem.

Chapter 6 contains the second contribution. It first provides an abstract model-checking technique for $\mathcal{L}_{hp}$ based on infinite nondeterministic automata in the effort to design a more efficient procedure. It is shown that such automata can be quotiented, using a suitably defined equivalence over the states, producing a finite automaton accepting the same language

whenever the equivalence is of finite index. Then a concrete instantiation of the procedure on finite safe Petri nets, based on the definition of a suitable equivalence is discussed, along with the presentation of a tool implementing the technique.

Chapter 7 presents the third main contribution of the work. It focuses on systems of fixpoint equations over lattices, which often arise as the formal representation of some verification task. For such systems a general approach to the solution given in the form of a parity game is devised. The game is proved to be sound and complete over a vast class of lattices, and various techniques for its solution are discussed. Then, it is shown how such a general technique can be instantiated and applied to the model-checking of $\mathcal{L}_{hp}$ over event structures, translating formulae into systems of equations. Furthermore, the relation with the previous methods is explored.

Chapter 8 reviews some related works on other logics for true concurrency, comparing them with $\mathcal{L}_{hp}$. Then it is shown that $\mathcal{L}_{hp}$ and the techniques devised in the previous chapters can be extended to include operators from those logics which are inexpressible in $\mathcal{L}_{hp}$, demonstrating also the flexibility of the framework developed.

Chapter 9 is the conclusive chapter of the thesis, in which are described directions of future research and ideas, as well as some relevant and interesting problems which remain open in this area of study.

1.2 Origin of the chapters

The work presented in this thesis originates from a series of papers published in the last years. The tableau-based model-checking procedure in Chapter 5 was first presented in [BP17]. The automata-theoretic model-checking technique in Chapter 6 first appeared in [BP18]. The game for systems of fixpoint equations in Chapter 7 was presented in [BKMMP19]. Finally, the study of the other true concurrent logics, the extension of $\mathcal{L}_{hp}$ and the model-checking of the extended logic in Chapter 8 has been presented in [Pad18].

Part I

Background

Chapter 2

Preliminaries

In this chapter we present the mathematical background relevant to the rest of the work. First we fix the notation for the basic mathematical concepts which will be extensively used in the subsequent chapters. Then we present some elements of order and domain theory which have a fundamental role in the modelling and verification of software systems, in particular of concurrent and distributed systems, as those of interest for this thesis.

2.1 Basic notation

Symbols for the representation of sets, relations, functions, sequences and tuples will be hereafter introduced, along with their definitions and some operations. They are completely standard, hence they can be skipped by the reader and used as a reference.

Sets

Informally a *set* is an unordered collection of elements, usually represented by an upper case letter. Given two sets S and S', we denote by $S \cup S'$ their *union*, $S \cap S'$ their *intersection*, $S \setminus S'$ their *difference*, $S \times S'$ their *Cartesian product*. We denote by $|S|$ the *cardinality* of a set S. Given a set S, a set S' is a subset of S, or S' is included in S, denoted by $S' \subseteq S$, if all the elements in S' are also contained in S. When S contains some other element which is not in S', then S' is strictly included in S, written $S' \subset S$.

By $\emptyset$ we denote the empty set, i.e. the set without any element in it. A set with a single element in it is called a *singleton*. We say that two sets are *disjoint* when their intersection is $\emptyset$, otherwise they are *overlapping*. $\mathbb{N}$ denotes the set of the *natural numbers*. The *interval* $[m, n]$ is the subset of $\mathbb{N}$ containing exactly all natural numbers between m and n, including m and n. We denote *ordinals* by Greek letters $\alpha, \beta, \gamma, \ldots$. Given a set S and a natural number n, we denote by S^n the *n-th Cartesian power* of S, corresponding to the n-fold Cartesian product of S with itself.

Given a set S, the *powerset* of S, denoted by $\mathbf{2}^S$, is defined as the set of all the subsets of S, included the empty set $\emptyset$ and S itself. Given a set of sets T, the *big union* $\bigcup_{S \in T} S$

represents the union of all the sets in T, and similarly for the *big intersection* $\bigcap_{S \in T} S$.

Relations and functions

A *relation* $\mathcal{R}$ on sets S and T is a subset of the Cartesian product $S \times T$. We usually write $s \mathcal{R} t$ to express that $(s,t) \in \mathcal{R}$. The *domain* and *codomain* of a relation $\mathcal{R}$ are defined by $dom(\mathcal{R}) = \{s \mid \exists t.\, s \mathcal{R} t\}$ and $cod(\mathcal{R}) = \{t \mid \exists s.\, s \mathcal{R} t\}$, respectively. Given a relation $\mathcal{R}$, we denote by $\mathcal{R}^{-1}$ its *inverse relation*, such that $t \mathcal{R}^{-1} s$ if and only if $s \mathcal{R} t$. Given two relations $\mathcal{R}$ on sets S and T and $\mathcal{R}'$ on T and U, the *composition* of $\mathcal{R}$ and $\mathcal{R}'$ is the relation on sets S and U defined by $\mathcal{R} \circ \mathcal{R}' = \{(s,u) \mid \exists t.\, s \mathcal{R} t \wedge t \mathcal{R}' u\}$. Given a relation $\mathcal{R}$ on sets S and T, the *restriction* of $\mathcal{R}$ to a subset $S' \subseteq S$ is the relation on sets S' and T defined by $\mathcal{R}|_{S'} = \{(s,t) \mid s \in S' \wedge s \mathcal{R} t\}$.

A relation $\mathcal{R}$ which is a subset of S^2, or simply a relation over S, is *reflexive* if every element of S is related to itself, i.e., $\forall\, s \in S.\, s \mathcal{R} s$. $\mathcal{R}$ is *irreflexive* if there is no element of S related to itself. $\mathcal{R}$ is *symmetric* if $\forall\, s,t \in S.\, s \mathcal{R} t \iff t \mathcal{R} s$. $\mathcal{R}$ is *antisymmetric* if $\forall\, s,t \in S.\, (s \mathcal{R} t \wedge t \mathcal{R} s) \implies s = t$. $\mathcal{R}$ is *transitive* if $\forall\, s,t,u \in S.\, (s \mathcal{R} t \wedge t \mathcal{R} u) \implies s \mathcal{R} u$. The inverse of a symmetric relation is the same relation. The *reflexive closure* of a relation $\mathcal{R}$ is defined as the smallest relation that contains $\mathcal{R}$ and it is reflexive. Similarly we define also the *transitive closure*, denoted by $\mathcal{R}^*$. A relation $\mathcal{R}$ is *acyclic* if its transitive closure is irreflexive. Furthermore, an acyclic relation is always antisymmetric.

A relation which is reflexive and transitive is called a *preorder*. A preorder which is symmetric is called an *equivalence*. Given a set S and an equivalence $\equiv$ over S, we say that two elements $s, s' \in S$ are $\equiv$*-equivalent* when $s \equiv s'$. We call the $\equiv$*-equivalence class* of an element $s \in S$, denoted by $[s]_{\equiv}$, the subset of S of the elements $\equiv$-equivalent to s. We call an element $s \in S$ a *representative* of its equivalence class $[s]_{\equiv}$. The set of $\equiv$-equivalence classes of S, that is, the set of all the disjoint subsets of S which are equivalence classes of some element of S, is called the *quotient* of S by $\equiv$ and denoted by $S_{/\equiv}$. An equivalence is of *finite index* when the corresponding quotient is finite.

Functions are a special class of relations. A *function* $f : S \to T$ is a relation on S and T such that for every $s \in S$ there exists at most one $t \in T$ such that $(s,t) \in f$. If such a t exists we write $f(s) = t$ and call t the *image of* s *through* f, otherwise we say that f is undefined on s. We extend this notation also to subsets of S, for instance, given a subset $S' \subseteq S$ we write $f(S')$ for the set $\{t \mid \exists\, s \in S'.\, f(s) = t\}$ that is the image of S' through f. Given a function $f : S \to T$, we denote by $f[s \mapsto t]$ the *update* of f such that $f[s \mapsto t](s) = t$ and $f[s \mapsto t](s') = f(s')$ for all $s' \neq s$. Given a set S, we call *identity* the special function from S to S defined as the smallest reflexive relation over S.

A function $f : S \to T$ is *partial* if it is undefined on at least one element of S, otherwise $dom(f) = S$ and it is called *total*. A function is *injective* if every element of the codomain is the image of exactly one element of the domain. A function from S to T is *surjective* if every element of T is the image of at least one element of S. A function which is injective

and surjective is called a *bijection*. The inverse of a total bijection is also a total bijection. The inverse of a partial bijection is only injective. Otherwise specified, bijections are always intended to be total.

Sequences and tuples

A *sequence* is an ordered possibly transfinite collection of objects in which repetitions are allowed. A sequence is usually enumerated by ordinals, so that it can be seen as a function from the ordinal numbers to the set of elements it contains. Because of this, we represent a sequence as a list of indexed elements $(s_1, s_2, \ldots)$, specifying the last element and its index when the sequence is known to be finite $(s_1, s_2, \ldots, s_n)$ and its size is n. The empty sequence is denoted by ϵ. A *subsequence* of a sequence σ is a sequence obtained by deleting some of the elements in σ and preserving the relative order of the others. Given two sequences $\sigma = (s_1, s_2, \ldots, s_n)$ and $\sigma' = (s'_1, s'_2, \ldots)$, their *concatenation* is the sequence $\sigma\sigma' = (t_1, t_2, \ldots) = (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots)$. Given a set S we define the *Kleene star* of S, denoted by S^*, as the set of all possible finite sequences of elements of S, possibly with repetitions, including the empty sequence ϵ.

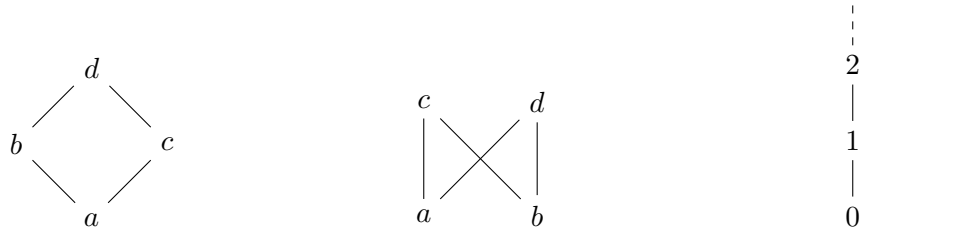
Tuples are a special class of sequences. A *tuple* is a finite sequence. More precisely, an *n-tuple* is a sequence of n elements, a *pair* is a sequence of 2 elements, a *triple* is a sequence of 3 elements, and so on. Given a natural number i and tuple σ of length at least i, we denote by $\pi_i(\sigma)$ the *i-th projection* of σ defined as the i-th element of σ. When clear from the context, given a tuple $\sigma = (s_1, s_2, \ldots, s_n)$, we allow to apply a function f defined on the elements of the tuple and we write $f(\sigma)$ for the tuple $(f(s_1), f(s_2), \ldots, f(s_n))$. We sometimes abuse the notation and use a tuple as a set meaning the set of all the elements appearing in the tuple. When a tuple contains only elements of the same kind, individually denoted by the same symbol with different indices, we represent the tuple by the boldface version of that symbol, for instance we write $\boldsymbol{x}$ for the tuple $(x_1, x_2, \ldots, x_n)$ where all x_i are variables.

Many mathematical objects appearing in the thesis are *structured objects*, defined by a tuple of components of various type. We often denote such objects by a slightly different notation using angle brackets instead of the normal parentheses and not assigning indices to the components, which may be of different nature, for instance, $A = \langle a, B, \gamma \rangle$ is a structured object made of three components. When the name of an object contains a subscript, its components usually carry the same subscript of the object. For example, to distinguish two objects A_1 and A_2 structured as above, we call their components $\langle a_1, B_1, \gamma_1 \rangle$ and $\langle a_2, B_2, \gamma_2 \rangle$, respectively.

2.2 Partial orders, lattices and domains

A number of paradigms and techniques in the theory of computation, in general, and in both software analysis and verification, in particular, have been built on the solid base of mathematical knowledge provided by the theory of orders and domains. In this section we review the core concepts of such theory on which the mathematical models, languages and methods discussed in the thesis are based. We first introduce the necessary notions of partial orders and then we go over the primary mathematical structures and properties built upon them. The definitions presented follow mainly those in [AJ94] which gives a very nice and detailed overview of the theory of domains started by Dana Scott (see [Sco70, Sco71, Sco72] just to mention a few).

A *partial order* $\sqsubseteq$ over a set P is a reflexive transitive antisymmetric relation on P. A set P with a partial order $\sqsubseteq$ is called a *partially ordered set* (poset) and it is denoted by $\langle P, \sqsubseteq \rangle$, or simply P omitting the order relation. Given two elements a and b in P, we say that a is *below* b when $a \sqsubseteq b$, dually we say that a is *above* b when $b \sqsubseteq a$, if none of the previous holds we say that a and b are *incomparable*. When the order relates every pair of elements in the set, that is, there are no incomparable elements, the order is called *total*. We graphically represent poset as Hasse diagrams, when it is possible given the size of the set or the structural organisation of the order relation. Some examples of poset are shown in Figure 2.1. Graphically, the partial order relation proceeds upwards along the lines. For instance, the element a in Figure 2.1a is below all the other elements in the set. Instead, elements b and c are incomparable since neither $b \sqsubseteq c$ nor $c \sqsubseteq b$.



(a) Four-element lattice (b) Non-lattice poset (c) Natural numbers

Figure 2.1: Examples of posets drawn as Hasse diagrams.

Let $\langle P, \sqsubseteq \rangle$ be a poset. A subset $S \subseteq P$ is called an *upper set* if for all $x \in S$ and $y \in P$, $x \sqsubseteq y$ implies $y \in S$. Dually, a subset S is called a *lower set* if for all $x \in S$ and $y \in P$, $y \sqsubseteq x$ implies $y \in S$. The *upper closure* of an element $x \in P$ is the set of all elements of P above x, or the smallest upper set containing x, and it is defined by $\uparrow x = \{y \in P \mid x \sqsubseteq y\}$. Dually, the *lower closure* of x is defined by $\downarrow x = \{y \in P \mid y \sqsubseteq x\}$. Upper and lower closures are extended to any subset $S \subseteq P$ by taking the union of the respective closures of every element in S. An element $x \in P$ is called an *upper bound* for a subset $S \subseteq P$ if $y \sqsubseteq x$ for all $y \in S$. Dually, x is a *lower bound* for S if $x \sqsubseteq y$ for all $y \in S$. Given a subset

$S \subseteq P$, an element $x \in S$ is called a *maximal* of S if for all $y \in S$, $x \sqsubseteq y$ implies $x = y$. Dually, $x \in S$ is called a *minimal* of S if for $y \in S$, $y \sqsubseteq x$ implies $x = y$. An element $x \in S$ is called the *maximum* or *greatest element* of S if for all $y \in S$, $y \sqsubseteq x$. Dually, $s \in S$ is called the *minimum* or *least element* of S, if for all $y \in S$, $x \sqsubseteq y$. The poset $\langle P, \sqsubseteq \rangle$ is called *well-ordered* when every non-empty subset $S \subseteq P$ has a least element. Given a subset $S \subseteq P$, if the set of the upper bounds of S has a least element, then such element is called the *least upper bound* or *supremum* of S, and it is denoted by $\bigsqcup S$. Dually, if the set of the lower bounds of S has a greatest element, then it is called the *greatest lower bound* or *infimum* of S, denoted by $\bigsqcap S$.

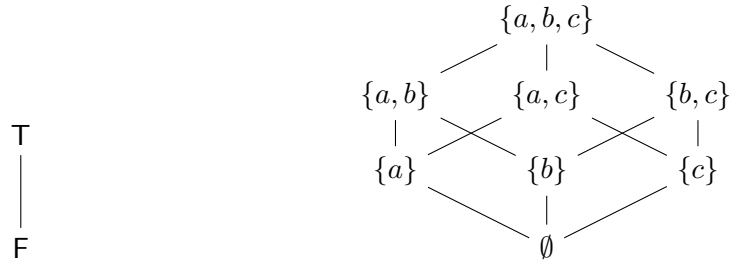
Definition 2.1 (complete lattice)**.** *A poset $\langle L, \sqsubseteq \rangle$ is a* lattice *if every non-empty finite subset of L has a supremum and an infimum. A lattice is* complete *if suprema and infima exist for all subsets, also empty and infinite ones.*

From the definition above it follows that a complete lattice L always has a greatest and least element, called *top* and *bottom* and usually denoted $\top = \bigsqcup L = \bigsqcap \emptyset$ and $\bot = \bigsqcap L = \bigsqcup \emptyset$, respectively. By looking at the poset in Figure 2.1a it can be seen that it is a complete lattice, in fact it has a top element d and a bottom element a. However the other two posets are not. The poset in Figure 2.1b is not even a lattice since the elements a and b (or c and d) have no least upper bound, rather they have no upper bound at all. Instead, the set of natural numbers ordered by the usual numerical order in Figure 2.1c is a lattice but it is not complete because it has no top element. Some classical examples of complete lattices are shown in Figure 2.2. The simple 2-elements lattice corresponding to the boolean values is clearly complete, its top element being T and bottom element F. A very common class of lattices are those obtained by ordering the powerset of a set by subset inclusion. Widely used in software verification, this kind of lattices will often recur in the thesis. Given any set S, its powerset $\mathbf{2}^S$ ordered by inclusion, or simply its powerset lattice $\langle \mathbf{2}^S, \subseteq \rangle$, is always a complete lattice, whatever S is. The empty set $\emptyset$ is the bottom, the whole set S is the top. Interestingly in a powerset lattice the least upper bounds and greatest lower bounds can be easily defined just using the two common operations on sets: the supremum of some sets is just their union and the infimum is their intersection. An example of powerset lattice is represented in Figure 2.2b.

The techniques discussed in the thesis, and more in general the analysis and the verification of software systems, often involve the computation of fixpoints of functions over this kind of mathematical structures. Given a poset P, a *fixpoint* of a function $f : P \to P$ is an element $x \in P$ such that $f(x) = x$. An element $x \in P$ such that $x \sqsubseteq f(x)$ is called a *post-fixpoint* of f. Dually, x is called a *pre-fixpoint* of f if $f(x) \sqsubseteq x$.

Almost invariably, the functions of interest comply with the following crucial property.

Definition 2.2 (monotone function)**.** *Let $\langle P, \sqsubseteq \rangle$ be a poset. A function $f : P \to P$ is* monotone *if for all $x, y \in P$, $x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$.*



(a) Booleans (b) Powerset lattice of $\{a, b, c\}$

Figure 2.2: Examples of complete lattices.

When the set of fixpoints of a function f has a minimum, such element is called the *least fixpoint* of f, denoted by μf. Dually, when there is a maximum, it is called the *greatest fixpoint* of f, denoted by νf. A well-known result, namely Knaster-Tarski's theorem [Tar55], establishes that every monotone function on a complete lattice has a least and a greatest fixpoint.

Theorem 2.1 (Knaster-Tarski)**.** *If $\langle L, \sqsubseteq \rangle$ is a complete lattice and $f : L \to L$ is a monotone function, then the set of fixpoints of f ordered by $\sqsubseteq$ is a non-empty complete lattice. Hence f has a least fixpoint being the infimum of the pre-fixpoints*

$$\mu f = \bigsqcap \{x \in L \mid f(x) \sqsubseteq x\}$$

and a greatest fixpoint being the supremum of the post-fixpoints

$$\nu f = \bigsqcup \{x \in L \mid x \sqsubseteq f(x)\}.$$

A more constructive way of defining the least and greatest fixpoints of a monotone function over a complete lattice is provided by another famous result, often referred to as Kleene's theorem. Actually the original theorem by Kleene [Kle52] applies to continuous functions on directed complete posets. It was later generalised in [CC79] for monotone functions on complete lattices. Given a complete lattice L and a monotone function f over L, the sequence $f^i(\bot)$ for i ranging over the natural numbers, where $f^0(\bot) = \bot$ and $f^{i+1}(\bot) = f(f^i(\bot))$, is an ascending chain $\bot \sqsubseteq f(\bot) \sqsubseteq f(f(\bot)) \sqsubseteq \ldots$. The chain can be further extended into a transfinite chain over the ordinals by letting also $f^\alpha(\bot) = \bigsqcup_{\beta<\alpha} f^\beta(\bot)$ for α limit ordinal. The so-defined transfinite chain is still an ascending chain and it is called the *ascending Kleene chain*. Then the theorem states that after a (ordinal) number of iterations the chain stabilizes reaching a fixpoint of f, which in fact is its least fixpoint. The greatest fixpoint can be dually characterised using the descending chain starting from $\top$.

Theorem 2.2 (Kleene's iteration)**.** *If $\langle L, \sqsubseteq \rangle$ is a complete lattice and $f : L \to L$ is a monotone function, then there exists an ordinal α such that the least fixpoint of f is $\mu f = f^\alpha(\bot)$. Dually, there exists α such that the greatest fixpoint is $\nu f = f^\alpha(\top)$.*

This result gives us a more practical way of computing the fixpoints, least or greatest, of monotone functions over complete lattices. To compute the least (greatest) fixpoint of a function we just need to iterate the function "enough" starting from $\bot$ ($\top$). We will often exploit this method in the techniques developed in the thesis.

Another important notion which can be formalised over partially ordered sets is that of approximation. Intuitively any element of a poset can be composed of (or approximated by) simpler ones. What these simpler elements are depends only on the poset itself. Here "approximation" does not necessarily mean imprecise. In fact, the next definition explains which elements are sufficient to correctly approximate every other element.

Definition 2.3 (basis)**.** *Let $\langle P, \subseteq \rangle$ be a poset. A subset $B \subseteq P$ is a* basis *for P if for every element $x \in P$, it holds $x = \bigsqcup(B \cap {\downarrow}x)$.*

In words this means that every element of the poset can be precisely approximated by the elements of the basis below it. Observe that every poset P has at least one basis since P itself is always a basis for it. However, more useful cases are when a "small" basis can be identified. A classical example is again the powerset lattice, where the basis can be the set of all singletons included in the whole set. So, a good basis for the powerset lattice in Figure 2.2b would be $B = \{\{a\}, \{b\}, \{c\}\}$. Notice that the bottom element $\emptyset$ is never needed in the basis since $\bigsqcup(B \cap {\downarrow}\emptyset) = \bigsqcup \emptyset = \emptyset$. Indeed, this is true for every poset with a bottom element. Because of this, when working on posets with a bottom element $\bot$, usually complete lattices, we will assume the associated basis not to contain $\bot$.

With the notions reviewed so far we can informally define *domains* as special posets admitting some useful basis. Different classes of domains can be identified depending on their features and those of their basis. For the purpose of this thesis we do not need more formal definitions. However, it should be noted that one of the main models involved in our work, that is, event structures (defined in the next chapter), can be seen as a special class of domains, namely finitary prime algebraic domains [Win87].

Chapter 3

Models for concurrent systems

This chapter reviews event structures, a classical semantic model for concurrency, and Petri nets, an operational model which allows for a natural representation of concurrent and distributed systems. In order to faithfully model the concurrent behaviour of systems we need formalisms that represent concurrency as a primitive notion. We conclude the chapter discussing notions of bisimulation and bisimilarity over these models, especially suited to deal with concurrent systems.

3.1 Event structures

Prime event structures [NPW81, Win87] are a widely known model to describe the true concurrent semantics of concurrent computational processes. They describe the behaviour of a system in terms of events and dependency relations between such events. Each event represents a unique occurrence of an action performed by the system and, as such, cannot appear more than once in a computation. The corresponding action is indicated by the label attached to the event. Two relations are used to express how events causally depend on, or are in conflict with, the others. An event causally depends on those which must take place before the event can occur, in this case we say that the event is caused by them. Two events are in conflict with each other when they can never appear both in the same computation, that is, the occurrence of one excludes the other for the rest of the execution. Since an event causing another must take place before the other can occur, the conflict relation is inherited by all the events caused by the ones in conflict. Throughout the thesis $\mathbb{E}$ is a fixed set of events from which all events are taken, Λ is a finite set of labels ranged over by $\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots$, and $\lambda : \mathbb{E} \to \Lambda$ is a labelling function.

Definition 3.1 (prime event structure)**.** *A (Λ-labelled)* prime event structure *(PES) is a tuple $\mathcal{E} = \langle E, \leq, \# \rangle$, where $E \subseteq \mathbb{E}$ is the set of* events *and $\leq$, $\#$ are binary relations on E, called* causality *and* conflict *respectively, such that:*

1. *$\leq$ is a partial order and $\downarrow e$ is finite for all $e \in E$;*

2. *$\#$ is irreflexive, symmetric and hereditary with respect to $\leq$, i.e., for all $e, e', e'' \in E$, if $e\#e' \leq e''$ then $e\#e''$.*

In the following, we will assume that the components of an event structure $\mathcal{E}$ are named as in the definition above, possibly with subscripts.

Two PESs which are the same up to renaming of their events are considered indistinguishable. This is formalised via the notion of isomorphism.

Definition 3.2 (isomorphism of PESs)**.** *The PESs $\mathcal{E}_1 = \langle E_1, \leq_1, \#_1 \rangle$ and $\mathcal{E}_2 = \langle E_2, \leq_2, \#_2 \rangle$ are* isomorphic*, written $\mathcal{E}_1 \sim \mathcal{E}_2$, when there is a bijection $\iota : E_1 \to E_2$ such that for all $e_1, e_1' \in E_1$ it holds $e_1 \leq_1 e_1'$ iff $\iota(e_1) \leq_2 \iota(e_1')$, $e_1 \#_1 e_1'$ iff $\iota(e_1) \#_2 \iota(e_1')$, and $\lambda(e_1) = \lambda(\iota(e_1))$.*

As mentioned, these models include concurrency as a primitive notion. Indeed, even though the concurrency relation does not explicitly appear among the components of a PES, it can be easily derived from them. The next definition explains how.

Definition 3.3 (consistency, concurrency)**.** *Let $\mathcal{E}$ be a PES. We say that events $e, e' \in E$ are* consistent*, written $e \frown e'$, if $\neg(e\#e')$. A subset $X \subseteq E$ is called* consistent *if $e \frown e'$ for all $e, e' \in X$. We say that e and e' are* concurrent*, written $e \,||\, e'$, if $e \frown e'$, $\neg(e \leq e')$ and $\neg(e' \leq e)$.*

Causality, concurrency and consistency will be sometimes used on set of events. Given $X \subseteq E$ and $e \in E$, by $X < e$ we mean that for all $e' \in X$, $e' < e$. Similarly $X \,||\, e$, resp. $X \frown e$, means that for all $e' \in X$, $e' \,||\, e$, resp. $e' \frown e$.

A state of the system is represented by the set of events which led to it, that is the past computation. These states or sets of events are captured by the notion of configuration.

Definition 3.4 (configuration)**.** *Let $\mathcal{E}$ be a PES. A* configuration *in $\mathcal{E}$ is a finite consistent subset of events $C \subseteq E$ closed w.r.t. causality (i.e., $\downarrow e \subseteq C$ for all $e \in C$). The set of finite configurations of $\mathcal{E}$ is denoted by $\mathcal{C}(\mathcal{E})$.*

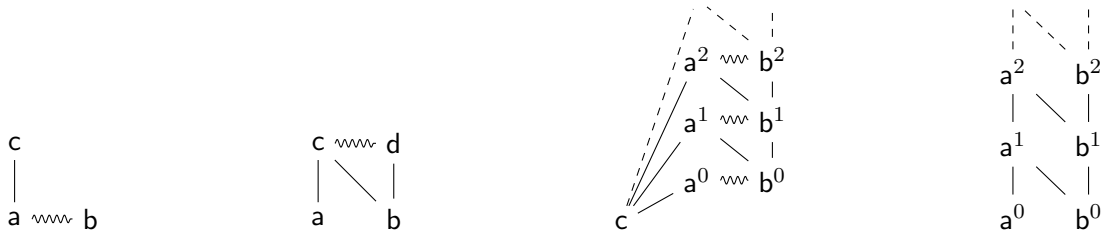
Since events in conflict are not admitted in the same computation, a configuration must be a consistent set of events. Furthermore, it must be a lower set with respect to causality because all the causes of an event must precede the event itself in the computation. The empty set of events $\emptyset$ is always a configuration, which can be interpreted as the initial state of the computation. Then, the evolution of a system can be represented by a transition system where configurations are states.

Definition 3.5 (transition system)**.** *Let $\mathcal{E}$ be a PES and let $C \in \mathcal{C}(\mathcal{E})$. Given $e \in E \setminus C$ such that $C \cup \{e\} \in \mathcal{C}(\mathcal{E})$, and $X, Y \subseteq C$ with $X < e$ and $Y \,||\, e$, we say that there is a transition $C \xrightarrow{X, Y < e}_{\lambda(e)} C \cup \{e\}$, possibly omitting the sets X, Y or the label $\lambda(e)$.*

Transitions are labelled by the executed event e. In addition, they can report its label $\lambda(e)$, a subset of causes X and a set of events $Y \subseteq C$ concurrent with e. Both sets X

and Y are not required to include all the events in the configuration C respectively causing or concurrent with e, in fact they can also just be empty sets. In such case they are normally omitted, e.g., we write $C \xrightarrow{X<e}_{\lambda(e)} C'$ for $C \xrightarrow{X,\bar{\emptyset}<e}_{\lambda(e)} C'$ and $C \xrightarrow{e}_{\lambda(e)} C'$ for $C \xrightarrow{\emptyset,\bar{\emptyset}<e}_{\lambda(e)} C'$.

Some simple examples of PESs are depicted in Figure 3.1. Since the set of events of a PES is partially ordered by the causality relation, it is graphically represented by the corresponding Hasse diagram. We add to this curly lines to represent immediate conflicts. Events are denoted by their labels, possibly with superscripts. For instance, in $\mathcal{E}_3$, the events a^0 and b^0, labelled by a and b, respectively, are in conflict. Event c causes the events a^i and it is concurrent with b^i for all $i \in \mathbb{N}$. The PESs $\mathcal{E}_1$ and $\mathcal{E}_2$ are finite, while $\mathcal{E}_3$ and $\mathcal{E}_4$ are infinite, as suggested by the dashed lines.



(a) $\mathcal{E}_1$ (b) $\mathcal{E}_2$ (c) $\mathcal{E}_3$ (d) $\mathcal{E}_4$

Figure 3.1: Examples of finite and infinite PESs.

Figure 3.2 shows the transition system associated with $\mathcal{E}_2$. It is the set of possible configurations $\mathcal{C}(\mathcal{E}_2)$ ordered by subset inclusion. Indeed, subset inclusion can be safely thought of as a computational ordering on configurations. A transition just adds the executed event to the configuration, so that the source is always strictly included in the target. Comparing Figures 3.1b and 3.2 should also give some intuition on how PESs mitigate the state space explosion problem.

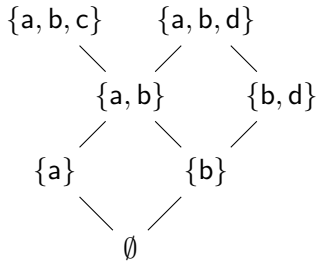


Figure 3.2: Transition system associated with $\mathcal{E}_2$.

We will often need to refer to the evolution of a system starting from some state, i.e., after some initial computation. For a PES and a configuration this is called the residual and it corresponds to the part of the PES that remains to be executed after the computation

expressed by the configuration. Since the events in the configuration have already taken place the events included in the residual must be consistent with them.

Definition 3.6 (residual)**.** *Let $\mathcal{E}$ be a PES. For a configuration $C \in \mathcal{C}(\mathcal{E})$, the* residual *of $\mathcal{E}$ after C is defined as $\mathcal{E}[C] = \{e \mid e \in E \setminus C \ \wedge \ C \frown e\}$.*

A residual $\mathcal{E}[C]$ endowed with the restrictions of the relations of causality and conflict of $\mathcal{E}$ is defined by the structure $\langle \mathcal{E}[C], \leq|_{\mathcal{E}[C]}, \#|_{\mathcal{E}[C]} \rangle$. Since $\mathcal{E}$ is a PES, it is easy to see that such a structure complies with all the requirements in Definition 3.1, and thus it can be seen as a PES (see [BGV17, Lemma 2]). For this reason a residual $\mathcal{E}[C]$ will be often confused with the corresponding PES $\langle \mathcal{E}[C], \leq|_{\mathcal{E}[C]}, \#|_{\mathcal{E}[C]} \rangle$.

Proposition 3.1 (residuals as PESs)**.** *Let $\mathcal{E}$ be a PES. Given a configuration $C \in \mathcal{C}(\mathcal{E})$, the residual $\mathcal{E}[C]$ endowed with the restrictions of $\leq$ and $\#$ is a PES.*

In general PESs can be infinite even when arising from finite state systems. Indeed, a PES associated with a non-trivial system exhibiting a cyclic behaviour will involve an infinite number of events, due to the fact that each occurrence of an action must be represented by a unique event. However, it is often the case that such PESs actually include infinitely many occurrences of finitely many sub-structures. More formally, this happens when the set of all the possible residuals of the PES is finite up to isomorphism of PESs. Intuitively, this is crucial for most verification tasks as it means that the possibly infinite semantic models to check can be effectively reduced to a finite size. Furthermore, to this aim PESs should also not have states from where infinitely many different transitions can be executed. Since each transition corresponds to the occurrence of an event, first we clarify which events can be executed at a given configuration.

Definition 3.7 (enabled event)**.** *Let $\mathcal{E}$ be a PES. Given a configuration $C \in \mathcal{C}(\mathcal{E})$, we say that an event $e \in E$ is* enabled *at C if $C \xrightarrow{e} C \cup \{e\}$. The set of enabled events at a configuration C is defined as $en(C) = \{e \in E \mid C \xrightarrow{e} C \cup \{e\}\}$.*

Then, the number of enabled events at a configuration determines the branching behaviour of the PES from that configuration. In particular, it is important to know if there is a bound on how much the PES can branch out from every configuration, in which case we say that it is boundedly branching.

Definition 3.8 (branching)**.** *Let $\mathcal{E}$ be a PES. We say that $\mathcal{E}$ is* k-boundedly branching *for some $k \in \mathbb{N}$ if $|en(C)| \leq k$ for all $C \in \mathcal{C}(\mathcal{E})$. We say that $\mathcal{E}$ is* boundedly branching *if it is k-boundedly branching for some $k \in \mathbb{N}$.*

We can now present the definition of regular event structures, introduced by [Thi02], capturing the class of PESs enjoying the finitary property roughly described above.

Definition 3.9 (regularity)**.** *A PES $\mathcal{E}$ is called* regular *when it is boundedly branching and the set of residuals $\{\mathcal{E}[C] \mid C \in \mathcal{C}(\mathcal{E})\}$ is finite up to isomorphism of PESs.*

Observe that the two requirements are independent of each other. To see this, let $\mathcal{E} = \langle \mathbb{N}^*, \leq, \# \rangle$ be the PES with events the set $\mathbb{N}^*$ of all finite sequences of natural numbers, all labelled by the same label, where $e \leq e'$ if and only if e is a prefix of e', and $e \# e'$ when neither e is a prefix of e' nor e' is a prefix of e. Then, clearly $\mathcal{E}$ is a PES and the set of its residuals $\{\mathcal{E}[C] \mid C \in \mathcal{C}(\mathcal{E})\}$ has a finite number of equivalence classes, just one in fact, since they are all isomorphic to $\mathcal{E}[\emptyset]$, that is, the whole PES $\mathcal{E}$ itself. However, at every configuration, an infinite number of events are enabled, one for each natural number, thus violating the boundedly branching condition. Conversely, the PES $\mathcal{E}_4$ in Figure 3.1d is boundedly branching, it has at most two enabled events at every state, but it can be seen to have infinitely many different residuals up to isomorphism, already considering just the residuals $\mathcal{E}_4[\{\mathsf{b}^0, \ldots, \mathsf{b}^i\}]$ for all $i \in \mathbb{N}$.

Obviously all finite PESs are regular, but often even infinite ones are. As further discussed in the next section, normally regular event structures arise as semantics of finite state systems. An example of infinite regular PES is $\mathcal{E}_3$ in Figure 3.1c. Indeed, $\mathcal{E}_3$ has just three (equivalence classes of) residuals: $\mathcal{E}_3[\emptyset]$ which is the whole PES $\mathcal{E}_3$ itself, $\mathcal{E}_3[\{\mathsf{c}\}] = \{\mathsf{a}^0, \mathsf{a}^1, \ldots, \mathsf{b}^0, \mathsf{b}^1, \ldots\}$, and $\mathcal{E}_3[\{\mathsf{c}, \mathsf{a}^0\}] = \emptyset$. Moreover, $\mathcal{E}_3$ is clearly (2-)boundedly branching.

3.2 Petri nets

In this section we introduce Petri nets [Pet62], a classical concrete formalism widely used to model concurrent and distributed systems, and their unfolding semantics. Then we review the close relationship they share with event structures.

A Petri net is a directed bipartite graph, where nodes are either places or transitions. We assume that the set of transitions is a subset of a fixed set $\mathbb{T}$ with a labelling $\lambda_N : \mathbb{T} \to \Lambda$.

Definition 3.10 (Petri net)**.** *A* Petri net *is a tuple $\mathcal{N} = \langle P, T, F, M_0 \rangle$, where P and T are disjoint sets of* places *and* transitions*, respectively, $F : (P \times T) \cup (T \times P) \to \{0, 1\}$ is the* flow function*, and $M_0 : P \to \mathbb{N}$ is the initial marking, i.e., the initial state of the net.*

In the following, we will assume that the components of a Petri net $\mathcal{N}$ are named as in the definition above.

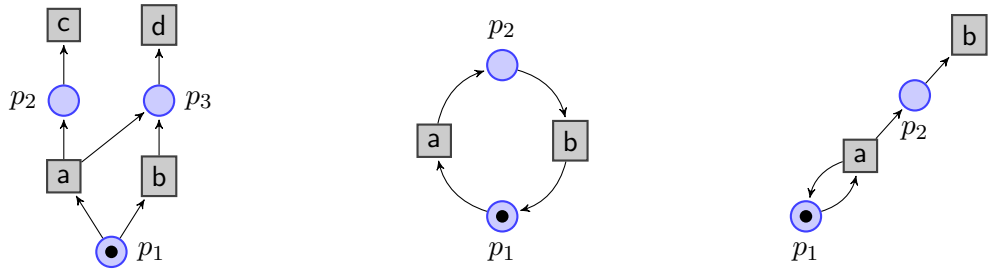
Places represent abstract resources, which can be available in a certain amount. A state of a net is called a *marking*, defined by a function $M : P \to \mathbb{N}$, indicating for each place how many instances, called *tokens*, of that resource are available in the place. Transitions are actions that the system can perform. In order to be executed, a transition t requires some instances of resources as specified by the flow function F (one token from each place $p \in P$ such that $F(p, t) = 1$). The execution of t consumes such resources and usually produces new ones (one new token in each place $p \in P$ such that $F(t, p) = 1$). In general, the flow function defines the pre and postconditions of every place and transition, namely the pre-set and post-set.

Definition 3.11 (pre-set, post-set)**.** *Let $\mathcal{N}$ be a Petri net. Given $x \in P \cup T$, the* pre-set *and* post-set *of x are defined by ${}^\bullet x = \{y \in P \cup T \mid F(y,x) = 1\}$ and $x^\bullet = \{y \in P \cup T \mid F(x,y) = 1\}$, respectively.*

Two Petri nets are isomorphic when their places and transitions coincide, respectively, in compliance with their flow functions, initial markings, and labels.

Definition 3.12 (isomorphism of Petri nets)**.** *The Petri nets $\mathcal{N} = \langle P, T, F, M_0 \rangle$ and $\mathcal{N}' = \langle P', T', F', M_0' \rangle$ are* isomorphic *when there is a bijection $\iota : P \cup T \to P' \cup T'$ such that $P' = \iota(P)$, $T' = \iota(T)$, $F = F' \circ \iota$, $M_0 = M_0' \circ \iota$, and for all $t \in T$ it holds $\lambda_N(t) = \lambda_N(\iota(t))$.*

Examples of Petri nets are shown in Figure 3.3. We adopt the standard graphical representation, where places are drawn as circles and transitions as boxes, denoted by their labels on the inside, possibly with superscripts. The flow function is rendered by means of directed arcs connecting places and transitions. The initial marking is represented by inserting tokens (black dots) in the corresponding places. For instance, in the leftmost net $\mathcal{N}_1$ from the initial marking either the transition labelled a or the one labelled b can be executed but not both, as they both require to use the only token placed in p_1. If a is executed, it consumes the token in p_1 and produces one token in both p_2 and p_3. From this new state both transitions c and d can be executed concurrently. The net $\mathcal{N}_1$ is acyclic and does not allow infinite computations. Instead, the other nets $\mathcal{N}_2$ and $\mathcal{N}_3$ are cyclic and infinite computations are made possible by looping infinitely many times through some of their transitions. For example, the only infinite computation that $\mathcal{N}_2$ can perform corresponds to the infinite sequence alternating the two transitions a and b.



(a) $\mathcal{N}_1$ (b) $\mathcal{N}_2$ (c) $\mathcal{N}_3$

Figure 3.3: Examples of acyclic and cyclic Petri nets.

Formally, a transition $t \in T$ is *enabled* at a marking M if $M(p) \geq F(p,t)$ for all $p \in P$. In this case it can be *fired* leading to a new marking M' defined by $M'(p) = M(p) + F(t,p) - F(p,t)$ for all places $p \in P$. This is written $M[t\rangle M'$. Since not all the combinations of places and tokens can be reached through a sequence of transitions, that is, a computation, only the so-called reachable markings are usually considered.

Definition 3.13 (reachable marking)**.** *Let $\mathcal{N}$ be a Petri net. A marking M is called* reachable *if there exists a sequence of transitions $(t_0, \ldots, t_n)$ such that $M_0[t_0\rangle \ldots [t_n\rangle M$. We denote by $\mathcal{R}(\mathcal{N})$ the set of all reachable markings in $\mathcal{N}$.*

An interesting class of Petri nets are those whose reachable markings contain at most one token in every place. These nets are called 1-safe nets, or simply safe nets.

Definition 3.14 (safe Petri net)**.** *A Petri net $\mathcal{N}$ is called* safe *if for every reachable marking $M \in \mathcal{R}(\mathcal{N})$ and all $p \in P$ it holds $M(p) \leq 1$. A safe net is called* finite *if both sets P and T are finite.*

For example the nets $\mathcal{N}_1$ and $\mathcal{N}_2$ in Figure 3.3 are safe and finite. Indeed, it can be seen that along every computation starting from their initial markings it never happens that a place contains more than one token. The net $\mathcal{N}_3$ instead is not safe. From the initial marking M_0 the sequence of firing of transitions $M_0[\mathsf{a}\rangle M_1[\mathsf{a}\rangle M_2$ produces a marking M_2 where the place p_2 contains two tokens, thus violating the requirement in the definition.

Since in safe nets places can contain at most one token, a marking M will be often confused with the corresponding subset of places $\{p \mid M(p) = 1\} \subseteq P$, and used as a set instead of a function. For instance, we might say that a transition t is enabled at a marking M if ${}^\bullet t \subseteq M$ and, after firing, it would produce the marking $M' = (M \smallsetminus {}^\bullet t) \cup t^\bullet$. Then, we say that a set of places X is *coverable* if there exists $M \in \mathcal{R}(\mathcal{N})$ such that $X \subseteq M$.

Safe nets are especially interesting because they are strongly related to event structures. For this reason, hereafter we will only consider nets in this class.

3.2.1 Unfolding

The concurrent behaviour of a Petri net $\mathcal{N}$ can be represented by its unfolding $\mathcal{U}(\mathcal{N})$, defined below as an acyclic net constructed inductively starting from the initial marking of $\mathcal{N}$ and then adding, at each step, an occurrence of each enabled transition of $\mathcal{N}$.

Definition 3.15 (net unfolding)**.** *Let $\mathcal{N}$ be a safe net. The* unfolding *of $\mathcal{N}$ is the least net $\mathcal{U}(\mathcal{N}) = \langle P^U, T^U, F^U, M_0^U \rangle$ such that:*

- *$M_0^U = \{(p, \bot) \mid p \in M_0\} \subseteq P^U$, where $\bot$ is a new element, not in P, T or F;*
- *if $t \in T$ and $X \subseteq P^U$ is coverable with $\pi_1(X) = {}^\bullet t$, then $(t, X) \in T^U$;*
- *for every $e = (t, X) \in T^U$, the set $Z = \{(p, e) \mid p \in t^\bullet\} \subseteq P^U$, moreover ${}^\bullet e = X$ and $e^\bullet = Z$;*
- *for all $x, y \in P^U \cup T^U$, $F^U(x, y) = 1 \iff x \in \pi_2(y)$.*

Places and transitions in the unfolding represent instances of tokens and firing of transitions, respectively, of the original net. The projection π_1 over the first component maps places and transitions of the unfolding to the corresponding items of the original net $\mathcal{N}$. The second component records the "causal history", i.e., for places the transition that generated the token (or $\bot$ for tokens in the initial marking that have not been generated by any transition) and for transitions the set of tokens required for the firing. The initial marking

M_0^U consists of the set of minimal places. The labels of transitions in the unfolding are the same of the labels of the corresponding transitions of the original net, i.e., a transition $e \in T^U$ is labelled by $\lambda_N(\pi_1(e))$. The unfolding of a non-trivial net is usually infinite even though the net is finite. For instance, a cyclic net allowing for infinite computations will generate an infinite unfolding. For historical reasons transitions and places in the unfolding are also called *events* and *conditions*, respectively.

As an example, consider the safe net $\mathcal{N}$ on the left of Figure 3.4, its unfolding is drawn on the right. The transition b in $\mathcal{N}$ can fire multiple times in a computation, therefore in the unfolding $\mathcal{U}(\mathcal{N})$ there are multiple instances (infinitely many) of its firing named b^i for $i \in \mathbb{N}$. Indeed, since the net is cyclic and allows for infinite computations, its unfolding is infinite. Interestingly in $\mathcal{U}(\mathcal{N})$ there are also multiple occurrences a^i of the original transition a, although in $\mathcal{N}$ such transition can fire at most one time in every computation. This is due to the fact that different occurrences of a in the unfolding corresponds to different conflictual computations, that is, at most one of them can actually fire, excluding the others. The reason why they appear as different occurrences is that each one has a different computational (causal) history.

(a) $\mathcal{N}$ (b) $\mathcal{U}(\mathcal{N})$

Figure 3.4: A finite safe Petri net and its unfolding.

One can define *causality* $\leq_N$ over the unfolding as the reflexive and transitive closure of the flow relation. Explicitly, $\leq_N$ is the smallest reflexive and transitive relation such that $x \leq_N y$ whenever $x \in {}^\bullet y$. Then, *conflict* $\#_N$ is the smallest relation such that $e \#_N e'$ if ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$ (i.e., when e and e' compete for a common resource) and inherited along causality, i.e., if $x \#_N y$ and $y \leq_N z$ then $x \#_N z$.

Once endowed with causality and conflict, the events T^U of the unfolding of a safe net can be interpreted as a PES. This result originates from [NPW81] where unfoldings are characterised as nets of a special subclass called occurrence nets.

Theorem 3.1 (PES for a net)**.** *Let $\mathcal{N}$ be a safe net and let $\mathcal{U}(\mathcal{N}) = \langle P^U, T^U, F^U, M_0^U \rangle$ be its unfolding. The triple defined by $\mathcal{E}(\mathcal{N}) = \langle T^U, \leq_N, \#_N \rangle$ is a PES.*

For instance, using this construction we can see that the net $\mathcal{N}$ in Figure 3.4 actually corresponds via its unfolding $\mathcal{U}(\mathcal{N})$ to the PES $\mathcal{E}_3$ in Figure 3.1c.

A configuration $C \in \mathcal{C}(\mathcal{E}(\mathcal{N}))$ is a subset of transitions of the unfolding $C \subseteq T^U$. It is not really a sequence of firing of transitions, as we would expect from a computation in the unfolding. However the properties of configurations of PESs ensure that there is at least one valid firing sequence, that is compatible with the flow/causality relation. Luckily we do not need to know the exact firing sequence of the transitions in C. The marking of the unfolding associated with C can be simply obtained by adding to the initial marking M_0^U all the tokens produced by transitions in C and then removing the ones consumed by transitions in C. We denote this by $C^\circ = (M_0^U \cup \bigcup_{e \in C} e^\bullet) \smallsetminus (\bigcup_{e \in C} {}^\bullet e)$. In turn, this corresponds to a marking of the original net denoted by $\mathsf{M}(C) = \pi_1(C^\circ)$. For the reason explained at the beginning of the paragraph, the markings $\mathsf{M}(C)$ and C° associated with a configuration $C \in \mathcal{C}(\mathcal{E}(\mathcal{N}))$ are always reachable in $\mathcal{N}$ and $\mathcal{U}(\mathcal{N})$, respectively.

Similarly to event structures, we can consider the substructure of an unfolding starting after some reachable marking, defined as the residual net including only the places and transitions that can still be involved in the remainder of the computation.

Definition 3.16 (residual unfolding)**.** *Let $\mathcal{N}$ be a safe net and let $\mathcal{U}(\mathcal{N}) = \langle P^U, T^U, F^U, M_0^U \rangle$ be its unfolding. For a reachable marking $M \in \mathcal{R}(\mathcal{U}(\mathcal{N}))$, the* residual *of $\mathcal{U}(\mathcal{N})$ after M is defined as $\mathcal{U}(\mathcal{N})[M] = \langle P_M^U, T_M^U, F_M^U, M \rangle$, where $P_M^U = \{b' \in P^U \mid \exists\, b \in M.\ b \leq b'\}$, $T_M^U = \{e \in T^U \mid \exists\, b \in M.\ b \leq e\}$, and F_M^U is the restriction of F^U to $(P_M^U \times T_M^U) \cup (T_M^U \times P_M^U)$.*

It is straightforward to see that the residual $\mathcal{U}(\mathcal{N})[M]$ of an unfolding $\mathcal{U}(\mathcal{N})$ after a reachable marking M is isomorphic to the unfolding of the net $\mathcal{N}$ with $\pi_1(M)$ as initial marking. Moreover, since the events of the PES arising from a net are just the transitions of the unfolding, this means that also the residual $\mathcal{E}(\mathcal{N})[C]$ after a configuration $C \in \mathcal{C}(\mathcal{E}(\mathcal{N}))$ is isomorphic to the PES obtained from the unfolding of the net $\mathcal{N}$ with $\mathsf{M}(C)$ as initial marking.

The previous facts lead to the observation that the residual of the PES arising from a safe net after a configuration C is uniquely determined by the marking produced by C and the net itself. Therefore, we obtain a simple but crucial result.

Proposition 3.2 (markings and residuals)**.** *Let $\mathcal{N}$ be a safe net. Given two configurations $C_1, C_2 \in \mathcal{E}(\mathcal{N})$, if $\mathsf{M}(C_1) = \mathsf{M}(C_2)$ then the residuals $\mathcal{U}(\mathcal{N})[C_1^\circ]$ and $\mathcal{U}(\mathcal{N})[C_2^\circ]$ are isomorphic and $\mathcal{E}(\mathcal{N})[C_1] \sim \mathcal{E}(\mathcal{N})[C_2]$.*

When the safe net $\mathcal{N}$ is finite, the set of reachable markings $\mathcal{R}(\mathcal{N})$ is necessarily also finite, bounded at most by the size of the powerset of the places of the net. Then, Proposition 3.2 immediately implies that the set of all residuals of the PES $\mathcal{E}(\mathcal{N})$ is finite up to isomorphism. Further, observe that, since $\mathcal{N}$ is finite, the set of transitions T is finite, and so at most $|T|$ transitions can be enabled at the same time. Which means that $\mathcal{E}(\mathcal{N})$ is also boundedly branching. As already mentioned, a PES with these properties is called regular. This is rounded up by the famous result referred to as the first half of Thiagarajan's conjecture [Thi02].

Theorem 3.2 (regularity)**.** *Let $\mathcal{N}$ be a finite safe Petri net. The corresponding PES $\mathcal{E}(\mathcal{N})$ is regular.*

The second half of Thiagarajan's conjecture states conversely that if a PES $\mathcal{E}$ is regular, then there exists a finite safe Petri net $\mathcal{N}$ such that $\mathcal{E}(\mathcal{N}) \sim \mathcal{E}$. While the first half can be easily proved, as reasoned above, the second half of the conjecture has actually been recently disproved [CC17]. This sheds new light on the relationship between finite safe Petri nets and regular event structures, before thought to be the same, whereas the former turned out to be a strict subclass of the latter.

3.3 Behavioural equivalences

An important task in software verification is that of checking if two systems or processes are equivalent. This is usually done to formally test a system against its specification or just to check if a system enjoys a desired property. The behaviour of the system can be approximated by abstracting the operational model taking into account specific features of the computation. To properly do this, a number of behavioural equivalences have been defined over the years. One of the most popular equivalences, especially in concurrency theory, is bisimilarity. It was introduced in [Mil80] as a mean to compare the behaviour of CCS processes, and, during the same years, in [Par81] for proving the equivalence between languages of automata on infinite sequences. Intuitively, two processes are bisimilar when they can match each other's moves, that is, if one can perform an action, the other can simulate it by doing the same action. Formally, this concept is captured by the notion of bisimulation.

Definition 3.17 (bisimulation)**.** *Let $\mathcal{E}_1, \mathcal{E}_2$ be PESs. A* bisimulation *is a relation $R \subseteq \mathcal{C}(\mathcal{E}_1) \times \mathcal{C}(\mathcal{E}_2)$ such that for every $(C_1, C_2) \in R$ it holds*

- *if $C_1 \xrightarrow{e_1}_{\mathsf{a}} C_1'$, then $C_2 \xrightarrow{e_2}_{\mathsf{a}} C_2'$ and $(C_1', C_2') \in R$;*
- *if $C_2 \xrightarrow{e_2}_{\mathsf{a}} C_2'$, then $C_1 \xrightarrow{e_1}_{\mathsf{a}} C_1'$ and $(C_1', C_2') \in R$.*

We say that $\mathcal{E}_1$ are $\mathcal{E}_2$ are bisimilar *if there exists a bisimulation R such that $(\emptyset, \emptyset) \in R$.*

For our purposes, all the definitions in this section are given on event structures, despite the fact that they could be defined for general transition systems.

Bisimilarity is the largest bisimulation relation between the states of the two systems, that is, the one relating exactly every pair of bisimilar states. For instance, consider the processes modelled by the PESs $\mathcal{E}_5$ and $\mathcal{E}_6$ in Figure 3.5. Both processes can only execute a and b actions. $\mathcal{E}_5$ can either perform a followed by b or the opposite, b followed by a. $\mathcal{E}_6$ can perform a and b in parallel, which, once linearised, means in any order. So, both processes can only do a and b a single time in some order. In fact, despite having different levels of parallelism, they are bisimilar. This happens because bisimilarity abstracts

concurrency of actions by approximating it with the nondeterministic choice between the possible sequentializations. In this case, the bisimilarity relation between $\mathcal{E}_5$ and $\mathcal{E}_6$ would contain the pairs $(\emptyset,\emptyset)$, $(\{\mathsf{a}^0\},\{\mathsf{a}\})$, $(\{\mathsf{b}^0\},\{\mathsf{b}\})$, $(\{\mathsf{a}^0,\mathsf{b}^1\},\{\mathsf{a},\mathsf{b}\})$, $(\{\mathsf{b}^0,\mathsf{a}^1\},\{\mathsf{a},\mathsf{b}\})$. Notably, the presence of the pair of initial states $(\emptyset,\emptyset)$ is the proof that the two processes are indeed bisimilar.

(a) $\mathcal{E}_5$ (b) $\mathcal{E}_6$

Figure 3.5: Example of bisimilar PESs.

Since our aim is to faithfully capture the true concurrent behaviour of systems, clearly bisimilarity is not satisfactory. In order to exploit the full expressiveness of partial order models, like event structures, and be able to properly verify their behaviour, an equivalence has to recognise at the very least when two transitions of a system are independent and can be executed in parallel. Thus we need finer bisimulations. Several true concurrent equivalences have been defined in order to capture the concurrency features of computations to different extents (see [vGG01] for an exhaustive collection and detailed descriptions). Here we present just two of them, those of interest in this thesis, which are among the most important and studied.

History-preserving (hp-)bisimilarity [DDNM88, RT88, BDKP91] naturally exploits the causal partial order over the actions of the systems. It requires that an action of one system is simulated by an action of the other with the same causal history. In order to characterise such behaviour we define a suitable bisimulation, called history-preserving (hp-)bisimulation, requiring that at each step of the bisimulation there is an isomorphism between the causal histories of the current states.

Definition 3.18 (history-preserving bisimulation)**.** *Let $\mathcal{E}_1,\mathcal{E}_2$ be PESs. A* history-preserving (hp-)bisimulation *is a set of triples $R \subseteq \mathcal{C}(\mathcal{E}_1)\times\mathcal{C}(\mathcal{E}_2)\times\mathbf{2}^{E_1\times E_2}$ such that for every $(C_1,C_2,\iota)\in R$ it holds*

- *if $C_1 \xrightarrow{e_1}_{\mathsf{a}} C_1'$, then $C_2 \xrightarrow{e_2}_{\mathsf{a}} C_2'$ and $(C_1',C_2',\iota[e_1\mapsto e_2])\in R$;*
- *if $C_2 \xrightarrow{e_2}_{\mathsf{a}} C_2'$, then $C_1 \xrightarrow{e_1}_{\mathsf{a}} C_1'$ and $(C_1',C_2',\iota[e_1\mapsto e_2])\in R$;*
- *ι is an isomorphism between the PESs $\langle C_1,\leq_1|_{C_1},\emptyset\rangle$ and $\langle C_2,\leq_2|_{C_2},\emptyset\rangle$.*

We say that $\mathcal{E}_1$ are $\mathcal{E}_2$ are history-preserving (hp-)bisimilar *if there exists a hp-bisimulation R such that $(\emptyset,\emptyset,\emptyset)\in R$.*

The fact that the causal histories are isomorphic at each step guarantees that independent actions cannot be confused with dependent ones, even if they represent the same

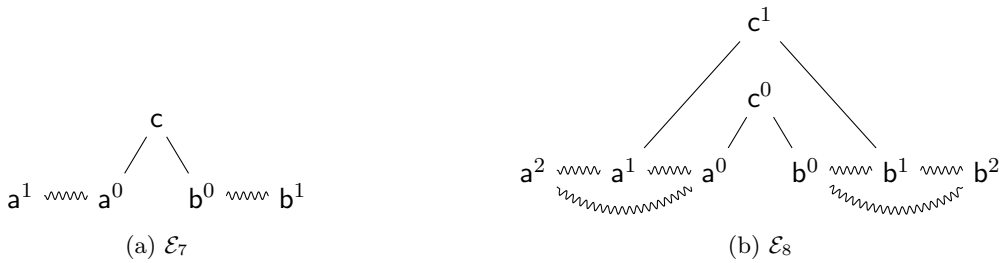
actions and are performed in the same order. So, looking back at the processes $\mathcal{E}_5$ and $\mathcal{E}_6$ in Figure 3.5, it can be easily seen that there is no hp-bisimulation between them including the initial states $(\emptyset, \emptyset)$, hence they are not hp-bisimilar. Indeed, those processes are a simple paradigmatic example of the difference between the interleaving and true concurrent approaches: the two PESs produce the same transition system (once causal dependencies are forgotten) in which a and b can be executed in any order, but they clearly differ from a true concurrent point of view since in $\mathcal{E}_6$, can execute a and b in parallel, while $\mathcal{E}_5$ cannot.

While this work is mainly focused on the verification of history-preserving properties, below we present another behavioural equivalence which is even finer than hp-bisimilarity, the finest in the true concurrent spectrum in fact. Hereditary history-preserving (hhp-)bisimilarity is a strengthening of hp-bisimilarity proposed by [Bed91] and shown to arise as a canonical equivalence across a range of different models for parallel computation [JNW96]. Hhp-bisimilarity precisely captures the interplay between branching, causality and concurrency by ensuring that the matching of states and transitions is not dependent on the order in which parallel actions are linearised in the bisimulation. This further constraint is captured by the addition of a backtracking requirement. First note that an action can be reversed only if it is maximal w.r.t. the causal partial order. Then, hhp-bisimilarity requires that for any two related computations, the states obtained by reversing a pair of related actions must be related too. Because of this hhp-bisimilarity turns out to be especially appropriate for systems with reversible computations [PU07, PU12, CKV13]. So, hhp-bisimilarity is obtained by means of a hp-bisimulation with backwards steps.

Definition 3.19 (hereditary history-preserving bisimulation)**.** *Let $\mathcal{E}_1, \mathcal{E}_2$ be PESs. A* hereditary history-preserving (hhp-)bisimulation *is a hp-bisimulation R such that if $(C_1, C_2, \iota) \in R$ and $C_1' \xrightarrow{e} C_1$ then $(C_1', \iota(C_1'), \iota|_{C_1'}) \in R$. We say that $\mathcal{E}_1$ are $\mathcal{E}_2$ are* hereditary history-preserving (hhp-)bisimilar *iff there exists a hhp-bisimulation R such that $(\emptyset, \emptyset, \emptyset) \in R$.*

To make clear the difference between hp-bisimilarity and hhp-bisimilarity examine the processes $\mathcal{E}_7$ and $\mathcal{E}_8$ in Figure 3.6. They are a classic counterexample given by [Frö04] to prove that hp-bisimilarity is in general strictly coarser than hhp-bisimilarity. Indeed, $\mathcal{E}_7$ and $\mathcal{E}_8$ are hp-bisimilar but not hhp-bisimilar. The first statement is based on the fact that even though $\mathcal{E}_8$ has more actions and branching paths to choose from, they can all be simulated by $\mathcal{E}_7$ with isomorphic causal histories. In fact, $\mathcal{E}_7$ can simulate every action of $\mathcal{E}_8$ with an isomorphic one depending on the specific event performed by $\mathcal{E}_8$ and those in the past. For example, the computation of $\mathcal{E}_8$ corresponding to the configuration $\{\mathsf{a}^0, \mathsf{b}^1\}$ is simulated by $\mathcal{E}_7$ with either the configuration $\{\mathsf{a}^0, \mathsf{b}^1\}$ or $\{\mathsf{a}^1, \mathsf{b}^0\}$ depending on the order in which the two actions performed by $\mathcal{E}_8$ have been linearised, that is, the order in which they happened in the hp-bisimulation. From what mentioned above, it should be clear that the same is not possible in a hhp-bisimulation, because the events a^0 and b^1 of $\mathcal{E}_8$ are independent but the choice of matching by $\mathcal{E}_7$ is dependent on their order. Indeed, allowing to backtrack, $\mathcal{E}_8$ could reverse one of the event and perform another one instead (with the same label) in

which case $\mathcal{E}_7$ may not be able to simulate it any more.



(a) $\mathcal{E}_7$ (b) $\mathcal{E}_8$

Figure 3.6: Hp-bisimilar PESs which are not hhp-bisimilar.

It is worth mentioning that hhp-bisimilarity, while being more precise, comes with a much greater cost. In fact, as proved in [JN00], the problem of checking whether two systems are hhp-bisimilar is undecidable for many basic models of concurrency, even for finite state systems (e.g., finite safe Petri nets [JNS03]). On the other hand, the decidability of hp-bisimilarity is well-established [Vog91, JM96, MP97] in particular for finite state systems (e.g., again, finite safe Petri nets). This contrasting difference should give some intuition on why working with history-preserving properties is more feasible than working with the more expressive hereditary history-preserving ones.

Chapter 4

Logics for concurrency

Modal and temporal logics are mathematical languages used to specify properties and specifications of systems. They have a formal syntax and semantics so that one can hope to have them automatically verified by a computer. Whereas techniques involving behavioural equivalences, as those mentioned in the previous chapter, to check a system against its specification require to model both entities with the same language, the use of logics allows to verify whether a system model, like, e.g., an event structure, complies with its specification expressed as a logical property. This kind of procedure is called model-checking, and it is the main subject of the thesis. In this chapter we review a number of modal logics especially relevant in the specification of properties of concurrent systems.

4.1 Hennessy–Milner logic

Hennessy–Milner logic (HML) is a modal logic that was introduced by [HM85] in the study of process calculi for concurrent systems. Differently from the traditional first and higher-order logics, which use quantifiers and other relations, modal logics include so-called *modalities*, or modal operators, in addition to the boolean operators of propositional logic. HML offers two modalities, namely the diamond and the box modality, to reason about the capability of performing some action. These operators are parametrised by the action to be performed.

Definition 4.1 (syntax of HML)**.** *The syntax of HML over the set of labels* Λ *is defined as follows:*

$$\varphi \ ::= \ \mathsf{T} \ \mid \ \mathsf{F} \ \mid \ \varphi \wedge \varphi \ \mid \ \varphi \vee \varphi \ \mid \ \langle \mathsf{a} \rangle \varphi \ \mid \ [\mathsf{a}] \varphi$$

T and F are the always true and always false constant formulae, respectively. $\wedge$ and $\vee$, called *conjunction* and *disjunction*, respectively, are the usual propositional connectives. $\langle \mathsf{a} \rangle$ is the *diamond modality* and $[\mathsf{a}]$ is the *box modality*, where a is the action to perform, represented by the label $\mathsf{a} \in \Lambda$. Notice the absence of *negation* $\neg$, which is unnecessary

since the logic is given in positive form. Indeed, for every operator the logic includes its dual.

The meanings of formulae, or more formally their semantics, is given by the set of states satisfying them, which in the case of event structures is a subset of their possible configurations. So, the constant T holds on every state, while F does not hold on any state. The conjunction of two formulae holds only on states satisfying both formulae, their disjunction holds on states satisfying at least one of them. The diamond modality $\langle \mathsf{a} \rangle \varphi$ holds on states which affords an a-labelled transition leading to a state satisfying φ. Dually, the box modality $[\mathsf{a}]\varphi$ holds on states from where all a-labelled transitions lead to states satisfying φ.

Definition 4.2 (semantics of HML)**.** *Let $\mathcal{E}$ be a PES. The denotation of a formula in HML is given by the function $\{\!|\cdot|\!\}^{\mathcal{E}} : HML \to \mathbf{2}^{\mathcal{C}(\mathcal{E})}$ defined inductively as follows:*

$$
\begin{aligned}
\{\!|\mathsf{T}|\!\}^{\mathcal{E}} &= \mathcal{C}(\mathcal{E}) \\
\{\!|\mathsf{F}|\!\}^{\mathcal{E}} &= \emptyset \\
\{\!|\varphi_1 \wedge \varphi_2|\!\}^{\mathcal{E}} &= \{\!|\varphi_1|\!\}^{\mathcal{E}} \cap \{\!|\varphi_2|\!\}^{\mathcal{E}} \\
\{\!|\varphi_1 \vee \varphi_2|\!\}^{\mathcal{E}} &= \{\!|\varphi_1|\!\}^{\mathcal{E}} \cup \{\!|\varphi_2|\!\}^{\mathcal{E}} \\
\{\!|\langle \mathsf{a} \rangle \varphi|\!\}^{\mathcal{E}} &= \{C \in \mathcal{C}(\mathcal{E}) \mid \exists\, C'.\ C \xrightarrow{e}_{\mathsf{a}} C' \wedge C' \in \{\!|\varphi|\!\}^{\mathcal{E}}\} \\
\{\!|[\mathsf{a}] \varphi|\!\}^{\mathcal{E}} &= \{C \in \mathcal{C}(\mathcal{E}) \mid \forall\, C'.\ C \xrightarrow{e}_{\mathsf{a}} C' \Rightarrow C' \in \{\!|\varphi|\!\}^{\mathcal{E}}\}
\end{aligned}
$$

When $C \in \{\!|\varphi|\!\}^{\mathcal{E}}$ we say that the PES $\mathcal{E}$ satisfies the formula φ in the configuration C. When $\mathcal{E}$ satisfies the formula φ in the empty configuration we simply say that $\mathcal{E}$ satisfies φ.

From a program logic such as HML it is possible to derive an equivalence over the systems or models involved. The idea is that different systems should be told apart by some formula of the logic holding only on one of them. This is formalised through the notion of logical equivalence.

Definition 4.3 (logical equivalence)**.** *Let $\mathcal{L}$ be a logic. We say that two systems P and Q are* logically equivalent in $\mathcal{L}$ *if and only if they satisfy the same (closed) formulae of $\mathcal{L}$.*

Finer logical equivalences will arise from more expressive logics, or, in other words, the expressive power of the logic will induce the level of abstraction of the equivalence.

One of the most interesting aspects of HML, and the root of its introduction, is that under mild hypotheses it fully characterises bisimilarity, that is, the logical equivalence induced by HML is bisimilarity [HM85] (for image-finite systems, i.e., systems where for each state s and label a the set $\{s' \mid s \xrightarrow{\mathsf{a}} s'\}$ is finite). In this way we obtain an alternative definition of bisimilarity through the logic. It follows that one can prove that two processes are not bisimilar by simply exhibiting a HML formula satisfied by only one of them. This methodology turns out to be very natural and effective to establish when two processes are not equivalent, and at the same time to reason on the causes of such difference. For

instance, a formula satisfied by the specification of a system but not by the system itself easily leads to understand why and where the system would fail. We will see that also the other more expressive logics, which will be presented later, share this kind of property with respect to finer behavioural equivalences.

4.2 Modal mu-calculus

Although, as mentioned before, HML is expressive enough to capture bisimilarity of processes, it is not as useful to express computational properties of systems, which often consist in temporal properties possibly of infinite computations. For this reason temporal logics that make use of fixpoint operators have been introduced, usually called mu-calculi [BS06]. Among them we focus on the most classical one: the modal mu-calculus, or simply mu-calculus. The *modal mu-calculus*, denoted by $\mathcal{L}_\mu$, widely adopted in the verificiation of programs properties, was ultimately defined in [Koz83], but the use of fixpoints in program logics originated already with [dBS69, Par69]. Roughly, fixpoints allow to add recursion to logical formulae, so that they can express properties of unbounded and infinite computations.

In practice, the mu-calculus is obtained by extending HML with fixpoint operators. To properly characterise recursion, such operators resort to propositional variables taken from a denumerable set $\mathcal{X}$ of propositions.

Definition 4.4 (syntax of $\mathcal{L}_\mu$)**.** *The syntax of $\mathcal{L}_\mu$ over the sets of proposition $\mathcal{X}$ and labels Λ is defined as follows:*

$$\varphi \ ::= \ \mathsf{T} \ \mid \ \mathsf{F} \ \mid \ \varphi \wedge \varphi \ \mid \ \varphi \vee \varphi \ \mid \ \langle \mathsf{a} \rangle \varphi \ \mid \ [\mathsf{a}] \varphi \ \mid \ \mu Z.\varphi \ \mid \ \nu Z.\varphi \ \mid \ Z$$

The formulae $\mu Z.\varphi$ and $\nu Z.\varphi$ employ the newly introduced fixpoint operators, and represent the least and the greatest fixpoint, respectively, of φ with respect to a proposition Z from the set $\mathcal{X}$. Fixpoints operators act as binders for the associated proposition Z. We say that a formula of $\mathcal{L}_\mu$ is *closed* when every proposition appearing in it is bound by some fixpoint operator.

The semantics of formulae in $\mathcal{L}_\mu$ are again sets of states on which the formulae hold. However, when a formula is not closed, its denotation depends on the evaluation of its free (not bound) propositions. The semantic interpretation of propositions is given by a so-called *proposition environment* providing for each proposition a set of states. The set of all proposition environments for a PES $\mathcal{E}$ is denoted by $PEnv_\mathcal{E}$, ranged over by π.

Definition 4.5 (semantics of $\mathcal{L}_\mu$)**.** *Let $\mathcal{E}$ be a PES. The denotation of a formula in $\mathcal{L}_\mu$ is given by the function $\{\!|\cdot|\!\}^\mathcal{E} : \mathcal{L}_\mu \to PEnv_\mathcal{E} \to \mathbf{2}^{\mathcal{C}(\mathcal{E})}$ defined inductively as follows, where we*

write $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ *instead of* $\{\!|\varphi|\!\}^{\mathcal{E}}(\pi)$*:*

$$
\begin{aligned}
\{\!|\mathsf{T}|\!\}^{\mathcal{E}}_{\pi} &= \mathcal{C}(\mathcal{E}) \\
\{\!|\mathsf{F}|\!\}^{\mathcal{E}}_{\pi} &= \emptyset \\
\{\!|Z|\!\}^{\mathcal{E}}_{\pi} &= \pi(Z) \\
\{\!|\varphi_1 \wedge \varphi_2|\!\}^{\mathcal{E}}_{\pi} &= \{\!|\varphi_1|\!\}^{\mathcal{E}}_{\pi} \cap \{\!|\varphi_2|\!\}^{\mathcal{E}}_{\pi} \\
\{\!|\varphi_1 \vee \varphi_2|\!\}^{\mathcal{E}}_{\pi} &= \{\!|\varphi_1|\!\}^{\mathcal{E}}_{\pi} \cup \{\!|\varphi_2|\!\}^{\mathcal{E}}_{\pi} \\
\{\!|\langle \mathsf{a} \rangle \varphi|\!\}^{\mathcal{E}}_{\pi} &= \{C \in \mathcal{C}(\mathcal{E}) \mid \exists\, C'.\ C \xrightarrow{e}_{\mathsf{a}} C' \wedge C' \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\} \\
\{\!|[\mathsf{a}] \varphi|\!\}^{\mathcal{E}}_{\pi} &= \{C \in \mathcal{C}(\mathcal{E}) \mid \forall\, C'.\ C \xrightarrow{e}_{\mathsf{a}} C' \Rightarrow C' \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\} \\
\{\!|\mu Z.\varphi|\!\}^{\mathcal{E}}_{\pi} &= \mu(f_{\varphi,Z,\pi}) \\
\{\!|\nu Z.\varphi|\!\}^{\mathcal{E}}_{\pi} &= \nu(f_{\varphi,Z,\pi})
\end{aligned}
$$

where $f_{\varphi,Z,\pi} : \mathbf{2}^{\mathcal{C}(\mathcal{E})} \to \mathbf{2}^{\mathcal{C}(\mathcal{E})}$ *is the* semantic function *of* φ*,* Z*,* π *defined by* $f_{\varphi,Z,\pi}(S) = \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi[Z \mapsto S]}$*. When* $C \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ *we say that the PES* $\mathcal{E}$ *satisfies the formula* φ *in the configuration* C *and environment* π*. When* φ *is closed, hence the environment* π *is irrelevant, and* $\mathcal{E}$ *satisfies the formula* φ *in the empty configuration, we simply say that* $\mathcal{E}$ *satisfies* φ*.*

Note that the semantic functions $f_{\varphi,Z,\pi}$ used for fixpoint operators are guaranteed to be monotone by the definition of the semantics and the fact that the logic is in positive form. Then, since the powerset of configurations $\mathbf{2}^{\mathcal{C}(\mathcal{E})}$ ordered by subset inclusion is a complete lattice, the fixpoints $\mu(f_{\varphi,Z,\pi})$ and $\nu(f_{\varphi,Z,\pi})$ are guaranteed to exist by Theorem 2.1.

The model-checking problem in the mu-calculus, and thus in HML as its fragment, has been proved to be decidable at least on finite state systems [EL86]. Furthermore, the problem has been studied quite in depth, and many practical and efficient methods for solving it have been developed [Cle90, SW91, SS98, EJS01].

4.3 History preserving logic

In [BC10, BC14] a logic for true concurrency was introduced whose induced logical equivalence is hhp-bisimilarity. The logic was designed in a way that fragments can be identified which correspond to other classic true concurrent behavioural equivalences in the literature. The logic studied in this work is the fragment characterising hp-bisimilarity. The logic, called *history-preserving (hp-)logic* and denoted by $\mathcal{L}_{hp}$, has formulae that predicate over executability of events in computations and their dependency relations (causality and concurrency).

4.3.1 Syntax

The logic includes least and greatest fixpoint operators like the mu-calculus ones, allowing one to express temporal properties of infinite computations. Indeed, the syntax of $\mathcal{L}_{hp}$

closely resembles that of the mu-calculus. However, the modalities are much more expressive since they allow to convey true concurrency properties, being able to specify how the executed actions causally depend or not with past ones. In order to do this, logic formulae include event variables which can be bound to events in computations, taken from a fixed denumerable set *Var*, ranged over by $x, y, \ldots$. The diamond modality $\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi$ holds when in the current configuration an a-labelled event e is enabled which causally depends on the events bound to the variables in $\boldsymbol{x}$ and is concurrent with those in $\boldsymbol{y}$. Event e is executed and bound to variable z, and then the formula φ must hold in the resulting configuration. Dually, the box modality $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!] \varphi$ is satisfied when all a-labelled events causally dependent on $\boldsymbol{x}$ and concurrent with $\boldsymbol{y}$ bring to a configuration where φ holds.

As in mu-calculus, fixpoint operators resort to propositional variables. In order to let them interact correctly with event variables, whose values can be passed from an iteration to the next one in the recursion, we use abstract propositions. We fix a denumerable set $\mathcal{X}^a$ of *abstract propositions*, ranged over by X, Y, ..., that are intended to represent formulae possibly containing (unnamed) free event variables. Each abstract proposition has an *arity* $ar(Z)$, which indicates the number of free event variables in Z. An abstract proposition Z can be turn into a formula by specifying a name for its free variables. For $\boldsymbol{x}$ such that $|\boldsymbol{x}| = ar(Z)$, we write $Z(\boldsymbol{x})$ to indicate the abstract proposition Z whose free event variables are named $\boldsymbol{x}$. When $ar(Z) = 0$ we will write Z instead of $Z(\epsilon)$ omitting the trailing empty tuple of variables. We call $Z(\boldsymbol{x})$ a *proposition* and denote by $\mathcal{X}$ the set of all propositions.

Definition 4.6 (syntax of $\mathcal{L}_{hp}$)**.** *The syntax of $\mathcal{L}_{hp}$ over the sets of event variables Var, abstract propositions $\mathcal{X}^a$ and labels Λ is defined as follows:*

$$\begin{array}{rcl} \varphi & ::= & \mathsf{T} \mid \varphi \wedge \varphi \mid \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi \mid (\mu Z(\boldsymbol{x}).\varphi)(\boldsymbol{y}) \mid Z(\boldsymbol{x}) \mid \\ & & \mathsf{F} \mid \varphi \vee \varphi \mid [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!] \varphi \mid (\nu Z(\boldsymbol{x}).\varphi)(\boldsymbol{y}) \end{array}$$

Fixpoint formulae $(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})$, for $\eta \in \{\mu, \nu\}$, are referred to as *η-formulae* and hereafter η ranges over $\{\mu, \nu\}$. The formula $(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{x})$ will be abbreviated as $\eta Z(\boldsymbol{x}).\varphi$. In [BC14, BP17, BP18] only the simplified form of the fixpoint syntax was used. This slight extension allows for a simpler treatment of substitutions. Moreover, in the modalities, when $\boldsymbol{x}$ or $\boldsymbol{y}$ are empty they are omitted, e.g., we write $\langle\!\langle \mathsf{a}\, z \rangle\!\rangle \varphi$ for $\langle\!\langle \emptyset, \overline{\emptyset} < \mathsf{a}\, z \rangle\!\rangle \varphi$. We also allow to use modalities referring to events with an arbitrary label represented by a wildcard operator. Recalling that the set of labels Λ is finite, we write

$$\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \_\, z \rangle\!\rangle \varphi$$

to denote the formula $\bigvee_{\mathsf{a} \in \Lambda} \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi$, and dually $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \_\, z]\!] \varphi$ for $\bigwedge_{\mathsf{a} \in \Lambda} [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!] \varphi$.

Observe that the mu-calculus is a strict fragment of $\mathcal{L}_{hp}$. Indeed, such fragment can be obtained by requiring that the tuples of variables in the modalities are always empty, that is, as if the modalities were defined just as $\langle\!\langle \mathsf{a}\, z \rangle\!\rangle \varphi$ and $[\![\mathsf{a}\, z]\!] \varphi$. In this way event variables can be ignored, since they are never really used, and so the fragment results to be clearly equivalent to the mu-calculus.

The free event variables of a formula are defined as follows.

Definition 4.7 (free variables)**.** *The* free event variables *of a formula φ in $\mathcal{L}_{hp}$, denoted by $fv(\varphi)$, are defined inductively as:*

$$\begin{aligned}
fv(\mathsf{T}) &= \emptyset \\
fv(\mathsf{F}) &= \emptyset \\
fv(\varphi_1 \wedge \varphi_2) &= fv(\varphi_1) \cup fv(\varphi_2) \\
fv(\varphi_1 \vee \varphi_2) &= fv(\varphi_1) \cup fv(\varphi_2) \\
fv(\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\ z|\!\rangle\, \varphi) &= (fv(\varphi) \setminus \{z\}) \cup \boldsymbol{x} \cup \boldsymbol{y} \\
fv([\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\ z]\!]\, \varphi) &= (fv(\varphi) \setminus \{z\}) \cup \boldsymbol{x} \cup \boldsymbol{y} \\
fv((\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})) &= \boldsymbol{y} \\
fv(Z(\boldsymbol{y})) &= \boldsymbol{y}
\end{aligned}$$

We define the free abstract propositions of a formula in a similar way.

Definition 4.8 (free propositions)**.** *The* free abstract propositions *of a formula φ in $\mathcal{L}_{hp}$, denoted by $fp(\varphi)$, are defined inductively as:*

$$\begin{aligned}
fp(\mathsf{T}) &= \emptyset \\
fp(\mathsf{F}) &= \emptyset \\
fp(\varphi_1 \wedge \varphi_2) &= fp(\varphi_1) \cup fp(\varphi_2) \\
fp(\varphi_1 \vee \varphi_2) &= fp(\varphi_1) \cup fp(\varphi_2) \\
fp(\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\ z|\!\rangle\, \varphi) &= fp(\varphi) \\
fp([\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\ z]\!]\, \varphi) &= fp(\varphi) \\
fp((\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})) &= fp(\varphi) \setminus \{Z\} \\
fp(Z(\boldsymbol{y})) &= \{Z\}
\end{aligned}$$

The modalities act as binders for the variable representing the event executed. Similarly, a fixpoint $(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})$ binds both the abstract proposition Z and the names $\boldsymbol{x}$ specified for its event variables, which are instantiated with the values of the variables in $\boldsymbol{y}$. Intuitively, the inner fixpoint part $\eta Z(\boldsymbol{x}).\varphi$ defines a recursive formula $Z(\boldsymbol{x})$ whose free variables are then instantiated with the values of $\boldsymbol{y}$. Furthermore, in such formulae it is required that $\boldsymbol{x}$ does not include repetitions and $fv(\varphi) = \boldsymbol{x}$. We say that a formula φ of $\mathcal{L}_{hp}$ is *closed* when both $fv(\varphi)$ and $fp(\varphi)$ are empty.

Given a formula φ and variables $x, y \in Var$, we denote by $\varphi[y/x]$ the formula obtained from φ via a substitution of the free occurrences of x in φ by y. Formulae are considered up to α-conversion of bound variables and substitution is assumed to be capture free. A

function σ mapping free variables of φ to other variables will be called a *substitution* and we will denote by $\varphi\sigma$ the formula $\varphi[{}^{\sigma(fv(\varphi))}/_{fv(\varphi)}]$.

When defining the notion of substitution of formulae for propositions some care is needed in the treatment of variables. In fact, when replacing the proposition $Z(\boldsymbol{x})$ in φ with some formula ψ, actually each free occurrence $Z(\boldsymbol{y})$ of Z must be replaced by a formula obtained from ψ by renaming its free event variables to $\boldsymbol{y}$. Given a proposition $Z(\boldsymbol{x}) \in \mathcal{X}$ and a formula ψ such that $fv(\psi) \subseteq \boldsymbol{x}$, we denote by $\varphi[Z(\boldsymbol{x}) \coloneqq \psi]$ the formula obtained from φ by replacing each free occurrence $Z(\boldsymbol{y})$ of Z by $\psi[\boldsymbol{y}/\boldsymbol{x}]$.

4.3.2 Semantics

Since the logic $\mathcal{L}_{hp}$ is interpreted over PESs, the satisfaction of a formula is defined with respect to a configuration C, representing the state of the computation and a (total) function $\rho : \mathit{Var} \to E$, called an *environment*, that binds free event variables in the formula to events in C. Namely, if $\mathit{Env}_{\mathcal{E}}$ denotes the set of environments, the semantics of a formula will be a set of pairs in $\mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}$. Given a set of pairs $S \subseteq \mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}$ and two tuples of variables $\boldsymbol{x}$ and $\boldsymbol{y}$, with $|\boldsymbol{x}| = |\boldsymbol{y}|$, we define $S[\boldsymbol{y}/\boldsymbol{x}] = \{(C, \rho') \mid \exists\, (C, \rho) \in S.\ \rho(\boldsymbol{x}) = \rho'(\boldsymbol{y})\}$. In addition to environments for the event variables, the semantics of $\mathcal{L}_{hp}$ also depends on a proposition environment providing a semantic interpretation for propositions, similar to those mentioned for the mu-calculus. This time, however, we provide a formal definition.

Definition 4.9 (proposition environment)**.** *Let $\mathcal{E}$ be a PES. A* proposition environment *is a function $\pi : \mathcal{X} \to \mathbf{2}^{\mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}}$ such that for all abstract propositions $X \in \mathcal{X}^a$ and tuples of variables $\boldsymbol{x}$, $\boldsymbol{y}$ with $|\boldsymbol{x}| = |\boldsymbol{y}| = ar(X)$ it holds $\pi(X(\boldsymbol{y})) = \pi(X(\boldsymbol{x}))[\boldsymbol{y}/\boldsymbol{x}]$. The set of proposition environments, ranged over by π, is denoted $\mathit{PEnv}_{\mathcal{E}}$.*

The condition posed on proposition environments ensures that the semantics of a formula only depends on the events that the environment associates with its free variables and that it does not depend on the naming of the variables.

We will often update environments in the way described in Section 2.1. However, updates to proposition environments require a slightly different definition in order to preserve the property above. For $\pi \in \mathit{PEnv}_{\mathcal{E}}$ and $S \subseteq \mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}$, we write $\pi[X(\boldsymbol{x}) \mapsto S]$ for the proposition environment defined by

$$\begin{aligned} \pi[X(\boldsymbol{x}) \mapsto S](X(\boldsymbol{y})) &= S[\boldsymbol{y}/\boldsymbol{x}] \\ \pi[X(\boldsymbol{x}) \mapsto S](Y(\boldsymbol{y})) &= \pi(Y(\boldsymbol{y})) \quad \text{for } Y \neq X. \end{aligned}$$

It will be useful to introduce a notation describing how a pair (C, ρ) can change after a transition. For a pair $(C, \rho) \in \mathcal{C}(\mathcal{E}) \times \mathit{Env}_{\mathcal{E}}$ and variables $\boldsymbol{x}$, $\boldsymbol{y}$, z, we define the $(\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z)$-*successors* of (C, ρ), as

$$\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C, \rho) = \{(C', \rho[z \mapsto e]) \ \mid\ C \xrightarrow{\rho(\boldsymbol{x}), \overline{\rho(\boldsymbol{y})} < e}_{\mathsf{a}} C'\}.$$

In words $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C,\rho)$ consists of the pairs (C',ρ') where C' is a configuration reachable from C, by executing an event e satisfying the requirement expressed by $\boldsymbol{x},\overline{\boldsymbol{y}} < \mathsf{a}z$. Namely events in $\rho(\boldsymbol{x})$ are causes for e and events in $\rho(\boldsymbol{y})$ are concurrent with e. The environment ρ' is the update of ρ where event e has been bound to variable z.

Now we can finally give the semantics of the logic $\mathcal{L}_{hp}$.

Definition 4.10 (semantics of $\mathcal{L}_{hp}$)**.** *Let $\mathcal{E}$ be a PES. The denotation of a formula in $\mathcal{L}_{hp}$ is given by the function $\{\!|\cdot|\!\}^{\mathcal{E}} : \mathcal{L}_{hp} \to PEnv_{\mathcal{E}} \to \mathbf{2}^{\mathcal{C}(\mathcal{E})\times Env_{\mathcal{E}}}$ defined inductively as follows, where we write $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ instead of $\{\!|\varphi|\!\}^{\mathcal{E}}(\pi)$:*

$$
\begin{aligned}
\{\!|\mathsf{T}|\!\}^{\mathcal{E}}_{\pi} &= \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \\
\{\!|\mathsf{F}|\!\}^{\mathcal{E}}_{\pi} &= \emptyset \\
\{\!|Z(\boldsymbol{y})|\!\}^{\mathcal{E}}_{\pi} &= \pi(Z(\boldsymbol{y})) \\
\{\!|\varphi_1 \wedge \varphi_2|\!\}^{\mathcal{E}}_{\pi} &= \{\!|\varphi_1|\!\}^{\mathcal{E}}_{\pi} \cap \{\!|\varphi_2|\!\}^{\mathcal{E}}_{\pi} \\
\{\!|\varphi_1 \vee \varphi_2|\!\}^{\mathcal{E}}_{\pi} &= \{\!|\varphi_1|\!\}^{\mathcal{E}}_{\pi} \cup \{\!|\varphi_2|\!\}^{\mathcal{E}}_{\pi} \\
\{\!|\langle\!|\boldsymbol{x},\overline{\boldsymbol{y}} < \mathsf{a}\,z|\!\rangle\, \varphi|\!\}^{\mathcal{E}}_{\pi} &= \{(C,\rho) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \mid \mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C,\rho) \cap \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi} \neq \emptyset\} \\
\{\!|[\![\boldsymbol{x},\overline{\boldsymbol{y}} < \mathsf{a}\,z]\!]\, \varphi|\!\}^{\mathcal{E}}_{\pi} &= \{(C,\rho) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \mid \mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C,\rho) \subseteq \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\} \\
\{\!|(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})|\!\}^{\mathcal{E}}_{\pi} &= \eta(f_{\varphi,Z(\boldsymbol{x}),\pi})[\boldsymbol{y}/\boldsymbol{x}]
\end{aligned}
$$

where $f_{\varphi,Z(\boldsymbol{x}),\pi} : \mathbf{2}^{\mathcal{C}(\mathcal{E})\times Env_{\mathcal{E}}} \to \mathbf{2}^{\mathcal{C}(\mathcal{E})\times Env_{\mathcal{E}}}$ is the semantic function of φ, $Z(\boldsymbol{x})$, π defined by $f_{\varphi,Z(\boldsymbol{x}),\pi}(S) = \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi[Z(\boldsymbol{x})\mapsto S]}$. When $(C,\rho) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ we say that the PES $\mathcal{E}$ satisfies the formula φ in the configuration C and environments ρ,π. When φ is closed, hence the environments ρ,π are irrelevant, and $\mathcal{E}$ satisfies the formula φ in the empty configuration, we simply say that $\mathcal{E}$ satisfies φ.

The semantics of boolean operators is as usual. The formula $\langle\!|\boldsymbol{x},\overline{\boldsymbol{y}} < \mathsf{a}\,z|\!\rangle\, \varphi$ holds in (C,ρ) when from configuration C there is an enabled a-labelled event e that is causally dependent on (at least) the events bound to the variables in $\boldsymbol{x}$ and concurrent with (at least) those bound to the variables in $\boldsymbol{y}$ and can be executed producing a new configuration $C' = C \cup \{e\}$ which, paired with the environment $\rho' = \rho[z \mapsto e]$, satisfies the formula φ. Dually, $[\![\boldsymbol{x},\overline{\boldsymbol{y}} < \mathsf{a}\,z]\!]\, \varphi$ holds when all a-labelled events executable from C, caused by $\rho(\boldsymbol{x})$ and concurrent with $\rho(\boldsymbol{y})$ lead to a configuration where φ is satisfied.

The fixpoints corresponding to the formulae $(\eta Z(\boldsymbol{y}).\varphi)(\boldsymbol{y})$ are guaranteed to exist by Theorem 2.1, since the set $\mathbf{2}^{\mathcal{C}(\mathcal{E})\times Env_{\mathcal{E}}}$ ordered by subset inclusion is a complete lattice and the functions, of which the fixpoints are calculated, are monotone. The latter is ensured by the definition of the semantics and the fact that the logic is in positive form.

Two standard properties on the renaming of free variables arise from the definitions of substitutions and the semantics of formulae of the logic. The first one is about the correspondence between syntactical and semantic substitutions of free event variables in formulae. The second one, instead, shows how substitutions of propositions in formulae

corresponds to updates to proposition environments. Both items can be easily proved by routine inductions on the formula.

Lemma 4.1 (renaming variables and propositions)**.** *Let $\mathcal{E}$ be a PES, let π a proposition environments and let φ be a formula of $\mathcal{L}_{hp}$.*

1. *Given two tuples of variables $\boldsymbol{x}, \boldsymbol{y}$ with $|\boldsymbol{x}| = |\boldsymbol{y}|$ and $fv(\varphi) \subseteq \boldsymbol{x}$, then $\{\!|\varphi[\boldsymbol{y}/\boldsymbol{x}]|\!\}^{\mathcal{E}}_{\pi} = \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}[\boldsymbol{y}/\boldsymbol{x}]$.*
2. *For all formula ψ and abstract proposition $Z \in \mathcal{X}^a$ such that $fv(\psi) = \boldsymbol{z}$ and $ar(Z) = |\boldsymbol{z}|$, it holds $\{\!|\varphi[Z(\boldsymbol{z}) \coloneqq \psi]|\!\}^{\mathcal{E}}_{\pi} = \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi[Z(\boldsymbol{z}) \mapsto \{\!|\psi|\!\}^{\mathcal{E}}_{\pi}]}$.*

From (1) above it follows that the semantics of a formula φ in $\mathcal{L}_{hp}$ only depends on the events that the environment associates with the free variables $\boldsymbol{x}$ of the formula, i.e., if $C \in \mathcal{C}(\mathcal{E})$ and ρ, ρ' are environments such that $\rho(\boldsymbol{x}) = \rho'(\boldsymbol{x})$ pointwise then $(C, \rho) \in \{\!|\varphi|\!\}^{\mathcal{E}}$ iff $(C, \rho') \in \{\!|\varphi|\!\}^{\mathcal{E}}$. This can be seen by considering the semantics of the trivial substitutions $\{\!|\varphi[\boldsymbol{x}/\boldsymbol{x}]|\!\}^{\mathcal{E}}_{\pi} = \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}[\boldsymbol{x}/\boldsymbol{x}]$. Analogously, from (2) we have that the semantics of φ only depends on the value of the proposition environment π on the free propositions of φ, while it is independent from the interpretation of those which do not occur free in it.

4.3.3 Examples

In the logic we can easily represent the possibility of performing concurrent events, each with its own dependencies. For instance, the formula $\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle \langle\!|\boldsymbol{x}', \overline{\boldsymbol{y}', z} < \mathsf{b}\, z'|\!\rangle \varphi$ declares the existence of two concurrent events, with their own dependencies, labelled by a and b, respectively, such that if we execute such events and bind them to variables z and z', respectively, then φ holds. In particular, the ability to perform a step consisting of two concurrent events labelled by a and b is expressed by the formula $\langle\!|\mathsf{a}\, x|\!\rangle \langle\!|\overline{x} < \mathsf{b}\, y|\!\rangle \mathsf{T}$. Dually the formula $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!][\![\boldsymbol{x}', \overline{\boldsymbol{y}', z} < \mathsf{b}\, z']\!]\varphi$ states that after the execution of all pairs of concurrent events, having the specified dependencies and labelled a and b, respectively, the formula φ holds.

In this way we can easily specify properties of concurrent steps of the execution. For example, the formula $\varphi = [\![\_\, x_1]\!][\![\overline{x_1} < \_\, x_2]\!][\![\overline{x_1, x_2} < \_\, x_3]\!]\mathsf{F}$ says that in the current state there are at most two concurrently enabled events. Adding a greatest fixpoint we can define a formula $\nu Z.(\varphi \wedge [\![\_\, x]\!]Z)$ stating that the level of parallelism in a system will never exceed two.

The formula $\langle\!|\mathsf{c}\, x|\!\rangle(\langle\!|x < \mathsf{a}\, y|\!\rangle \mathsf{T} \wedge \langle\!|\overline{x} < \mathsf{b}\, z|\!\rangle \mathsf{T})$ requires that, after the execution of a c-labelled event, one can choose between a causally dependent a-labelled event and a concurrent b-labelled event. This is satisfied by $\mathcal{E}_3$ in Figure 3.1c. Instead $\langle\!|\mathsf{c}\, x|\!\rangle(\langle\!|\overline{x} < \mathsf{a}\, y|\!\rangle \mathsf{T} \wedge \langle\!|\overline{x} < \mathsf{b}\, z|\!\rangle \mathsf{T})$ requiring both events to be concurrent would be false.

As an example of property of infinite computations, consider the formula

$$[\![\mathsf{b}\, x]\!]\nu Z(x).(\langle\!|\mathsf{c}\, w|\!\rangle \langle\!|\overline{w} < \mathsf{b}\, z|\!\rangle \mathsf{T} \wedge [\![x < \mathsf{b}\, y]\!]Z(y))$$

expressing that all non-empty causal chain of b-labelled events reach a state where it is possible to execute two concurrent events labelled c and b, respectively. Then the formula holds in $\mathcal{E}_3$. Another formula satisfied by $\mathcal{E}_3$ is $\langle\!|\mathsf{c}\, x|\!\rangle\langle\!|\overline{x} < \mathsf{b}\, y|\!\rangle\nu X(x,y).(\langle\!|y, \overline{x} < \mathsf{b}\, z|\!\rangle X(x,z))$ requiring the existence of an infinite causal chain of b-labelled events, concurrent with a c-labelled event.

Now consider the formula $\mu X.(\langle\!|\_\, z|\!\rangle X \vee \langle\!|\mathsf{b}\, x|\!\rangle\langle\!|x < \mathsf{a}\, y|\!\rangle\nu Y.\langle\!|\_\, z|\!\rangle Y)$. The formula requires the existence of an infinite run containing a b-labelled event immediately followed by a causally dependent a-labelled event, and it turns out to be false in $\mathcal{E}_3$. Intuitively this is because any a-labelled event causally dependent on a b-labelled event is in conflict with the rest of the infinite chain of events, and then, after its execution, the computation is guaranteed to terminate. The formula $\langle\!|\mathsf{b}\, x|\!\rangle\nu X(x).\mu Y(x).(\langle\!|x < \mathsf{b}\, y|\!\rangle X(y) \vee \langle\!|\_\, z|\!\rangle Y(x))$, states that there exists an infinite chain of causally related, possibly non-consecutive, b-labelled events and it is satisfied by $\mathcal{E}_3$.

We next present some examples of classical properties, related to the information flow of concurrent and distributed systems, which can be naturally expressed using the true concurrent operators of $\mathcal{L}_{hp}$.

Data races typically refer to situations where the same memory location is accessed concurrently by different processes or threads, of which at least one is writing the location (see, e.g., [AHMN91]). Such race conditions are often undesirable as they can lead to unexpected or erroneous behaviours. A simple safety property ensuring the absence of data races can be expressed using the operators of the logic as follows. Let M be the (finite) set of memory locations, and let events representing instructions reading (resp. writing) a memory location $\mathsf{i} \in M$ be labelled by $\mathsf{r_i}$ (resp. $\mathsf{w_i}$). The formula $\nu X.(\bigwedge_{\mathsf{i}\in M}[\![\mathsf{w_i}\, x]\!]([\![\overline{x} < \mathsf{r_i}\, y]\!]\mathsf{F} \wedge [\![\overline{x} < \mathsf{w_i}\, y]\!]\mathsf{F}) \wedge [\![\_\, z]\!]X)$ states that from every reachable configuration it is not possible to execute two concurrent events accessing the same memory location such that at least one (the first in the formula) corresponds to a write operation, hence asserting the absence of data races.

The non-interference property of information flow in [GM82] can be intuitively described as follows. A system is viewed as consisting of components at different levels of confidentiality, in the simplest case a high part H, which intuitively should be secret, and a low part L, which is public. A system is deemed secure when there is no flow of information from H to L, something which is captured by asking that the activity of H does not determine visible effects, according to some selected observational semantics, at low level L. The notion of non-interference given in [BG09] for safe Petri nets, roughly, requires that no high level transition causes or prevents (by means of conflict) a low level one. When the net does not contain self-loops at high level (i.e., high level transitions with overlapping pre- and post-sets), assuming that each transition is uniquely labelled by either $\mathsf{H_i}$ or $\mathsf{L_i}$, depending on its level, non-interference can be expressed in $\mathcal{L}_{hp}$ by the formula $\nu X.(\bigwedge_{\mathsf{i,j}}([\![\mathsf{H_i}\, x]\!][\![x < \mathsf{L_j}\, y]\!]\mathsf{F} \wedge ([\![\mathsf{L_j}\, y]\!]\mathsf{F} \vee [\![\mathsf{H_i}\, x]\!]\langle\!|\mathsf{L_j}\, y|\!\rangle\mathsf{T})) \wedge [\![\_\, z]\!]X)$, where the first subformula $[\![\mathsf{H_i}\, x]\!][\![x < \mathsf{L_j}\, y]\!]\mathsf{F}$ requires the absence of causal dependencies from high to low level, and

the second subformula ($[\![\mathsf{L_j}\, y]\!]\mathsf{F} \vee [\![\mathsf{H_i}\, x]\!]\langle\!\langle \mathsf{L_j}\, y\rangle\!\rangle \mathsf{T}$) states that the execution of a high level transition cannot preclude a low level one previously enabled. Moreover, in [BC15] it is shown that a safe net can be transformed into another one such that interference in the original net is reduced simply to the presence of causal dependencies from high to low level in the new one. Then, given a model without self-loops that has been preprocessed as mentioned, letting transitions being labelled by the level they belong to, i.e., H or L, the non-interference property can be easily expressed as $\nu X.([\![\mathsf{H}\, x]\!][x < \mathsf{L}\, y]\!]\mathsf{F} \wedge [\![\_\, z]\!]X)$. Observe that this is also possible whenever there are no direct conflicts between high and low level transitions.

As a final example, consider the notion of atomicity based on causality defined in [FM06]. The control flow of a program with threads is modelled as a Petri net that faithfully captures the independence and interaction between threads. The causality between events in the partially ordered executions of the Petri net is used to define the notion of causal atomicity for program blocks. Roughly, a program block A is causally atomic if there are no events e_1, e_2 from A and e outside A such that $e_1 < e < e_2$. Assuming that each event is labelled by the block it is in, the causal atomicity property can be expressed in $\mathcal{L}_{hp}$ as $\nu X.([\![\_\, w]\!]X \wedge [\![\mathsf{A}\, x]\!]\nu Y(x).(\bigwedge_{\mathsf{B}\in\Lambda\setminus\{\mathsf{A}\}}[x < \mathsf{B}\, y]\!][y < \mathsf{A}\, z]\!]\mathsf{F} \wedge [\![\_\, w]\!]Y(x)))$.

4.3.4 Alternation

A feature of the logic inherited from the mu-calculus is that least and greatest fixpoint operators can be mixed in profitable ways. This makes the logic quite powerful, but has effects on the complexity of most problems regarding it. Indeed, in verification problems, the more interdependencies between fixpoints in a formula the harder the property becomes to be proved. More specifically, two (or more) fixpoints in a formula can be either independent of one another, just one dependent on the other, or they can be mutually dependent. For example, consider the formulae:

$$\begin{aligned}
\varphi_1 &= (\mu X.[\![\mathsf{a}\, x]\!]X) \wedge \nu Y.\langle\!\langle _\, y\rangle\!\rangle Y \\
\varphi_2 &= \mu X.((\nu Y.\langle\!\langle \mathsf{a}\, y\rangle\!\rangle Y) \vee [\![_\, x]\!]X) \\
\varphi_3 &= \mu X.\nu Y.(\langle\!\langle \mathsf{a}\, x\rangle\!\rangle X \vee \langle\!\langle \mathsf{b}\, y\rangle\!\rangle Y).
\end{aligned}$$

The first formula φ_1 requires the existence of an infinite computation but not one consisting of only a actions. In this formula the two fixpoints are completely independent. Intuitively, this means that they can be treated separately, even in parallel. On the other hand, in φ_2 the outer least fixpoint depends on the inner greatest one. Therefore, the computation of the outer fixpoint depends on the meaning of the inner one on the current state. Such formula is satisfied when along every computational path a state is eventually reached from where an infinite chain of a actions can be executed. The third formula φ_3 requires the existence of an infinite computation consisting of infinitely many b actions and finitely many a ones, in no specific order. In this formula the two fixpoints are mutually dependent.

This is motivated by the fact that the proposition associated with the outer one appears in the subformula of the inner one. Formally, this is called *alternation* of (least and greatest) fixpoints. The verification of formulae with alternation is much harder because, intuitively, the computation of each fixpoint depends on the computation of the other. Note that the alternating fixpoints must be of different nature, i.e., least and greatest, otherwise there is no alternation.

The levels of alternation of fixpoints in formulae define a strict hierarchy, as shown for the mu-calculus in [Bra98]. With regards to our verification purposes it is useful to define when a fixpoint formula depends on another, and what is the level of alternation of a formula. More precisely, an order induced on propositions by the nesting of fixpoints and a notion of alternation depth of formulae of $\mathcal{L}_{hp}$ will play a role in the definition of the methods described in the following chapters. Here we adapt some definitions from [EJS01] for the mu-calculus. Hereafter we will assume that in every formula different bound propositions have different names, so that we can refer to *the* fixpoint subformula quantifying an abstract proposition. This requirement can always be fulfilled by alpha-renaming the propositions in the formula. This will help us to keep the notation simpler.

First, we give the definition of the subformulae of a formula of $\mathcal{L}_{hp}$.

Definition 4.11 (subformula)**.** *The set of* subformulae *of a formula φ in $\mathcal{L}_{hp}$, denoted by $\mathsf{sf}(\varphi)$, is defined inductively as:*

$$
\begin{aligned}
\mathsf{sf}(\mathsf{T}) &= \{\mathsf{T}\} \\
\mathsf{sf}(\mathsf{F}) &= \{\mathsf{F}\} \\
\mathsf{sf}(\varphi_1 \wedge \varphi_2) &= \{\varphi_1 \wedge \varphi_2\} \cup \mathsf{sf}(\varphi_1) \cup \mathsf{sf}(\varphi_2) \\
\mathsf{sf}(\varphi_1 \vee \varphi_2) &= \{\varphi_1 \vee \varphi_2\} \cup \mathsf{sf}(\varphi_1) \cup \mathsf{sf}(\varphi_2) \\
\mathsf{sf}(\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \varphi) &= \{\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \varphi\} \cup \mathsf{sf}(\varphi) \\
\mathsf{sf}([\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \varphi) &= \{[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \varphi\} \cup \mathsf{sf}(\varphi) \\
\mathsf{sf}((\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})) &= \{(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})\} \cup \mathsf{sf}(\varphi) \\
\mathsf{sf}(Z(\boldsymbol{y})) &= \{Z(\boldsymbol{y})\}
\end{aligned}
$$

Next we formalise when a subformula of a fixpoint depends on the fixpoint itself.

Definition 4.12 (active subformula)**.** *Given an η-formula $\varphi = (\eta X(\boldsymbol{x}).\varphi')(\boldsymbol{y})$, we say that a subformula ψ of φ is a* direct active subformula*, written $\psi \sqsubset_d \varphi$, if $X \in fp(\psi)$. When $\psi \sqsubset_d^* \varphi$, we say that ψ is an* active subformula *of φ. We denote by $\mathsf{asf}_\eta(\varphi)$ the set of active η-subformulae of φ.*

Note that $\sqsubset_d$ is acyclic since it refines the subformula relation and it is irreflexive, and thus the reflexive closure of $\sqsubset_d^*$ is a partial order. Using the notion of active subformula we can define a natural number indicating the level of alternation depth of a formula.

Definition 4.13 (alternation depth)**.** *The* alternation depth *of a formula φ in $\mathcal{L}_{hp}$, written* $\mathsf{ad}(\varphi)$*, is defined inductively as:*

- *for a ν-formula φ,* $\mathsf{ad}(\varphi) = \bigsqcup\{1 + \mathsf{ad}(\psi) \mid \psi \in \mathsf{asf}_{\mu}(\varphi)\}$
- *for a μ-formula φ,* $\mathsf{ad}(\varphi) = \bigsqcup\{1 + \mathsf{ad}(\psi) \mid \psi \in \mathsf{asf}_{\nu}(\varphi)\}$
- *for any other formula φ,* $\mathsf{ad}(\varphi) = \bigsqcup\{\mathsf{ad}(\psi) \mid \psi \in \mathsf{sf}(\varphi) \setminus \{\varphi\}\}$.

Suprema are intended in the set of natural numbers $\mathbb{N}$ with the standard order, so $\bigsqcup \emptyset = 0$. E.g., by the first clause above, the alternation depth of $\nu X(\boldsymbol{x}).\varphi$ is 0 due to the absence of active μ-subformulae.

As an example, consider the formula $\langle\!|\mathsf{b}\, x|\!\rangle \nu X(x).\, \mu Y(x).(\langle\!|x < \mathsf{b}\, y|\!\rangle X(y) \vee \langle\!|\_\, z|\!\rangle Y(x))$ from the previous Subsection 4.3.3 and write it as $\langle\!|\mathsf{b}\, x|\!\rangle \varphi$, where $\varphi = \nu X(x).\psi$ and $\psi = \mu Y(x).(\langle\!|x < \mathsf{b}\, y|\!\rangle X(y) \vee \langle\!|\_\, z|\!\rangle Y(x))$. It has alternation depth 1, since ψ is a (direct) active subformula of φ, given the fact that $X \in \mathit{fp}(\psi)$. It is not difficult to see that all other formulae in Subsection 4.3.3 have instead alternation depth 0. E.g., $\mu X.(\langle\!|\_\, z|\!\rangle X \vee \langle\!|\mathsf{b}\, x|\!\rangle \langle\!|x < \mathsf{a}\, y|\!\rangle \nu Y.\langle\!|\_\, z|\!\rangle Y)$ has alternation depth 0 despite the nesting of a ν-subformula into a μ-subformula, since X does not appear free in the ν-subformula.

Hereafter, in order to have a simpler notation, especially in proofs, we will refer to η-subformulae using the abstract proposition quantified in them. So, if X and X' are abstract propositions quantified in η-subformulae $(\eta X(\boldsymbol{x}).\varphi)(\boldsymbol{y})$ and $(\eta' X'(\boldsymbol{x}').\varphi')(\boldsymbol{y}')$, we will write $\mathsf{ad}(X)$ for $\mathsf{ad}((\eta X(\boldsymbol{x}).\varphi)(\boldsymbol{y}))$ and $X \sqsubset_d X'$ for $(\eta X(\boldsymbol{x}).\varphi)(\boldsymbol{y}) \sqsubset_d (\eta' X'(\boldsymbol{x}').\varphi')(\boldsymbol{y}')$.

Part II

Model Checking

Chapter 5

Tableau system

The first model-checking procedure that we introduce is based on tableaux inspired by those in [Cle90, SW91] for the propositional mu-calculus. The resulting method, while not being one of the most efficient, allows for a clear and intuitive presentation of the solution of the model-checking problem. Its development plays the role of preliminary, yet thorough, study of the decidability of the model-checking in the logic $\mathcal{L}_{hp}$.

A *tableau* [Hin55, Bet55] is a (finite) proof tree whose nodes are labelled by sequents. Usually sequents are of the kind $s \models^{\mathcal{M}} \varphi$, where s is a state of the model $\mathcal{M}$ and φ is a formula, but they can contain further elements if needed. The fact that a state s satisfies a formula φ amounts to the existence of a tableau rooted in $s \models^{\mathcal{M}} \varphi$ and successful, according to a suitable definition. Therefore, the model-checking problem, for testing whether a logical property φ is satisfied by a system model $\mathcal{M}$, is solved by searching for a successful tableau for the sequent $s_0 \models^{\mathcal{M}} \varphi$, where s_0 is the initial state of $\mathcal{M}$.

The construction of tableaux involves a set of rules, called tableau rules, which, roughly, reduce the satisfaction of a formula to that of proper subformulae. Intuitively, this allows to divide the original problem into smaller ones, until trivial cases are reached.

In this chapter we present a tableau system for model-checking formulae of the logic $\mathcal{L}_{hp}$ over semantic models given in the form of event structures. We prove that the tableau rules are sound and complete over a class of regular event structures.

5.1 Tableau rules

The tableau system works on *sequents* $C, \rho, \Delta \models^{\mathcal{E}} \varphi$, where φ is a formula of $\mathcal{L}_{hp}$, $C \in \mathcal{C}(\mathcal{E})$ is a configuration, ρ is an environment and Δ is a finite set of definitions of the form $Z(\boldsymbol{x}) = \psi$, where $\boldsymbol{x}$ is a tuple of (distinct) event variables and $fv(\psi) \subseteq \boldsymbol{x}$. Intuitively, a sequent $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ expresses that the formula φ holds in state C when free event variables are associated with events as specified by ρ and free propositions represent formulae as indicated by Δ. The triple $\Gamma = \langle C, \rho, \Delta \rangle$ is called the *context* and φ the *consequent*. In a tableaux built starting from a closed formula, each proposition $Z(\boldsymbol{x})$ in Δ will be defined

as the corresponding fixpoint subformula $\eta Z(\boldsymbol{x}).\,\psi$ where Z is quantified. For technical reasons, we also allow Δ to bind propositions to general formulae. This will be later used in the proofs of soundness and completeness, where we will need to bind propositions to suitably defined fixpoint approximants.

It will be useful to see Δ as a function. So, when $Z(\boldsymbol{x}) = \psi$ is in Δ, we write $Z(\boldsymbol{x}) \in dom(\Delta)$ and we denote the formula ψ as $\Delta(Z(\boldsymbol{x}))$. Furthermore, we denote by $\Delta[Z(\boldsymbol{x}) \mapsto \psi]$ the updated definition set obtained from Δ by removing the previous definition of Z, if any, and adding $Z(\boldsymbol{x}) = \psi$. We assume that for each abstract proposition there is at most one definition, i.e., if $Z(\boldsymbol{x}) = \psi$ and $Z(\boldsymbol{x}') = \psi'$ are in Δ, then $\boldsymbol{x} = \boldsymbol{x}'$ and $\psi = \psi'$.

We will work with a subclass of sequents where iterating the substitution of free propositions in the formula φ with their definitions in Δ, we eventually obtain a formula without free propositions. More precisely, let $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ be a sequent. For formulae ψ, ψ' we write $\psi \to_\Delta \psi'$ when $\psi' = \psi[Z(\boldsymbol{x}) := \Delta(Z(\boldsymbol{x}))]$ for some $Z(\boldsymbol{x}) \in dom(\Delta)$. Then the class of sequents we will work with can be defined as follows.

Definition 5.1 (well-formed sequent)**.** *A sequent $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ is* well-formed *if $\varphi \to_\Delta^* \psi$ for some ψ such that $fp(\psi) = \emptyset$. In this case, we denote the formula ψ by $(\varphi)_\Delta$.*

Since we assumed that for each abstract proposition there is at most one definition in Δ, it is easy to realise that the formula $(\varphi)_\Delta$ is well-defined, i.e., when it exists it is unique (up to alpha-renaming of event variables).

The truth of a well-formed sequent can be now defined in the obvious way.

Definition 5.2 (truth)**.** *A well-formed sequent $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ is* true *if $(C, \rho) \in \{\!|(\varphi)_\Delta|\!\}^{\mathcal{E}}_\pi$, where π is any proposition environment.*

Observe that in the definition above, the proposition environment π is irrelevant, since $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ is well-formed and thus the formula $(\varphi)_\Delta$ does not include any free proposition.

The *tableau rules* will be of the form

$$\frac{C, \rho, \Delta \;\models^{\mathcal{E}}\; \varphi}{C_1, \rho_1, \Delta_1 \;\models^{\mathcal{E}}\; \varphi_1 \;\ldots\; C_k, \rho_k, \Delta_k \;\models^{\mathcal{E}}\; \varphi_k}\;\; \delta$$

where $k > 0$ and δ is a possible side condition required to hold. The intuition is that the truth of the sequent in the premise reduces to the truth of those in the conclusion. In the following the index $\mathcal{E}$, when clear from the context, will be dropped. Moreover, all sequents will be, sometimes tacitly, assumed to be well-formed.

The tableau rules for the logic $\mathcal{L}_{hp}$, are reported in Table 5.1. The rules for propositional connectives are straightforward. For instance the truth of $\varphi \vee \psi$ is reduced to the truth of either φ or ψ. The context is not altered.

Similarly, the truth of a modal formula is reduced to the truth of the subformula after the modal operator, in suitable contexts chosen according to the semantics. For the formula $\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \varphi$ the rule $(\Diamond)$ prescribes the existence of at least one transition leading to a

$$(\wedge)\ \frac{C,\rho,\Delta \ \models\ \varphi \ \wedge\ \psi}{C,\rho,\Delta \ \models\ \varphi \qquad C,\rho,\Delta \ \models\ \psi}$$

$$(\vee_{\mathsf{L}})\ \frac{C,\rho,\Delta \ \models\ \varphi \ \vee\ \psi}{C,\rho,\Delta \ \models\ \varphi} \qquad\qquad (\vee_{\mathsf{R}})\ \frac{C,\rho,\Delta \ \models\ \varphi \ \vee\ \psi}{C,\rho,\Delta \ \models\ \psi}$$

$$(\Diamond)\ \frac{C,\rho,\Delta \ \models\ \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi}{C',\rho[z \mapsto e],\Delta \ \models\ \varphi} \qquad \exists\, e.\ C \xrightarrow{\rho(\boldsymbol{x}),\overline{\rho(\boldsymbol{y})} < e}_{\mathsf{a}} C'$$

$$(\Box)\ \frac{C,\rho,\Delta \ \models\ [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!] \varphi}{C_1,\rho_1,\Delta \ \models\ \varphi \quad \ldots \quad C_n,\rho_n,\Delta \ \models\ \varphi}$$

$$\text{where } \{(C_1,\rho_1),\ldots,(C_n,\rho_n)\} = \mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C,\rho)$$

$$(\mathsf{Int})\ \frac{C,\rho,\Delta \ \models\ (\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})}{C,\rho,\Delta' \ \models\ Z(\boldsymbol{y})} \qquad \Delta' = \Delta[Z(\boldsymbol{x}) \mapsto \eta Z(\boldsymbol{x}).\varphi]$$

$$(\mathsf{Unf}_{\eta})\ \frac{C,\rho,\Delta \ \models\ Z(\boldsymbol{z})}{C,\rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})],\Delta \ \models\ \varphi} \qquad \neg\gamma \text{ and } \Delta(Z(\boldsymbol{x})) = \eta Z(\boldsymbol{x}).\varphi$$

Table 5.1: The tableau rules for the logic $\mathcal{L}_{hp}$.

context where φ holds. The rule ($\Box$) for $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!] \varphi$ requires that all transitions lead to contexts where φ holds. Observe that, working with boundedly branching PESs rule ($\Box$) always has a finite number of sequents in the conclusion.

Rule (Int) reduces the truth of a fixpoint formula $(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})$ to that of the proposition $Z(\boldsymbol{y})$ in a context where the definition list Δ is extended by defining $Z(\boldsymbol{x})$ as the corresponding fixpoint subformula. We will say that the node $C,\rho,\Delta \models (\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})$ *introduces* the abstract proposition Z.

Rule (Unf_{η}) is applied when the consequent is a proposition $Z(\boldsymbol{z})$: it just unfolds the proposition according to its definition in Δ. When the tableau is rooted in a closed formula, the proposition will be always bound to a fixpoint formula. The component γ in the side condition will be described in the next section. It is called *stop condition* and, as suggested by its name, it is intended to prevent the reduction to continue unboundedly.

A simple example of tableau is depicted in Figure 5.1. Such tableau proves that the formula $\langle\!\langle \mathsf{c}\, x \rangle\!\rangle(\langle\!\langle x < \mathsf{a}\, y \rangle\!\rangle \mathsf{T} \wedge \langle\!\langle \overline{x} < \mathsf{b}\, z \rangle\!\rangle \mathsf{T})$ from Subsection 4.3.3 was correctly claimed to be satisfied by the PES $\mathcal{E}_3$ in Figure 3.1c. Notice how the state of the computation and the environment are updated after each application of a rule for modal operators (rule ($\Diamond$) in this case). For instance, in the left-hand branch of the tableau, the configuration $\{\mathsf{c}\}$ is updated to $\{\mathsf{c},\mathsf{a}\}$ after executing the event a, caused by the event $\rho(x)$, which is then bound

to variable y in $\rho[x \mapsto \mathsf{c}, y \mapsto \mathsf{a}]$. No rule can be applied to the sequents labelling the leaves, whose formula T holds in any state.

$$(\Diamond)\ \frac{\emptyset, \rho, \emptyset \models \langle\!\langle \mathsf{c}\, x \rangle\!\rangle(\langle\!\langle x < \mathsf{a}\, y \rangle\!\rangle \mathsf{T} \wedge \langle\!\langle \overline{x} < \mathsf{b}\, z \rangle\!\rangle \mathsf{T})}{\{\mathsf{c}\}, \rho[x \mapsto \mathsf{c}], \emptyset \models \langle\!\langle x < \mathsf{a}\, y \rangle\!\rangle \mathsf{T} \wedge \langle\!\langle \overline{x} < \mathsf{b}\, z \rangle\!\rangle \mathsf{T}}$$

$$(\wedge)\ \frac{\{\mathsf{c}\}, \rho[x \mapsto \mathsf{c}], \emptyset \models \langle\!\langle x < \mathsf{a}\, y \rangle\!\rangle \mathsf{T} \qquad \{\mathsf{c}\}, \rho[x \mapsto \mathsf{c}], \emptyset \models \langle\!\langle \overline{x} < \mathsf{b}\, z \rangle\!\rangle \mathsf{T}}{\{\mathsf{c}\}, \rho[x \mapsto \mathsf{c}], \emptyset \models \langle\!\langle x < \mathsf{a}\, y \rangle\!\rangle \mathsf{T} \wedge \langle\!\langle \overline{x} < \mathsf{b}\, z \rangle\!\rangle \mathsf{T}}$$

$$(\Diamond)\ \frac{}{\{\mathsf{c}, \mathsf{a}\}, \rho[x \mapsto \mathsf{c}, y \mapsto \mathsf{a}], \emptyset \models \mathsf{T}} \qquad (\Diamond)\ \frac{}{\{\mathsf{c}, \mathsf{b}\}, \rho[x \mapsto \mathsf{c}, z \mapsto \mathsf{b}], \emptyset \models \mathsf{T}}$$

Figure 5.1: A tableau for $\langle\!\langle \mathsf{c}\, x \rangle\!\rangle(\langle\!\langle x < \mathsf{a}\, y \rangle\!\rangle \mathsf{T} \wedge \langle\!\langle \overline{x} < \mathsf{b}\, z \rangle\!\rangle \mathsf{T})$ in $\mathcal{E}_3$.

To see that it is appropriate to unfold a proposition $Z(\boldsymbol{z})$ into the inner formula φ of the fixpoint where the proposition is quantified, we observe that, in a well-formed sequent, the denotation of $(\varphi)_\Delta$ is the same of the denotation of $(Z(\boldsymbol{x}))_\Delta$. Therefore, the truth of the corresponding sequents is the same. Formally, this technical result will later be used to show that the unfolding rule is in fact sound.

Lemma 5.1 (fixpoints and substitutions)**.** *Let $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ be a well-formed sequent and assume that $\Delta(Z(\boldsymbol{x})) = \eta Z(\boldsymbol{x}).\varphi$. Then $\{\!|(\varphi)_\Delta|\!\}^{\mathcal{E}}_\pi = \{\!|(Z(\boldsymbol{x}))_\Delta|\!\}^{\mathcal{E}}_\pi$.*

Proof. Let $\{Y_1, \ldots, Y_n\} = fp(\varphi) \setminus \{Z\}$ and let

$$\varphi' = \varphi[Y_1(\boldsymbol{y}_1) \coloneqq (Y_1(\boldsymbol{y}_1))_\Delta] \ldots [Y_n(\boldsymbol{y}_n) \coloneqq (Y_n(\boldsymbol{y}_n))_\Delta],$$

Observe that

$$(Z(\boldsymbol{x}))_\Delta = (\Delta(Z(\boldsymbol{x})))_\Delta = (\eta Z(\boldsymbol{x}).\varphi)_\Delta = \eta Z(\boldsymbol{x}).\varphi' \tag{5.1}$$

and thus

$$(\varphi)_\Delta = (\varphi')_\Delta = \varphi'[Z(\boldsymbol{x}) \coloneqq (Z(\boldsymbol{x}))_\Delta] = \varphi'[Z(\boldsymbol{x}) \coloneqq \eta Z(\boldsymbol{x}).\varphi']. \tag{5.2}$$

Putting things together we have

$$\begin{aligned}
\{\!|(Z(\boldsymbol{x}))_\Delta|\!\}^{\mathcal{E}}_\pi &= \{\!|\eta Z(\boldsymbol{x}).\varphi'|\!\}^{\mathcal{E}}_\pi && \text{[by (5.1)]} \\
&= \{\!|\varphi'|\!\}^{\mathcal{E}}_{\pi[Z(\boldsymbol{x}) \mapsto \{\!|\eta Z(\boldsymbol{x}).\varphi'|\!\}^{\mathcal{E}}_\pi]} && \text{[by the semantics of fixpoints]} \\
&= \{\!|\varphi'[Z(\boldsymbol{x}) \coloneqq \eta Z(\boldsymbol{x}).\varphi']|\!\}^{\mathcal{E}}_\pi && \text{[by Lemma 4.1(2)]} \\
&= \{\!|(\varphi)_\Delta]|\!\}^{\mathcal{E}}_\pi && \text{[by (5.2)]}
\end{aligned}$$

□

The tableau rules satisfy a form of backwards soundness, i.e., we can show that the premise is true when all the sequents in the conclusion are. This property will play a basic role in Section 5.3 for proving the soundness of the tableau system.

Lemma 5.2 (backwards soundness)**.** *Every rule of the tableau system for $\mathcal{L}_{hp}$ is backwards sound.*

Proof. For each rule we have to show that if the sequents in the conclusion are true then also the sequent in the premise is true. The proof follows almost directly from the definition of the semantics of the logic. We only inspect some cases:

- Consider the rule
$$(\wedge)\ \frac{C, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi \ \wedge\ \psi}{C, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi \qquad C, \rho, \Delta \ \models^{\mathcal{E}}\ \psi}$$
Assume that the sequents in the conclusion are true, i.e., that $(C, \rho) \in \{\!|(\varphi)_\Delta|\!\}^{\mathcal{E}}_\pi$ and $(C, \rho) \in \{\!|(\psi)_\Delta|\!\}^{\mathcal{E}}_\pi$, for $\pi \in PEnv_{\mathcal{E}}$. Just observe that $(\varphi \wedge \psi)_\Delta = (\varphi)_\Delta \wedge (\psi)_\Delta$. Then we immediately conclude that $(C, \rho) \in \{\!|(\varphi \wedge \psi)_\Delta|\!\}^{\mathcal{E}}_\pi$, i.e., that the sequent $C, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi \ \wedge\ \psi$ is true, as desired.

- Consider the rule
$$(\Diamond)\ \frac{C, \rho, \Delta \ \models^{\mathcal{E}}\ \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle\, \varphi}{C', \rho[z \mapsto e], \Delta \ \models^{\mathcal{E}}\ \varphi}$$
where $e \in \mathcal{E}[C]$ and $C \xrightarrow{\rho(\boldsymbol{x}), \overline{\rho(\boldsymbol{y})} < e}_{\mathsf{a}} C'$.

  Assume that the sequent in the conclusion is true, i.e., $(C', \rho[z \mapsto e]) \in \{\!|(\varphi)_\Delta|\!\}^{\mathcal{E}}_\pi$, for $\pi \in PEnv_{\mathcal{E}}$. By definition of the semantics, we immediately deduce that $(C, \rho) \in \{\!|\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle\, (\varphi)_\Delta|\!\}^{\mathcal{E}}_\pi$. Since $(\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle\, \varphi)_\Delta = \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle\, (\varphi)_\Delta$, this proves that the sequent $C, \rho, \Delta \ \models^{\mathcal{E}}\ \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle\, \varphi$ in the premise is true.

- Consider the rule:
$$(\mathsf{Int})\ \frac{C, \rho, \Delta \ \models^{\mathcal{E}}\ (\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})}{C, \rho, \Delta' \ \models^{\mathcal{E}}\ Z(\boldsymbol{y})}$$
where $\Delta' = \Delta[Z(\boldsymbol{x})x \mapsto \eta Z(\boldsymbol{x}).\varphi]$.

  Assume that the sequent in the conclusion is true, i.e., $(C, \rho) \in \{\!|(Z(\boldsymbol{y}))_{\Delta'}|\!\}^{\mathcal{E}}_\pi$, for $\pi \in PEnv_{\mathcal{E}}$. Just observe that $(Z(\boldsymbol{y}))_{\Delta'} = ((\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y}))_{\Delta'} = ((\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y}))_\Delta$ where the last equality is motivated by the fact that Δ and Δ' differ only on Z, which is not free in $(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})$ (and neither it is in any formula ψ such that $(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y}) \to^*_\Delta \psi$). Hence
$$\{\!|(Z(\boldsymbol{y}))_{\Delta'}|\!\}^{\mathcal{E}}_\pi = \{\!|((\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y}))_\Delta|\!\}^{\mathcal{E}}_\pi$$
  This immediately implies that the sequent $C, \rho, \Delta \ \models^{\mathcal{E}}\ (\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})$ in the premise is true.

- Consider the rule:
$$(\mathsf{Unf}_\eta)\ \frac{C, \rho, \Delta \ \models^{\mathcal{E}}\ Z(\boldsymbol{z})}{C, \rho', \Delta \ \models^{\mathcal{E}}\ \varphi}$$
where $\Delta(Z(\boldsymbol{x})) = \eta Z(\boldsymbol{x}).\varphi$, for $\rho' = \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]$.

Assume that the sequent in the conclusion is true, i.e., $(C, \rho') \in \{\!|(\varphi)_\Delta|\!\}^{\mathcal{E}}_{\pi}$, for some $\pi \in PEnv_{\mathcal{E}}$. By Lemma 5.1, $\{\!|(\varphi)_\Delta|\!\}^{\mathcal{E}}_{\pi} = \{\!|(Z(\boldsymbol{x}))_\Delta|\!\}^{\mathcal{E}}_{\pi}$. Thus we have

$$\begin{aligned}
&(C, \rho') = \\
&\quad (C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]) \in \{\!|(Z(\boldsymbol{x}))_\Delta|\!\}^{\mathcal{E}}_{\pi} = \\
&\quad = \{\!|(Z(\boldsymbol{z})[\boldsymbol{x}/\boldsymbol{z}])_\Delta|\!\}^{\mathcal{E}}_{\pi} = \\
&\quad = \{\!|(Z(\boldsymbol{z}))_\Delta[\boldsymbol{x}/\boldsymbol{z}]|\!\}^{\mathcal{E}}_{\pi} = \\
&\quad = \{\!|(Z(\boldsymbol{z}))_\Delta|\!\}^{\mathcal{E}}_{\pi}[\boldsymbol{x}/\boldsymbol{z}]
\end{aligned}$$

Thus $(C, \rho) \in \{\!|(Z(\boldsymbol{z}))_\Delta|\!\}^{\mathcal{E}}_{\pi}$ which means that the sequent in the premise is true.

□

5.2 The stop condition

The unfolding rule (Unf_η), in absence of any countermeasure, makes the tableau construction procedure possibly non-terminating. To solve this problem a side condition, called the *stop condition*, is added that prevents the application of the rule when a context is reached that is equivalent, in a suitable sense to be defined, to a context occurring in an ancestor of the current node, for the same fixpoint formula.

The notion of equivalence between contexts has to be chosen carefully not to break the soundness of the technique. Intuitively, two contexts should be equivalent for a formula if either both satisfy it or none of them does. The satisfaction of a formula of $\mathcal{L}_{hp}$ in a given configuration, surely depends on the future of the configuration. So, two contexts C, ρ, Δ and C', ρ', Δ' for a formula φ, in order to be considered equivalent, must have isomorphic futures, i.e., the residuals $\mathcal{E}[C]$ and $\mathcal{E}[C']$ must be isomorphic as PESs. This is not sufficient, though, since φ can express history dependent properties that relates the future to the past events via its free event variables. Hence we additionally ask that event variables of φ are mapped to events in C and C', respectively, which have the same relations (causality and concurrency) with the corresponding futures. This motivates the notion of pointed configuration introduced below, which, intuitively, is a configuration where some past events are singled out.

Definition 5.3 (pointed configuration)**.** *Let $\mathcal{E}$ be a PES and let V be a set. A* V-pointed configuration *is a pair $\langle C, \zeta \rangle$ where $C \in \mathcal{C}(\mathcal{E})$ and $\zeta : V \to C$ is a function.*

The notion will be used instantiating V with the set of variables free in a formula and ζ with the (restriction to such variables of the) environment in a context.

Now we can define the equivalence that we will use over contexts seen as configurations pointed by the free variables of a formula. In general, we say that two V-pointed configurations have isomorphic pointed residuals when their residuals are related by a bijection

ensuring that events pointed by the same $x \in V$ have the same causal relations with the future.

Definition 5.4 (isomorphism of pointed residuals)**.** *Let $\mathcal{E}$ be a PES, let V be a set, and let $\langle C, \zeta \rangle$ and $\langle C', \zeta' \rangle$ be two V-pointed configurations of $\mathcal{E}$. We say that $\langle C, \zeta \rangle$, $\langle C', \zeta' \rangle$ have isomorphic residuals, written $\langle C, \zeta \rangle \approx_r \langle C', \zeta' \rangle$, if there is an isomorphism of the residuals $\iota : \mathcal{E}[C] \to \mathcal{E}[C']$ such that for all $x \in V$, $e \in \mathcal{E}[C]$ we have $\zeta(x) \leq e$ iff $\zeta'(x) \leq \iota(e)$.*

We will show that pointed configurations with isomorphic residuals satisfy exactly the same formulae, when the free event variables of the formulae are associated with the pointed events. For properly stating such property, since we will work with formulae containing free propositions, we need to restrict to proposition environments that are well-behaved, in the sense formalised below, with respect to the free propositions of the formulae of interest.

Definition 5.5 (saturated proposition environment)**.** *Let $\mathcal{E}$ be a PES. Given a formula φ and a proposition environment $\pi \in PEnv_{\mathcal{E}}$, we say that π is* saturated for φ *when for all $Z \in fp(\varphi)$ and $(C, \rho), (C', \rho') \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$, with $\langle C, \rho|_{fv(\varphi)} \rangle \approx_r \langle C', \rho'|_{fv(\varphi)} \rangle$, if $(C, \rho) \in \pi(Z(\boldsymbol{y}))$, for some $\boldsymbol{y}$, then $(C', \rho') \in \pi(Z(\boldsymbol{y}))$.*

In words, in order to be saturated for a formula φ, a proposition environment must assign to each free proposition in φ an evaluation that respects the equivalence $\approx_r$ over $fv(\varphi)$-pointed configurations. Then we have the result we were looking for.

Lemma 5.3 (equisatisfaction in pointed configurations with isomorphic residuals)**.** *Let $\mathcal{E}$ be a PES, let φ be a formula of $\mathcal{L}_{hp}$, let $\pi \in PEnv_{\mathcal{E}}$ be a proposition environment saturated for φ, and let $(C_1, \rho_1), (C_2, \rho_2) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$. If $\langle C_1, \rho_1|_{fv(\varphi)} \rangle \approx_r \langle C_2, \rho_2|_{fv(\varphi)} \rangle$ then $(C_1, \rho_1) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ iff $(C_2, \rho_2) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$.*

Proof. Assume $\langle C_1, \rho_1|_{fv(\varphi)} \rangle \approx_r \langle C_2, \rho_2|_{fv(\varphi)} \rangle$, via an isomorphism $\iota : \mathcal{E}[C_1] \to \mathcal{E}[C_2]$. We prove that if $(C_1, \rho_1) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ then $(C_2, \rho_2) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$. Since the isomorphism ι is bijective, the other implication follows by symmetry. The proof proceeds by induction on the formula φ. We discuss only some representative cases:

- $\varphi = Z(\boldsymbol{x})$
  Let $(C_1, \rho_1) \in \{\!|Z(\boldsymbol{x})|\!\}^{\mathcal{E}}_{\pi} = \pi(Z(\boldsymbol{x}))$. Since π is saturated for $Z(\boldsymbol{x})$, by definition, we get that also $(C_2, \rho_2) \in \pi(Z(\boldsymbol{x}))$, as desired.

- $\varphi = \psi_1 \wedge \psi_2$
  If $(C_1, \rho_1) \in \{\!|\psi_1 \wedge \psi_2|\!\}^{\mathcal{E}}_{\pi}$, then by definition of the semantics $(C_1, \rho_1) \in \{\!|\psi_1|\!\}_{\pi}$ and $(C_1, \rho_1) \in \{\!|\psi_2|\!\}_{\pi}$. By inductive hypothesis we obtain $(C_2, \rho_2) \in \{\!|\psi_1|\!\}_{\pi}$ and $(C_2, \rho_2) \in \{\!|\psi_2|\!\}_{\pi}$ and thus $(C_2, \rho_2) \in \{\!|\psi_1 \wedge \psi_2|\!\}_{\pi}$.

- $\varphi = \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \psi$
  Assume that $(C_1, \rho_1) \in \{\!|\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \psi|\!\}_{\pi}$. By definition of the semantics there exists an event $e \in \mathcal{E}[C_1]$ such that $C_1 \xrightarrow{\rho_1(\boldsymbol{x}), \overline{\rho_1(\boldsymbol{y})} < e}_{\mathsf{a}} C_1'$ and $(C_1', \rho_1') \in \{\!|\psi|\!\}_{\pi}$ with

$\rho_1' = \rho_1[z \mapsto e]$. Since ι is an isomorphism of residuals of pointed configurations $\langle C_1, \rho_1|_{fv(\varphi)}\rangle \approx_r \langle C_2, \rho_2|_{fv(\varphi)}\rangle$, we readily deduce that $\iota(e)$ is enabled at C_2 and

$$C_2 \xrightarrow{\rho_2(\boldsymbol{x}),\overline{\rho_2(\boldsymbol{y})} < \iota(e)}_{\mathsf{a}} C_2' \tag{5.3}$$

Consider the restriction of ι to $\mathcal{E}[C_1']$ and call it $\iota' : \mathcal{E}[C_1'] \to \mathcal{E}[C_2']$. Clearly ι' is an isomorphism of PESs. Additionally, if we let $\rho_2' = \rho_2[z \mapsto \iota(e)]$, it is easy to see that for all $x \in fv(\psi)$, $e_1 \in \mathcal{E}[C_1']$, it holds $\rho_1'(x) \leq e_1$ iff $\rho_2'(x) \leq \iota'(e_1)$. Hence $\langle C_1', \rho_1'|_{fv(\psi)}\rangle \approx_r \langle C_2', \rho_2'|_{fv(\psi)}\rangle$. In fact, for all $x \in fv(\psi) \subseteq fv(\varphi) \cup \{z\}$:

1. if $x \in fv(\varphi)$ we can observe that $e_1 \in \mathcal{E}[C_1'] \subseteq \mathcal{E}[C_1]$ and $\rho_i(x) = \rho_i'(x)$ for $i \in \{1,2\}$. Then the desired property follows from the fact that ι is an isomorphism of residuals of the pointed configurations $\langle C_1, \rho_1|_{fv(\varphi)}\rangle$ and $\langle C_2, \rho_2|_{fv(\varphi)}\rangle$.
2. if $x = z$ then $\rho_1'(z) = e \in \mathcal{E}[C_1]$, $\rho_2'(z) = \iota(e) \in \mathcal{E}[C_2]$. Since $\iota : \mathcal{E}[C_1] \to \mathcal{E}[C_2]$ is an isomorphism of PESs, $\rho_1(x) \leq e_1'$ iff $\rho_2(x) = \iota(e) \leq \iota(e_1')$.

Since $\langle C_1', \rho_1'|_{fv(\psi)}\rangle \approx_r \langle C_2', \rho_2'|_{fv(\psi)}\rangle$, by inductive hypothesis, $(C_2', \rho_2') \in \{\!|\psi|\!\}_\pi$. Recalling (5.3), we conclude $(C_2, \rho_2) \in \{\!|\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z\rangle\!\rangle \psi|\!\}_\pi$, as desired.

- $\varphi = (\eta Z(\boldsymbol{x}).\psi)(\boldsymbol{y})$
  We know that $\{\!|(\eta Z(\boldsymbol{x}).\psi)(\boldsymbol{y})|\!\}_\pi = \{\!|\eta Z(\boldsymbol{x}).\psi|\!\}_\pi[\boldsymbol{y}/\!\!/\boldsymbol{x}]$. Thus it suffices to show that if $(C_1, \rho_1) \in \{\!|\eta Z(\boldsymbol{x}).\psi|\!\}_\pi$ then $(C_2, \rho_2) \in \{\!|\eta Z(\boldsymbol{x}).\psi|\!\}_\pi$.

  We focus on the case $\eta = \mu$ (the case $\eta = \nu$ is perfectly dual). Observe that, by definition of the semantics, we have

  $$\{\!|\eta Z(\boldsymbol{x}).\psi|\!\}_\pi = \mu f$$

  where $f = f_{\psi, Z(\boldsymbol{x}), \pi} : \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \to \mathbf{2}^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}$ is defined by $f(S) = \{\!|\psi|\!\}^{\mathcal{E}}_{\pi[Z(\boldsymbol{x}) \mapsto S]}$.
  By Theorem 2.2 we know that $\mu f = f^\gamma(\emptyset)$ for some ordinal γ. The thesis follows by showing that for any ordinal β, if $(C_1, \rho_1) \in f^\beta(\emptyset)$ then $(C_2, \rho_2) \in f^\beta(\emptyset)$. The proof is by induction on β. The base case $\beta = 0$ is trivial since $f^0(\emptyset) = \emptyset$. For a successor ordinal, if $(C_1, \rho_1) \in f^{\beta+1}(\emptyset) = f(f^\beta(\emptyset)) = \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi[Z(\boldsymbol{x}) \mapsto f^\beta(\emptyset)]}$. Observe that $\pi' = \pi[Z(\boldsymbol{x}) \mapsto f^\beta(\emptyset)]$ is saturated for ψ, as it easily follows from the fact that π is saturated for φ and free variables of ψ are $fp(\psi) \subseteq fp(\varphi) \cup \{Z\}$ and $fv(\varphi) = fv(\psi) = \boldsymbol{x}$. Thus the only doubt could concern the proposition Z, but the condition is satisfied also for Z, by the inner inductive hypothesis applied to $f^\beta(\emptyset)$. Therefore, we can apply the outer inductive hypothesis to ψ to deduce that $(C_2, \rho_2) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi[Z(\boldsymbol{x}) \mapsto f^\beta(\emptyset)]} = f^{\beta+1}(\emptyset)$. The case of a limit ordinal is straightforward.

□

Recall that the proposition environment is irrelevant for establishing the truth of a well-formed sequent because the instantiation of its formula is always closed. This goes well

with the result above, since in fact every proposition environment is vacuously saturated for a closed formula. Therefore, for well-formed sequents, when two contexts C, ρ, Δ and C', ρ', Δ' for the same formula φ, seen as pointed configurations, have isomorphic residuals $\langle C, \rho|_{fv(\varphi)}\rangle \approx_r \langle C', \rho'|_{fv(\varphi)}\rangle$, the lemma above ensures that the corresponding sequents have the same truth value, as long as the two instantiations of the formula $(\varphi)_\Delta$ and $(\varphi)_{\Delta'}$ are the same. The latter is guaranteed to happen if the sequents come from the same tableau built starting from a closed formula.

We can now proceed with the definition of the stop condition. Given a tableau for a closed formula and a node labelled by $C, \rho, \Delta \models Z(\boldsymbol{y})$, with $Z(\boldsymbol{x}) = \psi$ in Δ, necessarily $\psi = \eta Z(\boldsymbol{x}).\varphi$ and the node has some ancestor introducing Z. We will denote by $\Delta^\uparrow(Z)$ the closest of such ancestors. The notation is slightly abused since $\Delta^\uparrow(Z)$ is not determined by Δ but by the tableau itself, still it is suggestive since $\Delta^\uparrow(Z)$ is the node where the definition of Z in Δ has been updated more recently. More precisely, one can think that $\Delta^\uparrow(\cdot)$ is an additional component of the sequent.

Observe that, once it has been set, the definition $Z(\boldsymbol{x}) = \psi$ of a proposition remains unchanged in the tableau and it will be referred as the definition of Z in the tableau. Instead, the component $\Delta^\uparrow(Z)$ changes at each application of rule (Int).

Definition 5.6 (stop condition)**.** *The* stop condition γ *for rule* (Unf_η) *in Table 5.1 is as follows:*

there is an ancestor of the premise $C, \rho, \Delta \models Z(\boldsymbol{z})$ *labelled* $C', \rho', \Delta' \models Z(\boldsymbol{y})$*, such that* $\Delta^\uparrow(Z) = \Delta'^\uparrow(Z)$ *and* $\langle C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]|_{\boldsymbol{x}}\rangle \approx_r \langle C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{y})]|_{\boldsymbol{x}}\rangle$.

Informally, the stop condition holds when in an ancestor node in the tableau an instance of the abstract proposition Z has been unfolded in an equivalent context, and between such ancestor and the current node, Z has not been reintroduced. This intuitively means that the two Z's refer to the same fixpoint instance. In this case we can safely avoid to continue along this path. Instead, when the stop condition fails, it makes sense to further unfold the fixpoint since the current context is still "different enough" from those previously encountered. Note that the equivalence of contexts is checked after renaming the variables to those associated with Z in the fixpoint formula quantifying the proposition.

As an example, consider the formula $\langle\!|\mathsf{b}\, x|\!\rangle\, \nu X(x).\, \mu Y(x).(\langle\!|x < \mathsf{b}\, y|\!\rangle X(y) \vee \langle\!|\_\, z|\!\rangle Y(x))$ from Subsection 4.3.3, which was claimed to be satisfied by $\mathcal{E}_3$ in Figure 3.1c. Write it as $\langle\!|\mathsf{b}\, x|\!\rangle\varphi$, where $\varphi = \nu X(x).\psi$ and $\psi = \mu Y(x).(\langle\!|x < \mathsf{b}\, y|\!\rangle X(y) \vee \langle\!|\_\, z|\!\rangle Y(x))$. A tableau for the sequent $\emptyset, \rho, \emptyset \models \langle\!|\mathsf{b}\, x|\!\rangle\varphi$ is given in Figure 5.2. No rule can be applied to the bottom sequent $\{\mathsf{b}^0, \mathsf{b}^1\}, \rho[x \mapsto \mathsf{b}^0, y \mapsto \mathsf{b}^1], \{X(x) = \varphi, Y(x) = \psi\} \models X(y)$ because the stop condition holds, hence the side condition of the unfolding rule is not satisfied. In fact, there is an ancestor, namely $\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0], \{X(x) = \varphi\} \models X(x)$, having the same introduction node for the abstract proposition X and such that $\langle\{\mathsf{b}^0, \mathsf{b}^1\}, \rho[x \mapsto \mathsf{b}^1]|_x\rangle \approx_r \langle\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0]|_x\rangle$.

In [Cle90, SW91, CS01], the finiteness of the model is an essential ingredient that concurs to the finiteness of the tableaux. In our case, as already mentioned, the PES model is

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{\emptyset, \rho, \emptyset \models \langle\!|\mathsf{b}\, x|\!\rangle \varphi}{\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0], \emptyset \models \varphi}\,(\Diamond)
}{\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0], \{X(x) = \varphi\} \models X(x)}\,(\mathsf{Int})
}{\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0], \{X(x) = \varphi\} \models \psi}\,(\mathsf{Unf}_\nu)
}{\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0], \{X(x) = \varphi, Y(x) = \psi\} \models Y(x)}\,(\mathsf{Int})
}{\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0], \{X(x) = \varphi, Y(x) = \psi\} \models \langle\!|x < \mathsf{b}\, y|\!\rangle X(y) \vee \langle\!|\_\, z|\!\rangle Y(x)}\,(\mathsf{Unf}_\mu)
}{\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0], \{X(x) = \varphi, Y(x) = \psi\} \models \langle\!|x < \mathsf{b}\, y|\!\rangle X(y)}\,(\vee_{\mathsf{L}})
}{\{\mathsf{b}^0, \mathsf{b}^1\}, \rho[x \mapsto \mathsf{b}^0, y \mapsto \mathsf{b}^1], \{X(x) = \varphi, Y(x) = \psi\} \models X(y)}\,(\Diamond)
$$

Figure 5.2: A successful tableau for $\langle\!|\mathsf{b}\, x|\!\rangle \nu X(x).\mu Y(x).(\langle\!|x < \mathsf{b}\, y|\!\rangle X(y) \vee \langle\!|\_\, z|\!\rangle Y(x))$ in $\mathcal{E}_3$.

commonly infinite, even for finite state systems. When working with regular PESs, thanks to the fact that they are boundedly branching, only a finite part of the model is used at every step of the tableau construction. However, the stop condition is still not guaranteed to eventually hold because regular PESs, despite having finitely many equivalence classes of residuals, could still have possibly infinitely many pointed residuals up to isomorphism. For this reason we constrain the model-checking procedure to a more specific class of regular PESs defined below.

Given a configuration $C \in \mathcal{C}(\mathcal{E})$ and a subset of events $X \subseteq C$, we denote by $\mathcal{E}[C] \cup X$ the PES obtained from $\mathcal{E}[C]$ by adding the events in X with the causal dependencies they had in the original PES $\mathcal{E}$.

Definition 5.7 (strong regularity)**.** *A PES $\mathcal{E}$ is called* strongly regular *when it is boundedly branching and for each $k \in \mathbb{N}$ the set $\{\mathcal{E}[C] \cup \{e_1, \ldots, e_k\} \mid C \in \mathcal{C}(\mathcal{E}) \ \wedge \ e_1, \ldots, e_k \in C\}$ is finite up to isomorphism of PESs.*

Strong regularity is obtained from the notion of regularity, by replacing residuals with residuals extended with a bounded number of events from the past. Roughly, the requirement is that the PES has a finite number of extended residuals over which the computation cycles. Clearly, every strongly regular PES is regular since the property in Definition 5.7 must hold, in particular, for $k = 0$. We will later show in Section 6.3 that the PESs associated with finite safe Petri nets are not only regular (Theorem 3.2) but even strongly regular (Corollary 6.1). The class of strongly regular PESs may not be strictly included in the regular ones, the two classes could still be the same. The problem is left open. Answering this question is very hard, arguably harder than disproving the second half of Thiagarajan's conjecture. Indeed, any example of regular but not strongly regular PES would be a counterexample to the conjecture.

An example of strongly regular PES is $\mathcal{E}_3$ in Figure 3.1c. Note, e.g., that it has six

(equivalence classes of) residuals extended with a single event from the past $\mathcal{E}_3[\{\mathsf{c}\}] \cup \{\mathsf{c}\}$, $\mathcal{E}_3[\{\mathsf{b}^0\}] \cup \{\mathsf{b}^0\}$, $\mathcal{E}_3[\{\mathsf{c}, \mathsf{b}^0\}] \cup \{\mathsf{b}^0\}$, $\mathcal{E}_3[\{\mathsf{c}, \mathsf{a}^0\}] \cup \{\mathsf{c}\}$, $\mathcal{E}_3[\{\mathsf{c}, \mathsf{a}^0\}] \cup \{\mathsf{a}^0\}$ and $\mathcal{E}_3[\{\mathsf{c}, \mathsf{b}^0, \mathsf{a}^1\}] \cup \{\mathsf{b}^0\}$. As we already observed in Subsection 3.2.1, $\mathcal{E}_3$ is induced by the safe Petri net $\mathcal{N}$ in Figure 3.4a. Then, the easiest way to show that $\mathcal{E}_3$ is strongly regular is to use the mentioned result on finite safe Petri nets (Corollary 6.1).

The relation between residuals of pointed configurations and residuals extended with events from the past is quite trivial: the residual of a pointed configuration $\langle C, \zeta \rangle$ can be seen as the residual $\mathcal{E}[C]$ extended with the events in $cod(\zeta)$. Actually, intuitively, pointed residuals are more concrete. For instance, consider the configuration $C = \{\mathsf{c}, \mathsf{b}^0\}$ of $\mathcal{E}_3$ and let $\zeta_1, \zeta_2 : \{x, y\} \to C$ be defined by $\zeta_1(x) = \zeta_2(y) = \mathsf{b}^0$ and $\zeta_1(y) = \zeta_2(x) = \mathsf{c}$. The residuals of the $\{x, y\}$-pointed configurations $\langle C, \zeta_1 \rangle$ and $\langle C, \zeta_2 \rangle$ are not isomorphic since $\zeta_1(x)$ causes all events enabled at C, while $\zeta_2(x)$ causes only one of them. On the other hand, since $cod(\zeta_1) = cod(\zeta_2) = C$, the corresponding extended residuals are the same $\mathcal{E}_3[C] \cup C$, and thus isomorphic. Nevertheless, it can be shown that, for strongly regular PESs, the number of equivalence classes of pointed configurations with respect to $\approx_r$ is finite. The converse holds when the PES is boundedly branching, providing an alternative characterisation of strong regularity.

Lemma 5.4 (strong regularity and pointed configurations)**.** *A PES $\mathcal{E}$ is strongly regular if and only if it is boundedly branching and for any fixed finite set V, the equivalence $\approx_r$ is of finite index over V-pointed configurations of $\mathcal{E}$.*

Proof. ($\Rightarrow$) Assume that $\mathcal{E}$ is strongly regular. Then by definition it is boundedly branching. Let V be a finite set and let $\mathcal{C}_V = \{\langle C, \zeta \rangle \mid C \in \mathcal{C}(\mathcal{E}) \ \wedge \ \zeta : V \to C\}$ be the set of V-pointed configurations.

By strong regularity, the set $\{\mathcal{E}[C] \cup \zeta(V) \mid C \in \mathcal{C}(\mathcal{E}) \ \wedge \ \zeta : V \to C\}$ is finite up to isomorphism of PESs. This does not immediately imply that $\mathcal{C}_V$ is finite up to $\approx_r$. In fact, given two V-pointed configurations $\langle C, \zeta \rangle$, $\langle C', \zeta' \rangle$, an isomorphism of PES $\iota : \mathcal{E}[C] \cup \zeta(V) \to \mathcal{E}[C'] \cup \zeta'(V)$ is not necessarily an isomorphism of the corresponding pointed residuals. This is certainly the case if additionally it holds $\iota(\zeta(x)) = \zeta'(x)$ for all $x \in V$.

Assume, by contradiction, that $\mathcal{C}_V$ is not finite up to $\approx_r$. Then we can find a sequence of V-pointed configurations $\langle C_i, \zeta^i \rangle$, $i \in \mathbb{N}$ such that the PESs $\mathcal{E}[C_i] \cup \zeta^i(V)$ are all isomorphic, while $\langle C_i, \zeta^i \rangle$ are pairwise non-equivalent with respect to $\approx_r$. Let $\iota_i : \mathcal{E}[C_i] \cup \zeta^i(V) \to \mathcal{E}[C_{i+1}] \cup \zeta^{i+1}(V)$ be PES isomorphisms for all $i \in \mathbb{N}$ and denote by $\iota_{i,j} : \mathcal{E}[C_i] \cup \zeta^i(V) \to \mathcal{E}[C_j] \cup \zeta^j(V)$ the isomorphism resulting as the composition $\iota_{j-1} \circ \ldots \circ \iota_{i+1} \circ \iota_i$.

The key observations are the following. Define the causal depth of an event as $depth(e) = \max\{depth(e') \mid e' < e\}$. Then

1. the events in $\zeta^i(V)$ have causal depth bounded by $|V|$ in the PES $\mathcal{E}[C_i] \cup \zeta^i(V)$;
2. the PESs $\mathcal{E}[C_i] \cup \zeta^i(V)$ have a finite number of events of causal depth bounded by $|V|$, since the PES $\mathcal{E}$ is strongly regular (hence b-bounded for some $b \in \mathbb{N}$);

3. each isomorphism of PESs ι_i preserves the causal depth of events (i.e., $depth(\iota_i(e)) = depth(e)$).

By (1)-(3) above we deduce that there are $i, j \in \mathbb{N}$, $i \leq j$ such that for all $x \in V$ we have $\iota_{1,i}(\zeta^1(x)) = \zeta^i(x)$ and $\iota_{1,j}(\zeta^1(x)) = \zeta^j(x)$. This implies that $\iota_{i,j}(\zeta^i(x)) = \zeta^j(x)$ for all $x \in V$.

Therefore it is immediate to see that $\iota_{i,j}|_{\mathcal{E}[C_i]} : \mathcal{E}[C_i] \to \mathcal{E}[C_j]$ is an isomorphism of the residuals of the pointed configurations $\langle C_i, \zeta^i \rangle$ and $\langle C_j, \zeta^j \rangle$, contradicting the fact that the pointed configurations in the sequence had all non-isomorphic residuals.

($\Leftarrow$) Assume that $\mathcal{E}$ is boundedly branching and for all fixed finite set V, the equivalence $\approx_r$ is of finite index over V-pointed configurations of $\mathcal{E}$. We have to prove that $\mathcal{E}$ is strongly regular. Let k be a fixed integer and let $V = \{v_1, \ldots, v_k\}$ be a set of cardinality k. Each "extended" residual $\mathcal{E}[C] \cup \{e_1, \ldots, e_k\}$, with $e_1, \ldots, e_k \in C$ can be trivially seen as the residual of a V-pointed configuration $\langle C, \zeta \rangle$ with $\zeta(v_i) = e_i$ for $i \in \{1, \ldots, k\}$.

Again, from the fact that $\approx_r$ is of finite index over V-pointed configurations we cannot immediately conclude. In fact, consider two extended residuals $\mathcal{E}[C] \cup \{e_1, \ldots, e_k\}$ and $\mathcal{E}[C'] \cup \{e'_1, \ldots, e'_k\}$ such that the corresponding V-pointed configurations $\langle C, \zeta \rangle$ and $\langle C', \zeta' \rangle$ are in the same class, i.e., $\langle C, \zeta \rangle \approx_r \langle C', \zeta' \rangle$. Let $\iota : \mathcal{E}[C] \to \mathcal{E}[C']$ be the corresponding isomorphism such that for all $x \in V$, $e \in \mathcal{E}[C]$ we have $\zeta(x) \leq e$ iff $\zeta'(x) \leq \iota(e)$. Observe that it is not necessarily the case that

$$\iota[\zeta(v_1) \mapsto \zeta'(v_1), \ldots, \zeta(v_k) \mapsto \zeta'(v_k)] : \mathcal{E}[C] \cup \{e_1, \ldots, e_k\} \to \mathcal{E}[C'] \cup \{e'_1, \ldots, e'_k\}$$

is an isomorphism of PESs, since the mapping $\zeta(v_1) \mapsto \zeta'(v_1), \ldots, \zeta(v_k) \mapsto \zeta'(v_k)$ does not necessarily respect causal dependencies nor labels. However, since, up to isomorphism, there are only finitely many partial orders with k elements, and similarly only finitely many possible labelling, we conclude that each $\approx_r$-class of V-pointed configurations splits in a finite number of classes of isomorphic extended residuals and thus we conclude. □

From the result above it follows that, in the construction of a tableau over a strongly regular PES, after going sufficiently in depth, the PES starts "repeating" cyclically the same structure even if we take into account the events pointed by free variables. This will allow to show that the stop condition eventually holds. Later in Section 5.3, this will be used to prove that the satisfaction of a formula can be established by checking only a finite part of the PES.

5.3 Soundness and completeness of the tableau system

In this section we show that the truth of a formula of $\mathcal{L}_{hp}$ over a strongly regular event structure can be reduced to the existence of a suitably defined successful tableau. First, we formalise what is a tableau for a closed formula and a PES model.

Definition 5.8 (tableau for a closed formula)**.** *Let $\mathcal{E}$ be a PES. Given a closed formula φ in $\mathcal{L}_{hp}$, a* tableau for φ *is a tableau built applying the tableau rules for $\mathcal{L}_{hp}$ starting from a sequent $\emptyset, \rho, \emptyset \models^{\mathcal{E}} \varphi$, where $\rho \in Env_{\mathcal{E}}$.*

Note that the initial environment ρ can be chosen arbitrarily: it is irrelevant since the formula has no free event variables.

A *maximal tableau* is a tableau where all leaves are labelled by sequents to which no further rule applies. A maximal tableau is considered successful when it satisfies the condition given below.

Definition 5.9 (successful tableau)**.** *A* successful tableau *for $\mathcal{L}_{hp}$ is a finite maximal tableau where every leaf is labelled by a sequent $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ such that one of the following holds:*

1. $\varphi = \mathsf{T}$
2. $\varphi = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \psi$
3. $\varphi = Z(\boldsymbol{y})$ *and* $\Delta(Z(\boldsymbol{x})) = \nu Z(\boldsymbol{x}).\psi$.

An example of successful tableau can be found in Figure 5.2.

We will prove that in a successful tableau all leaves are labelled by true sequents, a fact that, along with backwards soundness of the rules (Lemma 5.2), will guarantee the truth of the sequent labelling the root.

Observe that the choice of the rule to be applied at a step of the construction of a tableau is nondeterministic in the case of diamond modalities $\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \varphi$ and disjunctions $\varphi \vee \psi$. This means that there can be various maximal tableaux for the same sequent. However, when we work on strongly regular PESs, the fact that they are boundedly branching ensures that at each step the number of possible nondeterministic choices is finite and bounded. This plays later a role for deducing that there can be only a finite number of maximal tableaux for a sequent.

We first focus on the finiteness issue and then move on to the soundness and completeness of the technique.

5.3.1 Finiteness

In order to have a terminating procedure we need that, given a formula and a model, tableaux are finite and finite in numbers. We will first prove that all tableaux for a sequent $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ are finite. Then we will show that the fact that there are only finitely many tableaux follows as a direct consequence of that. A first basic observation is that an infinite tableau for a sequent $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ necessarily includes a path where the same proposition is unfolded infinitely many times without being reintroduced. The proof relies on some properties of the order $\sqsubset_d^*$ over propositions (see Definition 4.12).

Lemma 5.5 (fixpoint introduction)**.** *Let $\mathcal{E}$ be a PES and let τ be a tableau for a closed formula φ in $\mathcal{L}_{hp}$. Let n be any node in the tableau labelled by $C, \rho, \Delta \models X(\boldsymbol{x})$ for some $X(\boldsymbol{x}) \in \mathcal{X}$.*

1. *If n has a descendant n' labelled by $C', \rho', \Delta' \models Y(\boldsymbol{y})$, for some $Y(\boldsymbol{y}) \in \mathcal{X}$, and Y is not introduced between n and n', then $X \sqsubset_d^* Y$.*

2. *If n has a descendant n' that introduces X, then there is a node n'' between n and n' with consequent $Y(\boldsymbol{y})$ such that $X \sqsubset_d Y$ (hence $X \sqsubset_d^* Y$ and $X \neq Y$).*

Proof. 1. We prove the property by induction on the number of sequents between n and n' labelled by a proposition. Let n_i be such nodes, labelled by $C_i, \rho_i, \Delta_i \models Z_i(\boldsymbol{z}_i)$ with $i \in \{1, \ldots, k\}$. Moreover, let $\eta X(\boldsymbol{x}').\psi$ be the definition of X in the tableau.

($k = 0$) The only rule applicable to node n is (Unf_η), hence the successor of n will have ψ as consequent. Since between n and n' there are no propositions and thus no other applications of (Unf_η) and Y is not introduced, necessarily $Y \in \mathit{fp}(\psi)$. Hence either $Y = X$ or $X \sqsubset_d Y$. In any case, $X \sqsubset_d^* Y$.

($k > 0$) Consider node n_1 labelled by the sequent $C_1, \rho_1, \Delta_1 \models Z_1(\boldsymbol{z}_1)$. By inductive hypothesis

$$Z_1 \sqsubset_d^* Y \tag{5.4}$$

Since between n and n_1 there are no propositions and thus no applications of the (Unf_η) rule, there are two possibilities: either $Z_1 \in \mathit{fp}(\psi)$ or Z_1 is introduced between n and n_1. In the first case, by inductive hypothesis $X \sqsubset_d^* Z_1$ and thus, recalling (5.4) we conclude $X \sqsubset_d^* Y$. In the second case, if φ_1 is the definition of Z_1 in the tableau, then φ_1 is a subformula of ψ, a fact that along with (5.4), again, implies $X \sqsubset_d^* Y$.

2. Let $\eta X(\boldsymbol{x}').\psi$ be the definition of X in the tableau. The only rule applicable to node n is (Unf_η), hence the successor of n will have ψ as consequent. Since all rules except (Unf_η) reduce the size of the formula, there must be between n and n' an occurrence of the unfolding rule applied to some $Y \in \mathit{fp}(\eta X(\boldsymbol{x}').\psi)$. By definition $X \sqsubset_d Y$ as desired (hence $X \sqsubset_d^* Y$ and, clearly, $X \neq Y$ since $\sqsubset_d$ is irreflexive).

□

For the next step we recall a classical result by König [Kön27] which applies to tableaux since they are trees, hence connected graphs.

Lemma 5.6 (König)**.** *Let G be an infinite graph. If G is connected, i.e., there is a path between every pair of vertices, and every vertex of G is adjacent to only finitely many other vertices, then G contains an infinite path with no repeated vertices.*

We already observed that the fact that strongly regular PESs are boundedly branching implies that also the constructed tableaux are finitely branching. Then, by Lemma 5.6, an infinite tableau necessarily includes an infinite path. Using Lemma 5.5 it is not difficult to show that such path includes infinitely many repetitions of the same abstract proposition without introductions, that is, we get the following property.

Lemma 5.7 (infinite occurrences of propositions in tableaux)**.** *Let $\mathcal{E}$ be a finitely branching PES and φ be a closed formula in $\mathcal{L}_{hp}$. An infinite tableau for φ contains an infinite path where some abstract proposition Z occurs (and thus is unfolded) infinitely often without being introduced.*

Proof. Let $\mathcal{E}$ be a finitely branching PES and let τ be a an infinite tableau for φ. Since the transition system of the PES is finitely branching, also the tableau τ is finitely branching. Hence, by Lemma 5.6, it include an infinite path p. Since all tableau rules (Table 5.1) apart from (Unf_η) reduce the size of the consequent, path p must include infinitely many application of (Unf_η) and thus infinitely many sequents having a proposition as consequent.

Since the only abstract propositions that can appear are those in φ, which are finitely many, there are abstract propositions that occur infinitely often (possibly with different event variables).

Let P be the set of such abstract propositions and consider the suffix p' of p, where only abstract propositions in P occur.

Now, let $X \in P$ be maximal with respect to $\sqsubset_d^*$ and let p'' be a suffix of p' starting with an occurrence of X. Necessarily X is never introduced in p'', otherwise, by Lemma 5.5(2), there would be a node n' in p'' with consequent $Y(\boldsymbol{y})$ such that $X \sqsubset_d^* Y$ and $X \neq Y$, contradicting the maximality of X. □

We can finally deduce the finiteness of the tableaux for a sequent that in turn implies that the number of tableaux is finite. The proof is based on the two features of strongly regular PESs, that is, the model is finitely (boundedly) branching and the equivalence $\approx_r$ is of finite index over configurations pointed by the free variables. This fact is essential for the termination of the model-checking procedure. Roughly, the proof proceeds as follows. An infinite tableau, by Lemma 5.7, would contain an abstract proposition occurring infinitely often in a path, without being reintroduced. By using Lemma 5.4 we would deduce that the proposition would occur infinitely often within contexts with isomorphic pointed residuals. This would lead to a contradiction, since at the first repetition the stop condition (see Definition 5.6) would have required to terminate the branch.

Theorem 5.1 (tableaux finiteness)**.** *Given a strongly regular PES $\mathcal{E}$ and a closed formula φ in $\mathcal{L}_{hp}$, every tableau for a sequent $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ is finite. Hence the number of tableaux for $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ is finite.*

Proof. The proof proceeds by contradiction. Suppose that there is an infinite tableau τ for the sequent $C, \rho, \Delta \models^{\mathcal{E}} \varphi$. By Lemma 5.7, in τ there is an infinite path p where a

proposition Z occurs infinitely many times without being introduced. Let $Z(\boldsymbol{x}) = \eta Z(\boldsymbol{x}).\,\psi$ be the definition of Z in the tableau.

By Lemma 5.4 the set of $\boldsymbol{x}$-pointed configurations of $\mathcal{E}$ is finite up to $\approx_r$. Since the proposition Z is unfolded infinitely many times along p without being introduced, there are infinitely many sequents $C', \rho', \Delta' \models^{\mathcal{E}} Z(\boldsymbol{x}')$ for which $Z^{\uparrow}(\Delta')$ is the same node. Hence the stop condition γ is necessarily satisfied at some point of the path, contradicting its infiniteness.

We next prove that also the number of tableaux is finite. Consider a tree where nodes are tableaux rooted $C, \rho, \Delta \models^{\mathcal{E}} \varphi$ and where the successors of each tableau τ are the tableaux obtained by extending τ with the application of a rule. Since $\mathcal{E}$ is strongly regular, the tree is finitely branching. If it were infinite, by Lemma 5.6 there would be an infinite path and thus there would be an infinite sequence of tableaux $(\tau_i)_{i\in\omega}$, such that τ_{i+1} extends τ_i. This in turn implies the existence of an infinite tableau, which contradicts the first part. □

5.3.2 Soundness and completeness

We can now prove the soundness and completeness of the tableau system. For this we use the possibility of reducing the satisfaction of a formula to the satisfaction of a finite approximant. While on finite models this is immediate, on event structures where the space of configurations is infinite, this does not work, in general. In fact, due to alternation, the semantic function associated with a formula might be non-continuous (in the sense of, e.g., [Fon08, CFVZ14]), hence it is not guaranteed that fixpoints will be reached in at most ω steps. However, here, for strongly regular PESs, the result can be obtained by exploiting the finiteness of pointed configurations up to $\approx_r$.

The idea of computing fixpoints as approximants arises directly from Kleene's theorem (Theorem 2.2). Indeed, the theorem tells us that by iterating enough times the semantic function associated with a fixpoint formula starting from either the bottom or the top element, depending on the type of fixpoint, we would eventually obtain its denotation. Furthermore, Lemma 4.1(2) suggests that we can actually perform this iteration syntactically instead of semantically. So, here we formally define how a fixpoint formula is iterated, that is, unfolded, in order to obtain the corresponding approximant.

Definition 5.10 (finite approximant)**.** *Let $\varphi = \nu Z(\boldsymbol{x}).\psi$ be a fixpoint formula. The i-th* approximant *φ^i, for $i \in \mathbb{N}$, is inductively defined by $\varphi^0 = \mathsf{T}$ and $\varphi^{i+1} = \psi[Z(\boldsymbol{x}) \coloneqq \varphi^i]$. Dually, if $\varphi = \mu Z(\boldsymbol{x}).\psi$, we define $\varphi^0 = \mathsf{F}$ and $\varphi^{i+1} = \psi[Z(\boldsymbol{x}) \coloneqq \varphi^i]$.*

We next observe that, as anticipated, despite the fact that the state space of configurations is infinite, for strongly regular PESs all formulae reach the fixpoint after a finite number of steps.

Lemma 5.8 (finite approximants properties)**.** *Let $\mathcal{E}$ be a strongly regular PES, let $\pi \in PEnv_{\mathcal{E}}$ a saturated proposition environment and let $\varphi = \eta Z(\boldsymbol{x}).\,\psi$ be a fixpoint formula.*

Then there exists $i \in \mathbb{N}$ such that $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi} = \{\!|\varphi^i|\!\}^{\mathcal{E}}_{\pi}$. Hence for any configuration $C \in \mathcal{C}(\mathcal{E})$ and environment $\rho \in Env_{\mathcal{E}}$:

1. *if $\varphi = \nu Z(\boldsymbol{x}).\psi$ and $(C,\rho) \notin \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$, then $(C,\rho) \in \{\!|\varphi^n|\!\}^{\mathcal{E}}_{\pi} \smallsetminus \{\!|\varphi^{n+1}|\!\}^{\mathcal{E}}_{\pi}$ for some $n \leq i$;*
2. *if $\varphi = \mu Z(\boldsymbol{x}).\psi$ and $(C,\rho) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$, then $(C,\rho) \in \{\!|\varphi^{n+1}|\!\}^{\mathcal{E}}_{\pi} \smallsetminus \{\!|\varphi^{n}|\!\}^{\mathcal{E}}_{\pi}$ for some $n \leq i$.*

Proof. Let us first focus on the case $\varphi = \nu Z(\boldsymbol{x}).\psi$. Clearly, for all $i \in \mathbb{N}$, $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi} \subseteq \{\!|\varphi^i|\!\}^{\mathcal{E}}_{\pi}$. Assume by contradiction that the inclusion is always strict, i.e., $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi} \subset \{\!|\varphi^i|\!\}^{\mathcal{E}}_{\pi}$ for all $i \in \mathbb{N}$. This implies that $\{\!|\varphi^{i+1}|\!\}^{\mathcal{E}}_{\pi} \subset \{\!|\varphi^i|\!\}^{\mathcal{E}}_{\pi}$ for all $i \in \mathbb{N}$.

Therefore there is an infinite sequence of pairs $(C_i,\rho_i) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$, for $i \in \mathbb{N}$, such that $(C_i,\rho_i) \in \{\!|\varphi^i|\!\}^{\mathcal{E}}_{\pi} \smallsetminus \{\!|\varphi^{i+1}|\!\}^{\mathcal{E}}_{\pi}$. We deduce that for each i we have $(C_i,\rho_i) \in \{\!|\varphi^i|\!\}^{\mathcal{E}}_{\pi}$ and $(C_j,\rho_j) \notin \{\!|\varphi^i|\!\}^{\mathcal{E}}_{\pi}$ for all $j < i$. Then, by Lemma 5.3, we would have that $\langle C_j, \rho_j|_{fv(\varphi^i)}\rangle \not\approx_r \langle C_i, \rho_i|_{fv(\varphi^i)}\rangle$ for all $i \in \mathbb{N}$ and $j < i$. This would mean that there are infinitely many different equivalence classes of $fv(\varphi^i)$-pointed configurations with respect to $\approx_r$. Since $|fv(\varphi^i)| \leq |fv(\varphi)| = |\boldsymbol{x}|$ and $\mathcal{E}$ is strongly regular this fact contradicts Lemma 5.4. Therefore there must exist $i \in \mathbb{N}$ such that $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi} = \{\!|\varphi^i|\!\}^{\mathcal{E}}_{\pi}$. Point (1) then immediately follows.

If $\varphi = \mu Z(\boldsymbol{x}).\psi$, a dual reasoning proves that there is $i \in \mathbb{N}$ such that $\{\!|\varphi^i|\!\}^{\mathcal{E}}_{\pi} = \{\!|\varphi^{i+1}|\!\}^{\mathcal{E}}_{\pi}$ and thus $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi} = \{\!|\varphi^i|\!\}^{\mathcal{E}}_{\pi}$ and, again, point (2) immediately follows. □

We can now show that the model-checking problem reduces to the construction of a successful tableau. We prove separately soundness and completeness, by resorting to some variations of the tableau system in Table 5.1 that we call ν-pseudo-tableau and μ-pseudo-tableau systems, respectively.

As a preliminary step we need to observe how the instantiation of a formula interacts with the approximants.

Lemma 5.9 (grounding, substitutions and approximants)**.** *Let $C, \rho, \Delta \models \varphi$ be a well-formed sequent where $\varphi = \eta Z(\boldsymbol{x}).\,\psi$ is a fixpoint formula. Then*

1. *$(\varphi)_\Delta$ is a fixpoint formula*
2. *$(\varphi)_\Delta[\boldsymbol{y}/\boldsymbol{x}] = (\varphi[\boldsymbol{y}/\boldsymbol{x}])_\Delta$*
3. *for any $n \in \mathbb{N}$, it holds $((\varphi)_\Delta)^n = (\varphi^n)_\Delta$.*

Proof. Concerning point (1), reasoning as in the proof of Lemma 5.1, it is easy to see that $(\varphi)_\Delta$ is a fixpoint formula. In fact, let $\{Y_1,\ldots,Y_n\} = fp(\psi) \smallsetminus \{Z\}$ and let

$$\psi' = \psi[Y_1(\boldsymbol{y}_1) \coloneqq (Y_1(\boldsymbol{y}_1))_\Delta] \ldots [Y_n(\boldsymbol{y}_n) \coloneqq (Y_n(\boldsymbol{y}_n))_\Delta].$$

Then, observe that

$$(\varphi)_\Delta = (\eta Z(\boldsymbol{x}).\psi)_\Delta = \eta Z(\boldsymbol{x}).\psi'.$$

We now prove point (2) $(\varphi[\boldsymbol{y}/\boldsymbol{x}])_\Delta = (\varphi)_\Delta[\boldsymbol{y}/\boldsymbol{x}]$. Using the fact that $\eta Z(\boldsymbol{x}).\psi = (\eta Z(\boldsymbol{x}).\psi)(\boldsymbol{x})$, this is also almost immediate. In fact:

$$(\varphi[\boldsymbol{y}/\boldsymbol{x}])_\Delta = ((\eta Z(\boldsymbol{x}).\psi)(\boldsymbol{x})[\boldsymbol{y}/\boldsymbol{x}])_\Delta = ((\eta Z(\boldsymbol{x}).\psi)(\boldsymbol{y}))_\Delta =$$
$$= (\eta Z(\boldsymbol{x}).\psi')(\boldsymbol{y}) = (\eta Z(\boldsymbol{x}).\psi')(\boldsymbol{x})[\boldsymbol{y}/\boldsymbol{x}] = (\varphi)_\Delta[\boldsymbol{y}/\boldsymbol{x}].$$

Finally, we prove point (3), i.e., for any $n \in \mathbb{N}$, $(\varphi^n)_\Delta = ((\varphi)_\Delta)^n$, by induction on n. We focus on the case $\eta = \nu$ (the proof is completely analogous when $\eta = \mu$).

($n = 0$) Immediate, since $(\varphi^0)_\Delta = (\mathsf{T})_\Delta = \mathsf{T} = ((\varphi)_\Delta)^0$.

($n > 0$) Observe that

$$(\varphi^n)_\Delta = (\psi[Z(\boldsymbol{x}) \coloneqq \varphi^{n-1}])_\Delta = \psi'[Z(\boldsymbol{x}) \coloneqq (\varphi^{n-1})_\Delta],$$

where the last equality holds since $fp(\varphi^{n-1}) \subseteq fp(\psi[Z(\boldsymbol{x}) \coloneqq \varphi^{n-1}]) \subseteq fp(\psi) \setminus \{Z\}$, and

$$((\varphi)_\Delta)^n = (\eta Z(\boldsymbol{x}).\psi')^n = \psi'[Z(\boldsymbol{x}) \coloneqq ((\varphi)_\Delta)^{n-1}].$$

Thus we can conclude since, by inductive hypothesis, we know that $(\varphi^{n-1})_\Delta = ((\varphi)_\Delta)^{n-1}$. □

In words, the results above show that the instantiations of the approximants of a fixpoint formula φ coincide with the approximants of the instantiation of φ. For this reason, in the sequel, we will abuse the notation and write $(\varphi)^n_\Delta$ for both $((\varphi)_\Delta)^n = (\varphi^n)_\Delta$.

We start by showing the soundness of the tableau system. The proof will rely on the notion of ν-pseudo-tableau introduced below.

Definition 5.11 (ν-pseudo-tableaux)**.** *The ν-pseudo-tableau system is obtained from that in Table 5.1, by working with sequents $C, \rho, \Delta \models \varphi$ where the definition list Δ contains definitions $Z(\boldsymbol{x}) = (\nu Z(\boldsymbol{x}).\psi)^n$ and inserting the new unfolding rule below:*

$$(\mathsf{Unf}^{\mathsf{a}}_\nu)\ \frac{C, \rho, \Delta \ \models\ Z(\boldsymbol{z})}{C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})], \Delta \ \models\ \psi} \qquad \neg\gamma \text{ and } \Delta(Z(\boldsymbol{x})) = (\nu Z(\boldsymbol{x}).\psi)^n$$

The notion of successful ν-pseudo-tableau is as in Definition 5.9, but we additionally allow a leaf to be labelled by a sequent $C, \rho, \Delta \models Z(\boldsymbol{z})$ where $\Delta(Z(\boldsymbol{x})) = (\nu Z(\boldsymbol{x}).\psi)^n$ and $(C, \rho) \in \{\!|(Z(\boldsymbol{z}))_\Delta|\!\}_\pi$ for $\pi \in PEnv_\mathcal{E}$.

It can be easily seen that also rule $(\mathsf{Unf}^{\mathsf{a}}_\nu)$ is backwards sound. Intuitively, the formula in the premise corresponds to the approximant $(\nu Z(\boldsymbol{x}).\psi)^n$, while the formula in the conclusion to the approximant $(\nu Z(\boldsymbol{x}).\psi)^{n+1}$. Thus if the conclusion is true, also the premise is, since we know that $\{\!|(\nu Z(\boldsymbol{x}).\psi)^{n+1}|\!\}^\mathcal{E}_\pi \subseteq \{\!|(\nu Z(\boldsymbol{x}).\psi)^n|\!\}^\mathcal{E}_\pi$. More formally, assume that the sequent in the conclusion is true, i.e.,

$$(C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]) \in \{\!|(\psi)_\Delta|\!\}^\mathcal{E}_\pi = \{\!|(\psi[Z(X) \coloneqq (\nu Z(\boldsymbol{x}).\psi)^n])_\Delta|\!\}^\mathcal{E}_\pi = \{\!|(\nu Z(\boldsymbol{x}).\psi)^{n+1}_\Delta|\!\}^\mathcal{E}_\pi$$

for some $\pi \in PEnv_\mathcal{E}$. Now observe that $(Z(\boldsymbol{x}))_\Delta = (\nu Z(\boldsymbol{x}).\psi)^n_\Delta$. Hence

$$(C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]) \in \{\!|(\nu Z(\boldsymbol{x}).\psi)^{n+1}_{\Delta}|\!\}^{\mathcal{E}}_{\pi} \subseteq \{\!|(\nu Z(\boldsymbol{x}).\psi)^{n}_{\Delta}|\!\}^{\mathcal{E}}_{\pi} = \{\!|(Z(\boldsymbol{x}))_{\Delta}|\!\}^{\mathcal{E}}_{\pi} = \{\!|Z(\boldsymbol{z})|\!\}^{\mathcal{E}}_{\pi}[\boldsymbol{x}/\boldsymbol{z}].$$

Thus $(C, \rho) \in \{\!|Z(\boldsymbol{z})|\!\}^{\mathcal{E}}_{\pi}$ which means that the sequent in the premise is true.

Soundness will be an immediate consequence of the following technical result that implies that there cannot be successful ν-pseudo-tableaux with false leaves. The central result shows that a successful tableau with a false leaf can be "shortened" to a subtableau with the same feature.

Lemma 5.10 (shortening ν-pseudo-tableaux)**.** *Let $\mathcal{E}$ be a strongly regular PES and let τ be a successful ν-pseudo-tableau. If τ has a false leaf and for all false leaves $C, \rho, \Delta \models X(\boldsymbol{z})$, the node $\Delta^{\uparrow}(X)$ is in τ, then there exists a successful ν-pseudo-tableau τ', strictly smaller than τ, with a false leaf and where for all false leaves $C, \rho, \Delta \models X(\boldsymbol{z})$, the node $\Delta^{\uparrow}(X)$ is in τ'.*

Proof. Since τ is successful its leaves are labelled by sequents $C, \rho, \Delta \models \varphi$, where φ is T, $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \psi$ or a proposition $X(\boldsymbol{y})$ defined in τ as a greatest fixpoint or as some ν-approximant. If $\varphi = \mathsf{T}$ then clearly the sequent is true. If $\varphi = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \psi$, the definition requires that the set $\mathsf{Succ}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a} z}_{\mathcal{E}}(C, \rho)$ is empty, and thus, observing that $(\varphi)_{\Delta} = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, (\psi)_{\Delta}$ we deduce that the leaf is vacuously true. Also leaves labelled by a proposition, defined as a ν-approximant are true by definition of successful tableau.

Therefore the only possibly false leaves must have as consequent some proposition $X(\boldsymbol{y})$ defined as a greatest fixpoint $\nu X(\boldsymbol{x}).\psi$. Amongst such false leaves choose one, labelled $C, \rho, \Sigma \models X(\boldsymbol{y})$ say, such that there is no other false leaf labelled $C', \rho', \Sigma' \models X'(\boldsymbol{y}')$ with $\Sigma'^{\uparrow}(X')$ above $\Sigma^{\uparrow}(X)$ in τ.

Call w the node $\Sigma^{\uparrow}(X)$. By definition of tableau the node w is labelled by a sequent with consequent $(\nu X(\boldsymbol{x}).\psi)(\boldsymbol{y}')$. Let τ_w be the subtableau of τ rooted in the successor of w, labelled $\tilde{C}, \tilde{\rho}, \tilde{\Delta} \models X(\tilde{\boldsymbol{y}})$ say.

Consider the other leaves in τ_w labelled by false sequents involving the same introduction of the abstract proposition X, i.e., whose consequent is of the kind $C, \rho, \Sigma \models X(\boldsymbol{y}'')$ for suitable C, ρ, Σ, $\boldsymbol{y}''$, with $\Sigma^{\uparrow}(X) = w$. For each such leaf, the fact that the sequent is false means $(C, \rho) \notin \{\!|(X(\boldsymbol{y}''))_{\Delta}|\!\}^{\mathcal{E}}$ for $\pi \in PEnv_{\mathcal{E}}$. Observe that by Lemma 5.9(2)

$$(X(\boldsymbol{y}''))_{\Delta} = (\Delta(X(\boldsymbol{x}))[\boldsymbol{y}''/\boldsymbol{x}])_{\Delta} = (\nu X(\boldsymbol{x}).\psi[\boldsymbol{y}''/\boldsymbol{x}])_{\Delta} = (\nu X(\boldsymbol{x}).\psi)_{\Delta}[\boldsymbol{y}''/\boldsymbol{x}].$$

Hence $(C, \rho) \notin \{\!|(\nu X(\boldsymbol{x}).\psi)_{\Delta}[\boldsymbol{y}''/\boldsymbol{x}]|\!\}^{\mathcal{E}} = \{\!|(\nu X(\boldsymbol{x}).\psi)_{\Delta}|\!\}^{\mathcal{E}}[\boldsymbol{y}''/\boldsymbol{x}]$, by Lemma 4.1(1), and thus we have $(C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{y}'')]) \notin \{\!|(\nu X(\boldsymbol{x}).\psi)_{\Delta}|\!\}^{\mathcal{E}}$. In turn, by applying first Lemma 5.9(1) and then Lemma 5.8(1), this implies that there is $n \in \mathbb{N}$ such that $(C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{y}'')]) \in \{\!|(\nu X(\boldsymbol{x}).\psi)^{n}_{\Delta}|\!\}_{\pi} \smallsetminus \{\!|((\nu X(\boldsymbol{x}).\psi)^{n+1}_{\Delta}|\!\}_{\pi}$.

Take one of such leaves l such that the corresponding n is as small as possible. Let l be labelled with the sequent $C, \rho, \Sigma \models X(\boldsymbol{z})$. Since l is a leaf, the stop condition must be satisfied, i.e., there is an ancestor k of l labelled $C', \rho', \Sigma' \models X(\boldsymbol{z}')$ such that $\Sigma^{\uparrow}(X) = \Sigma'^{\uparrow}(X) = w$ (hence k in τ_w) and $\langle C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]|_{\boldsymbol{x}}\rangle \approx_r \langle C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{z}')]|_{\boldsymbol{x}}\rangle$. Furthermore,

since $(C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]) \notin \{\!|(\nu X(\boldsymbol{x}).\psi)^{n+1}_{\Delta}|\!\}_{\pi}$ and $fv((\nu X(\boldsymbol{x}).\psi)^{n+1}_{\Delta}) \subseteq \boldsymbol{x}$, by Lemma 5.3 we have

$$(C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{z}')]) \notin \{\!|\nu X(\boldsymbol{x}).(\psi)^{n+1}_{\Delta}|\!\}_{\pi}. \tag{5.5}$$

Now transform the tableau τ_w into a new tableau τ'_w by replacing each definition list Δ in a sequent of τ_w with Δ' defined as follows:

- if $\Delta^{\uparrow}(X) = w$, then $\Delta' = \Delta[X \mapsto (\nu X(\boldsymbol{x}).\psi)^n]$,
- otherwise $\Delta' = \Delta$.

Clearly τ'_w is a well-defined ν-pseudo-tableau. Moreover, τ'_w is successful. The only doubt could concern leaves that in τ_w were labelled $C', \rho', \Delta' \models X(\boldsymbol{y}')$ with $\Delta'^{\uparrow}(X) = w$ and thus in τ'_w become $C', \rho', \Delta'' \models X(\boldsymbol{y}')$ with $\Delta'' = \Delta'[X(\boldsymbol{x}) \mapsto (\nu X(\boldsymbol{x}).\psi)^n]$. However, these leaves are true in τ'_w. In fact, otherwise, we would have that $(C', \rho') \notin \{\!|(X(\boldsymbol{y}'))_{\Delta''}|\!\}_{\pi} = \{\!|(\Delta''(X(\boldsymbol{y}')))_{\Delta'}|\!\}_{\pi} = \{\!|(\nu X(\boldsymbol{x}).\psi)^n_{\Delta'}|\!\}_{\pi} = \{\!|(\nu X(\boldsymbol{x}).\psi)^n_{\Delta}|\!\}_{\pi}$, where the last equality follows from the fact that $X \notin fp((\nu X(\boldsymbol{x}).\psi)^n)$. Therefore $(C'', \rho''[\boldsymbol{x} \mapsto \rho''(\boldsymbol{y}')]) \notin \{\!|(\nu X(\boldsymbol{x})).\psi)^n|\!\}_{\pi}$, which means that there would be $m < n$ such that $(C'', \rho''[\boldsymbol{x} \mapsto \rho''(\boldsymbol{y}')]) \in \{\!|(\nu X(\boldsymbol{x}).\psi)^m_{\Delta}|\!\}_{\pi} \setminus \{\!|(\nu X(\boldsymbol{x}).\psi)^{m+1}_{\Delta}|\!\}_{\pi}$, contradicting the choice of n.

Additionally, after the transformation, the successor of the node k defined above is labelled $C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{z}')], \Sigma'' \models \psi$ where $\Sigma'' = \Sigma'[X \mapsto (\nu X(\boldsymbol{x}).\psi)^n]$. It is easy to see that $(\psi)_{\Delta''} = (\nu X(\boldsymbol{x}).\psi)^{n+1}_{\Delta''} = (\nu X(\boldsymbol{x}).\psi)^{n+1}_{\Delta}$ where the last equality is motivated by the fact that X does not occur free in $(\nu X(\boldsymbol{x}).\psi)^{n+1}$. Jointly with (5.5) this gives $(C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{z}')]) \notin \{\!|\nu X(\boldsymbol{x}).(\psi)^{n+1}_{\Delta}|\!\}_{\pi} = \{\!|(\psi)_{\Delta''}|\!\}_{\pi}$, namely the sequent $C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{z}')], \Sigma'' \models \psi$ is false, since $(C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{z}')]) \notin \{\!|\psi|\!\}_{\pi'}$. Therefore, by backwards soundness, some leaf of τ'_w must be labelled by a false sequent.

Consider any such false leaf and the corresponding sequent $C'', \rho'', \Delta' \models Y(\boldsymbol{y}')$. If the corresponding leaf of τ_w is labelled $C'', \rho'', \Delta \models Y(\boldsymbol{y}')$ we can have $\Delta' = \Delta[X \mapsto (\nu X(\boldsymbol{x}).\psi)^n]$ if $\Delta^{\uparrow}(X) = w$, $\Delta' = \Delta$ otherwise. We analyse the two cases separately:

- if $\Delta' = \Delta[X \mapsto (\nu X(\boldsymbol{x}).\psi)^n]$, then $Y \neq X$ because otherwise, by construction, the leaf would be true, as observed above. Moreover, $\Delta'^{\uparrow}(Y) = \Delta'^{\uparrow}(Y) = w'$ must be in τ'_w, i.e., a descendant of w, otherwise it would contradict the choice of w.
- if $\Delta' = \Delta$, and thus $\Delta'^{\uparrow}(Y) = \Delta'^{\uparrow}(Y) = w' \neq w$, then again w' must be after w, hence in τ'_w, otherwise it would contradict the choice of w.

Therefore we reduced to a strictly smaller subtableau τ'_w of τ, which is successful, with a false leaf and where all false leaves involve propositions introduced in τ'_w, as desired. □

The above lemma shows that if there were a successful ν-pseudo-tableau τ where all false leaves $C, \rho, \Delta \models X(\boldsymbol{z})$ are such that $\Delta^{\uparrow}(X)$ is in τ, then there would be a strict subtableau of τ with the same features. By iterating the property indefinitely one would obtain an infinite chain of strict sub-tableaux contradicting the fact that all tableaux are

finite. Therefore, this immediately implies that actually such a successful ν-pseudo-tableau cannot include false leaves. Since a successful tableau for a closed formula is a successful ν-pseudo-tableau with the above property, we conclude that it cannot have false leaves and thus, by backwards soundness, all nodes, including the root, must be true, i.e., we have the following.

Lemma 5.11 (soundness)**.** *Let $\mathcal{E}$ be a strongly regular PES and let φ be a closed formula of $\mathcal{L}_{hp}$. If φ has a successful ν-pseudo-tableau (hence in particular, if it has a successful tableau) then $\mathcal{E}$ satisfies φ.*

Proof. Assume that the sequent $C, \rho, \emptyset \models^{\mathcal{E}} \varphi$ has a successful tableau τ. Then τ is a successful ν-pseudo-tableau. For all leaves (and thus for all false leaves) $C, \rho, \Delta \models X(\boldsymbol{z})$, $\Delta^{\uparrow}(X)$ is in τ. Therefore, if it had a false leaf, by Lemma 5.10 we could continue building strictly smaller ν-pseudo-tableau with the same properties, contradicting the finiteness of τ. Hence all the sequents labelling the leaves of τ must be true, a fact that, by Lemma 5.2 implies that all the nodes of τ are true and thus, in particular, the sequent $C, \rho, \emptyset \models^{\mathcal{E}} \varphi$ labelling the root is true, as desired. □

For proving completeness we resort to a variation of basic tableaux dual to ν-pseudo-tableaux, referred to as μ-pseudo-tableaux.

Definition 5.12 (μ-pseudo-tableaux)**.** *The μ-pseudo-tableau system is obtained from that in Table 5.1, by working with sequents $C, \rho, \Delta \models \varphi$ where Δ contains definitions $Z(\boldsymbol{x}) = (\mu Z(\boldsymbol{x}).\psi)^n$ and replacing (Unf_μ) by the new unfolding rule below:*

$$(\mathsf{Unf}^{\mathsf{a}}_\mu)\ \frac{C, \rho, \Delta \models Z(\boldsymbol{z})}{C, \rho', \Delta' \models \psi} \qquad \neg\gamma \text{ and } \Delta(Z(\boldsymbol{x})) = \mu Z(\boldsymbol{x}).\psi \text{ or } (\mu Z(\boldsymbol{x}).\psi)^n$$

where $\rho' = \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]$ and $\Delta' = \Delta[Z(\boldsymbol{x}) \mapsto (\mu Z(\boldsymbol{x}).\psi)^k]$ with $k \in \mathbb{N}$ such that

$$(C, \rho') \in \{\!|(\mu Z(\boldsymbol{x}).\psi)^{k+1}_{\Delta}|\!\}_\pi \smallsetminus \{\!|(\mu Z(\boldsymbol{x}).\psi)^{k}_{\Delta}|\!\}_\pi \tag{5.6}$$

for $\pi \in PEnv_{\mathcal{E}}$. Moreover, if $\Delta(Z(\boldsymbol{x})) = (\mu Z(\boldsymbol{x}).\psi)^n$ it is required that $k < n$.

The notion of successful μ-pseudo-tableau is exactly as in Definition 5.9.

An easy but crucial observation is that rule $(\mathsf{Unf}^{\mathsf{a}}_\mu)$ is applicable in all and only the situations in which the premise is true. In fact, whenever the rule is applicable, due the existence of $k \in \mathbb{N}$ such that $(C, \rho') \in \{\!|(\mu Z(\boldsymbol{x}).\psi)^{k+1}_{\Delta}|\!\}_\pi \smallsetminus \{\!|(\mu Z(\boldsymbol{x}).\psi)^{k}_{\Delta}|\!\}_\pi$ and $k < n$ if $\Delta(Z(\boldsymbol{x}))$ is an approximant, the premise is guaranteed to be true. On the other hand, if the premise is true, such a k is guaranteed to exists, and thus the rule is applicable. Additionally, rule $(\mathsf{Unf}^{\mathsf{a}}_\mu)$ preserves the well-formedness of the sequents when applied to a true premise, i.e., if the premise is well-formed and true then the conclusion is well-formed. This immediately follows from the observation that for $k < n$ it holds $fp((\mu Z(\boldsymbol{x}).\psi)^k) \subseteq fp((\mu Z(\boldsymbol{x}).\psi)^n) \subseteq fp(\mu Z(\boldsymbol{x}).\psi)$. Moreover, it is easy to see that rule $(\mathsf{Unf}^{\mathsf{a}}_\mu)$ is forwards sound, in the sense described by the next result.

Lemma 5.12 (unfolding μ-fixpoints)**.** *Let $C, \rho, \Delta \models Z(\boldsymbol{z})$ be a sequent such that $\Delta(Z(\boldsymbol{x})) = \mu Z(\boldsymbol{x}).\psi$ or $\Delta(Z(\boldsymbol{x})) = (\mu Z(\boldsymbol{x}).\psi)^n$. If $C, \rho, \Delta \models Z(\boldsymbol{z})$ is true and the stop condition γ does not hold, then rule $(\mathsf{Unf}^{\mathsf{a}}_{\mu})$ is applicable and the conclusion is true.*

Proof. Assume that the stop condition γ does not hold. If the sequent $C, \rho, \Delta \models Z(\boldsymbol{z})$ is true, i.e., $(C, \rho) \in \{\!|(Z(\boldsymbol{z}))_\Delta|\!\}_\pi$ for $\pi \in \mathit{PEnv}_{\mathcal{E}}$, then, by Lemma 4.1(1), $(C, \rho') \in \{\!|(Z(\boldsymbol{x}))_\Delta|\!\}_\pi$ and since $(Z(\boldsymbol{x}))_\Delta = (\Delta(Z(\boldsymbol{x})))_\Delta$, then $(C, \rho') \in \{\!|(\Delta(Z(\boldsymbol{x})))_\Delta|\!\}_\pi$. When $\Delta(Z(\boldsymbol{x})) = \mu Z(\boldsymbol{x}).\psi$ some $k \in \mathbb{N}$ such that condition (5.6) holds is guaranteed to exist by applying first Lemma 5.9(1) and then Lemma 5.8(2). When instead, $\Delta(Z(\boldsymbol{x})) = (\mu Z(\boldsymbol{x}).\psi)^n$, we need only Lemma 5.9(3). Moreover, in this case, we have $k < n$, since $(C, \rho') \in \{\!|(\mu Z(\boldsymbol{x}).\psi)^n_\Delta|\!\}_\pi$ and $(C, \rho') \in \{\!|(\mu Z(\boldsymbol{x}).\psi)^{k+1}_\Delta|\!\}_\pi \smallsetminus \{\!|(\mu Z(\boldsymbol{x}).\psi)^k_\Delta|\!\}_\pi$.

Since $Z \notin \mathit{fp}((\mu Z(\boldsymbol{x}).\psi)^k)$ and Δ and the definition lists Δ' differ only on Z, we have that $(\psi)_{\Delta'} = (\psi[Z(\boldsymbol{x}) \coloneqq (\mu Z(\boldsymbol{x}).\psi)^k_\Delta])_\Delta = (\mu Z(\boldsymbol{x}).\psi)^{k+1}_\Delta$ and thus $(C, \rho') \in \{\!|(\psi)_{\Delta'}|\!\}_\pi$, i.e., the successor sequent $C, \rho', \Delta' \models \psi$ is true. □

Theorem 5.1 can be easily extended to μ-pseudo-tableaux, i.e., we can prove that every μ-pseudo-tableau is finite. Moreover, notice that a successful μ-pseudo-tableau can be transformed into a successful tableau simply by replacing, in all sequents, the definition list Δ with Δ' where each approximant is substituted by the corresponding least fixpoint. More precisely, for all $X(\boldsymbol{x}) \in \mathit{dom}(\Delta)$, we let $\Delta'(X) = \mu X(\boldsymbol{x}).\psi$ if $\Delta(X(\boldsymbol{x})) = (\mu X(\boldsymbol{x}).\psi)^n$, and $\Delta'(X(\boldsymbol{x})) = \Delta(X(\boldsymbol{x}))$, otherwise. Using this fact one can prove completeness.

Lemma 5.13 (completeness)**.** *Let $\mathcal{E}$ be a strongly regular PES and let φ be a closed formula of $\mathcal{L}_{hp}$. If $\mathcal{E}$ satisfies φ then φ has a successful μ-pseudo tableau and thus a successful tableau.*

Proof. We prove that each true sequent $C, \rho, \Delta \models \varphi$ admits a successful μ-pseudo-tableau. We first construct the tableau inductively, showing which rule to apply and arguing that the conclusion of the rule is again a true sequent. Then we will show that it is successful. We distinguish various cases according to the shape of φ:

- $\varphi = \mathsf{T}$
  The node has no successors.

- $\varphi = \psi_1 \wedge \psi_2$
  We apply rule $(\wedge)$ which produces two successors labelled $C, \rho, \Delta \models \psi_1$ and $C, \rho, \Delta \models \psi_2$. Clearly $(\varphi)_\Delta = (\psi_1)_\Delta \wedge (\psi_2)_\Delta$, and since $(C, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$, by definition of the semantics both sequents are true.

- $\varphi = \psi_1 \vee \psi_2$
  Since $(C, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$ and clearly $(\varphi)_\Delta = (\psi_1)_\Delta \vee (\psi_2)_\Delta$, then by definition of the semantics $(C, \rho) \in \{\!|(\psi_i)_\Delta|\!\}_\pi$, for some $i \in \{1, 2\}$. We apply rule $(\vee_{\mathsf{L}})$ or $(\vee_{\mathsf{R}})$, accordingly, producing a single true successor labelled $C, \rho, \Delta \models \psi_i$.

- $\varphi = \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle\, \psi$
  Since $(C, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$ and $(\varphi)_\Delta = \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle\, (\psi)_\Delta$, by definition of the semantics there exists an event $e \in \mathcal{E}[C]$ such that $C \xrightarrow{\rho(\boldsymbol{x}), \rho(\boldsymbol{y}) < e}_{\mathsf{a}} C'$, $\rho' = \rho[z \mapsto e]$ and $(C', \rho') \in \{\!|(\psi)_\Delta|\!\}_\pi$. Then we can apply rule $(\Diamond)$ producing a single true successor labelled $C', \rho', \Delta \models \psi$.

- $\varphi = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \psi$
  Let the set of successor be $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a} z}(C, \rho) = \{(C_1, \rho_1), \ldots, (C_n, \rho_n)\}$. Note that successors $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a} z}(C, \rho)$ are finite since the PES $\mathcal{E}$ is strongly regular and thus finitely branching.

  If $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a} z}(C, \rho)$ is not empty, since $(C, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$ and $(\varphi)_\Delta = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, (\psi)_\Delta$, by definition of the semantics we have that $(C_i, \rho_i) \in \{\!|(\psi)_\Delta|\!\}_\pi$ for all $i \in [1, n]$. Then rule $(\Box)$ can be applied producing n true successors labelled $C_i, \rho_i, \Delta \models \psi$, for $i \in [1, n]$.

  Otherwise, if $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a} z}(C, \rho)$ is empty, the node has no successor.

- $\varphi = (\eta Z(\boldsymbol{x}).\psi)(\boldsymbol{z})$
  We apply rule (Int) that produces a single successor labelled $C, \rho, \Delta' \models Z(\boldsymbol{z})$ where $\Delta' = \Delta[Z(\boldsymbol{x}) \mapsto \eta Z(\boldsymbol{x}).\psi]$.

  We show that the sequent $C, \rho, \Delta' \models Z(\boldsymbol{z})$ is true, i.e., $(C, \rho) \in \{\!|(Z(\boldsymbol{z}))_{\Delta'}|\!\}_\pi$ for $\pi \in \mathit{PEnv}_{\mathcal{E}}$. First note that $(\varphi)_\Delta = (\varphi)_{\Delta'}$, since $Z \notin \mathit{fp}(\varphi)$ and Δ and Δ' coincide on all propositions except Z. Therefore $(Z(\boldsymbol{z}))_{\Delta'} = (\Delta'(Z(\boldsymbol{z})))_{\Delta'} = (\varphi)_{\Delta'} = (\varphi)_\Delta$. Since $(C, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$, we have that $(C, \rho) \in \{\!|(Z(\boldsymbol{z}))_{\Delta'}|\!\}_\pi$, as desired.

- $\varphi = Z(\boldsymbol{z})$
  We know that the sequent $C, \rho, \Delta \models \varphi$ is true, i.e., $(C, \rho) \in \{\!|(Z(\boldsymbol{z}))_\Delta|\!\}_\pi$ for $\pi \in \mathit{PEnv}_{\mathcal{E}}$. If γ holds then the node has no successors. Otherwise, if γ is false, we distinguish two cases depending on the definition of Z in Δ:

  1. $\Delta(Z(\boldsymbol{x})) = \nu Z(\boldsymbol{x}).\psi$
     Since $(C, \rho) \in \{\!|(Z(\boldsymbol{z}))_\Delta|\!\}_\pi$, we have that $(C, \rho') \in \{\!|(Z(\boldsymbol{x}))_\Delta|\!\}_\pi$ where $\rho' = \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]$. By Lemma 5.1 $\{\!|(Z(\boldsymbol{x}))_\Delta|\!\}_\pi = \{\!|(\psi)_\Delta|\!\}_\pi$ and thus $(C, \rho') \in \{\!|(\psi)_\Delta|\!\}_\pi$. Therefore by using rule (Unf_ν) we produce a single true successor labelled by $C, \rho', \Delta \models \psi$.
  2. $\Delta(Z(\boldsymbol{x})) = \mu Z(\boldsymbol{x}).\psi$ or $(\mu Z(\boldsymbol{x}).\psi)^n$
     By Lemma 5.12, since the sequent $C, \rho, \Delta \models \varphi$ is true, we can apply rule $(\mathsf{Unf}_\mu^{\mathsf{a}})$ thus producing a true successor labelled $C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})], \Delta' \models \psi$ where $\Delta' = \Delta[Z(\boldsymbol{x}) \mapsto (\mu Z(\boldsymbol{x}).\psi)^k]$, for some $k \in \mathbb{N}$ such that $(C, \rho') \in \{\!|(\mu Z(\boldsymbol{x}).\psi)^{k+1}_\Delta|\!\}_\pi \setminus \{\!|(\mu Z(\boldsymbol{x}).\psi)^k_\Delta|\!\}_\pi$ which is ensured to exist.

Observe that above we do not need to consider the case $\varphi = \mathsf{F}$ since the sequent $C, \rho, \Delta \models \varphi$ would be false.

The inductive construction produces a μ-pseudo-tableau where all sequents are true. The only reason why it might not be successful could be the presence of a leaf labelled $C, \rho, \Delta \models X(\boldsymbol{z})$ where $\Delta(X(\boldsymbol{x})) = (\mu X(\boldsymbol{x}).\psi)^k$. We next show that this would imply that $C, \rho, \Delta \models X(\boldsymbol{z})$ is false, hence it is not possible.

Assume that there is a leaf l labelled $C, \rho, \Delta \models X(\boldsymbol{z})$ where $\Delta(X(\boldsymbol{x})) = (\mu X(\boldsymbol{x}).\psi)^k$. We prove that $C, \rho, \Delta \models X(\boldsymbol{z})$ is false, namely that $(C, \rho) \notin \{\!|(X(\boldsymbol{z}))_\Delta|\!\}_\pi$ for $\pi \in PEnv_{\mathcal{E}}$. Observe that $(X(\boldsymbol{z}))_\Delta = ((\mu X(\boldsymbol{x}).\psi)^k[\boldsymbol{z}/\boldsymbol{x}])_\Delta = (\mu X(\boldsymbol{x}).\psi)^k_\Delta[\boldsymbol{z}/\boldsymbol{x}]$, hence, by Lemma 4.1(1), we can conclude by proving that

$$(C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]) \notin (\mu X(\boldsymbol{x}).\psi)^k_\Delta. \tag{5.7}$$

Since l is a leaf, it must satisfy the stop condition, i.e., it must have an ancestor v labelled $C', \rho', \Delta' \models X(\boldsymbol{z}')$ such that $\Delta^\uparrow(X) = \Delta'^\uparrow(X)$ and $\langle C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]|_{\boldsymbol{x}}\rangle \approx_r \langle C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{z}')]|_{\boldsymbol{x}}\rangle$. By construction, the successor of v is labelled $C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{z}')], \Delta'' \models \psi$, where $\Delta'' = \Delta'[X(\boldsymbol{x}) \mapsto (\mu X(\boldsymbol{x}).\psi)^n]$ for some $n \in \mathbb{N}$ such that $(C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{z}')]) \in \{\!|(\mu X(\boldsymbol{x}).\psi)^{n+1}_{\Delta'}|\!\}_\pi \setminus \{\!|((\mu X(\boldsymbol{x}).\psi)^{n+1}_{\Delta'}|\!\}_\pi$ for $\pi \in PEnv_{\mathcal{E}}$. Recall that each time a proposition is unfolded using rule $(\mathsf{Unf}^{\mathsf{a}}_\mu)$, the index of the approximant strictly decreases, hence $k \leq n$. Then, we have that $(C', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{z}')]) \notin \{\!|(\mu X(\boldsymbol{x}).\psi)^k_{\Delta'}|\!\}_\pi$ since $\{\!|(\mu X(\boldsymbol{x}).\psi)^k_{\Delta'}|\!\}_\pi \subseteq \{\!|(\mu X(\boldsymbol{x}).\psi)^n_{\Delta'}|\!\}_\pi$. Therefore, since π is vacuously saturated for $(\mu X(\boldsymbol{x}).\psi)^k_{\Delta'}$ that has no free propositions, by Lemma 5.3, we deduce that also $(C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]) \notin \{\!|(\mu X(\boldsymbol{x}).\psi)^k_{\Delta'}|\!\}_\pi$.

We then reach the desired conclusion (5.7) by showing that $(\mu X(\boldsymbol{x}).\psi)^k_{\Delta'} = (\mu X(\boldsymbol{x}).\psi)^k_\Delta$. Note that, since $\Delta^\uparrow(X) = \Delta'^\uparrow(X)$, no proposition $Y \in fp(\psi) \setminus \{X\}$ can be unfolded or introduced along the path from v to l. Otherwise, if φ_X and φ_Y are the fixpoint formula binding X and Y, respectively, we would have $\varphi_Y \sqsubset^*_d \varphi_Y \sqsubset^*_d \varphi_X$, while $\sqsubset^*_d$ is a partial order. Thus, since the definition list is updated only when a proposition is unfolded or introduced, Δ and Δ' coincide on all propositions in $fp((\mu X(\boldsymbol{x}).\psi)^k) \subseteq fp(\psi) \setminus \{X\}$. Clearly this implies that $(\mu X(\boldsymbol{x}).\psi)^k_{\Delta'} = ((\mu X(\boldsymbol{x}).\psi)^k)_\Delta$, as desired.

All in all, we can conclude that there is a successful μ-pseudo-tableau (hence a successful tableau) for each true sequent. □

Combining the previous lemmata we conclude the desired result.

Theorem 5.2 (soundness and completeness of the tableau system)**.** *Given a strongly regular PES $\mathcal{E}$ and a closed formula φ of $\mathcal{L}_{hp}$, the formula φ has a successful tableau if and only if $\mathcal{E}$ satisfies φ.*

Proof. Corollary of Lemmata 5.11 and 5.13. □

Chapter 6

Automata

Tableaux served well for providing a constructive decidability result for the model-checking problem in the logic $\mathcal{L}_{hp}$. However, it is often the case that tableaux are not the most efficient tool for practically solving this kind of problem. On the way to a practical implementation of a model-checker for $\mathcal{L}_{hp}$, in this chapter we develop an automata-theoretic approach to model-checking $\mathcal{L}_{hp}$. We show how the model-checking problem for $\mathcal{L}_{hp}$ on strongly regular PESs can be reduced to the non-emptiness of the language of suitably generated nondeterministic parity tree automata.

Parity tree automata [Mos85] are a class of automata on infinite trees with an acceptance condition based on priorities assigned to the states, requiring that the highest recurring priority in a run is even (hence their name). This class of automata is just one of the many that have been widely adopted in problems related to temporal logics, others are: Muller automata [Mul63], Rabin automata [Rab69], Streett automata [Str82] (see [Wol90] for an overview of the most fundamental results for all these and more). Formally, the choice among these classes is inconsequential as the languages definable by these automata are the same. Nevertheless, the slight differences can make some easier to use than others depending on the setting.

Usually automata obtained in model-checking methods are finite, mainly because of the finiteness of the models. Instead, as we will see, the automaton that naturally arises from a PES and a formula has an infinite number of states. We discuss how it can be quotiented to a finite automaton accepting the same language, which can thus be used for model-checking purposes.

We first introduce the abstract model-checking method over event structures, based on the construction of a nondeterministic parity tree automaton. Then, we show how it can be instantiated on finite safe Petri nets, providing a concrete effective procedure. We conclude by presenting a prototype tool for model-checking $\mathcal{L}_{hp}$ over finite safe Petri nets.

6.1 Infinite parity tree automata and their quotient

Automata on infinite trees revealed to be a powerful tool for various problems in the setting of branching temporal logics. Here we focus on nondeterministic parity tree automata with some slightly non-standard features. We work on k-trees (rather than on binary trees), a choice that will simplify the presentation, and we allow for possibly infinite state automata.

When automata are meant to be used for model-checking purposes it is standard to restrict to unlabelled trees. A *k-bounded branching tree*, or *k-tree* for short, is a subset $\mathcal{T} \subseteq [1,k]^*$, such that

1. $\mathcal{T}$ is prefix closed, i.e., if $uv \in \mathcal{T}$ then $u \in \mathcal{T}$
2. $u1 \in \mathcal{T}$ for all $u \in \mathcal{T}$
3. for all $i \in [2,k]$ if $ui \in \mathcal{T}$ then $u(i-1) \in \mathcal{T}$.

Elements of $\mathcal{T}$ are the nodes of the tree. The empty sequence ϵ corresponds to the root. A sequence of the form ui corresponds to the i-th child of node u. Hence by (2) each branch is infinite and by (3) the presence of the i-th child implies the presence of the j-th children for $j \leq i$.

Definition 6.1 (nondeterministic parity automaton)**.** *A* k-bounded nondeterministic parity tree automaton *(NPA) is a tuple* $\mathcal{A} = \langle Q, \to, q_0, \mathcal{F} \rangle$ *where*

- Q *is a set of states,*
- $\to \subseteq Q \times \bigcup_{i=1}^{k} Q^i$ *is the* transition relation*,*
- $q_0 \in Q$ *is the initial state,*
- $\mathcal{F} = (F_0, \ldots, F_h)$ *is the* acceptance condition*, where* $F_0, \ldots, F_h \subseteq Q$ *are mutually disjoint subsets of states. A state* $q \in Q$ *has* priority i *if* $q \in F_i$ *for some* $i \in [0,h]$*.*

Transitions are written as $q \to (q_1, \ldots, q_m)$ instead of $(q, (q_1, \ldots, q_m)) \in \to$.

An example of (finite) NPA is depicted in Figure 6.1. The initial state of the automaton is q_0. Transitions are represented by arrows. In particular, the only transition from q_0 goes into both q_1 and q_2, that is, $q_0 \to (q_1, q_2)$. Instead, all the other transitions are simply from one state to another (or itself in the case of $q_4 \to (q_4)$).

Possible runs of an NPA are based on its structure, that is, its transition relation. The acceptance of a run depends on the priorities assigned by the acceptance condition to the states the run goes through.

Definition 6.2 (accepting runs)**.** *Let* $\mathcal{A} = \langle Q, \to, q_0, \mathcal{F} \rangle$ *be an NPA where* $\mathcal{F} = (F_0, \ldots, F_h)$*. Given a* k*-tree* $\mathcal{T}$*, a* run *of* $\mathcal{A}$ *on* $\mathcal{T}$ *is a labelling of* $\mathcal{T}$ *over the states* $r : \mathcal{T} \to Q$ *consistent with the transition relation, i.e., such that* $r(\epsilon) = q_0$ *and for all* $u \in \mathcal{T}$*, with* m *children,*

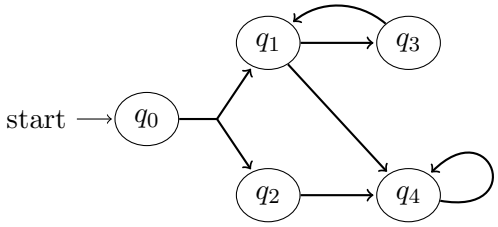


Figure 6.1: Example of a finite 2-bounded NPA.

there is a transition $r(u) \to (r(u1), \ldots, r(um))$ *in* $\mathcal{A}$. *A* path *in the run* r *is an infinite sequence of states* $p = (q_0, q_1, \ldots)$ *labelling a complete path from the root in the tree. It is called* accepting *if there exists an even number* $l \in [0, h]$ *such that the set* $\{j \mid q_j \in F_l\}$ *is infinite and the set* $\{j \mid q_j \in \bigcup_{l<i\leq h} F_i\}$ *is finite. In this case we denote such* l *by* $\mathcal{F}(p)$. *The run* r *is* accepting *if all paths in* r *are accepting.*

In words, given a run r, a path p in r is accepting if there are priorities i such that states having priority i occur infinitely often in the path p and the greatest of such priorities is even. The run is accepting if all paths in it are.

For instance, consider the NPA in Figure 6.1. Based on its transition relation, the only possible runs are labellings of the 2-tree $\mathcal{T}$ made of two infinite paths completely separated except for their initial node, which is the root of $\mathcal{T}$. An example of such a run is the one including the sequences $(q_0, q_1, q_3, q_1, q_4^\infty)$ and (q_0, q_2, q_4^∞), both looping indefinitely over q_4 once they reach it. Assume that the acceptance condition of the automaton is $\mathcal{F} = (F_0, F_1)$, with $F_0 = \{q_4\}$ and $F_1 = \{q_3\}$. This means that q_4 has priority 0 and q_3 has priority 1, while all the other states q_0, q_1, q_2 have no priority. Then, the run proposed before would be accepting, since in both its paths there is a state occurring infinitely often with even priority, that is q_4, and no higher priority appears infinitely many times. Indeed, although q_3, which has priority 1, occurs in one of the paths, it does so finitely many times (only once).

In the sequel we will also refer to a *path in an NPA*, meaning a possibly infinite sequence $p = (q_1, q_2, \ldots)$ such that for all $i < |p|$ there is a transition $q_i \to (\ldots, q_{i+1}, \ldots)$. Note that, given a run r of an automaton $\mathcal{A}$, each path in r is also a path in $\mathcal{A}$.

Definition 6.3 (language of an NPA)**.** *Let* $\mathcal{A}$ *be an NPA. The* language *of* $\mathcal{A}$*, denoted by* $L(\mathcal{A})$*, consists of the trees* $\mathcal{T}$ *that admit an accepting run.*

Observe that for a k-bounded NPA, the language $L(\mathcal{A})$ consists of a set of k-trees.

The possibility of having an infinite number of states in an NPA and the associated acceptance condition are somehow non-standard. However, it is easy to see that whenever an NPA is finite, the acceptance condition coincides with the standard one requiring a single state with maximal even priority to occur infinitely often in each path. In fact, for finite NPAs, all sets in the acceptance condition $\mathcal{F}$ are finite and thus asking that for a path

$p = (q_0, q_1, \ldots)$ the set $\{j \mid q_j \in F_i\}$ is infinite amounts to asking that some state in F_i repeats infinitely often in the path p.

The NPA naturally associated with a formula and an event structure will be infinite. In order to have an effective procedure for checking the satisfaction of a formula we will build a suitable quotient of the NPA, with respect to an equivalence which preserves emptiness of the language.

For this reason we next introduce a notion of bisimulation over NPAs and observe that bisimulation equivalences preserve the language of NPAs (and thus in particular language emptiness). An analogous notion is studied in [AKH06] in the setting of nondeterministic tree automata over finite trees.

Definition 6.4 (automaton bisimulation)**.** *Given an NPA $\mathcal{A} = \langle Q, \to, q_0, \mathcal{F}\rangle$ where $\mathcal{F} = (F_0, \ldots, F_h)$, a symmetric relation R over the set of states Q is an* automaton bisimulation*, or simply a bisimulation, whenever for all $(q, q') \in R$ it holds*

1. *for all $i \in [0, h]$, $q \in F_i$ iff $q' \in F_i$*
2. *if $q \to (q_1, \ldots, q_m)$ then $q' \to (q'_1, \ldots, q'_m)$ with $(q_i, q'_i) \in R$ for all $i \in [1, m]$.*

Given an NPA $\mathcal{A}$ and an equivalence $\equiv$ on the set of states which is a bisimulation, we define the quotient as $\mathcal{A}_{/\equiv} = \langle Q_{/\equiv}, \to_{/\equiv}, [q_0]_\equiv, \mathcal{F}_{/\equiv}\rangle$ where $[q]_\equiv \to_{/\equiv} ([q_1]_\equiv, \ldots, [q_m]_\equiv)$ if $q \to (q_1, \ldots, q_m)$ and $\mathcal{F}_{/\equiv} = (F_{0/\equiv}, \ldots, F_{h/\equiv})$. Note that by condition (1) in Definition 6.4 above, the acceptance condition is well-defined, i.e., all $F_{i/\equiv}$ are pairwise disjoint. We can show that an NPA and its quotient accept exactly the same language.

Theorem 6.1 (language preservation)**.** *Let $\mathcal{A}$ be an NPA and let $\equiv$ be an equivalence on the set of states which is a bisimulation. Then $L(\mathcal{A}_{/\equiv}) = L(\mathcal{A})$.*

Proof. Let $\mathcal{A} = \langle Q, \to, q_0, \mathcal{F}\rangle$ be an NPA where $\mathcal{F} = (F_0, \ldots, F_h)$, and let $\equiv$ be an equivalence on Q which is a bisimulation. We prove the two inclusions separately.

Let us first show that $L(\mathcal{A}) \subseteq L(\mathcal{A}_{/\equiv})$. Let $T \in L(\mathcal{A})$ be a tree. Then there exists an accepting run $r : T \to Q$. It is easy to show that $r' : T \to Q_{/\equiv}$ defined by $r'(u) = [r(u)]_\equiv$ for all $u \in T$ is an accepting run for T in $\mathcal{A}_{/\equiv}$.

In fact, it is a run on T since $r'(\epsilon) = [q_0]_\equiv$, which is the initial state of $\mathcal{A}_{/\equiv}$. Moreover, for all $u \in T$, if $u1, \ldots, un$ are its children, then $r(u) \to (r(u1), \ldots, r(un))$ in $\mathcal{A}$, hence $r'(u) = [r(u)]_\equiv \to_{/\equiv} ([r(u1)]_\equiv, \ldots, [r(un)]_\equiv) = (r'(u1), \ldots, r'(un))$, as desired.

Finally, r' is accepting since any path $([q_0]_\equiv, [q_1]_\equiv, [q_2]_\equiv, \ldots)$ in r' arises from a path $(q_0, q_1, q_2, \ldots)$ in r, which is accepting, and by construction $[q]_\equiv \in F_{i/\equiv}$ iff $q \in F_i$ for all $q \in Q$ and $i \in [0, h]$.

For the converse inclusion $L(\mathcal{A}_{/\equiv}) \subseteq L(\mathcal{A})$, let $T \in L(\mathcal{A}_{/\equiv})$ be a tree. Then there exists an accepting run $r' : T \to Q_{/\equiv}$. We can define a corresponding run $r : T \to Q$ in $\mathcal{A}$ as follows. For any node $u \in T$ we inductively define $r(u) \in Q$ in a way that $r'(u) = [r(u)]_\equiv$.

For the root $r(\epsilon) = q_0$. For any each node $u \in T$ with children $u1, \ldots, un$, we know that there is a transition in the quotient $r'(u) \to_{/\equiv} (r'(u1), \ldots, r'(un))$. By construction this means that there is a transition $q \to (q_1, \ldots, q_n)$ in $\mathcal{A}$ such that $r'(u) = [q]_\equiv$ and $r'(ui) = [q_i]_\equiv$ for $i \in [1, n]$. By induction we have $r(u) \in Q$ such that $r'(u) = [r(u)]_\equiv$, hence $r(u) \equiv q$. Since $\equiv$ is a bisimulation, there are $q'_1, \ldots, q'_n \in Q$ such that $r'(u) \to (q'_1, \ldots, q'_n)$ and $q_i \equiv q'_i$ for $i \in [1, n]$. Therefore, for $i \in [1, n]$, we can define $r(ui) = q'_i$ and obtain $[r(ui)]_\equiv = [q_i]_\equiv = r'(ui)$, as desired.

The fact that r is accepting immediately follows from the fact that r' is so and from the way $\mathcal{F}_{/\equiv}$ is defined from $\mathcal{F}$. □

Since NPAs are nondeterministic, different runs (possibly infinitely many) can exist for the same input tree. Still, the non-emptiness problem is decidable when the number of states is finite. In fact, although not discussed explicitly here, the problem can be solved by resorting to a corresponding parity game (see [Kla02]) based on the graph structure of the automaton, in particular of its loops. This holds also for the k-bounded variant of NPAs in this thesis, where the standard procedure for the automata on binary trees can be easily adapted.

6.2 NPAs for model checking

We show how, given a PES and a closed formula of $\mathcal{L}_{hp}$, we can build an NPA in a way that, for strongly regular PESs, the satisfaction of φ in $\mathcal{E}$ reduces to the non-emptiness of the language of the NPA. The construction is inspired by that in [EJS01] for the mu-calculus. The automaton is typically infinite, but we discuss how an effective model-checking procedure can be obtained by quotienting the infinite NPA to a finite one.

We first illustrate the construction of the NPA for a formula and a PES.

Definition 6.5 (NPA for a formula)**.** *Let $\mathcal{E}$ be a (boundedly branching) PES and let φ be a closed formula of $\mathcal{L}_{hp}$. The NPA for $\mathcal{E}$ and φ is $\mathcal{A}_{\mathcal{E},\varphi} = \langle Q, \to, q_0, \mathcal{F} \rangle$ defined as follows. The set of states $Q \subseteq \mathcal{C}(\mathcal{E}) \times Env_\mathcal{E} \times \mathsf{sf}(\varphi)$ is defined as $Q = \{(C, \rho, \psi) \mid \rho(fv(\psi)) \subseteq C\}$. The initial state $q_0 = (\emptyset, \rho, \varphi)$, for some chosen environment $\rho \in Env_\mathcal{E}$. The transition relation is defined, for any state $q = (C, \rho, \psi) \in Q$, by:*

- *if $\psi = \mathsf{T}$ or $\psi = \mathsf{F}$, then $q \to (q)$*
- *if $\psi = \psi_1 \wedge \psi_2$, then $q \to (q_1, q_2)$ where $q_i = (C, \rho, \psi_i)$, $i \in \{1, 2\}$*
- *if $\psi = \psi_1 \vee \psi_2$, then $q \to (q_1)$ and $q \to (q_2)$ where $q_i = (C, \rho, \psi_i)$, $i \in \{1, 2\}$*
- *if $\psi = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\psi'$ and $\mathsf{Succ}_\mathcal{E}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C, \rho) = \{(C_1, \rho_1), \ldots, (C_n, \rho_n)\} \neq \emptyset$ then $q \to (q_1, \ldots, q_n)$ where $q_i = (C_i, \rho_i, \psi')$ for $i \in [1, n]$, otherwise $q \to (q)$*
- *if $\psi = \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!\rangle \psi'$ and $\mathsf{Succ}_\mathcal{E}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C, \rho) = \{(C_1, \rho_1), \ldots, (C_n, \rho_n)\} \neq \emptyset$ then $q \to (q_i)$ where $q_i = (C_i, \rho_i, \psi')$ for $i \in [1, n]$, otherwise $q \to (q)$*

- *if* $\psi = (\eta X(\boldsymbol{x}).\psi')(\boldsymbol{y})$*, then* $q \to (q')$ *where* $q' = (C, \rho, X(\boldsymbol{y}))$
- *if* $\psi = X(\boldsymbol{z})$ *and* $\psi' \in \mathsf{sf}(\varphi)$ *is the unique subformula such that* $\psi' = (\eta X(\boldsymbol{x}).\psi'')(\boldsymbol{y})$ *then* $q \to (q')$ *where* $q' = (C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})], \psi'')$*.*

The acceptance condition is $\mathcal{F} = (F_0, \ldots, F_h)$ *where* $h = \mathsf{ad}(\varphi) + 1$ *and the sets* F_i *are as follows. Consider* $A_0, \ldots, A_h \subseteq \mathsf{sf}(\varphi)$ *such that for* $i \in [0, h]$*, if* i *is even (odd) then* A_i *contains exactly all propositions quantified in* ν*-subformulae (*μ*-subformulae) with alternation depth* i *or* $i - 1$*. Then* $F_0 = (\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \times (A_0 \cup \{\mathsf{T}\})) \cup B$ *where* $B = \{(C, \rho, [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\psi) \mid \mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C, \rho) = \emptyset\}$ *is the set of all subformulae of* φ *in a context where they are trivially true, and* $F_i = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \times A_i$*, for* $i \in [1, h]$*.*

States of $\mathcal{A}_{\mathcal{E},\varphi}$ are triples (C, ρ, φ) consisting of a configuration C, an environment ρ and a subformula ψ of the original formula φ. The environment ρ fixed for the initial state is irrelevant, since the formula φ is closed. The intuition is that a transition reduces the truth of a formula in a state to that of subformulae in possibly updated states. In this sense the automaton transitions are similar to the tableau rules of the previous chapter. Furthermore, we will see later that they also share similar properties. A transition can just decompose the formula, as it happens for $\wedge$ or $\vee$, check the satisfaction of a modal operator, thus changing the state consequently, or unfold a fixpoint. Whenever $q = (C, \rho, \psi)$ with $\psi = (\eta X(\boldsymbol{x}).\psi')(\boldsymbol{y})$ and thus $q \to (q')$ with $q' = (C, \rho, X(\boldsymbol{y}))$ we say that q *introduces* X. If $\psi = X(\boldsymbol{z})$ and thus $q \to (q')$ where $q' = (C, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})], \psi'')$, with ψ' the body of the fixpoint subformula quantifying X in φ, we say that q *unfolds* X. The automaton $\mathcal{A}_{\mathcal{E},\varphi}$ is bounded (by the branching bound of the PES) but normally infinite (whenever the PES $\mathcal{E}$ is infinite and the formula φ includes some non-trivial fixpoint).

We next show that for a strongly regular PES the truth of the formula φ on the PES $\mathcal{E}$ reduces to the non-emptiness of the language of $\mathcal{A}_{\mathcal{E},\varphi}$. A basic ingredient is an equivalence over the states of the NPA that can be defined relying on the notion of isomorphism of residuals of pointed configurations (Definition 5.4).

Definition 6.6 (future equivalence)**.** *Let* $\mathcal{E}$ *be a PES,* φ *be a formula and let* $q_i = (C_i, \rho_i, \psi_i)$*,* $i \in \{1, 2\}$ *be two states of the NPA* $\mathcal{A}_{\mathcal{E},\varphi}$*. We say that* q_1 *and* q_2 *are* future equivalent*, written* $q_1 \approx_f q_2$*, if there exists a formula* ψ *and substitutions* $\sigma_i : fv(\psi) \to fv(\psi_i)$ *such that* $\psi\sigma_i = \psi_i$*, for* $i \in \{1, 2\}$*, and the* $fv(\psi)$*-pointed configurations* $\langle C_i, \rho_i \circ \sigma_i \rangle$ *have isomorphic pointed residuals.*

Intuitively, two states are equivalent if they involve the same subformula (up to renaming of the event variables) and the configurations, pointed by the free variables in the formulae, have isomorphic residuals. Future equivalence can be shown to be a bisimulation on the NPA $\mathcal{A}_{\mathcal{E},\varphi}$ in the sense of Definition 6.4 and, whenever $\mathcal{E}$ is strongly regular, it is of finite index.

Proposition 6.1 ($\approx_f$ is a bisimulation)**.** *Let $\mathcal{E}$ be a strongly regular PES and let φ be a closed formula of $\mathcal{L}_{hp}$. Then the future equivalence $\approx_f$ on $\mathcal{A}_{\mathcal{E},\varphi}$ is a bisimulation and it is of finite index.*

Proof. Let $q_1 = (C_1, \rho_1, \psi_1), q_2 = (C_2, \rho_2, \psi_2) \in Q$ be states of $\mathcal{A}_{\mathcal{E},\varphi}$ such that $q_1 \approx_f q_2$. This means that

i. there exists a formula ψ and substitutions $\sigma_i : fv(\psi) \to fv(\psi_i)$, $i \in \{1, 2\}$, such that $\psi\sigma = \psi_i$ and

ii. the $fv(\psi)$-pointed configurations $\langle C_1, \rho_1 \circ \sigma_1\rangle$, $\langle C_2, \rho_2 \circ \sigma_2\rangle$ have isomorphic pointed residuals via some isomorphism $\iota : \mathcal{E}[C_1] \to \mathcal{E}[C_2]$.

The first observation is that (i) and (ii) above, imply that $q_1 \in F_i$ iff $q_2 \in F_i$, for $i \in [0, h]$, i.e., condition (1) of the definition of bisimulation (Definition 6.4).

In fact, by item (i), ψ_1 and ψ_2 differ at most for the name of the event variables, hence they have the same outer most operator and the same alternation depth. Moreover, if ψ_1 and hence ψ_2 are modal formulae, e.g., $\psi_1 = \langle\!|\boldsymbol{x}_1, \overline{\boldsymbol{y}_1} < \mathsf{a}\, z_1|\!\rangle \psi_1'$ and $\psi_2 = \langle\!|\boldsymbol{x}_2, \overline{\boldsymbol{y}_2} < \mathsf{a}\, z_2|\!\rangle \psi_2'$, then $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}_1, \overline{\boldsymbol{y}_1} < \mathsf{a} z_1}(C_1, \rho_1) = \emptyset$ iff $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}_2, \overline{\boldsymbol{y}_2} < \mathsf{a} z_2}(C_2, \rho_2) = \emptyset$. This follows from the more general observation that

$$C_1, \xrightarrow{\rho(\boldsymbol{x}_1), \overline{\rho(\boldsymbol{y}_1)} < e_1}_{\mathsf{a}} C_1' \quad \text{iff} \quad C_2, \xrightarrow{\rho(\boldsymbol{x}_2), \overline{\rho(\boldsymbol{y}_2)} < \iota(e_1)}_{\mathsf{a}} C_2' \tag{6.1}$$

In fact, let $C_1, \xrightarrow{\rho(\boldsymbol{x}_1), \overline{\rho(\boldsymbol{y}_1)} < e_1}_{\mathsf{a}} C_1'$ for some event e_1. Since e_1 is enabled in $\mathcal{E}[C_1]$ and ι is an isomorphism of the PESs $\mathcal{E}[C_1]$ and $\mathcal{E}[C_2]$, the image $\iota(e_1)$ is enabled in $\mathcal{E}[C_2]$. Moreover, for all $x_1 \in \boldsymbol{x}_1$, $\rho_1(x_1) < e_1$ and for all $y_1 \in \boldsymbol{y}_1$, $\neg(\rho_1(y_1) < e_1)$. Since $\psi\sigma_1 = \psi_1$, we deduce that for all $x \in \boldsymbol{x}$, $\rho_1 \circ \sigma_1(x) < e_1$ and for all $y \in \boldsymbol{y}$, $\neg(\rho_1 \circ \sigma_1(y) < e_1)$. Using the fact that ι is an isomorphism of residuals of pointed configurations, we deduce that for all $x \in \boldsymbol{x}$, $\rho_2 \circ \sigma_2(x) < \iota(e_1)$ and for all $y \in \boldsymbol{y}$, $\neg(\rho_2 \circ \sigma_2(y) < \iota(e_1))$. From this we finally get that for all $x_2 \in \boldsymbol{x}_2$, $\rho_2(x_2) < \iota(e_1)$ and for all $y_2 \in \boldsymbol{y}_2$, $\neg(\rho_2(y_2) < \iota(e_1))$, which means $C_2, \xrightarrow{\rho(\boldsymbol{x}_2), \overline{\rho(\boldsymbol{y}_2)} < \iota(e_1)}_{\mathsf{a}} C_2'$, as desired. The converse implication is immediate by symmetry.

From the above, recalling that the priority of a formula depends on the shape, the alternation depth, and for modal formulae, on the successor configurations, we conclude that condition (1) holds.

Now, let us focus on the condition (2). We proceed by cases, on the shape of the formula ψ_1 (which, as observed, is of the same shape of ψ_2) and we show only that all transitions of q_1 are simulated by q_2, since the other direction follows by symmetry.

- $\psi_1 = \mathsf{T}$ or $\psi_1 = \mathsf{F}$
  Trivial, since the only transitions of q_1 and q_2 are $q_1 \to (q_1)$ and $q_2 \to (q_2)$, and $q_1 \approx_f q_2$

- $\psi_1 = \psi_1' \wedge \psi_1''$
  The only transition of q_1 is $q_1 \to (q_1', q_1'')$ where $q_1' = (C, \rho, \psi_1')$ and $q_1'' = (C, \rho, \psi_1'')$.

By item (i) we immediately get that $\psi = \psi' \wedge \psi''$ and $\psi_2 = \psi'_2 \wedge \psi''_2$, with $\psi'\sigma_i = \psi'_i$ and $\psi''\sigma_i = \psi''_i$ for $i \in \{1, 2\}$.

We can thus consider the only transition of q_2, that is, $q_2 \to (q'_2, q''_2)$ where $q'_2 = (C_2, \rho_2, \psi'_2)$ and $q''_2 = (C_2, \rho_2, \psi''_2)$. Since we already knew that $\langle C_1, (\rho_1 \circ \sigma_1)|_{fv(\psi)}\rangle \approx_r \langle C_2, (\rho_2 \circ \sigma_2)|_{fv(\psi)}\rangle$, just observing that $fv(\psi'), fv(\psi'') \subseteq fv(\psi)$, we deduce $\langle C_1, (\rho_1 \circ \sigma_1)|_{fv(\psi')}\rangle \approx_r \langle C_2, (\rho_2 \circ \sigma_2)|_{fv(\psi')}\rangle$ and $\langle C_1, (\rho_1 \circ \sigma_1)|_{fv(\psi'')}\rangle \approx_r \langle C_2, (\rho_2 \circ \sigma_2)|_{fv(\psi'')}\rangle$. Therefore we conclude that $q'_1 \approx_f q'_2$ and $q''_1 \approx_f q''_2$, as desired.

- $\psi_1 = \psi'_1 \vee \psi''_1$, $\psi_1 = (\eta X_1(\boldsymbol{x}_1).\psi'_1)(\boldsymbol{y})$, and $\psi_1 = X_1(\boldsymbol{x}_1)$.
  Analogous to the previous case.

- $\psi_1 = \langle\!|\boldsymbol{x}_1, \overline{\boldsymbol{y}_1} < \mathsf{a}\, z_1|\!\rangle \psi'_1$
  There are two possibilities. If $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}_1, \overline{\boldsymbol{y}_1} < \mathsf{a} z_1}(C_1, \rho_1) = \emptyset$ then the only transition of q_1 is $q_1 \to (q_1)$. We already observed that in this case also $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}_2, \overline{\boldsymbol{y}_2} < \mathsf{a} z_2}(C_2, \rho_2) = \emptyset$, and thus we can take the only transition of q_2 i.e., $q_2 \to (q_2)$ and conclude since $q_1 \approx_f q_2$.

  If instead $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}_1, \overline{\boldsymbol{y}_1} < \mathsf{a} z_1}(C_1, \rho_1) = \emptyset$, the transitions of q_1 are $q_1 \to (q'_1)$ with $q'_1 = (C'_1, \rho[z_1 \mapsto e_1], \psi'_1)$ for $C_1 \xrightarrow{\rho(\boldsymbol{x}_1), \rho(\boldsymbol{y}_1) < e_1}_{\mathsf{a}} C'_1$. We observed that, in this case $C_2 \xrightarrow{\rho_2(\boldsymbol{x}), \rho_2(\boldsymbol{y}) < \iota(e_1)}_{\mathsf{a}} C'_2$ and thus for q_2 we can take the transition $q_2 \to (q'_2)$ with $q'_2 = (C'_2, \rho_2[z_2 \mapsto \iota(e_1)], \psi'_2)$. Since $e_1 \in \mathcal{E}[C_1]$, $\iota(e_1) \in \mathcal{E}[C_2]$, and ι is an isomorphism, for all $e'_1 \in \mathcal{E}[C_1]$, $e_1 < e'_1$ iff $\iota(e_1) < \iota(e'_1)$. Moreover, by (i) we get that $\psi = \langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle \psi'$ and observing that $fv(\psi'_i) \subseteq fv(\psi_i) \cup \{z_i\}$, and $fv(\psi') \subseteq fv(\psi) \cup \{z\}$, we have that $\psi'\sigma_i[z \mapsto z_i] = \psi'_i$ for $i \in \{1, 2\}$. Putting together the previous facts we obtain also that the pointed configurations $\langle C'_1, (\rho[z_1 \mapsto e_1] \circ \sigma_1[z \mapsto z_1])|_{fv(\psi')}\rangle$ and $\langle C'_2, (\rho_2[z_2 \mapsto \iota(e'_1)] \circ \sigma_2[z \mapsto z_2])|_{fv(\psi')}\rangle$ have isomorphic pointed residuals, with an isomorphism which is the restriction of ι to $\mathcal{E}[C'_1]$. Hence, we can conclude that $q'_1 \approx_f q'_2$.

- $\psi = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\psi_1$
  Analogous to the previous case.

In order to show that $\approx_f$ is of finite index, observe that for a fixed subformula $\psi \in \mathsf{sf}(\varphi)$, by Lemma 5.4 there is a finite number of $\approx_r$-equivalence classes of $fv(\psi)$-pointed configurations. Since the number of subformulae in $\mathsf{sf}(\varphi)$ is itself finite, we conclude. □

As an example, consider the formula $\langle\!|\mathsf{b}\, x|\!\rangle\varphi$ from Subsection 4.3.3, where $\varphi = \nu X(x).\psi$ and $\psi = \mu Y(x).(\langle\!|x < \mathsf{b}\, y|\!\rangle X(y) \vee \langle\!|\_ z|\!\rangle Y(x))$. The automaton $\mathcal{A}_{\mathcal{E}_3, \langle\!|\mathsf{b}\, x|\!\rangle\varphi}$ built for model-checking such formula in the PES $\mathcal{E}_3$ of Figure 3.1c would be infinite. The automaton resulting as the quotient of $\mathcal{A}_{\mathcal{E}_3, \langle\!|\mathsf{b}\, x|\!\rangle\varphi}$ with respect to the future equivalence $\approx_f$ is finite. A fragment of such automaton is reported in Figure 6.2. The acceptance condition of $\mathcal{A}_{\mathcal{E}_3, \langle\!|\mathsf{b}\, x|\!\rangle\varphi}$ is $\mathcal{F} = (F_0, F_1, F_2)$, since the formula $\langle\!|\mathsf{b}\, x|\!\rangle\varphi$ has alternation depth 1 and so the maximum priority is $\mathsf{ad}(\langle\!|\mathsf{b}\, x|\!\rangle\varphi) + 1 = 2$. Among the states in Figure 6.2 only two have a priority, which are $q_2 \in F_2$ having priority 2 and $q_4 \in F_1$ having priority 1. The two curly lines

represent transitions that "appear" as an effect of the quotient operation. For instance, in the infinite NPA, state q_6 would have had a single transition to state $(\{\mathsf{b}^0, \mathsf{b}^1\}, \rho[x \mapsto \mathsf{b}^0, y \mapsto \mathsf{b}^1], X(y))$. Since the latter is future equivalent to q_2, in the quotient it is merged with q_2 and the transition from q_6 instead loops back to the corresponding equivalence class (represented by q_2). From q_7 there is also a transition, represented by a dashed line, to state $(\{\mathsf{b}^0, \mathsf{c}\}, \rho[x \mapsto \mathsf{b}^0, z \mapsto \mathsf{c}], Y(x))$. This state is not future equivalent to any of the previous ones, and it would lead to the rest of the reachable states of the automaton, not shown in the figure. However, the states displayed are already sufficient to prove that the language of the NPA is non-empty. Indeed, the sequence of states $(q_0, q_1, (q_2, q_3, q_4, q_5, q_6)^\infty)$, where the loop $q_2, q_3, q_4, q_5, q_6, q_2$ repeats indefinitely, represents an accepting run. In fact the state with maximal priority repeating infinitely often is q_2 and its priority is even.

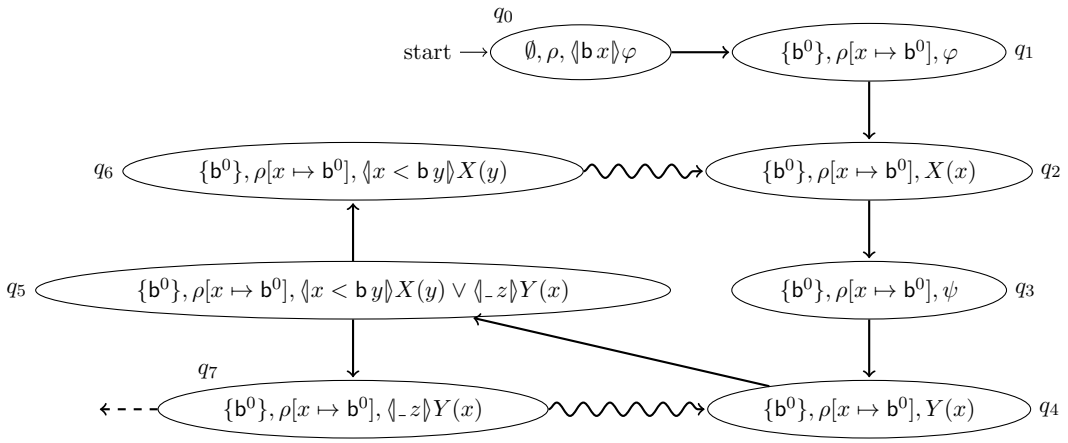


Figure 6.2: Quotient automaton obtained from the infinite automaton $\mathcal{A}_{\mathcal{E}_3, \langle\!|\mathsf{b}\, x|\!\rangle\varphi}$ via future equivalence.

We already hinted at the similarity between tableau rules and transitions in the automaton associated with a PES and a formula. We next formalise this relation by showing that future equivalence can be used to prune runs of the automaton $\mathcal{A}_{\mathcal{E},\varphi}$ in a way that a suitably chosen accepting run, after pruning, will correspond to a successful tableau.

Definition 6.7 (pruned k-tree)**.** *Let $\mathcal{E}$ be a PES, let φ be a closed formula of $\mathcal{L}_{hp}$, and let r be a run of the NPA $\mathcal{A}_{\mathcal{E},\varphi}$ on a k-tree $\mathcal{T}$. Given a path $p = (u_0, u_1, \ldots)$ in $\mathcal{T}$ we call a node u_j a* repetition *if one of the following conditions holds:*

1. *the formula in $r(u_j)$ is of the kind T, F, or $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\psi'$, $\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\psi'$ with $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C, \rho) = \emptyset$ (hence starting from u_j the path consists of the repetition of the state $r(u_j)$);*
2. *the formula in $r(u_j)$ is a proposition $X(\boldsymbol{x})$ and there is $i < j$ such that $r(u_i) \approx_f r(u_j)$ and X is not introduced between u_i and u_j.*

In case (1) we let $\Omega(u_j) = u_j$ while in case (2) we let $\Omega(u_j) = u_i$ where $i < j$ is the minimal index such that u_i satisfies the condition and we call $\Omega(u_j)$ the repetition witness *for u_j. If $r(u_j) \in F_l$ for some l, we say that u_j is a repetition of priority l. The* pruned run $\mathcal{T}^{(r)}$ *is the largest subtree of r where repetitions have no successor.*

The pruned run is obtained by cutting each path at the first repetition, hence in the pruned run each leaf has a repetition witness.

An adaptation of the results developed for tableaux in Subsection 5.3.1 allows us to prove that the pruning of a run is always finite.

We first observe some properties of paths in automata in relation to fixpoint formulae.

Lemma 6.1 (fixpoint introduction)**.** *Let $\mathcal{E}$ be a PES, let φ be a closed formula in $\mathcal{L}_{hp}$ and let $\mathcal{A}_{\mathcal{E},\varphi}$ be the corresponding automaton. For any path p in $\mathcal{A}_{\mathcal{E},\varphi}$ and state q with formula $X(\boldsymbol{x})$*

1. *if there exists a state q' after q, with formula $Y(\boldsymbol{y})$ and Y is not introduced between q and q', then $X \sqsubset_d^* Y$;*
2. *if there exists a state q' after q that introduces X, then there is a state between q and q' with formula $Y(\boldsymbol{x})$ with $X \sqsubset_d^* Y$;*

Proof. Straightforward adaptation of the proof of Lemma 5.5. □

It is easy to see that every infinite path in an NPA $\mathcal{A}_{\mathcal{E},\varphi}$, after a certain point, either loops over just one trivial state, or goes through infinitely many unfoldings and thus contains infinitely many occurrences of some propositions. In the latter case, using the lemma above, we are able to prove that among those propositions there must be one which is never introduced along the path and it has maximum alternation depth.

Lemma 6.2 (properties of infinite paths)**.** *Let $\mathcal{E}$ be a PES, let φ be a closed formula in $\mathcal{L}_{hp}$ and let $\mathcal{A}_{\mathcal{E},\varphi}$ be the corresponding NPA. Each infinite path in $\mathcal{A}_{\mathcal{E},\varphi}$ has a suffix $p = (q_0, q_1, \ldots)$ such that one the following properties holds:*

1. *for all j, $q_j = (C, \rho, \psi)$, with ψ of the kind T, F, or $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\psi'$, $\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \psi'$ with $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C, \rho) = \emptyset$;*
2. *the set of abstract propositions that occur infinitely often has a $\sqsubset_d^*$-greatest element X (which thus has the greatest alternation depth) and X is never introduced in p.*

Proof. Consider an infinite path in the automaton $\mathcal{A}_{\mathcal{E},\varphi}$. If from some point on $q_j = (C, \rho, \psi)$, where ψ of the kind T, F $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\psi'$ or $\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \psi'$ with $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C, \rho) = \emptyset$, we are done.

Otherwise, inspecting Definition 6.5, we see that the remaining transitions reduce the size of the formula associated with the state, the only exceptions being the unfolding transition. Therefore infinitely many transitions must be unfoldings, meaning that there are infinitely many states with a proposition as formula.

Consider a suffix $p = (q_0, q_1, \ldots)$ of the path such that all propositions occur infinitely often and let X one of these proposition that is $\sqsubset_d^*$-maximal. Consider a suffix $p' = (q_\ell, q_{\ell+1}, \ldots)$ starting with a state q_ℓ having an occurrence of X as formula. Reasoning exactly as in Lemma 5.5 we can deduce that X is never introduced in p'.

Moreover, X is a $\sqsubset_d^*$-maximum. In fact, let Y be another maximal abstract proposition occurring infinitely often in p'. Then we can consider states $q_i = (C_i, \rho_i, X(\boldsymbol{x}_i))$, $q_j = (C_j, \rho_j, Y(\boldsymbol{y}_j))$ and $q_i = (C_k, \rho_k, X(\boldsymbol{x}_k))$. If Y is not introduced between q_i and q_j then, by Lemma 6.1(1), $X \sqsubset_d^* Y$, hence, by maximality, $X = Y$. Similarly, if X is not introduced between q_j and q_k then $Y \sqsubset_d^* X$, hence, by maximality, $X = Y$. Otherwise, it is easy to see that if ψ_X and ψ_Y are the fixpoint formulae binding X and Y, respectively, ψ_Y should be a strict subformula of ψ_X that, in turn, should be a strict subformula of ψ_Y, leading to a contradiction. □

Now we show that every run becomes finite after being pruned. The proof is similar to that of the finiteness of tableaux for strongly regular PESs (Theorem 5.1). Indeed, we exploit again strong regularity, which, accompanied by the result above, implies that, even in the non-trivial cases, every infinite path must contain a repetition (in the sense of Definition 6.7).

Lemma 6.3 (pruned runs are finite)**.** *Let $\mathcal{E}$ be a strongly regular PES, let φ be a closed formula of $\mathcal{L}_{hp}$, and let $\mathcal{A}_{\mathcal{E},\varphi}$ be the corresponding NPA. For any run r of $\mathcal{A}_{\mathcal{E},\varphi}$ on a k-tree $\mathcal{T}$, the corresponding pruned run $\mathcal{T}^{(r)}$ is finite.*

Proof. We proceed by contradiction. We assume that $\mathcal{T}^{(r)}$ is infinite and we prove that it includes a repetition, in contrast with its definition. Since $\mathcal{T}^{(r)}$ is boundedly branching, by Lemma 5.6, there is an infinite path $p = (u_0, u_1, \ldots)$ in $\mathcal{T}^{(r)}$. Hence, by Lemma 6.2, there is a suffix $p' = (u_h, u_{h+1}, \ldots)$ such that one of the following properties holds and in both cases we conclude.

1. $r(u_i) = q = (C, \rho, \psi)$ for all $i \geq h$, with ψ is of the kind T, F, or $\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \psi'$, $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\psi'$ with $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C, \rho) = \emptyset$. Hence u_i is a repetition.

2. There is an abstract proposition X that occurs infinitely often and X is never introduced in p'. Since X occurs infinitely often in p' and, by Proposition 6.1, the equivalence $\approx_f$ is of finite index, there are two nodes u_i, u_j in p' with formulae $X(\boldsymbol{x}_i)$ and $X(\boldsymbol{x}_j)$, respectively, such that $r(u_i) \approx_f r(u_j)$. Therefore u_j is a repetition.

□

The results we have are already enough to show that any pruned run corresponds to a tableau. The next step would be to prove that accepting runs correspond to successful tableaux. However, this does not work directly: by pruning an accepting run, we could still get something that does not correspond to a successful tableau. The crux is that,

even though a path is accepting, and thus it includes repetitions over states of maximum priority which is even, it could also include, at the beginning, some repetitions over least fixpoints. In this case, the pruned run will have the corresponding states, with odd priority, at some leaves. For instance, consider the NPA in Figure 6.2. The run corresponding to the sequence of states $(q_0, q_1, q_2, q_3, q_4, q_5, q_7, q_4, q_5, q_6, (q_2, q_3, q_4, q_5, q_6)^\infty)$ is accepting, since the state q_2 occurs infinitely often. However, the first repetition along this run is q_4. Thus the pruned run would be the subsequence $(q_0, q_1, q_2, q_3, q_4, q_5, q_7, q_4)$, and so it would terminate in q_4 which has odd priority.

In order to prove that whenever there is an accepting run, there is one without "useless" repetitions, we formalise the intuition that some repetitions, like the one above, are avoidable.

Definition 6.8 (noisy repetition)**.** *Let $\mathcal{E}$ be a PES, let φ be a closed formula of $\mathcal{L}_{hp}$ and let $\mathcal{A}_{\mathcal{E},\varphi}$ be the corresponding NPA. Let r be an accepting run on a k-tree $\mathcal{T}$. A repetition u in r is called* noisy *if it has odd or no priority and no ancestor u' of u is a repetition of even priority.*

We show that noisy repetitions can be removed still getting a valid run. We first observe that in an accepting run, each infinite path p includes a repetition over a state whose priority is $\mathcal{F}(p)$ (hence even, see Definition 6.2).

Lemma 6.4 (maximal priority repetitions)**.** *Let $\mathcal{E}$ be a PES, let φ be a closed formula of $\mathcal{L}_{hp}$ and let r be an accepting run of the NPA $\mathcal{A}_{\mathcal{E},\varphi}$ on a k-tree $\mathcal{T}$. For each infinite accepting path $p = (u_0, u_1, \ldots)$ in r there exists a repetition u_i of priority $\mathcal{F}(p)$.*

Proof. Let $p = (u_0, u_1, \ldots)$ be an infinite accepting path in r. By Lemma 6.2, there is a suffix $p' = (u_h, u_{h+1}, \ldots)$ of p such that one of the following properties hold.

1. $r(u_i) = q = (C, \rho, \psi)$ for all $i \geq h$, with ψ the kind T, F, or $\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\psi'$, $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\psi'$ with $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C, \rho) = \emptyset$. The fact that the run and thus the path are accepting, reduces the possible shapes of ψ to T and $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\psi'$ with $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C, \rho) = \emptyset$. Hence u_i is a repetition. Moreover, $\mathcal{F}(q) = 0$ and since, $r(u_h) = r(u_i) = q \in F_0$ for $h \geq i$, this is the only priority repeating infinitely often, hence the largest.

2. The set of abstract proposition that occurs infinitely often in p' has a $\sqsubset_d^*$-largest element X (which thus has the largest alternation depth) and X is never introduced in p'. Since X occurs infinitely often in p' and, by Proposition 6.1, the equivalence $\approx_f$ is of finite index, there are two nodes u_i, u_j in p' with formulae $X(\boldsymbol{x}_i)$ and $X(\boldsymbol{x}_j)$, respectively, such that $r(u_i) \approx_f r(u_j)$. We next observe that $r(u_i), r(u_j) \in F_l$ with $l = \mathcal{F}(p)$. In fact, recall that $\mathsf{ad}(X)$ is maximal among the propositions occurring infinitely often. Thus if there were $l' > l$ such that $r(u_i) \in F_{l'}$ infinitely often, there should be some Y such that either

- Y is quantified in a ν-subformula and $\mathsf{ad}(Y) > \mathsf{ad}(X)$, contradicting the maximality of $\mathsf{ad}(X)$;
- Y is quantified in a μ-subformula and $\mathsf{ad}(Y) \geq \mathsf{ad}(X)$, contradicting the fact that the run r is accepting.

Summing up, $r(u_i) \approx_f r(u_j)$, state $r(u_i)$ has a proposition $X(\boldsymbol{x}_i)$ as a formula and X is not introduced between u_i and u_j, hence u_j is the desired repetition.

□

The above lemma implies that an accepting run r over a k-tree $\mathcal{T}$ has a finite number of noisy repetitions. In fact, it is immediate to see that each path in the run has a finite number of noisy repetitions, since they must precede the first repetition of priority $\mathcal{F}(p)$. We conclude by the fact that $\mathcal{T}$ has branching bounded by k.

We can now show that if an automaton has an accepting run, then it has an accepting run without noisy repetitions.

Lemma 6.5 (avoiding noisy repetitions)**.** *Let $\mathcal{E}$ be a PES and let φ be a closed formula of $\mathcal{L}_{hp}$. If $L(\mathcal{A}_{\mathcal{E},\varphi}) \neq \emptyset$ then $\mathcal{A}_{\mathcal{E},\varphi}$ has an accepting run r without noisy repetitions.*

Proof. Assume by contradiction that $L(\mathcal{A}_{\mathcal{E},\varphi}) \neq \emptyset$ but all accepting runs contain noisy repetitions. Let r be an accepting run on a k-tree $\mathcal{T}$ that has a minimal number of repetitions (by Lemma 6.4 this is finite, as already observed).

Let u be a noisy repetition in r. Since the run is accepting, the only possibility is that the formula of $r(u)$ is a proposition, say $X(\boldsymbol{x})$ quantified in a least fixpoint. Let $u' = \Omega(u)$, i.e., u' is the closest ancestor of u such that $r(u) \approx_f r(u')$ and X is not introduced between u' and u, and let $\mathcal{T}_u$ and $\mathcal{T}_{u'}$ be the subtrees of $\mathcal{T}$ rooted in u and u', respectively.

Since $r(u) \approx_f r(u')$ and $\approx_f$ is an NPA bisimulation (see Proposition 6.1) we can replace $\mathcal{T}_{u'}$ by $\mathcal{T}_u$ and get a new valid run r'. It is accepting since all paths in r' are obtained from paths in r by removing finitely many nodes.

We argue r' has strictly fewer repetitions than r, contradicting its minimality. In fact, u is no longer a repetition. Moreover, no new noisy repetitions are crated. In fact, in general, nodes that were not repetitions cannot become repetitions afterwards, still we could doubt that repetitions that were not noisy become so. For repetitions that were not in $\mathcal{T}_{u'}$, the ancestors are unchanged, hence they are noisy after the transformation if and only if they were before. Repetitions in $\mathcal{T}_{u'}$ but not in $\mathcal{T}_u$ are removed. Let us thus focus on repetitions in $\mathcal{T}_u$. Let v be one of such repetitions. Assume that u has odd priority and it is not noisy due to the presence of an ancestor w that is a repetition of even priority. Clearly, w cannot be an ancestor of u, otherwise u would not be noisy. Hence it must be between u and v. Still, we could think that after the transformation u could become noisy because $w' = \Omega(w)$ is between u' and u, hence it is removed and w ceases to be a repetition. However, if this were the case let $Y(\boldsymbol{y})$ and $Y(\boldsymbol{y}')$ be the formulae associated with w and w', respectively.

Since X is not introduced between u' and u, hence between w' and u, by Lemma 5.5(1), $Y \sqsubset_d^* X$. Since Y is not introduced between $w' = \Omega(w)$ and w, hence between u and w, again by Lemma 5.5(1), $X \sqsubset_d^* Y$. Therefore $X = Y$, but this is absurd since X is quantified in a least fixpoint and Y is quantified in a greatest fixpoint. □

We thus reach the desired conclusion, i.e., when an automaton has a non-empty language it has an accepting run that, once pruned, has all leaves with even priority.

Lemma 6.6 (pruned run with even leaves)**.** *Let $\mathcal{E}$ be a PES and let φ be a closed formula of $\mathcal{L}_{hp}$. If $L(\mathcal{A}_{\mathcal{E},\varphi}) \neq \emptyset$ then $\mathcal{A}_{\mathcal{E},\varphi}$ has an accepting run r on a k-tree $\mathcal{T}$ such that in $\mathcal{T}^{(r)}$ all leaves have even priority.*

Proof. By Lemma 6.5 we know that $\mathcal{A}_{\mathcal{E},\varphi}$ has an accepting run r on a k-tree $\mathcal{T}$ without noisy repetitions. Since by Definition 6.7 all leaves in $\mathcal{T}^{(r)}$ are repetitions and there are no repetitions among their ancestors, if a leaf had no or odd priority it would be noisy by Definition 6.8, contradicting the previous fact. Therefore, all leaves in $\mathcal{T}^{(r)}$ must have even priority. □

Using the above lemma, it is easy to prove that if $\mathcal{A}_{\mathcal{E},\varphi}$ has a non-empty language, then $\mathcal{E}$ satisfies φ. The proof relies on the fact that an accepting run for $\mathcal{A}_{\mathcal{E},\varphi}$ whose pruning has all leaves with even priority, can be easily transformed into a successful tableau for φ, so that Lemma 5.2 allows us to conclude.

Lemma 6.7 (non-emptiness implies satisfaction)**.** *Let $\mathcal{E}$ be a strongly regular PES and let $\check{\varphi}$ be a closed formula. If $L(\mathcal{A}_{\mathcal{E},\check{\varphi}}) \neq \emptyset$ then $\mathcal{E}$ satisfies $\check{\varphi}$.*

Proof. Let $\mathcal{E}$ be a strongly regular PES, let $\check{\varphi}$ in $\mathcal{L}_{hp}$ be a closed formula and and let $\mathcal{A}_{\mathcal{E},\check{\varphi}}$ be the corresponding automaton. Assume that $L(\mathcal{A}_{\mathcal{E},\check{\varphi}}) \neq \emptyset$. By Lemma 6.6, for $\mathcal{A}_{\mathcal{E},\check{\varphi}}$ there exists an accepting run r on a k-tree $\mathcal{T}$, such that all leaves of $\mathcal{T}^{(r)}$ have even priority. We show that $\mathcal{T}^{(r)}$ can be easily transformed into a successful tableau for the sequent $\emptyset, \check{\rho}, \emptyset \models \check{\varphi}$.

First, all nodes of $\mathcal{T}^{(r)}$ can be labelled with sequents as follows. Recall that r maps each to node u to a state $r(u) = (C, \rho, \psi)$. This is transformed into a sequent $C, \rho, \Delta_u \models \psi$, with the definition list Δ_u defined inductively as follows. For the root ϵ, we let $\Delta_u = \emptyset$. For non-root nodes ui, if $r(u) = (C, \rho, \psi)$ with $\psi = (\eta X(\boldsymbol{x}).\psi')(\boldsymbol{y})$, i.e., $r(u)$ introduces X, then $\Delta_{ui} = \Delta_u[X(\boldsymbol{x}) \mapsto \eta X(\boldsymbol{x}).\psi']$. In all other cases, $\Delta_{ui} = \Delta_u$.

Since the leaves l of $\mathcal{T}^{(r)}$ have even priority, the corresponding formulae are either

1. T or $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\psi'$ with $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C, \rho) = \emptyset$ or
2. $X(\boldsymbol{x})$ with X bound in a ν-subformula.

In the first case, by the definition of the transition relation and of pruned k-tree, the predecessor u of the leaf l is such that $r(u) = r(l)$ and $l = u1$ is its only successor. Call τ the tree obtained by removing such leaves.

Inspecting the automaton transitions and the tableau rules, it is immediate to realise that the sequents labelling each internal node of τ and its successors are the premise and the conclusions, respectively, of a tableau rule. The stop condition condition γ (see Definition 5.6) is not satisfied by internal nodes since they are not repetitions. Moreover, no tableau rule applies to the sequents labelling the leaves of τ. This is clearly the case for leaves in item (1) above. For the leaves in item (2), the rule (Unf_ν) cannot be applied since the stop condition γ is guaranteed to hold by Definition 6.7.

Therefore τ is a tableau, and it is clearly successful since, as already observed, every leaf is as required in Definition 5.9. □

Conversely, given a run r whose pruning has all leaves with even priority, it could still be the case that r is not accepting, because it does not take advantage of the possibility of cycling over the leaves. For instance, consider again the NPA in Figure 6.2. The run corresponding to the sequence of states $(q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_2, q_3, (q_4, q_5, q_7)^\infty)$ is not accepting, since the only state with a priority that occurs infinitely often is q_4 and, as already observed, q_4 has odd priority. However, the first repetition along this run is q_2 and thus the pruned run would be the subsequence $(q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_2)$, terminating in the state q_2 which has even priority. Indeed, recognising the presence of the loop $q_2, q_3, q_4, q_5, q_6, q_2$, one can construct the run $(q_0, q_1, (q_2, q_3, q_4, q_5, q_6)^\infty)$ which is accepting.

In general, we can prove that if there exists a run whose pruning has all leaves with even priority then there exists an accepting run.

Lemma 6.8 (accepting run for pruned runs)**.** *Let $\mathcal{E}$ be a PES and let φ be a closed formula of $\mathcal{L}_{hp}$. If there exists a run r of the NPA $\mathcal{A}_{\mathcal{E},\varphi}$ on a k-tree $\mathcal{T}$ such that in $\mathcal{T}^{(r)}$ all leaves have even priority then there exists also an accepting run of $\mathcal{A}_{\mathcal{E},\varphi}$.*

Proof. Let r be a run such that all leaves in $\mathcal{T}^{(r)}$ have even priority. Consider the directed graph obtained by adding to every leaf w of $\mathcal{T}^{(r)}$ a "back arc" in the following way. If $r(w) = (C, \rho, \psi)$ where ψ is T or $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \psi'$, then add a self-loop. Otherwise, ψ must be a proposition $Z(\boldsymbol{z})$ quantified in a ν-subformula, since all leaves have even priority. In this case the arc goes from l to the successor of its repetition witness $\Omega(l)$.

Unfold such a directed graph, starting from the root ϵ of $\mathcal{T}^{(r)}$, thus obtaining a k-tree $\mathcal{T}'$ (different from $\mathcal{T}$, in general). More formally, $\mathcal{T}'$ is the tree whose nodes are finite paths $v = (u_0, u_1, \ldots, u_n)$ in the graph and where each node $v = (u_0, u_1, \ldots, u_n)$ has $v' = (u_0, u_1, \ldots, u_{n-1})$ as parent. Clearly, all complete paths in $\mathcal{T}'$ are infinite, since every node of the graph has a successor. Moreover, paths in $\mathcal{T}'$ will be sometimes confused with the corresponding paths in the graph.

We next show that r can be used to build an accepting run r' of the quotient automaton $\mathcal{A}_{\mathcal{E},\varphi/\approx_f}$ over $\mathcal{T}'$. For each node $v = (u_0, u_1, \ldots, u_n)$, define $r'(v) = [r(u_n)]_{\approx_f}$. Since for each node u of the original run, $r(u) \approx_f r(\Omega(u))$, it is not difficult to realise that r' is a well-defined run of $\mathcal{A}_{\mathcal{E},\varphi/\approx_f}$ on $\mathcal{T}'$.

We argue that the run r' is accepting, that is, for any complete path $p = (v_0, v_1, \ldots)$ in r', if we consider the set of priorities that repeat infinitely often

$$I(p) = \{l \in [0, h] \mid \{i \mid r'(v_i) \in F_l\} \text{ is infinite}\},$$

then $I(P)$ is non-empty and its maximum is even.

Since p is infinite, $\mathcal{T}^{(r)}$ is finite (by Lemma 6.3), and the back arcs have been added only to the leaves, by construction some leaf w of $\mathcal{T}^{(r)}$ must repeat infinitely many times in p (or, more precisely, $v_i = v_{i-1}w$ infinitely often). Since w has even priority, there are two possibilities

- $r(w)$ is T or $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \psi'$ with $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a} z}(C, \rho) = \emptyset$, or
- $r(w)$ is a proposition quantified in a ν-subformula.

In the first case, from some point on, $v_i = v_{i-1}w$ and $r(w)$, hence $r'(v_i)$, have priority 0 and this is the only priority repeating infinitely often. Hence we are done.

In the second case, we know that $I(p) \neq \emptyset$. In order to conclude that the maximum is even we proceed as follows. We show that for any proposition X quantified in a μ-subformula such that X appears infinitely many times along p, there is a proposition Y quantified in a ν-subformula such that $\mathsf{ad}(Y) > \mathsf{ad}(X)$ and Y appears infinitely many times along p. Recalling how the priority is defined based on the alternation depth, it is easy to see that this implies that for any node of odd priority there is descendant node of larger even priority, whence the desired property.

Thus assume that the formula $X(\boldsymbol{x})$, with X quantified in a μ-subformula, occurs infinitely often in p. Since $\mathcal{T}^{(r)}$ is finite, this implies that there is a node u of $\mathcal{T}^{(r)}$ such that the formula of $r(u)$ is $X(\boldsymbol{x})$ and u is traversed infinitely often in p. Moreover, since all leaves of $\mathcal{T}^{(r)}$ have even priority, u must be an inner node.

Now, let v_i, v_j be two consecutive occurrences of u in p, i.e., $v_i = v_{i-1}u$ and $v_j = v_{j-1}u$ with $i < j$. Note that since u is an inner node of $\mathcal{T}^{(r)}$ and there is a path from v_i to v_j, i.e., from u to u in the graph, there must be an index k with $i < k < j$ such that u_k corresponds to a leaf w of $\mathcal{T}^{(r)}$, and its repetition witness $\Omega(w)$ is above u in $\mathcal{T}^{(r)}$. Assume that k is the smallest such index. Then w is a descendant of u in $\mathcal{T}^{(r)}$ (otherwise there should be an "earlier" leaf of index $u_{k'}$, with $i < k' < k$ whose witness is a common ancestor of u and w, contradicting the minimality of k). Therefore u is between $\Omega(w)$ and w in $\mathcal{T}^{(r)}$. Let $Y(\boldsymbol{y})$ be the formula in $r(w)$. Note that Y is quantified in a ν-subformula since v is a leaf. By definition of repetition witness, Y is not introduced $\Omega(w)$ and w. Thus, by Lemma 6.1(1) we have that $X \sqsubset_d^* Y$. Moreover $X \neq Y$ since X is quantified in a μ-subformula and Y in a ν-subformula. Hence, by definition of alternation depth, $\mathsf{ad}(X) < \mathsf{ad}(Y)$.

We proved that between any two consecutive occurrences of u there is a node with a formula $Y(\boldsymbol{y})$, quantified in a ν-subformula, such that $\mathsf{ad}(X) < \mathsf{ad}(Y)$. Since u occurs infinitely often in p, the variable Y will be the same in infinitely many cases. This is what we aimed at.

Summing up, every complete path of r' is accepting and thus the run r' on the k-tree $\mathcal{T}'$ is an accepting run for $\mathcal{A}_{\mathcal{E},\varphi/\approx_f}$ on $\mathcal{T}'$, thus $L(\mathcal{A}_{\mathcal{E},\varphi/\approx_f}) \neq \emptyset$. Since $\approx_f$ is a bisimulation on $\mathcal{A}_{\mathcal{E},\varphi}$, by Theorem 6.1, $L(\mathcal{A}_{\mathcal{E},\varphi/\approx_f}) = L(\mathcal{A}_{\mathcal{E},\varphi})$, hence we conclude. □

The above result allows us to conclude that if $\mathcal{E}$ satisfies φ, then $\mathcal{A}_{\mathcal{E},\varphi}$ has a non-empty language. Again we rely on the results proven for tableaux. By Lemma 5.13, whenever $\mathcal{E}$ satisfies φ there is a successful tableau for φ. The proof then shows that a successful tableau for φ can be viewed as the pruning of a run where all leaves have even priority. By Lemma 6.8 this can be transformed into an accepting run for $\mathcal{A}_{\mathcal{E},\varphi}$.

Lemma 6.9 (satisfaction implies non-emptiness)**.** *Let $\mathcal{E}$ be a strongly regular PES and let $\check{\varphi}$ be a closed formula. If $\mathcal{E}$ satisfies $\check{\varphi}$ then $L(\mathcal{A}_{\mathcal{E},\check{\varphi}}) \neq \emptyset$.*

Proof. Let $q_0 = (\emptyset, \check{\rho}, \check{\varphi})$ be the initial state of $\mathcal{A}_{\mathcal{E},\check{\varphi}}$. Let τ be a successful tableau for the formula $\check{\varphi}$, which is guaranteed to exist by Lemma 5.13.

The tableau τ can be transformed into a run r of $\mathcal{A}_{\mathcal{E},\check{\varphi}}$ in the following way. A prefix of r corresponds exactly to the tableau τ: each sequent $C, \rho, \Delta \models \varphi$ is transformed into a state (C, ρ, φ). In particular, the root of τ, labelled by the sequent $\emptyset, \check{\rho}, \emptyset \models \check{\varphi}$, corresponds to the initial state $q_0 = (\emptyset, \check{\rho}, \check{\varphi})$. By an inspection of the tableau rules and the automaton transitions it is immediate to realise that this is indeed an incomplete run of $\mathcal{A}_{\mathcal{E},\check{\varphi}}$ starting from the initial state q_0.

Now, since by definition of $\mathcal{A}_{\mathcal{E},\check{\varphi}}$, every state has a successor, clearly r can be extended to a full correct run r of $\mathcal{A}_{\mathcal{E},\check{\varphi}}$ on some k-tree $\mathcal{T}$.

Note that r might not be accepting. Still, the prefix of r, corresponding to τ does not contain repetitions except for those associated with the leaves of τ (in particular the stop condition γ never holds in the inner nodes of τ). Since τ is successful (Definition 5.9) leaves w are labelled by proposition quantified in a ν-subformula, T, or $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \psi$ with $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C, \rho) = \emptyset$. In the last two cases, a path in the run r reaching w will cycle on indefinitely on the same formula. In the first case, a path in r through w could possibly include repetitions after w, but these will not appear in the pruned run $\mathcal{T}^{(r)}$ by Definition 6.7 thanks to the presence of w being the first repetition. Therefore, by definition of the acceptance condition of $\mathcal{A}_{\mathcal{E},\check{\varphi}}$, all leaves in the pruned run $\mathcal{T}^{(r)}$ have even priority and thus, by Lemma 6.8, there exists a run r' of $\mathcal{A}_{\mathcal{E},\check{\varphi}}$ which is accepting. And so $\mathcal{L}(\mathcal{A}_{\mathcal{E},\check{\varphi}}) \neq \emptyset$. □

Joining Lemmata 6.7 and 6.9, we have that the model-checking problem of a formula φ over a strongly regular PES $\mathcal{E}$ reduces to the non-emptiness of the language of the automaton $\mathcal{A}_{\mathcal{E},\varphi}$.

Theorem 6.2 (model-checking via non-emptiness)**.** *Let $\mathcal{E}$ be a strongly regular PES and let φ be a closed formula of $\mathcal{L}_{hp}$. Then $L(\mathcal{A}_{\mathcal{E},\varphi}) \neq \emptyset$ if and only if $\mathcal{E}$ satisfies φ.*

Proof. Corollary of Lemmata 6.7 and 6.9. □

The above result, combined with a suitable bisimulation equivalence $\equiv$ of finite index, can be exploited to obtain an effective procedure for checking the satisfaction of a formula. In fact, given a bisimulation $\equiv$ over $\mathcal{A}_{\mathcal{E},\varphi}$ of finite index, the quotient automaton $\mathcal{A}_{\mathcal{E},\varphi/\equiv}$ is finite and, exploiting Theorems 6.2 and 6.1, we can verify whether $\mathcal{E} \models \varphi$ by checking the emptiness of the language accepted by $\mathcal{A}_{\mathcal{E},\varphi/\equiv}$. Clearly a concrete algorithm will not first generate the infinite state NPA and then take the quotient, but it rather performs the quotient on the fly: whenever a new state would be equivalent to one already generated, the transition loops back to the already existing state.

When $\mathcal{E}$ is strongly regular, a reference bisimulation equivalence of finite index on $\mathcal{A}_{\mathcal{E},\varphi}$ is future equivalence. An obstacle towards the use of the quotiented NPA for model checking purposes is the fact that the future equivalence could be hard to compute (or even undecidable). In order to make the construction effective we need a decidable bisimulation equivalence on the NPA and the effectiveness of the set of successors of a state. This is further discussed in the next section.

6.3 NPA for Petri nets

We show how the abstract automata-based model-checking approach outlined in the previous section can be instantiated on finite safe Petri nets by identifying a suitable effective bisimulation equivalence on the NPA.

We already saw that the PES associated with a finite safe Petri net is regular (Theorem 3.2). We next prove that it is also strongly regular and thus we can apply the theory developed so far for model-checking $\mathcal{L}_{hp}$ over finite safe Petri nets.

Let $\mathcal{N}$ be a safe Petri net. In Proposition 3.2 we observed that the residual of the PES $\mathcal{E}(\mathcal{N})$ after a configuration $C \in \mathcal{C}(\mathcal{E}(\mathcal{N}))$ is uniquely determined by the marking produced by C. This correspondence can be further extended to pointed configurations by considering markings which additionally record, for the events of interest in the past, the places in the marking which are caused by such events. This motivates the definition below.

Definition 6.9 (pointed marking)**.** *Let $\mathcal{N}$ be a safe Petri net. Given a set V, a V-*pointed marking *is a pair $\langle M, r\rangle$ where $M \subseteq P$ is a safe marking and $r : V \to \mathbf{2}^M$.*

A V-pointed configuration $\langle C, \zeta\rangle$ of $\mathcal{E}(\mathcal{N})$ naturally induces a V-pointed marking denoted by $\mathsf{M}(\langle C, \zeta\rangle) = \langle \mathsf{M}(C), r\rangle$ where $r(x) = \{\pi_1(b) \mid b \in C^\circ \ \wedge \ \zeta(x) < b\}$. We next observe that pointed configurations producing the same pointed marking have isomorphic pointed residuals.

Proposition 6.2 (pointed markings vs configurations)**.** *Let $\mathcal{N}$ be a safe Petri net. Given a set V and two V-pointed configurations $\langle C_1, \zeta_1\rangle$, $\langle C_2, \zeta_2\rangle$ in $\mathcal{E}(\mathcal{N})$, if $\mathsf{M}(\langle C_1, \zeta_1\rangle) = \mathsf{M}(\langle C_2, \zeta_2\rangle)$ then $\langle C_1, \zeta_1\rangle \approx_r \langle C_2, \zeta_2\rangle$.*

Proof. Let $\mathcal{N}$ be a safe Petri net, consider its unfolding $\mathcal{U}(\mathcal{N})$ and two V-pointed configurations $\langle C_1, \zeta_1\rangle$, $\langle C_2, \zeta_2\rangle$ such that $\mathsf{M}(\langle C_1, \zeta_1\rangle) = \mathsf{M}(\langle C_2, \zeta_2\rangle)$.

First note that since, in particular, $\mathsf{M}(C_1) = \mathsf{M}(C_2)$, the residuals $\mathcal{U}(\mathcal{N})[C_1]$ and $\mathcal{U}(\mathcal{N})[C_2]$ are isomorphic. Let $\iota : \mathcal{U}(\mathcal{N})[C_1] \to \mathcal{U}(\mathcal{N})[C_2]$ be the corresponding net isomorphism. Its restriction to the underlying PESs, abusing the notation, is still denoted by ι and it establishes an isomorphism of the residuals $\mathcal{E}(\mathcal{N})[C_1]$ and $\mathcal{E}(\mathcal{N})[C_2]$. We next prove that ι is also an isomorphism of residuals of the pointed configurations $\langle C_1, g_1\rangle$ and $\langle C_2, g_2\rangle$, i.e., that for all $x \in V$ and let $e_1 \in \mathcal{E}(\mathcal{N})[C_1]$ we have $g_1(x) \leq e_1$ iff $g_2(x) \leq \iota(e_1)$.

Let $x \in V$ and let $e_1 \in \mathcal{E}(\mathcal{N})[C_1]$. If $g_1(x) \leq e_1$ then there is $b_1 \in C_1^\circ$ such that $g_1(x) \leq b_1 \leq e_1$. Thus $\pi_1(b_1) \in\in r_1(x) = r_2(x)$. Therefore, there is $b_2 \in C_2^\circ$ such that $\pi_1(b_2) = \pi_1(b_1)$ and $g_2(x) \leq b_2$.

Now, since the net is safe the isomorphism ι must map b_1 to b_2. Therefore $g_2(x) \leq b_2 = \iota(b_1) \leq \iota(e_1)$, where the last step is motivated by the fact that the isomorphism preserves causality. Hence $g_2(x) \leq \iota(e_1)$, as desired.

Conversely, if $g_2(x) \leq \iota(e_1)$ we can deduce that $g_1(x) \leq e_1$ in an analogous way. □

By the above result the PES associated with a finite safe Petri net is strongly regular. In fact, the number of residuals of V-pointed configurations, up to isomorphism, by Proposition 6.2, is smaller than the number of V-pointed markings, which is clearly finite since the net is finite and safe. Furthermore, the PES is regular by Theorem 3.2, hence it is boundedly branching. And so one can conclude by using Lemma 5.4.

Corollary 6.1 (strong regularity)**.** *Let $\mathcal{N}$ be a finite safe Petri net. Then the corresponding PES $\mathcal{E}(\mathcal{N})$ is strongly regular.*

In order to instantiate the model-checking framework to finite safe Petri nets, the idea is to take an equivalence over the infinite NPA that equates states whose (pointed) configurations induce the same pointed marking.

Definition 6.10 (pointed-marking equivalence on NPA)**.** *Let $\mathcal{N}$ be a finite safe Petri net and let φ be a closed formula of $\mathcal{L}_{hp}$. Two states q_1, q_2 in the NPA $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi}$ are* pointed-marking equivalent*, written $q_1 \approx_m q_2$, if $q_i = (C_i, \rho_i, \psi)$, $i \in \{1, 2\}$, for some $\psi \in \mathsf{sf}(\varphi)$ and $\mathsf{M}(\langle C_1, \rho_1|_{fv(\psi)}\rangle) = \mathsf{M}(\langle C_2, \rho_2|_{fv(\psi)}\rangle)$.*

Using Proposition 6.2 we can immediately prove that $\approx_m$ refines $\approx_f$. Moreover we can show that $\approx_m$ is a bisimulation in the sense of Definition 6.4.

Proposition 6.3 ($\approx_m$ is a bisimulation)**.** *Let $\mathcal{N}$ be a finite safe Petri net and let φ be a closed formula of $\mathcal{L}_{hp}$. The equivalence $\approx_m$ on the automaton $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi}$ is a bisimulation and it is of finite index.*

Proof. Let $q = (C, \rho, \psi), q' = (C', \rho', \psi') \in Q$ be states of $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi}$ such that $q \approx_m q'$. This means that $\mathsf{M}(\langle C_1, \rho_1|_{fv(\psi)}\rangle) = \mathsf{M}(\langle C_2, \rho_2|_{fv(\psi)}\rangle) = \langle M, r\rangle$.

Concerning condition (1) of Definition 6.4, the fact that for all $i \in [0, h]$, $q \in F_i$ iff $q' \in F_i$ immediately follows from Proposition 6.1, recalling that $\approx_m$ refines future equivalence $\approx_f$.

Let us focus on condition (2). Assume that $q \to (q_1, \ldots, q_n)$. In order to prove that $q' \to (q'_1, \ldots, q'_n)$ with $q_i \approx_m q'_i$ for $i \in [1, n]$ we distinguish various cases according to the shape of ψ.

- $\psi = \mathsf{T}$ or $\psi = \mathsf{F}$
  Trivial, since the only transitions of q and q' are $q \to (q)$ and $q' \to (q')$.

- $\psi = \psi_1 \wedge \psi_2$
  The only transition of q is $q \to (q_1, q_2)$ where $q_i = (C, \rho, \psi_i)$, $i \in \{1, 2\}$. By construction also $q' \to (q'_1, q'_2)$ where $q'_i = (C', \rho', \psi_i)$, $i \in \{1, 2\}$. In order to conclude that $q_i \approx_m q_1 i$, for $i \in \{1, 2\}$, just observe that $fv(\psi_i) \subseteq fv(\psi)$ and thus such states are associated with the same pointed marking $\langle M, r|_{fv(\psi_i)} \rangle$.

- $\psi = \psi_1 \vee \psi_2$, $\psi = (\eta X(\boldsymbol{x}).\psi')(\boldsymbol{y})$, or $\psi = X(\boldsymbol{y})$
  These cases are analogous to the previous one.

- $\psi = \langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle \psi_1$
  in this case, if $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C, \rho) \neq \emptyset$ then $q \to (q_1)$ where $q_1 = (C_1, \rho_1, \psi_1)$ with $(C_1, \rho_1) \in \mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C, \rho)$, which means that $C \xrightarrow{\rho(\boldsymbol{x}), \overline{\rho(\boldsymbol{y})} < e}_a C_1$ and $\rho_1 = \rho[z \mapsto e]$.

  Let $t = \pi_1(e)$ be the transition in $\mathcal{N}$ corresponding to the event e of $\mathcal{U}(\mathcal{N})$. For each x in $\boldsymbol{x}$, since $\rho(x) < e$, there must be a condition $b_x \in {}^\bullet e$ such that $\rho(x) < b_x$ and thus, in the associated pointed marking $\langle M, r \rangle$, we have ${}^\bullet t \cap r(x) \neq \emptyset$. Dually, for each y in $\boldsymbol{y}$, since it is not the case that $\rho(y) < e$, for all conditions $b \in {}^\bullet e$ we have $\neg(\rho(y) < b)$ and thus ${}^\bullet t \cap r(y) = \emptyset$.

  Since, by hypothesis, q' is associated with the same pointed marking $\langle M, r \rangle$ as q, there is an event e' such that $\pi_1(e') = t$ and $e' \in en(C')$. Moreover, for all variables $w \in fv(\psi)$ we have $\rho'(w) < e'$ iff $\rho'(w) \cap {}^\bullet e' \neq \emptyset$ iff $r(w) \cap {}^\bullet t \neq \emptyset$. Therefore, by the considerations above, for each x in $\boldsymbol{x}$, $\rho'(x) < e'$ and for no y in $\boldsymbol{y}$, $\rho'(y) < e'$. This means that if we let $\rho'_1 = \rho'[z \mapsto e']$, then there exists $(C'_1, \rho'_1) \in \mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C', \rho')$ and thus $q' \to (q'_1)$ where $q'_1 = (C'_1, \rho'_1, \psi_1)$. Finally, $q_1 \approx_m q'_1$, i.e., they are associated with the same $fv(\psi_1)$-pointed marking $\langle M', r' \rangle$ where $M' = (M \setminus {}^\bullet t) \cup t^\bullet$. Moreover, $fv(\psi_1) \subseteq fv(\psi) \cup \{z\}$ and for each variable $w \in fv(\psi_1) \setminus \{z\}$ we have $r'(w) = (r(w) \cap M') \cup \{s \mid s \in t^\bullet \wedge\ r(w) \cap {}^\bullet t \neq \emptyset\}$ and $r'(z) = t^\bullet$.

  If instead $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C, \rho) = \emptyset$ then $q \to (q)$. From the fact that q and q' are associated with the same pointed marking, reasoning as in the previous case, we deduce that also $\mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C', \rho') = \emptyset$ and thus $q' \to (q')$. Hence we conclude

- $\psi = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!] \psi_1$
  Analogous to the previous case.

The fact that $\approx_m$ is of finite index immediately follows from the observation that the number of $fv(\psi)$-pointed markings of $\mathcal{N}$, where $\psi \in \mathsf{sf}(\varphi)$, is finite. In fact both $\mathsf{sf}(\varphi)$ and

$\mathcal{R}(\mathcal{N})$ are finite. Moreover, for each fixed subformula $\psi \in \mathsf{sf}(\varphi)$ and marking $M \in \mathcal{R}(\mathcal{N})$, the number of $fv(\psi)$-pointed markings $\langle M, r\rangle$, with $r : fv(\psi) \to \mathbf{2}^M$ is clearly finite. □

Relying on Propositions 6.2 and 6.3 we can provide an explicit construction of the quotient automaton $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi/\approx_m}$. We introduce a convenient notation for transitions between pointed markings. Given the tuples of variables $\boldsymbol{x}$, $\boldsymbol{y}$, a set V such that $\boldsymbol{x} \cup \boldsymbol{y} \subseteq V$ and a V-pointed marking $\langle M, r\rangle$, we write $\langle M, r\rangle \xrightarrow{\boldsymbol{x},\overline{\boldsymbol{y}}<t}_{\mathsf{a},z} \langle M', r'\rangle$ if $M[t\rangle M'$, $\lambda_N(t) = \mathsf{a}$, for all $x \in \boldsymbol{x}$ we have $r(x) \cap {}^\bullet t \neq \emptyset$, for all $y \in \boldsymbol{y}$ it holds $r(y) \cap {}^\bullet t = \emptyset$, and r' is defined by $r'(z) = t^\bullet$ and $r'(w) = (r(w) \cap M') \cup \{s \mid r(w) \cap {}^\bullet t \neq \emptyset \ \wedge \ s \in t^\bullet\}$, for $w \neq z$. In words, from the pointed marking $\langle M, r\rangle$, the transition t is fired and "pointed" by variable z. Transition t is required to consume tokens caused by $\boldsymbol{x}$ and not to consume tokens caused by $\boldsymbol{y}$, in order to be itself caused by $\boldsymbol{x}$ and independent from $\boldsymbol{y}$. After the firing, clearly, z causes $t^\bullet$ and variables which were causes of some $p \in {}^\bullet t$ become causes of the places in $t^\bullet$.

Construction 6.1 (quotient NPA)**.** *Let $\mathcal{N}$ be a finite safe Petri net and let φ be a closed formula of $\mathcal{L}_{hp}$. The quotient NPA $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi/\approx_m}$ is defined as follows. The set of states $Q = \{(M, r, \psi) \mid M \in \mathcal{R}(\mathcal{N}) \ \wedge \ r : fv(\psi) \to \mathbf{2}^M \ \wedge \ \psi \in \mathsf{sf}(\varphi)\}$. The initial state $q_0 = (M_0, \emptyset, \varphi)$. The transition relation is defined, for any state $q = (M, r, \psi) \in Q$, by:*

- *if $\psi = \mathsf{T}$ or $\psi = \mathsf{F}$, then $q \to (q)$*
- *if $\psi = \psi_1 \wedge \psi_2$, then $q \to (q_1, q_2)$ where $q_i = (M, r, \psi_i)$, $i \in \{1, 2\}$*
- *if $\psi = \psi_1 \vee \psi_2$, then $q \to (q_1)$ and $q \to (q_2)$ where $q_i = (M, r, \psi_i)$, $i \in \{1, 2\}$*
- *if $\psi = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\ z]\!]\psi'$, let $S = \{(M', r'|_{fv(\psi')}) \mid \langle M, r\rangle \xrightarrow{\boldsymbol{x},\overline{\boldsymbol{y}}<t}_{a,z} \langle M', r'\rangle\}$;*
  - *if $S = \{(M_1, r_1), \ldots, (M_n, r_n)\} \neq \emptyset$ then $q \to (q_1, \ldots, q_n)$ where $q_i = (M_i, r_i, \psi')$ for $i \in [1, n]$,*
  - *otherwise $q \to (q)$*
- *if $\psi = \langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\ z|\!\rangle\psi'$, let $S = \{(M', r'|_{fv(\psi')}) \mid \langle M, r\rangle \xrightarrow{\boldsymbol{x},\overline{\boldsymbol{y}}<t}_{a,z} \langle M', r'\rangle\}$;*
  - *if $S = \{(M_1, r_1), \ldots, (M_n, r_n)\} \neq \emptyset$ then $q \to (q_i)$ where $q_i = (M_i, r_i, \psi')$ for $i \in [1, n]$,*
  - *otherwise $q \to (q)$*
- *if $\psi = (\eta X(\boldsymbol{x}).\psi')(\boldsymbol{y})$, then $q \to (q')$ where $q' = (M, r, X(\boldsymbol{y}))$*
- *if $\psi = X(\boldsymbol{z})$ and $\psi' \in \mathsf{sf}(\varphi)$ is the unique subformula such that $\psi' = (\eta X(\boldsymbol{x}).\psi'')(\boldsymbol{y})$ then $q \to (q')$ where $q' = (M, r[\boldsymbol{x} \mapsto r(\boldsymbol{z})], \psi'')$.*

The acceptance condition is analogous to that in Definition 6.1.

The automaton $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi/\approx_m}$ is finite for all finite safe Petri nets $\mathcal{N}$. Thus, it can be used for model-checking a formula φ of $\mathcal{L}_{hp}$ over a finite safe Petri net by means of a language emptiness check. A prototypical implementation of the procedure is discussed in the next section.

6.4 A prototype tool

The algorithm for model-checking Petri nets outlined before is implemented in a prototype tool called TCWB (*True Concurrency Workbench*) [Pad], written in Haskell. The tool inputs a safe Petri net $\mathcal{N}$ and a closed formula φ of $\mathcal{L}_{hp}$ and outputs the truth value of the formula on the initial marking of $\mathcal{N}$. The truth of the formula is reduced to the non-emptiness of the language of the automaton $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi_{/\approx_m}}$ (Theorem 6.2). The algorithm builds the quotient NPA $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi_{/\approx_m}}$ "on demand", i.e., the states of the automaton are generated when they are explored in the search for an accepting run. A path is recognised as accepting when it includes a loop where a $\sqsubset_d^*$-maximal subformula (see Definition 4.12) is T, a $[\![\,]\!]$-subformula or a proposition quantified in a ν-subformula. In this way only a fragment of $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi_{/\approx_m}}$ relevant to decide the satisfaction of φ is actually built.

A loop is identified when a state is encountered that is pointed-marking equivalent to an ancestor state. According to Definition 6.10, in order to be pointed-marking equivalent two states must contain the very same subformula. It is easy to see that this requirement could be relaxed, e.g., by allowing different names for the free event variables (as it happens in Definition 6.6). In principle, a coarser pointed-marking equivalence could reduce the size of the quotient automaton and thus increase efficiency. However, in practice, the cost of checking whether formulae are the same up to variable renaming beats the advantages deriving from the reduction of the automaton size. The implementation adopts an intermediate approach, which tries to rename event variables only in the case of propositions. Moreover, states are kept in an ordered structure which allows for a binary search. Again, maintaining such structure has some cost, but since the queries vastly outnumber the updates this turns out to be convenient.

As a side remark, we tried a direct implementation of the tableau-based procedure. As anticipated, it resulted to have very poor performances mainly because of the repeated exploration of "equivalent" paths during the construction of a proof tree, which is avoided in the automata-theoretic procedure.

Given a net $\mathcal{N} = \langle P, T, F, M_0\rangle$ and a formula φ, the number of states in the quotient automaton $\mathcal{A}_{\mathcal{E}(\mathcal{N}),\varphi_{/\approx_m}}$ can be bounded as follows. Recall that a state consists of a triple (M, r, ψ) where $\psi \in \mathsf{sf}(\varphi)$, M is a reachable marking and $r : fv(\psi) \to \mathbf{2}^M$ is a function. This leads to an upper bound $O(|\mathsf{sf}(\varphi)| \cdot |\mathcal{R}(\mathcal{N})| \cdot \mathbf{2}^{|P|\cdot v})$, where $v = max\{|fv(\psi)| : \psi \in \mathsf{sf}(\varphi)\}$ is the largest number of event variables appearing free in a subformula of φ. In turn, since $|\mathcal{R}(\mathcal{N})| \leq \mathbf{2}^{|P|}$, this is bounded by $O(|\mathsf{sf}(\varphi)| \cdot \mathbf{2}^{|P|\cdot(v+1)})$. The size of the automaton is thus exponential in the size of the net and linear in the size of the formula. Moving from the interleaving fragment of the logic (where $v = 0$) to formulae capable of expressing true concurrency properties thus causes an exponential blow up. However, note that the worst case scenario requires all transitions to be related by causality and concurrency to all places in all possible ways, something that should be quite unlikely in practice. Indeed, despite the fact that the tool is very preliminary and more tweaks and optimisations could improve

its efficiency, for the practical tests we performed the execution time seems to be typically well below than the theoretical worst case upper bound.

We performed some simple tests. Despite the absence of another tool with the same purpose to compare with, they were useful to grasp some information. All tests were performed using Petri nets representing processes made of a number, indicated in the process name, of parallel copies of a cycler, which is a sequential loop always repeating the same four states.

- *Absence of deadlock.* This is a property that can be expressed in the interleaving fragment of $\mathcal{L}_{hp}$, i.e., the formula $Live = \nu X.(\langle\!\langle \_\, x \rangle\!\rangle \mathsf{T} \wedge [\![\_\, y]\!] X)$. Hence we can compare with a classical tool for model-checking formulae of the mu-calculus, i.e., Edinburgh Concurrency Workbench (CWB). The technique implemented in the CWB falls under a third approach, based on model-checking games. The algorithm generates a winning strategy for one of two players. The results are reported in Table 6.1. For each tool the CPU time is given in seconds, the symbol "-" indicates an execution time exceeding 1800 seconds. The CWB is faster, as expected since it is a well optimised tool and the greater expressiveness of $\mathcal{L}_{hp}$ requires the maintenance of more complex data structures. Still, the fact that the efficiency is comparable suggests that the overhead deriving by the need of setting up the data structures needed to deal with pointed markings is acceptably small.

- *Causal atomicity* [FM06] for a block labelled by a. We assume that every action of a process is either labelled by a or b. Hence the process has only one block, required to be atomic, labelled by a, everything else is outside the block and is labelled by b. Then, as mentioned in Subsection 4.3.3, causal atomicity can be expressed by the formula $Atom = \nu X.([\![\_\, w]\!] X \wedge [\![\mathsf{a}\, x]\!] \nu Y(x).([\![x < \mathsf{b}\, y]\!][\![y < \mathsf{a}\, z]\!] \mathsf{F} \wedge [\![\_\, w]\!] Y(x)))$. Note that all transitions in the cyclers are labelled by a. While this might seem strange for the task of interest, it has been chosen since it produces the worst case scenario. In fact, every pair of transitions must be checked for the absence of an atomicity violation since all transitions are in the same atomic block. In this case the "true concurrent" operators of $\mathcal{L}_{hp}$ are needed in order to express the property. Thus it cannot be tested on CWB. The same holds for all the properties reported next. The results of the tests of true concurrency properties on TCWB are displayed in Table 6.2.

- *Non-interference* [GM82]. Assume that every action is either a high level or a low level one. In general, to verify that activity at high level is not visible at low level, we should check for both causal dependencies and direct conflicts between high and low level actions. However, since there are no conflicts between the transitions in the cyclers, as we mentioned in Subsection 4.3.3, it is enough to check causal dependencies from high to low level transitions, therefore non-interference can be expressed by the formula $NonInterf = \nu X.([\![\mathsf{H}\, x]\!][\![x < \mathsf{L}\, y]\!] \mathsf{F} \wedge [\![\_\, z]\!] X)$. Since, as in the previous case,

we want to reproduce a worst case scenario, every transition in the cyclers is at high level, thus labelled by H, so that it must be checked.

- *Data race freedom* [AHMN91] with n memory locations. Assume that reading and writing operations on a memory location $\mathsf{i} \in [1, n]$ are labelled by $\mathsf{r_i}$ and $\mathsf{w_i}$, respectively. As explained in Subsection 4.3.3, a data race is identified by the presence of two concurrent operations on the same memory location, of which at least one is a write. To assess the absence of such race conditions we can use the formula $\mathit{RaceFree} = \nu X.(\bigwedge_{\mathsf{i}\in[1,n]} [\![\mathsf{w_i}\, x]\!]([\![\overline{x} < \mathsf{r_i}\, y]\!]\mathsf{F} \wedge [\![\overline{x} < \mathsf{w_i}\, y]\!]\mathsf{F}) \wedge [\![\_\, z]\!]X)$, which has variable size depending on the size of the memory. To reproduce the worst case scenario, we tested the property on processes made of n cyclers, each one continuously writing on a different memory location. More precisely, in each cycler i every transition is a write $\mathsf{w_i}$ on the memory location i assigned to that cycler. This means that the processes are, in fact, data race free, but every transition must be checked as it could possibly induce a data race.

Process name	Process states	Formula	Subformulae	TCWB (s)	CWB (s)
5 cyclers	1024	*Live*	6	< 1	< 1
6 cyclers	4096	*Live*	6	< 1	1
7 cyclers	16384	*Live*	6	8	5
8 cyclers	65536	*Live*	6	247	28
9 cyclers	262144	*Live*	6	-	570

Table 6.1: Results of tests performed on TCWB, compared with those of CWB.

Process name	Process states	Formula	Subformulae	TCWB (s)
5 cyclers	1024	*Atom*	12	< 1
6 cyclers	4096	*Atom*	12	5
7 cyclers	16384	*Atom*	12	97
8 cyclers	65536	*Atom*	12	-
5 cyclers	1024	*NonInterf*	7	< 1
6 cyclers	4096	*NonInterf*	7	1
7 cyclers	16384	*NonInterf*	7	11
8 cyclers	65536	*NonInterf*	7	251
5 cyclers	1024	*RaceFree*	38	< 1
6 cyclers	4096	*RaceFree*	45	2
7 cyclers	16384	*RaceFree*	52	17
8 cyclers	65536	*RaceFree*	59	280

Table 6.2: Results of tests performed on TCWB using true concurrency properties.

It is interesting to notice that in the tests above, moving from the interleaving fragment of the logic to true concurrency properties, the size of the automata does not grow exponentially, as the theoretical bound would suggest. Indeed, as mentioned before, the worst case would require all transitions to be related to all places in all possible ways, which is very unlikely in practice and surely not happening in the processes used for the tests. More precisely, in this particular case, the size of the automaton grows by a factor n equal to the number of cyclers in the system. This is because every reachable marking consists of n places and every transition causes exactly one of those places. Then, since the largest number of event variables appearing free in subformulae of the true concurrency properties is 1, the number of states of the automaton up to pointed-marking equivalence grows by the factor n, with respect to the interleaving case. So the upper bound to the size of the automaton for one of the true concurrency properties φ results to be just $O(|\mathsf{sf}(\varphi)| \cdot |\mathcal{R}(\mathcal{N})| \cdot n)$, where $\mathcal{N}$ is the Petri net representing the n cyclers process, while in the interleaving case the size coincides with the theoretical bound, i.e., $O(|\mathsf{sf}(\mathit{Live})| \cdot |\mathcal{R}(\mathcal{N})|)$.

Chapter 7

Games

Parity games [EJ91, Zie98] are two-player games of perfect information and, possibly, infinite duration played on directed graphs. Vertices of the game graph, called positions, are partitioned into two sets, one for each player, representing their own positions. The game starts from an initial position and consists in moving from a vertex to another along one of the outgoing edges. The player, whose positions include the current one, chooses along which outgoing edge to move. As in parity tree automata (hence the similar name), positions are labelled with priorities. An infinite play in the game, corresponding to a path in the graph, is won by a player or the other depending on the maximal priority appearing infinitely many times along the play, similarly to the acceptance condition of NPAs.

Consider, for example, the parity game shown in Figure 7.1. The positions of the two players, referred to as the existential player $\exists$ and the universal player $\forall$, are distinguished by the shape of the nodes: diamond nodes correspond to positions of player $\exists$, while box nodes correspond to positions of player $\forall$. Therefore, in this case, the existential player is allowed to move from positions b and e, while the universal player will move from positions a, c, d. The numbers in the nodes represent their priority. So, positions a and b have priority 0, position c has priority 1, and so on. Suppose that the game starts from position a. From there player $\forall$ can choose between moving to b or c. If $\forall$ chooses b, then player $\exists$ can move to either d or e. Since from every position the corresponding player is always able to move, every play of the game will be infinite. For instance, if the existential player always chooses to move to d from b, the resulting play would be the sequence of positions $(a, (b, d)^\infty)$ where positions b and d alternate indefinitely. Assuming the standard winning condition based on parity, similar to the one adopted for NPAs, such a play would be won by the universal player, because the maximal priority occurring infinitely often is 3 of position d, which is odd. If, instead, player $\exists$ from position b chooses to move to e, the resulting play would be (a, b, e^∞), which is won by $\exists$ since position e has priority 2, and thus the maximal priority occurring infinitely often along the play would be even. The choices of moves by a player can be defined as a strategy for the player, which fixes and specifies the move choice from each position of the player. For instance, a strategy for player $\exists$ is the one prescribing to always

move to position e from both positions b and e itself. A strategy is winning from a position if it allows the player to win every play starting from that position, independently from the moves of the opponent. Then, the aforementioned strategy for the existential player is winning from the starting position a, since player $\exists$ wins all possible plays following the strategy, which are, depending on the move of player $\forall$ from a, either (a, b, e^∞) or (a, c, e^∞), both won by $\exists$.

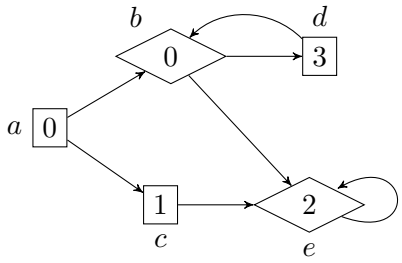


Figure 7.1: Example of a parity game.

In this chapter we present a general game-theoretic approach to the solution of systems of fixpoint equations over lattices. As we mentioned in the first part of the thesis, the verification and analysis of software systems often involve the computation of fixpoints over various kinds of lattices. For example, the semantics of fixpoint logics, as those we saw so far, are defined, at least when fixpoint operators are involved, by fixpoints over the powerset lattice of states of the system possibly paired with environments. In such cases the denotation of a least or greatest fixpoint formula can be seen as the least or greatest solution, respectively, of a suitably defined fixpoint equation. Moreover, formulae containing nested fixpoints, possibly with alternation, correspond to systems of such equations, where for each equation either the least or greatest solution is taken (see also [CKS92, Sei96]). We devise an approach based on a suitably defined parity game working over a vast class of lattices, namely continuous lattices, capturing most structures adopted in the analysis and verification of systems. Indeed, the method can be naturally applied in model-checking of a number of logics, also multi- or real-valued ones as quantitative mu-calculi (see, e.g., [Fit91, GLLS05, KL07]) and probabilistic mu-calculi (e.g., [MS17]), and many other verification tasks, such as bisimulation checking and computation of pseudo-metrics and behavioural distances [vBW05].

We first introduce continuous lattices and other notions on ordered structures which will be extensively used in what comes after. Then we define systems of fixpoint equations and their solutions, and propose a parity game for characterising the solution of such systems over continuous lattices. We discuss how such a game can be solved in terms of so-called progress measures [Jur00] and how to compute them. Finally, we see how the method can be applied for model-checking formulae of the logic $\mathcal{L}_{hp}$ over event structures.

7.1 Preliminaries on ordered structures

We focus on special lattices where elements are generated by suitably defined approximations. Given a lattice L, a subset $X \subseteq L$ is *directed* if $X \neq \emptyset$ and every pair of elements in X has an upper bound in X.

Definition 7.1 (way-below relation)**.** *Let $\langle L, \sqsubseteq \rangle$ be a complete lattice. Given two elements $l, l' \in L$ we say that l is* way-below *l', written $l \ll l'$, when for every directed set $D \subseteq L$, if $l' \sqsubseteq \bigsqcup D$ then there exists $d \in D$ such that $l \sqsubseteq d$. We denote by $\downarrow l$ the set of elements way-below l, i.e., $\downarrow l = \{l' \in L \mid l' \ll l\}$.*

Intuitively, the way-below relation captures a form of finitary approximation: if one imagines that $\sqsubseteq$ is an order on the information content of the elements, then $x \ll y$ means that whenever y can be "covered" by joining (possibly small) pieces of information, then x is covered by one of those pieces. Then a lattice is continuous if any element can be built by joining its finitary approximations.

Definition 7.2 (continuous lattices)**.** *A complete lattice $\langle L, \sqsubseteq \rangle$ is called* continuous *if for every element $l \in L$, it holds $l = \bigsqcup \downarrow l$.*

Concerning the origin of the name "continuous lattice", we can quote [Sco72] that says that "One of the justifications (by euphony at least) of the term *continuous lattice* is the fact that such spaces allow for so many continuous functions." For instance, one indication is the fact that suprema and infima are both continuous in such lattices (while normally only suprema are).

When L is a continuous lattice and B_L is a basis for it (Definition 2.3), for all $l \in L$, it holds that $l = \bigsqcup(B_L \cap \downarrow l)$.

Various lattices that are commonly used in the theory of computation enjoy a property stronger than continuity, defined below.

Definition 7.3 (compact element, algebraic lattice)**.** *Let $\langle L, \sqsubseteq \rangle$ be a complete lattice. An element $l \in L$ is called* compact *whenever $l \ll l$. The lattice L is* algebraic *if the set of compact elements $\{l \in L \mid l \ll l\}$ is a basis for L.*

Some examples are as follows:

- All finite lattices are continuous (since every finite directed set has a maximum). More generally, all algebraic lattices (which include all finite lattices) are continuous. The way-below relation is $x \ll y$ if x compact and $x \sqsubseteq y$.
- Given a set X, the powerset lattice $\mathbf{2}^X$, ordered by subset inclusion, is an algebraic lattice. The compact elements are the finite subsets. In fact, any set Y is the union of its finite subsets, i.e., $Y = \bigcup\{F \mid F \subseteq Y \wedge F \text{ finite}\}$. Since $\{F \mid F \subseteq Y \wedge F \text{ finite}\}$ is a directed set, compactness requires that $Y \subseteq F$ for some finite $F \subseteq Y$, hence $Y = F$. Therefore $Y \ll Z$ holds when Y is finite and $Y \subseteq Z$.

- The interval of real numbers $[0,1]_{\mathbb{R}}$ with the usual order $\leq$ is a continuous lattice. For $x, y \in [0,1]_{\mathbb{R}}$, we have $x \ll y$ when $x < y$ or $x = 0$. In fact, since the order is total, every non-empty subset $Y \subseteq [0,1]_{\mathbb{R}}$ is directed. Suppose that $y \leq \bigsqcup Y$ for such a Y. Then by standard properties of the reals there always exists a $y' \in Y$ such that $x \leq y'$ if and only if $x < y$ or $x = 0$. Note that this lattice is not algebraic since the only compact element is 0.

- The lattice W in Figure 7.2, obtained by adding to $\mathbb{N}$ a top element ω and an element a incomparable with all natural numbers except $0 < a < \omega$, is not continuous. In fact, $a \not\ll a$ since $a \sqsubseteq \bigsqcup \mathbb{N}$ but $a \not\sqsubseteq n$ for all $n \in \mathbb{N}$. Therefore $\downarrow a = \{0\}$, but $a \neq \bigsqcup \downarrow a$.

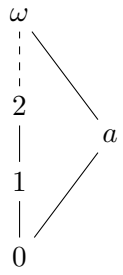


Figure 7.2: A complete lattice W which is not continuous.

A complete lattice L is *distributive* when suprema distribute over infima, i.e., if for all $x, y, z \in L$ it holds

$$x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z).$$

Moreover, an infinitary version of the property can also be defined in a similar way, namely, L is *completely distributive* if

$$\bigsqcap_{j \in J} \bigsqcup K_j = \bigsqcup_{c : J \to L} \bigsqcap_{j \in J} c(j)$$

where J is a set of indices, $(K_j)_{j \in J}$ is a family of subsets of L indexed by J, and $c : J \to L$ is a so-called *choice function* such that $c(j) \in K_j$ for all $j \in J$.

In the following we will often utilise tuples and subtuples of elements of the same type. Given a set X, an n-tuple $\boldsymbol{x} \in X^n$ and indices $i, j \in [1, n]$, we write $\boldsymbol{x}_{i,j}$ for the subtuple $(x_i, x_{i+1}, \ldots, x_j)$.

Definition 7.4 (pointwise order)**.** *Let $\langle P, \sqsubseteq \rangle$ be a poset. We denote by $\langle P^n, \sqsubseteq \rangle$ the set of n-tuples endowed with the* pointwise order *defined, for $\boldsymbol{x}, \boldsymbol{y} \in P^n$, by $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$ if and only if $x_i \sqsubseteq y_i$ for all $i \in [1, n]$.*

The structure $\langle L^n, \sqsubseteq \rangle$ is a complete lattice whenever L is a complete lattice, and it is continuous if L is continuous, with the way-below relation given by $\boldsymbol{l} \ll \boldsymbol{l}'$ if and only if $l_i \ll l'_i$ for all $i \in [1, n]$ [GHK+03, Proposition I-2.1]. More generally, for any set X, the set

of functions $L^X = \{f \mid f : X \to L\}$, endowed with pointwise order, is a lattice (complete and continuous when L is).

Definition 7.5 (lexicographic order)**.** *Let $\langle P, \sqsubseteq\rangle$ be a poset. We denote by $\langle P^n, \preceq\rangle$ the set of n-tuples endowed with the* lexicographic order *(where the last component is the most relevant), i.e., inductively, for $\boldsymbol{x}, \boldsymbol{y} \in P^n$, we let $\boldsymbol{x} \preceq \boldsymbol{y}$ if either $x_n \sqsubset y_n$ or $x_n = y_n$ and $\boldsymbol{x}_{1,n-1} \preceq \boldsymbol{y}_{1,n-1}$.*

When $\langle L, \sqsubseteq\rangle$ is a complete lattice also $\langle L^n, \preceq\rangle$ is a complete lattice. Given a set $X \subseteq L^n$, the infimum of X with respect to $\preceq$ can be obtained by taking the infimum of the single components, from the last to the first, i.e., it is defined inductively as $\bigsqcap X = \boldsymbol{l}$ where $l_i = \bigsqcap\{l'_i \mid \boldsymbol{l}' \in X \wedge \boldsymbol{l}'_{i+1,n} = \boldsymbol{l}_{i+1,n}\}$. The supremum can be defined analogously. Similarly, one can show that $\preceq$ is a well-order whenever $\sqsubseteq$ is.

As in [Jur00, HSC16], we will also need to consider tuples with a preorder arising from the lexicographic order, when some components are considered irrelevant.

Definition 7.6 (truncated lexicographic order)**.** *Let $\langle P, \sqsubseteq\rangle$ be a poset and let $n \in \mathbb{N}$. For $i \in [1, n]$ we define a preorder $\preceq_i$ on P^n by letting, for $\boldsymbol{x}, \boldsymbol{y} \in P^n$, $\boldsymbol{x} \preceq_i \boldsymbol{y}$ if $\boldsymbol{x}_{i,n} \preceq \boldsymbol{y}_{i,n}$. We write $=_i$ for the equivalence induced by $\preceq_i$ and $\boldsymbol{x} \prec_i \boldsymbol{y}$ for $\boldsymbol{x} \preceq_i \boldsymbol{y}$ and $\boldsymbol{x} \neq_i \boldsymbol{y}$. Whenever $\sqsubseteq$ is a well-order, given $X \subseteq P^n$ with $X \neq \emptyset$ and $i \in [1, n]$, we write $\min_{\preceq_i} X$ for the vector $\boldsymbol{x} = (\bot, \dots, \bot, x_i, \dots, x_n)$ where $\boldsymbol{x}_{i,n} = \min_{\preceq}\{\boldsymbol{l}_{i,n} \mid \boldsymbol{l} \in X\}$.*

In words, $\preceq_i$ is the lexicographic order restricted to the components $i, i+1, \dots, n$. For instance, if $P = \mathbb{N}$ with the usual order, then $(6, 1, 4, 7) \prec_2 (5, 2, 4, 7)$, while $(6, 1, 4, 7) =_3 (5, 2, 4, 7)$. More generally, since $\preceq$ is a lexicographic order, if $\boldsymbol{x} \prec_i \boldsymbol{y}$ for some $i \in [1, n]$, then also $\boldsymbol{x} \prec_j \boldsymbol{y}$ for all $j \in [1, i]$, but we can only assert that $\boldsymbol{x} \preceq_k \boldsymbol{y}$ for all $k \in [i+1, n]$.

The collection of all ordinals ordered by $\leq$ is well-ordered. Given any ordinal α, the collection of ordinals dominated by α is a set $[\alpha] = \{\lambda \mid \lambda \leq \alpha\}$, which, seen as an ordered structure, is a complete lattice. Infimum and supremum of a set X of ordinals will be denoted by $\inf X$ (which equals $\min X$ if $X \neq \emptyset$) and $\sup X$, respectively. From classical results (see [Ran52, Theorems 1 and 2]) it follows that the lattice $[\alpha]$ is completely distributive. Furthermore, from the properties discussed above, for a fixed $n \in \mathbb{N}$ and an ordinal α, the n-tuples of ordinals below α, referred to as *ordinal vectors*, endowed with the lexicographic order $\langle [\alpha]^n, \preceq\rangle$ is also a completely distributive lattice.

Given a complete lattice L, its *height* is the ordinal λ_L defined as the supremum of the lengths of all strictly ascending, possibly transfinite, chains.

7.2 Systems of fixpoint equations

We are interested in systems of fixpoint equations over some complete lattice, where, for each equation, one can choose either the least or the greatest solution.

Definition 7.7 (system of equations)**.** *Let L be a complete lattice. A* system of equations *E over L is a list of equations of the following form*

$$\begin{aligned} x_1 &=_{\eta_1} f_1(x_1, \ldots, x_m) \\ &\ldots \\ x_m &=_{\eta_m} f_m(x_1, \ldots, x_m) \end{aligned}$$

where $f_i : L^m \to L$ are monotone functions and $\eta_i \in \{\mu, \nu\}$. The system will often be denoted as $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, where $\boldsymbol{x}$, $\boldsymbol{\eta}$ and $\boldsymbol{f}$ are the obvious tuples. We denote by $\emptyset$ the system with no equations.

Systems of equations of this kind have been considered by various authors, e.g., [CKS92, Sei96, HSC16]. In particular, [HSC16] works on general lattices.

We next define the pre-solutions of a system as tuples of lattice elements that, replacing the variables, satisfy all the equations of the system. The solution will be a suitably chosen pre-solution, taking into account also the η_i annotations that specify for each equation whether the least or greatest solution is required.

Definition 7.8 (pre-solution)**.** *Let L be a complete lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. A* pre-solution *of E is any tuple $\boldsymbol{u} \in L^m$ such that $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{u})$.*

Note that $\boldsymbol{f}$ can be seen as a function $\boldsymbol{f} : L^m \to L^m$. In this view, pre-solutions are the fixpoints of $\boldsymbol{f}$. Since all components f_i are monotone, also $\boldsymbol{f}$ is monotone over $(L^m, \sqsubseteq)$. Then, the set of fixpoints of $\boldsymbol{f}$, i.e., the pre-solutions of the system, are a non-empty complete sublattice of L^m (Theorem 2.1). In order to define the solution of a system we need some further notation.

Definition 7.9 (substitution)**.** *Let L be a complete lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given an index $i \in [1, m]$ and an element $l \in L$ we write $E[x_i := l]$ for the system of $m-1$ equations obtained from E by removing the i-th equation and replacing x_i by l in the other equations, i.e., if $\boldsymbol{x} = \boldsymbol{x}' x_i \boldsymbol{x}''$, $\boldsymbol{\eta} = \boldsymbol{\eta}' \eta_i \boldsymbol{\eta}''$ and $\boldsymbol{f} = \boldsymbol{f}' f_i \boldsymbol{f}''$ then $E[x_i := l]$ is $\boldsymbol{x}'\boldsymbol{x}'' =_{\boldsymbol{\eta}'\boldsymbol{\eta}''} \boldsymbol{f}'\boldsymbol{f}''(\boldsymbol{x}', l, \boldsymbol{x}'')$.*

Let $f[x_i := l] : L^{m-1} \to L$ be defined by $f[x_i := l](\boldsymbol{x}', \boldsymbol{x}'') = f(\boldsymbol{x}', l, \boldsymbol{x}'')$. Then, explicitly, the system $E[x_i := l]$ has $m-1$ equations:

$$x_j =_{\eta_j} f_j[x_i := l](\boldsymbol{x}', \boldsymbol{x}'') \qquad j \in \{1, \ldots, i-1, i+1, \ldots, n\}$$

We can now recursively define the solution of a system of equations. The notion is the same as in [HSC16], although we find it convenient to adopt a more succinct formulation.

Definition 7.10 (solution)**.** *Let L be a complete lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. The* solution *of E, denoted $sol(E) \in L^m$, is defined inductively as follows:*

$$\begin{array}{lcl} sol(\emptyset) & = & \epsilon \\ sol(E) & = & (sol(E[x_m := u_m]),\, u_m) \text{ where } u_m = \eta_m(\lambda x.\, f_m(sol(E[x_m := x]), x)) \end{array}$$

The i-th component of the solution will be denoted $sol_i(E)$.

In words, for solving a system of m equations, the last variable is considered as a fixed parameter x and the system of $m-1$ equations that arises from dropping the last equation is recursively solved. This produces an $(m-1)$-tuple parametric on x, i.e., we get $\boldsymbol{u}_{1,m-1}(x) = sol(E[x_m := x])$. Inserting this parametric solution into the last equation, we get an equation in a single variable

$$x =_{\eta_m} f_m(\boldsymbol{u}_{1,m-1}(x), x)$$

that can be solved by taking for the function $\lambda x.\, f_m(\boldsymbol{u}_{1,m-1}(x), x)$, the least or greatest fixpoint, depending on whether the last equation is a μ- or ν-equation. This provides the m-th component of the solution $u_m = \eta_m(\lambda x.\, f_m(\boldsymbol{u}_{1,m-1}(x), x))$. The remaining components of the solution are obtained inserting u_m in the parametric solution $\boldsymbol{u}_{1,m-1}(x)$ previously computed, i.e., the $(m-1)$-tuple $\boldsymbol{u}_{1,m-1}(u_m)$.

The next lemma will be helpful in several circumstances. In particular, it shows that the definition above is well-given, since we are taking (least or greatest) fixpoints of monotone functions.

Lemma 7.1 (solution is monotone)**.** *Let L be a complete lattice and let E be a system of $m > 0$ equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. For $i \in [1, m]$ the function $g : L \to L^{m-1}$ defined by $g(x) = sol(E[x_i := x])$ is monotone.*

Proof. The proof proceeds by induction on m. The base case $m = 1$ holds trivially since necessarily $i = 1$ and for any $x \in L$, the system $E[x_i := x]$ is empty, with empty solution.

Let us assume $m > 1$. We distinguish two subcases according to whether $i = m$ or $i < m$. If $i = m$ then by definition of solution

$$g(x) = sol(E[x_m := x]) = (sol(E[x_m := x][x_{m-1} := u_{m-1}(x)]), u_{m-1}(x)) \tag{7.1}$$

where $u_{m-1}(x) = \eta_{m-1}(\lambda y.\, f_{m-1}(sol(E[x_m := x][x_{m-1} := y]), y, x)$.

Next observe that the function $h : L^2 \to L^{m-2}$ defined by

$$h(x, y) = sol(E[x_m := x][x_{m-1} := y])$$

is monotone. In fact, it is monotone in y by inductive hypothesis, and also in x, again by inductive hypothesis, since $E[x_m := x][x_{m-1} := y] = E[x_{m-1} := y][x_m := x]$. Observe that u_{m-1} can be written as

$$u_{m-1}(x) = \eta_{m-1}(\lambda y.\, f_{m-1}(h(x, y), y, x))$$

Recalling that also f_{m-1} is monotone, we deduce that u_{m-1} is monotone.

Finally, using the definition of g and u_{m-1}, from (7.1) we can derive

$$g(x) = (h(x, u_{m-1}(x)), u_{m-1}(x))$$

which allows us to conclude that g is monotone.

If instead, $i < m$, just note that

$$g(x) = sol(E[x_i := x]) = (sol(E[x_i := x][x_m := u_m(x)]), u_m(x)) \tag{7.2}$$

where $u_m(x) = \eta_m(\lambda y.f_m\,(sol(E[x_i := x][x_m := y]), y, x)$. Then the proof proceeds as in the previous case. □

It can be easily proved that the solution of a system is, as anticipated, a special pre-solution.

Lemma 7.2 (solution is pre-solution)**.** *Let E be a system of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $\boldsymbol{u}$ be its solution. Then $\boldsymbol{u}$ is a pre-solution, i.e., $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{u})$.*

Proof. The proof proceeds by induction on m. The base case $m = 0$ trivially holds. For any $m > 0$, let $\boldsymbol{u} = \boldsymbol{u}'u_m$, $\boldsymbol{f} = \boldsymbol{f}'f_m$ and $\boldsymbol{x} = \boldsymbol{x}'x_m$. Since $\boldsymbol{u}' = sol(E[x_m := u_m])$, by inductive hypothesis, we have that

$$\boldsymbol{u}' = \boldsymbol{f}'[x_m := u_m](\boldsymbol{u}') = \boldsymbol{f}'(\boldsymbol{u}). \tag{7.3}$$

Moreover, by definition of solution, we have that $u_m = \eta_m(\lambda x.\, f_m(sol(E[x_m := x]), x))$. Hence $u_m = f_m(sol(E[x_m := u_m]), u_m))$. Recalling that $sol(E[x_m := u_m]) = \boldsymbol{u}'$ we deduce $u_m = f_m(\boldsymbol{u}', u_m) = f_m(\boldsymbol{u})$, that together with (7.3) gives $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{u})$ as desired. □

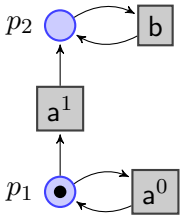
As a prototypical example, we discuss how mu-calculus formulae can be equivalently seen as systems of fixpoint equations. Later we will show how this can be done for $\mathcal{L}_{hp}$ as well, but for now, as an illustration of the technique, we deal with this simpler example.

First note that any mu-calculus formula can be expressed in equational form, by inserting an equation for each propositional variable (see [CKS92, Sei96]). The reverse translation is also possible, hence these specification languages are equally expressive. Here, we will only depict the relation via an example, the formal treatment will be presented later for $\mathcal{L}_{hp}$ (of which mu-calculus is a fragment).

Let $\varphi = \nu X_2.([\_]X_2 \wedge \mu X_1.(\langle \mathsf{b} \rangle \mathsf{T} \vee \langle \_ \rangle X_1))$ be a formula requiring that from all reachable states there exists a path that eventually reaches a state where it is possible to execute a b action. The equational form is quite straightforward and is reported in Figure 7.3a. Consider the Petri net $\mathcal{N}$ in Figure 7.3b whose reachable markings are $\mathcal{R}(\mathcal{N}) = \{M_1, M_2\}$, where the initial marking is $M_1 = \{p_1\}$, while $M_2 = \{p_2\}$. The resulting system of equations on the lattice $\mathbf{2}^{\mathcal{R}(\mathcal{N})}$ is given in Figure 7.3c, where $\Diamond, \Diamond_{\mathsf{b}}, \Box : \mathbf{2}^{\mathcal{R}(\mathcal{N})} \to \mathbf{2}^{\mathcal{R}(\mathcal{N})}$ are defined as $\Diamond(S) = \{M \in \mathcal{R}(\mathcal{N}) \mid \exists\, M' \in \mathcal{R}(\mathcal{N}).\, M[t\rangle M' \wedge M' \in S\}$, $\Diamond_{\mathsf{b}}$ is as $\Diamond$ but requiring that the transition is labelled by b, and $\Box(S) = \{M \in \mathcal{R}(\mathcal{N}) \mid \forall\, M' \in \mathcal{R}(\mathcal{N}).\, M[t\rangle M' \Rightarrow M' \in S\}$ for all $S \subseteq \mathcal{R}(\mathcal{N})$.

$$\begin{aligned} X_1 &=_\mu \langle \mathsf{b} \rangle \mathsf{T} \vee \langle \_ \rangle X_1 \\ X_2 &=_\nu [\_] X_2 \wedge X_1 \end{aligned}$$

(a) Equational form of φ



(b) $\mathcal{N}$

$$\begin{aligned} x_1 &=_\mu \Diamond_{\mathsf{b}} \mathcal{R}(\mathcal{N}) \cup \Diamond x_1 \\ x_2 &=_\nu \Box x_2 \cap x_1 \end{aligned}$$

(c) System of equations over $\mathbf{2}^{\mathcal{R}(\mathcal{N})}$

Figure 7.3: Equations for $\nu X_2.([\_]X_2 \wedge \mu X_1.(\langle \mathsf{b} \rangle \mathsf{T} \vee \langle \_ \rangle X_1))$ over states of $\mathcal{N}$.

The solution is $x_1 = x_2 = \mathcal{R}(\mathcal{N})$. In particular, $x_2 = \mathcal{R}(\mathcal{N})$ corresponds to the fact that the formula φ holds in every state of $\mathcal{N}$.

Now, consider the formula $\varphi' = \nu Y_2.([\_]Y_2 \wedge \mu Y_1.((\langle \mathsf{b} \rangle \mathsf{T} \wedge \langle \_ \rangle Y_2) \vee \langle \_ \rangle Y_1))$ requiring that from all reachable states there is a path along which infinitely often a b-labelled transition is enabled. The equational form of φ' is

$$\begin{aligned} Y_1 &=_\mu (\langle \mathsf{b} \rangle \mathsf{T} \wedge \langle \_ \rangle Y_2) \vee \langle \_ \rangle Y_1 \\ Y_2 &=_\nu [\_] Y_2 \wedge Y_1 \end{aligned}$$

On the net $\mathcal{N}$, the corresponding system of equations over the lattice $\mathbf{2}^{\mathcal{R}(\mathcal{N})}$ is

$$\begin{aligned} y_1 &=_\mu (\Diamond_{\mathsf{b}} \mathcal{R}(\mathcal{N}) \cap \Diamond y_2) \cup \Diamond y_1 \\ y_2 &=_\nu \Box y_2 \cap y_1 \end{aligned}$$

The solution of the system is again $y_1 = y_2 = \mathcal{R}(\mathcal{N})$. However, notice that this time the order of the equations is relevant, while in the previous example it was not. Indeed, if we swap the two equations in the system, the solution becomes $y_1 = y_2 = \emptyset$. In fact, the system would now correspond to the formula $\mu Y_1.((\langle \mathsf{b} \rangle \mathsf{T} \wedge \langle \_ \rangle \nu Y_2.([\_]Y_2 \wedge Y_1)) \vee \langle \_ \rangle Y_1)$ which never holds in any state of $\mathcal{N}$. In general, the order of the equations is important whenever there is an alternation of fixpoints, as it happens in φ'.

7.2.1 Approximating the solution

The game-theoretic characterisation of the solution of a system of fixpoint equations discussed in the next section will rely on a notion of approximation of the solution that is reminiscent of the lattice progress measure in [HSC16].

Definition 7.11 (approximants)**.** *Let L be a complete lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given any tuple $\boldsymbol{l} \in L^m$, let $f_{i,\boldsymbol{l}} : L \to L$ be the function defined as*

$$f_{i,\boldsymbol{l}}(x) = f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := x]), x, \boldsymbol{l}_{i+1,m}).$$

We say that a tuple $\boldsymbol{l} \in L^m$ is a μ-approximant *when for all $i \in [1, m]$*

- *if $\eta_i = \mu$ then $l_i = f^\alpha_{i,\boldsymbol{l}}(\bot)$ for some ordinal α, else*

- *if* $\eta_i = \nu$ *then* $l_i = \nu(f_{i,\boldsymbol{l}})$.

Dually, $\boldsymbol{l} \in L^m$ *is a* ν-approximant *when for all* $i \in [1, m]$

- *if* $\eta_i = \mu$ *then* $l_i = \mu(f_{i,\boldsymbol{l}})$, *else*
- *if* $\eta_i = \nu$ *then* $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\top)$ *for some ordinal* α.

Whenever $\boldsymbol{l}$ *is a* μ*-approximant we write* $ord(\boldsymbol{l})$ *to denote the least* m*-tuple of ordinals* $\boldsymbol{\alpha}$ *such that for all* $i \in [1, m]$, *if* $\eta_i = \mu$ *then* $l_i = f^{\alpha_i}_{i,\boldsymbol{l}}(\bot)$ *else, if* $\eta_i = \nu$, $l_i = f^{\alpha_i}_{i,\boldsymbol{l}}(\top) = \nu(f_{i,\boldsymbol{l}})$.

Observe that, spelling out the definition of the solution of a system of equations, it can be easily seen that $sol_i(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}]) = \eta_i(f_{i,\boldsymbol{l}})$. Then a μ-approximant is obtained by taking under-approximations for the least fixpoints and the exact value for greatest fixpoints. In fact, in the case of μ-approximants, for each $i \in [1, m]$, if $\eta_i = \nu$, the i-th component is set to $\nu(f_{i,\boldsymbol{l}})$ which is the i-th component $sol_i(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}])$ of the solution. Instead, if $\eta_i = \mu$ the component l_i is set to $f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ for some ordinal α, which is an under-approximation of $\mu(f_{i,\boldsymbol{l}}) = sol_i(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}])$, obtained by iterating $f_{i,\boldsymbol{l}}$ over $\bot$ up to ordinal α. For ν-approximants the situation is dual.

We remark that the function $f_{i,\boldsymbol{l}}$ depends only on the subtuple $\boldsymbol{l}_{i+1,m}$. In particular $f_{m,\boldsymbol{l}}$ does not depend on $\boldsymbol{l}$ at all. In fact, $f_{m,\boldsymbol{l}} = \lambda x.\, f_m(sol(E[x_m := x]), x)$. Using $\boldsymbol{l}$ as subscript instead of the corresponding subtuple is a slight abuse of the notation that makes it lighter and more readable.

Approximants can be given an inductive characterisation. Besides shedding some light on the notion of approximant, the following easy result will be useful at a technical level.

Lemma 7.3 (inductive characterisation of approximants)**.** *Let* L *be a complete lattice, let* E *be a system of* $m > 0$ *equations over* L *of the kind* $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, *and let* $g_m : L \to L$ *be the function* $g_m(x) = f_m(sol(E[x_m := x]), x)$. *A tuple* $\boldsymbol{l} \in L^m$ *is a* μ*-approximant if and only if the following conditions hold*

1. *either* $\eta_m = \mu$ *and* $l_m = g^{\alpha}_m(\bot)$ *for some ordinal* α, *or* $\eta_m = \nu$ *and* $l_m = \nu g_m$
2. $\boldsymbol{l}_{1,m-1}$ *is a* μ*-approximant of* $E[x_m := l_m]$.

Proof. It follows immediately from Definition 7.11. □

We next observe that the name approximant is appropriate, i.e., μ-approximants provide an approximation of the solution from below, while ν-approximants from above. The solution is thus the only pre-solution which is both a μ- and a ν-approximant.

Lemma 7.4 (solution and approximants)**.** *Let* L *be a complete lattice and let* E *be a system of* m *equations over* L *of the kind* $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. *The solution of* E *is the greatest* μ*-approximant and the least* ν*-approximant.*

Proof. The solution $\boldsymbol{u}$ is clearly a μ-approximant by definitions of solution and approximant and Theorem 2.2. We prove that it is the greatest one by induction on m. If $m = 0$ the thesis is vacuously true. If $m > 0$, consider another μ-approximant $\boldsymbol{l}$. We distinguish two subcases according to whether $\eta_m = \mu$ or $\eta_m = \nu$. If $\eta_m = \mu$, we know that $l_m = f^{\alpha}_{m,\boldsymbol{l}}(\bot)$ for some ordinal α. Observe that $f_{m,\boldsymbol{l}} = \lambda x.\, f_m(sol(E[x_m := x]), x))$ is the function for which u_m is the least fixpoint, hence

$$l_m \sqsubseteq u_m. \tag{7.4}$$

Moreover, by Lemma 7.3, $\boldsymbol{l}_{1,m-1}$ is a μ-approximant for the system $E[x_m := l_m]$. Hence, by inductive hypothesis

$$\boldsymbol{l}_{1,m-1} \sqsubseteq sol(E[x_m := l_m]) \tag{7.5}$$

Moreover, by Lemma 7.1, since $l_m \sqsubseteq u_m$, we get $sol(E[x_m := l_m]) \sqsubseteq sol(E[x_m := u_m]) = \boldsymbol{u}_{1,m-1}$. Therefore, combined with (7.4) and (7.5), we conclude $\boldsymbol{l} \sqsubseteq \boldsymbol{u}$.

The proof for ν-approximants is dual. □

We conclude with a technical lemma that will be used to locally modify approximations in the game.

Lemma 7.5 (updating approximants)**.** *Let L be a complete lattice, let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, and let $\boldsymbol{l}$ be a μ-approximant with $ord(\boldsymbol{l}) = \boldsymbol{\alpha}$. For an index $i \in [1, m]$ and an ordinal $\alpha \leq \alpha_i$*

1. *if $\eta_i = \mu$, then $\boldsymbol{l}' = (sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i]), l'_i, \boldsymbol{l}_{i+1,m})$, with $l'_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ for some ordinal α, is a μ-approximant*
2. *if $\eta_i = \nu$, then $\boldsymbol{l}' = (sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}]), \boldsymbol{l}_{i+1,m})$ is a μ-approximant*

and in both cases $ord(\boldsymbol{l}') \preceq_i ord(\boldsymbol{l})$. A dual result holds for ν-approximants.

Proof. Let us focus on (1). To show that $\boldsymbol{l}' = (sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i]), l'_i, \boldsymbol{l}_{i+1,m})$ is a μ-approximant, first observe that the components $l_{i+1}, \ldots, l_m$ do not change. Component l'_i is of the desired shape by definition. Finally, for $j < i$ the component l'_j is defined as $sol_j(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i])$ and thus, by definition of solution of a system, if $\eta_j = \nu$ then $l'_j = \nu(f_{j,\boldsymbol{l}'})$ and if $\eta_j = \mu$ then $l'_j = \mu(f_{j,\boldsymbol{l}'}) = f^{\beta}_{j,\boldsymbol{l}'}(\bot)$ for some ordinal β, as desired. Finally observe that since $\boldsymbol{l}$ and $\boldsymbol{l}'$ coincide on components $i+1, \ldots, m$, and $l_i = f^{\alpha_i}_{i,\boldsymbol{l}}(\bot)$, while $l'_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$, with $\alpha \leq \alpha_i$, clearly $ord(\boldsymbol{l}') \preceq_i ord(\boldsymbol{l})$.

The proof of (2) is analogous. In fact, also in this case the components $i+1, \ldots, m$ are unchanged and finally, for $j \leq i$ the component l'_j is defined as $sol_j(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}])$, thus the same reasoning as above applies here.

Both cases can be easily dualised for ν-approximants. □

7.3 Fixpoint game

In this section we present a game-theoretic approach to the solution of a system of fixpoint equations over a continuous lattice. More precisely, given a lattice with a fixed basis, the game allows us to check whether an element of the basis is smaller (with respect to $\sqsubseteq$) than the solution of a selected equation. This corresponds to solving the associated verification problem. For instance, when model-checking, one is interested in establishing whether a system satisfies a formula φ, which amounts to check whether $\{s_0\} \subseteq u_\varphi$ where s_0 is the initial state and u_φ is the solution of the system of equations associated with φ.

The fixpoint game that we introduce has been inspired by the unfolding game described in [Ven08], that works for a single fixpoint equation over the powerset lattice. We adopted the name *fixpoint game*, analogously to [HKMV17].

Definition 7.12 (fixpoint game)**.** *Let L be a continuous lattice and let B_L be a basis for L such that $\bot \notin B_L$. Given a system E of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, the corresponding fixpoint game is a parity game, with an existential player $\exists$ and a universal player $\forall$, defined as follows:*

- *the positions of $\exists$ are pairs (b, i) where $b \in B_L$ and $i \in [1, m]$ and those of $\forall$ are tuples $\boldsymbol{l} \in L^m$*
- *from (b, i) the possible moves of $\exists$ are $\mathbf{E}(b, i) = \{\boldsymbol{l} \in L^m \mid b \sqsubseteq f_i(\boldsymbol{l})\}$*
- *from $\boldsymbol{l} \in L^m$ the possible moves of $\forall$ are $\mathbf{A}(\boldsymbol{l}) = \{(b, i) \in B_L \times [1, m] \mid b \ll l_i\}$.*

The game is schematised in Table 7.1. For a finite play, the winner is the player whose opponent is unable to move. For an infinite play, let h be the highest index that occurs infinitely often in a pair (b, i). If $\eta_h = \nu$ then $\exists$ wins, otherwise $\forall$ wins.

Position	Player	Moves
(b, i)	$\exists$	$(l_1, \ldots, l_m)$ such that $b \sqsubseteq f_i(l_1, \ldots, l_m)$
$(l_1, \ldots, l_m)$	$\forall$	(b', j) such that $b' \ll l_j$

Table 7.1: The fixpoint game.

The fixpoint game is a parity game on a possibly infinite graph. Observe that the winning condition, while not exactly based on parity, is the natural formulation of the standard winning condition in this setting. Instead of saying that an index (or priority) is "even" or "odd", here the two roles are assigned depending on the kind of fixpoint associated with the index. Moreover, every play, either infinite or finite because a player cannot move any more, is won by a unique player.

Hereafter, whenever we consider a lattice L, we assume that a basis B_L is fixed such that $\bot \notin B_L$. Elements of the basis will be denoted by letters b with super or subscripts.

We will prove soundness and completeness of the game, i.e., we will show that if $\boldsymbol{u}$ is the solution of the system, given a basis element $b \in B_L$ and $i \in [1, m]$, if $b \sqsubseteq u_i$, then starting from (b, i) the existential player has a winning strategy, otherwise the universal player has a winning strategy. Since every play is won by a unique player, the existence of a winning strategy for a player implies the non-existence of one for the other.

As an example, consider the system of equations in Figure 7.3c, corresponding to the mu-calculus formula $\varphi = \nu X_2.([\_]X_2 \wedge \mu X_1.(\langle \mathsf{b} \rangle \mathsf{T} \vee \langle \_ \rangle X_1))$. Recall that the lattice is $\langle \mathbf{2}^{\mathcal{R}(\mathcal{N})}, \subseteq \rangle$, which is continuous since it is a powerset lattice, and let us fix as a basis the set of singletons $B = \{\{M_1\}, \{M_2\}\}$.

A portion of the fixpoint game is graphically represented as a parity game in Figure 7.4. Recall that diamond nodes correspond to positions of player $\exists$ and box nodes to positions of player $\forall$. Only a subset of the possible positions for $\forall$ are represented. The positions which are missing, such as $(\{a, b\}, \{a, b\})$, can be shown to be redundant, in a sense formalised later in Subsection 7.4.1, so that the subgame is equivalent to the full game. Numbers in the diamond nodes correspond to priorities. Box nodes do not have priorities (or we can assume priority 0). Since indices 1 and 2 corresponds to a μ- and a ν-equation, respectively, in this specific case the winning condition for player $\exists$ is: either the play is finite and $\exists$ plays last or the play is infinite and the highest priority that occurs infinitely often is 2.

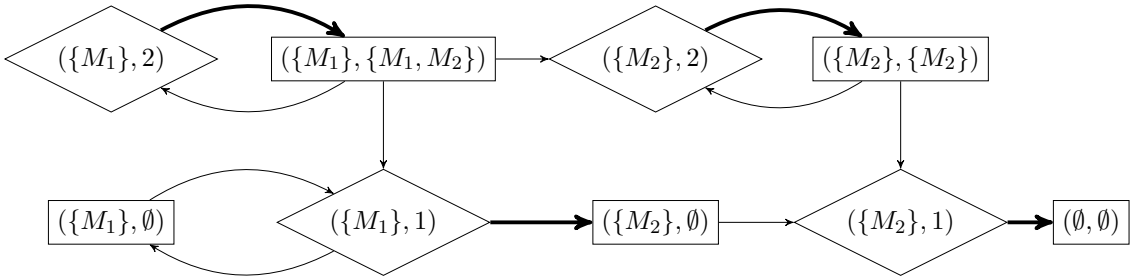


Figure 7.4: Graphical representation of a fixpoint game.

Let (u_1, u_2) be the solution of the system. We can check if $M_1 \in u_2$, i.e., if M_1 satisfies φ, by playing the game from the position $(\{M_1\}, 2)$. In fact, $\{M_1\} \subseteq u_2$ amounts to $M_1 \in u_2$. Player $\exists$ has a winning strategy that we can represent as a function ς from the positions of the game (for any play) to the corresponding moves of player $\exists$, i.e., $\varsigma : B \times \{1, 2\} \to \mathbf{2}^{\mathcal{R}(\mathcal{N})} \times \mathbf{2}^{\mathcal{R}(\mathcal{N})}$. A winning strategy for $\exists$ is given by

$$\begin{aligned} \varsigma(\{M_1\}, 1) &= (\{M_2\}, \emptyset) \\ \varsigma(\{M_1\}, 2) &= (\{M_1\}, \{M_1, M_2\}) \\ \varsigma(\{M_2\}, 1) &= (\emptyset, \emptyset) \\ \varsigma(\{M_2\}, 2) &= (\{M_2\}, \{M_2\}). \end{aligned}$$

In Figure 7.4 we depict by bold arrows the choices prescribed by such strategy. Observe that in this simple example the only relevant choice made by the strategy is on $(\{M_1\}, 1)$.

Since that is the only position where $\exists$ could pick an "erroneous" move and then lose.

A possible play of the game could be the following, where $\overset{p}{\rightsquigarrow}$ denotes a move of player $p \in \{\exists, \forall\}$:

$$(\{M_1\}, 2) \overset{\exists}{\rightsquigarrow} (\{M_1\}, \{M_1, M_2\}) \overset{\forall}{\rightsquigarrow} (\{M_1\}, 1) \overset{\exists}{\rightsquigarrow} (\{M_2\}, \emptyset) \overset{\forall}{\rightsquigarrow} (\{M_2\}, 1) \overset{\exists}{\rightsquigarrow} (\emptyset, \emptyset) \overset{\forall}{\not\rightsquigarrow}$$

hence $\exists$ wins. Another (infinite) play is the following. It is also won by $\exists$ since the highest index that occurs infinitely often is 2, which is a ν-index:

$$(\{M_1\}, 2) \overset{\exists}{\rightsquigarrow} (\{M_1\}, \{M_1, M_2\}) \overset{\forall}{\rightsquigarrow} (\{M_1\}, 2) \overset{\exists}{\rightsquigarrow} (\{M_1\}, \{M_1, M_2\}) \overset{\forall}{\rightsquigarrow} \dots$$

Note that if $\exists$ always plays as specified by ς, she will always win.

7.3.1 Soundness and completeness

Before proving soundness and completeness of the game in the general case, as a warm up, we give some intuition and outline the proof for the case of a single equation. Let $f : L \to L$ be a monotone function on a continuous lattice L and consider the equation $x =_\eta f(x)$, where $\eta \in \{\nu, \mu\}$, with solution $u = \eta f$. In this case the positions for $\exists$ are simply basis elements $b \in B_L$ and $\exists$ must choose $l \in L$ such that $b \sqsubseteq f(l)$. Positions of $\forall$ are lattice elements $l \in L$ and moves are elements of the basis $b \in B_L$, with $b \ll l$. In the case of $\eta = \mu$, player $\forall$ wins infinite plays, and in the case of $\eta = \nu$, player $\exists$ wins infinite plays.

When $\eta = \mu$, if $b \sqsubseteq u$, then $b \sqsubseteq f^\alpha(\bot)$ for some ordinal α (Theorem 2.2). The idea is that $\exists$ can win by descending the chain $f^\beta(\bot)$. E.g., if $\beta = \gamma + 1$ is a successor ordinal, then $\exists$ can play $f^\gamma(\bot)$. If instead, $\eta = \nu$, then the existential player can win just by identifying some post-fixpoint l such that $b \sqsubseteq l$. In fact, if l is a post-fixpoint, i.e., $l \sqsubseteq f(l)$ we know that $l \sqsubseteq u$. Moreover, if $b \sqsubseteq l$ then $b \sqsubseteq f(l)$ and thus $\exists$ can cycle on l and win. More formally:

(Case $\eta = \mu$)

In this case $u = f^\alpha(\bot)$ for some ordinal α by Theorem 2.2.

- *Completeness:* We show that whenever $b \sqsubseteq f^\beta(\bot)$, for some ordinal β (i.e., b is below some μ-approximant), then $\exists$ has a winning strategy, by transfinite induction on β. First observe that $\beta > 0$. In fact, otherwise $b \sqsubseteq f^0(\bot) = \bot$, hence $b = \bot$, while $\bot \notin B_L$ by hypothesis. Hence we have two possibilities:
  - If β is a limit ordinal, player $\exists$ plays $l = f^\beta(\bot)$, which is a post-fixpoint and hence $b \sqsubseteq f^\beta(\bot) \sqsubseteq f(f^\beta(\bot))$. Then $\forall$ chooses $b' \ll f^\beta(\bot) = \bigsqcup_{\gamma<\beta} f^\gamma(\bot)$. Since this is a directed supremum, by definition of the way-below relation there exists $\gamma < \beta$ with $b' \sqsubseteq f^\gamma(\bot)$.
  - If $\beta = \gamma + 1$, $\exists$ plays $l = f^\gamma(\bot)$ and $\forall$ chooses $b' \ll f^\gamma(\bot)$, hence $b' \sqsubseteq f^\gamma(\bot)$.

Note that $\exists$ always has a move and the answer of $\forall$ is some $b' \sqsubseteq f^\gamma(\bot)$, with $\gamma < \beta$, from which there exists a winning strategy for $\exists$ by the inductive hypothesis.

- *Soundness:* We show that whenever $b \not\sqsubseteq u$, player $\forall$ has a winning strategy.

  Observe that a move of $\exists$ will be some l such that $b \sqsubseteq f(l)$. Note that there must be a $b' \ll l$ with $b' \not\sqsubseteq u$. In fact, otherwise, if for all $b' \ll l$ it holds that $b' \sqsubseteq u$, since L is a continuous lattice, we would have $l = \bigsqcup\{b' \mid b' \ll l\} \sqsubseteq u$ and furthermore $b \sqsubseteq f(l) \sqsubseteq f(u) = u$, which is a contradiction.

  Hence $\forall$ can choose such a $b' \ll l$ with $b' \not\sqsubseteq u$ and the game can continue. Then either $\exists$ runs out of moves at some point or we end up in an infinite play. In both cases $\forall$ wins.

(Case $\eta = \nu$)

In this case $u = f^\alpha(\top)$ for some ordinal α by Theorem 2.2.

- *Completeness:* We show that when $b \sqsubseteq u$, then $\exists$ has a winning strategy. In fact, in this case $\exists$ simply plays $l = u$, which satisfies $b \sqsubseteq u = f(u)$ and $\forall$ answers with some $b \ll u$, hence $b \sqsubseteq u$. The game can thus continue forever, leading to an infinite play which is won by $\exists$.

- *Soundness:* We show that whenever $b \not\sqsubseteq f^\beta(\top)$, for some ordinal β (i.e., b is not below some ν-approximant), then $\forall$ has a winning strategy, by transfinite induction on β. First observe that $\beta > 0$. In fact, otherwise $b \not\sqsubseteq f^0(\top) = \top$ would be a contradiction. Hence we distinguish two cases:

  - If β is a limit ordinal $b \not\sqsubseteq f^\beta(\top) = \bigsqcap_{\gamma<\beta} f^\gamma(\top)$, which means that there exists $\gamma < \beta$ such that $b \not\sqsubseteq f^\gamma(\top)$.

    Now any move of $\exists$ is some l with $b \sqsubseteq f(l)$. Therefore $l \not\sqsubseteq f^\gamma(\top)$, since otherwise $b \sqsubseteq f(l) \sqsubseteq f(f^\gamma(\top)) = f^{\gamma+1}(\top) \sqsubseteq f^\beta(\top)$ (since $\gamma+1 < \beta$). Hence there must be $b' \ll l$ with $b' \not\sqsubseteq f^\gamma(\top)$. Otherwise, as above, if for all $b' \ll l$ we had $b' \sqsubseteq f^\gamma(\top)$, then by continuity of the lattice, we would conclude $l = \bigsqcup\{b' \mid b' \ll l\} \sqsubseteq f^\gamma(\top)$. Such a b' can be chosen by $\forall$, and the game continues.
  - If $\beta = \gamma + 1$ we know that $b \not\sqsubseteq f^\beta(\top) = f(f^\gamma(\top))$.

    Any move of $\exists$ is l with $b \sqsubseteq f(l)$, which as above implies that $l \not\sqsubseteq f^\gamma(\top)$ and thus the existence of $b' \ll l$ with $b' \not\sqsubseteq f^\gamma(\top)$. The basis element b' is chosen by $\forall$ and the game continues.

  Hence $\forall$ always has a move, ending up in $b' \not\sqsubseteq f^\gamma(\top)$, from which there exists a winning strategy for $\forall$ by the induction hypothesis.

Observe that cases of a μ- and a ν-equation are not completely symmetric. In the completeness part, for showing that $l \sqsubseteq \nu f$ we use the fact that νf is the greatest post-fixpoint. Instead, for showing that $l \sqsubseteq \mu f$ we use the fact that $l \sqsubseteq f^{\alpha}(\bot)$ for some α and provide a proof that we can descend to $\bot$, similarly to what happens for ranking functions in termination analysis. Note that in order to guarantee that we truly descend, also below limit ordinals, we require that $\forall$ plays b with $b \ll l$. Then we can use the fact that whenever b is way-below a directed supremum, then it is smaller than one of the elements over which the supremum is taken. We remark that choosing b with $b \sqsubseteq l$ instead would not be sufficient (see Proposition 7.2). In the soundness part, despite the asymmetry, both proofs use the fact that each element is the supremum of all elements way-below it, for which it is essential to be in a continuous lattice (see Proposition 7.1). Instead, for completeness, the continuity hypothesis does not play a role.

For the general case, soundness and completeness of the game are proved by relying on the notions of μ- and ν-approximant. We prove the two properties separately. Completeness exploits a result that shows how $\exists$ can play descending along a chain of μ-approximants and, as in the case of a single equation, it can be proved for general lattices, without assuming the continuity hypothesis.

Lemma 7.6 (descending on μ-approximants)**.** *Let E be a system of m equations over a complete lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. For each μ-approximant $\boldsymbol{l} \in L^m$ and $(b,i) \in \mathbf{A}(\boldsymbol{l})$ there exists a μ-approximant $\boldsymbol{l}' \in \mathbf{E}(b,i)$ such that $ord(\boldsymbol{l}) \succeq_i ord(\boldsymbol{l}')$. Moreover, if $\eta_i = \mu$, the i-th component strictly decreases and thus the inequality is strict.*

Proof. Let $\boldsymbol{l} \in L^m$ be a μ-approximant and let (b,i) in $\mathbf{A}(\boldsymbol{l})$, i.e., $b \in B_L$ and $i \in [1,m]$ with $b \ll l_i$. We distinguish various cases:

1. ($\eta_i = \mu$) This means that $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ for some ordinal α. Since $f^{0}_{i,\boldsymbol{l}}(\bot) = \bot$ and $b \ll \bot$ would imply $b = \bot$, while $\bot \notin B_L$, necessarily $\alpha \neq 0$. We distinguish two subcases:

   (a) $\alpha = \beta + 1$ *is a successor ordinal*
   Let $l'_i = f^{\beta}_{i,\boldsymbol{l}}(\bot)$ and $(l'_1, \ldots, l'_{i-1}) = sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i])$. Then define

   $$\boldsymbol{l}' = (l'_1, \ldots, l'_{i-1}, l'_i, \boldsymbol{l}_{i+1,m})$$

   Observe that $\boldsymbol{l}'$ is a μ-approximant by Lemma 7.5. Moreover $\boldsymbol{l}' \in \mathbf{E}(b,i)$. In fact

   $$\begin{aligned} b \sqsubseteq l_i &= f^{\beta+1}_{i,\boldsymbol{l}}(\bot) \\ &= f_{i,\boldsymbol{l}}(f^{\beta}_{i,\boldsymbol{l}}(\bot)) \\ &= f_{i,\boldsymbol{l}}(l'_i) \\ &= f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i]), l'_i, \boldsymbol{l}_{i+1,m}) \\ &= f_i(l'_1, \ldots, l'_{i-1}, l'_i, \boldsymbol{l}_{i+1,m}) \\ &= f_i(\boldsymbol{l}') \end{aligned}$$

Finally, note that $ord(\boldsymbol{l}') \prec_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l}'$ coincide on the components $i+1, \ldots, m$, and $l_i = f_{i,\boldsymbol{l}}^{\beta+1}(\bot)$ while $l'_i = f_{i,\boldsymbol{l}}^{\beta}(\bot)$.

(b) *α is a limit ordinal*
Since $b \ll l_i = f_{h,\boldsymbol{l}}^{\alpha}(\bot) = \bigsqcup_{\beta<\alpha} f_{i,\boldsymbol{l}}^{\beta}(\bot)$, which is a directed supremum, by definition of the way-below relation, there is $\beta < \alpha$ such that $b \sqsubseteq f_{i,\boldsymbol{l}}^{\beta}(\bot)$. We set $l'_i = f_{i,\boldsymbol{l}}^{\beta}(\bot)$ and $(l'_1, \ldots, l'_{i-1}) = sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i])$. Then we define

$$\boldsymbol{l}' = (l'_1, \ldots, l'_{i-1}, l'_i, \boldsymbol{l}_{i+1,m})$$

The vector $\boldsymbol{l}'$ is a μ-approximant by Lemma 7.5. Moreover $\boldsymbol{l}' \in \mathbf{E}(b, i)$ since

$$\begin{aligned}
b &\sqsubseteq l'_i \\
&\sqsubseteq f_{i,\boldsymbol{l}}(l'_i) && \text{[since } l'_i = f_{i,\boldsymbol{l}}^{\beta}(\bot) \text{ is a post-fixpoint]} \\
&= f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l'_i]), l'_i, \boldsymbol{l}_{i+1,m}) \\
&= f_i(l'_1, \ldots, l'_{i-1}, l'_i, \boldsymbol{l}_{i+1,m}) \\
&= f_i(\boldsymbol{l}')
\end{aligned}$$

Finally, note that $ord(\boldsymbol{l}') \prec_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l}'$ coincide on the components $i+1, \ldots, m$, and $l_i = f_{i,\boldsymbol{l}}^{\alpha}(\bot)$ while $l'_i = f_{i,\boldsymbol{l}}^{\beta}(\bot)$, with $\beta < \alpha$.

2. $(\eta_i = \nu)$
In this case $l_i = \nu(f_{i,\boldsymbol{l}})$. Let $(l'_1, \ldots, l'_{i-1}) = sol(E[\boldsymbol{x}_{i,m} := \boldsymbol{l}_{i,m}])$. Then define

$$\boldsymbol{l}' = (l'_1, \ldots, l'_{i-1}, \boldsymbol{l}_{i,m})$$

The vector $\boldsymbol{l}'$ is a μ-approximant by Lemma 7.5. Moreover, observe that $\boldsymbol{l}' \in \mathbf{E}(b, i)$, since

$$\begin{aligned}
b &\sqsubseteq l_i \\
&= f_{i,\boldsymbol{l}}(l_i) && \text{[since } l_i \text{ is a fixpoint]} \\
&= f_i(sol(E[\boldsymbol{x}_{i,m} := \boldsymbol{l}_{i+1,m}]), \boldsymbol{l}_{i,m}) \\
&= f_i(l'_1, \ldots, l'_{i-1}, \boldsymbol{l}_{i,m}) \\
&= f_i(\boldsymbol{l}')
\end{aligned}$$

Finally, note that $ord(\boldsymbol{l}') \preceq_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l}'$ coincide on the components $i, \ldots, m$.

□

The previous result allows us to prove that player $\exists$ can always win starting from a μ-approximant. Roughly, relying on Lemma 7.6, we can prove that player $\exists$ can play on

μ-approximants in a way that each time the i-th equation is chosen, the ordinal vector associated with the approximant decreases with respect to $\preceq_i$, and it strictly decreases when the i-th equation is a μ-equation. This, together with the fact that the order on ordinals is well-founded, allows one to conclude that either the play is finite and $\exists$ plays last or the highest index on which one can cycle is necessarily the index of a ν-equation. In both cases player $\exists$ wins.

Lemma 7.7 ($\exists$ wins on μ-approximants)**.** *Let E be a system of m equations over a complete lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $\boldsymbol{l} \in L^m$ be a μ-approximant. Then in a game starting from $\boldsymbol{l}$ (which is a position of $\forall$) player $\exists$ has a winning strategy.*

Proof. We first describe the strategy for player $\exists$ and then prove that it is a winning strategy.

The key observation is that $\exists$ can always play a μ-approximant, where she plays the solution in the first step. In fact, let $\boldsymbol{l}' \in L^m$ be the current μ-approximant. For any possible move $(b', i') \in \mathbf{A}(\boldsymbol{l}')$ of $\forall$, by Lemma 7.6 there always exists a move $\boldsymbol{l}'' \in \mathbf{E}(b', i')$ of $\exists$ which is a μ-approximant such that $ord(\boldsymbol{l}') \succeq_i ord(\boldsymbol{l}'')$. Additionally, if $\eta_i = \mu$ the inequality is strict.

Since $\exists$ player has always a move, either the play finishes because $\forall$ has no moves, hence $\exists$ wins or the play continues forever.

In this last case, note that, if h is the largest index occurring infinitely often, then necessarily $\eta_h = \nu$, hence $\exists$ wins. In fact, assume by contradiction that $\eta_h = \mu$. Consider the sequence of turns of the play starting from the point where all indices repeat infinitely often.

Let $\boldsymbol{l}'$, (b', j), $\boldsymbol{l}''$ be consecutive turns. By the choice of h, necessarily $j \leq h$. Moreover, by construction, if

$$ord(\boldsymbol{l}') \succeq_j ord(\boldsymbol{l}'')$$

Observing that for $j \leq j'$ it holds $\boldsymbol{\alpha} \succeq_j \boldsymbol{\alpha}'$ implies $\boldsymbol{\alpha} \succeq_{j'} \boldsymbol{\alpha}'$, we deduce that

$$ord(\boldsymbol{l}') \succeq_h ord(\boldsymbol{l}'')$$

i.e., the sequence is decreasing. Moreover, since $\eta_h = \mu$, whenever $j = h$, $ord(\boldsymbol{l}') \succ_h ord(\boldsymbol{l}'')$, i.e., the sequence strictly decreases. This contradicts the well-foundedness of $\succ_h$. □

Since the solution of a system of equation is a μ-approximant (the greatest one), completeness is an easy corollary of Lemma 7.7.

Corollary 7.1 (completeness)**.** *Let E be a system of m equations over a complete lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given any μ-approximant $\boldsymbol{l} \in L^m$, $b \in B_L$ and $i \in [1, m]$, if $b \sqsubseteq l_i$ then $\exists$ has a winning strategy from position (b, i).*

Proof. Just observe that at the first turn $\exists$ can play the μ-approximant $\boldsymbol{l}$ that is in $\mathbf{E}(b, i)$ by hypotheses. Then using Lemma 7.7 we conclude that $\exists$ wins. □

For soundness we rely on a result, dual to Lemma 7.6, that allows to ascend along ν-approximants. However, in this case, the fact of working in a continuous lattice is crucial (see Proposition 7.1).

Lemma 7.8 (ascending on ν-approximants)**.** *Let E be a system of m equations over a continuous lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given a ν-approximant $\boldsymbol{l} \in L^m$, an element $b \in B_L$ and an index $i \in [1, m]$ with $b \not\sqsubseteq l_i$, for all tuples $\boldsymbol{l'} \in \mathbf{E}(b, i)$ there are a ν-approximant $\boldsymbol{l''}$ and $(b'', j) \in \mathbf{A}(\boldsymbol{l'})$ such that (1) $b'' \not\sqsubseteq l''_j$ and (2) $ord(\boldsymbol{l}) \succeq_i ord(\boldsymbol{l''})$. Moreover, if $\eta_i = \nu$, the i-th component strictly decreases and thus the inequality in item 2 above is strict.*

Proof. Let $\boldsymbol{l} \in L^m$ be a ν-approximant, let $b \in B_L$ and let $i \in [1, m]$ with $b \not\sqsubseteq l_i$. Take $\boldsymbol{l'} \in \mathbf{E}(b, i)$, i.e., such that $b \sqsubseteq f_i(\boldsymbol{l'})$. We prove that there are a ν-approximant $\boldsymbol{l''}$ and $(b'', j) \in \mathbf{A}(\boldsymbol{l'})$ satisfying (1) and (2) above, by distinguishing various cases:

i. ($\eta_i = \mu$) Define $\boldsymbol{l''} = (sol(E[\boldsymbol{x}_{i,m} := \boldsymbol{l}_{i,m}]), l_i, \boldsymbol{l}_{i+1,m})$, which is a ν-approximant by Lemma 7.5. Note that, since $l_i = \mu(f_{i,\boldsymbol{l}})$,

$$l_i = f_{i,\boldsymbol{l}}(l_i) = f_i(sol(E[\boldsymbol{x}_{i,m} := \boldsymbol{l}_{i,m}]), l_i, \boldsymbol{l}_{i+1,m}) = f_i(\boldsymbol{l''})$$

We first prove (1), i.e., that there exists $(b'', j) \in \mathbf{A}(\boldsymbol{l'})$, i.e., $j \in [1, m]$ and $b'' \in B_L$, $b'' \ll l'_j$ with $b'' \not\sqsubseteq l''_j$. In fact, otherwise, if for any j and $b'' \ll l'_j$ we had $b'' \sqsubseteq l''_j$, then for any j, since B_L is a basis and L a continuous lattice:

$$l'_j = \bigsqcup\{b'' \mid b'' \in B_L \wedge\ b'' \ll l'_j\} \sqsubseteq l''_j.$$

However, by monotonicity of f_i, this would imply $f_i(\boldsymbol{l'}) \sqsubseteq f_i(\boldsymbol{l''}) = l_i$, that together with the hypothesis $b \sqsubseteq f_i(\boldsymbol{l'})$, would contradict $b \not\sqsubseteq l_i$.

For point (2), note that $ord(\boldsymbol{l'}) \preceq_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l'}$ coincide on all components $i, \ldots, m$, and $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$.

ii. ($\eta_i = \nu$) This means that $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\top)$ for some ordinal α, necessarily $\alpha \neq 0$ (since otherwise $l_i = \top$ and $b \not\sqsubseteq l_i$ could not hold). We distinguish two subcases

(a) $\alpha = \beta + 1$ *is a successor ordinal*
Define $\boldsymbol{l''} = (sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := f^{\beta}_{i,\boldsymbol{l}}(\top)]), f^{\beta}_{i,\boldsymbol{l}}(\top), \boldsymbol{l}_{i+1,m})$. Then we have

$$\begin{aligned} b &\not\sqsubseteq l_i \\ &= f^{\beta+1}_{i,\boldsymbol{l}}(\top) \\ &= f_{i,\boldsymbol{l}}(f^{\beta}_{i,\boldsymbol{l}}(\top)) \\ &= f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := f^{\beta}_{i,\boldsymbol{l}}(\top)]), f^{\beta}_{i,\boldsymbol{l}}(\top), \boldsymbol{l}_{i+1,m}) \\ &= f_i(\boldsymbol{l''}) \end{aligned}$$

Recalling that $b \sqsubseteq f_i(\boldsymbol{l}')$, as in case (i) we deduce point (1), i.e., that there exists $(b'', j) \in \mathbf{A}(\boldsymbol{l}')$ such that $b'' \not\sqsubseteq l''_j$.

Concerning point (2), note that $ord(\boldsymbol{l}') \prec_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l}'$ coincide on the components $i+1, \ldots, m$, and $l_i = f_{i,\boldsymbol{l}}^{\beta+1}(\bot)$ while $l'_i = f_{i,\boldsymbol{l}}^{\beta}(\bot)$.

(b) *α is a limit ordinal*
In this case

$$b \not\sqsubseteq l_i = f_{i,\boldsymbol{l}}^{\alpha}(\top) = \bigsqcap_{\beta<\alpha} f_{i,\boldsymbol{l}}^{\beta}(\top) = \bigsqcap_{\beta<\alpha} f_{i,\boldsymbol{l}}^{\beta+1}(\top)$$

Therefore there exists $\beta < \alpha$ such that $b \not\sqsubseteq f_{i,\boldsymbol{l}}^{\beta+1}(\top)$. Hence, we can define $l''_i = f_{i,\boldsymbol{l}}^{\beta}(\top)$ and take the ν-approximant

$$\boldsymbol{l}'' = (sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := l''_i]), l''_i, \boldsymbol{l}_{i+1,m})$$

Then we have

$$\begin{aligned} b &\not\sqsubseteq f_{i,\boldsymbol{l}}^{\beta+1}(\top) \\ &= f_{i,\boldsymbol{l}}(f_{i,\boldsymbol{l}}^{\beta}(\top)) \\ &= f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := f_{i,\boldsymbol{l}}^{\beta}(\top)]), f_{i,\boldsymbol{l}}^{\beta}(\top), \boldsymbol{l}_{i+1,m}) \\ &= f_i(\boldsymbol{l}'') \end{aligned}$$

and thus, again, recalling that $b \sqsubseteq f_i(\boldsymbol{l}')$, as in case (i) we deduce point (1), i.e., that there exists $(b'', j) \in \mathbf{A}(\boldsymbol{l}')$ such that $b'' \not\sqsubseteq l''_j$.
Concerning point (2), note that $ord(\boldsymbol{l}'') \prec_i ord(\boldsymbol{l})$ since vectors $\boldsymbol{l}$ and $\boldsymbol{l}''$ coincide on the components $i+1, \ldots, m$, and $l_i = f_{i,\boldsymbol{l}}^{\alpha}(\bot)$ while $l''_i = f_{i,\boldsymbol{l}}^{\beta}(\bot)$, with $\beta < \alpha$.

□

As in the dual case, soundness is an easy corollary of the above lemma, recalling that the solution is the least ν-approximant.

Lemma 7.9 (soundness)**.** *Let E be a system of m equations over a continuous lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. For a ν-approximant $\boldsymbol{l} \in L^m$, $b \in B_L$ and $i \in [1, m]$, if $b \not\sqsubseteq l_i$ then $\forall$ has a winning strategy from position (b, i).*

Proof. We first describe the strategy for the universal player and then prove that it is a winning strategy.

Let $\boldsymbol{l} \in L^m$ be a ν-approximant, $b \in L$ and $i \in [1, m]$ such that $b \not\sqsubseteq l_i$. Starting from (b, i), for any possible move $\boldsymbol{l}' \in \mathbf{E}(b, i)$ of $\exists$. Then $\forall$ can play a pair $(b', j) \in \mathbf{A}(\boldsymbol{l}')$, whose existence is ensured by Lemma 7.8, such that there is a ν-approximant $\boldsymbol{l}''$ satisfying $b'' \not\sqsubseteq l''_j$ and $ord(\boldsymbol{l}'') \prec_i ord(\boldsymbol{l})$. Additionally, if $\eta_i = \nu$ the inequality is strict.

According to the strategy defined above $\forall$ player has always a move. Thus either the play finishes because $\exists$ has no moves, hence $\forall$ wins or the play continues forever.

In this last case, with an argument dual with respect to that in Lemma 7.7, we can show that if h is the largest index occurring infinitely often, then necessarily $\eta_h = \mu$, hence $\forall$ win. □

Combining Corollary 7.1 and Lemma 7.9 we reach the desired result.

Theorem 7.1 (soundness and completeness)**.** *Given a system of m equations E over a continuous lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{u}$, then for all $b \in B_L$ and $i \in [1, m]$, it holds $b \sqsubseteq u_i$ iff $\exists$ has a winning strategy from position (b, i).*

Proof. Immediate corollary of Lemma 7.4, Corollary 7.1 and Lemma 7.9. □

Note that even when the fixpoint is reached in more than ω steps, thanks to the fact that the order on the ordinals is well-founded and players "descend" over the order, ordinals do not play an explicit role in the game. In particular plays are not transfinite and whenever $\forall$ or $\exists$ wins due to the fact that the other player cannot make a move, this happens after a *finite* number of steps. This can be a bit surprising at first since the game works for general continuous lattices, including, for instance, intervals over the reals.

We close this section by proving two results that, in a sense, show that the choice of continuous lattices and the way-below relation are "the right ones" for the game. We first observe that the restriction to continuous lattices is not only sufficient but also necessary for soundness of the game.

Proposition 7.1 (soundness holds exactly in continuous lattices)**.** *Let L be a complete lattice. The game is sound for every system of equations over L if and only if L is continuous.*

Proof. We already know from Lemma 7.9 that when L is continuous the game is sound.

Conversely, let L be a non-continuous lattice. This means that there is an element $l \in L$ such that $l \neq \bigsqcup \downdownarrows l$. Note that since $\downdownarrows l \subseteq \downarrow l$, we have $\bigsqcup \downdownarrows l \sqsubset \bigsqcup \downarrow l = l$. We prove that, for any basis B_L for L such that $\bot \notin B_L$, there are a monotone function $f : L \to L$ and an element $b \in B_L$ such that $b \not\sqsubseteq \mu f$, for which there is a winning strategy for the existential player for the corresponding fixpoint game starting from position b, while such a strategy should not exists. The function f is defined by:

$$f(x) = \begin{cases} \bigsqcup \downdownarrows l & \text{if } x \sqsubseteq \bigsqcup \downdownarrows l \\ \top & \text{otherwise.} \end{cases}$$

Notice that necessarily $\bigsqcup \downdownarrows l \neq \top$, since $\bigsqcup \downdownarrows l \sqsubset l \sqsubseteq \top$. Then, clearly f is monotone and its least fixpoint is $\mu f = \bigsqcup \downdownarrows l$. Moreover, since B_L is a basis, we know that $\bigsqcup (B_L \cap \downarrow l) = l$. Then there must be $b \in B_L$ such that $b \sqsubseteq l$ but $b \not\sqsubseteq \bigsqcup \downdownarrows l$. Otherwise, if for all $b \in B_L$ such that $b \sqsubseteq l$, $b \sqsubseteq \bigsqcup \downdownarrows l$, then we would have $\bigsqcup (B_L \cap \downarrow l) \sqsubseteq \bigsqcup \downdownarrows l$, contradicting the hypothesis $\bigsqcup \downdownarrows l \sqsubset l = \bigsqcup (B_L \cap \downarrow l)$. Now we show that player $\exists$ is able to win any play of the game, with a single equation $x =_\mu f(x)$, for checking whether such a $b \sqsubseteq \mu f$. The strategy is

actually quite simple and can be described by just one family of plays. We use the fact that $f(l) = \top$.

$$b \overset{\exists}{\rightsquigarrow} l \overset{\forall}{\rightsquigarrow} b' \overset{\exists}{\rightsquigarrow} \bot \overset{\forall}{\not\rightsquigarrow}$$

for any $b' \ll l$, since $b' \sqsubseteq \bigsqcup \mathop{\downarrow} l = f(\bot)$. Thus player $\exists$ can always win, despite the fact that $b \not\sqsubseteq \mu f$. □

As a counterexample, consider the lattice W in Figure 7.2, which is not continuous and let B_W be any basis such that $0 \notin B_W$. First note that necessarily $a \in B_W$, otherwise $a \neq \bigsqcup(B_W \cap \mathop{\downarrow} a) = \bigsqcup \emptyset = 0$. Secondly, $\mathop{\downarrow} a = \{0\}$ since $a \not\ll a$. Then, consider the equation $x =_\mu f(x)$, where the function $f : W \to W$ is defined by $f(0) = 0$, and $f(x) = \omega$ for $x \neq 0$. Clearly f is monotone and its least fixpoint is $\mu f = 0$. However, the player $\exists$ can win any play of the game from position a, despite the fact that $a \not\sqsubseteq \mu f = 0$. In fact, the first move of $\exists$ can be a, since $a \sqsubseteq f(a) = \omega$. But then player $\forall$ has no moves since $B_W \cap \mathop{\downarrow} a = \emptyset$. And so player $\exists$ always wins while she should not.

The second observation is that using the lattice order instead of the way-below relation may break completeness. More precisely, consider the natural variant of the game where the way-below relation is replaced by the lattice order. Let us call it *weak game*. Since the set of possible moves of player $\forall$ is enlarged, soundness clearly continues to hold. Instead, as we hinted before, completeness could fail. We show that it is exactly on algebraic lattices that completeness still holds for the weak game.

Proposition 7.2 (way-below is needed in non-algebraic lattices)**.** *Let L be a complete lattice. The weak game is complete on every system of equations over L if and only if B_L consists of compact elements (which in turn means that L is algebraic).*

Proof. Let K_L be the set of compact elements of L. If $B_L \subseteq K_L$, then for any $b \in B_L$ and $l \in L$, we have that $b \ll l$ if and only if $b \sqsubseteq l$. Therefore the weak game coincides with the original one and hence completeness clearly holds.

Conversely, assume that $B_L \not\subseteq K_L$. We show that we can identify a system consisting of a single equation $x =_\mu f(x)$ for which the weak game is not complete. Let $b \in B_L \setminus K_L$ be a non-compact element in the basis. Therefore $b \not\ll b$ which means that there exists a directed set D such that $b \sqsubseteq \bigsqcup D$ and $b \not\sqsubseteq d$ for all $d \in D$. Without loss of generality we can assume that D is a transfinite chain $D = (d_\alpha)_\alpha$ (see, e.g., [Mar76, Theorem 1]).

Consider the function $f : L \to L$ defined as

$$f(x) = \begin{cases} \top & \text{if } b \sqsubseteq x \\ d_\alpha & \text{otherwise, where } \alpha = \min\{\beta \mid d_\beta \not\sqsubseteq x\} \end{cases}$$

Observe that the function f is well-defined. In fact, when $b \not\sqsubseteq x$ there exists β such that $d_\beta \not\sqsubseteq x$ and hence the set $\{\beta \mid d_\beta \not\sqsubseteq x\}$ is not empty. In fact, if we had $d_\beta \sqsubseteq x$ for all elements of the chain, we would deduce $\bigsqcup D \sqsubseteq x$ and thus, recalling $b \sqsubseteq \bigsqcup D$, we would conclude $b \sqsubseteq x$.

Observe that f is monotone. In fact, let $x, y \in L$ with $x \sqsubseteq y$. If $b \sqsubseteq y$ and thus $f(y) = \top$, we trivially conclude $f(x) \sqsubseteq \top = f(y)$. Let us then consider the case in which $b \not\sqsubseteq y$ and thus $f(y) = d_\alpha$ where $\alpha = \min\{\beta \mid d_\beta \not\sqsubseteq y\}$. Obviously $b \not\sqsubseteq x$ and thus $f(x) = d_{\alpha'}$, where $\alpha' = \min\{\beta \mid d_\beta \not\sqsubseteq x\}$. Since $x \sqsubseteq y$, we have $\{\beta \mid d_\beta \not\sqsubseteq x\} \supseteq \{\beta \mid d_\beta \not\sqsubseteq y\}$, hence $\alpha' \leq \alpha$ and thus $f(x) = d_{\alpha'} \sqsubseteq d_\alpha = f(y)$, as desired.

By construction $\top$ is the only fixpoint of f, hence $\top = \mu f$. Thus $b \sqsubseteq \mu f = \top$. Now, if we play the weak game, since $b \not\sqsubseteq d_\alpha$ for all α, the possible moves for $\exists$ are initially only those $x \in L$ such that $b \sqsubseteq x$ and thus $b \sqsubseteq f(x) = \top$. However, if $\exists$ play such an x, in the weak game $\forall$ can answer b, getting back to the initial situation. Hence $\forall$ wins, providing the desired counterexample to completeness. □

Note that when the elements of the basis are compact, the way-below relation with respect to elements of the basis is the lattice order. Hence the result above essentially states that the weak game is complete exactly when it coincides with the original game, thus further supporting the appropriateness of our formulation of the game.

As a counterexample, consider the continuous lattice corresponding to the interval of real numbers $[0,1]_\mathbb{R}$ with the usual order and basis $B = \mathbb{Q} \cap [0,1]_\mathbb{R} \setminus \{0\}$. Recall that $[0,1]_\mathbb{R}$ is not algebraic (the only compact element is 0) and way-below relation is the strict order $<$. Let $g : [0,1]_\mathbb{R} \to [0,1]_\mathbb{R}$ be the function defined by $g(x) = \frac{x+1}{2}$. The fixpoint equation $x =_\mu g(x)$ has solution $\mu g = 1$.

In the weak game, from position $l \in [0,1]_\mathbb{R}$, player $\forall$ can play any $b \leq l$ (instead, of $b < l$). Then player $\exists$ loses any play starting from position 1, despite the fact that $1 \leq \mu g = 1$. In fact, the only possible move of player $\exists$ is 1, and $\forall$ can play any $x \leq 1$. In particular, playing 1 the game will continue forever and will thus be won by $\forall$.

Notice that, instead, in the original game, from position 1, player $\forall$ has to play an element $1 - \epsilon$ for some $\epsilon > 0$. Then, it is easy to see that at each step i player $\exists$ will be able to play some $z_i \leq 1 - 2^i\epsilon$. This means that after finitely many steps $\exists$ will be allowed to play 0, thus leaving no possible answer to $\forall$ and winning the game.

7.4 Progress measures

Along the lines of [Jur00], influenced by [HSC16], in this section we introduce a general notion of progress measure for fixpoint games over continuous lattices. We will show how a complete progress measure characterises the winning positions for the two players. The existence of a so-called small progress measure will allow us to express a complete progress measure as a least fixpoint, thus providing a technique for computing the progress measure and solving the corresponding system of equations.

Given an ordinal α we denote by $[\alpha]^m_\star = \{\beta \mid \beta \leq \alpha\}^m \cup \{\star\}$, the set of ordinal vectors with entries smaller or equal than α, with an added bound $\star$ being the top element.

Definition 7.13 (progress measure)**.** *Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given an ordinal λ, a* λ-progress measure *for E is a function $R : B_L \to [1,m] \to [\lambda]^m_\star$ such that for all $b \in B_L$, $i \in [1,m]$, either $R(b)(i) = \star$ or there exists $\boldsymbol{l} \in \mathbf{E}(b,i)$ such that for all $(b',j) \in \mathbf{A}(\boldsymbol{l})$ it holds*

- *if $\eta_i = \mu$ then $R(b)(i) \succ_i R(b')(j)$*
- *if $\eta_i = \nu$ then $R(b)(i) \succeq_i R(b')(j)$.*

A progress measure maps any basis element of the lattice and index $i \in [1,m]$ to an m-tuple of ordinals, with one component for each equation. Components relative to μ-equations roughly measure how many unfolding steps for the equation would be needed to reach an under-approximation l_i above b, and thus, for $\exists$, to win the game. Components relative to ν-equations, as in the original work of [Jur00], are less relevant, as we will see.

Intuitively, whenever $R(b)(i) \neq \star$, the progress measure R provides an evidence of the existence of a winning strategy for $\exists$ in a play starting from (b,i). The tuple $\boldsymbol{l}$, whose existence is required by the definition, is a move of player $\exists$ such that for any possible answer of $\forall$, the progress measure will not increase with respect to $\preceq_i$, and it will strictly decrease in the case of μ-equations. Since $\prec_i$ is well-founded, this ensures that we cannot cycle on a μ-equation. Also note that whenever the current index is i, all indices lower than i are irrelevant (expressed by the orders $\succeq_i$ resp. $\succ_i$), which is related to the fact that the highest index which is visited infinitely often is the only relevant index for determining the winner of the game. This idea is formalised in the following lemma.

Lemma 7.10 (progress measures are strategies)**.** *Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{u}$. For any $b \in B_L$ and $i \in [1,m]$, if there exists some ordinal λ and a λ-progress measure R such that $R(b)(i) \preceq_i (\lambda,\ldots,\lambda)$, then $b \sqsubseteq u_i$.*

Proof. We show that $\exists$ has a winning strategy from (b,i). The strategy consists in choosing a move $\boldsymbol{l} \in \mathbf{E}(b,i)$ such that for all $(b',j) \in \mathbf{A}(\boldsymbol{l})$, it holds

- $R(b)(i) \succ_i R(b')(j)$, if $\eta_i = \mu$
- $R(b)(i) \succeq_i R(b')(j)$, if $\eta_i = \nu$

which exists by definition of progress measure.

Now, observe that player $\exists$ can always take its turn. Therefore either the play stops because $\forall$ runs out of moves, hence $\exists$ win. Otherwise, the play is infinite, and, if we denote by h the largest index occurring infinitely often, then $\eta_h = \nu$, hence $\exists$ wins. In fact, assume by contradiction that $\eta_h = \mu$. Consider the sequence of turns of the play starting from the point where all indices repeat infinitely often and take the m-tuples of ordinals $R(b')(h)$ corresponding to the positions (b',i) where $\exists$ plays. For any two successive elements, say (b',i) and (b'',j), by construction

$$R(b')(i) \succeq_i R(b'')(j)$$

Observing that for $i \leq j$ it holds $\boldsymbol{\alpha} \succeq_i \boldsymbol{\alpha}'$ implies $\boldsymbol{\alpha} \succeq_j \boldsymbol{\alpha}'$, we deduce that

$$R(b')(i) \succeq_h R(b'')(j)$$

i.e., the sequence is decreasing. Moreover, since $\eta_h = \mu$, whenever $i = h$, $R(b')(i) \succ_h R(b'')(j)$, i.e., the sequence strictly decreases. This contradicts well-foundedness of $\prec_h$. $\square$

The above lemma, in a sense, says that progress measures provide sound characterisations of the solution. However, in general, they are not complete, since whenever $R(b)(i) = \star$ we cannot derive any information on (b, i), i.e., if $\boldsymbol{u}$ is the solution of the system, we cannot conclude that $b \not\sqsubseteq u_i$. This motivates the following definition.

Definition 7.14 (complete progress measures)**.** *Let L be a continuous lattice and let E be a system of equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{u}$. A λ-progress measure $R : B_L \to [1, m] \to [\lambda]^m_\star$ is called* complete *if for all $b \in B_L$ and $i \in [1, m]$, if $b \sqsubseteq u_i$ then $R(b)(i) \preceq_i (\lambda, \ldots, \lambda)$.*

Observe that in search of a complete progress measure, in principle, we would have to try all ordinals as a bound. We next show that we can take as bound the height λ_L of the lattice L. This provides a generalisation of the *small progress measure* in [Jur00].

Definition 7.15 (small progress measure)**.** *Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given an m-tuple of ordinals $\boldsymbol{\alpha}$, let us denote by $z_E(\boldsymbol{\alpha})$ the m-tuple of ordinals where ν-components are set to 0, i.e., $z_E(\boldsymbol{\alpha}) = \boldsymbol{\beta}$ with $\beta_i = \alpha_i$ if $\eta_i = \mu$, and $\beta_i = 0$ otherwise. We define the* small progress measure *$R_E : B_L \to [1, m] \to [\lambda_L]^m_\star$ by*

$$R_E(b)(i) = \min_{\preceq_i}\{z_E(\mathit{ord}(\boldsymbol{l})) \mid \boldsymbol{l} \text{ is a } \mu\text{-approximant } \wedge\ \boldsymbol{l} \in \mathbf{E}(b, i)\}$$

where $\min_{\preceq_i}$ is the minimum on $\preceq_i$ as given in Definition 7.6, with the convention that $\min_{\preceq_i} \emptyset = \star$.

Observe that R_E is well-defined, i.e., it actually takes values in $[\lambda_L]^m_\star$. In fact, the components of $z_E(\mathit{ord}(\boldsymbol{l}))$ corresponding to μ-indices are ordinals expressing the number of Kleene's iterations needed to reach under-approximations of the least fixpoint. These are clearly bounded by λ_L, since for a monotone function $f : L \to L$, the sequence $f^\alpha(\bot)$ is strictly increasing until it reaches the least fixpoint of f. For ν-indices, instead, $z_E(\mathit{ord}(\boldsymbol{l}))$ is always 0.

Observe that while formally $R_E(b)(i)$ takes values in $[\lambda_L]^m_\star$, whenever $j < i$ or $\eta_j = \nu$, due to the effect of the $\min_{\preceq_i}$ and of the z_E operations, the only possible value for the j-th component is 0. Despite being irrelevant, we keep such components for notational convenience.

The fact that R_E is indeed a progress measure follows from Lemma 7.6. Moreover, we can easily show that it is complete.

Lemma 7.11 (small progress measure)**.** *Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Then $R_E : B_L \to [1,m] \to [\lambda_L]^m_\star$ is a progress measure and it is complete.*

Proof. For the first part, let $R_E(b)(i) = \boldsymbol{\alpha} \neq \star$. Hence $R_E(b)(i) =_i z_E(ord(\boldsymbol{l}))$ for some μ-approximant $\boldsymbol{l}$ such that $\boldsymbol{l} \in \mathbf{E}(b,i)$. By Lemma 7.6, for all $(b',j) \in \mathbf{A}(\boldsymbol{l})$ there exists $\boldsymbol{l}' \in \mathbf{E}(b',j)$ such that $ord(\boldsymbol{l}) \succeq_i ord(\boldsymbol{l}')$ and, if $\eta_i = \mu$, the inequality is strict since the i-th component strictly decreases. Clearly, this implies $z_E(ord(\boldsymbol{l})) \succeq_i z_E(ord(\boldsymbol{l}'))$. Additionally, if $\eta_i = \mu$, the inequality remains strict since the i-th component is left unchanged by the z_E operation.

Therefore, by definition of R_E, for all $(b',j) \in \mathbf{A}(\boldsymbol{l})$ we have

$$R_E(b')(j) \preceq_i z_E(ord(\boldsymbol{l}')) \preceq_i z_E(ord(\boldsymbol{l})) =_i R_E(b)(i). \tag{7.6}$$

where, if $\eta_i = \mu$, the inequality is strict, as desired.

Let us now show that R_E is complete. Let $b \in B_L$ be such that $b \sqsubseteq u_i$. We know that the solution $\boldsymbol{u}$ is a μ-approximant. Moreover, since $b \sqsubseteq u_i = f_i(\boldsymbol{u})$, we have that $\boldsymbol{u} \in \mathbf{E}(b,i)$. Hence $R_E(b)(i) \preceq_i z_E(ord(\boldsymbol{u}))$ and thus $R_E(b)(i) \neq \star$. □

Here we show that a complete progress measure can be characterised as the least solution of a system of equations over tuples of ordinals, naturally induced by Definition 7.13.

Definition 7.16 (progress measure equations)**.** *Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Let $\boldsymbol{\delta}^{\eta}_i$, with $i \in [1,m]$, be, for $\eta = \nu$, the null vector and, for $\eta = \mu$, the vector where all components are 0 except the i-th which is 1. The* progress measure equations *for E over the lattice $[\lambda_L]^m_\star$, are defined, for $b \in B_L$, $i \in [1,m]$, as:*

$$R(b)(i) = \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in \mathbf{E}(b,i)\}$$

We will denote by Φ_E the corresponding endofunction on $L \to [1,m] \to [\lambda_L]^m_\star$ which is defined, for $R : B_L \to [1,m] \to [\lambda_L]^m_\star$, by

$$\Phi_E(R)(b)(i) = \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in \mathbf{E}(b,i)\}$$

Observe that, since $[\lambda_L]^m_\star$ is a complete lattice, also the corresponding set of progress measures, endowed with pointwise $\preceq$-order, is a complete lattice. It is immediate to see that Φ_E is monotone with respect to such order, i.e., if $R \preceq R'$ pointwise then $\Phi_E(R) \preceq \Phi_E(R')$ pointwise. This allows us to obtain a complete progress measure as a (least) fixpoint of Φ_E.

Lemma 7.12 (complete progress measure as a fixpoint)**.** *Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Then the least solution R_M of the progress measure equations (least fixpoint of Φ_E with respect to $\preceq$) is the least λ_L-progress measure, hence it is smaller than R_E and it is complete.*

Proof. We first observe that λ_L-progress measures R are all and only pre-fixpoints of Φ_E. This implies that R_M, which is the least pre-fixpoint, is the least progress measure.

In fact, if R is a pre-fixpoint, i.e., for all $b \in B_L$, $i \in [1,m]$, $\Phi_E(R)(b)(i) \preceq R(b)(i)$, which implies $\Phi_E(R)(b)(i) \preceq_i R(b)(i)$. Then, for $b \in B_L$ and $i \in [1,m]$, if $R(b)(i) \neq \star$, necessarily $\Phi_E(R)(b)(i) \neq \star$. Hence we can take $\boldsymbol{l} \in \mathbf{E}(b,i)$ that realises the minimum in the definition of $\Phi_E(R)(b)(i)$, namely such that $\Phi_E(R)(b)(i) = \sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\}$ and we have that for all $(b',j) \in \mathbf{A}(\boldsymbol{l})$

$$R(b)(i) \succeq_i \Phi_E(R)(b)(i) \succeq R(b')(j) + \boldsymbol{\delta}_i^{\eta_i}$$

which amounts to the validity of the progress measure property (it gives strict inequality for $\eta_i = \mu$ and general inequality for $\eta_i = \nu$).

Conversely, let R be a progress measure. We have to show that for all $b \in B_L$, $i \in [1,m]$

$$R(b)(i) \succeq_i \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in \mathbf{E}(b,i)\} \tag{7.7}$$

Given $b \in L$, $i \in [1,m]$, by definition of progress measure, there is $\boldsymbol{l} \in \mathbf{E}(b,i)$ such that for all $(b',j) \in \mathbf{A}(\boldsymbol{l})$, it holds $R(b)(i) \succeq_i R(b')(j)$, with strict inequality if $\eta_i = \mu$. This can be equivalently stated $R(b)(i) \succeq_i R(b')(j) + \boldsymbol{\delta}^{\eta_i}$. Hence $R(b)(i) \succeq_i \sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\}$. Namely, $R(b)(i)$ is larger than an element of the set of which we take the minimum, hence (7.7) immediately follows. Since in the right-hand side all entries with an index below i are 0, we even have $\succeq$ (instead of $\succeq_i$ in (7.7), which implies that R is a pre-fixpoint of Φ_E.

For completeness, recall that by Lemma 7.11, R_E is a λ_L-progress measure and it is complete. Therefore for all $b \in B_L$ and $i \in [1,m]$, we have $R_M(b)(i) \preceq R_E(b)(i)$, from which completeness of R_M immediately follows. □

Observe that, since $R_M \preceq R_E$, in particular, for all $b \in B_L$ and $i \in [1,m]$, if $R_M(b)(i) \neq \star$, then all components of $R_M(b)(i)$ corresponding to ν-indices are 0.

If we consider the system of equations in Figure 7.3c we obtain as least fixpoint the progress measure

$$\begin{aligned} R_M(\{M_1\})(1) &= (1,0) \\ R_M(\{M_1\})(2) &= R_M(\{M_2\})(1) = R_M(\{M_2\})(2) = (0,0). \end{aligned}$$

Note that R_M never assumes the top value $\star$, consistently with the fact that the solution is $(u_1,u_2) = (\mathcal{R}(\mathcal{N}),\mathcal{R}(\mathcal{N}))$. We will discuss how such R_M is obtained later when providing a more "efficient" way for computing it.

We next observe that the operator Φ_E creates monotone functions and, applied to functions that respect suprema, it produces functions enjoying the same property. To avoid confusion, note that the notion of monotonicity for a function $R : B_L \to [1,m] \to [\lambda_L]^m_\star$ is the standard one, with respect to the pointwise order on $[1,m] \to [\lambda_L]^m_\star$. Explicitly, R is monotone if for all $b,b' \in B_L$ and $i \in [1,m]$, if $b \sqsubseteq b'$ then $R(b)(i) \preceq R(b')(i)$. First we formalise when we say that such a function respects suprema.

Definition 7.17 (sup-respecting function)**.** *Let L be a complete lattice. A function $R : B_L \to [1,m] \to [\lambda_L]^m_\star$ is* sup-respecting *if for all $b \in B_L$ and $X \subseteq B_L$, if $b \sqsubseteq \bigsqcup X$ then $R(b)(i) \preceq \sup\{R(b')(i) \mid b' \in X\}$.*

Observe that R is defined only on the basis elements, which are possibly (and typically) not closed under suprema. The requirement of being sup-respecting ensures that R extends to a function on L which preserves suprema. Also note that a sup-respecting function R is always monotone.

Lemma 7.13 ($\Phi_E(R)$ is monotone)**.** *Let L be a complete lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. For every function and $R : B_L \to [1,m] \to [\lambda_L]^m_\star$, the function $\Phi_E(R)$ is monotone.*

Proof. Given $b \sqsubseteq b'$ we have to show that $\Phi_E(R)(b)(i) \preceq \Phi_E(R)(b')(i)$. Note that $\mathbf{E}(b,i) = \{\boldsymbol{l} \mid b \sqsubseteq f_i(\boldsymbol{l})\} \supseteq \{\boldsymbol{l} \mid b' \sqsubseteq f_i(\boldsymbol{l})\} = \mathbf{E}(b',i)$, hence in order to determine $\Phi_E(R)(b)(i)$ we take the $\min_{\preceq_i}$ over a larger set, resulting in a smaller vector of ordinals than for $\Phi_E(R)(b')(i)$. □

In order to show that Φ_E preserves sup-respecting functions R we first need a technical lemma that will also prove useful in the next subsection.

Lemma 7.14. *Let L be a continuous lattice and let $(U_k)_{k\in K}$ with $U_k \subseteq L^m$ be a collection of upper sets. Assume that $R : B_L \to [1,m] \to [\lambda_L]^m_\star$ is sup-respecting. Then it holds that:*

$$\begin{aligned} & \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \ \wedge \ b' \ll l_j\} \mid \boldsymbol{l} \in \bigcap_{k\in K} U_k\} \\ = \ & \sup_{k\in K} \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \ \wedge \ b' \ll l_j\} \mid \boldsymbol{l} \in U_k\} \end{aligned}$$

Proof. Since all U_k are upper sets, their intersection can be written as $\bigcap_{k\in K} U_k = \{\bigsqcup_{k\in K} \boldsymbol{l}^k \mid \forall\, k \in K.\ \boldsymbol{l}^k \in U_k\}$ (where suprema of L^m are taken pointwise). Hence the left-hand side of the equation can be rewritten to

$$\min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \ \wedge \ b' \ll \bigsqcup_{k\in K} l_j^k\} \mid \forall\, k \in K.\ \boldsymbol{l}^k \in U_k\}$$

We first show that for $j \in [1,m]$

$$\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid b' \ll \bigsqcup_{k\in K} l_j^k\} = \sup_{k\in K} \sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid b' \ll l_j^k\}$$

- ($\sqsupseteq$) This direction is obvious since $b' \ll l_j^k$ implies $b' \ll \bigsqcup_{k\in K} l_j^k$. Hence every ordinal vector of the form $R(b')(j) + \boldsymbol{\delta}_i^{\eta_i}$ which is contained in the right-hand side set is automatically a member of the left-hand side set.
- ($\sqsubseteq$) Let $b' \ll \bigsqcup_{k\in K} l_j^k$. This implies that $b' \ll \bigsqcup_{k\in K} l_j^k = \bigsqcup Y$ where $Y = \bigcup_{k\in K}(B_L \cap {\downarrow} l_j^k)$, since we are in a continuous lattice. Then there exists a finite subset $Y' \subseteq Y$ such that $b' \sqsubseteq \bigsqcup Y'$ (see [GHK+03, Remark on p. 50]).

Since R is sup-respecting we have

$$\begin{aligned} R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} &\preceq \big(\sup_{y \in Y'} R(y)(j)\big) + \boldsymbol{\delta}_i^{\eta_i} \\ &= \sup_{y \in Y'} \big(R(y)(j) + \boldsymbol{\delta}_i^{\eta_i}\big) \\ &\preceq \sup_{y \in Y} \big(R(y)(j) + \boldsymbol{\delta}_i^{\eta_i}\big) \\ &= \sup_{k \in K} \sup\{R(y)(j) + \boldsymbol{\delta}_i^{\eta_i} \mid y \in Y \cap {\downarrow} l_j^k\} \\ &= \sup_{k \in K} \sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid b' \ll l_j^k\} \end{aligned}$$

Note that the first equality is due to the fact that Y' is finite and non-empty.

Since the left-hand side of the equation is the supremum of all such $R(b')(j)$ and we have shown that the right-hand side is an upper bound, the result follows.

Now we can conclude by showing that

$$\begin{aligned} & \min_{\preceq_i}\{ \sup_{j \in [1,m]} \sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid b' \ll \bigsqcup_{k \in K} l_j^k\} \mid \forall\, k \in K.\ \boldsymbol{l}^k \in U_k\} \\ = \ & \min_{\preceq_i}\{ \sup_{j \in [1,m]} \sup_{k \in K} \sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid b' \ll l_j^k\} \mid \forall\, k \in K.\ \boldsymbol{l}^k \in U_k\} \\ = \ & \min_{\preceq_i}\{\sup_{k \in K} \sup_{j \in [1,m]} \sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid b' \ll l_j^k\} \mid \forall\, k \in K.\ \boldsymbol{l}^k \in U_k\} \\ = \ & \sup_{k \in K} \min_{\preceq_i}\{ \sup_{j \in [1,m]} \sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid b' \ll l_j\} \mid \boldsymbol{l} \in U_k\} \\ = \ & \sup_{k \in K} \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \wedge b' \ll l_j\} \mid \boldsymbol{l} \in U_k\} \end{aligned}$$

where the second-last equality is due to complete distributivity. □

Now we can show that, as anticipated, Φ_E preserves sup-respecting functions.

Lemma 7.15 (Φ_E preserves sup-respecting functions)**.** *Let L be a continuous lattice and let E be a system of equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. If $R : B_L \to [1,m] \to [\lambda_L]^m_\star$ is sup-respecting, then $\Phi_E(R)$ is sup-respecting as well.*

Proof. We assume that R is sup-respecting and $R' = \Phi_E(R)$ is as follows:

$$R'(b)(i) = \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid (b', j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in \mathbf{E}(b, i)\}$$

The aim is to show that R' is sup-respecting as well. Let $X \subseteq B_L$ be a set of basis elements such that $b \sqsubseteq \bigsqcup X$. Note furthermore that $\mathbf{E}(b,i)$ is upwards-closed. We first show that

$$\bigcap_{b' \in X} \mathbf{E}(b', i) \subseteq \mathbf{E}(b, i)$$

Let $\boldsymbol{l} \in \mathbf{E}(b', i)$ for all $b' \in X$, which means that $b' \sqsubseteq f_i(\boldsymbol{l})$. So if we take the supremum over all $b' \in X$ we obtain $b \sqsubseteq \bigsqcup X \sqsubseteq f_i(\boldsymbol{l})$. Hence $\boldsymbol{l} \in \mathbf{E}(b, i)$, as required.

Now we can apply Lemma 7.14 where $K = X$, $U_{b'} = \mathbf{E}(b', i)$ and we obtain:

$$\begin{aligned} R'(b)(i) &= \min_{\preceq_i}\{\sup\{R(b'')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid (b'', j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in \mathbf{E}(b, i)\} \\ &\preceq \min_{\preceq_i}\{\sup\{R(b'')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid (b'', j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in \bigcap_{b' \in X} \mathbf{E}(b', i)\} \\ &= \sup_{b' \in X} \min_{\preceq_i}\{\sup\{R(b'')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid (b'', j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in \mathbf{E}(b', i)\} \\ &= \sup_{b' \in X} R'(b')(i) \end{aligned}$$

□

7.4.1 Computing progress measures

Selections

In principle, at least on finite lattices, the previous results allow one to compute the progress measure and thus to prove properties of the solutions of systems of equations. However, the computation can be quite inefficient due to the fact that the existential player has a (uselessly) large number of possible moves. In fact, given a system $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ on a lattice L, from a position (b, i), given any move $\boldsymbol{l} \in \mathbf{E}(b, i)$ for player $\exists$, i.e., any tuple such that $b \sqsubseteq f_i(\boldsymbol{l})$, it is immediate to see that all $\boldsymbol{l}'$ such that $\boldsymbol{l} \sqsubseteq \boldsymbol{l}'$ are valid moves for $\exists$, since by monotonicity of f_i we have $b \sqsubseteq f_i(\boldsymbol{l}) \sqsubseteq f_i(\boldsymbol{l}')$. In other words, $\mathbf{E}(b, i)$ is an upper set. However, player $\exists$, in order to win, has to try to descend as much as possible, hence playing large elements is inconvenient.

We next introduce some machinery that formalises the above intuition and allows us to make the calculation more efficient. The idea is discussed for a single function first, and then for a system of equations. For this we need to extend a bit our notation. Given a monotone function $f : L^m \to L$ and $b \in B_L$, we write $\mathbf{E}(b, f) = \{\boldsymbol{l} \in L^m \mid b \sqsubseteq f(\boldsymbol{l})\}$.

Definition 7.18 (selection)**.** *Let L be a complete lattice. Given a monotone function $f : L^m \to L$, a* selection for f *is a function $\sigma : B_L \to \mathbf{2}^{L^m}$ such that for all $b \in B_L$ it holds $\mathbf{E}(b, f) = {\uparrow}\sigma(b)$. Given a system E of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, a* selection for E *is an m-tuple of functions $\boldsymbol{\sigma}$ such that, for each $i \in [1, m]$, the function σ_i is a selection for f_i.*

Intuitively, a selection provides for each element of the basis and function f_i, a subset of the moves $\mathbf{E}(b, i)$ that are sufficient to "cover" b in all possible ways. Indeed, we can show that when computing the complete progress measure R_M according to the equations in Lemma 7.12, we can restrict the moves of the existential player to a selection. Dually, since the moves of the universal player $\mathbf{A}(\boldsymbol{l})$ are a lower set and the progress measures of interest are monotone (see Lemma 7.13), we can restrict also such moves to a subset whose lower closure is $\mathbf{A}(\boldsymbol{l})$.

Lemma 7.16 (progress measure on a selection)**.** *Let L be a continuous lattice, let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $\boldsymbol{\sigma}$ be a selection for E. For all $\boldsymbol{l} \in L^m$ let $\mathbf{A}_r(\boldsymbol{l}) \subseteq B_L \times [1,m]$ be such that $\mathbf{A}(\boldsymbol{l}) = \{(b',i) \mid (b,i) \in \mathbf{A}_r(\boldsymbol{l}) \ \wedge \ b' \sqsubseteq b\}$. The system of equations over $[\lambda_L]^m_\star$ defined, for $b \in B_L$, $i \in [1,m]$, as:*

$$R(b)(i) = \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}_r(\boldsymbol{l})\} \mid \boldsymbol{l} \in \sigma_i(b)\}$$

has the same least solution as that in Lemma 7.12.

Proof. Let Φ'_E be the operator associated with the equations in the statement of the lemma. We prove that Φ_E and Φ'_E have the same fixpoint by showing that they coincide on every monotone function R.

Let $b \in B_L$ and $i \in [1,m]$. Let us write

$$\begin{aligned}\boldsymbol{\beta}_b = \Phi'_E(R)(b)(i) &= \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}_r(\boldsymbol{l})\} \mid \boldsymbol{l} \in \sigma_i(b)\}\\ \boldsymbol{\gamma}_b = \Phi_E(R)(b)(i) &= \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in \mathbf{E}(b,i)\}\end{aligned}$$

and we show $\boldsymbol{\beta}_b =_i \boldsymbol{\gamma}_b$. First observe that

$$\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}_r(\boldsymbol{l})\} = \sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\}$$

since $\mathbf{A}(\boldsymbol{l})$ is the lower closure of $\mathbf{A}_r(\boldsymbol{l})$ and R is monotone. Then, the fact that $\boldsymbol{\gamma}_b \preceq_i \boldsymbol{\beta}_b$ follows from the observation that, by Definition 7.18, $\sigma_i(b) \subseteq \mathbf{E}(b,i)$, i.e., the first is a minimum over a smaller set. The converse inequality follows from the fact that, by Definition 7.18, for each $b \in B_L$, $i \in [1,m]$ if $\boldsymbol{l} \in \mathbf{E}(b,i)$ then there exists $\boldsymbol{l}' \in \sigma_i(b)$ such that $\boldsymbol{l}' \sqsubseteq \boldsymbol{l}$, hence $\mathbf{A}(\boldsymbol{l}') \subseteq \mathbf{A}(\boldsymbol{l})$ and thus $\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}(\boldsymbol{l}')\} \preceq_i \sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\}$. □

Since the complete progress measure R_M witnesses the existence of a winning strategy for $\exists$, the above result implies that whenever $\exists$ has a winning strategy, it has one also in the game where the moves of $\exists$ are restricted to the selection. A similar property holds for $\forall$ and $\mathbf{A}_r(\boldsymbol{l})$.

Clearly, for computational purposes, we are interested in having the selections as small as possible. Given a monotone function $f : L^m \to L$, and two selections $\sigma, \sigma' : B_L \to \mathbf{2}^{L^m}$ for f, we write $\sigma \subseteq \sigma'$ if for all $b \in B_L$ it holds $\sigma(b) \subseteq \sigma'(b)$. We will use the same notation for the pointwise order on selections for systems of equations.

For instance, given a Petri net $\mathcal{N}$, consider the powerset lattice of its states $\mathbf{2}^{\mathcal{R}(\mathcal{N})}$ ordered by subset inclusion, with basis $B = \{\{M\} \mid M \in \mathcal{R}(\mathcal{N})\}$. Then, the basic mu-calculus operators that we used in the system in Figure 7.3c admit a least selection, as detailed below.

- Given $f : (\mathbf{2}^{\mathcal{R}(\mathcal{N})})^2 \to \mathbf{2}^{\mathcal{R}(\mathcal{N})}$ defined by $f(X_1, X_2) = X_1 \cup X_2$, then $\sigma : B \to \mathbf{2}^{(\mathbf{2}^{\mathcal{R}(\mathcal{N})})^2}$ is $\sigma(\{M\}) = \{(\emptyset, \{M\}), (\{M\}, \emptyset)\}$

- Given $f : (\mathbf{2}^{\mathcal{R}(\mathcal{N})})^2 \to \mathbf{2}^{\mathcal{R}(\mathcal{N})}$ defined by $f(X_1, X_2) = X_1 \cap X_2$, then $\sigma : B \to \mathbf{2}^{(\mathbf{2}^{\mathcal{R}(\mathcal{N})})^2}$ is $\sigma(\{M\}) = \{(\{M\}, \{M\})\}$
- Given $f : \mathbf{2}^{\mathcal{R}(\mathcal{N})} \to \mathbf{2}^{\mathcal{R}(\mathcal{N})}$ defined by $f(X) = \Diamond X$, then $\sigma : B \to \mathbf{2}^{\mathbf{2}^{\mathcal{R}(\mathcal{N})}}$ is $\sigma(\{M\}) = \{\{M'\} \mid M[t\rangle M'\}$
- Given $f : \mathbf{2}^{\mathcal{R}(\mathcal{N})} \to \mathbf{2}^{\mathcal{R}(\mathcal{N})}$ defined by $f(X) = \Box X$, then $\sigma : B \to \mathbf{2}^{\mathbf{2}^{\mathcal{R}(\mathcal{N})}}$ is $\sigma(\{M\}) = \{\{M' \mid M[t\rangle M'\}\}$.

We next provide sufficient conditions for a function to admit a least selection.

Lemma 7.17 (existence of least selections)**.** *Let L be a complete lattice and let $f : L^m \to L$ be a monotone function. If f preserves the infimum of descending chains, then it admits a least selection σ_m that maps each $b \in B_L$ to the set of minimal elements of $\mathbf{E}(b, f)$.*

Proof. Assume that f preserves the infimum of descending chains. First observe that given a descending chain $(\boldsymbol{l}_\alpha)_\alpha$ in $\mathbf{E}(b, f)$ we have that $\bigsqcap_\alpha \boldsymbol{l}_\alpha \in \mathbf{E}(b, f)$. In fact, for each α we have $b \sqsubseteq f(\boldsymbol{l}_\alpha)$ and thus $b \sqsubseteq \bigsqcap_\alpha f(\boldsymbol{l}_\alpha) = f(\bigsqcap_\alpha \boldsymbol{l}_\alpha)$.

The above implies that for each $\boldsymbol{l} \in \mathbf{E}(b, f)$ there exists $\boldsymbol{l}' \in \mathbf{E}(b, f)$, minimal, such that $\boldsymbol{l}' \sqsubseteq \boldsymbol{l}$. In fact, consider the (possibly transfinite) chain of tuples $\boldsymbol{l}_\alpha$ in $\mathbf{E}(b, f)$ defined as follows. Start from $\boldsymbol{l}_0 = \boldsymbol{l}$. For any ordinal α, if there is $\boldsymbol{l}' \in \mathbf{E}(b, f)$, such that $\boldsymbol{l}' \neq \boldsymbol{l}_\alpha$ and $\boldsymbol{l}' \sqsubseteq \boldsymbol{l}_\alpha$, let $\boldsymbol{l}_{\alpha+1} = \boldsymbol{l}'$. If α is a limit ordinal $\boldsymbol{l}_\alpha = \bigsqcap_{\beta<\alpha} \boldsymbol{l}_\beta$.

This is a strictly descending chain, that thus necessarily stops at some ordinal λ bounded by the length of the longest descending chain in L. By construction $\boldsymbol{l}_\lambda \sqsubseteq \boldsymbol{l}$, $\boldsymbol{l}_\lambda \in \mathbf{E}(b, f)$ and it is minimal in $\mathbf{E}(b, f)$.

Define $\sigma_m(b)$ as the set of minimal elements of $\mathbf{E}(b, f)$ for each $b \in B_L$. It is immediate to see that this is a selection. Moreover, it is the least selection. In fact, let σ' be another selection for f. Let $\boldsymbol{l} \in \sigma_m(b)$. Since $b \sqsubseteq f(\boldsymbol{l})$ and σ' is a selection, there is $\boldsymbol{l}' \in \sigma'(b)$ such that $\boldsymbol{l}' \sqsubseteq \boldsymbol{l}$. Now, $b \sqsubseteq f(\boldsymbol{l}')$ and thus there must be $\boldsymbol{l}'' \in \sigma_m(b)$ such that $\boldsymbol{l}'' \sqsubseteq \boldsymbol{l}'$. Therefore by transitivity $\boldsymbol{l}'' \sqsubseteq \boldsymbol{l}$, but $\boldsymbol{l}$ is minimal and thus $\boldsymbol{l} = \boldsymbol{l}' = \boldsymbol{l}'' \in \sigma'(b)$. This shows $\sigma_m(b) \subseteq \sigma'(b)$ for all $b \in B_L$. Thus $\sigma_m \subseteq \sigma'$, as desired. □

For example, consider the system of equations in Figure 7.3c. Least selections for the functions f_1 and f_2 associated with the first and second equation, respectively, are given by

- $\sigma_1(\{M_1\}) = \{(\{M_1\}, \emptyset), (\{M_2\}, \emptyset)\}$ and $\sigma_1(\{M_2\}) = \{(\emptyset, \emptyset)\}$
- $\sigma_2(\{M_1\}) = \{(\{M_1\}, \{M_1, M_2\})\}$ and $\sigma_2(\{M_2\}) = \{(\{M_2\}, \{M_2\})\}$.

Observe that the winning strategy for $\exists$ that we gave for the fixpoint game depicted in Figure 7.4 is a subset of the selection above. As we already proved for progress measures, this is a general fact: if a winning strategy exists, we can find one that is a subset of any given selection.

Selections can be constructed "compositionally", i.e., if a function f arises as the composition of some component functions then we can derive a selection for f from selections for the components.

In order to define selections compositionally we first need a technical lemma that extends selections to generic elements of the lattice, possibly not part of the basis.

Lemma 7.18 (extending selections)**.** *Let L be a continuous lattice, let $f : L^m \to L$ be a monotone function and let $\sigma : B_L \to \mathbf{2}^{L^m}$ be a selection for f. Define $\bar{\sigma} : L \to \mathbf{2}^{L^m}$ by $\bar{\sigma}(b) = \sigma(b)$ for $b \in B_L$ and $\bar{\sigma}(l) = \{\bigsqcup_{b \ll l} \boldsymbol{l}^b \mid \forall\, b \in B_L \cap {\downarrow} l\,.\ \boldsymbol{l}^b \in \sigma(b)\}$ for $l \in L \smallsetminus B_L$. Then for every $l \in L$*

1. *for all $\boldsymbol{l} \in \bar{\sigma}(l)$ it holds $l \sqsubseteq f(\boldsymbol{l})$;*
2. *for all $\boldsymbol{l}' \in L^m$, if $l \sqsubseteq f(\boldsymbol{l}')$ then there exists $\boldsymbol{l} \in \bar{\sigma}(l)$ such that $\boldsymbol{l} \sqsubseteq \boldsymbol{l}'$.*

Proof. For $l \in B_L$, there is nothing to prove since the properties hold by definition of selection.

Let $l \in L \smallsetminus B_L$. We start with point (1). Let $\boldsymbol{l} \in \bar{\sigma}(l)$, hence $\boldsymbol{l} = \bigsqcup_{b \ll l} \boldsymbol{l}^b$ with $\boldsymbol{l}^b \in \sigma(b)$ for each $b \ll l$. By the properties of selections, for all $b \ll l$, since $\boldsymbol{l}^b \in \sigma(b)$ it holds $b \sqsubseteq f(\boldsymbol{l}^b)$, hence

$$b \sqsubseteq \bigsqcup_{b' \ll l} f(\boldsymbol{l}^{b'}) \sqsubseteq f(\bigsqcup_{b' \ll l} \boldsymbol{l}^{b'}) = f(\boldsymbol{l})$$

the last inequality following by monotonicity of f. Therefore $l = \bigsqcup_{b \ll l} b \sqsubseteq f(\boldsymbol{l})$, as desired.

Concerning point (2), let $\boldsymbol{l}' \in L^m$ be such that $l \sqsubseteq f(\boldsymbol{l}')$. For all $b \ll l$, since $b \sqsubseteq f(\boldsymbol{l}')$ there is $\boldsymbol{l}^b \in \sigma(b)$ such that $\boldsymbol{l}^b \sqsubseteq \boldsymbol{l}'$. Then we can consider $\boldsymbol{l} = \bigsqcup_{b \ll l} \boldsymbol{l}^b \sqsubseteq \boldsymbol{l}'$ which is in $\bar{\sigma}(l)$ by definition. □

We can now define the selection for a composition of functions.

Lemma 7.19 (selection for composition)**.** *Let L be a continuous lattice, let $g : L^n \to L$ and $f_i : L^m \to L$, for $i \in [1, n]$, be monotone functions and let $\sigma : B_L \to \mathbf{2}^{L^n}$ and $\sigma_i : B_L \to \mathbf{2}^{L^m}$, for $i \in [1, n]$, be selections for $g, f_1, \ldots, f_n$. Consider the function $h : L^m \to L$ obtained as the composition $h(\boldsymbol{l}) = g(f_1(\boldsymbol{l}), \ldots, f_n(\boldsymbol{l}))$. Then the function $\sigma' : B_L \to \mathbf{2}^{L^m}$ defined by*

$$\sigma'(b) = \{\textstyle\bigsqcup_{i=1}^n \boldsymbol{l}^i \mid \exists\, \boldsymbol{l} \in \sigma(b).\ \forall\, i \in [1, n].\ \boldsymbol{l}^i \in \bar{\sigma}_i(l_i)\}$$

is a selection for h.

Proof. We need to show that for all $b \in B_L$ it holds $\uparrow\sigma'(b) = \mathbf{E}(b, h) = \{\boldsymbol{l} \in L^m \mid b \sqsubseteq h(\boldsymbol{l})\}$. We prove the two inclusions separately. Since $\mathbf{E}(b, h)$ is an upper set, in order to prove the inclusion $\uparrow\sigma'(b) \subseteq \mathbf{E}(b, h)$ it is enough to show that for all $\boldsymbol{l}' \in \sigma'(b)$ it holds $b \sqsubseteq h(\boldsymbol{l}')$. By definition above $\boldsymbol{l}' = \bigsqcup_{i=1}^n \boldsymbol{l}^i$ such that, for some $\boldsymbol{l} \in \sigma(b)$, for all $i \in [1, n]$ we have $\boldsymbol{l}^i \in \bar{\sigma}_i(l_i)$. Since $\boldsymbol{l}^i \in \bar{\sigma}(l_i)$, by Lemma 7.18 and monotonicity of f_i we have $l_i \sqsubseteq f_i(\boldsymbol{l}^i) \sqsubseteq f_i(\bigsqcup_{i=1}^n \boldsymbol{l}^i) = f_i(\boldsymbol{l}')$. Hence $\boldsymbol{l} \sqsubseteq (f_1(\boldsymbol{l}'), \ldots, f_n(\boldsymbol{l}'))$ and thus, by monotonicity of g,

$$g(\boldsymbol{l}) \sqsubseteq g(f_1(\boldsymbol{l}'), \ldots, f_n(\boldsymbol{l}')) = h(\boldsymbol{l}')$$

Recalling that $\boldsymbol{l} \in \sigma(b)$ and thus $b \sqsubseteq g(\boldsymbol{l})$ we conclude, by transitivity, $b \sqsubseteq h(\boldsymbol{l}')$, as desired.

To prove the other inclusion, i.e., $\uparrow\sigma'(b) \supseteq \mathbf{E}(b, h)$, we need to show that for all $\boldsymbol{l} \in L^m$ such that $b \sqsubseteq h(\boldsymbol{l}) = g(f_1(\boldsymbol{l}), \ldots, f_n(\boldsymbol{l}))$ there exists $\boldsymbol{l}' \in \sigma'(b)$ such that $\boldsymbol{l}' \sqsubseteq \boldsymbol{l}$. Since σ is a selection for g, there exists $\boldsymbol{l}'' \in \sigma(b)$ such that $\boldsymbol{l}'' \sqsubseteq (f_1(\boldsymbol{l}), \ldots, f_n(\boldsymbol{l}))$. Now, for all $i \in [1, n]$, since $l_i'' \sqsubseteq f_i(\boldsymbol{l})$, by Lemma 7.18, there is $\boldsymbol{l}^i \in \bar{\sigma}_i(l_i'')$ such that $\boldsymbol{l}^i \sqsubseteq \boldsymbol{l}$. If we let $\boldsymbol{l}' = \bigsqcup_{i=1}^n \boldsymbol{l}^i$, by definition above $\boldsymbol{l}' \in \sigma'(b)$ and clearly $\boldsymbol{l}' \sqsubseteq \boldsymbol{l}$, as desired. □

Consider again the selection discussed above for the system of equations in Figure 7.3c. It can be computed composing the least selections for the basic mu-calculus operators via Lemma 7.19.

It should be noted, however, that even though in this case the selections σ_1 and σ_2 arising from the construction in Lemma 7.19 are the least ones, this is not a general fact. For this reason we favour, instead, the approach presented in the next subsection. As a counterexample take $L = \{\bot, a, b, \top\}$ with a, b incomparable, $B_L = \{a, b\}$ and $f : L \to L$ defined by $f(a) = f(b) = f(\top) = \top$ and $f(\bot) = \bot$. Then $\sigma(a) = \sigma(b) = \sigma(\top) = \{a, b\}$ and $\sigma(\bot) = \{\bot\}$ is the least selection for f. However, if we compute $\bar{\sigma}$ according to Lemma 7.18 we obtain $\bar{\sigma}(\top) = \{a, b, \top\}$ which is not minimal. As a consequence, the selection σ', constructed as in Lemma 7.19 by taking f defined as above and g the identity, is not a least selection for the composition.

A logic for characterising the moves of the existential player

The sets of possible moves of the existential player are upper sets in the lattice. Such sets can be conveniently represented and manipulated in logical form (see, e.g., [DR00]). Intuitively, (least) selections describe a disjunctive normal form, but more compact representations can be obtained using arbitrary nesting of conjunctions and disjunctions. For instance, the least selection for the monotone function $f(X_1, \ldots, X_{2n}) = (X_1 \cup X_2) \cap (X_3 \cup X_4) \cap \cdots \cap (X_{2n-1} \cup X_{2n})$ would be of exponential size (think of the corresponding disjunctive normal form), but we can easily give a formula of linear size.

This motivates the introduction of a propositional logic for expressing the sets of moves of the existential player along with a technique for deriving the fixpoint equations for computing the progress measure, avoiding the potential exponential explosion.

Definition 7.19 (logic for upper sets)**.** *Let L be a continuous lattice. Given $m \in \mathbb{N}$, the logic $\mathcal{L}_m(B_L)$ has formulae defined as follows, where $b \in B_L$ and $j \in [1, m]$:*

$$\varphi \ ::= \ [b, j] \ \mid \ \bigvee_{k \in K} \varphi_k \ \mid \ \bigwedge_{k \in K} \varphi_k$$

We will write true *for the empty conjunction. The semantics of a formula φ is an upper*

set $[\![\varphi]\!] \subseteq L^m$*, defined as follows:*

$$\begin{aligned}
[\![[b,j]]\!] &= \{\boldsymbol{l} \in L^m \mid b \sqsubseteq l_j\} \\
[\![\bigvee_{k\in K} \varphi_k]\!] &= \bigcup_{k\in K} [\![\varphi_k]\!] \\
[\![\bigwedge_{k\in K} \varphi_k]\!] &= \bigcap_{k\in K} [\![\varphi_k]\!]
\end{aligned}$$

It is easy to see that indeed each upper set is denoted by a formula, showing that the logic is sufficiently expressive.

Lemma 7.20 (formulae for upper sets)**.** *Let L be a continuous lattice with basis B_L and let $X \subseteq L^m$ be an upper set. Then $X = [\![\varphi]\!]$ where φ is the formula in $\mathcal{L}_m(B_L)$ defined as follows:*

$$\varphi = \bigvee_{\boldsymbol{l}\in X} \bigwedge \big\{[b,j] \mid j \in [1,m] \ \wedge \ b \sqsubseteq l_j\big\}.$$

Proof. We have to show that $[\![\varphi]\!] = X$:

- ($\subseteq$) Let $\boldsymbol{l}' \in [\![\varphi]\!]$, hence

  $$\boldsymbol{l}' \in \bigcup_{\boldsymbol{l}\in X} \bigcap \{\{\boldsymbol{l}'' \in L^m \mid b \sqsubseteq l''_j\} \mid j \in [1,m] \wedge b \sqsubseteq l_j\}.$$

  Hence there exists $\boldsymbol{l} \in X$ such that for all $j \in [1,m]$ and $b \sqsubseteq l_j$ it holds that $b \sqsubseteq l'_j$. Then

  $$l_j = \bigsqcup \{b \mid b \sqsubseteq l_j\} \sqsubseteq \bigsqcup \{b \mid b \sqsubseteq l'_j\} = l'_j.$$

  Hence $\boldsymbol{l} \sqsubseteq \boldsymbol{l}'$ and, since X is an upper set, $\boldsymbol{l}' \in X$.

- ($\supseteq$) Let $\boldsymbol{l} \in X$. We show that $\boldsymbol{l} \in [\![\psi_{\boldsymbol{l}}]\!]$ where $\psi_{\boldsymbol{l}} = \bigwedge \big\{[b,j] \mid j \in [1,m] \ \wedge \ b \sqsubseteq l_j\big\}$. In fact

  $$[\![\psi_{\boldsymbol{l}}]\!] = \bigcap \{\{\boldsymbol{l}' \in L^m \mid b \sqsubseteq l'_j\} \mid j \in [1,m] \wedge b \sqsubseteq l_j\}.$$

  Now, if $j \in [1,m]$ and $b \sqsubseteq l_j$ then clearly $\boldsymbol{l} \in \{\boldsymbol{l}' \mid b \sqsubseteq l'_j\}$ and hence $\boldsymbol{l}$ is contained in the intersection.

□

For practical purposes we should restrict to finite formulae. This can surely be done in the case of finite lattices, but also for well-preorders (see, e.g., [DR00]).

Definition 7.20 (symbolic $\exists$-moves)**.** *Let L be a continuous lattice and let $f : L^m \to L$ be a monotone function. A* symbolic $\exists$-move *for f is a family $(\varphi_b)_{b\in B_L}$ of formulae in $\mathcal{L}_m(B_L)$ such that $[\![\varphi_b]\!] = \mathbf{E}(b, f)$ for all $b \in B_L$. If E is a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, a* symbolic $\exists$-move *for E is a family of formulae $(\varphi^i_b)_{b\in B_L, i\in[1,m]}$ such that for all $i \in [1,m]$, the family $(\varphi^i_b)_{b\in B_L}$ is a symbolic $\exists$-move for f_i.*

Interestingly, symbolic ∃-moves can be obtained compositionally, namely, the formulae corresponding to a functions arising as a composition can be obtained from those of the components.

Lemma 7.21 (symbolic ∃-moves, compositionally)**.** *Let L be a continuous lattice, let $g : L^n \to L$ and $f_j : L^m \to L$, for $j \in [1,n]$ be monotone functions and let $(\varphi_b)_{b\in B_L}$ and $(\varphi^j_b)_{b\in B_L}$, for $j \in [1,n]$, be symbolic ∃-moves for $g, f_1, \dots, f_n$. Consider the function $h : L^m \to L$ obtained as the composition $h(\boldsymbol{l}) = g(f_1(\boldsymbol{l}), \dots, f_n(\boldsymbol{l}))$. Define $(\varphi'_b)_{b\in B_L}$ as follows. For all $b \in B_L$, the formula φ'_b is obtained from φ_b by replacing each occurrence of $[b', j]$ by $\varphi^j_{b'}$. Then $(\varphi'_b)_{b\in B_L}$ is a symbolic ∃-move for h.*

Proof. We first show that given a formula $\varphi \in \mathcal{L}_n(B_L)$, if φ' is the formula in $\mathcal{L}_m(B_L)$ obtained from φ by replacing each occurrence of an atom $[b, j]$ by φ^j_b, then

$$[\![\varphi']\!] = \{\boldsymbol{l} \in L^m \mid (f_1(\boldsymbol{l}), \dots, f_n(\boldsymbol{l})) \in [\![\varphi]\!]\}$$

We proceed by induction on the shape of φ_b.

- ($\varphi = [b, j]$): in this case $\varphi' = \varphi^j_b$. Therefore we have

$$\begin{aligned}
[\![\varphi']\!] &= [\![\varphi^j_b]\!] \\
&= \{\boldsymbol{l} \in L^m \mid b \sqsubseteq f_j(\boldsymbol{l})\} \\
&= \{\boldsymbol{l} \in L^m \mid (f_1(\boldsymbol{l}), \dots, f_n(\boldsymbol{l})) \in [\![[b, j]]\!]\} \\
&= \{\boldsymbol{l} \in L^m \mid (f_1(\boldsymbol{l}), \dots, f_n(\boldsymbol{l})) \in [\![\varphi]\!]\}
\end{aligned}$$

- ($\varphi = \bigvee_{k\in K} \varphi_k$): in this case $\varphi' = \bigvee_{k\in K} \varphi'_k$, where each φ'_k is obtained from φ_k by by replacing each occurrence of an atom $[b, j]$ by φ^j_b. Then

$$\begin{aligned}
[\![\varphi']\!] &= [\![\bigvee_{k\in K} \varphi'_k]\!] \\
&= \bigcup_{k\in K} [\![\varphi'_k]\!] \\
&= \bigcup_{k\in K} \{\boldsymbol{l} \in L^m \mid (f_1(\boldsymbol{l}), \dots, f_n(\boldsymbol{l})) \in [\![\varphi_k]\!]\} && \text{[by inductive hyp.]} \\
&= \{\boldsymbol{l} \in L^m \mid (f_1(\boldsymbol{l}), \dots, f_n(\boldsymbol{l})) \in \bigcup_{k\in K} [\![\varphi_k]\!]\} \\
&= \{\boldsymbol{l} \in L^m \mid (f_1(\boldsymbol{l}), \dots, f_n(\boldsymbol{l})) \in [\![\bigvee_{k\in K} \varphi_k]\!]\} \\
&= \{\boldsymbol{l} \in L^m \mid (f_1(\boldsymbol{l}), \dots, f_n(\boldsymbol{l})) \in [\![\varphi]\!]\}
\end{aligned}$$

  as desired.

- ($\varphi = \bigwedge_{k\in K} \varphi_k$): analogous.

Now, given $b \in B_L$, we have to show that

$$[\![\varphi'_b]\!] = \mathbf{E}(b,h) = \{\boldsymbol{l} \mid \boldsymbol{l} \in L^m \ \wedge \ b \sqsubseteq h(\boldsymbol{l})\} = \{\boldsymbol{l} \mid \boldsymbol{l} \in L^m \ \wedge \ b \sqsubseteq g(f_1(\boldsymbol{l}), \ldots, f_n(\boldsymbol{l}))\}.$$

This is almost immediate. In fact

$$\begin{aligned} [\![\varphi'_b]\!] &= \{\boldsymbol{l} \in L^m \mid (f_1(\boldsymbol{l}), \ldots, f_n(\boldsymbol{l})) \in [\![\varphi_b]\!]\} && \text{[by the property proved above]} \\ &= \{\boldsymbol{l} \in L^m \mid b \sqsubseteq g(f_1(\boldsymbol{l}), \ldots, f_n(\boldsymbol{l}))\} && \text{[by def. of symbolic } \exists\text{-move]} \end{aligned}$$

□

A symbolic $\exists$-move for a system can be directly converted into a recipe for evaluating the fixpoint expressions for progress measures. Essentially, every disjunction simply has to be replaced by a minimum and every conjunction by a supremum (although the proof, which relies on complete distributivity of the lattice $[\lambda_L]^m_\star$ is not trivial). Furthermore, in the case of an algebraic lattice, where we can ensure that the elements of the basis are compact, an atom translates to a straightforward lookup of the progress measure without additional computation.

Proposition 7.3 (progress measure from symbolic $\exists$-moves)**.** *Let L be a continuous lattice, let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $(\varphi^i_b)_{b \in B_L, i \in [1,m]}$ be a symbolic $\exists$-move for E. Then the system of equations over $[\lambda_L]^m_\star$ for computing the progress measure can be written, for all $b \in B_L$ and $i \in [1,m]$, as $R(b)(i) = R^i_{\varphi^i_b}$ where R^i_ψ is defined inductively as follows:*

$$\begin{aligned} R^i_{[b,j]} &= \min\nolimits_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid b' \ll b\}\} \\ R^i_{\bigvee_{k \in K} \varphi_k} &= \min_{k \in K} R^i_{\varphi_k} \\ R^i_{\bigwedge_{k \in K} \varphi_k} &= \sup_{k \in K} R^i_{\varphi_k} \end{aligned}$$

Whenever the basis element b is compact it holds that $R^i_{[b,j]} = \min_{\preceq_i}\{R(b)(j) + \boldsymbol{\delta}^{\eta_i}_i\}$. (Note that the operator $\min_{\preceq_i}$ in the definition of $R^i_{[b,j]}$ above is just there to ensure that all entries in positions smaller than i are set to 0.)

Proof. Since $(\varphi^i_b)_{b \in B_L, i \in [1,m]}$ is a symbolic $\exists$-move for E, the equations of Definition 7.16 can be written as

$$R(b)(i) = \min\nolimits_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in [\![\varphi^i_b]\!]\}.$$

We conclude by proving that, when R is monotone

$$R^i_\psi = \min\nolimits_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in [\![\psi]\!]\}.$$

We proceed by induction on the shape of ψ.

- $(\psi = [b,k])$: by definition

$$\begin{aligned}&\min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in [\![[b,k]]\!]\} = \\ &\min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \ \wedge \ b' \ll l_j\} \mid \boldsymbol{l} \in L^m \ \wedge \ b \sqsubseteq l_k\}\end{aligned}$$

A vector $\boldsymbol{l} \in L^m$ satisfying $b \ll l_k$ has the form $(l_1, \ldots, l_m)$ where l_j is arbitrary if $j \neq k$ and $b \ll l_k$. Since we can assume that R is monotone and hence the inner supremum is monotone in $\boldsymbol{l}$, we can conclude that the minimum is reached for a vector $\boldsymbol{\ell}$ where $l_j = \bot$ if $j \neq k$ and $b \sqsubseteq l_k$. Hence we obtain

$$\min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \ \wedge \ b' \ll l_j\} \mid \boldsymbol{l} \in L^m, b \sqsubseteq l_k, l_j = \bot \text{ if } j \neq k\}.$$

Since there is no basis element b' with $b' \ll \bot$, it is sufficient if one takes the inner suprema only for elements with $j = k$ and $b' \sqsubseteq l_k$. And so we obtain

$$\min_{\preceq_i}\{\sup\{R(b')(k) + \boldsymbol{\delta}_i^{\eta_i} \mid b' \ll l\} \mid l \in L \ \wedge \ b \sqsubseteq l\}$$

We can now infer that b is the least value $l \in L$ such that $b \sqsubseteq l$ and hence – again by monotonicity – the above can be rewritten as

$$\min_{\preceq_i}\{\sup\{R(b')(k) + \boldsymbol{\delta}_i^{\eta_i} \mid b' \ll b\}\}$$

which is exactly $R^i_{[b,k]}$, as desired.

If b is compact, we know that b itself is the least element of all l such that $b \ll l$ and we can write the above as

$$R^i_{[b,k]} = \min_{\preceq_i}\{R(b)(k) + \boldsymbol{\delta}_i^{\eta_i}\}.$$

- $(\psi = \bigvee_{k\in K} \varphi_k)$:

$$\begin{aligned}R^i_{\bigvee_{k\in K}\varphi_k} &= \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \ \wedge \ b' \ll l_j\} \mid \boldsymbol{l} \in \bigcup_{k\in K} [\![\varphi_k]\!]\} \\ &= \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \ \wedge \ b' \ll l_j\} \mid \exists\, k \in K.\ \boldsymbol{l} \in [\![\varphi_k]\!]\} \\ &= \min_{k\in K} \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \ \wedge \ b' \ll l_j\} \mid \boldsymbol{l} \in [\![\varphi_k]\!]\} \\ &= \min_{k\in K} R^i_{\varphi_k}\end{aligned}$$

- $(\psi = \bigwedge_{k\in K} \varphi_k)$: since every set $[\![\varphi_k]\!]$ is an upper set we can immediately apply Lemma 7.14 and obtain

$$\begin{aligned}R^i_{\bigwedge_{k\in K}\varphi_k} &= \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \ \wedge \ b' \ll l_j\} \mid \boldsymbol{l} \in \bigcap_{k\in K} [\![\varphi_k]\!]\} \\ &= \sup_{k\in K} \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid j \in [1,m] \ \wedge \ b' \ll l_j\} \mid \boldsymbol{l} \in [\![\varphi_k]\!]\} \\ &= \sup_{k\in K} R^i_{\varphi_k}\end{aligned}$$

□

Consider again our running example in Figure 7.3. The selections specified in the previous subsection can be expressed in the logic as follows:

$$\varphi^1_{\{M_1\}} = [\{M_1\},1] \vee [\{M_2\},1] \qquad \varphi^1_{\{M_2\}} = \mathsf{true}$$
$$\varphi^2_{\{M_1\}} = [\{M_1\},1] \wedge [\{M_1\},2] \wedge [\{M_2\},2] \qquad \varphi^2_{\{M_2\}} = [\{M_2\},1] \wedge [\{M_2\},2].$$

These formulae can be obtained compositionally. For instance the formula $\varphi^2_{\{M_1\}}$ for the equation $x_2 =_\nu \Box x_2 \cap x_1$ is obtained by combining a logical formula for x_1 (namely $[\{M_1\},1]$) via conjunction with a logical formula for $\Box x_2$ (namely $[\{M_1\},2] \wedge [\{M_2\},2]$).

Using the logical formulae above, we obtain the following equations for the progress measure (where $\max_{\preceq_i}$ works analogously to $\min_{\preceq_i}$: it sets all vector entries in positions smaller than i to 0):

$$\begin{aligned}
R(\{M_1\})(1) &= \min_{\preceq_1}\{R(\{M_1\})(1)+(1,0), R(\{M_2\})(1)+(1,0)\}\\
R(\{M_1\})(2) &= \max_{\preceq_2}\{R(\{M_1\})(1), R(\{M_1\})(2), R(\{M_2\})(2)\}\\
R(\{M_2\})(1) &= (0,0)\\
R(\{M_2\})(2) &= \max_{\preceq_2}\{R(\{M_2\})(1), R(\{M_2\})(2)\}.
\end{aligned}$$

Observe that this coincides with the solution for the corresponding progress measure equations given in Section 7.4.

Complexity analysis

The benefit of the progress measures introduced in [Jur00] is to ensure that model-checking is polynomial in the number of states and exponential in (half of) the alternation depth. We will now perform a corresponding complexity analysis for our setting, based on symbolic $\exists$-moves and by assuming that we are working on a finite lattice.

Let E be a fixed system of m equations over a finite complete lattice L, let k be the number of μ-equations and let B_L be a basis for L. Let $(\varphi^i_b)_{b\in B_L, i\in[1,m]}$ be a symbolic $\exists$-move for E and assume that the size of every such formula is bounded by s. Note that the formulae are typically of moderate size. For instance, mu-calculus model-checking, the branching of a transition system (i.e., the number of successors of a single state) is a determining factor. In fact, as it can be grasped from our running example (see Figure 7.3), the size of the symbolic $\exists$-move φ^i_b will be linear in the number of propositional operators and, in the presence of modal operators, linear in the branching degree of the transition system. For arbitrary monotone functions it is more difficult to give a general rule.

The shape of the formulae in the symbolic $\exists$-move determines how the values of the progress measure at various positions (b,i) of the games are interrelated. These dependencies clearly play a role in the computation and thus are made explicit by following definition.

Definition 7.21 (dependency graph)**.** *Given two game positions $(b,i),(b',j) \in B_L \times [1,m]$ of $\exists$ we say that (b,i) is a* predecessor *of (b',j) if $[b',j]$ occurs in φ^i_b. We will write $pred(b',j)$ for the set of predecessors of (b',j). In this situation we will also call the pair $((b,i),(b',j))$ an* edge *and refer to corresponding graph as the* dependency graph *for E.*

As a first step we provide a bound to the number of edges in the dependency graph.

Proposition 7.4 (edges in the dependency graph)**.** *The number e of edges in the dependency graph for system E is such that $e \leq \min\{|B_L| \cdot m \cdot s, (|B_L| \cdot m)^2\}$, where m is the number of equations and s is the bound on the size of symbolic $\exists$-moves.*

Proof. There are at most $|B_L| \cdot m$ game positions and hence the number of edges is obviously bounded by $(|B_L| \cdot m)^2$. Moreover, for each game position, the number of outgoing edges is bounded by the size of the formula (symbolic $\exists$-move) associate to the position. Hence the thesis. □

In order to bound the complexity of the overall computation of the progress measure, first note that the lattice $[\lambda_L]^m_\star$ contains $(\lambda_L+1)^m+1$ elements. However only $h = (\lambda_L+1)^k+1$ are relevant, since the entries of ν-indices are always set to 0. As an example, when model-checking a mu-calculus formula over a finite state system, λ_L is the size of the state space of the corresponding model. In fact, if the lattice is $(\mathbf{2}^{\mathbb{S}}, \subseteq)$ where $\mathbb{S} = \{s_0, \ldots, s_n\}$ is the state space, then (one of) the longest ascending chain is $\emptyset \subseteq \{s_0\} \subseteq \{s_0, s_1\} \subseteq \ldots \subseteq \mathbb{S}$.

This fact and the observation that we can perform the fixpoint iteration for the progress measure using a worklist algorithm (like those used in data flow analysis, see, e.g., [NNH99]) on the dependency graph, lead to the following result.

Theorem 7.2 (computing progress measures)**.** *The time complexity for computing the least fixpoint progress measure for a system E over a finite complete lattice L is $O(s \cdot k \cdot e \cdot h)$, where s is the bound on the size of symbolic $\exists$-moves, k is the number of μ-equations, e the number of edges in the dependecy graph, and $h = (\lambda_L + 1)^k + 1$.*

Proof. We use a worklist algorithm, and the worklist initially contains all edges.

Processing an edge $((b,i),(b',j))$ means to update the value $R(b)(i)$ by evaluating the formula φ^i_b. Afterwards all edges originating from (b,i) can be removed from the worklist. Whenever a value $R(b')(j)$ increases, all edges $((b,i),(b',j))$ with $(b,i) \in pred(b',j)$ will be again inserted into the worklist. Hence, at most $\sum_{(b',j)\in B_L\times[1,m]} h \cdot pred(b',j) = e \cdot h$ edges will be inserted into the worklist and processed later.

In turn, processing an edge has complexity at most $O(s \cdot k)$, since we inductively evaluate a formula of size s on ordinal vectors of length k. (Since the lattice is finite, it is automatically algebraic and the simpler case for compact elements of Theorem 7.3 applies.) Everything combined, we obtain a runtime of $O(s \cdot k \cdot e \cdot h)$. □

We compare the above with the runtime in [Jur00], which is $O(dg\left(\frac{n}{d}\right)^{\lceil \frac{d}{2} \rceil})$, where d is the alternation depth of the formula, g the number of edges and n the number of nodes of the parity game.

The correspondence is as follows: g corresponds to our number e and n to λ_L (where we cannot exploit the optimisation by Jurdziński which uses the fact that every node in the parity game is associated with a single parity, leading to the division by d). Furthermore s is a new factor, which is due to the fact that we are working with arbitrary functions. But this is mitigated by the fact that we often obtain smaller parity games than in the standard mu-calculus case (see for instance Figure 7.4). The number $\frac{d}{2}$ corresponds to our k. However $\frac{d}{2}$ could potentially be strictly lower than k, since we did not take into account the fact that some equations might not be dependent on other equations.

To incorporate this and possibly further optimisations into the complexity analysis we need a notion of alternation depth for equation systems. This can be easily obtained by extending the one introduced in [CKS92, Sch04]. A system of equations can be split into closed subsystems corresponding to the strongly connected components of the dependency graph for the system. Then the alternation depth of the system is defined as the length of the longest chain of mutually dependent μ- and ν-equations within a closed subsystem. By solving every component separately we obtain a more efficient algorithm.

In particular, systems of fixpoint equations that consist only of μ-equations or ν-equations can be solved by a single fixpoint iteration on L^m, where m is the number of equations [Ven08]. Similarly, equations with indices $i, i+1$ where $\eta_i = \eta_{i+1}$ can be merged. This results in an equation system where subsequent equations alternate between μ and ν. (Notice that this transformation means that each equation is over L^j instead of L, but this can be easily adapted.)

Note also that the runtime might be substantially improved by finding a good strategy for computing the progress measure, as spelled out in [Jur00], in the same way as efficient ways can be found for implementing the worklist algorithm in program analysis.

7.5 Model checking $\mathcal{L}_{hp}$

In this section we discuss the application of the fixpoint game and the methods developed in the previous sections to the model-checking of $\mathcal{L}_{hp}$. First of all, we formally explain how formulae of the logic can be translated into systems of equations (and vice versa). Then we show how the fixpoint game instantiates on such systems and study the relation between strategies in the game and other model-checking methods.

7.5.1 Systems of fixpoint equations for logical formulae

Recall that formulae are assumed to be such that different bound propositions have different names, a requirement that can always be fulfilled by alpha-renaming. Hereafter, for every

proposition Z appearing in a closed formula φ we denote by φ_Z the fixpoint subformula quantifying Z (hence φ_Z will be of the kind $(\eta Z(\boldsymbol{x}).\psi)(\boldsymbol{y})$ for some ψ).

A (closed) formula of $\mathcal{L}_{hp}$ can be easily translated into a system of fixpoint equations. Each fixpoint subformula $(\eta Z(\boldsymbol{x}).\psi)(\boldsymbol{y})$ becomes an equation of the system $Z(\boldsymbol{x}) = \theta$, where θ is obtained from ψ by replacing each fixpoint subformula by the corresponding proposition. In order to simplify the translation we assume that the original formula is a fixpoint formula. This is not a restriction since any closed formula φ is equivalent to a fixpoint formula $\mu Z.\varphi$, where Z is a proposition not occurring in φ. Then the system will have as many equations as many different propositions appear in the initial formula. As we already observed, the order of the equations is relevant in certain cases. Luckily, there is a simple order which is always correct: the *reverse reading order* in which the fixpoint operators (and their corresponding propositions) appear from right to left in the formula. Actually, any order compatible with the order $\sqsubset_d^*$ over fixpoint subformulae would be correct. But since it is easier to just read the formula than computing such partial order, we favour the more naive and direct way. To see that the reverse reading order is compatible with $\sqsubset_d^*$, just observe that, over fixpoint subformulae, the reverse reading order is refined by the subformula relation, which in turn is refined by $\sqsubset_d^*$.

Definition 7.22 (equation system for a formula)**.** *Given a closed fixpoint formula φ of $\mathcal{L}_{hp}$, let $(X_1, \ldots, X_m)$ be the tuple of abstract propositions in φ, ordered by the reverse reading order in which their quantifications appear, and let each fixpoint subformula of φ be $\varphi_{X_i} = (\eta_i X_i(\boldsymbol{x}_i).\psi_i)(\boldsymbol{y}_i)$ for $i \in [1, m]$. The* equational form *of φ is $\{X_i(\boldsymbol{x}_i) =_{\eta_i} \theta_i\}_{i\in[1,m]}$, where each θ_i is the (open) formula obtained from ψ_i by replacing every fixpoint subformula φ_{X_j} with the corresponding proposition whose event variables are named $\boldsymbol{y}_j$.*

Once a PES model $\mathcal{E}$ is fixed, the formula in equational form can be interpreted as a system of equations over the powerset lattice $\langle \mathbf{2}^{\mathcal{C}(\mathcal{E})\times Env_{\mathcal{E}}}, \subseteq\rangle$, by replacing formulae with their semantics, i.e., for all $i \in [1, m]$, the equation $X_i(\boldsymbol{x}_i) =_{\eta_i} \theta_i$ becomes

$$v_i \quad =_{\eta_i} \quad \{\!|\theta_i|\!\}_{\pi[X_j(\boldsymbol{x}_j)\mapsto v_j]_{\forall j\in[1,m]}}$$

where v_i is a variable of the system, hence it assume values over the lattice. Note that π can be any proposition environment: it is irrelevant because the evaluation of every proposition in the formula is given by the corresponding update.

It is not difficult to see that also a converse transformation is possible. A system of fixpoint equations of the kind $\{X_i(\boldsymbol{x}_i) =_{\eta_i} \theta_i\}_{i\in[1,m]}$ where each θ_i is an open formula of $\mathcal{L}_{hp}$ with free propositions in $\boldsymbol{X}$ and without fixpoint subformulae, can be translated into a tuple of formulae of $\mathcal{L}_{hp}$, equivalent to the system in a sense formalised later.

Definition 7.23 (formulae for an equation system)**.** *Let E be a system of m equations with equational form $\{X_i(\boldsymbol{x}_i) =_{\eta_i} \psi_i\}_{i\in[1,m]}$ where each ψ_i is an open formula of $\mathcal{L}_{hp}$ with $fp(\psi_i) \subseteq \boldsymbol{X}$ and without fixpoint subformulae. The corresponding m-tuple of formulae of*

$\mathcal{L}_{hp}$, denoted by $\boldsymbol{\varphi}^E$, is defined inductively as follows, where E_i denotes the system consisting of the first i equations of E, for all $i \in [1,m]$, and $E_0 = \emptyset$.

$$\begin{array}{lcl} \boldsymbol{\varphi}^{\emptyset} & = & \epsilon \\ \boldsymbol{\varphi}^{E_i} & = & (\boldsymbol{\varphi}^{E_{i-1}}[X_i(\boldsymbol{x}_i) \coloneqq \varphi_i^{E_i}], \varphi_i^{E_i}) \ \textit{ where } \varphi_i^{E_i} = \eta_i X_i(\boldsymbol{x}_i).\psi_i[X_j(\boldsymbol{x}_j) \coloneqq \varphi_j^{E_{i-1}}]_{\forall j \in [1,i-1]} \end{array}$$

Then $\boldsymbol{\varphi}^E = \boldsymbol{\varphi}^{E_m}$.

Note that formulae in the tuple $\boldsymbol{\varphi}^E$ have no free propositions. However, they may have free event variables.

Similar procedures have been considered in [CKS92], for the characterisation of mu-calculus formulae in terms of equation systems, to allow an efficient model-checking algorithm, and also in [Mad97].

As an example, consider the formula $\mu X_2.(\langle\!\langle \_\, x \rangle\!\rangle X_2 \vee \langle\!\langle \mathsf{c}\, y \rangle\!\rangle \nu X_1(y).\langle\!\langle \overline{y} < \mathsf{b}\, z \rangle\!\rangle X_1(y))$ stating that eventually a state is reached by executing a c-labelled event from where an infinite chain of b-labelled events can be executed, all concurrent with the c-labelled one. The equational form of such formula is

$$\begin{array}{lcl} X_1(y) & =_{\nu} & \langle\!\langle \overline{y} < \mathsf{b}\, z \rangle\!\rangle X_1(y) \\ X_2 & =_{\mu} & \langle\!\langle \_\, x \rangle\!\rangle X_2 \vee \langle\!\langle \mathsf{c}\, y \rangle\!\rangle X_1(y) \end{array}$$

Then, after fixing a model $\mathcal{E}$, we can interpret the equations above as a system over the lattice $\langle \mathbf{2}^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}, \subseteq \rangle$, which can be rewritten as

$$\begin{array}{lcl} v_1 & =_{\nu} & \{\!|\langle\!\langle \overline{y} < \mathsf{b}\, z \rangle\!\rangle X_1(y)|\!\}^{\mathcal{E}}_{\pi[X_1(y) \mapsto v_1][X_2 \mapsto v_2]} \\ v_2 & =_{\mu} & \{\!|\langle\!\langle \_\, x \rangle\!\rangle X_2 \vee \langle\!\langle \mathsf{c}\, y \rangle\!\rangle X_1(y)|\!\}^{\mathcal{E}}_{\pi[X_1(y) \mapsto v_1][X_2 \mapsto v_2]} \end{array}$$

If now we apply the converse procedure we obtain the pair of formulae:

$$\begin{array}{lcl} \varphi_1 & = & \nu X_1(y).\langle\!\langle \overline{y} < \mathsf{b}\, z \rangle\!\rangle X_1(y) \\ \varphi_2 & = & \mu X_2.(\langle\!\langle \_\, x \rangle\!\rangle X_2 \vee \langle\!\langle \mathsf{c}\, y \rangle\!\rangle \nu X_1(y) \langle\!\langle \overline{y} < \mathsf{b}\, z \rangle\!\rangle X_1(y)) \end{array}$$

Note that φ_2 is the formula we started with. In this specific case φ_1 coincides with the inner fixpoint subformula quantifying the proposition X_1. This happens only because X_2 does not occur free in φ_1. In general, if proposition X_2 occurred in the fixpoint subformula quantifying X_1, each occurrence of X_2 in φ_1 would have been replaced by φ_2.

The following result shows that, indeed, the proposed translations are correct, as they preserve the semantics of formulae.

Proposition 7.5 (correspondence between formulae and equation systems)**.** *Let $\mathcal{E}$ be a PES. Given a closed fixpoint formula φ of $\mathcal{L}_{hp}$, let E be the system of m equations arising from its equational form. For any proposition environment $\pi \in PEnv_{\mathcal{E}}$, it holds $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi} = sol_m(E)$. Conversely, given a system E of m equations with the equational form $\{X_j(\boldsymbol{x}_j) =_{\eta_j} \psi_j\}_{j \in [1,m]}$ where each ψ_j has no fixpoint subformula and $fp(\psi_j) \subseteq \boldsymbol{X}$, it holds $sol_i(E) = \{\!|\varphi_i^E|\!\}^{\mathcal{E}}_{\pi}$ for all $i \in [1,m]$ and $\pi \in PEnv_{\mathcal{E}}$.*

Proof. We prove the two statements separately.

For the first part, observe that since φ is closed the proposition environment π used for its denotation is irrelevant. Recall that for every $i \in [1, m]$, the i-th equation of the system is $X_i(\boldsymbol{x}_i) =_{\eta_i} \theta_i$, where θ_i is obtained from the subformula ψ_i of the fixpoint formula $\varphi_{X_i} = (\eta_i X_i(\boldsymbol{x}_i).\psi_i)(\boldsymbol{y}_i)$, as described in Definition 7.22. In particular, $\varphi = \varphi_{X_m} = \eta_m X_m.\psi_m$, corresponds to the last equation of the equational form $X_m =_{\eta_m} \theta_m$, which has no event variables since φ is closed. Then, we actually prove that for all $i \in [1, m]$, $\{\!|\eta_i X_i(\boldsymbol{x}_i).\psi_i|\!\}^{\mathcal{E}}_{\pi'} = sol_i(E')$ where $\pi' = \pi[X_{i+1}(\boldsymbol{x}_{i+1}) \mapsto S_{i+1}] \dots [X_m(\boldsymbol{x}_m) \mapsto S_m]$ and $E' = E[v_m := S_m] \dots [v_{i+1} := S_{i+1}]$, and every $S_j \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$. Clearly this implies the desired result. The proof proceeds by induction on the index i with base case $i = 1$ because there must be at least one equation since φ is a fixpoint formula.

($i = 1$): by definition of substitution we know that the system E' consists of a single equation, i.e., $v_1 =_{\eta_1} \{\!|\theta_1|\!\}^{\mathcal{E}}\pi'$. By definition of solution of a system, we have that $sol_1(E') = \eta_1(\lambda S.\{\!|\theta_1|\!\}^{\mathcal{E}}_{\pi'[X_1(\boldsymbol{x}_1) \mapsto S]})$. Similarly, by definition of the semantics of $\mathcal{L}_{hp}$ we know that $\{\!|\eta_1 X_1(\boldsymbol{x}_1).\psi_1|\!\}^{\mathcal{E}}_{\pi'} = \eta_1(\lambda S.\{\!|\psi_1|\!\}^{\mathcal{E}}_{\pi'[X_1(\boldsymbol{x}_1) \mapsto S]})$. By the ordering of the variables given in Definition 7.22, we must have that ψ_1 does not contain any fixpoint subformula, otherwise its index could not be 1. Hence $\theta_1 = \psi_1$, and so, combining the previous facts, $\{\!|\eta_1 X_1(\boldsymbol{x}_1).\psi_1|\!\}^{\mathcal{E}}_{\pi'} = sol_1(E')$.

($i > 1$): in this case, by definition of solution, we have $sol_i(E') = \eta_i(\lambda S.\{\!|\theta_i|\!\}^{\mathcal{E}}_{\pi''})$ where $\pi'' = \pi'[X_i(\boldsymbol{x}_i) \mapsto S][X_j(\boldsymbol{x}_j) \mapsto sol_j(E'[v_i := S])]_{\forall j \in [1, i-1]}$. While, by definition of the semantics we have $\{\!|\eta_i X_i(\boldsymbol{x}_i).\psi_i|\!\}^{\mathcal{E}}_{\pi'} = \eta_i(\lambda S.\{\!|\psi_i|\!\}^{\mathcal{E}}_{\pi'[X_i(\boldsymbol{x}_i) \mapsto S]})$. By an inspection of the definition of $\boldsymbol{\theta}$ and the ordering of the variables, one can notice that for all $j \in [1, m]$

$$\psi_j = \theta_j[X_{j-1}(\boldsymbol{x}_{j-1}) \coloneqq \eta_{j-1} X_{j-1}(\boldsymbol{x}_{j-1}).\psi_{j-1}] \dots [X_1(\boldsymbol{x}_1) \coloneqq \eta_1 X_1(\boldsymbol{x}_1).\psi_1].$$

Thus, for all $S \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$, we have that $\{\!|\psi_i|\!\}^{\mathcal{E}}_{\pi'[X_i(\boldsymbol{x}_i) \mapsto S]} =$

$$\{\!|\theta_i[X_{i-1}(\boldsymbol{x}_{i-1}) \coloneqq \eta_{i-1} X_{i-1}(\boldsymbol{x}_{i-1}).\psi_{i-1}] \dots [X_1(\boldsymbol{x}_1) \coloneqq \eta_1 X_1(\boldsymbol{x}_1).\psi_1]|\!\}^{\mathcal{E}}_{\pi'[X_i(\boldsymbol{x}_i) \mapsto S]}.$$

Furthermore, by repeatedly applying Lemma 4.1(2) we obtain that $\{\!|\psi_i|\!\}^{\mathcal{E}}_{\pi'[X_i(\boldsymbol{x}_i) \mapsto S]} = \{\!|\theta_i|\!\}^{\mathcal{E}}_{\pi_i}$ where $\pi_1 = \pi'[X_i(\boldsymbol{x}_i) \mapsto S]$ and for all $j \in [1, i-1]$,

$$\pi_{j+1} = \pi_j[X_{i-j}(\boldsymbol{x}_{i-j}) \mapsto \{\!|\eta_{i-j} X_{i-j}(\boldsymbol{x}_{i-j}).\psi_{i-j}|\!\}^{\mathcal{E}}_{\pi_j}].$$

Note that actually $\pi_i =$

$$\pi'[X_i(\boldsymbol{x}_i) \mapsto S][X_{i-1}(\boldsymbol{x}_{i-1}) \mapsto \{\!|\eta_{i-1} X_{i-1}(\boldsymbol{x}_{i-1}).\psi_{i-1}|\!\}^{\mathcal{E}}_{\pi_1}] \dots [X_1(\boldsymbol{x}_1) \mapsto \{\!|\eta_1 X_1(\boldsymbol{x}_1).\psi_1|\!\}^{\mathcal{E}}_{\pi_{i-1}}]$$

Now we just need to prove that $\pi_i = \pi''$. To show this we can use the inductive hypothesis $i-1$ times, recalling the recursive structure of the solutions of systems. Eventually we conclude that $\{\!|\psi_i|\!\}^{\mathcal{E}}_{\pi'[X_i(\boldsymbol{x}_i) \mapsto S]} = \{\!|\theta_i|\!\}^{\mathcal{E}}_{\pi_i} = \{\!|\theta_i|\!\}^{\mathcal{E}}_{\pi''}$ for all $S \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$, and so $\{\!|\eta_i X_i(\boldsymbol{x}_i).\psi_i|\!\}^{\mathcal{E}}_{\pi'} = sol_i(E')$.

Let us now focus on the second part. We prove that for all $i \in [0, m]$, if the proposition environment π coincides with the one in the system E_i on the free propositions of every

formula $\varphi_k^{E_i}$, then $sol_j(E_i) = \{\!|\varphi_j^{E_i}|\!\}_\pi^{\mathcal{E}}$ for all $j \in [1,i]$. This implies that $sol_i(E) = \{\!|\varphi_i^{E}|\!\}_\pi^{\mathcal{E}}$ for all $i \in [1,m]$ since $E = E_m$ and, as we already mentioned, every formula in $\boldsymbol{\varphi}^E$ has no free propositions, hence the condition on the proposition environment π vacuously holds for any one. The proof proceeds by induction on i.

($i = 0$) The property vacuously holds.

($i > 0$) We assume that π coincides with the proposition environment in the system E_i on the free propositions of every formula $\varphi_k^{E_i}$. Actually, for convenience we assume that the two proposition environments are exactly the same π. This is not a restriction because the evaluations of propositions appearing free in the formulae are already guaranteed to be the same and all the other are irrelevant. Then, we need to show that $sol_j(E_i) = \{\!|\varphi_j^{E_i}|\!\}_\pi^{\mathcal{E}}$ for all $j \in [1,i]$.

First we show the property for the last index $j = i$. By definition of solution we have $sol_i(E_i) = \eta_i(\lambda S.\{\!|\psi_i|\!\}_{\pi'}^{\mathcal{E}})$ where $\pi' = \pi[X_i(\boldsymbol{x}_i) \mapsto S][X_k(\boldsymbol{x}_k) \mapsto sol_k(E')]_{\forall k \in [1,i-1]}$ and $E' = E_i[v_i := S]$. By Definition 7.23 we know that $\varphi_i^{E_i} = \eta_i X_i(\boldsymbol{x}_i).\psi_i[X_k(\boldsymbol{x}_k) := \varphi_k^{E_{i-1}}]_{\forall k \in [1,i-1]}$. Then, the denotation of the formula is

$$\{\!|\varphi_i^{E_i}|\!\}_\pi^{\mathcal{E}} = \eta_i(\lambda S.\{\!|\psi_i[X_k(\boldsymbol{x}_k) := \varphi_k^{E_{i-1}}]_{\forall k \in [1,i-1]}|\!\}_{\pi[X_i(\boldsymbol{x}_i) \mapsto S]}^{\mathcal{E}}).$$

By repeatedly applying Lemma 4.1(2) we obtain that, for all $S \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$

$$\{\!|\psi_i[X_k(\boldsymbol{x}_k) := \varphi_k^{E_{i-1}}]_{\forall k \in [1,i-1]}|\!\}_{\pi[X_i(\boldsymbol{x}_i) \mapsto S]}^{\mathcal{E}} = \{\!|\psi_i|\!\}_{\pi''}^{\mathcal{E}}$$

where $\pi'' = \pi[X_i(\boldsymbol{x}_i) \mapsto S][X_k(\boldsymbol{x}_k) \mapsto \{\!|\varphi_k^{E_{i-1}}|\!\}_{\pi[X_i(\boldsymbol{x}_i) \mapsto S]}^{\mathcal{E}}]_{\forall k \in [1,i-1]}$. Now we just need to show that $\pi'' = \pi'$, that is, $\{\!|\varphi_k^{E_{i-1}}|\!\}_{\pi[X_i(\boldsymbol{x}_i) \mapsto S]}^{\mathcal{E}} = sol_k(E')$ for all $k \in [1, i-1]$ and $S \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$. Observe that by definition of substitution and Definition 7.23 necessarily the systems E' is E_{i-1} with the proposition environment $\pi[X_i(\boldsymbol{x}_i) \mapsto S]$. Then, since E_{i-1} has $i-1$ equations, by inductive hypothesis we immediately conclude that $sol_k(E') = \{\!|\varphi_k^{E_{i-1}}|\!\}_{\pi[X_i(\boldsymbol{x}_i) \mapsto S]}^{\mathcal{E}}$ for all $S \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ and $k \in [1, i-1]$, hence $\pi'' = \pi'$, and so $sol_i(E_i) = \{\!|\varphi_i^{E_i}|\!\}_\pi^{\mathcal{E}}$.

Now, it is easy to extend the result to the other $j \in [1, i-1]$. Indeed, since we just proved that $sol_k(E_i[v_i := S]) = \{\!|\varphi_k^{E_{i-1}}|\!\}_{\pi[X_i(\boldsymbol{x}_i) \mapsto S]}^{\mathcal{E}}$ for all $S \subseteq \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ and $k \in [1, i-1]$, it is enough to observe that, if we take $S = sol_i(E_i) = \{\!|\varphi_i^{E_i}|\!\}_\pi^{\mathcal{E}}$, we obtain that for all $j \in [1, i-1]$

$$sol_j(E_i[v_i := sol_i(E_i)]) = sol_j(E_i) = \{\!|\varphi_j^{E_{i-1}}|\!\}_{\pi[X_i(\boldsymbol{x}_i) \mapsto \{\!|\varphi_i^{E_i}|\!\}_\pi^{\mathcal{E}}]}^{\mathcal{E}} = \{\!|\varphi_j^{E_{i-1}}[X_i(\boldsymbol{x}_i) := \varphi_i^{E_i}]|\!\}_\pi^{\mathcal{E}}$$

where the last equality holds by Lemma 4.1(2). Then, noticing that by Definition 7.23 we have $\varphi_j^{E_i} = \varphi_j^{E_{i-1}}[X_i(\boldsymbol{x}_i) := \varphi_i^{E_i}]$, we can conclude that $sol_j(E_i) = \{\!|\varphi_j^{E_i}|\!\}_\pi^{\mathcal{E}}$. □

7.5.2 Fixpoint game for model checking

We first discuss the model-checking of $\mathcal{L}_{hp}$ over event structures using the fixpoint game. Since the game may in general be infinite, we will need some countermeasure similar to those adopted in the other procedures. We close the section with a(n informal) comparison

between the model-checking techniques presented in the previous chapters and the one proposed here.

Given a closed fixpoint formula φ of $\mathcal{L}_{hp}$, using the procedure presented in the previous subsection, we get the system of m equations associated with φ, and by Proposition 7.5 we are guaranteed that the denotation of φ is exactly the solution of the m-th equation of the system. Furthermore, the lattice over which the system is defined is clearly continuous since it is a powerset lattice. Taking the (possibly infinite) set of all singletons as a basis, we can build the fixpoint game corresponding to the system. Then, if $\exists$ has a winning strategy from a position (b, m), we know that the initial formula holds in b. This means that searching for a winning strategy from such a position corresponds, in some sense, to solving a local model-checking problem. Moreover, computing the progress measure associated with the fixpoint game would allow us to perform the global model-checking of the formula over the model.

In principle everything works fine, however, notice that, without any countermeasure, also in this case we encounter the usual problem with infiniteness that we already had with the other methods: for every infinite PES and fixpoint formula the game graph is infinite. Let $\mathcal{E}$ be a PES, let φ be a closed fixpoint formula of $\mathcal{L}_{hp}$ and let E be the corresponding system of m equations as in Definition 7.22 with equational form $\{X_i(\boldsymbol{x}_i) =_{\eta_i} \theta_i\}_{i\in[1,m]}$. Then, equations of the system are over the powerset lattice $L = \langle \mathbf{2}^{\mathcal{C}(\mathcal{E})\times Env_{\mathcal{E}}}, \subseteq\rangle$, with basis $B = \{\{(C, \rho)\} \mid C \in \mathcal{C}(\mathcal{E}) \wedge \rho \in Env_{\mathcal{E}}\}$. Clearly, when the PES is infinite, the lattice and the basis are also infinite, and so are the positions of the fixpoint game. While the game is still sound and complete by continuity of the lattice, practical attempts to search for winning strategies from some positions may lead to non-termination.

As we did for the other methods we can avoid this problem by considering positions up to some equivalence preserving, in this case, winning strategies. We already know that configurations, with isomorphic residuals pointed by the free variables of a formula, either both satisfy the formula or none of them does (see Lemma 5.3). Since each position $(\{(C, \rho)\}, i)$ of player $\exists$ is associated with the formula θ_i in the i-th equation $X_i(\boldsymbol{x}_i) =_{\eta_i} \theta_i$ of the system, and we know that the game is sound and complete, then for two positions $(\{(C_1, \rho_1)\}, i)$ and $(\{(C_2, \rho_2)\}, i)$, such that the corresponding $\boldsymbol{x}_i$-pointed configurations are $\approx_r$-equivalent $\langle C_1, \rho_1|_{\boldsymbol{x}_i}\rangle \approx_r \langle C_2, \rho_2|_{\boldsymbol{x}_i}\rangle$, either player $\exists$ has a winning strategy from both or none of them. Thus we can safely quotient the positions of player $\exists$ by considering, for each $i \in [1, m]$, only the $\approx_r$-equivalence classes of the corresponding $\boldsymbol{x}_i$-pointed configurations. Positions of $\forall$ can also be quotiented by deeming equivalent positions from which the possible moves of $\forall$ are the same up to the equivalence above. Then, the formal definition of the equivalence over positions of the game is given below.

Definition 7.24 (strategy equivalence)**.** *Let $\mathcal{E}$ be a PES, let φ be a closed fixpoint formula of $\mathcal{L}_{hp}$ and let E be the corresponding system of m equations with equational form $\{X_i(\boldsymbol{x}_i) =_{\eta_i} \theta_i\}_{i\in[1,m]}$. We say that two positions p, p' of the same player in the fixpoint game are* strategy

equivalent, *written* $p \approx_g p'$, *when either*

- p, p' *are positions of player* $\exists$ *with the same index* i, *then* $p = (\{(C, \rho)\}, i)$ *and* $p' = (\{(C', \rho')\}, i)$, *and* $\langle C, \rho|_{\boldsymbol{x}_i}\rangle \approx_r \langle C', \rho'|_{\boldsymbol{x}_i}\rangle$
- p, p' *are positions of player* $\forall$ *and the sets of possible moves of* $\forall$ *from* p *and* p' *are the same up to* $\approx_g$*-equivalence, i.e.,* $\mathbf{A}(p)_{/\approx_g} = \mathbf{A}(p')_{/\approx_g}$.

Note that the definition above is well-given because moves of player $\forall$ are always positions of player $\exists$.

Let G be the fixpoint game for a system of equations like the one given before. The positions of the quotient game $G_{/\approx_g}$ are the equivalence classes of the positions of G with respect to $\approx_g$. In $G_{/\approx_g}$ the moves of $\forall$ from a position $[p]_{\approx_g}$ are simply the quotient $\mathbf{A}(p)_{/\approx_g}$ of the moves in G from any position in that equivalence class (since by definition they are all the same up to $\approx_g$). Instead, the moves of $\exists$ from a position $[p]_{\approx_g}$ are defined by $\mathbf{E}([p]_{\approx_g}) = \{[q]_{\approx_g} \mid p' \approx_g p \ \wedge \ q \in \mathbf{E}(p')\}$. We can show that winning strategies for player $\exists$ are preserved in the quotient game (and vice versa).

Proposition 7.6 (strategy preservation)**.** *Let* $\mathcal{E}$ *be a PES, let* φ *be a closed fixpoint formula of* $\mathcal{L}_{hp}$ *and let* E *be the corresponding system of* m *equations with equational form* $\{X_i(\boldsymbol{x}_i) =_{\eta_i} \theta_i\}_{i\in[1,m]}$. *There is a winning strategy for* $\exists$ *from a position* $(\{(C, \rho)\}, i)$ *in the fixpoint game associated with* E *if and only if there is one from* $[(\{(C, \rho)\}, i)]_{\approx_g}$ *in the quotient game.*

Proof. The proof is similar to that of Theorem 6.1 combined with that of Proposition 6.1. The reasoning is roughly as follows. Assume that there is a winning strategy for $\exists$ from position $(\{(C, \rho)\}, i)$, then there is one, say ς, such that for all $p, p' \in B \times [1, m]$, if $p \approx_g p'$ then $\varsigma(p) \approx_g \varsigma(p')$. Then, the function ς' defined by $\varsigma'([p]_{\approx_g}) = [\varsigma(p)]_{\approx_r}$ is well-defined and it is a winning strategy for $\exists$ from $[(\{(C, \rho)\}, i)]_{\approx_g}$ in the quotient game, since ς was a winning strategy for $\exists$ from $(\{(C, \rho)\}, i)$.

Assume that there is a winning strategy ς for $\exists$ from $[(\{(C, \rho)\}, i)]_{\approx_g}$ in the quotient game, then we can define a strategy $\varsigma' : B \times [1, m] \to L^m$ as follows. For every $(b, i) \in B \times [1, m]$, if $\varsigma([(b, i)]_{\approx_g}) = [\boldsymbol{S}]_{\approx_g}$, where $\boldsymbol{S} \in L^m$, then there must exists $(b', i) \in B \times [1, m]$ such that $(b', i) \approx_g (b, i)$ and $\boldsymbol{S} \in \mathbf{E}(b', i)$. For all $j \in [1, m]$ let $S'_j = \{(C', \rho') \mid \exists\, (C'', \rho'') \in S_j . \langle C'', \rho''|_{\boldsymbol{x}_j}\rangle \approx_r \langle C', \rho'|_{\boldsymbol{x}_j}\rangle\}$, then clearly $\boldsymbol{S} \subseteq \boldsymbol{S'} \in \mathbf{E}(b', i)$, which means that $b' \in \{\!|\theta_i|\!\}^{\mathcal{E}}_{\pi'}$, where $\pi' = \pi[X_j(\boldsymbol{x}_j) \mapsto S'_j]_{\forall j\in[1,m]}$ for any π. Since π' is saturated, by Lemma 5.3 we have that also $b \in \{\!|\theta_i|\!\}^{\mathcal{E}}_{\pi'}$, and thus $\boldsymbol{S'} \in \mathbf{E}(b, i)$. And so we may define the strategy as $\varsigma'(b, i) = \boldsymbol{S'}$. Observing that by definition of $\boldsymbol{S'}$ it holds also $\mathbf{A}(\varsigma'(b, i))_{/\approx_g} = \mathbf{A}(\boldsymbol{S})_{/\approx_g} = \mathbf{A}(\varsigma([(b, i)]_{\approx_g}))$ for all $(b, i) \in B \times [1, m]$, we can deduce that the strategy ς' is winning for $\exists$ from $(\{(C, \rho)\}, i)$, since ς was a winning strategy for $\exists$ from $[(\{(C, \rho)\}, i)]_{\approx_g}$. □

Moreover, when the PES is strongly regular, it is easy to see that the quotient game is finite.

Proposition 7.7 (quotient game finiteness)**.** *Let $\mathcal{E}$ be a strongly regular PES, let φ be a closed fixpoint formula of $\mathcal{L}_{hp}$ and let E be the corresponding system of m equations. The equivalence $\approx_g$ on the fixpoint game associated with E is of finite index.*

Proof. Observe that, for a fixed bounded set of event variables $\boldsymbol{x}$, by Lemma 5.4 there is a finite number of $\approx_r$-equivalence classes of $\boldsymbol{x}$-pointed configurations. Since the event variables $\boldsymbol{x}_i$ associated with each equation are bounded, we immediately deduce that there is a finite number of $\approx_g$-equivalence classes of positions $(\{(C,\rho)\},i) \in B \times [1,m]$ of player $\exists$. In turn, since for every position $\boldsymbol{S}$ of $\forall$ we know that $\mathbf{A}(\boldsymbol{S}) \subseteq B \times [1,m]$, this implies that also there is a finite number of $\approx_g$-equivalence classes of positions of $\forall$. □

Together these results allow us to verify whether a strongly regular PES $\mathcal{E}$ satisfies a closed formula φ of $\mathcal{L}_{hp}$, by searching for a winning strategy for player $\exists$ from position $[(\{(\emptyset,\rho)\},m)]_{\approx_g}$ in the quotient game, where the environment ρ is irrelevant since φ is closed.

To get a better understanding, we show how this works with an example. Consider the formula $\langle\!|\mathsf{b}\,x|\!\rangle\varphi$ from Subsection 4.3.3, where $\varphi = \nu X(x).\psi$ and $\psi = \mu Y(x).(\langle\!|x < \mathsf{b}\,y|\!\rangle X(y) \vee \langle\!|\_\, z|\!\rangle Y(x))$, and the PES $\mathcal{E}_3$ in Figure 3.1c. Note that the formula can be rewritten, without changing its denotation, as $\mu Z.\langle\!|\mathsf{b}\,x|\!\rangle\varphi$, in order to have a fixpoint formula. Then, the corresponding equational form would be

$$\begin{aligned} Y(x) &=_\mu \langle\!|x < \mathsf{b}\,y|\!\rangle X(y) \vee \langle\!|\_\, z|\!\rangle Y(x) \\ X(x) &=_\nu Y(x) \\ Z &=_\mu \langle\!|\mathsf{b}\,x|\!\rangle X(x) \end{aligned}$$

We call the right-hand side of the three equations θ_Y, θ_X and θ_Z, respectively. Fixing the model $\mathcal{E}_3$, this can be interpreted as the system E, over the powerset lattice $L = \mathbf{2}^{\mathcal{C}(\mathcal{E}_3)\times Env_{\mathcal{E}_3}}$ with the set of all singletons as basis B, given as

$$\begin{aligned} v_1 &=_\mu \{\!|\langle\!|x < \mathsf{b}\,y|\!\rangle X(y) \vee \langle\!|\_\, z|\!\rangle Y(x)|\!\}^{\mathcal{E}}_{\pi[X(x)\mapsto v_2][Y(x)\mapsto v_1]} \\ v_2 &=_\nu \{\!|Y(x)|\!\}^{\mathcal{E}}_{\pi[Y(x)\mapsto v_1]} \\ v_3 &=_\mu \{\!|\langle\!|\mathsf{b}\,x|\!\rangle X(x)|\!\}^{\mathcal{E}}_{\pi[X(x)\mapsto v_2]} \end{aligned}$$

where π is irrelevant and for each equation we specified only the evaluations for propositions appearing in the corresponding subformula. For instance, there is no evaluation for Z in any proposition environment since the proposition Z does not appear in any subformula. Supposing that we want to verify that $\mathcal{E}_3$ satisfies $(\mu Z.)\,\langle\!|\mathsf{b}\,x|\!\rangle\varphi$, in Figure 7.5 we depict a relevant part of the corresponding fixpoint game for plays starting from the position $(\{(\emptyset,\rho)\},3)$.

In the figure, the two curly lines represent moves obtained via the quotient, that is, moves that would actually go into another position that, however, is equivalent to the one where the arc actually goes. For instance, from position $(\{(\{\mathsf{b}^0\},\rho[x\mapsto \mathsf{b}^0])\},1)$ player $\exists$ can move to $(\emptyset,\{(\{\mathsf{b}^0,\mathsf{b}^1\},\rho[x\mapsto \mathsf{b}^1])\},\emptyset)$, which is $\approx_g$-equivalent to $(\emptyset,\{(\{\mathsf{b}^0\},\rho[x\mapsto \mathsf{b}^0])\},\emptyset)$ because

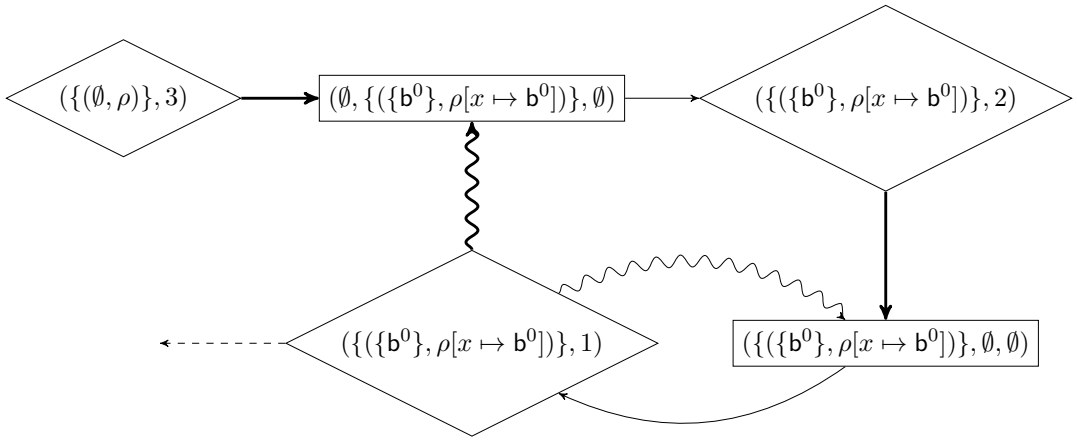


Figure 7.5: Quotient game obtained from the infinite fixpoint game for $\mu Z.\langle\!|\mathsf{b}\, x|\!\rangle\varphi$ in $\mathcal{E}_3$ via strategy equivalence.

from there the only possible move of $\forall$ is $(\{(\{\mathsf{b}^0, \mathsf{b}^1\}, \rho[x \mapsto \mathsf{b}^1])\}, 2)$, whose corresponding x-pointed configuration $\langle\{\mathsf{b}^0, \mathsf{b}^1\}, \rho[x \mapsto \mathsf{b}^1]|_x\rangle$ is $\approx_r$-equivalent to the one associated with the position $(\{(\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0])\}, 2)$. Therefore, the two positions of $\forall$ are merged and the arc from position $(\{(\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0])\}, 1)$ actually loops back to $(\emptyset, \{(\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0])\}, \emptyset)$ (taken as representative of their equivalence class). From $(\{(\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0])\}, 1)$ there is also another possible move for $\exists$, represented by the dashed line, which would go to position $(\{(\{\mathsf{b}^0, \mathsf{c}\}, \rho[x \mapsto \mathsf{b}^0, z \mapsto \mathsf{c}])\}, \emptyset, \emptyset)$ and then $(\{(\{\mathsf{b}^0, \mathsf{c}\}, \rho[x \mapsto \mathsf{b}^0, z \mapsto \mathsf{c}])\}, 1)$ after the only possible move of $\forall$. This position is not equivalent to any other in the figure, and it would lead to other reachable positions of the game, but they are not shown since they are not relevant in this case. Indeed, with just the displayed positions, we can already establish that $\exists$ has a winning strategy from $(\{(\emptyset, \rho)\}, 3)$. Such strategy, highlighted by the bold arrows in the figure, is given by the function $\varsigma : B \times [1, 3] \to L^3$ defined by

$$\begin{aligned}
\varsigma(\{(\emptyset, \rho)\}, 3) &= (\emptyset, \{(\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0])\}, \emptyset) \\
\varsigma(\{(\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0])\}, 2) &= (\{(\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0])\}, \emptyset, \emptyset) \\
\varsigma(\{(\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0])\}, 1) &= (\emptyset, \{(\{\mathsf{b}^0, \mathsf{b}^1\}, \rho[x \mapsto \mathsf{b}^1])\}, \emptyset) \approx_g (\emptyset, \{(\{\mathsf{b}^0\}, \rho[x \mapsto \mathsf{b}^0])\}, \emptyset).
\end{aligned}$$

On all other positions ς can have any value since they are irrelevant. If $\exists$ follows the strategy ς, starting from $(\{(\emptyset, \rho)\}, 3)$, we would obtain an infinite play where the highest index occurring infinitely often is 2, which is the index of a ν-equation, thus $\exists$ wins.

Comparing Figure 7.5 with Figure 6.2, showing the (quotiented) automaton built for the same formula and model, it should be recognisable that the underlying structure is the same. Roughly, positions of player $\exists$ here correspond to nodes of the automaton whose formula is a proposition. Moreover, the quotient we obtained in the case of the automaton is clearly very similar to the one for this game. It can be seen that accepting runs in the automaton correspond, in some sense, to trees of plays won by $\exists$, where $\forall$ plays all possible moves each time. Indeed, both the language of the automaton in Figure 6.2 is non-empty,

and player $\exists$ has a winning strategy from $(\{(\emptyset, \rho)\}, 3)$ in the game above. This should not be surprising since the acceptance conditions of NPAs is similar to the win condition for player $\exists$ in the fixpoint game.

We conclude by informally discussing also the relation between the fixpoint game and tableaux for $\mathcal{L}_{hp}$. We do this by using again the example above. Recall that the task was to verify that the PES $\mathcal{E}_3$ in Figure 3.1c satisfies the formula $\langle\!|\mathtt{b}\, x|\!\rangle\varphi$, where $\varphi = \nu X(x).\psi$ and $\psi = \mu Y(x).(\langle\!|x < \mathtt{b}\, y|\!\rangle X(y) \vee \langle\!|\_ z|\!\rangle Y(x))$. A successful tableau for such formula and model is given in Figure 5.2. It is not difficult to see that such tableau corresponds to the winning strategy ς for $\exists$ given above. In fact, consider the reduced tree in Figure 7.6, which is obtained from the tableau in Figure 5.2 by keeping only the sequents whose formula is a proposition, and adding a sequent for the dummy proposition Z at the root. Each sequent $C, \rho, \Delta \models P(p)$, where P is one of the three propositions, can be seen in the fixpoint game as a position $(\{(C, \rho)\}, i)$ of player $\exists$, where $i \in [1, 3]$ is the index of the equation corresponding to proposition P. The successor sequents correspond to the move from $(\{(C, \rho)\}, i)$ prescribed by the strategy ς. In this specific case all sequents have just one successor, so the strategy is very simple. For instance, the sequent $\{\mathtt{b}^0\}, \rho[x \mapsto \mathtt{b}^0], \Delta'' \models Y(x)$ corresponds to the position $(\{(\{\mathtt{b}^0\}, \rho[x \mapsto \mathtt{b}^0])\}, 1)$. The successor $\{\mathtt{b}^0, \mathtt{b}^1\}, \rho[x \mapsto \mathtt{b}^0, y \mapsto \mathtt{b}^1], \Delta'' \models X(y)$ determines the move prescribed by the strategy, that is, $\varsigma(\{(\{\mathtt{b}^0\}, \rho[x \mapsto \mathtt{b}^0])\}, 1) = (\emptyset, \{(\{\mathtt{b}^0, \mathtt{b}^1\}, \rho[x \mapsto \mathtt{b}^1])\}, \emptyset)$, where, technically, variable y in the environment has been renamed to x due to the difference between the propositions $X(y)$ in the sequent and $X(x)$ of the second equation of the system. In general, up to the due renamings of event variables, positions of the game and sequents of tableaux share the same intimate relationship observed before with automata. Indeed, it can also be seen that plays in the fixpoint game correspond, in some sense, to paths in reduced tableaux (not necessarily successful, e.g., plays won by player $\forall$). In this case, for example, the only path in the reduced tableau in Figure 7.6 corresponds to the play where player $\exists$ follows the strategy ς.

$$\dfrac{\dfrac{\dfrac{\emptyset, \rho, \Delta \models Z}{\{\mathtt{b}^0\}, \rho[x \mapsto \mathtt{b}^0], \Delta' \models X(x)}}{\{\mathtt{b}^0\}, \rho[x \mapsto \mathtt{b}^0], \Delta'' \models Y(x)}}{\{\mathtt{b}^0, \mathtt{b}^1\}, \rho[x \mapsto \mathtt{b}^0, y \mapsto \mathtt{b}^1], \Delta'' \models X(y)}$$

Figure 7.6: Reduced tree obtained from a successful tableau for $\mu Z.\langle\!|\mathtt{b}\, x|\!\rangle\varphi$ in $\mathcal{E}_3$.

Chapter 8

Related logics

In this final chapter of the thesis we study some related work on logics for true concurrency. We review some other expressive logics for true concurrency from the literature, comparing their expressive power with that of $\mathcal{L}_{hp}$. All the logics that will be presented are event-based, naturally interpreted over event structures, so that we can easily compare them with $\mathcal{L}_{hp}$. Furthermore, they are all modal logics, therefore the main differences between them and $\mathcal{L}_{hp}$ lie in the modalities. Then, we show how $\mathcal{L}_{hp}$ can be extended with operators taken from the other logics inexpressible in $\mathcal{L}_{hp}$, obtaining in this way a more powerful logic for true concurrency. We conclude proving that the model-checking problem in the extended logic is still decidable over strongly regular event structures, by providing a decision procedure based on an adaptation of one of those presented in the previous chapters.

8.1 Event identifier logic

In [PU14] the authors propose an extension of HML called *event identifier logic* (EIL), originally introduced in [PU11]. The most relevant difference with respect to $\mathcal{L}_{hp}$ is that EIL allows for forward as well as backward modalities. The first are alike those of HML, the second instead allow to reverse steps already executed, under suitable conditions. These operators make the logic sufficiently expressive to provide a logical characterisation of hhp-bisimilarity, intuitively because the possibility of performing backward steps can be a mean of exploring alternative different futures.

Similarly to $\mathcal{L}_{hp}$, modal operators of EIL include event variables, taken from a set Var, which can be bound to events in computations. The diamond forward modality $\langle x : \mathsf{a} \rangle\!\rangle \varphi$ declares the executability of an a-labelled event, which is bound to variable x, and then φ must hold in the resulting configuration. The diamond backward modality $\langle\!\langle x \rangle \varphi$ requires that the event bound to x can be undone from the current configuration, reaching a state where φ holds. There is also a third operator $(x : \mathsf{a})\varphi$ which states that there is an a-labelled event executed in the past, such that, binding it to x, the formula φ holds. Originally in [PU11, PU14] the logic was introduced with negation. In accordance with the rest of the

thesis, here we give the syntax of EIL in positive form, without negation, with all due dual operators.

Definition 8.1 (syntax of EIL)**.** *The syntax of EIL over the sets of event variables Var and labels Λ is defined as follows:*

$$\begin{aligned}\varphi \;::=\;& \mathsf{T} \mid \varphi \wedge \varphi \mid \langle x : \mathsf{a} \rangle\!\rangle\, \varphi \mid (x : \mathsf{a})\, \varphi \mid \langle\!\langle x \rangle\, \varphi \mid \\ & \mathsf{F} \mid \varphi \vee \varphi \mid [x : \mathsf{a}]]\, \varphi \mid |x : \mathsf{a}|\, \varphi \mid [[x]\, \varphi\end{aligned}$$

The free variables of a formula can be defined in the obvious way, similarly to $\mathcal{L}_{hp}$ (Definition 4.7). Just observe that $fv(\langle x : \mathsf{a} \rangle\!\rangle\, \varphi) = fv((x : \mathsf{a})\, \varphi) = fv(\varphi) \setminus \{x\}$, while $fv(\langle\!\langle x \rangle\, \varphi) = fv(\varphi) \cup \{x\}$, and the same for the dual operators. Substitutions of variables are also allowed analogously to $\mathcal{L}_{hp}$.

The satisfaction is again defined with respect to a configuration and an environment. Therefore, the semantics of formulae of EIL over a PES $\mathcal{E}$ are sets of pairs in $\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$.

Definition 8.2 (semantics of EIL)**.** *Let $\mathcal{E}$ be a PES. The denotation of a formula in EIL is given by the function $\{\!|\cdot|\!\}^{\mathcal{E}} : EIL \to \mathbf{2}^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}$ defined inductively as follows:*

$$\begin{aligned}
\{\!|\mathsf{T}|\!\}^{\mathcal{E}} &= \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \\
\{\!|\mathsf{F}|\!\}^{\mathcal{E}} &= \emptyset \\
\{\!|\varphi_1 \wedge \varphi_2|\!\}^{\mathcal{E}} &= \{\!|\varphi_1|\!\}^{\mathcal{E}} \cap \{\!|\varphi_2|\!\}^{\mathcal{E}} \\
\{\!|\varphi_1 \vee \varphi_2|\!\}^{\mathcal{E}} &= \{\!|\varphi_1|\!\}^{\mathcal{E}} \cup \{\!|\varphi_2|\!\}^{\mathcal{E}} \\
\{\!|\langle x : \mathsf{a} \rangle\!\rangle\, \varphi|\!\}^{\mathcal{E}} &= \{(C, \rho) \mid \exists\, e \in E.\ C \xrightarrow{e}_{\mathsf{a}} C' \ \wedge\ (C', \rho[x \mapsto e]) \in \{\!|\varphi|\!\}^{\mathcal{E}}\} \\
\{\!|[x : \mathsf{a}]]\, \varphi|\!\}^{\mathcal{E}} &= \{(C, \rho) \mid \forall\, e \in E.\ C \xrightarrow{e}_{\mathsf{a}} C' \ \Rightarrow\ (C', \rho[x \mapsto e]) \in \{\!|\varphi|\!\}^{\mathcal{E}}\} \\
\{\!|(x : \mathsf{a})\, \varphi|\!\}^{\mathcal{E}} &= \{(C, \rho) \mid \exists\, e \in C.\ \lambda(e) = \mathsf{a} \ \wedge\ (C, \rho[x \mapsto e]) \in \{\!|\varphi|\!\}^{\mathcal{E}}\} \\
\{\!||x : \mathsf{a}|\, \varphi|\!\}^{\mathcal{E}} &= \{(C, \rho) \mid \forall\, e \in C.\ \lambda(e) = \mathsf{a} \ \Rightarrow\ (C, \rho[x \mapsto e]) \in \{\!|\varphi|\!\}^{\mathcal{E}}\} \\
\{\!|\langle\!\langle x \rangle\, \varphi|\!\}^{\mathcal{E}} &= \{(C, \rho) \mid \exists\, C' \in \mathcal{C}(\mathcal{E}).\ C' \xrightarrow{\rho(x)} C \ \wedge\ \rho(fv(\varphi)) \subseteq C' \ \wedge\ (C', \rho) \in \{\!|\varphi|\!\}^{\mathcal{E}}\} \\
\{\!|[[x]\, \varphi|\!\}^{\mathcal{E}} &= \{(C, \rho) \mid \forall\, C' \in \mathcal{C}(\mathcal{E}).\ (C' \xrightarrow{\rho(x)} C \ \wedge\ \rho(fv(\varphi)) \subseteq C') \ \Rightarrow\ (C', \rho) \in \{\!|\varphi|\!\}^{\mathcal{E}}\}
\end{aligned}$$

When $(C, \rho) \in \{\!|\varphi|\!\}^{\mathcal{E}}$ we say that the PES $\mathcal{E}$ satisfies the formula φ in the configuration C and environment ρ. When φ is closed, hence the environment ρ is irrelevant, and $\mathcal{E}$ satisfies the formula φ in the empty configuration, we simply say that $\mathcal{E}$ satisfies φ.

The formula $\langle x : \mathsf{a} \rangle\!\rangle\, \varphi$ holds in (C, ρ) when from configuration C an a-labelled event e can be executed producing a configuration $C' = C \cup \{e\}$ which, paired with the updated environment $\rho[x \mapsto e]$, satisfies the rest of the formula φ. Dually, $[x : \mathsf{a}]]\, \varphi$ requires that all a-labelled events executable from C lead to a configuration where φ holds.

The operator $(x : \mathsf{a})\, \varphi$ is satisfied by (C, ρ) if there exists an a-labelled event e in the configuration C such that the environment produced from ρ by binding e to the variable x, i.e., $\rho[x \mapsto e]$, paired with the configuration C, satisfies φ. Dually, $|x : \mathsf{a}|\, \varphi$ requires that φ

holds on the environments obtained by all the possible different bindings of the variable x to an a-labelled event in C.

The backward modality $\langle\langle x\rangle\rangle\,\varphi$ holds in (C,ρ) if there is a configuration C' from where the event $\rho(x)$ could have been executed in order to produce the configuration C (i.e., $C' = C \setminus \{\rho(x)\} \in \mathcal{C}(\mathcal{E})$), and then the pair (C',ρ) satisfies the formula φ, with the additional requirement that the reversed event e is not bound to variables appearing free in φ. The latter is required because free variables must always be bound to events in the configuration, and this is not guaranteed any more since the backward modalities can remove events from the configurations by reversing them. The dual $[[x]]\,\varphi$ requires that φ holds on all the configurations from where it is possible to execute the event $\rho(x)$, reaching the configuration C, and the aforementioned constrain on free variables still holds. Note that if such a configuration exists, it must be $C \setminus \{\rho(x)\}$, otherwise the formula vacuously holds.

Notice that the forward and backward modalities are not capable to assert explicitly the dependency relations between the computational steps. However, causality and concurrency between actions can be captured by properly combining such operators. For example, the formula $\langle x : \mathsf{a}\rangle\rangle\langle y : \mathsf{a}\rangle\rangle\langle\langle x\rangle\rangle\mathsf{T}$ expresses the possibility to execute two a-labelled events and then reverse the first one. This necessarily requires that the two events are concurrent, otherwise the first could not be reversed after executing the second one.

As already mentioned, the logical equivalence induced by EIL is hhp-bisimilarity. In [PU11, PU14] fragments of the logic are identified which correspond to other true concurrent behavioural equivalences. In particular, here we report the fragment, referred to as $\mathrm{EIL_h}$, characterising hp-bisimilarity. The operators of $\mathrm{EIL_h}$ are the same of EIL, however forward modalities are no longer allowed after backward modalities. Therefore, the syntax of $\mathrm{EIL_h}$ can be formulated as follows.

Definition 8.3 (syntax of $\mathrm{EIL_h}$)**.** *The syntax of EIL_h over the sets of event variables Var and labels Λ is defined as follows:*

$$\varphi \;::=\; \mathsf{T} \;\mid\; \mathsf{F} \;\mid\; \varphi \wedge \varphi \;\mid\; \varphi \vee \varphi \;\mid\; \langle x : \mathsf{a}\rangle\rangle\,\varphi \;\mid\; [x : \mathsf{a}]]\,\varphi \;\mid\; (x : \mathsf{a})\,\varphi \;\mid\; |x : \mathsf{a}|\,\varphi \;\mid\; \varphi_r$$

where φ_r, the reverse-only *formulae, are defined by*

$$\varphi_r \;::=\; \mathsf{T} \;\mid\; \mathsf{F} \;\mid\; \varphi_r \wedge \varphi_r \;\mid\; \varphi_r \vee \varphi_r \;\mid\; (x : \mathsf{a})\,\varphi_r \;\mid\; |x : \mathsf{a}|\,\varphi_r \;\mid\; \langle\langle x\rangle\rangle\,\varphi_r \;\mid\; [[x]]\,\varphi_r$$

Since only reverse-only formulae can contain backward modalities, and their syntax does not include forward modalities, it follows that forward modalities cannot occur after any backward modalities. Variable bindings are still permitted in reverse-only formulae via the operator $(x : \mathsf{a})$ and its dual version.

The expressive power of full EIL is clearly greater than that of $\mathcal{L}_{hp}$, since the induced logical equivalence is hhp-bisimilarity, while $\mathcal{L}_{hp}$ induces the coarser hp-bisimilarity. An interesting problem would be to study the relation between EIL and the whole logic in

[BC14] whose induced logical equivalence is hhp-bisimilarity as well. Although hinted in [BC14], the formal relationship between such logics is still to be understood and represents a direction of future research. Here, instead, in the spirit of the work, we focus on the fragment $\mathrm{EIL_h}$ inducing hp-bisimilarity for the comparison with $\mathcal{L}_{hp}$. In the following we compare the two logics, showing that $\mathrm{EIL_h}$ can be fully encoded in $\mathcal{L}_{hp}$. Moreover, the model-checking problem in EIL (and $\mathrm{EIL_h}$) has not yet been investigated. As a secondary result, in the next subsection we also show that the model-checking in $\mathrm{EIL_h}$ is decidable over strongly regular PESs.

8.1.1 Encoding $\mathrm{EIL_h}$

Here we show that $\mathrm{EIL_h}$ can be encoded in $\mathcal{L}_{hp}$. Therefore, we know that every property expressible in $\mathrm{EIL_h}$ is also captured by some formula of $\mathcal{L}_{hp}$, i.e., $\mathcal{L}_{hp}$ is at least as expressive as $\mathrm{EIL_h}$. It also implies that the model-checking in $\mathrm{EIL_h}$ is decidable over strongly regular PESs, by reduction to $\mathcal{L}_{hp}$.

The encoding uses functions to remember the variables bound by forward modalities, their causal dependencies with past variables, and their labels. Function $\gamma : \mathit{Var} \to \mathbf{2}^{\mathit{Var}}$ associates variables with the set of past variables asserted to cause them. Function $l : \mathit{Var} \to \Lambda$ associates variables with their labels. Note that γ and l are partial functions updated at every forward or backward step so that they are defined only on the subset of variables which, at that point, would be bound to events in the configuration, represented by their domain. We assume that in every formula of $\mathrm{EIL_h}$ different bound variables have different names. This requirement can always be fulfilled by alpha-renaming. Then, the encoding procedure is as follows:

$$
\begin{aligned}
[\mathsf{T}](\gamma,l) &= \mathsf{T} & [\mathsf{F}](\gamma,l) &= \mathsf{F}\\
[\varphi \wedge \psi](\gamma,l) &= [\varphi](\gamma,l) \wedge [\psi](\gamma,l) & [\varphi \vee \psi](\gamma,l) &= [\varphi](\gamma,l) \vee [\psi](\gamma,l)\\
[\langle x : \mathsf{a}\rangle\rangle\, \varphi](\gamma,l) &= \bigvee_{\boldsymbol{y} \subseteq dom(\gamma)} \langle\!\langle \boldsymbol{y}, \overline{\boldsymbol{y}^c} < \mathsf{a}\, x \rangle\!\rangle\, [\varphi](\gamma[x \mapsto \boldsymbol{y}], l[x \mapsto \mathsf{a}])\\
[[x : \mathsf{a}]]\, \varphi](\gamma,l) &= \bigwedge_{\boldsymbol{y} \subseteq dom(\gamma)} [\![\boldsymbol{y}, \overline{\boldsymbol{y}^c} < \mathsf{a}\, x]\!]\, [\varphi](\gamma[x \mapsto \boldsymbol{y}], l[x \mapsto \mathsf{a}])\\
[(x : \mathsf{a})\, \varphi](\gamma,l) &= \bigvee_{z \in dom(\gamma).\, l(z) = \mathsf{a}} [\varphi[z/x]](\gamma,l)\\
[|x : \mathsf{a}|\, \varphi](\gamma,l) &= \bigwedge_{z \in dom(\gamma).\, l(z) = \mathsf{a}} [\varphi[z/x]](\gamma,l)\\
[\langle\!\langle x \rangle\!\rangle\, \varphi](\gamma,l) &= \begin{cases} [\varphi](\gamma|_{\{x\}^c}, l|_{\{x\}^c}) & \text{if } \forall\, z \in dom(\gamma).\ x \notin \gamma(z) \text{ and } \mathit{fv}(\varphi) \subseteq \{x\}^c \\ \mathsf{F} & \text{otherwise} \end{cases}\\
[[[x]\, \varphi](\gamma,l) &= \begin{cases} [\varphi](\gamma|_{\{x\}^c}, l|_{\{x\}^c}) & \text{if } \forall\, z \in dom(\gamma).\ x \notin \gamma(z) \text{ and } \mathit{fv}(\varphi) \subseteq \{x\}^c \\ \mathsf{T} & \text{otherwise} \end{cases}
\end{aligned}
$$

where $V^c = dom(\gamma) \setminus V$ for a set of variables V, empty conjunctions are always true and empty disjunctions are always false.

The procedure allows for the encoding of every closed formula φ of $\mathrm{EIL_h}$ by computing the formula $[\varphi](\emptyset, \emptyset)$ of $\mathcal{L}_{hp}$.

In order to prove the correctness of the encoding procedure, we need some technical lemmata. First, we show that the encodings of reverse-only formulae actually are either always satisfied or never, depending on the formula and the functions γ and l.

Lemma 8.1 (encoding of a reverse-only formula is T or F)**.** *Let $\mathcal{E}$ be a PES and let φ be a reverse-only formula. Given γ and l s.t. $fv(\varphi) \subseteq dom(\gamma) = dom(l)$, either $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ or $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \emptyset$.*

Proof. The proof proceeds by induction on the formula φ, we show only some significant cases:

- $\varphi = \psi_1 \wedge \psi_2$ and $[\varphi](\gamma, l) = [\psi_1](\gamma, l) \wedge [\psi_2](\gamma, l)$
  By definition of the semantics $\{\!|[\psi_1](\gamma, l) \wedge [\psi_2](\gamma, l)|\!\}^{\mathcal{E}} = \{\!|[\psi_1](\gamma, l)|\!\}^{\mathcal{E}} \cap \{\!|[\psi_2](\gamma, l)|\!\}^{\mathcal{E}}$. Since by inductive hypothesis $\{\!|[\psi_i](\gamma, l)|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ or $\emptyset$ for all $i \in \{1, 2\}$, if at least one is $\emptyset$, then also $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \emptyset$, otherwise $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$. So, we can only have $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ or $\emptyset$.

- $\varphi = (x : \mathsf{a})\, \psi$ and $[\varphi](\gamma, l) = \bigvee_{z \in dom(\gamma).l(z)=\mathsf{a}} [\psi[z/x]](\gamma, l)$
  By definition of the semantics

  $$\{\!|\bigvee_{z \in dom(\gamma).l(z)=\mathsf{a}} [\psi[z/x]](\gamma, l)|\!\}^{\mathcal{E}} = \bigcup_{z \in dom(\gamma).l(z)=\mathsf{a}} \{\!|[\psi[z/x]](\gamma, l)|\!\}^{\mathcal{E}}.$$

  Let $V = \{z \in dom(\gamma) \mid l(z) = \mathsf{a}\}$, then we have two cases depending on V. If $V = \emptyset$, then clearly $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \emptyset$. Otherwise, by inductive hypothesis $\{\!|[\psi[z/x]](\gamma, l)|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ or $\emptyset$ for all $z \in V$. Hence, if there is $z \in V$ such that $\{\!|[\psi[z/x]](\gamma, l)|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$, then $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$, otherwise $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \emptyset$. So, in any case, we have $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ or $\emptyset$.

- $\varphi = \langle\!\langle x \rangle\!\rangle\, \psi$ and we have two cases for $[\varphi](\gamma, l)$
  If $fv(\psi) \subseteq \{x\}^c$ and for all $z \in dom(\gamma)$, $x \notin \gamma(z)$, then $[\varphi](\gamma, l) = [\psi](\gamma|_{\{x\}^c}, l|_{\{x\}^c})$. Otherwise, $[\varphi](\gamma, l) = \mathsf{F}$. Since, in the first case, by inductive hypothesis we have $\{\!|[\psi](\gamma|_{\{x\}^c}, l|_{\{x\}^c})|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ or $\emptyset$, and, in the second case, by definition of the semantics $\{\!|\mathsf{F}|\!\}^{\mathcal{E}} = \emptyset$, then we can immediately conclude that in any case $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ or $\emptyset$.

□

Using the above result for reverse-only formulae, when the functions γ and l satisfy some suitable properties, we can prove that the semantics of an open formula of $\mathrm{EIL_h}$ and

its encoding are the same by induction on the shape of the formula. Thus, we obtain the following.

Lemma 8.2 (encoding open $\mathrm{EIL_h}$)**.** *Let $\mathcal{E}$ be a PES, let φ be a formula of EIL_h, let $(C,\rho) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$, and let γ, l be s.t. $fv(\varphi) \subseteq dom(\gamma) = dom(l)$, $\rho(dom(\gamma)) = C$, and for all $x \in dom(\gamma)$, $\rho(\gamma(x)) = \lceil \rho(x) \rceil \setminus \{\rho(x)\}$ and $l(x) = \lambda(\rho(x))$. Then, (C,ρ) satisfies φ iff it satisfies $[\varphi](\gamma, l)$.*

Proof. The proof proceeds by induction on the formula φ, we show only some significant cases:

- $\varphi = \psi_1 \wedge \psi_2$ and $[\varphi](\gamma,l) = [\psi_1](\gamma,l) \wedge [\psi_2](\gamma,l)$
  By definition of the semantics (C,ρ) satisfies $\psi_1 \wedge \psi_2$ iff it satisfies both ψ_1 and ψ_2, which by inductive hypothesis happens iff (C,ρ) satisfies both $[\psi_1](\gamma,l)$ and $[\psi_2](\gamma,l)$, which in turn by definition of the semantics holds iff it satisfies $[\psi_1](\gamma,l) \wedge [\psi_2](\gamma,l)$. So we conclude that (C,ρ) satisfies $\psi_1 \wedge \psi_2$ iff it satisfies $[\psi_1](\gamma,l) \wedge [\psi_2](\gamma,l)$.

- $\varphi = \langle x : \mathsf{a} \rangle\!\rangle\, \psi$ and $[\varphi](\gamma,l) = \bigvee_{\boldsymbol{y} \subseteq dom(\gamma)} \langle\!| \boldsymbol{y}, \overline{\boldsymbol{y}^c} < \mathsf{a}\, x |\!\rangle\, [\psi](\gamma[x \mapsto \boldsymbol{y}], l[x \mapsto \mathsf{a}])$
  We show this one direction at a time. Assume that (C,ρ) satisfies $\langle x : \mathsf{a} \rangle\!\rangle\, \psi$. Then, by definition of the semantics there exists an event e such that $C \xrightarrow{e} C'$, $\lambda(e) = \mathsf{a}$, and $(C', \rho[x \mapsto e])$ satisfies ψ. Since $C \xrightarrow{e} C'$, then we must have that $(\lceil e \rceil \setminus \{e\}) \subseteq C$. Furthermore, since by hypothesis $\rho(dom(\gamma)) = C$, then there exists $\boldsymbol{y} \subseteq dom(\gamma)$ such that $\rho(\boldsymbol{y}) = \lceil e \rceil \setminus \{e\}$, hence $C \xrightarrow{\rho(\boldsymbol{y}), \overline{\rho(\boldsymbol{y}^c)} < e}_{\mathsf{a}} C'$.

  Observe that $C' = C \cup \{e\}$, $fv(\psi) \subseteq fv(\varphi) \cup \{x\} \subseteq dom(\gamma) \cup \{x\} = dom(\gamma[x \mapsto \boldsymbol{y}]) = dom(l[x \mapsto \mathsf{a}])$, $\rho[x \mapsto e](x) = e$, $\rho(\boldsymbol{y}) = \lceil e \rceil \setminus \{e\}$, and $\lambda(e) = \mathsf{a}$, so all the conditions of the hypothesis still hold on ψ, $(C', \rho[x \mapsto e])$, $\gamma[x \mapsto \boldsymbol{y}]$, and $l[x \mapsto \mathsf{a}]$. Thus we can use the inductive hypothesis to obtain that $(C', \rho[x \mapsto e])$ satisfies $[\psi](\gamma[x \mapsto \boldsymbol{y}], l[x \mapsto \mathsf{a}])$. Then, since $C \xrightarrow{\rho(\boldsymbol{y}), \overline{\rho(\boldsymbol{y}^c)} < e}_{\mathsf{a}} C'$, by definition of the semantics we have that (C,ρ) satisfies $\langle\!| \boldsymbol{y}, \overline{\boldsymbol{y}^c} < \mathsf{a}\, x |\!\rangle\, [\psi](\gamma[x \mapsto \boldsymbol{y}], l[x \mapsto \mathsf{a}])$. From this and the fact that $\boldsymbol{y} \subseteq dom(\gamma)$, by definition of the semantics of the disjunction we can conclude that (C,ρ) satisfies $\bigvee_{\boldsymbol{y} \subseteq dom(\gamma)} \langle\!| \boldsymbol{y}, \overline{\boldsymbol{y}^c} < \mathsf{a}\, x |\!\rangle\, [\psi](\gamma[x \mapsto \boldsymbol{y}], l[x \mapsto \mathsf{a}])$.
  The other direction can be proved by reversing all the implications.

- $\varphi = (x : \mathsf{a})\, \psi$ and $[\varphi](\gamma,l) = \bigvee_{z \in dom(\gamma).\, l(z) = \mathsf{a}} [\psi[{}^{z}\!/_{x}]](\gamma,l)$
  We show this one direction at a time. Assume that (C,ρ) satisfies $(x : \mathsf{a})\, \psi$. Then, by definition of the semantics there exists $e \in C$ such that $\lambda(e) = \mathsf{a}$ and $(C, \rho[x \mapsto e])$ satisfies ψ. Since by hypothesis $\rho(dom(\gamma)) = C$ and for all $x \in dom(\gamma)$, $l(x) = \lambda(\rho(x))$, then there exists $z \in dom(\gamma)$ such that $\rho(z) = e$, $l(z) = \mathsf{a}$. Since $(C, \rho[x \mapsto e])$ satisfies ψ and $\rho[x \mapsto e](x) = \rho(z) = e$, we may substitute z to x and have that (C,ρ) satisfies $\psi[{}^{z}\!/_{x}]$. Since $fv(\psi[{}^{z}\!/_{x}]) \subseteq fv(\varphi) \cup \{z\}$ and $z \in dom(\gamma)$, then the conditions of the hypothesis still hold, and so we can use the inductive hypothesis

to obtain that (C, ρ) satisfies $[\psi[^{z}\!/_{x}]](\gamma, l)$. From this and the fact that $l(z) = \mathsf{a}$, by definition of the semantics of the disjunction we can conclude that (C, ρ) satisfies $\bigvee_{z \in dom(\gamma).l(z)=\mathsf{a}} [\psi[^{z}\!/_{x}]](\gamma, l)$.

The other direction can be proved by reversing all the implications.

- $\varphi = \langle\!\langle x \rangle\!\rangle\, \psi$ and we have two cases for $[\varphi](\gamma, l)$

  1. If $fv(\psi) \subseteq \{x\}^c$ and for all $z \in dom(\gamma)$ it holds $x \notin \gamma(z)$, then $[\varphi](\gamma, l) = [\psi](\gamma|_{\{x\}^c}, l|_{\{x\}^c})$. Since by hypothesis $\rho(dom(\gamma)) = C$ and for all $z \in dom(\gamma)$, $\rho(\gamma(z)) = \lceil \rho(z) \rceil \setminus \{\rho(z)\}$, and we know that $x \notin \gamma(z)$, then $\rho(x)$ is maximal in C w.r.t. to causality. Hence $\exists\ C' = C \setminus \{\rho(x)\} \in \mathcal{C}(\mathcal{E})$ such that $C' \xrightarrow{\rho(x)} C$. Furthermore, we know that $\rho(fv(\psi)) \subseteq C'$, since $fv(\psi) \subseteq \{x\}^c$ and $C' = C \setminus \{\rho(x)\} = \rho(\{x\}^c)$. Thus, by definition of the semantics we have that (C, ρ) satisfies $\langle\!\langle x \rangle\!\rangle\, \psi$ iff (C', ρ) satisfies ψ.

     Observing also that $dom(\gamma|_{\{x\}^c}) = dom(l|_{\{x\}^c}) = \{x\}^c$, clearly all the conditions of the hypothesis still hold on ψ, (C', ρ), $\gamma|_{\{x\}^c}$, and $l|_{\{x\}^c}$. Then by inductive hypothesis we have that (C', ρ) satisfies ψ iff it satisfies $[\psi](\gamma|_{\{x\}^c}, l|_{\{x\}^c})$. From this and the previous fact we obtain that (C, ρ) satisfies φ iff (C', ρ) satisfies $[\varphi](\gamma, l)$. Now, recalling that by definition of the syntax φ is guaranteed to be a reverse-only formula, by Lemma 8.1 we know that either $\{\!|[\varphi](\gamma, l)|\!\}^{\mathcal{E}} = \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ or $\emptyset$. This implies that (C', ρ) satisfies $[\varphi](\gamma, l)$ iff (C, ρ) satisfies it. And so we can conclude that (C, ρ) satisfies φ iff it satisfies $[\varphi](\gamma, l)$.

  2. Otherwise, $[\varphi](\gamma, l) = \mathsf{F}$. Clearly (C, ρ) never satisfies F. Then we need to prove that (C, ρ) also never satisfies $\langle\!\langle x \rangle\!\rangle\, \psi$. We split this case in three more cases:

     (a) if $fv(\psi) \not\subseteq \{x\}^c$, since $fv(\psi) \subseteq fv(\varphi)$ and by hypothesis $fv(\varphi) \subseteq dom(\gamma)$, then we must have that $x \in fv(\psi)$. Then we have two cases, either $\rho(x)$ is maximal in C w.r.t. to causality or not. In the first case, we have that $\exists\ C' = C \setminus \{\rho(x)\} \in \mathcal{C}(\mathcal{E})$ such that $C' \xrightarrow{\rho(x)} C$ but $\rho(fv(\psi)) \not\subseteq C'$ since $\rho(x) \notin C'$. In the second, such a C' does not exists. So, in both cases, by definition of the semantics (C, ρ) does not satisfy $\langle\!\langle x \rangle\!\rangle\, \psi$

     (b) if $fv(\psi) \subseteq \{x\}^c$ but there exists $z \in dom(\gamma)$ such that $x \in \gamma(z)$, since by hypothesis $\rho(\gamma(z)) = \lceil \rho(z) \rceil \setminus \{\rho(z)\}$ and $\rho(x), \rho(z) \in C$, then $\rho(x)$ causes $\rho(z)$, hence $\rho(x)$ is not maximal in C w.r.t. to causality. Thus, $\nexists\ C' \in \mathcal{C}(\mathcal{E})$ such that $C' \xrightarrow{\rho(x)} C$. So, by definition of the semantics (C, ρ) does not satisfy $\langle\!\langle x \rangle\!\rangle\, \psi$.

□

Now we can easily deduce the correctness of the encoding for closed formulae of $\mathrm{EIL_h}$, observing that the conditions in the lemma above vacuously hold when $C = \gamma = l = \emptyset$.

Proposition 8.1 (encoding $\mathrm{EIL_h}$)**.** *Let $\mathcal{E}$ be a PES and let φ be a closed formula of EIL_h. Then, $\mathcal{E}$ satisfies φ if and only if it satisfies $[\varphi](\emptyset,\emptyset)$.*

Proof. Corollary of Lemma 8.2. □

Since we can encode any closed formula of $\mathrm{EIL_h}$ into a (closed) formula of $\mathcal{L}_{hp}$, we can also apply any of the model-checking techniques discussed in the previous chapters to such a formula. For instance, we have the following.

Theorem 8.1 (model-checking $\mathrm{EIL_h}$)**.** *Given a strongly regular PES $\mathcal{E}$ and a closed formula φ of EIL_h, the formula $[\varphi](\emptyset,\emptyset)$ has a successful tableau (defined as in Chapter 5) if and only if $\mathcal{E}$ satisfies φ.*

Proof. Corollary of Theorem 5.2 and Proposition 8.1. □

We conclude this section by showing that also formulae of $\mathcal{L}_{hp}$, without fixpoints, can be encoded in $\mathrm{EIL_h}$. Since, additionally, the encoding of $\mathrm{EIL_h}$ into $\mathcal{L}_{hp}$ does not use fixpoints, this means that $\mathrm{EIL_h}$ is actually "equivalent" to the fragment of $\mathcal{L}_{hp}$ without fixpoints.

As we did in the other encoding, we use a function $\gamma : \mathit{Var} \to 2^{\mathit{Var}}$ to associate variables with the set of past variables asserted to cause them. Moreover, variables encountered bound in modalities are recorded, in the order in which they appear, in a sequence υ. Intuitively, $\upsilon = (v_1, v_2, \ldots, v_n)$ can be thought of as a totally ordered configuration, such that if v_i causes v_j then $i < j$. Actually, υ is not needed for the encoding, but it makes the procedure much easier. As before, γ and υ are updated after every step forward, that is, after encountering a modality, and we assume that in every formula of $\mathcal{L}_{hp}$ different bound variables have different names. Then, the encoding is defined as follows:

$$[\mathsf{T}](\gamma,\upsilon) = \mathsf{T} \qquad [\mathsf{F}](\gamma,\upsilon) = \mathsf{F}$$

$$[\varphi \wedge \psi](\gamma,\upsilon) = [\varphi](\gamma,\upsilon) \wedge [\psi](\gamma,\upsilon) \qquad [\varphi \vee \psi](\gamma,\upsilon) = [\varphi](\gamma,\upsilon) \vee [\psi](\gamma,\upsilon)$$

$$[\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \varphi](\gamma,\upsilon) = \bigvee_{\boldsymbol{x}\subseteq\boldsymbol{u}\subseteq\boldsymbol{y}^c} \langle z:\mathsf{a}\rangle \Big(\bigwedge_{x\in\boldsymbol{u}} [[v_n^x]\ldots[[v_1^x]\,[[x]\,\mathsf{F} \wedge \bigwedge_{y\in\boldsymbol{u}^c} \langle\langle v_m^y\rangle\rangle \ldots \langle\langle v_1^y\rangle\rangle\, \langle\langle y\rangle\rangle\, \mathsf{T} \wedge [\varphi](\gamma[z\mapsto\boldsymbol{u}], \upsilon z)\Big)$$

$$[[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \varphi](\gamma,\upsilon) = \bigwedge_{\boldsymbol{x}\subseteq\boldsymbol{u}\subseteq\boldsymbol{y}^c} [z:\mathsf{a}]] \Big(\bigvee_{x\in\boldsymbol{u}} \langle\langle v_n^x\rangle\rangle \ldots \langle\langle v_1^x\rangle\rangle\, \langle\langle x\rangle\rangle\, \mathsf{T} \vee \bigvee_{y\in\boldsymbol{u}^c} [[v_m^y]\ldots[[v_1^y]\,[[y]\,\mathsf{F} \vee [\varphi](\gamma[z\mapsto\boldsymbol{u}], \upsilon z)\Big)$$

where $V^c = \upsilon \setminus V$ for a set of variables V, empty conjunctions are always true and empty disjunctions are always false, and for a variable x, the sequence $(v_1^x, \ldots, v_n^x)$ is the largest subsequence of υ such that $x \in \gamma(v_i^x)$ for all $i \in [1, n]$.

Then, the encoding of a closed formula φ of $\mathcal{L}_{hp}$ without fixpoints is obtained by computing the formula $[\varphi](\emptyset, \epsilon)$. Observe that the encoding of such a formula φ is always a formula

of $\mathrm{EIL_h}$. In fact, since backward modalities are only used properly separated from forward modalities, it is immediate to see that the resulting formulae comply with the syntax of $\mathrm{EIL_h}$.

In order to prove the correctness of the encoding, first, we show that the semantics of an open formula without fixpoints of $\mathcal{L}_{hp}$ and its encoding are the same by induction on the shape of the formula. Thus, we obtain the following.

Lemma 8.3 (encoding open $\mathcal{L}_{hp}$)**.** *Let $\mathcal{E}$ be a PES, let φ be a formula of $\mathcal{L}_{hp}$ without fixpoints or propositions, let $(C, \rho) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$, and let γ, υ be s.t. $|\upsilon| = |dom(\gamma)| = |C|$, $fv(\varphi) \subseteq \upsilon = dom(\gamma)$, $\rho(\upsilon) = C$, and for all $x, y \in \upsilon$, $\rho(\gamma(x)) = \lceil \rho(x) \rceil \setminus \{\rho(x)\}$ and if $x \in \gamma(y)$ then x comes before y in υ. Then, (C, ρ) satisfies φ iff it satisfies $[\varphi](\gamma, \upsilon)$.*

Proof. The proof proceeds by induction on the formula φ, we show only some significant cases:

- $\varphi = \psi_1 \wedge \psi_2$ and $[\varphi](\gamma, \upsilon) = [\psi_1](\gamma, \upsilon) \wedge [\psi_2](\gamma, \upsilon)$
  By definition of the semantics (C, ρ) satisfies $\psi_1 \wedge \psi_2$ iff it satisfies both ψ_1 and ψ_2. Since γ and υ are unchanged, all conditions on them are still preserved, thus by inductive hypothesis we have that (C, ρ) satisfies ψ_1 and ψ_2 iff it satisfies both $[\psi_1](\gamma, \upsilon)$ and $[\psi_2](\gamma, \upsilon)$, which in turn by definition of the semantics holds iff it satisfies $[\psi_1](\gamma, \upsilon) \wedge [\psi_2](\gamma, \upsilon)$. So we conclude that (C, ρ) satisfies $\psi_1 \wedge \psi_2$ iff it satisfies $[\psi_1](\gamma, \upsilon) \wedge [\psi_2](\gamma, \upsilon)$.

- $\varphi = \langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \psi$ and $[\varphi](\gamma, \upsilon) =$

$$\bigvee_{\boldsymbol{x} \subseteq \boldsymbol{u} \subseteq \boldsymbol{y}^c} \langle z : \mathsf{a} \rangle \big(\bigwedge_{x \in \boldsymbol{u}} [[v_n^x] \dots [[v_1^x]\, [[x]\, \mathsf{F} \wedge \bigwedge_{y \in \boldsymbol{u}^c} \langle\!\langle v_m^y \rangle\!\rangle \dots \langle\!\langle v_1^y \rangle\!\rangle\, \langle\!\langle y \rangle\!\rangle\, \mathsf{T} \wedge [\psi](\gamma[z \mapsto \boldsymbol{u}], \upsilon z) \big)$$

  We show one direction at a time. Assume that (C, ρ) satisfies $\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \psi$. Then, by definition of the semantics there exists an event $e \in \mathcal{E}[C]$ such that $C \xrightarrow{\rho(\boldsymbol{x}), \overline{\rho(\boldsymbol{y})} < e}_{\mathsf{a}} C'$ and $(C', \rho[z \mapsto e])$ satisfies ψ. Since $C \xrightarrow{\rho(\boldsymbol{x}), \overline{\rho(\boldsymbol{y})} < e}_{\mathsf{a}} C'$, then we must have that $\rho(\boldsymbol{x}) \subseteq (\lceil e \rceil \setminus \{e\}) \subseteq (C \setminus \rho(\boldsymbol{y}))$. Furthermore, since by hypothesis $fv(\varphi) \subseteq \upsilon$ and $\rho(\upsilon) = C$, then there exists $\boldsymbol{u}$ such that $\boldsymbol{x} \subseteq \boldsymbol{u} \subseteq \boldsymbol{y}^c$ and $\rho(\boldsymbol{u}) = \lceil e \rceil \setminus \{e\}$, hence $C \xrightarrow{\rho(\boldsymbol{u}), \overline{\rho(\boldsymbol{u}^c)} < e}_{\mathsf{a}} C'$.

  Let $\theta_1 = \bigwedge_{x \in \boldsymbol{u}} [[v_n^x] \dots [[v_1^x]\, [[x]\, \mathsf{F}$ and $\theta_2 = \bigwedge_{y \in \boldsymbol{u}^c} \langle\!\langle v_m^y \rangle\!\rangle \dots \langle\!\langle v_1^y \rangle\!\rangle\, \langle\!\langle y \rangle\!\rangle\, \mathsf{T}$. We proceed by separately proving that $(C', \rho[z \mapsto e])$ satisfies θ_1, θ_2 and $[\psi](\gamma[z \mapsto \boldsymbol{u}], \upsilon z)$.

  - We show that for all $x \in \boldsymbol{u}$, $(C', \rho[z \mapsto e])$ satisfies $[[v_n^x] \dots [[v_1^x]\, [[x]\, \mathsf{F}$. Assume that there exists a sequence of transitions $C_0 \xrightarrow{\rho(v_1^x)} C_1 \xrightarrow{\rho(v_2^x)} \dots \xrightarrow{\rho(v_n^x)} C_n$ such that $C_n = C'$. Note that if such a sequence of transitions would not exist, then by definition of the semantics of the backward box modality we would immediately have that $(C', \rho[z \mapsto e])$ vacuously satisfies $[[v_n^x] \dots [[v_1^x]\, [[x]\, \mathsf{F}$. Recall that $(v_1^x, \dots, v_n^x)$ is a subsequence of υ. Since $z \notin \upsilon$, we must have that $\rho[z \mapsto$

$e](z) = e \in C_0 = C' \setminus \{v_1^x, \ldots, v_n^x\}$. Recalling also that $\rho(x) < e$ because $x \in \boldsymbol{u}$, then the formula $[\![x]\!]\,\mathsf{F}$ must be vacuously satisfied by $(C_0, \rho[z \mapsto e])$. And so, by definition of the semantics we have that $(C', \rho[z \mapsto e])$ satisfies $[\![v_n^x]\!] \ldots [\![v_1^x]\!]\,[\![x]\!]\,\mathsf{F}$.

– We show that for all $y \in \boldsymbol{u}^c$, $(C', \rho[z \mapsto e])$ satisfies $\langle\!\langle v_m^y \rangle\!\rangle \ldots \langle\!\langle v_1^y \rangle\!\rangle\,\langle\!\langle y \rangle\!\rangle\,\mathsf{T}$, where $(v_1^y, \ldots, v_m^y)$ is the largest subsequence of υ such that $y \in \gamma(v_i^y)$ for all $i \in [1, m]$, which means that $\rho(y) < \rho(v_i^y)$. It also means that all variables caused by y are in the subsequence, while y itself is not. Then, for all $i \in [1, m]$, for all $w \in \upsilon$ such that $v_i^y \in \gamma(w)$, that is, $\rho(v_i^y) < \rho(w)$, by transitivity of the causality relation we have that $\rho(y) < \rho(w)$, and thus $w = v_j^y$ for some $j \in [1, m]$. Furthermore, by hypothesis we must have that such an index j comes after i. So, for all $i \in [1, m]$, every variable caused by v_i^y is another element v_j^y of the sequence such that $i < j$. In particular, v_m^y is not the cause of any other variable in υ. Since $\rho(y)$ is concurrent with $e = \rho[z \mapsto e](z)$ because $y \in \boldsymbol{u}^c$, we must have that also $\rho(v_i^y)$ is concurrent with e for all $i \in [1, m]$, otherwise by transitivity it would contradict the fact that $\rho(y) < \rho(v_i^y)$. Since by hypothesis $\rho(\upsilon) = C$ and $|\upsilon| = |C|$, from the previous facts it follows that there must be a sequence of transitions $C_0 \xrightarrow{\rho(v_1^x)} C_1 \xrightarrow{\rho(v_2^x)} \ldots \xrightarrow{\rho(v_n^x)} C_n$ such that $C_n = C' = C \cup \{e\}$ and $\rho(y) \in C_0 = C' \setminus \{v_1^y, \ldots, v_m^y\}$. Since $\rho(y)$ is concurrent with e, and all the other events caused by $\rho(y)$ where associated with variables v_i^y, then by definition of the semantics we know that $(C_0, \rho[z \mapsto e])$ satisfies the formula $\langle\!\langle y \rangle\!\rangle\,\mathsf{T}$. And so, again by the semantics, we can deduce that also $(C', \rho[z \mapsto e])$ satisfies $\langle\!\langle v_m^y \rangle\!\rangle \ldots \langle\!\langle v_1^y \rangle\!\rangle\,\langle\!\langle y \rangle\!\rangle\,\mathsf{T}$.

– Observe that $C' = C \cup \{e\}$, $|\upsilon z| = |dom(\gamma[z \mapsto \boldsymbol{u}])|$, $fv(\psi) \subseteq fv(\varphi) \cup \{z\} \subseteq \upsilon z = dom(\gamma[z \mapsto \boldsymbol{u}])$, $\rho[z \mapsto e](z) = e$, $\rho(\boldsymbol{u}) = \lceil e \rceil \setminus \{e\}$, and all variables in $\boldsymbol{u} \subseteq \upsilon$ come before z in υz, so all the conditions of the hypothesis still hold on ψ, $(C', \rho[z \mapsto e])$, $\gamma[z \mapsto \boldsymbol{u}]$ and υz. Thus we can use the inductive hypothesis to obtain that $(C', \rho[z \mapsto e])$ satisfies $[\psi](\gamma[z \mapsto \boldsymbol{u}], \upsilon z)$ since we know that $(C', \rho[z \mapsto e])$ satisfies ψ.

So, by definition of the semantics, we know that $(C', \rho[z \mapsto e])$ satisfies $\theta_1 \wedge \theta_2 \wedge [\psi](\gamma[z \mapsto \boldsymbol{u}], \upsilon z)$. Then, since $C \xrightarrow{e} C'$, by definition of the semantics we have that (C, ρ) satisfies $\langle z : \mathsf{a} \rangle\!\rangle\,(\theta_1 \wedge \theta_2 \wedge [\psi](\gamma[z \mapsto \boldsymbol{u}], \upsilon z))$. From this and the fact that $\boldsymbol{x} \subseteq \boldsymbol{u} \subseteq \boldsymbol{y}^c$, by definition of the semantics of the disjunction we can conclude that (C, ρ) satisfies $\bigvee_{\boldsymbol{x} \subseteq \boldsymbol{u} \subseteq \boldsymbol{y}^c} \langle z : \mathsf{a} \rangle\!\rangle\,\big(\bigwedge_{x \in \boldsymbol{u}} [\![v_n^x]\!] \ldots [\![v_1^x]\!]\,[\![x]\!]\,\mathsf{F} \wedge \bigwedge_{y \in \boldsymbol{u}^c} \langle\!\langle v_m^y \rangle\!\rangle \ldots \langle\!\langle v_1^y \rangle\!\rangle\,\langle\!\langle y \rangle\!\rangle\,\mathsf{T} \wedge [\psi](\gamma[z \mapsto \boldsymbol{u}], \upsilon z)\big)$.

The other direction can be proved in a similar way, reversing the implications.

□

Now we can easily deduce the correctness of the encoding for closed formulae of $\mathcal{L}_{hp}$ without fixpoints, observing that the conditions in the lemma above vacuously hold when $C = \gamma = \emptyset$ and $\upsilon = \epsilon$.

Proposition 8.2 (encoding $\mathcal{L}_{hp}$)**.** *Let $\mathcal{E}$ be a PES and let φ be a closed formula of $\mathcal{L}_{hp}$ without fixpoints. Then, $\mathcal{E}$ satisfies φ if and only if it satisfies $[\varphi](\emptyset, \epsilon)$.*

Proof. Corollary of Lemma 8.3. □

8.2 Separation and trace fixpoint logics

In [Gut11] two true concurrent logics have been proposed, namely *separation fixpoint logic* (SFL) and *trace fixpoint logic* (denoted $\mathbb{L}_\mu$). SFL was first introduced in [Gut09] and further explored in [GB09, GB11], while $\mathbb{L}_\mu$ was also studied more recently in [Gut15]. Both logics are based on a core logic in mu-calculus style with modalities that allow to express properties about causality and concurrency of actions. The main difference, with respect to the modalities of $\mathcal{L}_{hp}$, is that those of SFL and $\mathbb{L}_\mu$ can only express causality and concurrency between consecutive steps of computations. For this reason the logic does not need to resort to event variables. Moreover, for each kind of modality two operators are offered, one for concurrent steps and one for causally dependent steps (w.r.t. the previous step). For instance, the diamond modality $\langle \mathsf{a} \rangle_{nc} \varphi$ declares the possibility to execute an a-labelled event, which is concurrent with (not caused by) the one executed before.

The logics also provide two kinds of operators, very different from those we saw so far, which capture the duality between concurrency and conflict. Both work on conflict-free sets of events, called support sets. SFL offers a *separating operator* $*$ (and its dual $\bowtie$) that behaves as a structural conjunction, allowing for local reasoning on conflict-free sets of executable events. The formula $\varphi * \psi$ requires the existence of two concurrent disjunct subsets of the enabled events, such that the subformula φ holds on the first subset and ψ on the second. $\mathbb{L}_\mu$ instead has a second-order modality $\langle \otimes \rangle$ (and its dual $[\otimes]$) that recognises maximal concurrent subsets of the executable events. The formula $\langle \otimes \rangle \varphi$ requires that the subformula φ holds when restricting, locally, the computation of the system to a set of events that can actually execute all concurrently.

As mentioned, both logics include fixpoint operators in mu-calculus style. Therefore, they resort to propositions, taken from a denumerable set $\mathcal{X}$, exactly like the mu-calculus. We first introduce the syntax of SFL.

Definition 8.4 (syntax of SFL)**.** *The syntax of SFL over the sets of proposition $\mathcal{X}$ and labels Λ is defined as follows:*

$$\begin{aligned} \varphi \ ::= \ & \mathsf{T} \mid \varphi \wedge \varphi \mid \langle \mathsf{a} \rangle_c \varphi \mid \langle \mathsf{a} \rangle_{nc} \varphi \mid \varphi * \varphi \mid \mu Z.\varphi \mid Z \mid \\ & \mathsf{F} \mid \varphi \vee \varphi \mid [\mathsf{a}]_c \varphi \mid [\mathsf{a}]_{nc} \varphi \mid \varphi \bowtie \varphi \mid \nu Z.\varphi \end{aligned}$$

Then, the syntax of $\mathbb{L}_\mu$ is almost identical, except for the second-order modalities.

Definition 8.5 (syntax of $\mathbb{L}_\mu$)**.** *The syntax of $\mathbb{L}_\mu$ over the sets of proposition $\mathcal{X}$ and labels Λ is defined as follows:*

$$\begin{aligned}\varphi \ ::= \ & \mathsf{T} \mid \varphi \wedge \varphi \mid \langle \mathsf{a} \rangle_c \, \varphi \mid \langle \mathsf{a} \rangle_{nc} \, \varphi \mid \langle \otimes \rangle \, \varphi \mid \mu Z.\varphi \mid Z \mid \\ & \mathsf{F} \mid \varphi \vee \varphi \mid [\mathsf{a}]_c \, \varphi \mid [\mathsf{a}]_{nc} \, \varphi \mid [\otimes] \, \varphi \mid \nu Z.\varphi\end{aligned}$$

In order to define the semantics of logics SFL and $\mathbb{L}_\mu$ we need some notions, taken from [Gut11], about conflict-free sets of enabled events, providing specific kinds of second-order quantification over them. The most general is the concept of support set.

Definition 8.6 (support set)**.** *Let $\mathcal{E}$ be a PES. Given a configuration $C \in \mathcal{C}(\mathcal{E})$, a* support set *$R$ for C is either the set of all enabled events $en(C)$ or a non-empty consistent subset of enabled events, either way $R \subseteq en(C)$. We call $\mathcal{R}(C)$ the set of all support sets for C, and $\mathcal{R}_\mathcal{E} = \bigcup_{C \in \mathcal{C}(\mathcal{E})} \mathcal{R}(C)$ the set of all support sets for all possible configurations of $\mathcal{E}$.*

Support sets are the basic element that the logics use for local reasoning on executable events. According to the definition they can be (non-empty) consistent sets (Definition 3.3), where local reasoning becomes possible since they can be decomposed into smaller ones with the same property. Alternatively a support set can contain conflicts (or be empty) when it is the whole set of enabled events. In the latter case proper maximal conflict-free subsets can be isolated using so-called complete supsets.

Definition 8.7 (complete supset)**.** *Let $\mathcal{E}$ be a PES and $C \in \mathcal{C}(\mathcal{E})$ be a configuration. Given a support set $R \in \mathcal{R}(C)$, a* complete supset *M of R is a consistent support set $M \in \mathcal{R}(C)$ such that $M \subseteq R$ and for all $e \in R \smallsetminus M$ there exists $e' \in M$ s.t. $e \# e'$. We call $\mathcal{M}(R)$ the set of all complete supsets of R.*

Observe that by definition of complete supset there is always at least one complete supset for every support set R. More precisely, if R is consistent, then the only complete supset is R itself. Otherwise, there are many complete supset, one for each maximal consistent subset of R. Even when $R = \emptyset$, hence it is consistent, we have $\mathcal{M}(R) = \{\emptyset\}$.

Intuitively, to decompose conflict-free sets into smaller ones means to separate different parallel components of the system, to allow local reasoning on them. This decomposition is captured by the notion of separation.

Definition 8.8 (separation)**.** *Let $\mathcal{E}$ be a PES and $C \in \mathcal{C}(\mathcal{E})$ be a configuration. Given a support set $R \in \mathcal{R}(C)$, a* separation *(R_1, R_2) of R is a pair of support sets $R_1, R_2 \in \mathcal{R}(C)$ such that $R_1 \cap R_2 = \emptyset$ and $R_1 \cup R_2 \in \mathcal{M}(R)$. We call $\mathsf{Sep}(R)$ the set of all possible separations of R.*

For instance, consider $\mathcal{E}_3$ in Figure 3.1c. For the initial configuration $\emptyset$ we have three possible support sets $R_1 = \{\mathsf{b}^0, \mathsf{c}\}$, $R_2 = \{\mathsf{b}^0\}$ and $R_3 = \{\mathsf{c}\}$. Since support sets cannot be empty unless the set of enabled events is itself empty, then among those only R_1 admits a separation (two when considering both possible ordering), i.e., $\mathsf{Sep}(R_1) =$

$\{(R_2, R_3), (R_3, R_2)\}$. Moreover, the set of enabled events at configuration $\{\mathsf{c}\}$ of the same PES, i.e., $en(\{\mathsf{c}\}) = \{\mathsf{a}^0, \mathsf{b}^0\}$, does not admit a separation, because the two events are in conflict and separations can only split (maximal) consistent sets of enabled events.

Both logics operate over support sets in general, and, in particular, the separating operators of SFL reason on separations of support sets, while the second-order modalities of $\mathbb{L}_\mu$ work over complete supsets of executable events. Therefore, the satisfaction of a formula is defined with respect to a configuration C, representing the state of the computation, and a support set R for C, constraining the next step of the execution. Furthermore, since the standard modalities predicate over the causality and concurrency between the last and the next step in computations, the satisfaction of a formula depends also on the last executed event e in C. When the state is the initial empty configuration we use a placeholder event, which is assumed to cause every other event.

As for the other fixpoint logics, the denotation of an open formula depends on the evaluation of its free propositions. The semantic interpretation of propositions is given by a proposition environment $\pi : \mathcal{X} \to \mathbf{2}^{\mathcal{C}(\mathcal{E}) \times \mathcal{R}_\mathcal{E} \times E}$ from the set $PEnv_\mathcal{E}$.

Definition 8.9 (semantics of SFL)**.** *Let $\mathcal{E}$ be a PES. The denotation of a formula in SFL is given by the function $\{\!|\cdot|\!\}^\mathcal{E} : SFL \to PEnv_\mathcal{E} \to \mathbf{2}^{\mathcal{C}(\mathcal{E}) \times \mathcal{R}_\mathcal{E} \times E}$ defined inductively as follows, where we write $\{\!|\varphi|\!\}^\mathcal{E}_\pi$ instead of $\{\!|\varphi|\!\}^\mathcal{E}(\pi)$:*

$$
\begin{aligned}
\{\!|\mathsf{T}|\!\}^\mathcal{E}_\pi &= \mathcal{C}(\mathcal{E}) \times \mathcal{R}_\mathcal{E} \times E \\
\{\!|\mathsf{F}|\!\}^\mathcal{E}_\pi &= \emptyset \\
\{\!|Z|\!\}^\mathcal{E}_\pi &= \pi(Z) \\
\{\!|\varphi_1 \wedge \varphi_2|\!\}^\mathcal{E}_\pi &= \{\!|\varphi_1|\!\}^\mathcal{E}_\pi \cap \{\!|\varphi_2|\!\}^\mathcal{E}_\pi \\
\{\!|\varphi_1 \vee \varphi_2|\!\}^\mathcal{E}_\pi &= \{\!|\varphi_1|\!\}^\mathcal{E}_\pi \cup \{\!|\varphi_2|\!\}^\mathcal{E}_\pi \\
\{\!|\langle \mathsf{a} \rangle_c \, \varphi|\!\}^\mathcal{E}_\pi &= \{(C, R, e) \mid \exists\, e' \in R.\ C \xrightarrow{e < e'}_\mathsf{a} C' \ \wedge\ (C', en(C'), e') \in \{\!|\varphi|\!\}^\mathcal{E}_\pi\} \\
\{\!|\langle \mathsf{a} \rangle_{nc} \, \varphi|\!\}^\mathcal{E}_\pi &= \{(C, R, e) \mid \exists\, e' \in R.\ C \xrightarrow{\overline{e} < e'}_\mathsf{a} C' \ \wedge\ (C', en(C'), e') \in \{\!|\varphi|\!\}^\mathcal{E}_\pi\} \\
\{\!|[\mathsf{a}]_c \, \varphi|\!\}^\mathcal{E}_\pi &= \{(C, R, e) \mid \forall\, e' \in R.\ C \xrightarrow{e < e'}_\mathsf{a} C' \ \Rightarrow\ (C', en(C'), e') \in \{\!|\varphi|\!\}^\mathcal{E}_\pi\} \\
\{\!|[\mathsf{a}]_{nc} \, \varphi|\!\}^\mathcal{E}_\pi &= \{(C, R, e) \mid \forall\, e' \in R.\ C \xrightarrow{\overline{e} < e'}_\mathsf{a} C' \ \Rightarrow\ (C', en(C'), e') \in \{\!|\varphi|\!\}^\mathcal{E}_\pi\} \\
\{\!|\varphi_1 * \varphi_2|\!\}^\mathcal{E}_\pi &= \{(C, R, e) \mid \exists\, (R_1, R_2) \in \mathsf{Sep}(R).\ \textstyle\bigwedge_{i \in \{1,2\}} (C, R_i, e) \in \{\!|\varphi_i|\!\}^\mathcal{E}_\pi\} \\
\{\!|\varphi_1 \bowtie \varphi_2|\!\}^\mathcal{E}_\pi &= \{(C, R, e) \mid \forall\, (R_1, R_2) \in \mathsf{Sep}(R).\ \textstyle\bigvee_{i \in \{1,2\}} (C, R_i, e) \in \{\!|\varphi_i|\!\}^\mathcal{E}_\pi\} \\
\{\!|\eta Z.\varphi|\!\}^\mathcal{E}_\pi &= \eta(f_{\varphi, Z, \pi})
\end{aligned}
$$

where $f_{\varphi,Z,\pi} : \mathbf{2}^{\mathcal{C}(\mathcal{E}) \times \mathcal{R}_\mathcal{E} \times E} \to \mathbf{2}^{\mathcal{C}(\mathcal{E}) \times \mathcal{R}_\mathcal{E} \times E}$ is the semantic function of φ, Z, π defined by $f_{\varphi,Z,\pi}(S) = \{\!|\varphi|\!\}^\mathcal{E}_{\pi[Z \mapsto S]}$. When $(C, R, e) \in \{\!|\varphi|\!\}^\mathcal{E}_\pi$ we say that the PES $\mathcal{E}$ satisfies the formula φ in the configuration C, support set R, last executed event e, and environment π. When φ is closed, hence the environment π is irrelevant, and $\mathcal{E}$ satisfies the formula φ in the empty configuration, with the initial support set $en(\emptyset)$ and the initial placeholder event, we simply

say that $\mathcal{E}$ satisfies φ.

The diamond modality $\langle \mathsf{a} \rangle_c \varphi$ holds in (C, R, e) when there is an a-labelled event e' in the support set R, hence enabled, which is caused by the last executed event e, and, once executed obtaining the configuration $C' = C \cup \{e'\}$, the triple $(C', en(C'), e')$ satisfies the formula φ. Similarly, the other diamond modality $\langle \mathsf{a} \rangle_{nc} \varphi$ holds in (C, R, e) when the same happens with an event e' which is concurrent with e. Note that after such a step the support set always becomes the whole set of (newly) enabled events. This means that the reasoning on executable events is always local, relevant only in the next step of the computation.

The separating operator $\varphi_1 * \varphi_2$ holds in (C, R, e) if there exists a separation (R_1, R_2) of R such that the two formulae φ_1 and φ_2 still hold after restricting the support set to R_1 and R_2, respectively, i.e., (C, R_i, e) satisfies φ_i for $i \in \{1, 2\}$. Dually, $\varphi_1 \bowtie \varphi_2$ holds in (C, R, e) when for every separation (R_1, R_2) of R at least one formula φ_i, for $i \in \{1, 2\}$, holds after restricting the next step to the support set R_i.

The semantics of $\mathbb{L}_\mu$ is defined in the same way. Thus, we just need to specify that of its own second-order operators.

Definition 8.10 (semantics of $\mathbb{L}_\mu$)**.** *Let $\mathcal{E}$ be a PES. The denotation of a formula in $\mathbb{L}_\mu$ is given by the function $\{\!|\cdot|\!\}^{\mathcal{E}} : \mathbb{L}_\mu \to PEnv_{\mathcal{E}} \to \mathbf{2}^{\mathcal{C}(\mathcal{E}) \times \mathcal{R}_{\mathcal{E}} \times E}$ defined inductively as in Definition 8.9 (ignoring the separating operators) with the addition of:*

$$\{\!|\langle \otimes \rangle \varphi|\!\}^{\mathcal{E}}_{\pi} = \{(C, R, e) \mid \exists\, M \in \mathcal{M}(R).\ (C, M, e) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\}$$
$$\{\!|[\otimes] \varphi|\!\}^{\mathcal{E}}_{\pi} = \{(C, R, e) \mid \forall\, M \in \mathcal{M}(R).\ (C, M, e) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\}$$

The diamond second-order modality $\langle \otimes \rangle \varphi$ holds in (C, R, e) when there is a complete supset M of R that, along with C and e, satisfies the rest of the formula φ. Dually, $[\otimes] \varphi$ holds in (C, R, e) if every complete supset M of R, along with C and e, satisfies φ.

As we observed for $\mathcal{L}_{hp}$, the mu-calculus is a strict fragment of both logics SFL and $\mathbb{L}_\mu$. Indeed, ignoring the second-order modalities, the mu-calculus fragment can be obtain by requiring that the standard modalities occur always in pairs, caused and not caused by, that is, the diamond modality of mu-calculus $\langle \mathsf{a} \rangle \varphi$ corresponds to the formula $\langle \mathsf{a} \rangle_c \varphi \vee \langle \mathsf{a} \rangle_{nc} \varphi$, while the box modality $[\mathsf{a}]\varphi$ corresponds to $[\mathsf{a}]_c \varphi \wedge [\mathsf{a}]_{nc} \varphi$.

Interestingly, the two logics have incomparable expressive power, as they characterise incomparable behavioural equivalences [Gut11], in turn incomparable with hp-bisimilarity. Obviously this arise from the (incomparable) expressiveness of the different operators offered by the logics and the fact that they cannot express causality and concurrency between (multiple) actions in computations arbitrarily distant from each other. For example, consider two event structures consisting of two and three, respectively, parallel infinite causal chains of a-labelled events. They can be easily told apart by SFL via the formula $\langle \mathsf{a} \rangle_c \mathsf{T} * \langle \mathsf{a} \rangle_c \mathsf{T} * \langle \mathsf{a} \rangle_c \mathsf{T}$. Instead, the two PESs are indistinguishable by $\mathbb{L}_\mu$, because, intuitively, the logic cannot recognise the size of equally labelled support sets.

To see that the two logical equivalences induced by SFL and $\mathbb{L}_\mu$ are incomparable with hp-bisimilarity, first consider the PESs in Figure 8.1. Taken from [JNW96], the example displays two PESs which are hp-bisimilar but not hhp-bisimilar. Since they are hp-bisimilar, there is no formula of $\mathcal{L}_{hp}$ telling them apart. However, they can be distinguished by both logics SFL and $\mathbb{L}_\mu$. The formula $\langle \mathsf{a} \rangle_c \langle \mathsf{c} \rangle_c \mathsf{T} * \langle \mathsf{b} \rangle_c \langle \mathsf{d} \rangle_c \mathsf{T}$ of SFL is satisfied only by the PES $\mathcal{E}_{10}$ on the right. Similarly, the formula $\langle \otimes \rangle (\langle \mathsf{a} \rangle_c \langle \mathsf{c} \rangle_c \mathsf{T} \wedge \langle \mathsf{b} \rangle_c \langle \mathsf{d} \rangle_c \mathsf{T})$ of $\mathbb{L}_\mu$ also holds only on $\mathcal{E}_{10}$. Roughly, both formulae require the existence of a maximal conflict-free set of enabled events from where it is possible to execute the two sequences of actions $\mathsf{a}\,\mathsf{c}$ and $\mathsf{b}\,\mathsf{d}$. This can never be satisfied by $\mathcal{E}_9$ since the events b^0 and a^1 are in conflict.

(a) $\mathcal{E}_9$ (b) $\mathcal{E}_{10}$

Figure 8.1: Hp-bisimilar PESs which can be distinguished by SFL and $\mathbb{L}_\mu$.

Now, examine the PESs in Figure 8.2. This time the two PESs are not hp-bisimilar. This is proved by the formula $\langle\!\langle \mathsf{a}\, x \rangle\!\rangle \langle\!\langle \mathsf{b}\, y \rangle\!\rangle \langle\!\langle x, y < \mathsf{c}\, z \rangle\!\rangle \mathsf{T}$ of $\mathcal{L}_{hp}$ requiring the executability of a c-labelled event caused by both an a and a b-labelled event. Clearly the formula is satisfied only by the PES $\mathcal{E}_{12}$ on the right since in the other there is no such a c-labelled event. On the other hand, the two PESs are logically equivalent for both SFL and $\mathbb{L}_\mu$. Intuitively, because neither logic is able to express multiple causal dependencies for the same event unless such event is uniquely identified by its label. Indeed, any formula of SFL or $\mathbb{L}_\mu$ will confuse the event c^2 in $\mathcal{E}_{12}$ with either c^0 or c^1, depending on the required causal relation with the one executed before.

(a) $\mathcal{E}_{11}$ (b) $\mathcal{E}_{12}$

Figure 8.2: Two PESs which are equivalent in SFL and $\mathbb{L}_\mu$ but not hp-bisimilar.

Moreover, both logics SFL and $\mathbb{L}_\mu$ are strictly less expressive than the whole logic in [BC14] characterising hhp-bisimilarity. This immediately arises from the fact that the corresponding logical equivalences are strictly coarser than hhp-bisimilarity [Gut11].

The model-checking problem has been proved decidable in both logics [GB09, GB11, Gut11] over regular trace event structures, which is the class of PESs corresponding exactly

to finite safe Petri nets [Thi02], hence a (possibly equivalent) subclass of strongly regular PESs.

8.3 A more powerful logic

In this section we show that the logic $\mathcal{L}_{hp}$ can be extended with the operators from SFL and $\mathbb{L}_\mu$ not expressible in $\mathcal{L}_{hp}$, thus obtaining a more powerful and general true concurrent logic, for which most results and verification techniques provided for $\mathcal{L}_{hp}$ still work.

We will first introduce the syntax and the semantics of the logic $\mathcal{L}_{hp}^{*\otimes}$ which arises as a join of the logics for true concurrency $\mathcal{L}_{hp}$, SFL and $\mathbb{L}_\mu$. The logic has formulae that predicate over executability of events in computations and their dependency relations (causality, concurrency), and provide second-order power on conflict-free sets of events in the two different flavours described in the previous section.

Then we study the model-checking problem in the new logic $\mathcal{L}_{hp}^{*\otimes}$. We prove that it is decidable over strongly regular PESs, providing a decision procedure based on the one presented in Chapter 5. These results can be seen as a witness of the fact that the framework developed in the thesis is flexible and easy to extend.

8.3.1 Syntax

As already mentioned, formulae of $\mathcal{L}_{hp}$ include event variables, and so do formulae of $\mathcal{L}_{hp}^{*\otimes}$. They belong to a fixed denumerable set Var. The logic, in positive form, includes the diamond and box modalities from $\mathcal{L}_{hp}$, the separating operators from SFL and the second-order modalities from $\mathbb{L}_\mu$, described before.

Like the three logics we combine, $\mathcal{L}_{hp}^{*\otimes}$ is also a fixpoint logic in mu-calculus style. Fixpoint operators resort to propositional variables, expressed by abstract propositions to let them interact correctly with event variables. Their definitions follow directly those for $\mathcal{L}_{hp}$ given in Section 4.3. Abstract propositions belong to a fixed denumerable set $\mathcal{X}^a$, ranged over by $X, Y, \ldots$. Each abstract proposition X has an arity $ar(X)$ and represents a formula with $ar(X)$ (unnamed) free event variables. For $\boldsymbol{y}$ such that $|\boldsymbol{y}| = ar(X)$, $X(\boldsymbol{y})$ indicates the abstract proposition X whose free event variables are named $\boldsymbol{y}$. We call $X(\boldsymbol{y})$ a proposition and $\mathcal{X}$ the set of all propositions.

Definition 8.11 (syntax of $\mathcal{L}_{hp}^{*\otimes}$)**.** *The syntax of $\mathcal{L}_{hp}^{*\otimes}$ over the sets of event variables Var, abstract propositions $\mathcal{X}^a$ and labels Λ is defined as follows:*

$$\begin{array}{rcl} \varphi & ::= & \mathsf{T} \mid \varphi \wedge \varphi \mid \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi \mid \varphi * \varphi \mid \langle \otimes \rangle \varphi \mid (\mu Z(\boldsymbol{x}).\varphi)(\boldsymbol{y}) \mid Z(\boldsymbol{x}) \mid \\ & & \mathsf{F} \mid \varphi \vee \varphi \mid [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!] \varphi \mid \varphi \bowtie \varphi \mid [\otimes] \varphi \mid (\nu Z(\boldsymbol{x}).\varphi)(\boldsymbol{y}) \end{array}$$

The free event variables of a formula φ are denoted $fv(\varphi)$ and defined in the usual way (Definition 4.7). The free propositions in φ are again denoted by $fp(\varphi)$ (Definition 4.8).

We recall that for formulae $(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})$ it is required that $fv(\varphi) = \boldsymbol{x}$. Substitutions of variables are also defined analogously to $\mathcal{L}_{hp}$.

The possibility to mix operators from the three different logics $\mathcal{L}_{hp}$, SFL and $\mathbb{L}_\mu$ allows to express quite interesting and profound properties. For example, the formula $\langle\otimes\rangle(\mathsf{F} \bowtie \mathsf{F})$ requires the existence of a maximal conflict-free set of enabled events which cannot be further separated. The meaning of the property can also be formulated as the existence of an enabled event which is in conflict with all the other (enabled) events, if any. This is false in the initial state of the PES $\mathcal{E}_3$ in Figure 3.1c since two concurrent events are initially enabled. As an example of property of infinite computations, consider $[\![\mathsf{b}\, x]\!]\, \nu Z(x).(\langle\!|\mathsf{c}\, y|\!\rangle \mathsf{T} * (\langle\!|x < \mathsf{b}\, z|\!\rangle \mathsf{T} \wedge [\![x < \mathsf{b}\, w]\!] Z(w)))$, expressing that all non-empty causal chains of b-labelled events reach a state where the system can be separated into two parallel components, one continuing the chain of b events, while the other can execute a c-labelled event. Such a formula is satisfied by $\mathcal{E}_3$.

8.3.2 Semantics

The satisfaction of a formula of $\mathcal{L}_{hp}^{*\otimes}$ is defined with respect to a configuration C, a support set R for C and an environment $\rho : Var \to E$ for its free variables. Namely, the semantics of a formula will be a set of triples in $\mathcal{C}(\mathcal{E}) \times \mathcal{R}_\mathcal{E} \times Env_\mathcal{E}$, where $Env_\mathcal{E}$ is the set of environments. Given $S \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{R}_\mathcal{E} \times Env_\mathcal{E}$ and two tuples of variables $\boldsymbol{x}$ and $\boldsymbol{y}$, with $|\boldsymbol{x}| = |\boldsymbol{y}|$, we define $S[\boldsymbol{y}/\!\!/\boldsymbol{x}] = \{(C, R, \rho') \mid \exists\, (C, R, \rho) \in S \wedge \rho(\boldsymbol{x}) = \rho'(\boldsymbol{y})\}$. The semantics of $\mathcal{L}_{hp}^{*\otimes}$ also depends on a proposition environment $\pi : \mathcal{X} \to \mathbf{2}^{\mathcal{C}(\mathcal{E})\times\mathcal{R}_\mathcal{E}\times Env_\mathcal{E}}$ providing an interpretation for propositions. As we required for $\mathcal{L}_{hp}$, to ensure that the semantics of a formula depends only on the events associated with its free variables and is independent on the naming of the variables, it is required that for all tuples of variables $\boldsymbol{x}$, $\boldsymbol{y}$ with $|\boldsymbol{x}| = |\boldsymbol{y}| = ar(X)$ it holds $\pi(X(\boldsymbol{y})) = \pi(X(\boldsymbol{x}))[\boldsymbol{y}/\!\!/\boldsymbol{x}]$. The set of proposition environments is $PEnv_\mathcal{E}$. Updates to proposition environments are defined as in Subsection 4.3.2.

We extend the notation $\mathsf{Succ}_\mathcal{E}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}$ introduced for $\mathcal{L}_{hp}$ in order to accommodate support sets. So, for a triple $(C, R, \rho) \in \mathcal{C}(\mathcal{E}) \times \mathcal{R}_\mathcal{E} \times Env_\mathcal{E}$ and variables $\boldsymbol{x}$, $\boldsymbol{y}$, z, we define the $(\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z)$-*successors* of (C, R, ρ), as

$$\mathsf{Succ}_\mathcal{E}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}(C, R, \rho) = \{(C', en(C'), \rho[z \mapsto e]) \;\mid\; e \in R \;\wedge\; C \xrightarrow{\rho(\boldsymbol{x}),\overline{\rho(\boldsymbol{y})} < e}_{\mathsf{a}} C'\}.$$

Observe that only events in the support set R are considered executable, and, after executing an event, the support set always becomes to the whole set of enabled events.

Then, the definition of the semantics of formulae in $\mathcal{L}_{hp}^{*\otimes}$ is as below.

Definition 8.12 (semantics of $\mathcal{L}_{hp}^{*\otimes}$)**.** *Let $\mathcal{E}$ be a PES. The denotation of a formula φ in $\mathcal{L}_{hp}^{*\otimes}$ is given by the function $\{\!|\cdot|\!\}^\mathcal{E} : \mathcal{L}_{hp}^{*\otimes} \to PEnv_\mathcal{E} \to \mathbf{2}^{\mathcal{C}(\mathcal{E})\times\mathcal{R}_\mathcal{E}\times Env_\mathcal{E}}$ defined inductively as*

follows, where we write $\{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ *instead of* $\{\!|\varphi|\!\}^{\mathcal{E}}(\pi)$*:*

$$
\begin{aligned}
\{\!|\mathsf{T}|\!\}^{\mathcal{E}}_{\pi} &= \mathcal{C}(\mathcal{E}) \times \mathcal{R}_{\mathcal{E}} \times \mathit{Env}_{\mathcal{E}} \\
\{\!|\mathsf{F}|\!\}^{\mathcal{E}}_{\pi} &= \emptyset \\
\{\!|Z(\boldsymbol{y})|\!\}^{\mathcal{E}}_{\pi} &= \pi(Z(\boldsymbol{y})) \\
\{\!|\varphi_1 \wedge \varphi_2|\!\}^{\mathcal{E}}_{\pi} &= \{\!|\varphi_1|\!\}^{\mathcal{E}}_{\pi} \cap \{\!|\varphi_2|\!\}^{\mathcal{E}}_{\pi} \\
\{\!|\varphi_1 \vee \varphi_2|\!\}^{\mathcal{E}}_{\pi} &= \{\!|\varphi_1|\!\}^{\mathcal{E}}_{\pi} \cup \{\!|\varphi_2|\!\}^{\mathcal{E}}_{\pi} \\
\{\!|\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi|\!\}^{\mathcal{E}}_{\pi} &= \{(C,R,\rho) \mid \mathsf{Succ}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}_{\mathcal{E}}(C,R,\rho) \cap \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi} \neq \emptyset\} \\
\{\!|[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!] \varphi|\!\}^{\mathcal{E}}_{\pi} &= \{(C,R,\rho) \mid \mathsf{Succ}^{\boldsymbol{x},\overline{\boldsymbol{y}}<\mathsf{a}z}_{\mathcal{E}}(C,R,\rho) \subseteq \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\} \\
\{\!|\varphi_1 * \varphi_2|\!\}^{\mathcal{E}}_{\pi} &= \{(C,R,\rho) \mid \exists\, (R_1,R_2) \in \mathsf{Sep}(R).\, \textstyle\bigwedge_{i\in\{1,2\}} (C,R_i,\rho) \in \{\!|\varphi_i|\!\}^{\mathcal{E}}_{\pi}\} \\
\{\!|\varphi_1 \bowtie \varphi_2|\!\}^{\mathcal{E}}_{\pi} &= \{(C,R,\rho) \mid \forall\, (R_1,R_2) \in \mathsf{Sep}(R).\, \textstyle\bigvee_{i\in\{1,2\}} (C,R_i,\rho) \in \{\!|\varphi_i|\!\}^{\mathcal{E}}_{\pi}\} \\
\{\!|\langle\otimes\rangle \varphi|\!\}^{\mathcal{E}}_{\pi} &= \{(C,R,\rho) \mid \exists\, M \in \mathcal{M}(R).\, (C,M,\rho) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\} \\
\{\!|[\otimes] \varphi|\!\}^{\mathcal{E}}_{\pi} &= \{(C,R,\rho) \mid \forall\, M \in \mathcal{M}(R).\, (C,M,\rho) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}\} \\
\{\!|(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})|\!\}^{\mathcal{E}}_{\pi} &= \eta(f_{\varphi,Z(\boldsymbol{x}),\pi})[\boldsymbol{y}/\boldsymbol{x}]
\end{aligned}
$$

where $f_{\varphi,Z(\boldsymbol{x}),\pi} : \mathbf{2}^{\mathcal{C}(\mathcal{E})\times\mathcal{R}_{\mathcal{E}}\times\mathit{Env}_{\mathcal{E}}} \to \mathbf{2}^{\mathcal{C}(\mathcal{E})\times\mathcal{R}_{\mathcal{E}}\times\mathit{Env}_{\mathcal{E}}}$ *is the semantic function of* φ*,* $Z(\boldsymbol{x})$*,* π *defined by* $f_{\varphi,Z(\boldsymbol{x}),\pi}(S) = \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi[Z(\boldsymbol{x})\mapsto S]}$*. When* $(C,R,\rho) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ *we say that the PES* $\mathcal{E}$ *satisfies the formula* φ *in the configuration* C*, support set* R*, and environments* ρ, π*. When* φ *is closed, hence the environments* ρ, π *are irrelevant, and* $\mathcal{E}$ *satisfies the formula* φ *in the empty configuration, with the initial support set* $en(\emptyset)$*, we simply say that* $\mathcal{E}$ *satisfies* φ*.*

The semantics of boolean connectives is as usual. The formula $\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle \varphi$ holds in (C,R,ρ) when an a-labelled event e included in the support set R (hence enabled in configuration C), that causally depends on (at least) the events bound to the variables in $\boldsymbol{x}$ and is concurrent with (at least) those bound to the variables in $\boldsymbol{y}$, can be executed producing a new configuration $C' = C \cup \{e\}$ which, along with the events enabled in C' and the environment $\rho' = \rho[z \mapsto e]$, satisfies the formula φ. Dually, $[\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!] \varphi$ holds when all a-labelled events in R, caused by $\rho(\boldsymbol{x})$ and concurrent with $\rho(\boldsymbol{y})$, bring to a configuration where φ is satisfied.

The semantics of the second-order operators are the same of those of SFL and $\mathbb{L}_{\mu}$. The formula $\varphi_1 * \varphi_2$ is satisfied by (C,R,ρ) if there is a separation (R_1,R_2) of R such that each formula φ_i holds in the corresponding R_i with the same configuration C and environment ρ. Dually, $\varphi_1 \bowtie \varphi_2$ holds if in all the possible separations of R at least one component satisfies the corresponding subformula. The operator $\langle\otimes\rangle\varphi$ is satisfied by (C,R,ρ) when the formula φ holds after restricting R to one of its complete supsets M. Dually, $[\otimes]\varphi$ requires that φ holds for all possible restrictions of R to complete supsets.

The fixpoints corresponding to the formulae $(\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})$ are guaranteed to exist by Theorem 2.1, since the set $\mathbf{2}^{\mathcal{C}(\mathcal{E})\times\mathcal{R}_{\mathcal{E}}\times\mathit{Env}_{\mathcal{E}}}$ ordered by subset inclusion is a complete lattice and the functions $f_{\varphi,Z(\boldsymbol{x}),\pi}$ are monotone.

As we did for $\mathcal{L}_{hp}$, we assume that in every formula different bound propositions have different names, so that we can refer to the fixpoint subformula quantifying an abstract proposition. This requirement can always be fulfilled by alpha-renaming. All the definitions about alternation depth and active subformulae given in Subsection 4.3.4 transfer naturally to the logic $\mathcal{L}_{hp}^{*\otimes}$. Moreover, observe that the results on substitutions in Lemma 4.1 hold in the new logic as well, and they will be later used as if they were defined over $\mathcal{L}_{hp}^{*\otimes}$.

Notably, fragments of $\mathcal{L}_{hp}^{*\otimes}$ can be easily identified which correspond to the three original logics $\mathcal{L}_{hp}$, SFL and $\mathbb{L}_\mu$. Hence $\mathcal{L}_{hp}^{*\otimes}$ is indeed more powerful than all of them. In fact, all the PESs in the examples presented in Section 8.2 are distinguishable by formulae of $\mathcal{L}_{hp}^{*\otimes}$, making its induced logical equivalence strictly finer then those induced by the three source logics. Still, the logical equivalence induced by $\mathcal{L}_{hp}^{*\otimes}$ is strictly coarser than hhp-bisimilarity. This is motivated by the fact that no formula of $\mathcal{L}_{hp}^{*\otimes}$ can tell apart the two PESs in Figure 3.6 which, as mentioned, are not hhp-bisimilar.

8.3.3 Model checking $\mathcal{L}_{hp}^{*\otimes}$

In this subsection we provide a model-checking procedure for $\mathcal{L}_{hp}^{*\otimes}$, showing that it is sound and complete over strongly regular PESs. The procedure is given in the form of a tableau system that follows closely that of Chapter 5. Adapting automata and game-theoretic methods would work analogously.

The tableau system works on sequents $C, R, \rho, \Delta \models^{\mathcal{E}} \varphi$, where φ is a formula of $\mathcal{L}_{hp}^{*\otimes}$, $C \in \mathcal{C}(\mathcal{E})$ is a configuration, R is a support set for C, ρ is an environment, and Δ is a finite set of definitions defined as in Chapter 5 with all the properties already mentioned there. Again, the tuple $\langle C, R, \rho, \Delta \rangle$ is the context of the sequent and the formula φ is the consequent.

Most definitions are the same of those in Chapter 5, like the definition of the instantiation $(\varphi)_\Delta$ of a formula φ with respect to a set of definitions Δ, or that of well-formed sequent, which again sequents will be tacitly assumed to comply with. Furthermore, the results on the interactions between instantiations, substitutions and fixpoint approximants (Lemmata 5.1 and 5.9) can be easily proved to hold also in $\mathcal{L}_{hp}^{*\otimes}$. Below we recall the formal meaning of true sequent adapted to $\mathcal{L}_{hp}^{*\otimes}$.

Definition 8.13 (truth)**.** *A well-formed sequent $C, R, \rho, \Delta \models^{\mathcal{E}} \varphi$ is* true *if $(C, R, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi^{\mathcal{E}}$, where π is any proposition environment.*

The tableau rules for the operators already appearing in $\mathcal{L}_{hp}$ are obtained by adapting those in Table 5.1, by adding support sets R in every sequent and applying some changes where necessary. We introduce new rules for the operators originating from SFL and $\mathbb{L}_\mu$. Then, the new tableau rules are reported in Table 8.1.

The rules for propositional connectives and fixpoint introduction and unfolding are exactly the same of before, with the addition of support sets in the contexts. Just note that the stop condition γ is not the same, although very similar, and will be defined later. The

$$(\wedge)\ \frac{C, R, \rho, \Delta \ \models\ \varphi \ \wedge\ \psi}{C, R, \rho, \Delta \ \models\ \varphi \qquad C, R, \rho, \Delta \ \models\ \psi}$$

$$(\vee_{\mathsf{L}})\ \frac{C, R, \rho, \Delta \ \models\ \varphi \ \vee\ \psi}{C, R, \rho, \Delta \ \models\ \varphi} \qquad (\vee_{\mathsf{R}})\ \frac{C, R, \rho, \Delta \ \models\ \varphi \ \vee\ \psi}{C, R, \rho, \Delta \ \models\ \psi}$$

$$(\Diamond)\ \frac{C, R, \rho, \Delta \ \models\ \langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \varphi}{C', en(C'), \rho[z \mapsto e], \Delta \ \models\ \varphi} \qquad e \in R \text{ and } C \xrightarrow{\rho(\boldsymbol{x}), \overline{\rho(\boldsymbol{y})} < e}_{\mathsf{a}} C'$$

$$(\Box)\ \frac{C, R, \rho, \Delta \ \models\ [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \varphi}{C_1, en(C_1), \rho_1, \Delta \ \models\ \varphi \ \ \ldots \ \ C_n, en(C_n), \rho_n, \Delta \ \models\ \varphi}$$

where $\{(C_1, en(C_1), \rho_1), \ldots, (C_n, en(C_n), \rho_n)\} = \mathsf{Succ}_{\mathcal{E}}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}(C, R, \rho)$

$$(*)\ \frac{C, R, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi * \psi}{C, R', \rho, \Delta \ \models^{\mathcal{E}}\ \varphi \qquad C, R'', \rho, \Delta \ \models^{\mathcal{E}}\ \psi} \qquad (R', R'') \in \mathsf{Sep}(R)$$

$$(\bowtie)\ \frac{C, R, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi_1 \bowtie \varphi_2}{C, R_1^{p_1}, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi_{p_1} \ \ \ldots \ C, R_n^{p_n}, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi_{p_n}}$$

where $\{(R_1^1, R_1^2), \ldots, (R_n^1, R_n^2)\} = \mathsf{Sep}(R)$ and $\forall\, i \in [1, n].\ p_i \in \{1, 2\}$

$$(\langle\otimes\rangle)\ \frac{C, R, \rho, \Delta \ \models^{\mathcal{E}}\ \langle\otimes\rangle\, \varphi}{C, M, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi} \qquad M \in \mathcal{M}(R)$$

$$([\otimes])\ \frac{C, R, \rho, \Delta \ \models^{\mathcal{E}}\ [\otimes]\, \varphi}{C, M_1, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi \ \ \ldots \ \ C, M_n, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi} \qquad \text{where } \{M_1, \ldots, M_n\} = \mathcal{M}(R)$$

$$(\mathsf{Int})\ \frac{C, R, \rho, \Delta \ \models\ (\eta Z(\boldsymbol{x}).\varphi)(\boldsymbol{y})}{C, R, \rho, \Delta' \ \models\ Z(\boldsymbol{y})} \qquad \Delta' = \Delta[Z(\boldsymbol{x}) \mapsto \eta Z(\boldsymbol{x}).\varphi]$$

$$(\mathsf{Unf}_{\eta})\ \frac{C, R, \rho, \Delta \ \models\ Z(\boldsymbol{z})}{C, R, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})], \Delta \ \models\ \varphi} \qquad \neg\gamma \text{ and } \Delta(Z(\boldsymbol{x})) = \eta Z(\boldsymbol{x}).\varphi$$

Table 8.1: The tableau rules for the logic $\mathcal{L}_{hp}^{*\otimes}$.

rules ($\Diamond$) and ($\Box$) for the standard modalities are obtained from those for $\mathcal{L}_{hp}$ by adding the requirement that the executed event must be in the support set R, which then becomes the set of enabled events in the reached configuration.

Rule ($*$) reduces the truth of a separating formula to the truth of the two subformulae

with their respective support sets restricted to a pair forming a separation of the one in the premise. Dually, rule ($\bowtie$) reduces the truth of the premise to that of a list of sequents produced by pairing one of the components of each possible separation of the initial support set with the corresponding subformula, as required by the semantics of the separating operator.

Rule ($\langle\otimes\rangle$) reduces the truth of the second-order modal formula $\langle\otimes\rangle\varphi$ to the truth of the subformula φ after restricting the support set to one of its complete supset. Dually, rule ($[\otimes]$) for the formula $[\otimes]\varphi$ requires that the subformula φ holds in all the possible complete supsets of the initial support set.

Observe that, working with boundedly branching PESs, like strongly regular PESs, not only rule ($\Box$) has always a finite number of sequents in the conclusion, but the same holds for rules ($[\otimes]$) and ($\bowtie$), since every support set is a subset of the enabled events, which are bounded, therefore the sets of possible separations and complete supsets are also finite and bounded. Furthermore, even though rule ($\bowtie$) allows to choose which support set to take from each separation and the corresponding subformula, the number of possible different choices is clearly limited to 2^n where n is the number of possible separations, which, as we just said, is finite and bounded. Moreover, rules ($\Diamond$), ($*$) and ($\langle\otimes\rangle$) will also have a finite number of possible different applications at every sequent.

It is easy to see that all the rules derived from those for $\mathcal{L}_{hp}$ are still backwards sound (Lemma 5.2). Here we just show that also the new ones are so.

Lemma 8.4 (backwards soundness)**.** *Every rule of the tableau system for $\mathcal{L}_{hp}^{*\otimes}$ is backwards sound.*

Proof. For each new (or substantially changed) rule we show that if the sequents in the conclusion are true then also the sequent in the premise is true. The proof follows almost directly from the definition of the semantics of the logic. We only inspect some cases:

- Consider the rule:
$$(\Diamond)\ \frac{C, R, \rho, \Delta \ \models^{\mathcal{E}}\ \langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \varphi}{C', en(C'), \rho[z \mapsto e], \Delta \ \models^{\mathcal{E}}\ \varphi}$$
where $e \in R$ and $C \xrightarrow{\rho(\boldsymbol{x}), \overline{\rho(\boldsymbol{y})} < e}_{\mathsf{a}} C'$.

  Assume that the sequent in the conclusion is true, i.e., $(C', en(C'), \rho[z \mapsto e]) \in \{\!|(\varphi)_\Delta|\!\}^{\mathcal{E}}_{\pi}$, for $\pi \in PEnv_{\mathcal{E}}$. By definition of the semantics, we immediately deduce that $(C, R, \rho) \in \{\!|\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, (\varphi)_\Delta|\!\}^{\mathcal{E}}_{\pi}$. Since $(\langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \varphi)_\Delta = \langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, (\varphi)_\Delta$, this proves that the sequent $C, R, \rho, \Delta \ \models^{\mathcal{E}}\ \langle\!|\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z|\!\rangle\, \varphi$ in the premise is true.

- Consider the rule:
$$(*)\ \frac{C, R, \rho, \Delta \ \models^{\mathcal{E}}\ \varphi * \psi}{C, R', \rho, \Delta \ \models^{\mathcal{E}}\ \varphi \qquad C, R'', \rho, \Delta \ \models^{\mathcal{E}}\ \psi}$$
where $(R', R'') \in \mathsf{Sep}(R)$.

Assume that the sequents in the conclusion are true, i.e., that $(C, R', \rho) \in \{\!|(\varphi)_\Delta|\!\}^{\mathcal{E}}_{\pi}$ and $(C, R'', \rho) \in \{\!|(\psi)_\Delta|\!\}^{\mathcal{E}}_{\pi}$, for $\pi \in PEnv_{\mathcal{E}}$. By definition of the semantics, we immediately deduce that $(C, R, \rho) \in \{\!|(\varphi)_\Delta * (\psi)_\Delta|\!\}^{\mathcal{E}}_{\pi}$. Since $(\varphi * \psi)_\Delta = (\varphi)_\Delta * (\psi)_\Delta$, this proves that the sequent $C, R, \rho, \Delta \models^{\mathcal{E}} \varphi * \psi$ in the premise is true.

- Consider the rule:

$$(\langle\otimes\rangle)\ \frac{C, R, \rho, \Delta \models^{\mathcal{E}} \langle\otimes\rangle\, \varphi}{C, M, \rho, \Delta \models^{\mathcal{E}} \varphi}$$

where $M \in \mathcal{M}(R)$.

Assume that the sequent in the conclusion is true, i.e., $(C, M, \rho) \in \{\!|(\varphi)_\Delta|\!\}^{\mathcal{E}}_{\pi}$, for $\pi \in PEnv_{\mathcal{E}}$. By definition of the semantics, we immediately deduce that $(C, R, \rho) \in \{\!|\langle\otimes\rangle\, (\varphi)_\Delta|\!\}^{\mathcal{E}}_{\pi}$. Since $(\langle\otimes\rangle\, \varphi)_\Delta = \langle\otimes\rangle\, (\varphi)_\Delta$, this proves that the sequent $C, R, \rho, \Delta \models^{\mathcal{E}} \langle\otimes\rangle\, \varphi$ in the premise is true.

□

A finer stop condition

In order to ensure termination of the construction procedure, we resort again to a stop condition in rule (Unf_η). Such side condition will work similarly to that for $\mathcal{L}_{hp}$, but, clearly, will be characterised by a different equivalence. Due to the greater expressive power of $\mathcal{L}^{*\otimes}_{hp}$ we will need an equivalence refining $\approx_r$, taking into account the state of the computation, the causal dependencies between past and future events, as well as the current support set. The aim is still the same: deem two contexts equivalent for a formula if they share the satisfaction of the formula. Recall that properties of $\mathcal{L}^{*\otimes}_{hp}$ allow to impose some second-order constraints on the executable events using support sets, while predicating over their dependency relations. Therefore, first we shall extend the notion of pointed configuration (Definition 5.3) as follows.

Definition 8.14 (pointed supported configuration)**.** *Let $\mathcal{E}$ be a PES and let V be a set. A V-*pointed supported configuration *is a triple $\langle C, R, \zeta\rangle$ where $C \in \mathcal{C}(\mathcal{E})$, $R \in \mathcal{R}(C)$, and $\zeta : V \to C$ is a function.*

As we did in Section 5.2, the notion will be used instantiating the set V with the free variables of a formula and the function ζ with the restriction of the environment in a context. Then, we can define the new equivalence over contexts seen as pointed supported configurations. Two V-pointed supported configurations have isomorphic residuals when they are $\approx_r$-equivalent, i.e., when their residuals are related by an isomorphism ensuring that events pointed by the same $x \in V$ have the same causal relations with the future, and such isomorphism couples the corresponding support sets.

Definition 8.15 (isomorphism of pointed supported residuals)**.** *Let $\mathcal{E}$ be a PES, let V be a set, and let $\langle C, R, \zeta\rangle$ and $\langle C', R', \zeta'\rangle$ be two V-pointed supported configurations of $\mathcal{E}$. We*

say that $\langle C, R, \zeta\rangle$, $\langle C', R', \zeta'\rangle$ *have isomorphic residuals, written* $\langle C, R, \zeta\rangle \approx_{sr} \langle C', R', \zeta'\rangle$, *if there is an isomorphism of the residuals* $\iota : \mathcal{E}[C] \to \mathcal{E}[C']$ *such that* $R' = \iota(R)$ *and for all* $x \in V$, $e \in \mathcal{E}[C]$ *we have* $\zeta(x) \leq e$ *iff* $\zeta'(x) \leq \iota(e)$.

Observe that, in general, if there is an isomorphism ι between the residuals of two configurations C, C', then it must hold that $\mathcal{R}(C') = \{\iota(R) \mid R \in \mathcal{R}(C)\}$, because the isomorphism of PESs preserves conflicts and enabled events. Furthermore, we must also have that for all $R \in \mathcal{R}(C)$ and $R' \in \mathcal{R}(C')$, if $R' = \iota(R)$, then $\mathcal{M}(R') = \{\iota(M) \mid M \in \mathcal{M}(R)\}$ and $\mathsf{Sep}(R') = \{(\iota(R_1), \iota(R_2)) \mid (R_1, R_2) \in \mathsf{Sep}(R)\}$. Clearly this is crucial if we want that equivalent pointed supported configurations satisfy the same separating and second-order modal formulae of $\mathcal{L}^{*\otimes}_{hp}$.

As we mentioned in the tableau system for $\mathcal{L}_{hp}$, since we are dealing with formulae containing free propositional variables, we need proposition environments satisfying a suitable property, that we called saturated proposition environments (Definition 5.5). Such notion is needed here as well, extended in the obvious way for $\mathcal{L}^{*\otimes}_{hp}$ (therefore we just point to the previous definition). In words, in order to be saturated for a formula φ, a proposition environment must assign to each free proposition in φ an evaluation that respects the equivalence $\approx_{sr}$ over $fv(\varphi)$-pointed supported configurations. Then we can show that pointed supported configurations with isomorphic residuals satisfy exactly the same formulae of $\mathcal{L}^{*\otimes}_{hp}$ when free event variables correspond to the pointed events.

Lemma 8.5 (equisatisfaction in equivalent pointed supported configurations)**.** *Let* $\mathcal{E}$ *be a PES, let* φ *be a formula of* $\mathcal{L}^{*\otimes}_{hp}$, *let* $\pi \in PEnv_{\mathcal{E}}$ *be a proposition environment saturated for* φ, *and let* $(C_1, R_1, \rho_1), (C_2, R_2, \rho_2) \in \mathcal{C}(\mathcal{E}) \times \mathcal{R}_{\mathcal{E}} \times Env_{\mathcal{E}}$. *If* $\langle C_1, R_1, \rho_1|_{fv(\varphi)}\rangle \approx_{sr} \langle C_2, R_2, \rho_2|_{fv(\varphi)}\rangle$ *then* $(C_1, R_1, \rho_1) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ *iff* $(C_2, R_2, \rho_2) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$.

Proof. Assume that $\langle C_1, R_1, \rho_1|_{fv(\varphi)}\rangle \approx_{sr} \langle C_2, R_2, \rho_2|_{fv(\varphi)}\rangle$, via an isomorphism ι. We prove that if $(C_1, R_1, \rho_1) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$ then $(C_2, R_2, \rho_2) \in \{\!|\varphi|\!\}^{\mathcal{E}}_{\pi}$. Since the isomorphism ι is bijective, the other implication follows by symmetry. The proof proceeds by induction on the formula φ. Being similar to the proof of Lemma 5.3, we just examine some significant cases among those for the new operators (originating from logics SFL and $\mathbb{L}_\mu$) and those with substantial changes in the semantics:

- $\varphi = \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z\rangle\!\rangle\, \psi$
  Assume that $(C_1, R_1, \rho_1) \in \{\!|\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z\rangle\!\rangle\, \psi|\!\}_{\pi}$. By definition of the semantics there exists an event $e \in R_1$ such that $C_1 \xrightarrow{\rho_1(\boldsymbol{x}), \overline{\rho_1(\boldsymbol{y})} < e}_{\mathsf{a}} C_1'$, $\rho_1' = \rho_1[z \mapsto e]$, and $(C_1', en(C_1'), \rho_1') \in \{\!|\psi|\!\}_{\pi}$. Since ι is an isomorphism of pointed supported residuals, we immediately deduce that $\iota(e)$ is enabled at C_2, $\iota(e) \in R_2$ and
  $$C_2 \xrightarrow{\rho_2(\boldsymbol{x}), \overline{\rho_2(\boldsymbol{y})} < \iota(e)}_{\mathsf{a}} C_2' \tag{8.1}$$
  Clearly, the restriction of ι to $\mathcal{E}[C_1']$ is an isomorphism $\iota' : \mathcal{E}[C_1'] \to \mathcal{E}[C_2']$. Let $\rho_2' = \rho_2[z \mapsto \iota(e)]$. It is easy to see that for all $x \in fv(\psi)$, $e_1 \in \mathcal{E}[C_1']$, it holds $\rho_1'(x) \leq e_1$ iff $\rho_2'(x) \leq \iota'(e_1)$. In fact, let $x \in fv(\psi) \subseteq fv(\varphi) \cup \{z\}$:

(i) if $x \neq z$ we can observe that $e_1 \in \mathcal{E}[C_1'] \subseteq \mathcal{E}[C_1]$ and $\rho_i(x) = \rho_i'(x)$ for $i \in \{1,2\}$. Then we conclude using the fact that ι is an isomorphism of pointed supported residuals

(ii) if $x = z$ then $\rho_1'(z) = e \in \mathcal{E}[C_1]$ and $\rho_2'(z) = \iota(e) \in \mathcal{E}[C_2]$. Since ι is in particular an isomorphism of PESs, we know that $\rho_1'(x) = e \leq e_1$ iff $\rho_2'(x) = \iota(e) \leq \iota(e_1) = \iota'(e_1)$.

Furthermore, since ι' is in particular an isomorphism of PESs between $\mathcal{E}[C_1']$ and $\mathcal{E}[C_2']$, we have also that $en(C_2') = \iota'(en(C_1'))$.

The facts above show that $\langle C_1', en(C_1'), \rho_1'|_{fv(\psi)}\rangle \approx_{sr} \langle C_2', en(C_2'), \rho_2'|_{fv(\psi)}\rangle$. Then, by inductive hypothesis, $(C_2', en(C_2'), \rho_2') \in \{\!|\psi|\!\}_\pi$. Recalling (8.1), we conclude that $(C_2, R_2, \rho_2) \in \{\!|\langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\ z\rangle\!\rangle\ \psi|\!\}_\pi$ as desired.

- $\varphi = \psi_1 * \psi_2$
  Assume that $(C_1, R_1, \rho_1) \in \{\!|\psi_1 * \psi_2|\!\}_\pi$. By definition of the semantics there exists a separation $(R_1', R_1'') \in \mathsf{Sep}(R_1)$ such that $(C_1, R_1', \rho_1) \in \{\!|\psi_1|\!\}_\pi$ and $(C_1, R_1'', \rho_1) \in \{\!|\psi_2|\!\}_\pi$. Since ι is an isomorphism of pointed supported residuals, we know that $R_2 = \iota(R_1)$. Then, as observed before, we have that $\mathsf{Sep}(R_2) = \{(\iota(R'), \iota(R'')) \mid (R', R'') \in \mathsf{Sep}(R_1)\}$. Hence, there exists $(R_2', R_2'') \in \mathsf{Sep}(R_2)$ such that $R_2' = \iota(R_1')$ and $R_2'' = \iota(R_1'')$. So we know that ι is also a valid isomorphism between the residuals of the corresponding pointed supported configurations $\langle C_1, R_1', \rho_1|_{fv(\psi_1)}\rangle \approx_{sr} \langle C_2, R_2', \rho_2|_{fv(\psi_1)}\rangle$ and $\langle C_1, R_1'', \rho_1|_{fv(\psi_2)}\rangle \approx_{sr} \langle C_2, R_2'', \rho_2|_{fv(\psi_2)}\rangle$, since $fv(\psi_i) \subseteq fv(\varphi)$ for $i \in \{1,2\}$. Now we can use the inductive hypothesis to obtain $(C_2, R_2', \rho_2) \in \{\!|\psi_1|\!\}_\pi$ and $(C_2, R_2'', \rho_2) \in \{\!|\psi_2|\!\}_\pi$ and conclude $(C_2, R_2, \rho_2) \in \{\!|\psi_1 * \psi_2|\!\}_\pi$.

- $\varphi = \langle\otimes\rangle\ \psi$
  Assume that $(C_1, R_1, \rho_1) \in \{\!|\langle\otimes\rangle\ \psi|\!\}_\pi$. By definition of the semantics there exists a complete supset $M_1 \in \mathcal{M}(R_1)$ such that $(C_1, M_1, \rho_1) \in \{\!|\psi|\!\}_\pi$. Since ι is an isomorphism of pointed supported residuals, we know that $R_2 = \iota(R_1)$. Then, we also have that $\mathcal{M}(R_2) = \{\iota(M) \mid M \in \mathcal{M}(R_1)\}$. Hence, there exists $M_2 \in \mathcal{M}(R_2)$ such that $M_2 = \iota(M_1)$. So we know that ι is also a valid isomorphism between the residuals of the pointed supported configurations $\langle C_1, M_1, \rho_1|_{fv(\psi)}\rangle \approx_{sr} \langle C_2, M_2, \rho_2|_{fv(\psi)}\rangle$, since $fv(\psi) = fv(\varphi)$. Now we can use the inductive hypothesis to obtain $(C_2, M_2, \rho_2) \in \{\!|\psi|\!\}_\pi$ and conclude $(C_2, R_2, \rho_2) \in \{\!|\langle\otimes\rangle\ \psi|\!\}_\pi$.

□

Recall that the proposition environment is irrelevant for establishing when a well-formed sequent is true because the instantiation of the formula in the sequent is always closed. Furthermore, every proposition environment is vacuously saturated for a closed formula. Therefore, when two contexts C, R, ρ, Δ and C', R', ρ', Δ' for a formula φ seen as pointed supported configurations have isomorphic residuals $\langle C, R, \rho|_{fv(\varphi)}\rangle \approx_r \langle C', R', \rho'|_{fv(\varphi)}\rangle$, the

lemma above ensures that the corresponding sequents share the truth value, as long as the two instantiations of the formula $(\varphi)_\Delta$ and $(\varphi)_{\Delta'}$ are the same. The latter is guaranteed to happen if the sequents come from the same tableau built starting from a closed formula.

We can now proceed with the definition of the new stop condition. Given a tableau for a closed formula and a node labelled by $C, R, \rho, \Delta \models Z(\boldsymbol{y})$, with $Z(\boldsymbol{x}) = \psi$ in Δ, necessarily $\psi = \eta Z(\boldsymbol{x}).\varphi$ and the node has some ancestor introducing Z. Then, recall that we denote by $\Delta^{\uparrow}(Z)$ the closest ancestor introducing Z.

Definition 8.16 (stop condition)**.** *The* stop condition γ *for rule* (Unf_η) *in Table 8.1 is as follows:*

there is an ancestor of the premise $C, R, \rho, \Delta \models Z(\boldsymbol{z})$ *labelled* $C', R', \rho', \Delta' \models Z(\boldsymbol{y})$*, such that* $\Delta^{\uparrow}(Z) = \Delta'^{\uparrow}(Z)$ *and* $\langle C, R, \rho[\boldsymbol{x} \mapsto \rho(\boldsymbol{z})]|_{\boldsymbol{x}}\rangle \approx_{sr} \langle C', R', \rho'[\boldsymbol{x} \mapsto \rho'(\boldsymbol{y})]|_{\boldsymbol{x}}\rangle$.

Informally, the stop condition holds when in a previous step of the construction of the tableau an instance of the same abstract proposition has been unfolded in an equivalent context, without being reintroduced. Then we can safely avoid to continue along this path because, intuitively, it would not add new information. Instead, when the stop condition fails, it makes sense to further unfold the fixpoint since the current context is still "different enough" from those previously encountered. As it happened in the other tableau system, the equivalence of contexts is checked after renaming the variables to those associated with Z in the fixpoint formula quantifying the proposition.

In the example below we show how the tableau system behaves on some of the new operators introduced. Consider the formula $\varphi = \mu X.(\langle\otimes\rangle(\mathsf{F} \bowtie \mathsf{F}) \vee \langle\!|\_ x|\!\rangle X)$, obtained by inserting that in Subsection 8.3.1 in an "eventually construct". Indeed, the property requires that eventually a state is reached from where an event can be executed which is in conflict with all the other enabled events, if any. We previously claimed that the proper subformula $\langle\otimes\rangle(\mathsf{F} \bowtie \mathsf{F})$ was not satisfied by $\mathcal{E}_3$ in Figure 3.1c. Here we show that eventually the property holds anyway. A tableau for the sequent $\emptyset, en(\emptyset), \rho, \emptyset \models \varphi$ is given in Figure 8.3. Observe that the stop condition does not hold in the second application of the unfolding rule (hence its application) since $\langle\emptyset, en(\emptyset), \rho|_\emptyset\rangle \not\approx_{sr} \langle\{\mathsf{c}\}, en(\{\mathsf{c}\}), \rho[x \mapsto \mathsf{c}]|_\emptyset\rangle$. No rule can be applied to the bottom sequent $\{\mathsf{c}\}, \{\mathsf{a}^0\}, \rho[x \mapsto \mathsf{c}], \{X = \varphi\} \models \mathsf{F} \bowtie \mathsf{F}$ because there is no possible separation of the support set $\{\mathsf{a}^0\}$, i.e., $\mathsf{Sep}(\{\mathsf{a}^0\}) = \emptyset$.

Correctness of the tableau system

We conclude by showing that, in fact, the truth of a closed formula of $\mathcal{L}^{*\otimes}_{hp}$ over a strongly regular PES reduces to the existence of a successful tableau. Given a PES $\mathcal{E}$, a tableau for a formula φ of $\mathcal{L}^{*\otimes}_{hp}$ is a tableau built applying the rules given in Table 8.1 starting from a sequent $\emptyset, en(\emptyset), \rho, \emptyset \models^{\mathcal{E}} \varphi$, where ρ is any environment (it is irrelevant since φ is closed). A maximal tableau, where no rule applies to any leaf, is considered successful when it satisfies the following conditions.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\emptyset, en(\emptyset), \rho, \emptyset \models \varphi}{\emptyset, en(\emptyset), \rho, \{X = \varphi\} \models X}\,(\mathsf{Int})}{\emptyset, en(\emptyset), \rho, \{X = \varphi\} \models \langle\otimes\rangle(\mathsf{F} \bowtie \mathsf{F}) \vee \langle\!|\_ x|\!\rangle X}\,(\mathsf{Unf}_\mu)}{\emptyset, en(\emptyset), \rho, \{X = \varphi\} \models \langle\!|\_ x|\!\rangle X}\,(\vee_{\mathsf{R}})}{\{\mathsf{c}\}, en(\{\mathsf{c}\}), \rho[x \mapsto \mathsf{c}], \{X = \varphi\} \models X}\,(\Diamond)}{\{\mathsf{c}\}, en(\{\mathsf{c}\}), \rho[x \mapsto \mathsf{c}], \{X = \varphi\} \models \langle\otimes\rangle(\mathsf{F} \bowtie \mathsf{F}) \vee \langle\!|\_ x|\!\rangle X}\,(\mathsf{Unf}_\mu)}{\{\mathsf{c}\}, en(\{\mathsf{c}\}), \rho[x \mapsto \mathsf{c}], \{X = \varphi\} \models \langle\otimes\rangle(\mathsf{F} \bowtie \mathsf{F})}\,(\vee_{\mathsf{L}})}{\{\mathsf{c}\}, \{\mathsf{a}^0\}, \rho[x \mapsto \mathsf{c}], \{X = \varphi\} \models \mathsf{F} \bowtie \mathsf{F}}\,(\langle\otimes\rangle)
$$

Figure 8.3: A successful tableau for $\mu X.(\langle\otimes\rangle(\mathsf{F} \bowtie \mathsf{F}) \vee \langle\!|\_ x|\!\rangle X)$ in $\mathcal{E}_3$.

Definition 8.17 (successful tableau)**.** *A* successful tableau *for $\mathcal{L}_{hp}^{*\otimes}$ is a finite maximal tableau where every leaf is labelled by a sequent $C, R, \rho, \Delta \models^{\mathcal{E}} \varphi$ such that one of the following holds:*

1. $\varphi = \mathsf{T}$
2. $\varphi = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \psi$
3. $\varphi = \psi_1 \bowtie \psi_2$
4. $\varphi = Z(\boldsymbol{y})$ *and* $\Delta(Z(\boldsymbol{x})) = \nu Z(\boldsymbol{x}).\psi$.

Notice the absence of the case $\varphi = [\otimes]\psi$ motivated by the fact that every support set R has at least one complete supset (see Definition 8.7), i.e., $\mathcal{M}(R) \neq \emptyset$, hence the sequent could not label a leaf. For instance, the tableau in Figure 8.3 is successful.

As in the case of tableaux for $\mathcal{L}_{hp}$, the construction of a tableau can be nondeterministic. However, as we already observed, when the PES is boundedly branching, at each step of the procedure the number of possible different applications of rules is finite and bounded. This will guarantee that the number of maximal tableaux for a sequent is finite as long as all tableaux are finite. So, we first aim at proving that all tableaux for a sequent are finite.

As we described in depth in Subsection 5.3.1, a basic observation is that an infinite tableau would necessarily include a path where the same proposition is unfolded infinitely many times without being reintroduced. This can be proved relying on the same Lemmata 5.5 and 5.7, which can be easily seen to hold also in the case of $\mathcal{L}_{hp}^{*\otimes}$. Now, in order to conclude, we observe that in strongly regular PESs the number of pointed supported residuals is finite up to isomorphism. Again also the converse holds when the PES is boundedly branching.

Lemma 8.6 (strong regularity and pointed supported configurations)**.** *A PES $\mathcal{E}$ is strongly regular if and only if it is boundedly branching and for any fixed finite set V, the equivalence $\approx_{sr}$ is of finite index over V-pointed supported configurations of $\mathcal{E}$.*

Proof. Let V be a finite set and let $\mathcal{C}_V = \{\langle C, R, \zeta\rangle \mid C \in \mathcal{C}(\mathcal{E}) \wedge R \in \mathcal{R}(C) \wedge \zeta : V \to C\}$ be the set of V-pointed supported configurations.

Since clearly $\approx_{sr}$ refines $\approx_r$, if $\mathcal{C}_V$ is finite up to $\approx_{sr}$, then the set of pointed configurations of $\mathcal{E}$ is finite up to $\approx_r$. Thus, if $\mathcal{E}$ is boundedly branching, by Lemma 5.4 we immediately obtain that $\mathcal{E}$ is strongly regular.

Now we prove the other direction. By strong regularity $\mathcal{E}$ is boundedly branching, and by Lemma 5.4 the set $\{\langle C, \zeta\rangle \mid C \in \mathcal{C}(\mathcal{E}) \wedge \zeta : V \to C\}$ is finite up to $\approx_r$. This does not immediately imply that $\mathcal{C}_V$ is finite up to $\approx_{sr}$. In fact, given two V-pointed supported configurations $\langle C, R, \zeta\rangle$, $\langle C', R'\zeta'\rangle$, an isomorphism of pointed residuals $\iota : \mathcal{E}[C] \to \mathcal{E}[C']$ is not necessarily an isomorphism of the corresponding pointed supported residuals since we additionally need that $\iota(R) = R'$.

Assume, by contradiction, that $\mathcal{C}_V$ is not finite up to $\approx_{sr}$. Then we can find an infinite sequence of V-pointed supported configurations $\langle C_i, R_i, \zeta^i\rangle$, $i \in \mathbb{N}$ such that the corresponding V-pointed configurations $\langle C_i, \zeta^i\rangle$ are all $\approx_r$-equivalent, while $\langle C_i, R_i, \zeta^i\rangle$ are pairwise non-equivalent with respect to $\approx_{sr}$. Let $\iota_i : \mathcal{E}[C_i] \to \mathcal{E}[C_{i+1}]$ be isomorphisms of pointed residuals for all $i \in \mathbb{N}$ and denote by $\iota_{i,j} : \mathcal{E}[C_i] \to \mathcal{E}[C_j]$ the isomorphism resulting as the composition $\iota_{j-1} \circ \ldots \circ \iota_{i+1} \circ \iota_i$.

Since the pointed supported configurations in the sequence are pairwise non-equivalent, for all $i, j \in \mathbb{N}$, $i < j$, we must have $R_j \neq \iota_{i,j}(R_i)$. Then the key observations are the following. For all $i \in \mathbb{N}$

1. each set of support sets $\mathcal{R}(C_i)$ is finite, since $\mathcal{E}$ is boundedly branching
2. $\mathcal{R}(C_i) = \{\iota_{1,i}(R) \mid R \in \mathcal{R}(C_1)\}$, since $\iota_{1,i} : \mathcal{E}[C_1] \to \mathcal{E}[C_i]$ is a PES isomorphism between the residuals of the configurations C_1, C_i.

So, for all $j \in \mathbb{N}$, there exists $R \in \mathcal{R}(C_1)$ such that $R_j = \iota_{1,j}(R)$, and for all $i < j$, there must exist $R' \in \mathcal{R}(C_1)$ such that $R_i = \iota_{1,i}(R')$ and $R_j = \iota_{1,j}(R) \neq \iota_{i,j}(R_i) = \iota_{1,j}(R')$, hence $R \neq R'$. This would imply the existence of infinitely many different support sets in $\mathcal{R}(C_1)$, contradicting its finiteness. Therefore $\mathcal{C}_V$ must be finite up to $\approx_{sr}$. □

From the existence of an infinite path where the same proposition is unfolded infinitely many times without being reintroduced, and the fact that strongly regular PESs have only finitely many equivalence classes of pointed supported residuals, it is easy to deduce that actually there cannot be an infinite tableau, since at some point the stop condition would have held preventing the continuation of the path. Therefore, all tableaux for strongly regular PESs are finite and the number of possible tableaux for a sequent is also finite.

Theorem 8.2 (tableaux finiteness)**.** *Given a strongly regular PES $\mathcal{E}$ and a closed formula φ in $\mathcal{L}_{hp}^{*\otimes}$, every tableau for a sequent $C, R, \rho, \Delta \models^{\mathcal{E}} \varphi$ is finite. Hence the number of tableaux for $C, R, \rho, \Delta \models^{\mathcal{E}} \varphi$ is finite.*

Proof. The proof proceeds by contradiction. Suppose that there is an infinite tableau τ for the sequent $C, R, \rho, \Delta \models^{\mathcal{E}} \varphi$. By Lemma 5.7, in τ there is an infinite path p where a proposition Z occurs infinitely many times without being introduced. Let $Z(\boldsymbol{x}) = \eta Z(\boldsymbol{x}).\, \psi$ be the definition of Z in the tableau.

By Lemma 8.6 the set of $\boldsymbol{x}$-pointed supported configurations of $\mathcal{E}$ is finite up to $\approx_{sr}$. Since the proposition Z is unfolded infinitely many times along p without being introduced, there are infinitely many sequents $C', R', \rho', \Delta' \models^{\mathcal{E}} Z(\boldsymbol{x}')$ for which $\Delta'^{\uparrow}(Z)$ is the same node. Hence the stop condition γ is necessarily satisfied at some point of the path, contradicting its infiniteness.

We next prove that also the number of tableaux is finite. Consider a tree where nodes are tableaux rooted $C, R, \rho, \Delta \models^{\mathcal{E}} \varphi$ and where the successors of each tableau τ are the tableaux obtained by extending τ with the application of a rule. Since $\mathcal{E}$ is strongly regular, the tree is finitely branching. If it was infinite, by Lemma 5.6 there would be an infinite path meaning that there would be an infinite sequence of tableaux $(\tau_i)_{i \in \omega}$, such that τ_{i+1} extends τ_i. This in turn implies the existence of an infinite tableau, that contradicts the first part. □

Soundness and completeness of the tableau system can be proved in the same way as for $\mathcal{L}_{hp}$, relying on the reduction of the semantics of fixpoint formulae to that of finite approximants and the backwards soundness of the rules. Here we do not report all the proof passages, described in depth in Subsection 5.3.2. We just mention that approximants for fixpoint formulae of $\mathcal{L}_{hp}^{*\otimes}$ can be defined in the same way of those for $\mathcal{L}_{hp}$ (Definition 5.10). Then, they satisfy the properties in Lemma 5.8, as a result of Lemma 8.6.

The notions of ν- and μ-pseudo-tableaux for $\mathcal{L}_{hp}^{*\otimes}$ can be given as in Definitions 5.11 and 5.12, adapting the rules defined there in the obvious way, adding support sets. Then, the crucial result about ν-pseudo-tableaux (Lemma 5.10) can be easily extended to $\mathcal{L}_{hp}^{*\otimes}$, noticing that any leaf labelled by a sequent of the kind $C, R, \rho, \Delta \models \varphi_1 \bowtie \varphi_2$ is clearly true. The fact that it is a leaf implies that the support set R has no possible separations, i.e., $\mathsf{Sep}(R) = \emptyset$, hence the formula is always satisfied by definition of the semantics. Then, we immediately obtain the same corresponding soundness result by virtue of backwards soundness of the tableau rules for $\mathcal{L}_{hp}^{*\otimes}$.

Lemma 8.7 (soundness)**.** *Let $\mathcal{E}$ be a strongly regular PES and let φ be a closed formula of $\mathcal{L}_{hp}^{*\otimes}$. If φ has a successful ν-pseudo-tableau (hence in particular, if it has a successful tableau) then $\mathcal{E}$ satisfies φ.*

Proof. Assume that the sequent $C, R, \rho, \emptyset \models^{\mathcal{E}} \varphi$ has a successful tableau τ. Then τ is a successful ν-pseudo-tableau. For all leaves (and thus for all false leaves) $C, R, \rho, \Delta \models X(\boldsymbol{z})$,

$\Delta^{\uparrow}(X)$ is in τ. Therefore, if it had a false leaf, by Lemma 5.10 we could continue building strictly smaller ν-pseudo-tableau with the same properties, contradicting the finiteness of τ. Hence all the sequents labelling the leaves of τ must be true, a fact that, by Lemma 8.4 implies that all the nodes of τ are true and thus, in particular, the sequent $C, R, \rho, \emptyset \models^{\mathcal{E}} \varphi$ labelling the root is true, as desired. □

We can prove completeness of the tableau system by extending the proof of Lemma 5.13 with the cases for the new operators and reviewing those for the diamond and box modalities.

Lemma 8.8 (completeness)**.** *Let $\mathcal{E}$ be a strongly regular PES and let φ be a closed formula of $\mathcal{L}^{*\otimes}_{hp}$. If $\mathcal{E}$ satisfies φ then φ has a successful μ-pseudo-tableau and thus a successful tableau.*

Proof. Recall that the proof proceeds by constructing a successful tableau inductively, based on the shape of the consequent of the current sequent $C, R, \rho, \Delta \models \varphi$, showing which rule to apply and arguing that the conclusion of the rule is again a true sequent. Here we show only the cases which are new or different from before.

- $\varphi = \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle\, \psi$
  Since $(C, R, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$ and $(\varphi)_\Delta = \langle\!\langle \boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z \rangle\!\rangle\, (\psi)_\Delta$, by definition of the semantics there exists an event $e \in R$ such that $C \xrightarrow{\rho(\boldsymbol{x}), \rho(\boldsymbol{y}) < e}_{\mathsf{a}} C'$, $\rho' = \rho[z \mapsto e]$ and $(C', en(C'), \rho') \in \{\!|(\psi)_\Delta|\!\}_\pi$. Then we can apply rule $(\Diamond)$ producing a single true successor labelled $C', en(C'), \rho', \Delta \models \psi$.

- $\varphi = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, \psi$
  Note that the successors $\mathsf{Succ}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}_{\mathcal{E}}(C, R, \rho)$ are finite since the PES $\mathcal{E}$ is strongly regular and thus finitely branching. Let the set of successor be $\mathsf{Succ}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}_{\mathcal{E}}(C, R, \rho) = \{(C_1, en(C_1), \rho_1), \ldots, (C_n, en(C_n), \rho_n)\}$.
  If $\mathsf{Succ}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}_{\mathcal{E}}(C, R, \rho)$ is not empty, since $(C, R, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$ and $(\varphi)_\Delta = [\![\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}\, z]\!]\, (\psi)_\Delta$, by definition of the semantics we have that $(C_i, en(C_i), \rho_i) \in \{\!|(\psi)_\Delta|\!\}_\pi$ for all $i \in [1, n]$. Then rule $(\Box)$ can be applied producing n true successors labelled $C_i, en(C_i), \rho_i, \Delta \models \psi$, for $i \in [1, n]$.
  Otherwise, if $\mathsf{Succ}^{\boldsymbol{x}, \overline{\boldsymbol{y}} < \mathsf{a}z}_{\mathcal{E}}(C, R, \rho)$ is empty, the node has no successor.

- $\varphi = \psi_1 * \psi_2$
  Since $(C, R, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$ and $(\varphi)_\Delta = (\psi_1)_\Delta * (\psi_2)_\Delta$, by definition of the semantics there exists a separation $(R_1, R_2) \in \mathsf{Sep}(R)$ such that $(C, R_1, \rho) \in \{\!|(\psi_1)_\Delta|\!\}_\pi$ and $(C, R_2, \rho) \in \{\!|(\psi_2)_\Delta|\!\}_\pi$. Then we can apply rule $(*)$ producing two true successors labelled $C, R_1, \rho, \Delta \models \psi_1$ and $C, R_2, \rho, \Delta \models \psi_2$, respectively.

- $\varphi = \psi_1 \bowtie \psi_2$
  Let the set of separations of R be $\mathsf{Sep}(R) = \{(R^1_1, R^2_1), \ldots, (R^1_n, R^2_n)\}$. Note that the set $\mathsf{Sep}(R)$ is finite since the PES $\mathcal{E}$ is strongly regular and thus finitely branching.

If $\mathsf{Sep}(R)$ is not empty, since $(C, R, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$ and $(\varphi)_\Delta = (\psi_1)_\Delta \bowtie (\psi_2)_\Delta$, by definition of the semantics we have that for each separation $(R_i^1, R_i^2) \in \mathsf{Sep}(R)$, for all $i \in [1, n]$, there exists $p_i \in \{1, 2\}$ such that $(C, R_i^{p_i}, \rho) \in \{\!|(\psi_{p_i})_\Delta|\!\}_\pi$. Then rule $(\bowtie)$ can be applied producing n true successors labelled $C, R_i^{p_i}, \rho, \Delta \models \psi_{p_i}$, for $i \in [1, n]$.

Otherwise, if $\mathsf{Sep}(R)$ is empty, the node has no successor.

- $\varphi = \langle\otimes\rangle\, \psi$
  Since $(C, R, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$ and $(\varphi)_\Delta = \langle\otimes\rangle\, (\psi)_\Delta$, by definition of the semantics there exists a complete supset $M \in \mathcal{M}(R)$ such that $(C, M, \rho) \in \{\!|(\psi)_\Delta|\!\}_\pi$. Then we can apply rule $(\langle\otimes\rangle)$ producing a single true successor labelled $C, M, \rho, \Delta \models \psi$.

- $\varphi = [\otimes]\, \psi$
  Let the set of complete supsets of R be $\mathcal{M}(R) = \{M_1, \ldots, M_n\}$. Note that the set $\mathcal{M}(R)$ is finite since the PES $\mathcal{E}$ is strongly regular and thus finitely branching. Furthermore $\mathcal{M}(R)$ cannot be empty by definition of complete supset.

  Since $(C, R, \rho) \in \{\!|(\varphi)_\Delta|\!\}_\pi$ and $(\varphi)_\Delta = [\otimes]\, (\psi)_\Delta$, by definition of the semantics we have that $(C, M_i, \rho) \in \{\!|(\psi)_\Delta|\!\}_\pi$ for all $i \in [1, n]$. Then rule $([\otimes])$ can be applied producing n true successors labelled $C, M_i, \rho, \Delta \models \psi$, for $i \in [1, n]$.

Then the rest of the proof, showing that the constructed tableau is successful, is as in Lemma 5.13. □

Combining the lemmata for soundness and completeness we obtain the desired result.

Theorem 8.3 (soundness and completeness of the tableau system)**.** *Given a strongly regular PES $\mathcal{E}$ and a closed formula φ of $\mathcal{L}_{hp}^{*\otimes}$, the formula φ has a successful tableau if and only if $\mathcal{E}$ satisfies φ.*

Proof. Corollary of Lemmata 8.7 and 8.8. □

Chapter 9

Conclusions

We studied the model-checking problem in the logic for true concurrency $\mathcal{L}_{hp}$, representing the logical counterpart of a classical true concurrent behavioural equivalence, i.e., history-preserving bisimilarity. We explored three different approaches to the problem. First, resorting to a tableau-based technique we showed that the problem is decidable for the class of strongly regular PESs, that include regular trace PESs. We then devised an automata-theoretic model-checking procedure relying on parity tree automata, amenable of a more efficient implementation. As an example of instantiation on a concrete formalism, we showed how the technique can be implemented on finite safe Petri nets, also producing a proof-of-concept tool. Lastly, we took a broader approach to the problem and to formal verification in general. We developed a theoretical framework for the characterisation of the solution of systems of fixpoint equations over lattices which intervene as an essential ingredient in a number of verification problems, showing how it instantiates to the model-checking of $\mathcal{L}_{hp}$. We characterised the solution of such systems by means of a parity game, identifying continuous lattices as a general and appropriate setting for the theory. We provided techniques for the computation of winning strategies in the game, based on progress measures à la Jurdziński, selections and a logic for the symbolic representation of players' moves.

We proved that the class of regular trace PESs, corresponding exactly to finite safe Petri nets, is included in that of strongly regular PESs which, in turn, is included in the class of regular PESs. The precise relation of strongly regular PESs with the other two classes is still unclear and interesting in view of [CC17] that recently showed that regular trace PESs are strictly included in regular PESs, disproving Thiagarajan's conjecture. We remark that any example of regular but not strongly regular PES would be a counterexample to the conjecture. Indeed, especially since the proof in [CC17] is quite involved, we hope that a formal study of the relation between these classes of PESs will lead to a simpler counterargument to Thiagarajan's conjecture.

In the literature there are several other works dealing with model-checking for logics on event structures. In [Pen97] a technique is proposed for model-checking a CTL-style logic

with modalities for immediate causality and conflict on a subclass of PESs. The logic is quite different from $\mathcal{L}_{hp}$ as formulae are satisfied by single events, the idea being that an event, with its causes, represents the local state of a component. The procedure involves the construction of a finite representation of the PES associated with a program which has some conceptual relation with our quotienting phase for automata and games. In [Mad03] the author shows that first-order logic and monadic trace logic (MTL), a restricted form of monadic second-order logic (MSOL), are decidable on regular trace event structures. The possibility of directly observing conflicts in MTL and thus of distinguishing behaviourally equivalent PESs (e.g., the PESs consisting of a single or two conflicting copies of an event), and the presence in $\mathcal{L}_{hp}$ of propositions which are non-monadic with respect to event variables, make the logics not immediate to compare. Still, a deeper investigation is definitively worth to pursue, especially in view of the fact that, in the propositional case, the mu-calculus corresponds to the bisimulation invariant fragment of MSOL [JW96]. Moreover, understanding which are the bisimulation invariant fragments of MSOL over event structures, with respect to the various concurrent bisimulations, represents an interesting program in itself.

The true concurrent approach sees also recent applications in the design and development of tools for the specification and verification of consistency models and properties (see [AMT14]) for concurrent and distributed systems, both software and hardware, particularly in the setting of weak memory consistency [DSB86]. Among these tools, the `cat` language [ACM16] provides means to succinctly describe consistency models and properties in terms of relations over instructions, seen as events, and constraints on such relations, such as acyclicity or irreflexiveness, or even more complex ones. Properly defined relations over events provide partial order semantics, allowing to identify and verify truly concurrent behaviours, such as race conditions, for example, in an automatic way. Not surprisingly, these features are, in a sense, similar to those offered by a logic for true concurrency, indeed, the modelled properties could possibly be seen as logical properties about the dependency relations over the events in the computations.

The automata-based tool, for the model-checking of $\mathcal{L}_{hp}$ over finite safe Petri nets, is still very preliminary. As suggested by its (wishful) name (inspired to the classical Edinburgh Concurrency Workbench [SS98]) we would like to bring the TCWB to a more mature stage, working on optimisations and adding an interface that gives access to a richer set of commands. In particular, the tool is missing two main optimizations: a strategy for an efficient exploration of the automaton, and efficient data structures and ways to maintain and update them. Both have been carefully considered in [SS98] for CWB and shown to lead to great improvements in performances. For instance, CWB uses so-called *assumptions* and *decisions* for choice-points during the exploration in order to efficiently build the winning strategy for a player. Despite the fact that our procedure is based on automata instead of games, we believe that similar techniques could be exploited also in the TCWB. Moreover, we would like to investigate the applicability of the tool to the verification of properties

that can naturally take advantage from true concurrent models and specification languages. For instance, we already mentioned that the data race freedom [AHMN91], non-interference [GM82] and causal atomicity [FM06] properties can be expressed in $\mathcal{L}_{hp}$. We would like to extend the scope to other properties benefiting from the true concurrent approach, and possibly compare our tool with others from the literature developed specifically for the verification of those properties. From a more theoretical standpoint, in Section 6.4 we provided some upper bounds to the size of the automaton built for model-checking a formula on a safe net, however more work is required towards an adequate study of the complexity of the model-checking problem at hand. In particular, the question remains open about the lower bounds on the time or space. An interesting research direction would be to study the problem in the setting of alternating Turing machines [CKS81], especially suited for decision problems involving alternating quantifiers such as this one.

The logic $\mathcal{L}_{hp}^{*\otimes}$, proposed as an extension of $\mathcal{L}_{hp}$ with incomparable operators from the logics in [Gut11], is clearly one of the most expressive true concurrent logics for which model-checking has been proved decidable. The automata-based tool could be adapted for accepting also formulae of $\mathcal{L}_{hp}^{*\otimes}$, though the practical interest in such an expressive logic might be limited. Moreover, it remains open which behavioural equivalence $\mathcal{L}_{hp}^{*\otimes}$ corresponds to and whether it is decidable on some class of models. Still, as we mentioned, such a behavioural equivalence would surely be strictly coarser than hereditary history-preserving bisimilarity. Some ideas in this regard can be already derived from [Gut11], at least for a class of systems, called fc-structured systems, corresponding to a subclass of safe Petri nets.

Another open issue concerns the possibility of generalising the results in the thesis to the full logic $\mathcal{L}$ in [BC14]. This is quite challenging: the full logic $\mathcal{L}$ induces hereditary history-preserving bisimilarity, which is known to be undecidable already for finite state Petri nets [JNS03]. This does not imply undecidability of the corresponding model-checking problem. However, even if it were decidable, such problem would probably lie on the very border of decidability. On the semantic side, also relaxing the restriction to strongly regular PESs appears to be quite problematic, unless one is willing to deal with transfinite objects (tableaux, automata runs, or strategies depending on the approach) which, however, would be of very limited practical interest.

As we mentioned, the game-theoretic method, devised for the solution of systems of fixpoint equations over continuous lattices, actually provides a very general framework which finds many other applications in formal verification.

On the model-checking side, besides the model-checking of standard mu-calculi, the framework applies also to cases where the mu-calculus is not a classical logic but lattice-valued as in [KL07] or real-valued as in [HK97], which presents an algorithm based on the simplex method for the non-nested case. Solving equation systems over the reals was considered in [GS11] and in [MS17]. In particular, [MS17] presents an algorithm for solving nested fixpoint equation systems over the real interval $[0, 1]_{\mathbb{R}}$ by a direct algorithm which represents and manipulates piecewise linear functions as conditioned linear expressions. The

method presented in this thesis can offer an alternative way to solve such equation systems. Games for quantitative or probabilistic mu-calculi have been studied in [MM07, Mio12]. As opposed to our game, such games closely follow the structure of the mu-calculus formula on which the game is based (e.g., $\exists$ makes a choice at an $\vee$-node, $\forall$ at an $\wedge$-node). It is an interesting question whether the conceptual simplicity of our game could lead to a new perspective on existing games. It would also be compelling to determine whether the framework can handle quantitative logics whose modalities interact with (lattice) truth values in a non-trivial way, such as logics with discounted modalities as studied in [ABK14]. Expressing such logics as systems of fixpoint equations over suitable continuous lattices and thus obtaining a game-theoretic characterisation of the corresponding model-checking problem seems reasonably easy. However, turning such characterisation into an effective technique requires some non-trivial symbolic approach due to the fact that the lattice is of infinite height.

Other interesting areas of application are fixpoint equations on the (non-distributive, but continuous) lattices of equivalence relations and pseudo-metrics. The computation of fixpoints for equivalence relations is essential for behavioural equivalences, and the same holds for pseudo-metrics and behavioural distances [vBW05].

Depending on the approach to the solution of the fixpoint game associated with a system of fixpoint equations, one can obtain a global or local decision procedure for the corresponding problem. We would like to study (efficient) algorithms for the solution of the game, especially in the local case to check whether a specific lattice element is below the solution. Examples of such local algorithms are backtracking methods studied in [SS98, Hir98]. Furthermore, we are interested in the integration of local methods with up-to techniques for general lattices, see for instance [BGGP18, PS11, Pou07]. Up-to techniques also relate in an interesting way to the method for quotienting the game described in Subsection 7.5.2. In fact, the quotienting of the game might seem an ad hoc procedure for the model-checking of $\mathcal{L}_{hp}$, but, actually, preliminary investigations reveal that it can be seen as an instance of a more general approach based on the abstraction of the domain on which the system of equations is defined, in the style of abstract interpretation, of which up-to techniques could also be seen as an instance.

Bibliography

[ABK14] Shaull Almagor, Udi Boker, and Orna Kupferman. Discounting in LTL. In *Proceedings of TACAS '14*, volume 8413 of *Lecture Notes in Computer Science*, pages 424–439. Springer, 2014.

[ACM16] Jade Alglave, Patrick Cousot, and Luc Maranget. Syntax and semantics of the weak consistency model specification language cat. arXiv:1608.07531. URL `http://arxiv.org/abs/1608.07531`, 2016.

[AHMN91] Sarita V. Adve, Mark D. Hill, Barton P. Miller, and Robert H. B. Netzer. Detecting data races on weak memory systems. In *Proceedings of the 18th Annual International Symposium on Computer Architecture*, pages 234–243. ACM, 1991.

[AJ94] Samson Abramsky and Achim Jung. Domain theory. In Samson Abramsky, Dov Gabbay, and Thomas Stephen Edward Maibaum, editors, *Handbook of Logic in Computer Science*, pages 1–168. Oxford University Press, 1994.

[AKH06] Parosh Aziz Abdulla, Lisa Kaati, and Johanna Högberg. Bisimulation minimization of tree automata. In Oscar H. Ibarra and Hsu-Chun Yen, editors, *Proceedings of CIAA '06*, volume 4094, pages 173–185. Springer, 2006.

[AMT14] Jade Alglave, Luc Maranget, and Michael Tautschnig. Herding cats: Modelling, simulation, testing, and data mining for weak memory. *ACM Trans. Program. Lang. Syst.*, 36(2):7:1–7:74, 2014.

[BC10] P. Baldan and S. Crafa. A logic for true concurrency. In P. Gastin and F. Laroussinie, editors, *Proceedings of CONCUR '10*, volume 6269 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2010.

[BC14] P. Baldan and S. Crafa. A logic for true concurrency. *Journal of the ACM*, 61(4):24:1–24:36, 2014.

[BC15] P. Baldan and A. Carraro. A causal view on non-intereference. *Fundamenta Informaticae*, 140(1):1–38, 2015.

[BDKP91] Eike Best, Raymond Devillers, Astrid Kiehn, and Lucia Pomello. Fully concurrent bisimulation. *Acta Informatica*, 28:231–261, 1991.

[Bed87] Marek A. Bednarczyk. *Categories of Asynchronous Systems*. PhD thesis, University of Sussex, 1987.

[Bed91] Marek A. Bednarczyk. Hereditary history preserving bisimulations or what is the power of the future perfect in program logics. Technical report, Polish Academy of Sciences, 1991.

[Bet55] Evert W. Beth. Semantic entailment and formal derivability. *Mededelingen der Koninklijke Nederlandse Akademie van Wetenschappen, Nieuwe Reeks*, 18(13):309–342, 1955.

[BF02] J. Bradfield and S. Fröschle. Independence-friendly modal logic and true concurrency. *Nordic Journal of Computing*, 9(1):102–117, 2002.

[BG09] Nadia Busi and Roberto Gorrieri. Structural non-interference in elementary and trace nets. *Mathematical Structures in Computer Science*, 19(6):1065—-1090, 2009.

[BGGP18] Filippo Bonchi, Pierre Ganty, Roberto Giacobazzi, and Dusko Pavlovic. Sound up-to techniques and complete abstract domains. In *Proceedings of LICS '18*, pages 175–184. ACM, 2018.

[BGV17] Eike Best, Nataliya Gribovskaya, and Irina Virbitskaite. Configuration- and residual-based transition systems for event structures with asymmetric conflict. In Bernhard Steffen, Christel Baier, Mark van den Brand, Johann Eder, Mike Hinchey, and Tiziana Margaria, editors, *Proceedings of SOFSEM '17*, volume 10139 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2017.

[BKMMP19] Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. Fixpoint games in continuous lattices. In Stephanie Weirich, editor, *Proceedings of POPL '19*, volume 3, pages 26:1–26:29. ACM, 2019.

[BP17] Paolo Baldan and Tommaso Padoan. Local model checking in a logic for true concurrency. In Javier Esparza and Andrzej S. Murawski, editors, *Proceedings of FoSSaCS '17*, volume 10203 of *Lecture Notes in Computer Science*, pages 407–423. Springer, 2017.

[BP18] Paolo Baldan and Tommaso Padoan. Automata for true concurrency properties. In Christel Baier and Ugo Del Lago, editors, *Proceedings of FoSSaCS '18*, volume 10803, pages 165–182. Springer, 2018.

[Bra98] J.C. Bradfield. The modal mu-calculus alternation hierarchy is strict. *TCS*, 195(2):133–153, 1998.

[BS06] Julian Bradfield and Colin Stirling. Modal mu-calculi. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, pages 721–756. Elsevier, 2006.

[CC79] Radhia Cousot and Patrick Cousot. Constructive versions of tarski's fixed point theorems. *Pacific Journal of Mathematics*, 82(1):43–57, 1979.

[CC17] Jérémie Chalopin and Victor Chepoi. A counterexample to Thiagarajan's conjecture on regular event structures. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *Proceedings of ICALP '17*, volume 80 of *LIPIcs*, pages 101:1–101:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[CFVZ14] F. Carreiro, A. Facchini, Y. Venema, and F. Zanasi. Weak MSO: automata and expressiveness modulo bisimilarity. In T. A. Henzinger and D. Miller, editors, *Proceedings of CSL-LICS '14*, pages 27:1–27:27. ACM press, 2014.

[CKS81] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[CKS92] Rance Cleaveland, Marion Klein, and Bernhard Steffen. Faster model checking for the modal mu-calculus. In *Proceedings of CAV '92*, volume 663 of *Lecture Notes in Computer Science*, pages 410–422. Springer, 1992.

[CKV13] I. Cristescu, J. Krivine, and D. Varacca. A compositional semantics for the reversible p-calculus. In *Proceedings of LICS '13*, pages 388–397. IEEE Computer Society, 2013.

[Cle90] Rance Cleaveland. Tableau-based model checking in the propositional mu-calculus. *Acta Informatica*, 27(8):725–747, 1990.

[CS01] E. M. Clarke and B.-H. Schlingloff. Model checking. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, chapter 21. Elsevier, 2001.

[dBS69] Jaco W. de Bakker and Dana S. Scott. A theory of programs. Unpublished manuscript, 1969.

[DDNM88] Pierpaolo Degano, Rocco De Nicola, and Ugo Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In Jaco W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX Workshop*, volume 354 of *Lecture Notes in Computer Science*, pages 438–466, Heidelberg, DE, 1988. Springer.

[DNF90] R. De Nicola and G. Ferrari. Observational logics and concurrency models. In K. V. Nori and C. E. V. Madhavan, editors, *Proceedings of FST-TCS '90*, volume 472 of *Lecture Notes in Computer Science*, pages 301–315. Springer, 1990.

[DR00] Giorgio Delzanno and Jean-François Raskin. Symbolic representation of upward-closed sets. In Susanne Graf and Michael I. Schwartzbach, editors, *Proceedings of TACAS '00*, volume 1785 of *Lecture Notes in Computer Science*, pages 426–440. Springer, 2000.

[DSB86] M. Dubois, C. Scheurich, and F. Briggs. Memory access buffering in multiprocessors. In *Proceedings of the 13th Annual International Symposium on Computer Architecture*, pages 434–442. IEEE Computer Society, 1986.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of SFCS '91*, pages 368–377. IEEE, 1991.

[EJS01] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for the μ-calculus and its fragments. *Theoretical Computer Science*, 258(1-2):491–522, 2001.

[EL86] E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proceedings of LICS '86*, pages 267–278. IEEE Computer Society, 1986.

[Fit91] Melvin Fitting. Many-valued modal logics. *Fundamenta Informaticae*, 15:235–254, 1991.

[FM06] A. Farzan and P. Madhusudan. Causal atomicity. In *Proceedings of CAV '06*, volume 4144 of *Lecture Notes in Computer Science*, pages 315–328, 2006.

[Fon08] G. Fontaine. Continuous fragment of the mu-calculus. In M. Kaminski and S. Martini, editors, *Proceedings of CSL '08*, volume 5213 of *Lecture Notes in Computer Science*, pages 139–153. Springer, 2008.

[Frö04] S. Fröschle. *Decidability and Coincidence of Equivalences for Concurrency.* PhD thesis, Univeristy of Edinburgh, 2004.

[GB09] J. Gutierrez and J. C. Bradfield. Model-checking games for fixpoint logics with partial order models. In M. Bravetti and G. Zavattaro, editors, *Proceedings of CONCUR '09*, volume 5710 of *Lecture Notes in Computer Science*, pages 354–368. Springer, 2009.

[GB11] J. Gutierrez and J. C. Bradfield. Model-checking games for fixpoint logics with partial order models. *Information and Computation*, 209(5):354–368, 2011.

[GHK+03] Gerhard Gierz, Karl H. Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael W. Mislove, and Dana S. Scott. *Continuous Lattices and Domains*. Cambridge University Press, 2003.

[GLLS05] Orna Grumberg, Martin Lange, Martin Leucker, and Sharon Shoham. Don't know in the μ-calculus. In Radhia Cousot, editor, *Proceedings of VMCAI '05*, volume 3385 of *Lecture Notes in Computer Science*, pages 233–249. Springer, 2005.

[GM82] J. A. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[GS11] Thomas Martin Gawlitza and Helmut Seidl. Solving systems of rational equations through strategy iteration. *ACM Trans. Program. Lang. Syst.*, 33(3):11:1–11:48, 2011.

[Gut09] J. Gutierrez. Logics and bisimulation games for concurrency, causality and conflict. In L. de Alfaro, editor, *Proceedings of FoSSaCS '09*, volume 5504 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2009.

[Gut11] J. Gutierrez. *On bisimulation and model-checking for concurrent systems with partial order semantics*. PhD thesis, University of Edinburgh, 2011.

[Gut15] J. Gutierrez. On fixpoint logics and equivalences for processes with restricted nondeterminism. *Journal of Logic and Computation*, 28(4):779–807, 2015.

[Hin55] Jaakko Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, 8:7–55, 1955.

[Hir98] Daniel Hirschkoff. Automatically proving up to bisimulation. In *Proceedings of MFCS '98 Workshop on Concurrency*, number 18 in Electronic Notes in Theoretical Computer Science, pages 75–89. Elsevier, 1998.

[HK97] Michael Huth and Marta Kwiatkowska. Quantitative analysis and model checking. In *Proceedings of LICS '97*, pages 111–122. IEEE, 1997.

[HKMV17] Helle Hvid Hansen, Clemens Kupke, Johannes Marti, and Yde Venema. Parity games and automata for game logic. In *Proceedings of DALI '17*, volume 10669 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2017.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[HSC16] Ichiro Hasuo, Shunsuke Shimizu, and Corina Cîrstea. Lattice-theoretic progress measures and coalgebraic model checking. In *Proceedings of POPL '16*, pages 718–732. ACM, 2016.

[JM96] Lalita Jategaonkar and Albert R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1):107–143, 1996.

[JN00] Marcin Jurdziński and Mogens Nielsen. Hereditary history preserving bisimilarity is undecidable. In Horst Reichel and Sophie Tison, editors, *Proceedings of STACS '00*, pages 358–369. Springer, 2000.

[JNS03] M. Jurdzinski, M. Nielsen, and J. Srba. Undecidability of domino games and hhp-bisimilarity. *Information and Computation*, 184(2):343–368, 2003.

[JNW96] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127(2):164–185, 1996.

[Jur00] Marcin Jurdziński. Small progress measures for solving parity games. In *Proceedings of STACS '00*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2000.

[JW96] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *Proceedings of CONCUR '96*, pages 263–277. Springer, 1996.

[Kel76] Robert M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, 1976.

[KL07] Orna Kupfermann and Yoad Lustig. Latticed simulation relations and games. In *Proceedings of ATVA '07*, volume 4672 of *Lecture Notes in Computer Science*, pages 316–330. Springer, 2007.

[Kla02] Hartmut Klauck. Algorithms for parity games. In Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors, *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500, pages 107–129. Springer, 2002.

[Kle52] Stephen Cole Kleene. *Introduction to Metamathematics*. North Holland Pub. Co., 1952.

[Kön27] Dénes König. Über eine schlussweise aus dem endlichen ins unendliche. *Acta Scientiarum Mathematicarum*, 3(2-3):121–130, 1927.

[Koz83] Dexter C. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

[Kri63] Saul Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[Mad97] Angelika Mader. *Verification of Modal Properties Using Boolean Equation Systems*. PhD thesis, TU München, 1997.

[Mad03] P. Madhusudan. Model-checking trace event structures. In *Proceedings of LICS '13*, pages 371–380. IEEE Computer Society, 2003.

[Mar76] George Markowsky. Chain-complete posets and directed sets with applications. *Algebra Universalis*, 6:53–68, 1976.

[Mil80] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[Mio12] Matteo Mio. On the equivalence of game and denotational semantics for the probabilistic μ-calculus. *Logical Methods in Computer Science*, 8(2:07):1–21, 2012.

[MM07] Annabelle McIver and Carroll Morgan. Results on the quantitative μ-calculus qmμ. *ACM Trans. Comp. Log.*, 8(1:3), 2007.

[Mos85] A. W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In Andrzej Skowron, editor, *Proceedings of Computation Theory: Fifth Symposium '84*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 1985.

[MP97] U. Montanari and M. Pistore. Minimal transition systems for history-preserving bisimulation. In R. Reischuk and M. Morvan, editors, *Proceedings of STACS '97*, volume 1200 of *Lecture Notes in Computer Science*, pages 413–425. Springer, 1997.

[MS17] Matteo Mio and Alex Simpson. Łukasiewicz μ-calculus. *Fundamenta Informaticae*, 150(3-4):317–346, 2017.

[Mul63] David E. Muller. Infinite sequences and finite machines. In *Proceedings of the 4th Annual Symposium on Switching Circuit Theory and Logical Design*, pages 3–16. IEEE Computer Society, 1963.

[NC95] M. Nielsen and C. Clausen. Games and logics for a noninterleaving bisimulation. *Nordic Journal of Computing*, 2(2):221–249, 1995.

[NNH99] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer, 1999.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part 1. *Theoretical Computer Science*, 13:85–108, 1981.

[Pad] Tommaso Padoan. True concurrency workbench. Available at `http://github.com/tpadoan/TCWB`.

[Pad18] Tommaso Padoan. Relating some logics for true concurrency. In Alessandro Aldini and Marco Bernardo, editors, *Proceedings of ICTCS '18*, volume 2243 of *CEUR Workshop Proceedings*, pages 242–253. CEUR-WS.org, 2018.

[Par69] David Park. Fixpoint induction and proofs of program properties. In Bernard Meltzer and Donald Michie, editors, *Machine Intelligence*, volume 5, pages 59–78. Edinburgh Univeristy Press, 1969.

[Par81] David Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[Pen95] W. Penczek. Branching time and partial order in temporal logics. In *Time and Logic: A Computational Approach*, pages 179–228. UCL Press, 1995.

[Pen97] W. Penczek. Model-checking for a subclass of event structures. In E. Brinksma, editor, *Proceedings of TACAS '97*, volume 1217 of *Lecture Notes in Computer Science*, pages 145–164. Springer, 1997.

[Pet62] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Universität Hamburg, 1962.

[Pou07] Damien Pous. Complete lattices and up-to techniques. In *Proceedings of APLAS '07*, pages 351–366. Springer, 2007. LNCS 4807.

[PS11] Damien Pous and Davide Sangiorgi. Enhancements of the bisimulation proof method. In Davide Sangiorgi and Jan Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*. Cambridge University Press, 2011.

[PU07] I. Phillips and I. Ulidowski. Reversing algebraic process calculi. *Journal of Logic and Algebraic Programming*, 73(1-2):70–96, 2007.

[PU11] Iain Phillips and Irek Ulidowski. A logic with reverse modalities for history-preserving bisimulations. In B. Luttik and F. Valencia, editors, *Proceedings of EXPRESS '11*, volume 64 of *EPTCS*, pages 104–118, 2011.

[PU12] I. Phillips and I. Ulidowski. A hierarchy of reverse bisimulations on stable configuration structures. *Mathematical Structures in Computer Science*, 22(2):333–372, 2012.

[PU14] Iain Phillips and Irek Ulidowski. Event identifier logic. *Mathematical Structures in Computer Science*, 24(2):1–51, 2014.

[Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[Ran52] George. N. Raney. Completely distributive complete lattices. *Proceedings of the AMS*, 3(5):677–680, 1952.

[RT88] Alexander M. Rabinovich and Boris A. Trakhtenbrot. Behaviour structures and nets. *Fundamenta Informaticae*, 11:357–404, 1988.

[Sch04] Klaus Schneider. *Verification of Reactive Systems: Formal Methods and Algorithms*. Springer, 2004.

[Sco70] Dana S. Scott. Outline of a mathematical theory of computation. In *Proceedings of the 4th Annual Princeton Conference on Information Sciences and Systems*, pages 169–176, 1970.

[Sco71] Dana S. Scott. The lattice of flow diagrams. In E. Engeler, editor, *Symposium on Semantics of Algorithmic Languages*, volume 188 of *Lecture Notes in Mathematics*, pages 311–366. Springer, 1971.

[Sco72] Dana S. Scott. Continuous lattices. In F. W. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer, 1972.

[Sei96] Helmut Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59(6):303–308, 1996.

[Shi85] M W. Shields. Concurrent machines. *The Computer Journal*, 28(5):449–465, 1985.

[SS98] Perdita Stevens and Colin Stirling. Practical model-checking using games. In Bernhard Steffen, editor, *Proceedings of TACAS '98*, volume 1384 of *Lecture Notes in Computer Science*, pages 85–101. Springer, 1998.

[Sti95] Colin Stirling. Local model checking games. In *Proceedings of CONCUR '95*, volume 962 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1995.

[Str82] Robert S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54(1):121–141, 1982.

[SW91] C. Stirling and D. Walker. Local model checking in the modal mu-calculus. *Theoretical Computer Science*, 89(1):161–177, 1991.

[Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[Thi02] P. S. Thiagarajan. Regular event structures and finite Petri nets: A conjecture. In W. Brauer, H. Ehrig, J. Karhumäki, and A. Salomaa, editors, *Formal and Natural Computing - Essays Dedicated to Grzegorz Rozenberg [on occasion of his 60th birthday]*, volume 2300 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2002.

[Val98] Antti Valmari. *The state explosion problem*, volume 1491 of *Lecture Notes in Computer Science*, pages 429–528. Springer, Berlin, Heidelberg, 1998.

[vBW05] Franck van Breugel and James Worrell. A behavioural pseudometric for probabilistic transition systems. *Theoretical Computer Science*, 331:115–142, 2005.

[Ven08] Yde Venema. Lectures on the modal μ-calculus. Lecture notes, Institute for Logic, Language and Computation, University of Amsterdam, 2008.

[vGG01] R.J. van Glabbeek and U. Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37(4/5):229–327, 2001.

[Vog91] W. Vogler. Deciding history preserving bisimilarity. In J. Albert, B. Monien, and M. Artalejo, editors, *Proceedings of ICALP '91*, volume 510 of *Lecture Notes in Computer Science*, pages 495–505. Springer, 1991.

[Win87] G. Winskel. Event Structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1987.

[WN95] G. Winskel and M. Nielsen. Models for concurrency. In S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of logic in Computer Science*, volume 4, pages 1–148. Oxford University Press, 1995.

[Wol90] Thomas Wolfgang. Automata on infinite objects. In Jan Van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, pages 133–191. Elsevier, 1990.

[Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.