Games and Logical Expressiveness

von der fakultät für mathematik, informatik und naturwissenschaften der rheinisch-westfälischen technischen hochschule aachen zur erlangung des akademischen grades eines doktors der naturwissenschaften genehmigte dissertation

vorgelegt von

Diplom-Informatiker Dietmar Berwanger

aus lugoj, rumänien

berichter

Universitätsprofessor Dr. Erich Grädel
Dr. Igor Walukiewicz

tag der mündlichen prüfung

12. Mai 2005

aachen ♦ Mai 2005

Preface

Game theory is a framework of analytical tools for reasoning about decisions under circumstances beyond the immediate control of the individual decision maker. The foundations of modern game theory have been laid by von Neumann and Morgenstern in the first half of the 20th century [74] at the intersection between mathematics and economy. Since then, game theory has undergone an outstanding evolution, transgressing the boundaries of its parent disciplines, and reaching core positions in every scientific area where interaction and decision matter.

The effectiveness of game theory can be ascribed to two main concerns. As a means for *description*, games represent a unified model for interactive situations which abstracts from the contingencies of the specific environment. On the background of this model, game theory offers an extensive and intuitive language for reasoning about the intricacies of interactions. In return, the insight gained within the model supports decisions on subsequent actions, thus assigning the theory a *prescriptive* competence.

To accomplish these functions, game theory shall establish a provision of viable *models* and *languages* that are able to capture the relevant distinctions and similarities in the concrete setting, while remaining operative, on the other hand. As pointed out by Aumann in [3], game theory is in this sense a science of *classification*. In view of its aims and methods, game theory is closely related to logic.[1] On the common ground of the two sciences, logic has to offer a rich foundational framework of formal languages and models.

Conversely, game-theoretic techniques turned out to provide a fruitful approach to several essential issues in logic [35]. A moment of major impact on logical methodology is marked by the understanding of quantifiers in terms of games, proposed by Henkin [31] and consolidated by Hintikka [34]. In this view, the value of quantified first-order variables is assigned by two antagonistic players, Verifier and Falsifier, reflecting the intuition that Verifier tries to make a formula true under the challenge of Falsifier's choices. The notions of truth and provability can thus be phrased in terms of winning strategies for Verifier, providing a game-theoretic semantics for predicate logic which extends naturally to logics with generalised quantifiers.

Another fundamental result at the boundary between games and logics is

[1] Interestingly, the first formal theorem in game theory was contributed by a logician: in 1913 Zermelo proved that chess is determined [79].

the characterisation of elementary equivalence in term of Ehrenfeucht-Fraïssé games [18]. Here, the players are concerned with the question whether two structures are distinguishable by means of a first-order formulae. If so, a winning strategy corresponds to a separating formula. Structure-comparison games of this kind provide an invaluable model-theoretic tool and have been successfully adapted to a large range of logics.

The theory of concurrent systems in computer science is a prominent application area of game theory and logic. In this framework, the task of decision making is conveyed to computational agents, equipped with a formal specification of their objective. The challenge is to design these agents *as if* they would be rational in the original game-theoretic sense, i.e., acting deliberately towards achieving their individual objective taking into account all possible actions of the other agents.

A common interpretation of concurrent systems is based on Kripke structures, also called transition systems. In this model, the elements are associated to states of the system, and binary transition relations represent actions that can be performed on or by the system. Due to the occurrence of actions, the system evolves along transitions, forming a path through the model. In an interactive setting, an individual agent may have control only over a restricted set of states so that the system's behaviour, i.e., the formed path, also depends on the actions of other agents. The question is whether and how an agent can ensure the system to behave according to his objective.

In line with this interpretation, formal languages are designed to describe the possible behaviours of concurrent systems and to specify the objectives of agents. Accordingly, questions about the properties of a system can be translated into questions about satisfaction or validity of logical formulae in Kripke structures. Immediate applications of this approach arise, for instance, in control theory, where we have two agents, the controller and the environment, and wish to specify that the controller can keep the system reliable under interference with the environment. More generally, Kripke structures together with formal specifications of objectives can be regarded as a a model for a large variety of interactive situations based on discrete states and evolving sequentially over time. This allows us to rely on established logical methods towards reasoning about games.

Conversely, logics over Kripke structures can be naturally embedded into the realm of game theory, by way of the appropriate game-theoretic semantics. This two-way correspondence between the classical and formal framework of modal logics, i.e., logics intended for reasoning about Kripke structures, on the one hand,

and the intuitive world of game theory on the other hand, is of great potential.

In order to take advantage of this potential we shall, of course, not persist at the definitional level. Unfortunately, regarding aspects of internal structure, the gap between classical formal logics and game-oriented languages is very large. For instance, already the notion of equivalence, which is fundamental for classical logics, is far from being well-understood in terms of games [69]. Also, concepts of rationality intrinsic to the game perspective are often very hard to capture in terms of classical formalisms.

In the present thesis, we address the question of relating classical formal logics and game logics with regard to their fine-structure, and try to bridge the gap between these in a specific setting concerning two-player games over Kripke structures. On the classical side, we consider the μ-calculus L_μ, a very expressive and robust formalism with outstanding model-theoretic properties which, however, is widely agreed to be little intuitive. On the other side, we investigate formalisms with generalised quantifiers defined via games arising naturally in the context of concurrent systems: Parikh's Game Logic GL and so-called path-game logics.

Parikh's Game Logic, discussed in Chapter 2, is a dynamic formalism with quantifiers ranging over games built by sequential composition, nondeterministic choice, iteration and game dualisation, starting from a given set of atomic actions. We show that the resulting language is very powerful, being able to express the semantic games of the μ-calculus. Further, we prove that the syntactic device of dualisation induces a strict semantic hierarchy which parallels the alternation hierarchy of the μ-calculus.

Chapter 3 is dedicated to temporal logics with quantifiers associated to infinite paths formed interactively by the players. According to the schedule of the forming process and the conditions on the outcoming path, we first classify the underlying families of games under topological viewpoint. Then, we study the structure of winning strategies in regular games, and show that, in significant cases, these are automatic or even memoryless. On basis of this, we embed the corresponding temporal logics into the μ-calculus and show that, if first-order logic is used to describe the quantified paths, the obtained language is equiexpressive to a well-studied formalism, namely CTL*.

In the last two chapters we investigate the fine-structure of the μ-calculus with respect to the question of how many variables are needed to specify a given property. The question arises naturally in the context of GL, CTL*, and other process languages, which all turn out to translate into the two-variable fragment

of L_μ. Our approach towards settling this question relies essentially on games. We consider model-comparison games, more precisely, simulation games over finite Kripke structures, which can be described in the μ-calculus.

In Chapter 4, we first identify a structural parameter, called entanglement, representing an upper bound on the number of variables needed to define a given finite Kripke structure, or, the (bi-)simulation game for that structure, in L_μ. Besides being a measure of descriptive complexity, it turns out that the entanglement also captures relevant computational properties of the structure. We show that parity games can be solved in polynomial time, if their entanglement is bounded by a constant. This is significant since no polynomial time algorithm for solving this problem in the general case is known (although there are no strong reasons to assume none exists).

Finally in Chapter 5 we show that the entanglement of a structure represents indeed a lower bound on the number of variables needed to describe the corresponding simulation game. As a consequence, it follows that the variable hierarchy of the μ-calculus is strict. In particular, this result separates the expressive power of GL from L_μ answering an open question posed by Parikh when he introduced Game Logic in 1983 [55].

Acknowledgement. I would like to express my deep gratitute to my dear friend and supervisor, Erich Grädel, for being close and caring through all this years. I thank Igor Walukiewicz and David Janin heartily for reviewing my thesis and for being wonderful hosts and great guests. I also wish to thank Anuj Dawar and Wolfgang Thomas for their support and encouragement.

Love to my parents and my brother, and to Anne, Laura, Tessa, Christof, Gabi, Ghiţi, and Sebastian, who make the world be a wonderful place. To my co-authors Achim, Giacomo, and Stephan: thanks for sharing. Antje, Aliaa & Eric, Elisabeth, and Henrik were my office mates in Aachen, I couldn't have dreamed of a better company. Colin, Jens, Jacques, Łukasz, Madhu, Marc, Marcin, Olivier, Philipp, Stefan, Thomas, and Vince – thanks for the shiny memories.

I dedicate this thesis to Andreea, sine qua non. . .

Contents

1 Background

The formal apparatus of this thesis is founded on state-based models of two-player games, and logical languages for reasoning about these models. In the present chapter, we review some fundamental game-theoretic concepts which we relate to specific questions in logic and computation. Departing from these, we introduce Kripke structures as a raw model for concurrent behaviour together with a criterion of observational equivalence captured by the notion of bisimulation. The second section is dedicated to logical formalisms for describing the behaviour of Kripke structures. In the last section we introduce the semantic games for the μ-calculus.

1.1 Models for interaction

Game models describe situations of strategic interaction representing the actions or decisions players can take and their preferences over the possible outcomes arising as a consequence of these. To solve a game means to describe the possible outcomes that may arise when players proceed rationally.

In the context of concurrent systems, we use Kripke structures to model the dynamics of a system and logical specification languages to describe the player's interest, which we assume to be conflicting. In game-theoretic terms, this corresponds to extensive zero-sum games of perfect information. The case when the outcome of such a game can be determined beforehand, is of particular interest, especially in view of the implementation of good strategies for computational agents.

1.1.1 Game models

The most simple and abstract model of a game represents only the actions available to the players and the utility they derive from the outcome of a play, depending on

the actions they choose.

Definition 1.1.1 (Game in strategic form). A game in *strategic form* for n players is represented by a tuple $\big((\textsc{Act}_i)_{i<n}, (u_i)_{i<n}\big)$, where, for each player $i < n$,

- $\textsc{Act}_i$ is a nonempty set of *actions*, and
- $u_i : \bigtimes_{i<n} \textsc{Act}_i \to \mathbb{R}$ is a *utility* function.

In this perspective, a game has no internal structure. All that players can do is to choose an action. A strategy of a player in a strategic game is any action available to him. An *outcome* of such a game, or simply a *play*, is a tuple of actions, one for each player: $(a_0, \ldots, a_{n-1}) \in \bigtimes_{i<n} \textsc{Act}_i$.The utility u_i reflects how much the given outcome is worth to player i.

In game-theoretic literature, utility is often assigned to a set of *consequences* of outcomes rather than to the outcomes themselves. Formally, this means to extend the model by a set C of consequences to which outcomes are mapped via a function $c : \bigtimes_{i<n} \textsc{Act}_i \to C$, and to define utilities in terms of outcomes: $u_i : C \to \mathbb{R}$. By detaching actions from immediate utility, this allows a higher level of abstraction. In our discussion of Game Logic in Chapter 2 we will refer to a further abstraction of strategic games, called *game forms*, in which utility is completely discarded.

Despite its simplicity, the model of a strategic game already displays two essential ingredients of a game. It captures *which* decisions the players can take, and, by specifying their utility, a determination of *why* they may choose one or another action.

However, any details about what else players may know about each other are obscured. In practice, decisions are not encountered by all players once and simultaneously. The course of a play is often rather sequential and players may be more or less aware of events that have previously occurred. These aspects are captured by games of perfect information in extensive form.

Such a game is adequately represented as a tree, i.e., an directed connected acyclic graph, where the nodes are coloured to indicate which player is in turn to move. We will use a partitioning of the set T of tree nodes into disjoint subsets T_i, one for each player, to represent this colouring. By a *full path* in a tree we mean a maximal path starting from the root. Such a path may be either finite, in which case it terminates at a leaf, or infinite.

Definition 1.1.2 (Game in extensive form). A game in *extensive form* for n players is represented by a tuple $\big(\mathcal{T}, (T_i)_{i<n}, (u_i)_{i<n}\big)$, where $\mathcal{T}$ is a tree, called *history*, with

nodes associated to players, and, for each player, the utility function u_i associates to any full path in $\mathcal{T}$ a real number.

Extensive games are played in turns: At the beginning of a play, we are at the root of the history $\mathcal{T}$. Whenever the current node is in T_i, player i can choose a successor from which the play continues. If the current node has no successors, the play ends, otherwise it goes on infinitely. In either case the utility is determined, for each player, by the utility of the path formed during the play. In this way, an *outcome* of an extensive game is just a full path through the history. In the same way as in strategic games, we do not distinguish between plays and their outcome.

A strategy for a certain player in an extensive game is a plan that tells him how to choose at every moment at which he is in turn. In that event, his decision may depend on the previous history.

Definition 1.1.3 (Strategy). Given an extensive game $\left(\mathcal{T}, (T_i)_{i<n}, (u_i)_{i<n}\right)$, a *strategy* for player i is a function $f : T_i \to T$ associating to every history node v where player i moves, some successor in $\mathcal{T}$. We say that a play $v_0, v_1, \ldots$ of the game is *according* to the strategy f, if for every $v_j \in T_i$, we have $v_{j+1} = f(i)$.

Observe that the notion of utility introduced here allows to model situations of either conflicting or common interests, or a mix of the two. However, the games considered in logic and computer science usually focus on the analysis of conflict. Moreover, in the general setting, only two antagonistic players are involved. For the remainder of this thesis we will therefore restrict to 2-player games of a simplified structure.

Definition 1.1.4 (Win-or-lose game). A *win-or-lose* game is a 2-player game where, for each play, precisely one of the players has utility 1 and the other one 0. An play is winning for a player, if his utility for the play is 1.

Henceforth, whenever we refer to a game, we mean a win-or-lose game. Obviously, in such games, it is sufficient to specify the utility for one player. To simplify notation, for a pair (f, g) of strategies for the two players in a game, we write $f \hat{} g$ to denote the unique play according to the two strategies.

Definition 1.1.5 (Winning strategy). A strategy f for a player in a given game is *winning* if he wins each play $f \hat{} g$, for every strategy g of his opponent.

A central issue in the context of purely competitive, win-or-lose games between two players is whether one of the players can win regardless of the actions of his opponent. If this is the case, there is a clear concept of a solution to this game.

Definition 1.1.6 (Determinacy). A win-or-lose game is *determined* if either one or the other player has a winning strategy.

Throughout this thesis, we will usually work with games given in extensive form. However, rather than specifying all ingredients explicitly, we will give a less formal description of the dynamic of the game from which the history tree can be easily reconstructed. Also, instead of referring to utility functions, we will rather speak about *winning conditions* representing the set of infinite paths where a certain player, usually Player 0, wins. As a general convention for two-player games, we assume that the player who has to move at a history node with no successors loses.

1.1.2 Kripke structures

Our concern with decision and interaction relies upon the assumption that, as a consequence of action, the world is changing. An adequate model for representing dynamically changing systems is provided by Kripke structures. This prospective relates the system to a set of states, presenting the possible transitions from one state to another as being triggered by actions. The particularities of each state are recorded as atomic propositions. In view of this, Kripke structures are also called *(labelled) transition systems*.

Definition 1.1.7 (Kripke structure). A *Kripke structure* over a set Act of actions and a set Prop of atomic propositions is a structure

$$\mathcal{K} = \big(V,\ (E_a)_{a\in \mathrm{Act}},\ (V_p)_{p\in \mathrm{Prop}}\big),$$

with a domain V of elements called *states*, binary *transition* relations $E_a \subseteq V \times V$ associated to the actions $a \in \mathrm{Act}$, and monadic relations $V_p \subseteq V$, associated to the atomic propositions $p \in \mathrm{Prop}$.

We will sometimes abstract from propositional and action labels and refer to the graph $(V, \bigcup \{ E_a \mid a \in \mathrm{Act} \})$ as the graph *underlying* to $\mathcal{K}$. Conversely, we say that $\mathcal{K}$ is a Kripke structure over this graph. It is appropriate to consider *rooted* structures $\mathcal{K}, u$, where all states are reachable from the distinguished state u in the underlying graph. To refer to the set of successors of a state v via a binary relation E, we use the notation $vE := \{ w \mid (v, w) \in E \}$. Unless otherwise stated, we will assume that the constituents of a Kripke structure are always named as in this definition.

In the context of computer science, Kripke structures are used as a fundamental model for describing the behaviour of *reactive systems*, involving programs that maintain an ongoing interaction with the environment, such as communication protocols or resource schedulers in operating systems. In such a model, the program continuously performs actions in response to the requests of the environment, which are themselves represented as actions. Accordingly, reactive systems can be viewed as two-player extensive game forms. Conversely, if we abstract from the aspect of utility, extensive games can be naturally described by Kripke structures.

Bisimulation and simulation

An issue of central importance regarding Kripke structures in general, and reactive systems in particular, is whether two systems display the same behaviour. The idea of behavioural equivalence, is captured by the notion of bisimulation introduced by Hennessy and Milner [32].

Definition 1.1.8 (Bisimulation). A *bisimulation* between two Kripke structures

$$\mathcal{K} = \big(V,\ (E_a)_{a\in \textsc{Act}},\ (V_p)_{p\in \textsc{Prop}}\big) \quad \text{and} \quad \mathcal{K}' = \big(V',\ (E'_a)_{a\in \textsc{Act}},\ (V'_p)_{p\in \textsc{Prop}}\big)$$

is a non-empty relation $Z \subseteq V \times V'$ that respects the atomic propositions $p \in \textsc{Prop}$, in the sense that $v \in V_p$ if, and only if, $v' \in V'_p$, for all $(v, v') \in Z$, and satisfies the following forth and back conditions.

forth: for all $(v, v') \in Z$, $a \in \textsc{act}$, and every $w \in vE_a$, there exists a state $w' \in vE'_a$ such that $(w, w') \in Z$.

back: for all $(v, v') \in Z$, $a \in \textsc{act}$ and every w' such that $w' \in v'E'_a$, there exists a state $w \in E_a$ such that $(w, w') \in Z$.

We say that two rooted Kripke structure $\mathcal{K}, u$ and $\mathcal{K}', u'$ are *bisimilar*, and write $\mathcal{K}, u \sim \mathcal{K}', u'$, if there exists a bisimulation Z between them with $(u, u') \in Z$.

The concept of bisimulation can be easily understood as a game in which two players, called Challenger and Duplicator, compare the structures by moving two pebbles, one for each structures. This game is described as follows: at the beginning, the pebbles are at u and u'. If the atomic type of the pebbled nodes differs, i.e., if there is an atomic proposition p satisfied by either u or u' but not by both, then Challenger immediately wins. Otherwise, he chooses one of the structures and moves the pebble to a successor of its current position along some action

$a \in \text{Act}$. In turn, Duplicator has to move the pebble in the other structure along the same action. If an agent gets stuck he loses. Otherwise the game goes on forever. Duplicator wins, if he never loses.

It is straightforward to show that every winning strategy in such a bisimulation game can be turned into a bisimulation relation, and vice versa.

Lemma 1.1.9. *Duplicator wins the bisimulation game between $\mathcal{K}, u$ and $\mathcal{K}, u'$ if, and only if, $\mathcal{K}, u \sim \mathcal{K}', u'$.*

A normalised form for the behaviour of a system modelled by a Kripke structure is given by its unravelling. Intuitively, this is the tree consisting of all paths through the structure that start at the initial state.

Definition 1.1.10 (Unravelling). The *unravelling* of a Kripke structure

$$\mathcal{K} = \big(V,\ (E_a)_{a\in\text{Act}},\ (V_p)_{p\in\text{Prop}}\big)$$

from a node $u \in V$ is a Kripke structure $\mathcal{T}$ over a tree, so that

- the domain $V^{\mathcal{T}}$ of $\mathcal{T}$ is the set of all sequences $\pi := v_0a_1v_1a_2 \cdots v_{r-1}a_rv_r$ with $v_i \in V$ and $a_i \in \text{act}$, such that $v_0 = u$ and $v_i \in v_{i-1}E_{a_i}$;
- for every atomic proposition $p \in \text{Prop}$, we have $v_0a_1v_1a_2 \dots v_{r-1}a_rv_r \in V_p^{\mathcal{T}}$ if, and only if, $v_r \in V_p^{\mathcal{K}}$;
- for all actions a, the transition $E_a^{\mathcal{T}}$ contains the pairs $(\pi, \pi av)$ in $V^{\mathcal{T}} \times V^{\mathcal{T}}$.

Obviously, the natural projection $\mathcal{T}, u \mapsto \mathcal{K}, u$ which sends every sequence $\pi = v_0a_1v_1a_2 \dots v_{r-1}a_rv_r \in V^{\mathcal{T}}$ to its last node v_r defines a bisimulation between $\mathcal{T}$ and $\mathcal{K}, u$.

Besides behavioural equivalence, we are sometimes interested in the question whether a system is able to reproduce the behaviour of another system.

Definition 1.1.11 (Simulation). A *simulation* of a structure $\mathcal{K}$ by a structure $\mathcal{K}'$ is a non-empty relation $Z \subseteq K \times K'$ that respects the atomic propositions $p \in \text{Prop}$, in the sense that $v \in V_p$ if, and only if, $v' \in V'_p$, for all $(v, v') \in Z$, and satisfies the following condition.

forth: for all $(v, v') \in Z$, $a \in \text{act}$, and every $w \in vE_a$, there exists a state $w' \in vE'_a$ such that $(w, w') \in Z$.

We say that $\mathcal{K}', u'$ *simulates* $\mathcal{K}, u$ and write $\mathcal{K}, u \lesssim \mathcal{K}', u'$, if there is a simulation from $\mathcal{K}$ to $\mathcal{K}'$ that contains (u, u').

The bisimulation game described above 1.1.8 can easily be adapted to capture the concept of simulation. It suffices to require the Challenger to always move the token in $\mathcal{K}$, so that Duplicator must move in $\mathcal{K}'$.

1.2 Logical specification formalisms

In this section, we review some general formalisms for reasoning about interactive systems represented as Kripke structures. Our focus lies on specification logics i.e., languages in which we express statements about what the system should do. Among the various formalism developed for this purpose, we consider representatives of three main groups: dynamic logics, oriented towards the live behaviour of a system, temporal logics, representing either an a priori or an a posteriori view on the system's execution, and modal logics, designed for reasoning about Kripke structures in general.

In order to compare logics with regard to their expressive power, we need to relate formulae of different formalism. Thus, we say that a formula φ from a logic $\mathcal{L}$ is *equivalent* to a formula φ' from a possibly different logic $\mathcal{L}'$, if the two formulae have the same models. Accordingly, we write $\mathcal{L} \leq \mathcal{L}'$, if for every formula φ in $\mathcal{L}$ there exist an equivalent formula $\varphi' \in \mathcal{L}'$. Further, we write $\mathcal{L} \equiv \mathcal{L}'$ if both $\mathcal{L} \leq \mathcal{L}'$ and $\mathcal{L}' \leq \mathcal{L}$ and $\mathcal{L} < \mathcal{L}'$ if $\mathcal{L} \leq \mathcal{L}'$, but $\mathcal{L} \not\equiv \mathcal{L}'$.

For extensive surveys on the matter of temporal and dynamic logics, and a more gentle introduction to the μ-calculus, we refer the reader to [19] and [14].

1.2.1 Predicate logics

First-order predicate logic FO provides a frame of reference for any investigation about logical formalism. Nevertheless, when speaking about Kripke structures, FO is rather inappropriate for several reasons. On the one hand, the language is too weak to express properties in which we are naturally interested, e.g., whether from a given initial state a certain target state is reachable. On the other hand, FO is too complex from a computational point of view, since its satisfiability problem is undecidable. Finally, when used to describe behavioural properties of systems modelled by Kripke structures, the language is over-expressive, in a certain sense, as it is able to distinguish between bisimilar states.

Monadic second-order logic MSO, is an extension of first-order logic which allows quantification over monadic variables ranging over sets of elements rather than individual elements. For a Kripke structure $\mathcal{K}$, we may thus formulate a property $\varphi(Z)$ involving a new propositional predicate $Z \notin \textsc{Prop}$ and express by $\exists Z.\varphi(Z)$ that φ holds in some expansion of $\mathcal{K}$. Intuitively, MSO has the ability to "guess" propositional predicates in addition to those provided by the structure. This gives rise to a very powerful language, able to express most of the relevant properties of Kripke structures.

On the other hand, MSO is characterised by a very tight connection to the theory of automata, which we will discuss later. Via this connection, Büchi proved that the monadic theory of one successor S1S, i.e., of MSO interpreted over infinite words, or paths, is decidable [17]. Generalising Büchi's automata model, Rabin proved the analogon of this result for MSO interpreted over the infinite binary tree, showing that the monadic theory of two successors *S2S* is also decidable [61]. Further extensions of this decidability result to arbitrary trees, due to Le Tourneau [48] and Shelah [64], and iterated structures, due to Semenov [62] and Walukiewicz [76], distinguish MSO as a milestone at the frontier of expressiveness and computability.

Throughout this thesis we will also refer to a sublogic of MSO, called *monadic path logic* MPL, where the interpretation of monadic variables is restricted to range not over arbitrary sets, but over infinite paths. When reasoning about sequential processes, this turns out to be a natural quantification pattern allowing us to express many interesting properties of Kripke structures in a succinct way.

To cope with the excess of expressiveness of FO and its extensions, we may restrict ourselves to formulae which respect the notion of behavioural equivalence.

Definition 1.2.1. A formula φ of a given logic $\mathcal{L}$ is *bisimulation-invariant*, if it does not distinguish between bisimilar structures. That is, for every pair $\mathcal{K}, u \sim \mathcal{K}', u'$, we have $\mathcal{K}, u \vDash \varphi$ if, and only if, $\mathcal{K}', u' \vDash \varphi$. We denote by $\mathcal{L}/\!\sim$ the fragment of $\mathcal{L}$ consisting of all bisimulation-invariant formulae.

Unfortunately, relying on bisimulation-invariant fragments of FO, MPL, or MSO is not a viable solution. As van Benthem points out in [71], it is already undecidable whether a given FO-formula is bisimulation invariant or not. For this reason, predicate logics are not directly used as specification languages for systems modelled by Kripke structures. However, they represent yardsticks for measuring the expressive power of more specialised formalisms, like those we discuss in the following.

1.2.2 Modal and dynamic logics

Modal logics are languages for reasoning about dynamic aspects of truth. Although preoccupations with modal logics date back to ancient times, a rigorous semantics was developed only in the last century reaching its current form with Kripke [46]. Therefore, Kripke structures are inherently connected with modal logics.

Propositional modal logic extends propositional logic with a *modal* operator $\Diamond$. Modal formulae φ are interpreted at states in Kripke structures. Intuitively, $\Diamond\varphi$ asserts that from the current state, a state where φ holds is directly reachable. Dually, the expression $\neg\Diamond(\neg\varphi)$, denoted as $\Box\varphi$, asserts that, at every successor state, φ is true.

Hennessy-Milner Logic

In the original setting, transitions in a Kripke structure reflected whether a state is accessible from another state or not. Consequently, this model features only one transition relation, also called reachability relation. However, when modelling systems that shift from one state into another in response to the occurrence of certain actions, it is appropriate to associate modalities to individual actions and to use polymodal logics for reasoning about them. We consider here the polymodal logic underlying Hennessy and Milner's seminal study [32] on the behavioural equivalence of interactive systems.

Definition 1.2.2 (Syntax of Hennessy-Milner Logic)**.** Given a set Act of actions and a set Prop of atomic propositions, *Hennessy-Milner* logic ML consists of the formulae constructed according to the following rules:

$$\varphi := \bot \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle a\rangle\varphi$$

where $p \in \text{Prop}$ and $a \in \text{Act}$.

To define the meaning of a formula φ in a given Kripke structure $\mathcal{K}$, we describe its *extension* $[\![\varphi]\!]_{\mathcal{K}}$, that is, the set of worlds in $\mathcal{K}$ where φ holds. To simplify notation, we may omit the subscript, when clear from the context. Alternatively we will also refer to the satisfaction relation $\mathcal{K}, u \models \varphi$ defined by $u \in [\![\varphi]\!]_{\mathcal{K}}$.

Definition 1.2.3 (Semantics of Hennessy-Milner Logic)**.** Given a Kripke structure

$$\mathcal{K} = (V, (E_a)_{a\in\text{Act}}, (V_p)_{p\in\text{Prop}}),$$

the extension of ML-formulae over Act and Prop is defined inductively as follows:

$$\begin{aligned}
[\![\bot]\!] &:= \varnothing;\\
[\![p]\!] &:= V_p;\\
[\![\neg\varphi]\!] &:= V \smallsetminus [\![\varphi]\!];\\
[\![\varphi_1 \vee \varphi_2]\!] &:= [\![\varphi_1]\!] \cup [\![\varphi_2]\!];\\
[\![\langle a\rangle\varphi]\!] &:= \{\, v \mid (\exists w \in vE_a)\ w \in [\![\varphi]\!] \,\}.
\end{aligned}$$

Dual connectives are introduced as a shorthand:

$$\top := \neg\bot \qquad \varphi_1 \wedge \varphi_2 := \neg(\neg\varphi_1 \vee \neg\varphi_2); \qquad [a]\varphi := \langle a\rangle\varphi.$$

When Act consists only of one action a, we will simply write $\Diamond$ and $\Box$ instead of $\langle a\rangle$ and $[a]$.

Hennessy-Milner Logic provides a logical characterisation of bisimulation over *finitely branching* Kripke structures, where for every state $v \in V$ and every action $a \in$ Act, the set vE_a is finite.

Theorem 1.2.4 ([32]). *For any pair of finitely branching Kripke structures $\mathcal{K}, u$ and $\mathcal{K}', u'$, we have:*

$$\mathcal{K}, u \sim \mathcal{K}', u' \quad \text{iff} \quad \{\, \varphi \in \mathrm{ML} \mid \mathcal{K}, u \vDash \varphi \,\} = \{\, \varphi \in \mathrm{ML} \mid \mathcal{K}', u' \vDash \varphi \,\}.$$

According to this, ML-formulae are in particular invariant under bisimulation. Also, they can be easily translated into FO. The Modal Characterisation Theorem of van Benthem states that, conversely, every bisimulation-invariant FO-formula can be translated into ML. In other words, ML provides an effective syntax for first-order properties that are invariant under bisimulation.

Theorem 1.2.5 ([70]). FO/$\sim$ $\equiv$ ML : *An* FO*-formula is bisimulation invariant if, and only if, it is equivalent to an* ML*-formula.*

Since ML corresponds to a fragment of FO, its expressive power does certainly not suffice to describe relevant properties of reactive systems. Nevertheless, it provides a robust foundation for highly expressive formalism. Moreover, as pointed out by Vardi [73] and Grädel [25], the modal quantification pattern guarantees good algorithmic properties, even when the base logic is extended by more powerful operators.

Propositional Dynamic Logic

Propositional Dynamic Logic was first introduced by Fischer and Ladner in [22] for reasoning about the dynamic behaviour of nondeterministic programs. Syntactically, this formalism extends Hennessy-Milner Logic by associating modalities to programs built up from actions and tests.

Definition 1.2.6 (Syntax of PDL). Given a set Prop of atomic propositions and a set Act of atomic actions, the expressions of PDL are of two sorts, formulae and programs, generated respectively by the following grammar:

$$\varphi := \bot \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\rho\rangle\varphi$$
$$\rho := a \mid \varphi? \mid \rho;\rho \mid \rho \cup \rho \mid \rho^*$$

where $p \in$ Prop and $a \in$ Act.

Intuitively, the program construction $\rho_1;\rho_2$ stands for sequential composition: "execute ρ_1 followed by ρ_2". The nondeterministic choice operator $\rho_1 \cup \rho_2$ means: "choose nondeterministically ρ_1 or ρ_2 and execute it". The iteration operator ρ^* invests the language with a notion of unbounded recursion. It is interpreted as follows: "execute ρ any nondeterministically chosen finite number of times (zero or more)". Finally, the test operator $\varphi?$ corresponds to the scenario "check whether φ holds. If so, proceed, otherwise fail."

Before proceeding to the formal definition of semantics, let us introduce some notation for handling binary relations. For the composition of two relations, we write $E_1 \circ E_2 := \{ (v, w) \mid (\exists u \in vE_1)\, w \in uE_2 \}$. The reflexive transitive closure of a relation is $E^* := \bigcup_{i<\omega} E^i$ with E^0 corresponding to the identity over the relation's domain and $E^{i+1} := E \circ E^i$, for all i.

Definition 1.2.7 (Semantics of PDL). Given a Kripke structure $\mathcal{K}$ providing the meaning of atomic propositions $p \in$ Prop and actions $a \in$ Act, formulae φ and program expressions ρ extend to subsets $[\![\varphi]\!] \in V$ respectively to binary relations

$[\![\rho]\!] \in V \times V$ via simultaneous induction, as follows.

$$\begin{aligned}
[\![a]\!] &:= E_a;\\
[\![\varphi?]\!] &:= \{\ (v, v) \mid v \in [\![\varphi]\!]\ \}.\\
[\![\rho_1;\rho_2]\!] &:= [\![\rho_1]\!] \circ [\![\rho_2]\!];\\
[\![\rho_1 \cup \rho_2]\!] &:= [\![\rho_1]\!] \cup [\![\rho_2]\!];\\
[\![\rho^*]\!] &:= [\![\rho]\!]^*.
\end{aligned}$$

The boolean connectives are interpreted as in ML. Likewise, for the modal operator we set:

$$[\![\langle\rho\rangle\varphi]\!] := \{\ v \mid (\exists w.w \in v[\![\rho]\!])\ w \in [\![\varphi]\!]\ \}.$$

As in the case of ML, operators $\top$, $\wedge$, and $[a]$ are introduced as abbreviations for the respective dual of $\bot$, $\vee$, and $\langle a\rangle$.

When interpreted over linear-time models, i.e. rooted Kripke structures where each state has precisely one successor, PDL is very expressive, as the following result due to Henriksen and Thiagarajan reveals.

Proposition 1.2.8 ([33]). *Over linear-time models,* PDL ≡ MSO.

Nevertheless, the ability of PDL to express properties of programs has severe limitations. A relevant issue is related to the notion of *total correctness*, meaning that from a given state, every execution sequence of a certain program halts. This property cannot be expressed in plain PDL, and therefore several extensions have been suggested. One of these extensions, proposed by Streett [66], adds, for every program ρ, a construct $\Delta\rho$ with the intended meaning that the program ρ can be iterated infinitely often, i.e, that ρ has a non-halting execution sequence.

Definition 1.2.9 (PDL with looping). ΔPDL extends PDL by adding, for every program ρ the operator $\Delta\rho$ yielding a formula interpreted, in an appropriate Kripke structure $\mathcal{K}$, as follows:

$$\begin{aligned}
[\![\Delta\rho]\!] := \{\ v \mid\ & \text{there exists an infinite sequence } (v_i)_{i<\omega}\\
& \text{such that } v_0 = v \text{ and } v_{i+1} = v_i[\![\rho]\!],\ \text{for all } i < \omega\ \}.
\end{aligned}$$

1.2.3 Temporal logics

A different prospective on computational systems modelled by Kripke structures is given by the framework of temporal logics, founded by the work of Pnueli and Manna [60, 50]. While modal logics as those introduced earlier refer explicitly to the execution of programs, temporal logics are geared towards their behaviour in the flow of time, referring to sequences of states that may occur during a run. According to how the flow of time is perceived, a distinction is made between linear and branching time. Linear time corresponds to executions of deterministic programs, or an *a posteriori* view on the execution of a nondeterministic or concurrent program, where each state has a unique successor. In contrast to this, branching time sees the instants of time partially ordered, corresponding to an *a priori* view on the possible executions of a nondeterministic or concurrent program.

Linear time

The structure underlying the linear-time prospective is isomorphic to the ordering of naturals $(\omega, <)$. Every atomic proposition $p \in \textsc{Prop}$ is associated to the set N_p of instances of time i at which it is true. In this way, we can view every linear-time model as an *infinite word* over the alphabet $\wp(\textsc{Prop})$ represented by a structure $\alpha := (\omega, <, (N_p)_{p \in \textsc{Prop}})$.

To reason about such structures, the syntax of LTL provides, in its basic variant, temporal operators $\mathsf{X}\varphi$ and $\varphi_1 \mathsf{U} \varphi_2$. Intuitively, $\mathsf{X}\varphi$ asserts that the formula φ will hold at the *next* moment of time; $\varphi_1 \mathsf{U} \varphi_2$ states that φ_1 holds *until*, after finitely many moments, φ_2 holds.

Definition 1.2.10 (Syntax of LTL). Given a set Prop of propositions, the formulae of Linear Temporal Logic LTL are constructed according to the following grammar:

$$\varphi := \bot \mid p \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi$$

where $p \in \textsc{Prop}$.

Observe that, in contrast to ML and PDL, the syntax of LTL does not refer to actions.

Definition 1.2.11 (Semantics of LTL). Given a linear-time model

$$\alpha := (\omega, <, (N_p)_{p \in \textsc{Prop}}),$$

we define the truth of LTL-formulae inductively, using the notation $\alpha|_i$ for the *suffix* of α starting at position i, i.e., the linear-time model induced in α by $\{ j \mid i \leq j < \omega \}$:

$\alpha \not\models \bot$;
$\alpha \models p$ iff $0 \in V_p$;
$\alpha \models \neg\varphi$ iff it is not the case that $\alpha \models \varphi$;
$\alpha \models \varphi_1 \vee \varphi_2$ iff $\alpha \models \varphi_1$ or $\alpha \models \varphi_2$;
$\alpha \models \mathsf{X}\varphi$ iff $\alpha|_1 \models \varphi$;
$\alpha \models \varphi_1 \mathsf{U} \varphi_2$ iff $\exists j\, (\alpha|_j \models \varphi_2 \wedge (\forall i < j)\, \alpha|_i \models \varphi_1)$.

Clearly, LTL-formulae can be translated into FO. That the converse also holds is a very deep result showed by Kamp and later generalised by Gabbay, Pnueli, Shelah, and Stavi.

Theorem 1.2.12 ([41, 23]). *Over linear-time models,* LTL ≡ FO.

Branching time

The structures underlying branching-time logics are of tree-like nature, where each moment of time may have several successors. The full paths of such a tree are then linear-time structures, corresponding to possible executions of a program. Essentially, computation tree logics allow to quantify over these paths and to speak about them in a way similar to LTL.

Definition 1.2.13 (Syntax of CTL*). Given a set of atomic propositions Prop, the formulae of the Computation Tree Logic CTL* are of two sorts, state and path formulae, generated respectively by the following grammar:

$$\varphi := \bot \mid p \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathsf{E}\eta$$
$$\eta := \varphi \mid \eta \vee \eta \mid \neg\eta \mid \mathsf{X}\eta \mid \eta \mathsf{U} \eta$$

where $p \in$ Prop.

To define the semantics of branching-time logics in terms of Kripke structures, we associate to every structure $\mathcal{K}$, its *computation tree*, that is, the tree obtained by unravelling $\mathcal{K}$ and dropping all action labels. For uniformity, we will consider only Kripke structures where each state has at least one successor. The infinite

sequence of labels from $\wp(\text{Prop})$ seen on a path π through such a Kripke structure then induces a linear-time structure, which we call the *trace* of π, denoted by $\mathcal{K}, \pi$. Intuitively, the formula $\mathsf{E}\eta$ asserts that there exists a path starting at the current node whose trace models η.

Definition 1.2.14 (Semantics of CTL*)**.** Given a Kripke structure $\mathcal{K}$, the truth of CTL*-formulae is defined by mutual induction over path and state formulae. Thereby, path formulae are interpreted over traces of full paths through (the computation tree of) $\mathcal{K}$ according to the rules for LTL. The quantifier E transforms any path formula η into a state formula $\mathsf{E}\eta$ with the following extension:

$$[\![\mathsf{E}\eta]\!] := \{\, v \mid \text{there exists an infinite path } \pi = (v_0, v_1, \dots) \text{ in } \mathcal{K} \text{ such that } v_0 = v \text{ and } \mathcal{K}, \pi \vDash \eta \,\};$$

Boolean connectives in state formulae are interpreted as in ML.

Although we can phrase the semantics of CTL* in terms of Kripke structures, the languages speaks, in fact, about trees, namely the computation trees associated to the structures under consideration. Since over paths, CTL* corresponds to LTL which can be translated into FO, by Theorem 1.2.12, and monadic quantification operates on paths, it is easy to see that CTL* can be translated into monadic path logic MPL. The following theorem, initially demonstrated by Hafer and Thomas for binary trees and extended by Moller and Rabinovich over arbitrary trees states that the converse is true as well, showing that CTL* is expressively complete for MPL, over the class of tree models, up to bisimulation.

Theorem 1.2.15 ([30, 52])**.** *Over trees,* MPL/$\sim \equiv$ CTL* : *An* MPL*-formula is invariant under bisimulation over trees if, and only if it is equivalent to a* CTL**-formula.*

Under the paradigm of temporal logics, LTL and CTL* cannot distinguish between individual actions. If we wish to compare the expressive power of temporal and dynamic logics, it is therefore reasonable to restrict our consideration to structures with only one action. Yet, it turns out that even on such structures, CTL* cannot formalise properties which, can be expressed, e.g., in PDL.

Proposition 1.2.16 ([77, 78])**.** CTL* $<$ ΔPDL *over Kripke structures with a single action.*

Proof. In [77], Wolper describes a translation of CTL* into ΔPDL. On the negative side, the same author shows in [78] that the property asserting that proposition p holds at every second state of a structure cannot be formalised in CTL*. Already in PDL, this property can be expressed as $[(a;a)^*]p$.

Another way to see that CTL* is less expressive than ΔPDL is by looking at linear-time models, where CTL* collapses to LTL, and hence to FO, while already PDL attains the expressive power of MSO (see Proposition 1.2.8). □

1.2.4 The modal μ-calculus

Dynamic and temporal logics can be understood as extensions of Hennessy-Milner Logic with different recursion mechanisms. As a common feature, recursion in these settings corresponds in fact to fixed point iteration. Typically in CTL*, for instance, the equivalence

$$\mathsf{E}(\varphi \,\mathsf{U}\, \psi) \equiv \psi \vee (\varphi \wedge \Diamond \mathsf{E}(\varphi \,\mathsf{U}\, \psi))$$

allows us to see the extension of $\mathsf{E}(\varphi \,\mathsf{U}\, \psi)$ as a solution Z of the equation

$$[\![Z]\!] = [\![\psi \vee (\varphi \wedge \Diamond Z)]\!].$$

A closer analysis shows that $[\![\mathsf{E}(\varphi \,\mathsf{U}\, \psi)]\!]$ is actually the least solution of this equation, with respect to set inclusion. If we view the expression on the right side of the equation as an operator $Z \mapsto [\![\psi \vee (\varphi \wedge \Diamond Z)]\!]$, then the value for Z in which we are interested, is just the least fixed point of this operator.

In a similar way, we can use the characterisations of the basic iterative constructions of PDL and ΔPDL,

$$\langle \rho^* \rangle \varphi \equiv \varphi \vee \langle \rho^* \rangle \varphi, \quad \text{and} \quad \Delta\rho \equiv \langle \rho \rangle \Delta\rho,$$

to formulate their semantics in terms of least, respectively greatest fixed points of the operators induced by the equations

$$[\![Z]\!] = [\![\varphi \vee \langle \rho \rangle Z]\!], \quad \text{and} \quad [\![Z]\!] = [\![\langle \rho \rangle Z]\!].$$

Notice that all operators $Z \mapsto F(Z)$ involved in these descriptions are monotone, in the sense that $Z \subseteq Z'$ implies $F(Z) \subseteq F(Z')$.

The modal μ-calculus L_μ introduced in its current form by Kozen [44], extends Hennessy-Milner Logic by incorporating into the language a constructor for building least fixed points of definable monotone operators. This provides a notion of recursion which invests the logic with very high expressive power, far beyond that of CTL* and ΔPDL, as we shall see.

Definition 1.2.17 (Syntax of L_μ). Given a set Prop of atomic propositions, a set Act of atomic actions and a set Var of monadic variables, the formulae of L_μ are constructed according to the following grammar:

$$\varphi := \bot \mid p \mid X \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle a \rangle \varphi \mid \mu X.\varphi$$

where $p \in$ Prop, $a \in$ Act, and $X \in$ Var, and the fixed point rule $\mu X.\varphi$ applies to formulae $\varphi(X)$ in which the free variable X appears only positively, that is, under an even number of negations.

In contrast to the modal and temporal logics introduced so far, the language of L_μ includes variables. The number of distinct variables appearing in an L_μ-formula induces the following syntactic hierarchy.

Definition 1.2.18 (Variable hierarchy of L_μ). For any $k \in \mathbb{N}$, the *k-variable fragment* $\mathrm{L}_\mu[k]$ of the μ-calculus is the set of formulae $\psi \in \mathrm{L}_\mu$ that contain at most k distinct variables.

Since we conceive the operator μ as a quantifier, the notions of variable binding, free and bound occurrence, and quantifier scope are used in the sense familiar from predicate logics. A formula in which all occurring variables are bound is called *closed*.

Definition 1.2.19 (Semantics of L_μ). To define the meaning of a formula $\varphi \in \mathrm{L}_\mu$ in a Kripke structure

$$\mathcal{K} = (V, (E_a)_{a \in \mathrm{Act}}, (A_p)_{p \in \mathrm{Prop}}),$$

we describe its extension $[\![\varphi]\!]_\chi$ referring to an assignment $\chi : \mathrm{Var} \to V$ that provides interpretations of the free variables in φ. As in the case of Hennessy-Milner Logic, the constant $\bot$ corresponds to the empty set, atomic propositions $p \in$ Prop extend to the sets V_p, and the extension of free variables X is given by $\chi(X) \subseteq V$. For the

propositional and modal operators, we have

$$\begin{aligned} [\![\neg\varphi]\!]_\chi &:= V \smallsetminus [\![\varphi]\!]_\chi; \\ [\![\varphi_1 \vee \varphi_2]\!]_\chi &:= [\![\varphi_1]\!]_\chi \cup [\![\varphi_2]\!]\chi; \\ [\![\langle a\rangle\varphi]\!]_\chi &:= \{\, v \mid (\exists w \in vE_a)\, w \in [\![\varphi]\!]_\chi \,\}. \end{aligned}$$

To understand the semantics of fixed point formulae $\mu X.\varphi$, note that a formula $\varphi(X)$ with a propositional variable X defines on every Kripke structure $\mathcal{K}$ (with state set V, and with interpretations χ for free variables other than X occurring in φ) an operator $\varphi^{\mathcal{K}} : \wp(V) \to \wp(V)$ which maps any set $T \subseteq V$ to the extension $[\![\varphi]\!]_{\chi[X:=T]}$ obtained with the assignment χ when the value of X is set to T. By the requirement on φ to contain X only positively, this operator is *monotone* for every $\mathcal{K}$. From Knaster and Tarski's classical fixed point theorem [67], it follows that $\varphi^{\mathcal{K}}$ has a least fixed point,

$$\mathrm{lfp}(\varphi^{\mathcal{K}}) = \bigcap \{\, T \subseteq V \;:\; T = [\![\varphi]\!]_{\chi[X/T]} \,\}.$$

Now, we put $[\![\mu X.\varphi]\!]_\chi := \mathrm{lfp}(\varphi^{\mathcal{K}})$.

As usual, we introduce the operators $\top$, $\wedge$, and $[a]$ to abbreviate the duals of $\bot$, $\wedge$, respectively $\langle a\rangle$. The operator ν is introduced as an abbreviation for the dual of μ. In this way, $\nu X.\varphi := \neg\mu X.\neg\varphi[\neg X/X]$ is interpreted as the greatest fixed point of $\varphi^{\mathcal{K}}$. By exploiting the duality between these operators, every L_μ-formula can be easily brought into a form where negation does not interfere with the other operators.

Definition 1.2.20 (Negation normal form). An L_μ-formula is in *negation normal form*, if the negation operator appears only in front of atomic propositions.

Definition 1.2.21 (Existential L_μ). A μ-calculus formula is called *existential*, if its presentation in negation normal form contains no universal modality $[a]\varphi$.

It can be easily verified that the validity of existential formulae is preserved under simulation: For every pair of structures $\mathcal{K}, u \lesssim \mathcal{K}', u'$ and any existential formula ψ such that $\mathcal{K}, u \vDash \psi$, we have $\mathcal{K}', u' \vDash \psi$.

It is often useful to refer to the syntax of formulae in terms of graphs. Considering presentations in negated normal form, we first define the *syntax tree* $\mathcal{T}_\psi$ of a formula $\psi \in \mathrm{L}_\mu$ inductively, by associating atomic formulae (propositions and their negations, variables, and the constants $\bot$, $\top$) to isolated nodes, unary constructs

$\langle a\rangle\varphi$, $[a]\varphi$, $\mu X.\varphi$, $\nu X.\varphi$ to trees with a single immediate subtree $\mathcal{T}_\varphi$, and binary constructs $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$ to trees with two immediate subtrees $\mathcal{T}_{\varphi_1}$ and $\mathcal{T}_{\varphi_2}$. If we now introduce, for every leaf corresponding to a variable occurrence X, a link to (the unique node which corresponds to) to its binding definition $\mu X.\varphi$ or $\nu X.\varphi$, we obtain an operational representation of ψ as a tree with back edges, which we call its *syntax graph* $\mathcal{G}_\psi$. Sometimes it is convenient to eliminate variables from the representation of a formula by identifying every occurence of a variable with its binding definition in the syntax tree thus obtaining a *contracted* syntax graph.

Essentially, syntax trees and graphs reflect the building process of a formula. Conversely, from a consistent representation, i.e., a binary tree, with or without back edges, where the leafs are labelled with atoms and the inner nodes with boolean connectives, modalities, or fixed-point definitions μX, νX, we can easily reconstruct the corresponding formula.

Let us fix a closed formula $\psi \in \mathrm{L}_\mu$ and consider, for any subformula φ, the graph $\mathcal{G}_\psi, \varphi$ obtained from the contracted syntax graph of ψ by choosing φ as a root and discarding all nodes unreachable from φ. Then, $\mathcal{G}_\psi, \varphi$ is itself a contracted syntax graph, corresponding to the formula obtained by replacing recursively every free occurrence of a variable in φ by its binding definition.

Definition 1.2.22 (Closure of an L_μ-formula)**.** Let ψ be an L_μ-formula without free variables. For each subformula φ in ψ, we define its *closure* $\mathrm{cl}_\psi(\varphi)$ as the formula constructed according to $\mathcal{G}_\psi, \varphi$ viewed as a syntax graph. By $\mathrm{cl}(\psi)$ we denote the set of closures of all subformulae in ψ.

Intuitively, the notion of closure captures the meaning of a subformula within a closed formula. All formulae in $\mathrm{cl}(\psi)$ are themselves closed. As an immediate consequence of the above definition, we obtain the following characterisation.

Lemma 1.2.23. *The* closure $\mathrm{cl}(\psi)$ *of an* L_μ*-formula* ψ *without free variables is the smallest set of formulae so that* $\psi \in \mathrm{cl}(\psi)$ *and*

(i) *if* $\varphi_1 \vee \varphi_2 \in \mathrm{cl}(\psi)$ *or* $\varphi_1 \wedge \varphi_2 \in \mathrm{cl}(\psi)$*, then* $\{\varphi_1, \varphi_2\} \subseteq \mathrm{cl}(\psi)$*;*

(ii) *if* $\langle a\rangle\varphi \in \mathrm{cl}(\psi)$ *or* $[a]\varphi \in \mathrm{cl}(\psi)$*, then* $\varphi \in \mathrm{cl}(\psi)$*;*

(iii) *if* $\mu X.\varphi(X) \in \mathrm{cl}(\psi)$ *or* $\nu X.\varphi(X) \in \mathrm{cl}(\psi)$*, then* $\varphi(\mu X.\varphi(X)) \in \mathrm{cl}(\psi)$ *respectively* $\varphi(\nu X.\varphi(X)) \in \mathrm{cl}(\psi)$*.*

Least and greatest fixed points can also be constructed inductively. Given a formula $\nu X.\psi$, we define for each ordinal α, the stage X^α of the gfp-induction of

$\psi^{\mathcal{K}}$ by $X^0 := V$, $X^{\alpha+1} := [\![\psi]\!]_{\chi[X/X^\alpha]}$, and $X^\alpha := \bigcap_{\beta<\alpha} X^\beta$ if α is a limit ordinal. By monotonicity, the stages of the gfp-induction decrease until a fixed point is reached. By ordinal induction, one easily proves that this inductively constructed fixed point coincides with the greatest fixed point. The *finite approximations* of a formula $\nu X.\varphi$ are defined by $\varphi_0 := \top$ and $\varphi_{n+1} = \varphi[X/\varphi_n]$ (the formula obtained by replacing every occurrence of X in φ, by φ_n. Obviously $\nu X.\varphi \vDash \varphi_n$ for all n, and on finite Kripke structures (in fact, even on finitely branching ones) also the converse holds: If $\mathcal{K}, v \vDash \varphi_n$ for all n, then also $\mathcal{K}, v \vDash \nu X.\varphi$.

Simultaneous Fixed Points. There is a variant of L_μ that admits simultaneous fixed points of several formulae. This does not increase the expressive power but allows more transparent formalisations. The mechanism for building simultaneous fixed-point formulae is the following. Given formulae $\varphi_1, \ldots, \varphi_n$ and variables $X_1, \ldots, X_n$, we can write an *equational system* $S := \{X_1 = \varphi_1, \ldots, X_n = \varphi_n\}$ and build formulae $(\mu X_i : S)$ and $(\nu X_i : S)$. On every structure $\mathcal{K}$, the system S defines an operator $S^{\mathcal{K}}$ mapping an n-tuple $\bar{X} = (X_1, \ldots, X_n)$ of sets of states to $S_1^{\mathcal{K}}(\bar{X}), \ldots, S_n^{\mathcal{K}}(\bar{X})$ so that, for each i we have: $S_i^{\mathcal{K}}(\bar{X}) := [\![\varphi_i]\!]^{(\mathcal{K},\bar{X})}$. As $S^{\mathcal{K}}$ is monotone, it has extremal fixed points $\mathrm{lfp}(S) = (X_1^\mu, \ldots, X_n^\mu)$ respectively $\mathrm{gfp}(S) = (X_1^\nu \ldots, X_n^\nu)$, and we set $[\![(\mu X_i : S)]\!]^{\mathcal{K}} := X_i^\mu$ and $[\![(\nu X_i : S)]\!]^{\mathcal{K}} := X_i^\nu$.

It is known that simultaneous least fixed points can be eliminated in favour of nested individual fixed points.

Proposition 1.2.24 ([2]). *Every formula in L_μ with simultaneous fixed points can be translated into an equivalent formula in plain* L_μ *without increasing the number of variables.*

The μ-calculus displays a series of pleasant model-theoretic properties. Being a modal logic, L_μ is invariant under bisimulation, i.e. for every $\mathcal{K}, u \sim \mathcal{K}', u'$ and for any $\psi \in \mathrm{L}_\mu$, we have $\mathcal{K}, u \vDash \psi$ if, and only, if $\mathcal{K}', u' \vDash \psi$.

As a consequence of bisimulation invariance and because every Kripke structure is bisimilar to its tree unravelling, the μ-calculus enjoys the *tree model property*, meaning that every satisfiable formula is satisfiable in a tree.

Another significant feature of L_μ is its *finite model property*.

Theorem 1.2.25 ([45]). *Every satisfiable* L_μ*-formula has a finite model.*

Since the unravelling of a finite model is a finitely branching tree, we obtain the following corollary.

Corollary 1.2.26. *Every satisfiable* L_μ*-formula is satisfied in some finitely branching tree.*

Obviously, the least and greatest fixed point constructions of L_μ can be replicated using monadic second-order quantification, thus, embedding L_μ into MSO. A crucial insight into the expressive power of the μ-calculus is yield by the Modal Characterisation Theorem of Janin and Walukiewicz, which identifies L_μ as the bisimulation-invariant fragment of MSO.

Theorem 1.2.27 ([38]). $\mathrm{MSO}/{\sim} \equiv \mathrm{L}_\mu$: *An* MSO*-formula is invariant under bisimulation if, and only if, it is equivalent to a formula of* L_μ.

Since MSO is considered to capture all reasonably desirable properties of Kripke structures, this theorem indeed states that the μ-calculus is, in a broad sense, expressively complete.

Indeed, all other modal and temporal logics studied throughout this sections are embeddable into L_μ, but are considerably weaker in terms of expressive power. An explicit embedding of ΔPDL into L_μ results from our treatment of Game Logic in Chapter 2. The relation between CTL* and ΔPDL was already established in Proposition 1.2.16. In Proposition 2.2.1 we will also give a concrete example of a property expressible in L_μ but not in ΔPDL. Consequently, the specification formalisms discussed in the present section are ordered as follows, according to their expressiveness.

Proposition 1.2.28. $\mathrm{LTL} < \mathrm{CTL}^* < \Delta\mathrm{PDL} < \mathrm{L}_\mu$.

1.3 Model checking and parity games

In the previous sections, we have considered concurrent systems modelled as games over Kripke structures with winning conditions derived from behavioural specifications represented by formulae of a certain logic. In this section we argue that questions regarding the meaning of formulae over Kripke structures can conversely be phrased (and solved) in terms of games.

Given a Kripke structure $\mathcal{K}$ modelling a reactive system, and a specification φ in a certain logic $\mathcal{L}$, the associated model-checking problem consists in deciding whether the system $\mathcal{K}$ meets the specification φ, that is, whether $\mathcal{K} \models \varphi$. This

problem can be equivalently formulated in terms of the semantic games of the logic under consideration. The semantic games associated to L_μ (and to fixed point logics in general) are parity games.

1.3.1 Parity games

Definition 1.3.1 (Parity game). A *parity game* is represented by a rooted Kripke structure $\mathcal{G}, v_0$ with

$$\mathcal{G} = (V, V_0, E, (\Omega_i)_{i<n}),$$

where V is a set of *positions* with a designated subset V_0, $E \subseteq V \times V$ is a transition relation, and $\Omega = (\Omega_i)_{i<n}$ is a labelling of V with priorities $0, \ldots, n-1$ determining the winning condition. We denote the set $V \setminus V_0$ by V_1. The number n of different priorities is called the *index* of $\mathcal{G}$.

In a play of $\mathcal{G}, v_0$ two players, generically called Player 0 and Player 1, move a token along the transitions of E starting from v_0, thus forming a path $v_0, v_1, \ldots$. Once a position v is reached, Player 0 performs the move if $v \in V_0$, otherwise Player 1. If the current position allows no further transitions, then the player in turn to move loses. In case this never happens, the play is infinite and the winner is established by looking at the sequence $\Omega(v_0), \Omega(v_1), \ldots$ If the least priority appearing infinitely often in this sequence is even, Player 0 wins the play, otherwise Player 1 wins.

Alternatively, we may think of a parity game $\mathcal{G}, v_0$ as a game in extensive form, where the history tree coresponds to the unravelling of $\mathcal{G}, v_0$ and the history nodes are assigned to the players according to their last state. Then, the winning condition for Player 0 is satisfied by a full path in this history if it is finite and its last state belongs to Player 1, or if is infinite and the least priority appearing infinitely often on the path is even.

For these games, strategies that do not depend on the entire history, but only on the current position are particularly relevant.

Definition 1.3.2 (Memoryless strategy). A *memoryless* strategy for Player 0 in the parity game $\mathcal{G}, v_0$ is a function $\sigma : V_0 \to V$ assigning to each position $v \in V_0$ a successor $w \in vE$.

When viewing $\mathcal{G}$ as an extensive game, any such strategy naturally corresponds to a strategy in the sense of Definition 1.1.3.

The Memoryless Determinacy Theorem of Emerson and Jutla states that parity games are always determined with memoryless strategies.

Theorem 1.3.3 (Memoryless determinacy [21]). *In any parity game, either Player* 0 *or Player* 1 *has a memoryless winning strategy.*

Every memoryless strategy σ induces in $\mathcal{G}$ a subgame $\mathcal{G}_\sigma$ obtained by removing the transitions (v, w) from $v \in V_0$ to $w \neq \sigma(v)$. Notice that if σ is a winning strategy for a player, this player wins every play on $\mathcal{G}_\sigma$.

Since memoryless strategies are small objects and it can be checked efficiently whether such a strategy is winning, the winner of a parity game can be established in NP ∩ co-NP. However, the best known deterministic algorithms for solving this problem have running times that are polynomial with respect to the size of the game graph, but exponential with respect to the index of the game [40].

1.3.2 Model checking games for L_μ

In terms of games, the interpretation of negation is an intricate matter. To avoid this difficulty, we will henceforth assume wihtout loss that formulae are presented in negation normal form, where only atomic propositions appear negated.

Given a Kripke structure $\mathcal{K}, u$ and a L_μ-formula ψ, the model-checking game $\mathcal{G}(\mathcal{K}, \psi, u)$ is a parity game associated to the problem whether $\mathcal{K}, v_0 \vDash \psi$. There are several, essentially equivalent, ways to define this game. In the more transparent one, positions are pairs (v, φ) where φ is any, not necessarily closed subformula of ψ, and it is assumed that every variable is bound at most once by a fixed-point definition (see, e.g., [8, 65]). However, because we later want to reuse variables, we resort to a variant more familiar in automata theory which, instead of subformulae, refers to their closure [20, 47].

The positions in the game $\mathcal{G}(\mathcal{K}, \psi, u)$ are pairs (v, φ) of states $v \in K$ and formulae $\varphi \in \mathrm{cl}(\psi)$. The first player, called Verifier, moves from the positions $(v, \varphi_1 \vee \varphi_2)$, $(v, \langle a \rangle \varphi)$, (v, p) with $v \notin p$, and $(v, \neg p)$ with $v \in p$ and his opponent, called Falsifier, moves from every other position. All plays start at (u, ψ) and can proceed as follows:

- no moves are possible from (v, α) where α is atomic or negated atomic;
- from $(v, \varphi_1 \vee \varphi_2)$ or $(v, \varphi_1 \wedge \varphi_2)$ available moves lead to (v, φ_1) and (v, φ_2);

- from $(v, \langle a \rangle \varphi)$ or $(v, [a]\varphi)$ there are available moves to all positions (w, φ) where w is an a-successor of v;
- from $(v, \mu X.\varphi(X))$ or $(v, \nu X.\varphi(X))$ only one move is possible, leading to $(v, \varphi(\mu X.\varphi(X))$ respectively $(v, \varphi(\nu X.\varphi(X))$.

Thus, a play proceeds along the paths in $\mathcal{K}$ and in the syntax tree of ψ, up to the point where a fixed-point variable is met (a leaf in the syntax tree). There, the play resumes with the variable's binding definition in the second component. We call this event *regeneration* of a variable. Observe that before a variable is regenerated, its binding definition has already been met when the play descended the syntax tree. We say that the variable is *generated* at that position.

One technically useful property of fixed point formulae is guardedness. In terms of games this guarantees that between the binding definition of a variable and its regeneration we always have at least one modal move.

Definition 1.3.4. An L_μ-formula ψ is *guarded* if each path in the syntax tree of ψ from a fixed point definition $\lambda X.\varphi$ to an occurrence of X passes through a modality, $\langle a \rangle \eta$ or $[a]\eta$.

In [47], Kupferman, Vardi, and Wolper give a procedure to transform any L_μ-formula into an equivalent guarded formula. This procedure does not increase the number of variables and preserves existential formulae.

Proposition 1.3.5. *Every existential formula in* $L_\mu[k]$ *is equivalent to a guarded existential formula in* $L_\mu[k]$.

While repeatedly regenerating variables, it may happen that neither Verifier nor Falsifier gets stuck. To decide such plays, priorities have to be defined appropriately. The intuition is that, to establish the truth of a μ-formula, Verifier should regenerate it only finitely often whereas ν-formulae can be regenerated infinitely often. Of course the difficulty may be that μ- and ν-formulae are deeply nested and there are several fixed-point formulae that are regenerated infinitely often during a play. But it can be shown that among these, there is always an outermost one, which determines the winner: if it is a ν-formula Verifier wins, if it is a μ-formula, Falsifier wins. Hence, the priority labeling assigns even priorities to positions $(v, \nu X.\varphi)$ and odd priorities to positions $(v, \mu X.\varphi)$. Further, priorities respect dependencies. If $\nu Y.\varphi$ depends on $\mu X.\eta$ then priorities of positions $(v, \nu Y.\varphi)$ are higher than those of positions $(w, \mu X.\eta)$. The remaining positions receive priorities that are higher than those associated with fixed-point formulae.

Theorem 1.3.6 ([20, 65]). *Verifier has a winning strategy in the model checking game $\mathcal{G}(\mathcal{K}, \psi, u)$ from position (u, ψ) iff $\mathcal{K}, u \models \psi$.*

Besides defining the truth of L_μ-formulae in terms of games, this characterisation also gives us access to a notion of game-theoretic *proof* or *rejection* of the validity of a formula. To be more precise, for a given structure $\mathcal{K}$ and a formula $\psi \in \mathrm{L}_\mu$, a (memoryless) winning strategy for Player 0 in the model checking game $\mathcal{G}(\mathcal{K}, \psi, u)$ can be viewed as a proof of $\mathcal{K}, v \models \psi$ in an interactive proof system. Verifier can convince Falsifier that ψ holds at u, by choosing according to its strategy whenever a disjunction or an existential subformula of ψ is considered. In the same way, a (memoryless) winning strategy for Player 1 can be seen as a rejection of $\mathcal{K}, v \models \psi$.

The following property follows from Theorem 1.3.6.

Corollary 1.3.7. *Let $\mathcal{K}, u$ be a model of a formula $\psi \in \mathrm{L}_\mu$ and let σ be a winning strategy for Verifier in the associated model-checking game $\mathcal{G}(\mathcal{K}, \psi)$. Then, for every position (v, φ) reachable in $\mathcal{G}_\sigma$ from the initial position (u, ψ), we have $\mathcal{K}, v \models \mathrm{cl}_\psi(\varphi)$.*

A different way to define model-checking games for L_μ refers to the closure of subformulae rather than their occurrence [20, 47]. The games obtained in this way are equivalent to those introduced here – in fact, they are bisimilar. We may therefore use this alternative definition where a more semantic viewpoint is appropriate.

We remark that the game theoretic semantics of L_μ in terms of parity games was sucessfully generalised to least fixed point extensions of first-order logic and its fragments, providing new insight beyond the scope of modal logics [8, 28, 26, 6, 9].

1.3.3 Expressing parity semantics in L_μ

A crucial feature of the μ-calculus is that it can express the notion of winning in its natural model-checking games.

Theorem 1.3.8. ([21]) *For every index n, there is a formula $\mathrm{W}^n \in \mathrm{L}_\mu$, such that in any parity game $\mathcal{G}, v_0$ with n priorities Player 0 has a winning strategy if, and only if, $\mathcal{G}, v_0 \models \mathrm{W}^n$.*

We develop a variant of the formula W^n here. For convenience, let us abbreviate the formula expressing that Player 0 can ensure that a position where φ holds is reached in one move by

$$\triangleright\varphi := (V_\Diamond \wedge \Diamond\varphi) \vee (V_\Box \wedge \Box\varphi).$$

Further, we write $\Omega_{>i}$ for $\bigvee_{k=i+1}^{n} \Omega_k$. Empty disjunctions, like $\Omega_{>n}$, are interpreted as false. For simplicity, let us assume that n is odd.

Definition 1.3.9 (Parity characterisation). The formula W^n expressing that Player 0 has a winning strategy in a parity game with n priorities is

$$\mathrm{W}^n := \mu Z_1 \nu Z_2 \ldots \mu Z_n . \bigvee_{i=1}^{n} (\Omega_i \wedge \rhd Z_i).$$

To understand this expression, let us consider the formulae $W_i(\varphi)$ describing those positions from which Player 0 can ensure that

(i) either he wins while no priority less than i is ever played, or

(ii) some position where φ holds is being reached.

We obtain, for odd i,

$$W_i(\varphi) := \mu Z . W_{i+1}\big(\varphi \vee (\Omega_i \wedge \rhd Z)\big),$$

and, for even i,

$$W_i(\varphi) := \nu Z . W_{i+1}\big(\varphi \vee (\Omega_i \wedge \rhd Z)\big).$$

Thus, the above expression for W^n is given by $W_1(\bot)$.

1.3.4 The L_μ alternation hierarchy

A well-studied measure for the descriptive complexity of L_μ is the the *alternation depth*, that is, the number of (genuine) alternations between least and greatest fixed points occurring in a formula.

Definition 1.3.10 (Alternation hierarchy of L_μ). The μ-calculus *alternation hierarchy* is defined as follows:

(i) The first level of the hierarchy, $\Sigma_0^\mu = \Pi_0^\mu$, consists of the formulae in which no fixed-point operators occur.

(ii) For every higher level, Σ_{i+1}^μ is formed by closing $\Sigma_i^\mu \cup \Pi_i^\mu$ under positive boolean and modal operators, $\wedge$, $\vee$, $\langle\cdot\rangle$, and $[\cdot]$, and under the least fixed-point operator μ.

(iii) The level Π_{i+1}^* is obtained dually, by closing $\Sigma_i^\mu \cup \Pi_i^\mu$ under ν instead of μ.

Interestingly, most most temporal and dynamic logics used for describing concurrent systems allow translations into low levels of the L_μ alternation hierarchy. On its first level this hierarchy already captures, for instance, PDL as well as CTL, while their expressive extensions ΔPDL and CTL* do not exceed the second level. Still, the low levels of this hierarchy do not exhaust the significant properties expressible in L_μ.

In [15] Bradfield showed that the alternation hierarchy of the μ-calculus is semantically strict and that the formulae W^n are hard instances for the n-th level of this hierarchy. Variants of this result, relying as well on variants of W^n, have also been proved by Lenzi [49] and Arnold [1].

Theorem 1.3.11 ([15, 49]). *For any index n, the formula* W^n *is contained on the n-th level of the μ-calculus hierarchy but there is no formula equivalent to* W^n *at level* $n-1$.

2 Game Logic

While games offer a very powerful and intuitive language for reasoning about interaction, this creates, in turn, the exigence to devise appropriate metalanguages for reasoning about games. An important contribution to this aim has been brought by Parikh's Game Logic GL, introduced in [56], a formalism concerned with the dynamic of games, more specifically, with the evolution of a player's power in a game.

In first instance, GL provides a syntax to compose complex games starting from primitive ones, just in the same way as PDL programs are built up from atomic actions. This gives rise to game expressions specifying a schedule for two players, Angel and Demon, according to the following outline. The sequential composition of two games $\gamma_1;\gamma_2$ means: play γ_1 first, then γ_2. The nondeterministic choice operator $\gamma_1 \cup \gamma_2$ lets Angel decide which of the two games γ_1 or γ_2 is to be played. The iteration operator γ^* allows to play the game γ repeatedly, for a finite number of times, whereby Angel can decide before each round whether a new round is to be played. Finally, the test operator $(\varphi?)$ invokes an independent observer to check whether φ holds; if so, the play just ends, otherwise it breaks and Angel loses.

In extension to these, the formalism introduces an explicit alternation operator γ^{d}, which directs Angel and Demon to play γ with interchanged roles. As we shall see, this notion of dualisation, corresponding to a form of game-theoretic negation, represents a deep source of expressive power.

Similar to the games on Kripke structures discussed in the previous chapter, in GL the interpretation of games refers to the states of an external system (the world). The evolution of this system is determined by the interaction of the two players: as an outcome of a game played at a given state, the system shifts into a new state. Accordingly, the semantic model associates to every game the outcomes, in terms of states, that may arise when the game is played at a given state. However, in contrast to PDL, where the possible outcomes of executing a program can be described as a subset of the state set, transitions determined by games have a finer structure. This is captured by the notion of *effectivity functions* representing the power of a

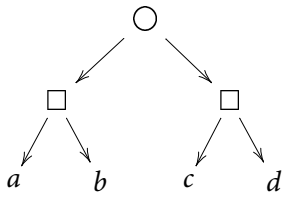


Figure 2.1: An extensive game form with outcomes in $\{a, b, c, d\}$

player in a game. Given a set V of states, an effectivity function $f : V \mapsto \wp(\wp(V))$ describes, for every state v, a collection of subsets of V within which the player can force the outcome when the game is being played at v. In line with this intuition, effectivity functions are required to be monotonic in the sense that, if $Z \in f(v)$ and $Z \subseteq Z' \subseteq V$, then also $Z' \in f(v)$. For a formal review on effectivity functions, we refer the reader to the survey of Vannucci [72] on this topic.

To illustrate the notion of effectivity, let us consider the power of the two players in the game form depicted in Figure 2.1. Although the first player cannot guarantee the play to end in any specific state of $\{a, b, c, d\}$, he can ensure, e.g., that an outcome in $\{a, b\}$ is achieved. In detail, the value of his effectivity function at a state v associated to this game is

$$f(v) = \{\, \{a, b\}, \{c, d\}, \{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}, \{a, b, c, d\} \,\}.$$

Dually, his opponent, can ensure that the play ends in any of the sets $\{a, c\}$, $\{a, d\}$, $\{b, c\}$, and $\{b, d\}$, or in any superset of these.

Effectivity functions bear a high level of abstraction. Non-determined games, for instance, can be easily modelled by providing separate effectivity functions for the two opponents. However, in Game Logic, games are assumed to be determined and strictly competitive. Therefore, the values $f(v)$ of one player's effectivity implicitly define the opponent's effectivity $\overline{f}(v) := \{\, Z \subseteq V \mid \overline{Z} \notin f(v) \,\}$.

In Parikh's original setting, primitive games are represented by the associated effectivity function, generalising the notion of transition. Consequently, the intended models for GL are higher-order transition systems, called neighbourhood models, with propositional state labels as usual, but effectivity relations over $V \times \wp(V)$ rather than transitions over $V \times V$. Statements about these models are constructed by associating game expressions with modalities. Typically, the statement $\langle\gamma\rangle\varphi$ expresses that, at the current state, Angel has a strategy to play the game γ in such a way that either φ is true when the play ends, or the game breaks and Demon fails.

Some conceptual remarks are in place here. In the classical view of game theory, the notion of utility, or winning, and the specification of the actions available to the players are essential to the definition of a game. What we refer to as a game in GL is, to a large extent, detached from both of these aspects. At the atomic level, games model only *what* the players can achieve, not *how* nor *why* they should pursuit a certain outcome. The composition of games, however, gradually adds possible actions (choose one or the other game to play, reiterate or not) and also a sense of losing (by reitering a game γ^* infinitely often, or by failing a test). Finally, when a game expression γ is casted into a propositional statement $\langle\gamma\rangle\varphi$, a determination of winning is provided. Thus, game expressions in GL define rules for constructing an extensive game form over internal positions while the atomic game forms and the player's utilities are provided in terms of external states of the world. This interplay of game-theoretical constituents with concepts from computer science results in a non-classical notion of great potential. See [57] for an extensive exposition on this subject.

In the present chapter, we investigate the expressive power of Game Logic interpreted in Kripke structures. These correspond to neighbourhood models with one-player atomic games. Alternation between players in complex games can be induced syntactically, via the dualisation operator. We show that each new level of syntactic alternations makes the logic stronger, i.e., that the alternation hierarchy of Game Logic is strict. Our proof technique relies in encoding the winning conditions for parity games into GL. This further allows us to conclude that Game Logic intersects nontrivially every finite level of the μ-calculus alternation hierarchy. Finally, it shows that the model checking problem for the μ-calculus can be solved efficiently if, and only if, this is the case for GL.

The results developed in this chapter also raise a new question regarding the structure of the μ-calculus variable hierarchy which will be answered in Chapter 5 together with the open question posed by Parikh in [55], whether Game Logic is equal in expressive power to the μ-calculus.

2.1 Syntax and Semantics

Syntactically, Game Logic extends the language of PDL by adding a dualisation operator for complex programs, conceived as games.

Definition 2.1.1 (Syntax of Game Logic). Starting from a set PROP of atomic propositions and a set ACT of atomic game actions, the expressions of GL are of two sorts, formulae and games, generated respectively by the following grammar:

$$\varphi := \bot \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\gamma\rangle\varphi$$
$$\gamma := a \mid \varphi? \mid \gamma;\gamma \mid \gamma \cup \gamma \mid \gamma^* \mid \gamma^{\mathrm{d}}$$

where $p \in \text{PROP}$ and $a \in \text{ACT}$.

The meaning of these expressions is defined in terms of neighbourhood models, i.e., higher-order transition structures that supply descriptions of the player's effectivities in atomic games, for each state. In place of effectivity functions $f : V \to \wp(\wp(V))$, we will use a relational encoding $F \subseteq V \times \wp(V)$ consisting of the pairs (v, Z) with $Z \in f(v)$.

Definition 2.1.2 (Effectivity relation). An *effectivity relation* over a set V of states is a relation $F \subseteq V \times \wp(V)$ closed under the following monotonicity condition: if $(v, Z) \in F$ and $Z \subseteq Z' \subseteq V$, then $(v, Z') \in F$.

We shall find it useful to think of an effectivity relation F as an operator from $\wp(V)$ into itself, mapping any set $Z \subseteq V$ to $F(Z) := \{ v \mid (v, Z) \in F \}$. Notice that, by monotonicity of the effectivity relation, the associated operator is monotone too. The natural composition of two operators $(F_1 \circ F_2)(Z) := F_1(F_2(Z))$ translates back into the relational view as

$$F_1 \circ F_2 := \{ (v, Z) \mid (v, F_2(Z)) \in F_1 \}.$$

Assuming that game atomic forms are determined, it is sufficient to model only the effectivity of one player explicitely.

Definition 2.1.3 (Neighbourhood model). A *neighourhood model* over a set PROP of atomic propositions and a set ACT of atomic actions is a structure

$$\mathcal{M} = \left(V, (F_a)_{a \in \text{ACT}}, (V_p)_{p \in \text{PROP}}\right)$$

where V is a set of states, $F_a \subseteq V \times \wp(V)$ are effectivity relations for all $a \in \text{ACT}$, and $V_p \subseteq A$ are monadic relations for all $p \in \text{PROP}$.

The relations F_a are intended to describe the effectivity of Angel in the atomic game a. As in the case of ordinary transition systems, the sets V_p consist of the states where p holds. Starting from these, the semantics of GL extends complex formulae to subsets of states and game expressions to effectivity functions over V.

Definition 2.1.4 (Semantics of Game Logic). Given a neighbourhood model $\mathcal{M}$ providing the meaning of primitive propositions $p \in \textsc{Prop}$ and actions $a \in \textsc{Act}$, formulae φ and game expressions γ extend to subsets $[\![\varphi]\!] \in V$ and effectivity relations $[\![\gamma]\!] \in V \times \wp(V)$ via simultaneous induction, as follows.

For game expressions, we set:

$$\begin{aligned}
[\![a]\!] &:= F_a;\\
[\![\varphi?]\!] &:= \{\, (v, Z) \mid v \in Z \cap [\![\varphi]\!] \,\};\\
[\![\gamma_1; \gamma_2]\!] &:= [\![\, \gamma_1]\!] \circ [\![\gamma_2]\!];\\
[\![\gamma_1 \cup \gamma_2]\!] &:= [\![\gamma_1]\!] \cup [\![\gamma_2]\!];\\
[\![\gamma^*]\!] &:= \{\, (v, Z) \mid v \in \mathrm{lfp}\big(X \mapsto Z \cup [\![\gamma]\!](X)\big) \,\};\\
[\![\gamma^{\mathrm{d}}]\!] &:= \{\, (v, Z) \mid (v, \overline{Z}) \notin [\![\gamma]\!] \,\}.
\end{aligned}$$

The boolean connectives are interpreted as usual; for the modal operator, we set:

$$[\![\langle\gamma\rangle\varphi]\!] := \{\, v \mid (v, [\![\varphi]\!]) \in [\![\gamma]\!] \,\} = [\![\gamma]\!]([\![\varphi]\!]).$$

To see that the definition is sound, observe that the monotonicity of effectivity relations is preserved along the composition process. In particular, this means that the operator $X \mapsto Z \cup [\![\gamma]\!](X)$ defining the extension of γ^* is monotone, and, hence, has a least fixed point. To recover the intuitive understanding of iteration, we can think of the inductive definition of this fixed point as the limit of the transfinite sequence:

$$\begin{aligned}
\mathrm{Attr}^0(Z) &:= Z\\
\mathrm{Attr}^{i+1}(Z) &:= Z \cup [\![\gamma]\!](\mathrm{Attr}^i(Z)), \quad \text{for all ordinals } i, \text{ and}\\
\mathrm{Attr}^{\lambda}(Z) &:= \bigcup_{i<\lambda} \mathrm{Attr}^i(Z), \quad \text{for limit ordinals } \lambda.
\end{aligned}$$

Each inductive stage $\mathrm{Attr}^{i+1}(Z)$ extends the previous one by adding those states from which Angel can attract the play γ into $\mathrm{Attr}^i(Z)$ (or the game breaks and Demon loses). By well-foundedness of the ordinals, it follows that from every state appearing along this increasing sequence, Angel has a strategy to achieve an outcome within Z by iterating γ finitely often. The set of states at which the sequence finally stabilises, coincides with the least fixed point of the operator $X \mapsto Z \cup [\![\gamma]\!](X)$. By determinacy we can conclude that from every state outside this fixed point, Demon has a strategy to keep any finite iteration of γ outside Z.

2.2 Interpretation over Kripke structures

When interpreted over neighbourhood models, Game Logic features interaction already at the elementary level of atomic games. However, as we are concerned with the expressive power inherent to the language rather than the underlying model, and because we wish to view Game Logic in the context of established formalisms for specifying interactive systems, we shall restrict ourself to interpretation of GL over Kripke structures.

The current understanding of transitions E_a as assertions about consequences of actions ("if action a occurs to the system in state v, then it shifts into some state $w \in vE_a$"), can be comprehended, in terms of games, as assertions about outcomes of one-player game forms ("when playing game a at state v, the player in turn can bring about any outcome $w \in vE_a$"). Concretely, this corresponds to casting transition relations $E_a \subseteq V \times V$ as effectivity functions:

$$F_a := \{\, (v, Z) \in V \times \wp(V) \mid (\exists w \in vE_a)\, w \in Z \,\}.$$

Via this translation, Kripke structures can be viewed as a subclass of neighbourhood models, where atomic games are, in fact, one-player games. All interactive aspects are, hence, controlled syntactically. Note that in the absence of alternation, i.e., when the dualisation operator is not involved, GL on these models is nothing but PDL. In the presence of alternation, instead, the expressiveness of GL increases significantly.

In Subection 1.2.2 we have argued that ΔPDL, the extension of PDL by a looping operator for expressing that a program can be executed infinitely often, is a powerful specification formalism which subsumes CTL* in expressive power. It turns out that ΔPDL can be translated into GL but not vice versa.

Proposition 2.2.1. *Over Kripke structures,* $\Delta\mathrm{PDL} < \mathrm{GL}$.

Proof. For the positive direction, notice that the GL-expression $(\gamma^{\mathrm{d}})^{*\mathrm{d}}$ corresponds to the looping operator $\Delta\gamma$ of ΔPDL.

On the negative side, Niwinski pointed out in [53] that the μ-calculus formula $\nu X.\langle a\rangle X \wedge \langle b\rangle X$ is not expressible in ΔPDL. Essentially, the formula describes the models able to simulate the following Kripke structure.

In the corresponding simulation game Challenger chooses, in each round, either action a or b, and Duplicater has to move to a corresponding successor in the model, infinitely often. At the attempt of describing this game, ΔPDL fails because it can only capture finitely many alternations. In GL, instead, the property can be easily described by interleaving iteration and dualisation: $\langle((a^{\mathrm{d}} \cup b^{\mathrm{d}})^*)^{\mathrm{d}}\rangle\bot$. □

At this point, the question arises, whether the ability of GL to describe alternation exhibits similar limitations as ΔPDL. The most promising approach to investigate this question is offered by the fine-structure of the μ-calculus alternation hierarchy.

2.2.1 Translating Game Logic into the μ-calculus

Since all operations used to define the semantics of GL appear as built-in operations in L_μ, it is an easy exercise to translate the syntax of GL into L_μ. Thereby, game expressions γ are translated into formulae $\eta_\gamma(Z)$ with a free fixed-point variable Z, so that on any Kripke structure $\mathcal{K}$, the operators $[\![\gamma]\!](\cdot)$ and $\eta^{\mathcal{K}}$ coincide. Furthermore, by repeatedly reusing variables, the image of this translation can be kept within the two-variable fragment of L_μ.

Proposition 2.2.2 ([58]). *Every* GL*-formula can be translated into an equivalent formula of the μ-calculus using at most two fixed-point variables.*

Proof. Let X and Y be two fixed point variables. To translate a formula $\psi \in \mathrm{GL}$ into L_μ, we construct three mappings $\cdot^X$, $\cdot^Y$, and $\cdot^\natural$ inductively over the subexpressions of ψ.

The operator translations $\cdot^X$ and $\cdot^Y$ associate to every occurring game expression γ, an L_μ-formula $\gamma^X(X)$ respectively $\gamma^Y(Y)$ with one free fixed-point variable:

$$
\begin{aligned}
g^X &:= \Diamond X & g^Y &:= \Diamond Y\\
(\gamma_1 \cup \gamma_2)^X &:= \gamma_1^X \vee \gamma_2^X & (\gamma_1 \cup \gamma_2)^Y &:= \gamma_1^Y \vee \gamma_2^Y\\
(\gamma_1;\gamma_2)^X &:= \gamma_1^X[X := \gamma_2^X] & (\gamma_1;\gamma_2)^Y &:= \gamma_1^Y[Y := \gamma_2^Y]\\
(\varphi?)^X &:= \varphi^\natural \wedge X & (\varphi?)^Y &:= \varphi^\natural \wedge Y\\
(\gamma^d)^X &:= \neg\gamma^X[X := \neg X] & (\gamma^d)^Y &:= \neg\gamma^Y[Y := \neg Y]\\
(\gamma^*)^X &:= \mu Y.X \vee \gamma^Y & (\gamma^*)^Y &:= \mu X.Y \vee \gamma^X
\end{aligned}
$$

The proposition translation $\cdot^\natural$ associates to every GL-formula an L_μ-sentence:

$$\begin{aligned} p^\natural &:= p \\ (\neg\varphi)^\natural &:= \neg\varphi^\natural \\ (\varphi_1 \vee \varphi_2)^\natural &:= \varphi_1^\natural \vee \varphi_2^\natural \\ (\langle\gamma\rangle\varphi)^\natural &:= \gamma^X[X := \varphi^\natural] \end{aligned}$$

Then, for every structure $\mathcal{K}$ and every subset Z of states, we have for all subexpressions γ and φ of ψ:

$$[\![\gamma]\!](Z) = [\![\gamma^X]\!]_{[X:=Z]}, \quad [\![\gamma]\!](Z) = [\![\gamma^Y]\!]_{[Z:=T]}, \quad \text{and } [\![\varphi]\!] = [\![\varphi^\natural]\!].$$

In particular, it follows that $\psi \in \mathrm{GL}$ and $\psi^\natural$ are equivalent. □

Notice, that the translation rule for nondeterministic choice introduces two occurrences of the free variable. While substituting the game modality, these are both replaced with the same expression, thus leading to a possible exponential blow-up of obtained L_μ-sentence. However, this phenomenon can be avoided by translating into the equational μ-calculus rather than its linear variant. Obviously, for the closure of the translation we obtain a size $|\mathrm{cl}(\psi^\natural)|$ which linearly bounded in the length of ψ.

2.2.2 Parity semantics for Game Logic

Despite its conceptual elegance, the interpretation of GL game expressions in terms of effectivity functions has a major disadvantage we work on Kripke structures: these objects are not representable within the system itself. Even if the effectivity relations for atomic games are interpretable as transitions, the extension of composed expressions will usually not fit any more into this interpretation. For example, the extension over the expression $a; a^\mathrm{d}$ over a Kripke structure as in Figure 2.1 does no not correspond to any transition relation over this system. On the one hand, this is because transitions represent one-player games while, in complex game expressions, alternation really matters. On the other hand, effectivity functions may encompass internal states, that do not correspond to states of the external Kripke structure over which they are interpreted.

To bridge this gap, we rephrase the semantics of GL in terms of parity games. In this way, we will be able to refer not only to the interpretation of a GL-formula on a

given Kripke structure, but also about the game-expressions occurring within this formula. Moreover, since parity games are themselves Kripke structures, we will be able to refer not only about the truth value of a sentence but also about its proof within the same formal system, in similar way as we do for the μ-calculus.

At this point we can rely on the games obtained via the translation of GL into L_μ. However, to access the details of these games, it is convenient to review their construction explicitly.

For better readability, let us agree on a precedence order over GL-operators, assuming that unary operators bind tighter than binary ones and that sequential composition ; binds tighter than the choice operator $\cup$. Additionally, we define dual operators as a shorthand:

$$\varphi_1 \wedge \varphi_2 := \neg(\neg\varphi_1 \vee \neg\varphi_2) \qquad \top := \neg\bot$$
$$\gamma_1 \cap \gamma_2 := (\gamma_1^{\mathrm{d}} \cup \gamma_2^{\mathrm{d}})^{\mathrm{d}} \qquad \gamma^{\circ} := (\gamma^{\mathrm{d}})^{*^{\mathrm{d}}}$$

To disentangle formulae and games, we transform each game in such a way that tests apply only to $\bot$, $\top$, atomic, or negated atomic propositions as follows:

$$(\varphi_1 \vee \varphi_2)? \equiv \varphi_1? \cup \varphi_2? \qquad (\neg\varphi)? \equiv (\varphi?^{\,\mathrm{d}}; \bot?) \cap \top?$$
$$(\varphi_1 \wedge \varphi_2)? \equiv \varphi_1? \cap \varphi_2? \qquad (\langle\gamma\rangle\varphi)? \equiv \gamma; \varphi?$$

Further, we can exploit the following equivalences to put any GL-formula into a *negation normal form* where negation applies only to atomic propositions, and dualisation applies only to atomic games or surrender ($\bot?$).

$$\neg\langle\gamma\rangle\varphi \equiv \langle\gamma^{\mathrm{d}}\rangle\neg\varphi \qquad \varphi?^{\mathrm{d}} \equiv (\neg\varphi?\,; \bot?^{\,\mathrm{d}}) \cup \top?$$
$$(\gamma_1; \gamma_2)^{\mathrm{d}} \equiv \gamma_1^{\mathrm{d}}; \gamma_2^{\mathrm{d}} \qquad (\gamma^*)^{\mathrm{d}} \equiv (\gamma^{\mathrm{d}})^{\circ}$$

The positions of the parity game associated to a GL-formula and a given Kripke structure will comprise both external states of the system and internal states stemming from the formula.

Definition 2.2.3 (Closure). Given a formula $\psi \in \mathrm{GL}$ in negation normal form, we define its *closure* $\mathrm{cl}(\psi)$ as the smallest set which contains ψ and is closed under the following operations:

(i) taking of subformulae: for each $\varphi \in \mathrm{cl}(\psi)$ any subformula η of φ is also contained in $\mathrm{cl}(\psi)$;

(ii) choice: for each $\langle \gamma_1 \cup \gamma_2 \rangle \varphi \in \mathrm{cl}(\psi)$ we have $\{\langle \gamma_1 \rangle \varphi, \langle \gamma_2 \rangle \varphi\} \subseteq \mathrm{cl}(\psi)$ and likewise for $\cap$;

(iii) unrolling: for each $\langle \gamma^* \rangle \varphi \in \mathrm{cl}(\psi)$ also $\langle \gamma; \gamma^* \rangle \varphi \in \mathrm{cl}(\psi)$ and likewise for $^\circ$;

(iv) splitting: for each $\langle \gamma_1; \gamma_2 \rangle \varphi \in \mathrm{cl}(\psi)$ also $\langle \gamma_1 \rangle \langle \gamma_2 \rangle \varphi \in \mathrm{cl}(\psi)$.

Observe that $|\mathrm{cl}(\psi)|$ is linearly bounded by the number of symbols in ψ.

Now, we are ready now to define the semantic games for Game Logic.

Definition 2.2.4 (Model checking game for GL). To any Kripke structure $\mathcal{K}$, u and any formula $\psi \in \mathrm{GL}$, we associate a parity game $\mathcal{G}(\mathcal{K}, \psi, u)$ with positions

$$V := \{(v, \varphi) \ : \ \psi \in \mathrm{cl}(\psi) \text{ and } v \in V\}.$$

Thereof, Player 0 holds all positions where the formula is of the shape

$$\bot,\ \varphi_1 \vee \varphi_2,\ \langle \alpha ? \rangle \varphi,\ \langle g \rangle \varphi,\ \langle \gamma_1 \cup \gamma_2 \rangle \varphi,\ \text{or } \langle \gamma^* \rangle \varphi,$$

where α stands for $\bot$, $\top$, or atomic propositions, possibly negated. Additionally, V_0 includes

$$\{(v, p) \ : \ v \notin V_p\} \cup \{(v, \neg p) \ : \ v \in V_p\}.$$

The remaining positions belong to Player 1.

All plays start at position (ψ, u). The transitions are given as follows.

- From positions $(v, \bot)$, $(v, \top)$, (v, p), or $(v, \neg p)$ no moves can be done.
- From $(v, \varphi_1 \vee \varphi_2)$ or $(v, \varphi_1 \wedge \varphi_2)$ two transitions lead to (v, φ_1) and (v, φ_2).
- From $(v, \langle \alpha ? \rangle \varphi)$ there is a transition to (v, φ), if one of the following holds:
  - α is $\top$ or $\top^{\mathrm{d}}$;
  - α is p or p^d and $a \in V_p$;
  - α is $\neg p$ or $(\neg p)^d$ and $a \notin V_p$.

  Otherwise, no moves can be done.
- From $(v, \langle \gamma_1 \cup \gamma_2 \rangle \varphi)$ transitions lead to $(v, \langle \gamma_1 \rangle \varphi)$ and $(v, \langle \gamma_2 \rangle \varphi)$.
- From $(v, \langle a \rangle \varphi)$ there are transitions to each of $\{(w, \varphi) \mid (v, w) \in E_a\}$, for all $a \in \textsc{Act}$.
- From $(v, \langle \gamma^* \rangle \varphi)$ or $(v, \langle \gamma^\circ \rangle \varphi)$ two transitions lead to (v, φ) and respectively $(v, \langle \gamma; \gamma^* \rangle \varphi)$.

- From $(v, \langle \gamma_1; \gamma_2 \rangle)$ there is a transition to $(v, \langle \gamma_1 \rangle \langle \gamma_2 \rangle \varphi)$.

In order to determine the priority assignment, we first introduce a measure for the alternated nesting of iteration in GL-games.

Definition 2.2.5 (Star hierarchy). For GL-games in negation normal form we define the following *star alternation hierarchy*:

(i) The first level of the hierarchy, $\Sigma_0^* = \Pi_0^*$, consists of the *- and $^\circ$-free games.

(ii) For every higher level, Σ_{i+1}^* is formed by closing $\Sigma_i^* \cup \Pi_i^*$ under $;, \cap, \cup$ and *.

(iii) The level Π_{i+1}^* is obtained dually, by closing under $^\circ$ instead of *.

Only the positions of the form $(v, \langle \gamma^* \rangle \varphi)$ or $(v, \langle \gamma^\circ \rangle \varphi)$ receive significant priority colourings. Towards this, we look at the least i such that $\gamma \in \Sigma_i \cup \Pi_i$ and assign $(v, \langle \gamma^* \rangle \varphi)$ to Ω_{2i+1} or, respectively, $(v, \langle \gamma^\circ \rangle \varphi)$ to Ω_{2i+2}. All remaining positions are set to some irrelevant, high priority.

When we refer to the game $\mathcal{G}(\mathcal{K}, \psi, u)$, (ψ, u) we will usually not mention the root (ψ, u) explicitly and write $\mathcal{G}(\mathcal{K}, \psi, u)$, or simply $\mathcal{G}(\mathcal{K}, \psi)$.

Essentially, the parity game obtained in this way for ψ follows the construction of the semantic game for the translation of ψ into L_μ. Therefore, the correctness of the construction follows immediately from the correctness of the translation.

Proposition 2.2.6. *A formula $\psi \in$ GL holds in a Kripke structure $\mathcal{K}, u$ if, and only if, Player 0 has a winning strategy in the game $\mathcal{G}(\mathcal{K}, \psi, u)$.*

Notice, that the number of positions in $G(\mathcal{K}, \psi, u)$ is bounded by $O(|A| \cdot |\psi|)$. Since the problem, whether Player 0 has a winning strategy in a parity game is known to be in NP ∩ Co-NP we can immediately conclude

Corollary 2.2.7. *The model checking problem for* GL *over finite structures is in* NP ∩ Co-NP.

2.3 Expressing parity semantics

In this section we will show that Game Logic is able to express the winning conditions for parity games. Since these games capture the semantics of GL, this means that the formalism of GL is strong enough to express its own definition of

truth via the associated game structure. Moreover, it can also express the truth of L_μ-formulae via these games. Intuitively, this indicates that the power of alternation of Game Logic cannot be lower than that of the μ-calculus.

To make this precise, we first need to define our measure of alternation in GL.

The alternation levels of the L_μ hierarchy record the number of nested alternations between least and greatest fixed point operators. For formulae in negation normal form, this corresponds to the nesting of * and $^\circ$ operators within the game modalities. We can thus extend the star hierarchy over games from Definition 2.2.5 to a hierarchy over formulae.

Definition 2.3.1 (Alternation hierarchy for GL). The *alternation hierarchy* of GL is the sequence $(\Sigma_i)_{i<\omega}$ of sets consisting of formulae ψ in negation normal form, such that

$$\psi \in \Sigma_i \quad \text{iff} \quad \{\, \gamma \,:\, \langle\gamma\rangle\varphi \in \mathrm{cl}(\psi) \,\} \subseteq \Sigma_i^* \cup \Pi_i^*.$$

Next, we construct a GL formula to express the winning condition for Player 0 in a parity game $\mathcal{K} = (V, V_0, E, \Omega)$ with n priorities. In line with our remarks to the construction of the corresponding formula in the μ-calculus (see Definition 1.3.9), we will compose, for every priority i, a game γ_i corresponding to a subgame of the parity game $\mathcal{K}$. Our intention is to tailor γ_i so that it reflects the effectivity of Angel to ensure in each play of the parity game $\mathcal{K}$ that he either wins, or the play reaches a priority less than i. Then, γ_1 describes its effectivity in the whole parity game.

Let a be the unique atomic game action corresponding to a move from one game position to another. Then, Angel's effectivity in a single game move is described by the composite game:

$$f := V_0?; a \cup V_1?; a^{\mathrm{d}}$$

Assuming n is odd, consider the sequence $(\gamma_i)_{1\leq i\leq n}$ of GL-games starting with

$$\gamma_n := (\Omega_n?; f)^*; \Omega_{<n}?$$

and, for any even index $i < n$,

$$\gamma_i := \big(\Omega_{\geq i}?^{\,\mathrm{d}}; (\Omega_i?; f \cup \Omega_{>i}?; \gamma_{i+1})\big)^\circ; \Omega_{<i}?^{\,\mathrm{d}}$$

while for $i < n$ odd,

$$\gamma_i := (\Omega_i?; f \cup \Omega_{>i}?; \gamma_{i+1})^*; \Omega_{<i}?$$

Before we proceed, let us see which options the players actually have in a game γ_i. First, for a player to hold the star (or circle) means only little choice here, since he *can* stop iterating γ_i only when some priority less then i is seen. This is required by the guards $\Omega_{<i}?$ and $\Omega_{<i}?^{\,\mathrm{d}}$ at the exit point of the iteration. But note that, in that case he is forced to stop iterating, otherwise he loses. For Demon this is stated explicitly by the condition $\Omega_{\geq i}$ at the entry of the iteration, when i is even. For Angel instead, this guard needs not to be set as he has to make his choices in such a way that f is finally being played and it is pointless for him to cheat at that point: if the current position in $\mathcal{K}$ has priority j he will always choose towards reaching the subgame $(\Omega_j?; f)$ in γ_j. Since all $\cup$ choices are determined by the value of j, entering a game γ_i with $i < j$ would lead Player 0 to fail the test after his next $\cup$ choice. Thus, the actual choices take place in the structure, that is, when f is being played.

We are interested in γ_1. Please note, that the meaning of any formula $\langle\gamma_1\rangle\varphi$ does not depend on φ, since γ_1 is either finished by surrender or it never ends. Thus we can safely choose $\varphi := \bot$. Let us denote $\langle\gamma_1\rangle\bot$ by W^n_*.

Example. For $n = 3$ we obtain, by replacing $\Omega_{<1}$ with $\bot$ and omitting the ; operator,

$$\left\langle\left(\Omega_1?f \cup \Omega_{>1}?\left(\Omega_{\geq 2}?^{\,\mathrm{d}}\left(\Omega_2?f \cup \Omega_{>2}?(\Omega_3?f)^*\Omega_{<3}?\right)\right)^\circ \Omega_{<2}?^{\,\mathrm{d}}\right)^* \bot\right\rangle\bot$$

Proposition 2.3.2. *For every parity game $\mathcal{K}$, u of index n we have*

$$\mathcal{K}, u \vDash \mathrm{W}^n_* \quad \textit{iff} \quad \mathcal{K}, u \vDash \mathrm{W}^n.$$

Proof. Our intention is to translate the proof of the $\mathrm{W}^n \in \mathrm{L}_\mu$ on $\mathcal{K}, v$ into a proof of $\mathrm{W}^n_* \in \mathrm{GL}$ on the same structure, and vice versa. Towards this, we look at the model checking games resulting from the two formulae $\mathcal{G} := \mathcal{G}(\mathcal{K}, \mathrm{W}^n, u)$, and respectively, $\mathcal{G}_* := \mathcal{G}(\mathcal{K}, \mathrm{W}^n_*, u)$. Herein, the proofs appear as winning strategies. Thus, we can rephrase our aim in terms of parity games: If Player 0 has a winning strategy in $\mathcal{G}$ then he also has a strategy in $\mathcal{G}_*$ (and we are able to construct it), and vice versa.

Let us assume that $\mathcal{K}, v$ satisfies W^n, i.e., Player 0 has a winning strategy σ in $\mathcal{K}, u$. Hence, he also has a winning strategy τ in $\mathcal{G}$. Observe that the relevant advices of τ are all transitions of the type $(v, \Diamond Z_i) \to (w, Z_i)$ where is i the priority of v. In other words, the strategy τ of Player 0 in $\mathcal{G}$ is uniquely determined by his strategy σ in $\mathcal{K}, u$.

Now, getting back to GL, in the light of the above remarks concerning the freedom of choice in γ_i, we can see that for any winning strategy of Player 0 in $\mathcal{G}_*$ the relevant

choices are structure choices of the type $(v, a; \xi_i) \to (v, \xi_i)$ where, for i the priority of v (assumed odd),

$$\xi_i = \langle (\Omega_i?; f \cup \Omega_{>i}?; \gamma_{i+1})^*; \Omega_{<i}?; \gamma_{i-1}; \gamma_{i-2}; \cdots \gamma_1 \rangle \bot.$$

Let us consider the strategy τ_* for Player 0 in $\mathcal{G}_*$ which works like σ (and τ) on structure choices while preventing him on formula choices ($*$ or $\cap$) to lose within the next two steps.

Clearly, τ_* carries precisely the same information as τ. In fact, both strategies mirror the winning strategy σ on $\mathcal{K}, u$. It is easy to verify that the priorities in $\mathcal{G}$ and $\mathcal{G}_*$ are assigned in a compatible way, such that the set of plays according to these strategies are essentially the same for both model checking games, consequently, all wins for Player 0.

By the same token, we can also show conversely, that a winning strategy for Player 0 in $\mathcal{G}_*$ can be transferred via projection onto $\mathcal{K}, u$ to a winning strategy in $\mathcal{G}$. This concludes our proof. □

Since for every number n, the formula W_n describing the parity conditions for a game with n priorities is hard for the n-th level of the μ-calculus alternation hierarchy, the translation of these formulae into GL implies the following interesting result.

Theorem 2.3.3. *No finite level of the μ-calculus alternation hierarchy captures the expressive power of* GL.

Moreover, we can conclude that the strictness of the alternation hierarchy for the μ-calculus, formulated Theorem 1.3.11, carries over to Game Logic.

Theorem 2.3.4. *The alternation hierarchy of Game Logic is strict.*

Proof. Obviously, W^n_* is contained in Σ_n. Since the translation of GL-formulae into L_μ preserves the alternation level, that is, the number of alternated nestings of * and ° translates into the same number of nested least and greatest fixed point operators, and the L_μ alternation hierarchy is strict, no GL-formula $\psi \in \Sigma_{n-1}$ can be equivalent to W^n_*. □

Finally, observe that the length of W^n_* is at most quadratic in n. Since the model-checking problem for an L_μ-formula ψ of alternation level n in a structure $\mathcal{K}, u$ can be reduced to the problem of establishing whether Player 0 has a winning strategy

in $\mathcal{G}(\mathcal{K}, \mathrm{W}^n, u)$, or equivalently, to the model-checking problem for Game Logic $\mathcal{G}(\mathcal{K}, \psi, u) \vDash \mathrm{W}^n_*$, we obtain the following connection between the complexity of GL and L_μ.

Theorem 2.3.5. *Model-checking for the μ-calculus can be performed in polynomial time, if and only if, this is the case for Game Logic.*

Although the above results show that we can define classes in GL which are arbitrarily hard for L_μ, an indication not to underestimate its expressive power, the question whether Game Logic attains the full power of L_μ remains open. A definitive answer to this question results from our investigation of the L_μ variable hierarchy in Chapter 5.

2.4 Hierarchies within the μ-calculus

Formulations of the parity winning condition in L_μ have been used by several authors as a fundamental tool in different contexts. In [21], Emerson and Jutla proved that parity games are determined by characterising the winning positions of each player in L_μ. Since these characterisations are complements of each other, it follows that there are no undetermined positions. Via the correspondence established by Gurevich and Harrington [29] between parity games and Rabin tree automata, this also implies that Rabin automata are closed under complement, thus yielding a considerable simplification for the key argument in the proof of Rabin's decidability theorem [61] of the MSO-theory of trees. As an even more powerful result, in [76] Walukiewicz proved that the MSO-theory of iterated structures is decidable, using a very subtle refinement of this technique (see also [7] for a treatment on this topic). Considering parity games over infinite push-down graphs [75], Walukiewicz showed that these games are determined with automatic strategies, by deriving a progress measure from an L_μ formulation of the parity winning condition. Finally, the witnessing formulae developed in the different proofs of the L_μ alternation hierarchy by Arnold, Lenzi, and Bradfield [1, 15, 16, 13, 49] all variants of this formula.

Interestingly, in each of these contexts, the characterisation of a parity game with n priorities was written over n variables. However, our formulation of this property in Game Logic, which is embedded in the two-variable fragment of L_μ, shows that already two variables are sufficient to express this property.

Corollary 2.4.1. *The set of winning positions for Player* 0 *in a parity game with n priorities can be characterised by a μ-calculus formula using at most two variables, for any number n.*

Since we believe this matter to be interesting beyond our investigation of Game Logic, we present here an explicit L_μ-formulation of the parity condition in a game with n priorities, over two variables.

Using the notation from Definition 1.3.9 for any n, the sequence of formulae φ_i for $i = n, \ldots, 1$ is constructed inductively as follows:

$$\varphi_i(X) := \mu Y.\big((\Omega_i \wedge \rhd Y) \vee (\Omega_{<i} \wedge X) \vee (\Omega_{>i} \wedge \varphi_{i+1}(Y))\big)$$

when i is odd and, otherwise,

$$\varphi_i(Y) := \nu X.\big((\Omega_i \wedge \rhd X) \vee (\Omega_{<i} \wedge Y) \vee (\Omega_{>i} \wedge \varphi_{i+1}(X))\big).$$

Recall that empty disjunctions as $\Omega_{<1}$ or $\Omega_{>n}$ are interpreted as false.

The formula φ_1 obtained in this way characterises the winning positions of Player 0 in a party game with n, obviously with not more than two variables.

Example. For $n = 3$ we get

$$\mu X.\big(\Omega_1 \rhd X \vee \Omega_{>1}\nu Y.\big(\Omega_2 \rhd Y \vee \Omega_{<2}X \vee \Omega_{>2}(\mu X.\Omega_3 \rhd X \vee \Omega_{<3}Y)\big)\big).$$

Particularly, this implies that, in terms of alternation, the entire complexity of the μ-calculus is encountered already in the second level of its variable hierarchy.

Corollary 2.4.2. *The alternation hierarchy of the two-variable fragment of* L_μ *is strict and not contained in any finite level of the μ-calculus alternation hierarchy.*

3 Path games

Parikh's Game Logic considered in the previous chapter provides an interpretation of modalities by means of games. In the present chapter, we study a different way of defining quantification in terms of games. Here, instead of transitions, the target object will be paths resulting from infinitely many interactions between two players.

In the games underlying to this prospective, the players select, in each move, a path of arbitrary finite length, rather than just an edge. The outcome of a play is an infinite path, and the winning condition is given by a formula from MSO, LTL, or FO. Such games have a long tradition in descriptive set theory (in the form of Banach-Mazur games) and have recently been shown to have interesting application for planning in nondeterministic domains. In a first instance, we will investigate the structure of winning strategies for certain subclasses of path games.

With each formalism defining a winning condition on infinite paths, we then associate a logic over graphs, defining the winning regions of the associated path games. We investigate the expressive power of the logics obtained in this way. It turns out that the winning regions of path games with MSO-winning conditions are definable in L_μ. Further, if the winning condition is defined in first-order logic (over paths), then the winning regions are definable in monadic path logic, or, for a large class of games, even in first-order logic. As a consequence, winning regions of LTL path games are definable in CTL*.

3.1 Origins

Path games have been studied in descriptive set theory, in the form of *Banach-Mazur games* (see [42, Chapter 6] or [43, Chapter 8.H]). In their original variant (see [51, pp. 113–117], the winning condition is a set W of real numbers; in the first move, one of the players selects an interval d_1 on the real line, then his opponent chooses an interval $d_2 \subset d_1$, then the first player selects a further refinement $d_3 \subset d_2$ and so

on. The first player wins if the intersection $\bigcap_{n\in\omega} d_n$ of all intervals contains a point of W, otherwise his opponent wins. This game is essentially equivalent to a path game on the infinite binary tree T^2 or the ω-branching tree T^ω. An important issue in descriptive set theory is determinacy: to characterise the winning conditions W such that one of the two players has a winning strategy for the associated game. This is closely related to topological properties of W (see Section 3.3).

In a quite different setting, Pistore and Vardi [59] have used path games for task planning in nondeterministic domains. In their scenario, the desired infinite behaviour is specified by formulae in linear temporal logic LTL, and it is assumed that the outcome of actions may be nondeterministic; hence a plan does not have only one possible execution path, but an execution tree. Between weak planning (some possible execution path satisfies the specification) and strong planning (all possible outcomes are consistent with the specification) there is a spectrum of intermediate cases such as strong cyclic planning: every possible partial execution of the plan can be extended to an execution reaching the desired goal. In this context, planning can be modelled by a game between a friendly player E and a hostile player A selecting the outcomes of nondeterministic actions. The game is played on the execution tree of the plan, and the question is whether the friendly player E has a strategy to ensure that the outcome (a path through the computation tree) satisfies the given LTL-specification. In contrast to the path games arising in descriptive set theory, the main interest here are path games with finite alternations between players. For instance, strong cyclic planning corresponds to a AE^ω-game where a single move by A is followed by actions of E. Also the relevant questions are quite different: Rather than determinacy (which is clear for winning conditions in LTL) algorithmic issues play the central role. Pistore and Vardi show that the planning problems in this context can be solved by automata-based methods in 2Exptime.

Outline of this chapter. Here we consider path games in a general, abstract setting, but with emphasis on definability and complexity issues. In Section 3.2 we describe path games and discuss their basic structure. In Section 3.3 we review the classical results on determinacy of Banach-Mazur games. We then study in Section 3.3.1 path games that are positionally determined, i.e., admit winning strategies that only depend on the current position, not on the history of the play. In Section 3.4 we investigate definability issues. We are interested in the question how the logical complexity of defining a winning condition (a property of infinite paths) is related to the logical complexity of defining who wins the associated game

(a property of game graphs). In particular, we will see that the winner of path games with LTL winning conditions is definable in CTL*.

3.2 Path games and their values

Path games are a class of zero-sum infinite two-player games with complete information, where moves of players consist of selecting and extending finite paths through a graph. The players will be called Ego and Alter (in short E and A). All plays are infinite, and there is a utility function u, defining for each play a real number. The goal of Ego is to maximise the payoff while Alter wants to minimise it.

A strategy for a player is a function, assigning to every initial segment of a play a next move. Given a strategy f for Ego and a strategy g for Alter in a game $\mathcal{G}$, we write $f \hat{} g$ for the unique play defined by f and g, and $u(f \hat{} g)$ for its utility. The values of a game $\mathcal{G}$, from the point of view of Ego and Alter, respectively, are

$$e(\mathcal{G}) := \max_f \min_g u(f \hat{} g) \quad \text{and} \quad a(\mathcal{G}) := \min_g \max_f u(f \hat{} g).$$

A game is *determined* if $e(\mathcal{G}) = a(\mathcal{G})$. In the case of win-or-lose games, where the utility of any play is either 0 or 1, this amounts to saying that one of the two players has a winning strategy. For two games $\mathcal{G}$ and $\mathcal{H}$ we write $\mathcal{G} \preceq \mathcal{H}$ if $e(\mathcal{G}) \leq e(\mathcal{H})$ and $a(\mathcal{G}) \leq a(\mathcal{H})$. Finally, $\mathcal{G} \equiv \mathcal{H}$ if $\mathcal{G} \preceq \mathcal{H}$ and $\mathcal{H} \preceq \mathcal{G}$.

Let $G = (V, F, v)$ be an arena consisting of a directed graph (V, F) without terminal nodes, a distinguished start node v, and let $u : V^\omega \to \mathbb{R}$ be a utility function that assigns a real number to each infinite path through the graph.

We denote a move where Ego selects a finite path of length ≥ 1 by E and an ω-sequence of such moves by E^ω; for Alter, we use corresponding notation A and A^ω. Hence, for any arena G and utility function u we have the following games.

- $(EA)^\omega(G, u)$ and $(AE)^\omega(G, u)$ are the path games with infinite alternation of finite path moves.
- $(EA)^k E^\omega(G, u)$ and $A(EA)^k E^\omega(G, u)$, for arbitrary $k \in \mathbb{N}$, are the games ending with an infinite path extension by Ego.
- $(AE)^k A^\omega(G, u)$ and $E(AE)^k A^\omega(G, u)$ are the games ending with an infinite path extension by Alter.

All these games together form the collection $\mathrm{Path}(G, u)$ of *path games*. (Obviously two consecutive finite path moves by the same players correspond to a single move, so there is no need for prefixes containing EE or AA.)

It turns out that this infinite collection of games collapses to a finite lattice of just eight different games. This has been observed independently by Pistore and Vardi [59].

Theorem 3.2.1. *For every arena G and every utility function u, we have*

$$\begin{array}{ccccc} E^\omega(G,u) & \geq & EAE^\omega(G,u) & \geq & AE^\omega(G,u) \\ & & \wedge\mid & & \wedge\mid \\ & & (EA)^\omega(G,u) & \geq & (AE)^\omega(G,u) \\ & & \wedge\mid & & \wedge\mid \\ & & EA^\omega(G,u) & \geq & AEA^\omega(G,u) \quad \geq \quad A^\omega(G,u) \end{array}$$

Further, every path game $\mathcal{H} \in \mathrm{Path}(G, u)$ is equivalent to one of these eight games.

Proof. The comparison relations in the diagram follow by trivial arguments. We just illustrate them for one case. To show that $\mathcal{G} \geq \mathcal{H}$ for $\mathcal{G} = EAE^\omega(G, u)$ and $\mathcal{H} = (EA)^\omega(G, u)$, consider first an optimal strategy f of Ego in $\mathcal{H}$, with $e(\mathcal{H}) = \min_g P(f \hat{} g)$. Ego can use this strategy also for $\mathcal{G}$: he just plays as if he would play $\mathcal{G}$, making an arbitrary move whenever it would be A's turn in $\mathcal{H}$. Any play in $\mathcal{G}$ that is consistent with this strategy, is also a play in $\mathcal{H}$ that is consistent with f, and therefore has utility at least $e(\mathcal{H})$. Hence $e(\mathcal{G}) \geq e(\mathcal{H})$. Second, consider an optimal strategy g of Alter in $\mathcal{G}$, with $a(\mathcal{G}) = \max_f P(f \hat{} g)$. In $\mathcal{H} = (EA)^\omega(G, u)$, Alter answers the first move of E as prescribed by g, and moves arbitrarily in all further moves. Again, every play that can be produced against this strategy is also a play of $\mathcal{G}$ that is consistent with g, and therefore has utility at most $a(\mathcal{G})$. Hence $a(\mathcal{G}) \geq a(\mathcal{H})$. In all other cases the arguments are analogous.

To see that any other path game over G is equivalent to one of those displayed, it suffices to show that

(1) $(EA)^k E^\omega(G, u) \equiv EAE^\omega(G, u)$, for all $k \geq 1$, and

(2) $A(EA)^k E^\omega(G, u) \equiv AE^\omega(G, u)$, for all $k \geq 0$.

By duality, we can then infer the following equivalences:

(3) $(AE)^k A^\omega(G,u) \equiv AEA^\omega(G,u)$ for all $k \geq 1$, and

(4) $E(AE)^k A^\omega(G,u) \equiv EA^\omega(G,u)$ for all $k \geq 0$.

The equivalences (1) and (2) follow with similar reasoning as above. Ego can modify a strategy f for $EAE^\omega(G,u)$ to a strategy for $(EA)^k E^\omega(G,u)$. He chooses the first move according to f and makes arbitrary moves the next $k-1$ times; he then considers the entire $A(EA)^{k-1}$-sequence of moves, which were played after his first move, as one single move of A in $EAE^\omega(G,u)$ and completes the play again according to f. The resulting play of $(EA)^k E^\omega(G,u)$ is a consistent play with f in $EAE^\omega(G,u)$. Conversely a strategy of Ego for $(EA)^k E^\omega$ also works if his opponent lets Ego move for him in all moves after the first one, i.e., in the game $EAE^\omega(G,u)$. This proves that the e-values of the two games coincide. All other equalities are treated in a similar way. □

The question arises whether the eight games displayed in the diagram are really different or whether they can be collapsed further. The answer depends on the game graph and the utility function, but for each comparison $\geq$ in the diagram we find simple cases where it is strict. Indeed, standard winning conditions $W \subseteq \{0,1\}^\omega$ (defining the utility function $u(\pi) = 1$ if $\pi \in W$, and $u(\pi) = 0$ otherwise) show that the eight games in the diagram are distinct on appropriate game graphs. Let us consider here the completely connected graph with two nodes 0 and 1.

If the winning condition requires some initial segment then Ego wins the path games where he moves first and loses those where Alter moves first. Thus, starting conditions separate the left half of the diagram from the right one.

Reachability conditions and safety conditions separate games in which only one player moves, i.e., with prefix E^ω or A^ω respectively, from the other ones.

A game with a Büchi condition is won by Ego if he has infinite control and lost if he only has a finite number of finite moves (prefix ending with A^ω). Similarly, Co-Büchi conditions separate the games which are controlled by Ego from some time onwards (with prefix ending in E^ω) from the others.

3.3 Determinacy

From now on we consider win-or-lose games, with a winning condition given by a set of plays W. Player E wins the path game if the resulting infinite path belongs to W, otherwise Player A wins.

The topological properties of winning conditions W implying that the associated path games are determined are known from descriptive set theory. We just recall the basic topological notions and the results. Then, we will proceed to the issue of *memoryless determinacy*, i.e. to the question which path games admit winning strategies that only depend on the current position, not on the history of the play.

Note that path games with only finite alternations between the two players are trivially determined, for whatever winning condition; hence we restrict attention to path games with prefix $(EA)^\omega$ or $(AE)^\omega$, and by duality, it suffices to consider $(EA)^\omega$. By unravelling the game graph to a tree, we can embed any game $(EA)^\omega(G, W)$ in a Banach-Mazur game over the ω-branching tree T^ω. The determinacy of Banach-Mazur games is closely related to the Baire property, a notion that arose from topological classifications due to René Baire.

Topology. We consider the space B^ω of infinite sequences over a set B, endowed with the topology whose basic open sets are $O(x) := x \cdot B^\omega$, for $x \in B^*$. A set $L \subseteq B^\omega$ is *open* if it is a union of sets $O(x)$, i.e., if $L = W \cdot B^\omega$ for some $W \subseteq B^*$. A tree $T \subseteq B^*$ is a set of finite words that is closed under prefixes. It is easily seen that $L \subseteq B^\omega$ is *closed* (i.e., the complement of an open set) if L is the set of infinite branches of some tree $T \subseteq B^*$, denoted $L = [T]$. This topological space is called *Cantor space* in case $B = \{0, 1\}$, and *Baire space* in case $B = \omega$.

The class of *Borel sets* is the closure of the open sets under countable union and complementation. Borel sets form a natural hierarchy of classes $\mathbf{\Sigma}^0_\eta$ for $1 \leq \eta < \omega_1$, whose first levels are

$\mathbf{\Sigma}^0_1$ (or G) : the open sets
$\mathbf{\Pi}^0_1$ (or F) : the closed sets
$\mathbf{\Sigma}^0_2$ (or F_σ) : countable unions of closed sets
$\mathbf{\Pi}^0_2$ (or G_δ) : countable intersections of open sets

In general, $\mathbf{\Pi}^0_\eta$ contains the complements of the $\mathbf{\Sigma}^0_\eta$-sets, $\mathbf{\Sigma}^0_{\eta+1}$ is the class of countable unions of $\mathbf{\Pi}^0_\eta$-sets, and $\mathbf{\Sigma}^0_\lambda = \bigcup_{\eta<\lambda} \mathbf{\Sigma}^0_\eta$ for limit ordinals λ.

We recall that a set X in a topological space is *nowhere dense* if its closure does not contain a non-empty open set. A set is *meager* if it is a union of countably many nowhere dense sets and it has the *Baire property* if its symmetric difference with some open set is meager. In particular, every Borel set has the Baire property.

We are now ready to formulate the Theorem of Banach and Mazur (see e.g. [42, 43]). To keep in line with our general notation for path games we write

$(EA)^\omega(T^\omega, W)$ for the Banach-Mazur game on the ω-branching tree with winning condition W.

Theorem 3.3.1 (Banach-Mazur). (i) *Player A has a winning strategy for the game $(EA)^\omega(T^\omega, W)$ if, and only if, W is meager.*

(ii) *Player E has a winning strategy for $(EA)^\omega(T^\omega, W)$ if, and only if, there exists a finite word $x \in \omega^*$ such that $x \cdot \omega^\omega \setminus W$ is meager (i.e., W is co-meager in some basic open set).*

As a consequence, it can be shown that for any class $\Gamma \subseteq (\omega^\omega)$ that is closed under complement and under union with open sets, all games $(EA)^\omega(T^\omega, W)$ with $W \in \Gamma$ are determined if, and only if, all sets in Γ have the Baire property. Since Borel sets have the Baire property, it follows that Banach-Mazur games are determined for Borel winning conditions. (Via a coding argument, this can also been easily derived form Martin's Theorem, saying that Gale-Stewart games with Borel winning conditions are determined.)

Standard winning conditions used in applications (in particular the winning conditions that can be described in S1S, i.e., MSO interpreted over paths) are contained in very low levels of the Borel hierarchy. Hence all path games of this form are determined.

3.3.1 Memoryless determinacy

In general, winning strategies can be very complicated. As in the case of parity games considered in the previous chapter, we are particularly interested in memoryless, or positional strategies which only depend on the current position, not on the history of the play. On a game graph $G = (V, F)$ a memoryless strategy has the form $f : V \to V^*$ assigning to every move v a finite path from v through G.

To start, we present a simple example of a path game, that is determined, but does not admit a memoryless strategy.

Example. Let G_2 be the completely connected directed graph with nodes 0 and 1, and let the winning condition for Ego be the set of infinite sequences with infinitely many initial segments that contain more ones than zeros. Clearly, Ego has a winning strategy for $(EA)^\omega(G, W)$, but not a memoryless one.

Note that this winning condition is on the $\mathbf{\Pi}_2$-level of the Borel hierarchy. In fact, this is the lowest level with such an example.

Proposition 3.3.2. *If Ego has a winning strategy for a path game* $(EA)^\omega(G, W)$ *with* $W \in \mathbf{\Sigma}^0_2$*, then he also has a memoryless winning strategy.*

Proof. Let $G = (V, F)$ be the game graph. Since W is a countable union of closed sets, we have $W = \bigcup_{n<\omega}[T_n]$ where each $T_n \subseteq V^*$ is a tree. Further, let f be any (non-positional) winning strategy for Ego. We claim that, in fact, Ego can win with one move.

We construct this move by induction. Let x_1 be the initial path chosen by Ego according to f. Let $i \geq 1$ and suppose that we have already constructed a finite path $x_i \notin \bigcup_{n<i} T_n$. If $x_i y \in T_i$ for all finite y, then all infinite plays extending x_i remain in W, hence Ego wins with the initial move $w = x_i$. Otherwise choose some y_i such that $x_i y_i \notin T_i$, and suppose that Alter prolongs the play from x_i to $x_i y_i$. Let $x_{i+1} := f(x_i y_i)$ the result of the next move of Ego, according to his winning strategy f.

If this process did not terminate, then it would produce an infinite play that is consistent with f and won by Alter. Since f is a winning strategy for Ego, this is impossible. Hence there exists some $m < \omega$ such that $x_m y \in T_m$ for all y. Thus, if Ego moves to x_m in his opening move, then he wins, no matter how the play proceeds afterwards. In particular, Ego wins with a memoryless strategy. □

While many important winning conditions are outside $\mathbf{\Sigma}^0_2$, they may well be Boolean combinations of $\mathbf{\Sigma}^0_2$-sets. For instance, this is the case for parity conditions, Muller conditions, and more generally, S1S-definable winning conditions. In the classical framework of infinite games on graphs (where moves are along edges rather than paths) it is well-known that parity games admit memoryless winning strategies, whereas there are simple games with Muller conditions that require strategies with some memory. We will see that for path games, the class of winning conditions admitting positional winning strategies is much larger than for classical graph games.

Let $G = (V, F)$ be a game graph with a colouring $\lambda : V \to C$ of the nodes with a finite number of colours. The winning condition is given by an ω-regular set $W \subseteq C^\omega$ which is defined by a formula in some appropriate logic over infinite paths. In the most general case, we have S1S-formulae (i.e., MSO-formulae on infinite paths with vocabulary $\{<\} \cup \{P_c : c \in C\}$) but we will also consider weaker formalisms like first-order logic or, equivalently, LTL.

Muller and parity conditions. As we mentioned before, typical examples of winning conditions for which strategies require memory on single-step

games are Muller conditions. Such a condition is specified by a family $\mathcal{F} \subseteq 2^C$ of winning sets; a play is winning if the set of colours seen infinitely often belongs to $\mathcal{F}$.

Proposition 3.3.3. *All Muller path games $(EA)^\omega(G, \mathcal{F})$ and $(AE)^\omega(G, \mathcal{F})$ admit memoryless winning strategies.*

Proof. We will write $w \geq v$ to denote that position w is reachable from position v. For every position $v \in V$, let $C(v)$ be the set of colours reachable from v, that is, $C(v) := \{\lambda(w) : w \geq v\}$. Obviously, $C(w) \subseteq C(v)$ whenever $w \geq v$. In case $C(w) = C(v)$ for all $w \geq v$, we call v a *stable* position. Note that from every $u \in V$ some stable position is reachable. Further, if v is stable, then every reachable position $w \geq v$ is stable as well.

We claim that Ego has a winning strategy in $(EA)^\omega(G, \mathcal{F})$ iff there is a stable position v that is reachable from the initial position v_0, so that $C(v) \in \mathcal{F}$.

To see this, let us assume that there is such a stable position v with $C(v) \in \mathcal{F}$. Then, for every $u \geq v$, we choose a path p from u so that, when moving along p, each colour of $C(u) = C(v)$ is visited at least once, and set $f(u) := p$. In case v_0 is not reachable from v, we assign $f(v_0)$ to some path that leads from v_0 to v. Now f is a memoryless winning strategy for Ego in $(EA)^\omega(G, \mathcal{F})$, because, after the first move, no colours other then those in $C(v)$ are seen. Moreover, every colour in $C(v)$ is visited at each move of Ego, hence, infinitely often.

Conversely, if for every stable position v reachable from v_0 we have $C(v) \notin \mathcal{F}$, we can construct a winning strategy for Alter in a similar way. □

Note that in a finite arena all positions of a strongly connected component that is terminal, i.e., with no outgoing edges, are stable. Thus, the above characterisation translates as follows: Ego wins the game iff there is a terminal component whose set of colours belongs to $\mathcal{F}$. Obviously this can be established in linear time w.r.t. the size of the arena and the description of $\mathcal{F}$.

Corollary 3.3.4. *On a finite arena G, path games with a Muller winning condition $\mathcal{F}$ can be solved in time $O(|G| \cdot |\mathcal{F}|)$.*

The parity condition can be seen as a special case of the Muller condition. Recall that, given an arena $G = (V, F)$ with positions coloured by a priority function $\Omega : V \to \mathbb{N}$ of finite range, this condition requires that the least priority seen infinitely often on a play is even. It turns out that path games with parity conditions

are positionally determined for any game prefix. (By Theorem 3.2.1 we can restrict attention to the eight prefixes E^ω, A^ω, AE^ω, EA^ω, EAE^ω, AEA^ω, $(EA)^\omega$, and $(AE)^\omega$.)

Proposition 3.3.5. *Every parity path game $\gamma(G, \text{parity})$ is determined via a memoryless winning strategy.*

General S1S-winning conditions. In the following, we will use parity games as an instrument to investigate path games with winning conditions specified in the monadic second-order logic of paths, S1S. It is well known that every S1S-definable class of infinite words can be recognised by a deterministic parity automaton (see e.g. [27]). For words over the set of colours C, such an automaton has the form $\mathcal{A} = (Q, C, q_0, \delta, \Omega)$, where Q is a finite set of states, q_0 the initial state, $\delta : Q \times C \to Q$ a deterministic transition function, and $\Omega : Q \to \mathbb{N}$ a priority function. Given an input word, a run of $\mathcal{A}$ starts at the first word position in state q_0; if, at the current position v the automaton is in state q, it proceeds to the next position assuming the state $\delta(q, \lambda(v))$. The input is accepted if the least priority of a state occurring infinitely often in the run is even.

Via a reduction to parity games, we will first show that S1S-games admit finite-memory (or, automatic) strategies. By refining these, we will then establish strategies that are independent of the memory state, that is, positional.

Proposition 3.3.6. *For any winning condition $\psi \in$ S1S and any game prefix γ, the path games $\gamma(G, \psi)$ admit finite-memory winning strategies.*

Proof. Let $\mathcal{A} = (Q, C, q_0, \delta, \Omega)$ be an automaton that recognises the set of words defined by ψ. Given an arena $G = (V, E)$ with starting position v_0, we define the *synchronised product* $G \times \mathcal{A}$ to be the arena with positions $V \times Q$, edges from (v, q) to (v', q') whenever $(v, v') \in E$ and $\delta(q, \lambda(v)) = q'$, and designated starting position (v_0, q_0). We will use two sets of colours for $G \times \mathcal{A}$: one inherited from G, $\lambda(v, q) := \lambda(v)$, and the other one inherited from $\mathcal{A}$, $\Omega(v, q) := \Omega(q)$. When referring to a specific colouring we write, respectively, $G \times \mathcal{A}\restriction_\lambda$ and $G \times \mathcal{A}\restriction_\Omega$. Between the games on G and $G \times \mathcal{A}$ we can observe a strong relationship.

(i) For every prefix γ, a play starting from position (v_0, q_0) is winning in $\gamma(G \times \mathcal{A}\restriction_\lambda, \psi)$ if, and only if, it is winning in $\gamma(G \times \mathcal{A}\restriction_\Omega, \text{parity})$.

(ii) The arenas G, v_0 and $G \times \mathcal{A}\restriction_\lambda$, (v_0, q_0) are bisimilar.

The first assertion follows from the meaning of the automaton $\mathcal{A}$, and entails a strategical equivalence between the two games: Any winning strategy for a certain

player in $\gamma(G{\times}\mathcal{A}{\restriction_\Omega}, \text{parity})$ is also a winning strategy for that player in $\gamma(G{\times}\mathcal{A}{\restriction_\lambda}, \psi)$ and vice versa. By Proposition 3.3.5, there always exists a memoryless winning strategy for the former game and, hence, for the latter one as well.

The second statement holds because $\mathcal{A}$ is deterministic. It implies that every winning strategy for a path game $\gamma(G, \psi)$ starting from position v_0 is also a winning strategy for the game $\gamma(G \times \mathcal{A}, \psi)$ starting from (v_0, q_0). Conversely, every winning strategy f for the latter game induces a winning strategy f' for the former one, namely $f'(v, s) := f\big((v, q'), s\big)$ where $q' := \delta(q_0, s)$ is the state reached by the automaton after processing the word s. Since f can be chosen to be positional, we obtain a winning strategy f' on $\gamma(G, \psi)$ that does not depend on the entire history, but only on a finite memory, namely the set of states Q. □

Note that the finite-memory strategy f' constructed above does not yet need to be positional, since a position v in G has several copies (v, q) in $G \times \mathcal{A}$ at which the prescriptions of f may differ. In order to obtain a state-independent winning strategy for $\gamma(G, \psi)$ we will unify, for each node $v \in V$, the prescriptions $f(v, q)$ for those position (v, q) which are reachable in a play of according to f.

Theorem 3.3.7. *For any winning condition $\psi \in$ S1S, the games $(EA)^\omega(G, \psi)$ and $(AE)^\omega(G, \psi)$ admit memoryless winning strategies.*

Proof. Let us assume that Ego wins the game $(EA)^\omega(G, \psi)$ starting from position v_0. We will base our argumentation on the game $(EA)^\omega(G \times \mathcal{A}, \psi)$, where Ego has a memoryless winning strategy f.

For any $v \in V$, we denote by $Q_f(v)$ the set of states q so that the position (v, q) can be reached from position (v_0, q_0) in a play according to f:

$$Q_f(v) := \{\delta(q_0, s) : s \text{ prolongs } f(v_0, q_0) \text{ and leads to } v\}.$$

Let $\{q_1, q_2, \ldots, q_n\}$ be an enumeration of $Q_f(v)$, in which the initial state q_0 is taken first, in case it belongs to $Q_f(v)$. We construct a path associated to v along the following steps. First, set $p_1 := f(v, q_1)$; for $1 < i \leq n$, let (v', q') be the node reached after playing the path $p_1 \cdot p_2 \cdot \cdots \cdot p_{i-1}$ from position (v, q_i) and set $p_i := f(v', q')$. Finally, let $f'(v)$ be the concatenation of $p_1, p_2, \ldots, p_n$.

Now, consider a play on $(EA)^\omega(G \times \mathcal{A}, \psi)$ in which Ego chooses the path $f'(v)$ at any node $(v, q) \in V \times Q$. This way, the play will start with $f(q_0, v_0)$. Further, at any position (v, q) at which Ego moves, the prescription $f'(v)$ contains some segment of the form $(v', q') \cdot f(v', q')$. In other words, every move of Ego has some

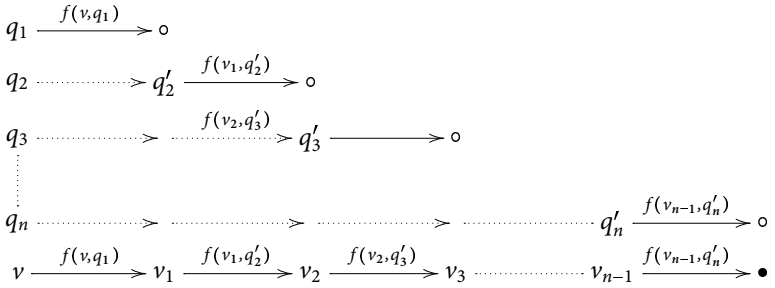


Figure 3.1: Merging strategies at node v

"good part" which would also have been produced by f at the position (v', q'). But this means that the play cannot be distinguished, post-hoc, from a play where Ego always moved according to the strategy f while all the "bad parts" were produced by Alter. Accordingly, Ego wins every play of $(EA)^\omega(G \times \mathcal{A}, \psi)$ starting from (q_0, v_0).

This proves that f' is a memoryless strategy for Ego in the game $(EA)^\omega(G \times \mathcal{A}, \psi)$. Since the values do not depend on the second component, f' induces a memoryless strategy for Ego in $(EA)^\omega(G, \psi)$.

The same construction works for the case $(AE)^\omega(G, \psi)$, if we take instead of $Q_f(v)$ the set $Q(v) := \{\delta(q_0, s) : s \text{ is a path from } v_0 \text{ to } v\}$. □

The above proof relies upon the fact that the players always take turns. If we consider games where the players alternate only finitely many times, the situation changes. Intuitively, a winning strategy of the solitaire player eventually forms an infinite path which may not be broken apart into finite pieces to serve as a memoryless strategy.

Proposition 3.3.8. *For any prefix γ with finitely many alternations between the players, there are arenas G and winning conditions $\psi \in$ S1S so that no memoryless strategy is winning in the game $\gamma(G, \psi)$.*

Proof. Consider, for instance, the arena G_2 from Example 3.3.1 and a winning condition $\psi \in$ S1S that requires the number of zeroes occurring in a play to be odd. When starting from position 1, Ego obviously has winning strategies for each of the games $E^\omega(G, \psi)$, $AE^\omega(G, \psi)$, and $EAE^\omega(G, \psi)$, but no memoryless ones. □

Nevertheless, these games are positionally determined for one of the players. Indeed, if a player wins a game $\gamma(G, \psi)$ finally controlled by his opponent, he always has a memoryless winning strategy. This is trivial when $\gamma \in \{E^\omega, A^\omega, AE^\omega, EA^\omega\}$;

for the remaining cases EAE^ω and AEA^ω a memoryless strategy can be constructed as in the proof of Theorem 3.3.7.

Finally we consider winning conditions that do not depend on initial segments. We say that ψ is a *future*-formula, if, for any ω-word π and any finite words x and y, we have $x\pi \vDash \psi$ if, and only if, $y\pi \vDash \psi$.

Theorem 3.3.9. *For any winning condition $\psi \in$ S1S specified by a future-formula and every prefix γ, the games $\gamma(G, \psi)$ admit a memoryless winning strategies.*

Proof. The core of our argument consists in showing that, given a solitaire game $E^\omega(G, \psi)$, Ego has a uniform positional winning strategy that works for all starting positions in his winning region.

We again consider the game $E^\omega(G \times \mathcal{A}\restriction_\Omega, \text{parity})$ (see item (i) in the proof of Theorem 3.3.7). When playing solitaire, path games do not differ from single-step games, and it is well known that parity games admit winning strategies that are uniform on the entire winning region. Let f be such a strategy. We use f to define a memoryless strategy f' for $\exists^\omega(G, \psi)$ as follows. Starting from any winning position (v_0, q_0) in $E^\omega(G \times \mathcal{A}\restriction_\Omega, \text{parity})$, let $(v_n, q_n)_{n<\omega}$ be the unique play according to f. There are two cases. If the play visits only finitely many different positions, we have $(v_i, q_i) = (v_j, q_j)$ for some i, j and set $f'(v_0) := v_0, v_1, \ldots, v_i$ $f'(v_i) := v_{i+1}, \ldots, v_j$ (overwriting $f'(v_0)$ if $v_i = v_0$). Otherwise, there are infinitely many positions (v_j, q_j) where v_j is fresh, in the sense that $v_j \neq v_i$ for all $i < j$. In that case, we assign to each fresh position v_j the path $f'(v_j) := v_{j+1}, \ldots, v_k$ which leads to the next fresh position v_k in the play. Next, for every node v where f' is still undefined but from which a position $v' \in \text{dom}(f')$ is reachable in G, we choose a path t from v to v' and set $f'(v) := t$. After this, if $\text{dom}(f')$ does not yet contain the entire winning set W of Ego, we take a new starting position $(v'_0, q_0) \in W$ with $v'_0 \in V \setminus \text{dom}(f')$, and proceed as above, until f' is defined everywhere.

We claim that f' is a winning strategy. Consider any play π' that starts at a winning position v for Ego in $E^\omega(G, \psi)$ and that is consistent with f'. By the construction of f' there exists a play π in the arena $G \times \mathcal{A}$, consistent with f, such that the projection of π to G differs from π' only by an initial segment. Now π is a winning for Ego in $E^\omega(G \times \mathcal{A}\restriction_\Omega, \text{parity})$ and therefore also for $\exists^\omega(G \times \mathcal{A}\restriction_\lambda, \psi)$ (by item (1) in the proof of Theorem 3.3.7). By item (2), and since ψ is a future condition this implies that π' is winning for Ego in the game $E^\omega(G, \psi)$.

The case AE^ω follows now immediately since Ego wins $AE^\omega(G, \psi)$ if all positions v reachable from v_0 are in his winning region. For the case EAE^ω, let g be a winning

strategy for Ego. If $g(v_0)$ leads to a position v from which v_0 is again reachable, then f' (constructed above for $E^\omega(G, \psi)$ is a winning strategy also for $EAE^\omega(G, \psi)$. Otherwise, we may change f' for the initial position by $f'(v_0) := g(v_0)$ to obtain a memoryless winning strategy. The other cases follow by duality. □

3.4 Definability

We now study the question in what logics (MSO, μ-calculus, FO, CTL*, ...) winning positions of path games with ω-regular winning conditions can be defined. Given any formula φ from a logic on infinite paths (like S1S or LTL) and a quantifier prefix γ for path games, we define the game formula $\gamma.\varphi$, to be evaluated over game graphs, with the meaning that

$$G \vDash \gamma\,.\,\varphi \quad \text{iff} \quad \text{Player } E \text{ wins the path game } \gamma(G, \varphi).$$

Note that the operation $\varphi \mapsto \gamma.\varphi$ maps a formula over infinite paths to a formula over graphs. Given a logic L over infinite paths, and a prefix γ, let $\gamma.L := \{\gamma.\varphi : \varphi \in L\}$. As usual we write $L \leq L'$ to denote that every formula in the logic L is equivalent to some formula from the logic L'.

Our main definability result can be stated as follows.

Theorem 3.4.1. *For any game prefix γ,*

(i) $\gamma\,.\,\mathrm{S1S} \leq L_\mu$;

(ii) $\gamma\,.\,\mathrm{LTL} \equiv \gamma\,.\,\mathrm{FO} \leq \mathrm{CTL}^*$.

Obviously, the properties expressed by formulae $\gamma.\varphi$ are invariant under bisimulation. This has two relevant consequences:

(a) We can restrict attention to trees (obtained for instance by unravelling the given game graph from the start node).

(b) It suffices to show that, on trees, $\gamma\,.\,\mathrm{S1S} \leq \mathrm{MSO}$, and $\gamma\,.\,\mathrm{FO} \leq \mathrm{MPL}$ where MPL is *monadic path logic*, i.e., monadic second-order logic where second-order quantification is restricted to infinite paths.

The first observation follows directly from the Modal Characterisation Theorem 1.2.27 of Janin and Walukiewicz that every bisimulation-invariant class of trees

that is MSO-definable is also definable in the modal μ-calculus. The second observation is a direct consequence of the Characterisation Theorem 1.2.15 of CTL* in terms of MPL due to Hafer and Thomas [30] and Moller and Rabinovich [52].

Proposition 3.4.2. *On trees,* $(EA)^\omega$. S1S $\leq$ MSO *and* $(AE)^\omega$. S1S $\leq$ MSO.

Proof. Let $x \leq y$ denote that y is reachable from x. A strategy for Player E in a game $(EA)^\omega(T, W)$ on a tree $T = (V, F)$ is a partial function $f : V \to V$, such that $w < f(w)$ for every w; it is winning if every infinite path through T containing $f(\epsilon), y_1, f(y_1), y_2, f(y_2) \ldots$, where $f(y_i) < y_{i+1}$ for all i, satisfies W. An equivalent description can be given in terms of the set $X = f(V)$. A set $X \subseteq V$ defines a winning strategy for Player E in the game $(EA)^\omega(T, W)$ if

(i) $(\forall x \in X)\forall y(x < y \to (\exists z \in X)(y < z))$

(ii) every path hitting X infinitely often is in W (i.e., it is winning for Player E)

(iii) X is non-empty.

Clearly these conditions are expressible in MSO. For the game $(AE)^\omega(G, W)$ we only have to replace (3) by the condition that the start node v is contained in X. □

Proposition 3.4.3. *Let γ be a game prefix with a bounded number of alternations between E and A. Then γ.* S1S $\leq$ MSO *and γ.* FO $\leq$ MPL.

Proof. Every move is represented by a path quantification; by relativising the formula φ that defines the winning condition to the infinite path produced by the players, we obtain an MSO-formula expressing that Player E has a winning strategy for the game given by γ and φ. If φ a first-order formula over paths, then the entire formula remains in MPL. □

The most interesting case concerns winning conditions defined in first-order logic (or equivalently, LTL). In our proof, we will use a normal form for first-order logic on infinite paths (with $<$) that has been established by Thomas [68]. Recall that a first-order formula $\varphi(\bar{x})$ is *bounded* if it only contains bounded quantifiers of form $(\exists y \leq x_i)$ or $(\forall y \leq x_i)$.

Proposition 3.4.4 ([68])**.** *On infinite paths, every first-order formula is equivalent to a formula of the form*

$$\bigvee_i \Big(\exists x (\forall y \geq x) \varphi_i \wedge \forall y (\exists z \geq y) \vartheta_i \Big)$$

where φ_i and ϑ_i are bounded.

Theorem 3.4.5. *On trees,* $(EA)^\omega$. FO $\leq$ FO *and* $(AE)^\omega$. FO $\leq$ FO.

Proof. Let $\psi = \bigvee_i \Big(\exists x(\forall y \geq x)\varphi_i \wedge \forall y(\exists z \geq y)\vartheta_i\Big)$ be a first-order formula on infinite paths describing a winning condition. We claim that, on trees, $(EA)^\omega\psi$ is equivalent to the first-order formula

$$\psi^* := (\exists p_1)(\forall p_2 \geq p_1)(\exists p_3 \geq p_2) \bigvee_{i\in I} \psi_i^{(b)} \quad \text{where}$$
$$\psi_i^{(b)} := (\exists x \leq p_1)(\forall y\,.\, x \leq y \leq p_2)\varphi_i \wedge (\forall y \leq p_2)(\exists z\,.\, y \leq z \leq p_3)\vartheta_i.$$

Let $T = (V, E)$ and suppose first that Alter has a winning strategy for the game $(EA)^\omega(T, \psi)$. We prove that $T \vDash \neg\psi^*$. To see this we have to define an appropriate Skolem function $g : p_1 \mapsto p_2$ such that for all $p_3 \geq p_2$ and all $i \in I$

$$T \vDash \neg\psi_i^{(b)}(p_1, p_2, p_3).$$

Fix any p_1 which we can consider as the first move of Ego in the game $(EA)^\omega(T, \psi)$ and any play P (i.e., any infinite path through T) that prolongs this move and that is consistent with Alter's winning strategy. Since Alter wins, we have that $P \vDash \neg\psi$. Hence there exists some $J \subseteq I$ such that

$$P \vDash \bigwedge_{i\in J} \forall x(\exists y \geq x)\neg\varphi_i \wedge \bigwedge_{i\in I-J} \exists y(\forall z \geq y)\neg\vartheta_i.$$

To put it differently, there exist

- for every $i \in J$ and every $a \in P$ a witness $h_i(a) \in P$ such that $P \vDash \neg\varphi_i(a, h_i(a))$, and
- for every $i \in I - J$ an element b_i such that $P \vDash (\forall z \geq b_i)\neg\vartheta_i(b_i, z)$.

Now set

$$p_2 := \max(\{h_i(a) : a \leq p_1, i \in J\} \cup \{b_i : i \in I - J\}).$$

For any p_3 we now obviously have that $T \vDash \neg\psi_i^{(b)}(p_1, p_2, p_3)$.

For the converse, let $f : V \to V$ be a winning strategy for Ego in the game $(EA)^\omega(T, \psi)$. We claim that $T \vDash \psi^*$. Toward a contradiction, suppose that $T \vDash \neg\psi^*$. Hence there exists a Skolem function $g : V \to V$ assigning to each p_1 an appropriate $p_2 \geq p_1$ such that $T \vDash \neg\psi_i^{(b)}(p_1, p_2, p_3)$ for all $p_3 \geq p_2$ and all $i \in I$. We can view g as a strategy for Alter in the game $(EA)^\omega(T, \psi)$. If Ego plays according to f and Alter

plays according to g, then the resulting infinite play $f \hat{} g = q_1 q_2 q_3 \ldots$ satisfies ψ (because f is a winning strategy). Hence there exists some $i \in I$ such that

$$f \hat{} g \vDash \exists x (\forall y \geq x) \varphi_i \wedge \forall y (\exists z \geq y) \vartheta_i.$$

Let a be a witness for x so that $f \hat{} g \vDash (\forall y \geq a) \varphi_i(a, y)$. Choose the minimal odd k, such that $a \leq q_k$, and set $p_1 := q_k$. Then $q_{k+1} = g(q_k) = g(p_1) = p_2$. Since $f \hat{} g \vDash \forall y (\exists z \geq y) \vartheta_i(y, z)$, we have, in particular, for every $b \leq p_2$ a witness $h(b) \geq b$ on $f \hat{} g$ such that $f \hat{} g \vDash \vartheta_i(b, h(b))$. Choose $p_3 = \max\{h(b) : b \leq p_2\}$ It follows that $f \hat{} g \vDash \psi_i^{(b)}(p_1, p_2, p_3)$. Since $\psi_i^{(b)}$ is bounded, its evaluation on T is equivalent to its evaluation on $f \hat{} g$. Hence we have shown that there exists p_1 such that for $p_2 = g(p_1)$, given by the Skolem function g, we can find a p_3 with $T \vDash \psi_i^{(b)}(p_1, p_2, p_3)$. But this contradicts the assumption that g is an appropriate Skolem function for $\neg \psi^*$.

We have shown that whenever Ego has a winning strategy for $(EA)^\omega(T, \psi)$ then $T \vDash \psi^*$ and whenever Alter has a winning strategy, then $T \vDash \neg \psi^*$. By contraposition and determinacy, the reverse implications also hold. For games of form $(AE)^\omega(T, \psi)$ the arguments are analogous. □

4 Entanglement

Throughout this chapter, we develop a new parameter for the complexity of finite directed graphs which measures to what extent the cycles of the graph are intertwined. This measure, called entanglement, is defined by way of a game that is somewhat similar in spirit to the robber-and-cops games used to describe tree width, directed tree width, and hypertree width. Nevertheless on many classes of graphs, there are significant differences between entanglement and the various incarnations of tree width.

We show that entanglement is intimately connected to the computational and descriptive complexity of the modal μ-calculus. One one hand, the number of fixed point variables needed to describe a finite graph up to bisimulation is captured by its entanglement. This will play a crucial role in the next chapter, where we prove that the variable hierarchy of the μ-calculus is strict.

In addition to this, we show that parity games of bounded entanglement can be solved in polynomial time. Specifically, we establish that the complexity of solving a parity game can be parametrised in terms of the minimal entanglement of subgames induced by a winning strategy.

4.1 Defining bisimulation and simulation types of finite structures

We are concerned with formulae that describe finite Kripke structures, more precisely, the bisimulation-invariant properties at a given state. In particular, we are interested in existential properties preserved under simulation.

Definition 4.1.1. Let $\mathcal{K}$ be a Kripke structure with a designated state u. A formula $\psi \in \mathrm{L}_\mu$ describes the *bisimulation type* of $\mathcal{K}, u$ if, for any structure $\mathcal{K}'$, we have $\mathcal{K}', u' \models \psi$ iff $\mathcal{K}, u \sim \mathcal{K}', u'$. Likewise, we say that ψ describes the *simulation type* of $\mathcal{K}, u$ if, for any Kripke structure $\mathcal{K}'$, we have $\mathcal{K}', u' \models \psi$ iff $\mathcal{K}, u \lesssim \mathcal{K}', u'$.

A straightforward approach to describing a finite structure up to bisimulation consists in forming a system of simultaneous fixed points associated to the individual states. Given a finite structure $\mathcal{K} = (V, (E_a)_{a\in\textsc{Act}}, (V_p)_{p\in\textsc{Prop}})$, the atomic type of any node $v \in V$ is described by the formula

$$\alpha_v := \bigwedge_{\substack{p\in\textsc{Prop}\\ v\in V_p}} p \wedge \bigwedge_{\substack{p\in\textsc{Prop}\\ v\notin V_p}} \neg p.$$

Let S be the system defining, for every node $v \in V$, a proposition X_v via the equation

$$X_v = \alpha_v \wedge \bigwedge_{a\in\textsc{Act}} \Big(\bigwedge_{(v,w)\in E_a} \langle a\rangle X_w \wedge [a] \big(\bigvee_{(v,w)\in E_a} X_w \big) \Big).$$

It can be easily seen that on any Kripke structure $\mathcal{K}'$, the greatest solution of this system maps each variable X_v to the set $\{\, v' \in V' \mid \mathcal{K}, v \sim \mathcal{K}', v' \,\}$. Hence, the bisimulation type of $\mathcal{K}, u$ is described by $\nu X_u : S$.

If we restrict the definitions of X_v in S to their existential part,

$$X_v = \alpha_v \wedge \bigwedge_{\substack{a\in\textsc{Act}\\ (v,w)\in E_a}} \langle a\rangle X_w,$$

the greatest solution of the obtained system maps every variable X_v to the set $\{\, v' \in V' \mid \mathcal{K}, v \lesssim \mathcal{K}', v' \,\}$ and thus $\nu X_u : S$ describes the simulation type of $\mathcal{K}, u$.

In general, however, this approach uses much more variables than needed. Any acyclic finite structure can be described already in basic modal logic. Typically, this is achieved by a formula whose syntax follows the finite tree obtained by unravelling the structure. We may proceed similarly to describe structures with cycles in the μ-calculus. Syntactically, L_μ-formulae are trees with back edges; each reference to a fixed-point variable semantically instantiates its binding definition, which occurred previously in the syntax tree. This allows us to describe any Kripke structure over a tree with back edges by associating greatest fixed-point variables to each node with incoming back edges. We obtain a defining formula following the tree edges, as in the acyclic case, additionally referencing for every back edge the fixed-point variable associated to its target. For instance, the simulation type of the structure from Figure 4.1 at state 0 is described by $\nu X.\big(\langle a\rangle X \wedge \langle b\rangle\langle b\rangle\langle a\rangle X\big)$.

Likewise, it is possible to characterise any finite structure $\mathcal{K}$ by describing a tree with back edges bisimilar to $\mathcal{K}$. Such a tree can be obtained, for example, by partially performing an unravelling of $\mathcal{K}$ as in Definition 1.1.10, but with the difference that, whenever a node that occurred previously on the current path is reached, a back

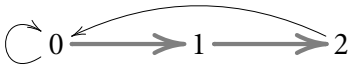


Figure 4.1: A simple structure with cycles, (a-transitions plain, b-transitions thicker)

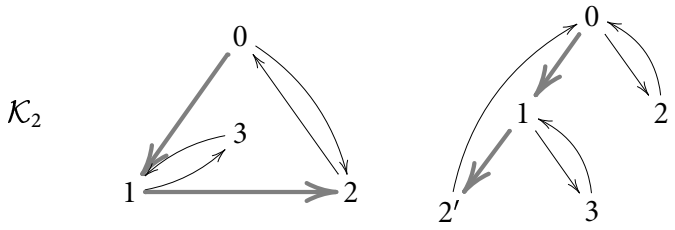


Figure 4.2: Viewing a structure as a tree with back edges

edge to this occurrence is added instead of creating a new copy. (Later on, we will formally introduce the notion of unravelling by generalising this procedure.) For the simulation type of the structure from Figure 4.2, we thus obtain the formula:

$$\nu X.\Big(\langle b\rangle\mu Y.\big(\langle b\rangle\langle a\rangle X \wedge \langle a\rangle\langle a\rangle Y\big) \wedge \langle a\rangle\langle a\rangle X\Big).$$

Notice, however, that a given structure may have several structurally different trees with back edges as bisimilar companions, leading to syntactically different descriptions. In particular, since we introduce variables for every note entered by a back edge, the number of variables involved in those descriptions may differ, as illustrated by the formulae obtained for the two bisimilar structures in Figure 4.3:

$$\begin{aligned}&\nu X.\langle b\rangle\nu Y.\langle b\rangle(\,\langle a\rangle X \wedge \langle a\rangle Y \wedge \nu Z.\langle a\rangle Z\,)\\ &\quad\equiv \langle b\rangle\langle b\rangle\nu X.\big(\langle a\rangle\langle b\rangle\langle b\rangle X \wedge \langle a\rangle\langle b\rangle X \wedge \langle a\rangle X\big).\end{aligned}$$

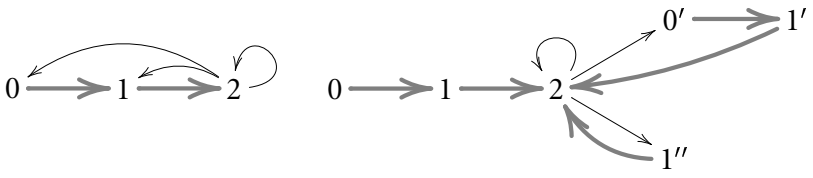


Figure 4.3: Bisimilar companions with different cyclic structure

To understand this phenomenon, we set out for an investigation of the cyclic structure of graphs taking into account their unravelling as finite trees with back edges. Towards this, we introduce a structural parameter for the complexity of finite directed graphs which measures to what extent the cycles of the graph are intertwined. The definition of this measure, called entanglement, is given in terms game similar in spirit to the robber and cops games used to describe tree width, directed tree width, and hypertree width [63, 39, 24]. Nevertheless, there are significant differences between entanglement and the various incarnations of tree width.

As we will show, the entanglement of a finite Kripke structure provides an upper bound for the number of fixed point variables needed to describe it up to bisimulation. In Chapter 5, we will further prove that this bound is tight, in a fairly general sense.

4.2 The entanglement game: Catching the thief

Let $\mathcal{G} = (V, E)$ be a finite directed graph with a designated root u. The *entanglement* of $\mathcal{G}, u$, denoted $\mathrm{ent}(\mathcal{G}, u)$, is defined by way of a game, played by a thief against k detectives on $\mathcal{G}$ according to the following rules. At the beginning, the thief is at the given initial position u of $\mathcal{G}$ and the detectives are outside the graph. In any round, the detectives may either stay where they are, or place one of them on the current position v of the thief. The thief, in turn, has to move to a successor w of v that is not occupied by any detective. If no such position exists, the thief is caught and the detectives have won. Note that the thief sees the move of the detectives before he decides on his own move, and that he is forced to leave his current position, regardless whether the detectives move or not.

Definition 4.2.1. The entanglement of $\mathcal{G}, u$ is the minimal number $k \in \mathbb{N}$ such that k detectives have a strategy to catch the thief on $\mathcal{G}$ starting from position u.

Notice that if, in a graph $\mathcal{G}$, we consider two nodes u, u' from which all other nodes are reachable, then $\mathrm{ent}(\mathcal{G}, u) = \mathrm{ent}(\mathcal{G}, u')$. When we deal with graphs where all nodes are reachable from some root, we may simply write $\mathrm{ent}(\mathcal{G})$ instead of $\mathrm{ent}(\mathcal{G}, u)$, for any root u.

The entanglement is an interesting measure on *directed* graphs. To deal with undirected graphs, we view undirected edges $\{u, v\}$ as pairs (u, v) and (v, u) of directed edges. In the following a graph is always meant to be directed.

To get a feeling for this measure we collect a few simple observations concerning the entanglement of certain familiar graphs. The proofs are simple and left to the reader.

Proposition 4.2.2. *Let $\mathcal{G}$ be a finite directed graph.*

(i) $\mathrm{ent}(\mathcal{G}) = 0$ *if, and only if, $\mathcal{G}$ is acyclic.*

(ii) *If $\mathcal{G}$ is the graph of a unary function, then* $\mathrm{ent}(\mathcal{G}) = 1$.

(iii) *If $\mathcal{G}$ an undirected tree, then* $\mathrm{ent}(\mathcal{G}) \leq 2$.

(iv) *If $\mathcal{G}$ is the fully connected directed graph with n nodes, then* $\mathrm{ent}(\mathcal{G}) = n$.

Let C_n denote the directed cycle with n nodes. Given two graphs $\mathcal{G} = (V, E)$ and $\mathcal{G}' = (V', E')$ their asynchronous product is the graph $\mathcal{G} \times \mathcal{G}' = (V \times V', F)$ where

$$F = \big\{(uu', vv') : [\,(u, v) \in E \wedge u' = v'\,] \vee [\,u = v \wedge (u'v') \in E'\,]\big\}.$$

Note, that $T_{mn} := C_m \times C_n$ is the $(m \times n)$-torus or, to put it differently, the graph obtained from the directed $(m + 1) \times (n + 1)$-grid by identifying the left and right border and the upper and lower border.

Proposition 4.2.3. (i) *For every n,* $\mathrm{ent}(T_{nn}) = n$.

(ii) *For every $m \neq n$,* $\mathrm{ent}(T_{mn}) = \min(m, n) + 1$.

Proof. On T_{nn}, a group of n detectives can catch the thief by placing themselves on a diagonal, thus blocking every row and every column of the torus. On the other side, it is obvious that the thief can escape against $n - 1$ detectives.

On T_{mn} with $m < n$, m detectives are needed to block every row, and an additional detective forces the thief to leave any column after at most n moves, so that he finally must run into a detective. Again, it is obvious that the thief escapes if there are less than $m + 1$ detectives.

□

The following proposition characterises the graphs with entanglement one.

Proposition 4.2.4. *The entanglement of a directed graph is one, if and only if, the graph is not acyclic, and in every strongly connected component, there is a node whose removal makes the component acyclic.*

Proof. On any graph with this property, one detective catches the thief by placing himself on the critical node in the current strongly connected component when the thief passes there. The thief will have to return to this node or leave the current component. Eventually she will be caught in a terminal component.

Conversely if there is a strongly connected component without such a critical node, then the thief may always proceed from his current position towards an unguarded cycle and thus escape forever.

□

Corollary 4.2.5. *For $k = 0$ and $k = 1$, the problem whether a given graph has entanglement k is* NLOGSPACE*-complete.*

The definition of entanglement is reminiscent of the characterisation of tree width via robber and cops games introduced by Seymour and Thomas in [63], and especially of *directed tree width* as defined by Johnson, Robertson, Seymour, and Thomas [39]. However, we will see that entanglement is a quite different, and for some purposes more accurate, measure than directed tree width. This becomes particularly apparent on trees with back edges which also play an important role in our analysis of the variable hierarchy of the modal μ-calculus. It is easy to see that the directed tree width of any tree with back edges is one. However, we will see that the entanglement of trees with back edges can be arbitrarily large.

For undirected graphs, the precise relationship between tree width and entanglement is not known.

To obtain some insight, we can use the following sufficient criterion for the existence of a winning strategy for k detectives.

Lemma 4.2.6. *Let $\mathcal{G} = (V, E)$ be a game graph. If for some $k \in \mathbb{N}$, there exists a partial labelling $i : V \to [k]$ under which every strongly connected subgraph $\mathcal{C} \subseteq \mathcal{G}$ contains a vertex u with a unique label in $\mathcal{C}$, i.e. $i(u) \neq i(v)$ for all $v \in \mathcal{C}$, then we have* $\mathrm{ent}(\mathcal{G}) \leq k$.

Proof. We interpret the labelling i as a memoryless strategy for the detectives as follows: whenever the thief reaches a position $v \in \mathrm{dom}(i)$ in a play on $\mathcal{G}$, detective $i(v)$ is placed at this position v, if $i(v)$ is defined. Otherwise no detective moves.

Towards a contradiction, suppose that although the detectives follow this strategy, the thief can form an infinite path without meeting any detective. Let now C be the set of positions seen infinitely often on this path. Clearly, C induces in $\mathcal{G}$ a strongly connected subgraph. Let $u \in C$ be a node whose label $i(v)$ is unique in $\mathcal{C}$.

According to the memoryless strategy described by i, the detective $i(u)$ must have been posted at u all the time since the play stabilised in $\mathcal{C}$. On the other hand, every position in C, in particular u, must have been re-visited infinitely often. But this contradicts our assumption that the thief is not captured. □

Proposition 4.2.7. *For every n, the undirected $(n \times n)$-grid has entanglement at most $3n$.*

Proof. Consider the labelling $i : [n] \times [n] \to [3n]$ obtained by first assigning the values $0, \dots, n$ to the horizontal median of the grid, i.e., $i(\frac{n}{2}, j) := j$ for all $j \in [n]$. For the two $\frac{n}{2} \times n$ grids obtained when removing the positions already labelled, we proceed independently and assign the values $n, \dots, n + \frac{n}{2}$ to their respective medians, and so on, in step k applying the procedure to the still unlabelled domain consisting of 2^k many $\frac{n}{2^k} \times \frac{n}{2^k}$ disconnected grids. It is easy to verify that the labelling obtained this way satisfies the criterion of Lemma 4.2.6. □

Proposition 4.2.8. *For any finite undirected graph $\mathcal{G}$ of tree width k, we have that* $\mathrm{ent}(\mathcal{G}) \leq (k+1) \cdot \log |G|$.

Proof. By definition, every graph $\mathcal{G} = (V, E)$ of tree width k can be decomposed as a tree $\mathcal{T}$ labelled with subsets of at most $k+1$ elements of V, called *blocks*, such that (1) every edge $\{u, v\} \in E$ is included in some block and (2) for any element $v \in V$ the set of blocks which contain v is connected.

In every subtree $\mathcal{S}$ of such a decomposition tree, there exists a node s, we may call it the *center* of $\mathcal{S}$, which balances $\mathcal{S}$ in the sense that the two subtrees in $\mathcal{S} \setminus \{s\}$ carry almost the same number of vertices in their blocks (differences up to k are admissible). Consider now the following memoryless detective strategy. First, all vertices in the centre s of the decomposition tree receive indices $0, \dots, k$. Then, we repeat the process independently for the two subtrees disconnect by the removal of s and assign to the vertices in their respective centres indices $k+1, \dots, 2k+2$. The process ends when all vertices of $\mathcal{G}$ are labelled. In this way, at most $(k+1) \log |V|$ detective indices are assigned. Since the blocks of a tree decomposition separate the graph, every strongly connected component of $\mathcal{G}$ will contain at least one unique label. This shows that the constructed labelling indeed represents a memoryless strategy for at most $(k+1) \log |V|$ detectives.

□

4.2.1 Trees with back edges and finite unravellings

Let $\mathcal{T} = (V, E)$ be a directed tree. We write $\preceq_E$ for the associated partial order on $\mathcal{T}$. Note that $\preceq_E$ is just the reflexive, transitive closure of E.

Definition 4.2.9 (Tree with back edges). A directed graph $\mathcal{T} = (V, F)$ is a *tree with back edges* if there is a partition $F = E \cup B$ of the edges into tree edges and back edges such that (V, E) is indeed a directed tree, and whenever $(u, v) \in B$, then $v \preceq_E u$.

The following observation shows that up to the choice of the root, the decomposition into tree edges and back edges is unique.

Lemma 4.2.10. *Let $\mathcal{T} = (V, F)$ be a tree with back edges and $v \in V$. Then there exists at most one decomposition $F = E \cup B$ into tree edges and back edges such that (V, E) is a tree with root v.*

Definition 4.2.11 (Feedback). Let $\mathcal{T} = (V, E, B)$ be a tree with back edges. The *feedback* of a node v of $\mathcal{T}$ is the number of ancestors of v that are reachable by a back-edge from a descendant of v. The feedback of $\mathcal{T}$, denoted $\mathrm{fb}(T)$ is the maximal feedback of nodes on $\mathcal{G}$. More formally,

$$\mathrm{fb}(T) = \max_{v \in V} |\{u \in V : \exists w (u \preceq_E v \preceq_E w \wedge (w, u) \in B)\}|.$$

We call a back-edge (w, u), and likewise its target u, *active* at a node v in $\mathcal{T}$, if $u \preceq_E v \preceq_E w$.

Note that the feedback of $\mathcal{T}$ may depend on how the edges are decomposed into tree edges and back edges, i.e., on the choice of the root. Consider, for instance the left graph from Figure 4.3. If node 0 is taken as the root, then the feedback is 3; instead, if we take node 1 as the root, then the feedback is 2.

Lemma 4.2.12. *Let $\mathcal{T} = (V, E, B)$ be a tree with back edges of feedback k. Then there exists a partial labelling $i : V \mapsto \{0, \ldots, k-1\}$ assigning to every target u of a back-edge an index $i(u)$ in such a way that no two nodes u, u' that are active at the same node v have the same index.*

Proof. The values of this labelling are set while traversing the tree in breadth-first order. Notice that every node u with an incoming back edge is active at itself. As $\mathcal{T}$

has feedback k, there can be at most $k-1$ other nodes active at u. All of these are ancestors of u, hence their index is already defined. There is at least one index which we can assign to u so that no conflict with the other currently active nodes arises.

□

Lemma 4.2.13. *The entanglement of a tree with back edges is at most its feedback:* $\mathrm{ent}(\mathcal{T}) \leq \mathrm{fb}(\mathcal{T})$.

Proof. Suppose that $\mathrm{fb}(\mathcal{T}) = k$. By Lemma 4.2.12 there is a labelling i of the targets of the back edges in $\mathcal{T}$ by numbers $0, \ldots, k-1$ assigning different values to any two nodes u, u' that are active at the same node v. This labelling induces the following strategy for the k detectives: at every node v reached by the thief, send detective number $i(v)$ to that position or, if the value is undefined, do nothing. By induction over the stages of the play, we can now show that this strategy maintains the following invariant: at every node v occurring in a play on $\mathcal{T}$, all active nodes $u \neq v$ are occupied and, if the current node is itself active, a detective is on the way. To see this, let us trace the evolution of the set $Z \subseteq T$ of nodes occupied by a detective. In the beginning of the play, Z is empty. A node v can be included into Z if it is visited by the thief and active with regard to itself. At this point, our strategy appoints detective $i(v)$ to move to v. Since, by construction of the labelling, the designated detective $i(v)$ must come from a currently inactive position and, hence, all currently active positions except v remain in Z. But if every node which becomes active is added to Z and no active node is ever given up, the thief can never move along a back-edge, so that after a finite number of steps he reaches a leaf of the tree and loses. But this means that we have a winning strategy for k detectives, hence $\mathrm{ent}(\mathcal{T}) \leq k$.

□

According to Definition 1.1.10, every graph $\mathcal{G}$ can be unravelled from any node v to a tree $\mathcal{T}_{G,v}$ whose nodes are the paths in $\mathcal{G}$ from v. Clearly $\mathcal{T}_{G,v}$ is infinite unless $\mathcal{G}$ is finite and no cycle in $\mathcal{G}$ is reachable from v. A *finite unravelling* of a (finite) graph $\mathcal{G}$ is defined in a similar way, but rather than an infinite tree, it produces a finite tree with back edges. To construct a finite unravelling we proceed as in the usual unravelling process with the following modification: whenever we have a path $v_0 v_1 \ldots v_n$ in $\mathcal{G}$ with corresponding node $\overline{v} = v_0 v_1 \ldots v_n$ in the unravelling, and a successor w of v_n that coincides with v_i (for any $i \leq n$), then we may, instead of creating the new node $\overline{v}w$ (with a tree edge from $\overline{v}$ to $\overline{v}w$) put a back-edge from $\overline{v}$ to its ancestor $v_0 \ldots v_i$. Clearly this process is nondeterministic. In this way, any finite

graph can be unravelled, in many different ways, to a finite tree with back edges. Observe that different finite unravellings of a graph may have different feedback and different entanglement.

Obviously, the entanglement of a graph is bounded by the entanglement of its finite unravellings. Indeed, a winning strategy for k detectives on a finite unravelling of $\mathcal{G}$ immediately translates to a winning strategy on $\mathcal{G}$.

Proposition 4.2.14. *The entanglement of a graph is the minimal feedback (and the minimal entanglement) of its finite unravellings:*

$$\begin{aligned}\mathrm{ent}(\mathcal{G}, u) &= \min\{\mathrm{fb}(\mathcal{T}) : \mathcal{T} \textit{ is a finite unravelling of } \mathcal{G}, u\} \\ &= \min\{\mathrm{ent}(\mathcal{T}) : \mathcal{T} \textit{ is a finite unravelling of } \mathcal{G}, u\}.\end{aligned}$$

Proof. For any finite unravelling $\mathcal{T}$ of a graph $\mathcal{G}, u$, we have

$$\mathrm{ent}(\mathcal{G}, u) \leq \mathrm{ent}(\mathcal{T}) \leq \mathrm{fb}(\mathcal{T}).$$

It remains to show that for any graph $\mathcal{G}$ with a designated node u, there exists some finite unravelling $\mathcal{T}$ from u with $\mathrm{fb}(\mathcal{T}) \leq \mathrm{ent}(\mathcal{G}, u)$.

To prove this, we view winning strategies for the detectives as descriptions of finite unravellings. A strategy for k detectives tells us, for any finite path πv of the thief whether a detective should be posted at the current node v, and if so, which one. Such a strategy can be represented by a partial function g mapping finite paths in $\mathcal{G}$ to $\{0, \ldots, k-1\}$. On the other hand, during the process of unravelling a graph to a (finite) tree with back edges, we need to decide, for every successor v of the current node, whether to create a new copy of v or to return to a previously visited one, if any is available. To put this notion on a formal ground, we define an *unravelling function* for a rooted graph $\mathcal{G}, u$ as a partial function ρ between finite paths from u through $\mathcal{G}$, mapping any path $v_0, \ldots, v_{r-1}, v_r$ from $v_0 = u$ in its domain to a strict prefix $v_0, v_1, \cdots, v_{j-1}$ such that $v_{j-1} = v_r$. Such a function gives rise to an unravelling of $\mathcal{G}$ in the following way: we start at the root and follow finite paths through $\mathcal{G}$. Whenever the current path π can be prolonged by a position v and the value of ρ at πv is undefined, a fresh copy of v corresponding to πw is created as a successor of π. In particular, this always happens if v was not yet visited. Otherwise, if $\rho(\pi\, v)$ is defined, then the current path π is bent back to its prefix $\rho(\pi)$ which also corresponds to a copy of v. Formally, the unravelling of $\mathcal{G}, u$ *driven* by ρ is the tree with back edges $\mathcal{T}$ defined as follows:

- the domain of $\mathcal{T}$ is the smallest set T which contains $u =: v_0$ and for each path $\pi \in T$, it also contains all prolongations πv in $\mathcal{G}$ at which ρ is undefined;
- the tree edge partition is

$$E^{\mathcal{T}} := \{ (v_0, \ldots, v_{r-1},\ v_0, \ldots, v_{r-1}, v_r) \in T \times T \mid (v_{r-1}, v_r) \in E^{\mathcal{G}} \};$$

- for all paths $\pi := v_0, \ldots, v_{r-1} \in T$ where $\rho(\pi v)$ is defined, the back-relation $B^{\mathcal{T}}$ contains the pair $(\pi,\ \rho(\pi v))$ if $(v_{r-1}, v) \in E^{\mathcal{G}}$.

We are now ready to prove that every winning strategy g for the k detectives on $\mathcal{G}, u$ corresponds to an unravelling function ρ for $\mathcal{G}, u$ that controls a finite unravelling with feedback k.

Note that the strategy g gives rise to a k-tuple $(g_0, \ldots, g_{k-1})$ of functions mapping every initial segment π of a possible play according to g to a k-tuple $(\, g_0(\pi), \ldots, g_{k-1}(\pi)\,)$ where each $g_i(\pi)$ is a prefix of π recording the state of the play (i.e., the current path of the thief) at the last move of detective i.

Now, for every path π and possible prolongation by v, we check whether, after playing π, there is any detective posted at v. If this is the case, i.e, when, for some i, the end node of $g_i(\pi)$ is v, we set $\rho(\pi\, v) := \pi_i$. Otherwise we leave the value of ρ undefined at π, v. It is not hard to check that, if g is a winning strategy for the detectives, the associated unravelling is finite and has feedback k.

□

4.3 Descriptive complexity

In this section we start investigating the connection between the entanglement of a Kripke structure and the L_μ-formulae defining it. First, observe that the feedback of the syntax graph of a formula φ in L_μ is not greater than the number of variables occurring in φ.

The entanglement of a Kripke structure $\mathcal{K} = \big(\, V, (E_a)_{a \in \mathrm{Act}}, (V_p)_{p \in \mathrm{Prop}} \,\big)$ is the entanglement of the underlying graph (V, E) where $E = \bigcup_{a \in \mathrm{Act}} E_a$. We now show that every Kripke structure of entanglement k can be described, up to bisimulation, in the μ-calculus using only k fixed-point variables.

Proposition 4.3.1. *Let $\mathcal{K}$ be a finite Kripke structure with* $\mathrm{ent}(\mathcal{K}) = k$. *Then, for any node u of $\mathcal{K}$, there exist formulae in $\mathrm{L}_\mu[k]$ that describe the bisimulation type, respectively the simulation type of $\mathcal{K}, u$.*

Proof. According to Proposition 4.2.14, the system $\mathcal{K}$ can be unravelled from any node u to a finite tree $\mathcal{T}$ with back edges, with root u and feedback k. Clearly $\mathcal{T}, u \sim \mathcal{K}, u$. Hence, it is sufficient to prove the proposition for $\mathcal{T}, u$. Assume that for every action $a \in \textsc{Act}$, the transitions in $\mathcal{T}$ are partitioned into tree edges and back edges $E_a \cup B_a$.

Let $i : \mathcal{T} \mapsto \{0, \ldots, k-1\}$ be the partial labelling of $\mathcal{T}$ defined in Lemma 4.2.12. On the basis of this labelling, we construct a sequence of formulae $(\psi_v)_{v \in T}$ over fixed-point variables $X_0, \ldots, X_{k-1}$ while traversing the nodes of $\mathcal{T}$ in reverse breadth-first order. For every action $a \in \textsc{Act}$, the transitions in $\mathcal{T}$ are partitioned into tree edges and back edges $E_a \cup B_a$.

To describe a state $v \in \mathcal{T}$ and the relationship with its successors, let

$$\varphi_v := \alpha_v \wedge \bigwedge_{a \in \textsc{Act}} \Bigg(\bigwedge_{(v,w) \in E_a} \langle a \rangle \psi_w \wedge \bigwedge_{(v,w) \in B_a} \langle a \rangle X_{i(w)} \\ \wedge\ [a] \Bigg(\bigvee_{(v,w) \in E_a} \psi_w \vee \bigvee_{(v,w) \in B_a} X_{i(w)} \Bigg) \Bigg),$$

where α_v expresses the atomic type of v:

$$\alpha_v := \bigwedge_{\substack{p \in \textsc{Prop} \\ v \in V_p}} p \wedge \bigwedge_{\substack{p \in \textsc{Prop} \\ v \notin V_p}} \neg p.$$

If v has an incoming back-edge, we set $\psi_v := \nu X_{i(v)} \,.\, \varphi_v$; otherwise, we let $\psi_v := \varphi_v$.

Note that since we proceed from the leaves of $\mathcal{T}$ to the root, this process is well-defined, and that in ψ_v the variables $X_{i(z)}$ occur free, for any node $z \neq v$ that is active at v. In particular, all variables in the formula ψ_u, corresponding to the root u of $\mathcal{T}$, are bound.

We claim that $\mathcal{K}, v \models \psi_u$ iff $\mathcal{K}, v \sim \mathcal{T}, u$. First, we show that $\mathcal{T}, u \models \psi_u$, and hence $\mathcal{K}, v \models \psi_u$ for any $\mathcal{K}, v \sim \mathcal{T}, u$. To see this, we prove that Verifier has a winning strategy for the associated model-checking game.

Note that, since ψ_u has only greatest fixed points, any infinite play of the model-checking game is won by Verifier. It thus suffices to show that from any position of form (v, φ_v), Verifier has a strategy to make sure that the play proceeds to a next position of form (w, φ_w), unless Falsifier moves to position (v, α_v) and then loses in the next move. But by the construction of the formula, it is obvious that Verifier can play so that any position at which he moves is of one of the following three types.

(i) $(v, \langle a \rangle \psi_w)$, where $(v, w) \in E_a$: then, Verifier moves to position (w, ψ_w).

(ii) $(v, \langle a\rangle X_{i(w)})$, where $(v, w) \in B_a$: in this case, he moves to $(w, X_{i(w)})$.

(iii) $(w, \bigvee_{(v,z)\in E_a} \psi_z \vee \bigvee_{(v,z)\in B_a} X_{i(z)})$ for some edge $(v, w) \in E_a \cup B_a$: in this case, Verifier selects the appropriate disjunct with $z = w$ and moves accordingly either to (w, ψ_w) or to $(w, X_{i(w)})$.

In all cases the play will proceed to (w, φ_w). Hence, Falsifier can force a play to be finite only by moving to a position (v, α_v), where he loses. Otherwise the resulting play is infinite and thus always won by Verifier.

For the converse, suppose that $\mathcal{K}, v \not\sim \mathcal{T}, u$. Since $\mathcal{T}$ is finite, the non-bisimilarity it witnessed at a finite stage. That is, there is a basic modal formula separating $\mathcal{K}, v$ from $\mathcal{T}, u$, and Falsifier can force the model-checking game for ψ_u on $\mathcal{K}, v$ in finitely many moves to a position of form (w, α_w) such that w and w' have distinct atomic types. This proves that $\mathcal{K}, v \not\models \psi_u$.

By the same argument, we obtain a description of the simulation type of $\mathcal{K}, u$ using formulae φ_v restricted to their existential part:

$$\varphi_v := \alpha_v \wedge \bigwedge_{a\in \mathrm{Act}} \left(\bigwedge_{(v,w)\in E_a} \langle a\rangle \psi_w \wedge \bigwedge_{(v,w)\in B_a} \langle a\rangle X_{i(w)} \right).$$

□

As the entanglement of a Kripke structure regards only the underlying graph, one can easily find examples of high entanglement that can be described with very few variables. For instance, in a Kripke structure over a strongly connected finite graph with no atomic propositions and only a single action a, all states are bisimilar, and can be described by $\nu X.(\langle a\rangle X \wedge [a]X)$, regardless of the entanglement of the underlying graph. Nevertheless, in the following chapter we shall see that we can establish a strong relationship between the notion of entanglement and the descriptive complexity of L_μ, under fairly general assumptions.

4.4 Computational complexity

An intriguing open problem related to the μ-calculus regards the computational complexity of its model checking problem, or equivalently, the problem to establish the winner in a parity game.

Parity games were introduced in Section 1.3 as path-forming games played between two players on labelled graphs $\mathcal{G} = (V, V_0, E, (\Omega)_{i<n})$. In this place, it is

convenient to think of the priority partition as a function $\Omega : V \to [n]$, mapping a position $v \in V_i$ to the priority i. Recall that these games are determined with memoryless strategies. Any memoryless strategy σ induces a subgraph $\mathcal{G}_\sigma$ of the original game graph. If σ is a winning strategy for a player, he wins every play on $\mathcal{G}_\sigma$. Since these subgames are small objects and it can be checked efficiently whether a player wins every play on a given graph, the winner of a finite parity game can be determined in NP ∩ co-NP. In general, the best known deterministic algorithms to decide the winner of a parity game have running times that are polynomial with respect to the size of the game graph, but exponential with respect to the number of different priorities occurring in the game [40]. However, for game graphs of bounded tree width, Obdrzalek has showed in [54], that the problem can be solved in polynomial time with respect to the the size of the graph, independently of the number of priorities.

In the remainder of this chapter we will show that the entanglement of a parity game graph is a pivotal parameter for its computational complexity. To maintain the relationship between games and algorithms conceptually close, we base our analysis on alternating machines (for a comprehensive introduction on alternating computation, see e.g. [4]).

4.4.1 Alternating cycle detection

Many algorithmic issues in graph theory are related to the problem of cycle detection, typically, to determine whether a given graph contains a cycle satisfying certain properties. When alternation comes into play, that is, when we consider paths formed interactively, the questions become particularly interesting but often rather complex, too. In this framework, we will study the entanglement of a graph as a measure of how much memory is needed to determine whether a path formed on-the-fly enters a cycle.

As a basis for later development, let us first consider a procedure for deciding whether k detectives are sufficient to capture the thief on a given graph. The following algorithm represents a straightforward implementation of the game as an alternating algorithm, where the role of the thief is played by the existential player while the detectives are controlled by the universal player.

procedure Entanglement($\mathcal{G}$, v_0, k)
input graph $\mathcal{G} = (V, E)$, initial position v_0, candidate $k \leq |V|$

```
// accept iff ent(G, v_0) ≤ k
v := v_0, (d_i)_{i∈[k]} := ⊥;          // current position of thief and detectives
do
  existentially guess i ∈ [k] ∪ {pass}  // appoint detective i or pass
  if i ≠ pass then d_i := v             // guard current node
  if vE \ d = ∅ then accept
  else universally choose v ∈ vE;
repeat
```

Since this algorithm requires space only to store the current positions of the thief and the k detectives, it runs in alternating space $O((k+1)\log|V|)$ which corresponds to deterministic polynomial time.

Lemma 4.4.1. *The problem of deciding, for a fixed parameter k, whether a given graph* $\mathcal{G}$ *has* $\mathrm{ent}(\mathcal{G}) \leq k$ *can be solved in polynomial time with respect to the size of* $\mathcal{G}$.

Notice that, if we regard the parameter k as part of the input, the algorithm yields an EXPTIME upper bound for the complexity of deciding the entanglement of a graph. At the present time, we do not know whether this bound is strict; even, hardness for *NP* does not seem obvious. To settle the precise complexity of the problem remains subject to further research.

4.4.2 Parity games

Similar to the thief and detective game, the dynamics of a parity game consists in forming a path through a graph. However, while in the former game the detectives can influence the forming process only indirectly, by obstructing ways of return, in a parity game both players determine directly how the path is prolonged in their turn. Besides this dynamic aspect, also the objectives of players are quite different at a first sight. While the detectives aim at turning the play back to a guarded position, each player of a parity game tries to achieve that the least priority seen infinitely often on the path is of a certain parity.

The key insight which brings the two games to a common ground is the Memoryless Determinacy Theorem for parity games: whichever player has a winning strategy in a game $\mathcal{G} = (V, V_0, E, \Omega)$ from a given initial position, also has a memoryless one. This means, that either player may commit, for each reachable position $v \in V$ which he controls, to precisely one successor $\sigma(v) \in vE$ and henceforth

follows this commitment in every play of $\mathcal{G}$ without losing any chance to win. It follows that, whenever a play returns to a previously visited position v, the winner can be established by looking at the least priority seen since the first occurrence of v. Therefore can view parity games on finite game graphs as path forming games of finite duration where the objective is to reach a cycle with minimal priority of a certain parity.

In light of this, we obtain an immediate method to determine the winner of a parity game by simulating the players' moves while maintaining the history of visited positions in order to detect whether a cycle was reached and to retrace the occurring priorities. To store the full history, an implementation of this method requires space $O(|V| \log |V|)$ in the worst case; since the procedure uses alternation to simulate the single game moves, this situates us in $\textsc{Aspace}(O(|V| \log |V|))$, or $\textsc{Dtime}(|V|^{O(|V|)})$.

What makes this approach highly impractical is its extensive representation of the play's history. In fact, the power of alternation is limited to the formation of the path, while the history is surveyed in a deterministic way. We can significantly improve this by interleaving thief and detective games with parity games in such a way that the formation of cycles in history is surveyed interactively.

4.4.3 Interleaving different games

Intuitively, we may think of a parity game as an affair between three agents, Player 0 and 1, and a referee who wishes to establish which of the two indeed wins the game. In our approach, the referee memorises the entire history of the game. But as we have seen, the occurrence of a cycle in a path-forming game on a graph $\mathcal{G}$ can already be detected by storing at most $\mathrm{ent}(\mathcal{G})$ many positions. Hence, if we could provide the referee with the power of sufficiently many detectives, this would reduce the space requirement. The crux of the matter is how to fit such a three-player setting into the two-player model of alternating computation.

Our proposal to overcome this difficulty is to let one of the players act as a referee who challenges the other player in the parity game, but in the same time controls the detectives in an overlying thief and detective game which regards the interactively formed path as if it would be formed by the thief alone.

Formally, this leads to a new game. For a game graph $\mathcal{G} = (V, V_0, E, \Omega)$, a Player $i \in \{0, 1\}$, and a number k, the *superdetective* game $\mathcal{G}[i, k]$ is played between the

Superdetective controlling k detectives and the positions of V_i, and the Challenger in hold of the positions in V_{1-i}. Starting from an initial position position v_0, in any move the Superdetective may place one of the k detectives on the current position v, or leave them in place. If the current position v belongs to V_{1-i}, Challenger has to move to some position $w \in vE$, otherwise the Superdetective moves. (If a player gets stuck, he immediately loses.) The play ends if a position w occupied by a detective is reached and the Superdetective wins if, and only if, the least priority seen since the detective was placed there is even, for $i = 0$ respectively odd, for $i = 1$.

The following lemma states that parity games can be reduced to Superdetective games with an appropriate number of detectives.

Lemma 4.4.2. (i) *If Player i has a winning strategy for the parity game $\mathcal{G}$, then the Superdetective wins the superdetective game $\mathcal{G}[i, k]$ with $k = \mathrm{ent}(\mathcal{G})$.*

(ii) *If for some $k \in \mathbb{N}$, the Superdetective wins the game $\mathcal{G}[i, k]$, then Player i has a winning strategy for the parity game $\mathcal{G}$.*

Proof. Let σ be a memoryless winning strategy of Player i for the game $\mathcal{G}$ and let $\mathcal{G}_\sigma$ be the subgame of $\mathcal{G}$ induced by this strategy. Then, the least priority seen on any cycle of $\mathcal{G}_\sigma$ is favourable to Player i. This remains true for any cycle formed in $\mathcal{G}[i, k]$ where Player i acting as a Superdetective follows the same strategy σ. On the other hand, obviously $\mathrm{ent}(\mathcal{G}_\sigma) \leq \mathrm{ent}(\mathcal{G}) = k$, which means that the Superdetective also has a strategy to place the k detectives so that every path through $\mathcal{G}_\sigma$ will finally meet a guarded position v and hence form a cycle, witnessing that he wins.

For the converse, assume otherwise that Player $1 - i$ has a memoryless winning strategy τ in the parity game $\mathcal{G}$. But then he could follow this strategy when acting as a Challenger in the $\mathcal{G}[i, k]$, so that the play would actually remain in $\mathcal{G}_\sigma[i, k]$ where no cycle is favourable to Player i. Hence, regardless of the number k of detectives, the Superdetective i cannot win, in contradiction to our assumption.

□

Note that computing the winner of a superdetective game $\mathcal{G}[i, k]$ requires alternating space $(2k + 1) \log |V|$. Indeed, one just plays the game recording the current position of the thief and the current position of each detective, along with the minimal priority that has been seen since he was last posted.

procedure Superdetective($\mathcal{G}$, v_0, j, k)
input parity game $\mathcal{G} = (V, V_0, E, \Omega)$, initial position $v_0 \in V$, Player j, k detectives
// accept iff Superdetective has a winning strategy in $\mathcal{G}[j, k]$ with k detectives

$v := v_0$ // current position
$(d_i)_{i\in[k]} := \bot$ // positions guarded by detectives
$(h_i)_{i\in[k]} := \bot$ // most significant priorities
repeat
  if $j = 0$ **then**
    existentially guess $i \in [k] \cup \{\text{pass}\}$ // appoint detective i or pass
  else
    universally choose $i \in [k] \cup \{\text{pass}\}$ // other player's detective
  if $i \neq$ pass **then**
    $d_i := v$; $h_i := \Omega(v)$ // guard current node
  $v := \text{Move}(\mathcal{G}, v)$ // simulate a game step
  forall $i \in [k]$ **do** // update history
    $h_i := \min(h_i, \Omega(v))$
  repeat
until ($v = d_i$ for some i) // cycle detected
if ($j = 0$ and h_i is even) or ($j = 1$ and h_i is odd) **then accept**
else reject

We are now ready to prove that parity games of bounded entanglement can be solved in polynomial time. In fact we establish a more specific result, taking into account the minimal entanglement of subgames induced by a winning strategy.

Theorem 4.4.3. *The winner of a parity game $\mathcal{G} = (V, V_0, E, \Omega)$ can be determined in* $\textsc{Aspace}(\mathcal{O}(k \log |V|))$, *where k is the minimum entanglement of a subgame $\mathcal{G}_\sigma$ induced by a memoryless winning strategy σ in $\mathcal{G}$.*

Proof. We first describe the procedure informally, in the form of a game. Given a parity game $\mathcal{G} = (V, V_0, E, \Omega)$ and an initial position v_0, each player i selects a number k_i and claims that he has a winning strategy from v_0 such that $\text{ent}(\mathcal{G}_\sigma) \leq k_i$. The smaller of the two numbers k_0, k_1 is then chosen to verify that Superdetective wins the game $\mathcal{G}[i, k_i]$. If this is the case the procedure accepts the claim of Player i, otherwise Player $(1 - i)$ is declared the winner.

Here is a more formal description of the procedure:

procedure SolveParity($\mathcal{G}, v$)
input parity game $\mathcal{G} = (V, V_0, E, \Omega)$, initial position $v \in V$
// accept iff Player 0 wins the game
existentially guess $k_0 \leq |V|$
universally choose $k_1 \leq |V|$

if $k_0 \leq k_1$ **then**
  if Superdetective$(\mathcal{G}, v, 0, k_0)$ **then accept**
  else reject
else
  if Superdetective$(\mathcal{G}, v, 1, k_1)$ **then reject**
  else accept
endif

We claim that Player 0 has a winning strategy in a parity game $\mathcal{G}, v$ if, and only if, the alternating procedure ParitySolve$(\mathcal{G}, v)$ accepts.

To see this, assume that Player 0 has a memoryless winning strategy σ from v. Then, the guess $k_0 := \mathrm{ent}(\mathcal{G}_\sigma)$ leads to acceptance. Indeed, for $k_1 \geq k_0$, Player o wins the superdetective game $\mathcal{G}[0, k_0]$ by using the strategy σ as a parity player together with the detective strategy for $\mathcal{G}_\sigma$. On the other hand, for $k_1 < k_0$, the procedure accepts as well, since Player 1 cannot win the superdetective game $\mathcal{G}[1, k_1]$ without having a winning strategy for the parity game.

The converse follows by symmetric arguments exchanging the roles of the two players. □

Corollary 4.4.4. *Parity games of bounded entanglement can be solved in polynomial time.*

5 The μ-Calculus Variable Hierarchy

The μ-calculus extends basic modal logic by adding monadic variables bound by least and greatest fixed points of definable operators. As we have seen, this provides a notion of recursion which invests the logic with very high expressive power.

On the other hand, the variables also import a considerable conceptual complexity. The alternation depth of L_μ-formulae is a well-studied measure of conceptual complexity. Since the hierarchy induced by this measure is semantically strict, this notion of syntactic complexity of a formula is reflected in its semantic complexity.

Interestingly, most of the formalisms commonly used for process description allow translations into low levels of the L_μ alternation hierarchy. On its first level this hierarchy already captures, for instance, PDL as well as CTL, while their expressive extensions ΔPDL and CTL* do not exceed the second level. Still, the low levels of this hierarchy do not exhaust the significant properties expressible in L_μ. As stated in Theorem 1.3.11, e.g., the formula W^n stating that the first player has a winning strategy in parity games of index n.

By reusing fixed point variables several times it is possible to write many L_μ-formulae, even with highly nested fixed-point definitions, using only very few variables. This is actually the case for GL which subsumes the aforementioned formalisms, ΔPDL and CTL*, but also contains formulae describing the winning position of a parity game.

In this context, the question arises, whether a higher number of variables is indeed necessary, or, in other words, whether the number of variables of a formula is reflected as a measure of its semantic complexity.

In the previous chapter we have analysed the descriptive complexity of formulae defining the simulation types of finite Kripke structures showing that the number of variables needed to describe such structures up to bisimulation, or up to simulation, is captured by their entanglement. In the present chapter we will prove that the

variable hierarchy of the μ-calculus is indeed strict, by first showing that this number of variables is indeed required, if we allow only existential modalities. Then, we prove an existential preservation theorem for the family of L_μ-formulae over at most k variables that define simulation types of finite strongly connected structures. Since hard formulae for the level k of the existential hierarchy belong to this family, this leads us to the strictness of the hierarchy in the general case.

5.1 The existential hierarchy

As we have seen in the previous section, the entanglement of a graph provides an upper bound for the number of variables required to describe any Kripke structure over this graph. However, the descriptive complexity does not depend only on the underlying graph, but also on the labelling of transitions and states with actions and atomic propositions. For instance, the simulation type of any strongly connected Kripke structure over a language with only one action and no atomic properties is described by the formula $\nu X.\langle a\rangle X$, regardless of the entanglement of the underlying graph.

In the sequel of this article we show that, with a particular labelling of edges, the structural complexity of a graph, in terms of entanglement, is reflected in the descriptive complexity of its simulation type measured by the number of variables needed to describe it in the μ-calculus.

Definition 5.1.1. A Kripke structure $\mathcal{K}$ is *deterministic* if every state $v \in V$ has at most one a-successor, for all actions $a \in \textsc{Act}$; it is *co-deterministic* if every state has at most one a-predecessor, for all actions a. Further, we say that a structure is *singular* with respect to simulation, if there are no two states $v \neq w$ such that $\mathcal{K}, v \lesssim \mathcal{K}, w$. A finite structure *rigid*, if it is deterministic, co-deterministic, and singular with respect to simulation.

Lemma 5.1.2. *Every connected finite graph can be labelled in such a way that the resulting Kripke structure is rigid.*

Proof. Given a finite graph $\mathcal{G} = (V, E)$, the Kripke structure which assigns to every edge $(v, w) \in E$ a distinct action label is obviously rigid. Formally, this yields a structure over a set of actions $\textsc{Act} := E$, with state domain V and singleton transition relations $E_{vw} := \{(v, w)\}$, for all $(v, w) \in E$. □

According to Proposition 4.3.1, the simulation type of any structure with entanglement k can be described by an existential formula in $\mathrm{L}_\mu[k]$. In this section we prove that, if the structure is rigid, no existential formula from $\mathrm{L}_\mu[k-1]$ can describe its simulation type. This establishes that the variable hierarchy of is strict for the existential fragment of L_μ.

For a simple example of a rigid Kripke structure with entanglement k, consider the complete graph over k vertices labelled as in the previous lemma.

Our argument pivots around the model-checking game $\mathcal{G}(\mathcal{K}, \psi)$ associated to a Kripke structure $\mathcal{K}, u$ and a formula ψ defining its simulation type. Obviously, Verifier has a winning strategy in this game. In general, we may understand the subgame $\mathcal{G}_\sigma$ induced by a (memoryless) winning strategy σ of Verifier in $\mathcal{G}(\mathcal{K}, \psi)$ as a proof for $\mathcal{K}, u \vDash \psi$. We will argue that, on the one hand, if $\mathcal{K}$ is rigid, the entanglement of such a proof cannot be lower than the entanglement of $\mathcal{K}$ itself. On the other hand, we will show that this proof is already contained in the syntax graph of ψ, and hence its entanglement is at least as high as the number of variables used in ψ.

5.1.1 Definite formulae

The rigidity of a structure ensures that the simulation types of its states do not overlap. This allows us to narrow the gap between the semantics and the syntax of formulae ψ describing the simulation type of a rigid structure $\mathcal{K}, u$. Concretely, we show that the proof of $\mathcal{K}, u \vDash \psi$, i.e., the subgame induced by a winning strategy in the associated model-checking game, can be embedded into the syntax graph of ψ.

Definition 5.1.3. We call a formula ψ *definite* on a Kripke structure $\mathcal{K}$, if for every subformula $\eta \in \mathrm{cl}(\psi)$, there exists precisely one state v such that $\mathcal{K}, v \vDash \eta$.

The notion is meaningful only over structures without propositional symbols. Notice that, if we consider rigid structures under this proviso, the formulae constructed in Proposition 4.3.1 as a description of their simulation type are indeed definite.

Lemma 5.1.4. *Let $\mathcal{K}$ be a rigid Kripke structure with a designated state u. Then, every existential formula $\psi \in \mathrm{L}_\mu$ defining the simulation type of $\mathcal{K}, u$ can be transformed, without increasing the number of variables, into an equivalent existential formula ψ' that is definite on $\mathcal{K}$.*

Proof. First, we dispose of the subformulae of ψ that do not hold at any node of $\mathcal{K}$. Let ψ' be the formula obtained from ψ by replacing every such subformula with $\bot$. Then, ψ' is still true on $\mathcal{K}$ and, being existential, on all models of ψ. On the other hand, ψ' obviously implies ψ so that we have $\psi' \equiv \psi$.

Further, we successively eliminate all subformulae true at more than one node. Assume that for some $\eta \in \mathrm{cl}(\psi)$ we have $\mathcal{K}, v_1 \vDash \eta$ and $\mathcal{K}, v_2 \vDash \eta$ with $v_1 \neq v_2$ and let ψ' be the formula obtained from ψ by replacing η with $\top$.

Clearly, ψ implies ψ'. To prove the converse, we will construct for every tree an extension that satisfies η at all nodes while preserving the validity of ψ. Notice that every extension of a tree $\mathcal{T}$ is similar to $\mathcal{T}$. Consequently, existential formulae are preserved under extensions.

Let $\mathcal{T}$ be a tree with edges labelled by Act. We establish a matching correspondence between the nodes of $\mathcal{T}$ and $\mathcal{K}$ in the following way. For any node $x \in T$, consider the sequence of actions on the path from the root to x. In $\mathcal{K}$, there is at most one node w reachable from the designated root u via this sequence of actions, since the structure is deterministic. We set $\mathrm{match}_T(x) := w$, if the sequence is indeed executable in $\mathcal{K}$, otherwise we leave the value undefined. Now let $\mathcal{T}'$ be the extension of $\mathcal{T}$ obtained by adding at every node x the unravelling $\mathcal{T}_w^{\mathcal{K}}$ of $\mathcal{K}$ from the node $w := v_1$ if $\mathrm{match}_T(x) \neq v_1$ and $w := v_2$ otherwise.

Claim. *For any tree $\mathcal{T}$, the constructed extension $\mathcal{T}'$ has the following properties:*

(i) *If $\mathcal{T} \vDash \psi'$ then $\mathcal{T}' \vDash \psi$.*

(ii) *If $\mathcal{T}' \vDash \psi$ then $\mathcal{T} \vDash \psi$.*

(i) Every subtree of $\mathcal{T}'$ rooted at a node $x \in T$ extends an unravelling $\mathcal{T}_w^{\mathcal{K}}$ of $\mathcal{K}, w$, where η holds. Since η is existential and thus preserved under extensions, it follows that also $\mathcal{T}'_x$, the subtree of $\mathcal{T}'$ rooted at x, is a model of η. Moreover, if σ_w is a winning strategy for Verifier in $\mathcal{G}(\mathcal{T}_w^{\mathcal{K}}, \eta)$, it will also be a winning strategy in $\mathcal{G}(\mathcal{T}'_x, \eta)$.

By means of this, we can extend any winning strategy σ of Verifier in $\mathcal{G}(\mathcal{T}, \psi')$ to a strategy in $\mathcal{G}(\mathcal{T}', \psi)$ as follows. At every position (x, φ) where $x \in T$ and $\varphi \neq \eta$ choose according to σ. As Falsifier cannot move in the tree, the play will stay on nodes of $\mathcal{T}$ unless a position (x, η) is reached. When this occurs, Verifier drops σ and proceeds with the strategy σ_w which is winning in $\mathcal{G}(\mathcal{T}_w^{\mathcal{K}}, \eta)$ and thus in $\mathcal{G}(\mathcal{T}'_x, \eta)$. In that way, every play of $\mathcal{G}(\mathcal{T}', \psi)$ is won by Verifier which means that $\mathcal{T}' \vDash \psi$.

(ii) Assuming that $\mathcal{T}' \models \psi$, let Z be a simulation relation witnessing that $\mathcal{K}, u \lesssim \mathcal{T}'$. Then, the relation

$$Z' := \{ (v, x) \in Z \mid x \in T \text{ and } v = \mathrm{match}_{T'}(x) \}$$

is a simulation from $\mathcal{K}, u$ to $\mathcal{T}$.

Obviously, Z' relates u with the root of $\mathcal{T}$. Since Z is a simulation, for any $(v, x) \in Z'$, $a \in \textsc{Act}$, and every a-successor u of v there exists an a-successor y of x such that $(u, y) \in Z$. Clearly, $\mathrm{match}_{T'}(y) = u$, so we just need to show that $y \in T$. Let us assume, towards a contradiction, that y is a new node, $y \in T' \setminus T$. Then, in $\mathcal{K}$ there is a node $w \neq \mathrm{match}_T(x)$ with a-successor u', so that $y \sim u'$. On the other hand, $(u, y) \in Z$, hence $u \lesssim y$. As $\mathcal{K}$ is singular with respect to simulation, this means that u and u' are actually the same node. But then this node would have two different a-predecessors, w and v, in contradiction to the fact that $\mathcal{K}$ is co-deterministic. Hence, Z' witnesses the simulation $\mathcal{K} \lesssim \mathcal{T}$. This proves the second part of our claim and we can conclude that $\psi' \equiv \psi$.

Notice that whenever the subformulae $\langle a \rangle \top$ and $\eta \vee \top$ occur, they are also removed as they hold at more than one node; if the atom $\top$ appears as a conjunct we can safely drop it.

With this rewriting, ψ will eventually consist only of subformulae satisfied at precisely one node of $\mathcal{K}$. □

In particular, definiteness implies that every fixed-point definition holds at precisely one state. Accordingly, for any winning strategy σ in this game, the projection $(v, \varphi) \mapsto \varphi$ induces an embedding of $\mathcal{G}_\sigma(\mathcal{K}, \psi)$ into the syntax graph $\mathcal{S}_\psi$.

Corollary 5.1.5. *Let ψ be an existential formula that is definite on a rigid structure $\mathcal{K}$ and assume $\mathcal{K}, u \models \psi$. Then, for any winning strategy σ for Verifier in the game $\mathcal{G}(\mathcal{K}, \psi)$ the induced subgame $\mathcal{G}_\sigma(\mathcal{K}, \psi)$, is embeddable into the syntax graph of ψ.*

5.1.2 Cast structure

Up to now, we have seen that, in our specific setting, any formula describing a structure contains a proof of its validity on that structure. In the next step we argue that, moreover, this proof essentially contains (a bisimilar copy of) the structure in question.

Observe that a model-checking game does not necessarily explore the entire structure on which it is played. For example, if we are interested in the property $\mu X.(\langle a\rangle X \vee \langle b\rangle \top)$ expressing that a b-transition is reachable in the model, a winning strategy for Verifier would just display an a-path ending with a b-transition. To capture the part of a model explored by a winning strategy, we introduce the notion of structure cast by a strategy.

Definition 5.1.6. Given a Kripke structure $\mathcal{K}$, and an existential L_μ-formula ψ such that $\mathcal{K}, u \vDash \psi$, let σ be a winning strategy for Verifier in the model-checking game $\mathcal{G}(\mathcal{K}, \psi)$, inducing the subgame $\mathcal{G}_\sigma$. The *cast* of σ, is the Kripke structure $\check{\mathcal{G}}_\sigma = (\check{V}, (\check{E}_a)_{a\in \mathrm{ACT}})$ over a subset $\check{V}$ of vertices from $\mathcal{G}_\sigma$ consisting of the root (u, ψ) and the target (w, η) of every possible move $(v, \langle a\rangle\eta) \to (w, \eta)$ in $\mathcal{G}_\sigma$. Between two of these vertices (v, φ), (w, η) we allow an $\check{E}_a$-transition if in $\mathcal{G}_\sigma$ there is a path from (v, φ) to a predecessor $(v, \langle a\rangle\eta)$ of (w, η) which avoids $\check{V}$.

Model-checking games are constructed out of a Kripke structure and a formula. By casting a winning strategy we perform a reverse operation, where we set out from a specific game, or a proof, and extract the relevant structural component. In line with this intuition, the following lemma points out that every model-checking game for an existential formula contains a model of the formula.

Lemma 5.1.7. *Let ψ be an existential L_μ-formula and let $\mathcal{K}, u$ be a model of ψ. Then, for any winning strategy σ of Verifier in the model-checking game $\mathcal{G} := \mathcal{G}(\mathcal{K}, \psi)$, the cast of $\mathcal{G}_\sigma$ at state (u, ψ) is also a model of ψ.*

Proof. We show that Verifier wins the model-checking game $\mathcal{G}' := \mathcal{G}(\check{\mathcal{G}}_\sigma, \psi)$ for ψ on the cast structure starting from position $((u, \psi), \psi)$. Towards this, we perform a generic play of $\mathcal{G}'$ while replicating, on the side, a play of $\mathcal{G}$ according to the Verifier strategy σ. Thereby we transfer every move of Falsifier from $\mathcal{G}'$ to $\mathcal{G}$ and, conversely, every move of Verifier back to $\mathcal{G}'$ so that the parallel plays maintain the following invariant in each turn: whenever the current position in $\mathcal{G}'$ is $((v, \alpha), \beta)$, the current position in $\mathcal{G}$ is (v, β).

This is done in the following way. At the starting position, the proposed invariant obviously holds. If Falsifier moves in the main game $\mathcal{G}'$ from some position $((v, \alpha), \eta_1 \wedge \eta_2)$ to $((v, \alpha), \eta_i)$, we move in the secondary game $\mathcal{G}$ from the current position $(v, \eta_1 \wedge \eta_2)$ to (v, η_i). If Verifier is in turn to move in the main game $\mathcal{G}'$ at a disjunction, e.g., $((v, \alpha), \eta_1 \vee \eta_2)$, the current position in the secondary game $\mathcal{G}$ is $(v, \eta_1 \vee \eta_2)$. In this case we first execute the move in $\mathcal{G}$ to (v, η_i) according to σ

and then choose $((v, \alpha), \eta_i)$ in $\mathcal{G}'$. Similarly, when the current position in the main game is $((v, \alpha), \langle a \rangle \beta)$, and, hence, $(v, \langle a \rangle \beta)$ in the secondary game, we first perform in $\mathcal{G}$ the move (w, β) indicated by σ and then choose $((w, \beta), \beta)$ in $\mathcal{G}'$.

It can be easily checked that the choices transferred between the games are always available. Particularly, in the case of modal moves, this follows from the definition of the cast structure. Hence, Verifier can always move in $\mathcal{G}'$ if he can move in $\mathcal{G}$. Since σ is a winning strategy for the latter game, a play can end only at positions $(v, \top)$ in which case the play of $\mathcal{G}'$ also reaches a terminal position $((w, \beta), \top)$, where Verifier wins. Otherwise, both plays are infinite and the sequences of formulae they visit are the same; accordingly, thanks to his winning strategy in $\mathcal{G}$, Verifier simultaneously wins $\mathcal{G}'$. □

For the case of formulae describing the simulation type of a Kripke structure, the cast of a winning strategy must, hence, be similar to the structure itself. Moreover, for structures that are singular with respect to simulation, this relation has natural witnesses.

Lemma 5.1.8. *Given a $\lesssim$-singular structure $\mathcal{K}, u$, let $\psi \in \mathrm{L}_\mu$ be an existential formula describing its simulation type, and let σ be a winning strategy for Verifier in $\mathcal{G}(\mathcal{K}, \psi)$. Then, there exists a simulation from $\mathcal{K}, u$ to $\check{\mathcal{G}}_\sigma, (u, \psi)$ such that, $Z \subseteq \{ (v, (v, \varphi)) \mid \mathcal{K}, v \vDash \varphi \}$.*

Proof. According to Lemma 5.1.7, we have $\check{\mathcal{G}}_\sigma, (u, \psi) \vDash \psi$. As ψ describes the simulation type of $\mathcal{K}, u$, this implies that $\mathcal{K}, u \lesssim \check{\mathcal{G}}_\sigma, (u, \psi)$. Among the simulations witnessing this, let Z be minimal (with respect to set inclusion). Then, for any pair $(v, (w, \varphi)) \in Z$, we have $\mathcal{K}, v \lesssim \check{\mathcal{G}}_\sigma, (w, \varphi)$. On the other hand, we also have $\check{\mathcal{G}}_\sigma, (w, \varphi) \lesssim \mathcal{K}, w$, since $\mathcal{G}_\sigma$ is a model checking game and the modal moves follow the transitions of $\mathcal{K}$. Thus, we obtain $\mathcal{K}, v \lesssim \mathcal{K}, w$ and, by our assumption that $\mathcal{K}$ is $\lesssim$-singular, it follows that $v = w$. □

It is not hard to show that a relation Z of the above kind is, in fact, a bisimulation between $\mathcal{K}'$ and the structure induced by its range in $\check{\mathcal{G}}_\sigma$.

5.1.3 The separation theorem

As a last step towards proving that simulation-type descriptions are hard formulae for the existential variable hierarchy, we show that the entanglement of the structure

is bounded by the number of variables of any existential formula describing its simulation type.

Lemma 5.1.9. *Let $\mathcal{K}$ be a $\lesssim$-singular structure with a distinguished node u. If there exists an existential formula $\psi \in \mathrm{L}_\mu[k]$ that is definite on $\mathcal{K}$ and describes the simulation type of $\mathcal{K}, u$, then* $\mathrm{ent}(\mathcal{K}) \leq k$.

Proof. According to Lemma 1.3.4, we may assume that ψ is guarded. Let $\psi_1, \dots \psi_n$ be the fixed-point subformulae in ψ, i.e., $\psi_i = \lambda Y.\varphi_i$ with $\lambda \in \{\mu, \nu\}$ and $Y \in \{X_0, \dots, X_{k-1}\}$, and let $\mathrm{cl}(\psi_i)$ be the closure of ψ_i in ψ. Recall that by the definiteness of ψ, there is a unique node v_i with $\mathcal{K}, v_i \vDash \mathrm{cl}(\psi_i)$. Recall further that ψ_j *depends* on ψ_i, if in the syntax graph S_ψ (which is a tree with back edges), the node ψ_i is active at ψ_j, i.e., there is a descendent Y of ψ_j with a back-edge to ψ_i.

Consider a winning strategy σ for Verifier in the model checking game associated to $\mathcal{K}, u \vDash \psi$. By Corollary 5.1.5, the induced subgame $\mathcal{G}_\sigma$ is embedded in the syntax graph $\mathcal{S}_\psi$ via the projection $(v, \varphi) \mapsto \varphi$. On the other hand, its cast simulates $\mathcal{K}, u$. We fix a simulation Z from $\mathcal{K}, u$ to $\check{\mathcal{G}}_\sigma$, (u, ψ) as in Lemma 5.1.8.

On the basis of this, we define a strategy for k detectives in the entanglement game on $\mathcal{K}$ starting at u. To each state v of $\mathcal{K}$ reached by the thief in a play against this strategy, we will associate a position (v, φ) in $\check{\mathcal{G}}_\sigma$ such that $(v, (v, \varphi)) \in Z$.

The initial state u is associated to position (u, ψ). Suppose that, in a round of the play, the thief sits at some position v in $\mathcal{K}$ which is associated to (v, φ) in $\check{\mathcal{G}}_\sigma$. Each free variable X_j in φ is defined at a fixed-point subformula $\psi_{j,\varphi} \in \{\psi_1, \dots, \psi_n\}$ and, by definiteness, there exists precisely one state $v_{\varphi,j}$ in $\mathcal{K}$ where the closure $\mathrm{cl}(\psi_{j,\varphi})$ holds. The strategy of the detectives is to move those detectives $j < k$ to v for which $v_{j,\varphi} = v$. If now the thief, in turn, moves from v to some successor w not occupied by any detective, we associate with w a successor (w, ϑ) of (v, φ) in $\check{\mathcal{G}}_\sigma$ such that $(w, (w, \vartheta)) \in Z$, and proceed to the next round. Lemma 5.1.8 guarantees that a suitable successor (w, ϑ) always exists in $\check{\mathcal{G}}_\sigma$. Accordingly, in $\mathcal{G}_\sigma$ there is a path from (v, φ) leading to (w, ϑ) via positions of the form (v, φ'). This establishes a correspondence between plays of the entanglement game on $\mathcal{K}, u$ and paths in $\mathcal{G}_\sigma$ and furthermore, their projections to paths in $\mathcal{S}_\psi$.

We shall prove that the strategy defined in this way is winning for the detectives. Towards a contradiction, assume that the thief can form an infinite path π from u through $\mathcal{K}$ when playing against this strategy. We look at the associated path π' through $\mathcal{G}_\sigma$ and at its projection π'' to a path through the syntax graph S_ψ. Since π, and hence π'' is infinite, some fixed-point definition ψ_j must be regenerated

infinitely often on π''. We want to show that this cannot happen.

Indeed, suppose that at node (v, φ) the fixed-point formula ψ_i is regenerated. This means that there is a variable X_j such that $\psi_{j,\varphi} = \psi_i$ and $v_{j,\varphi} = v$. Since ψ is guarded, X_j must be free in φ. By definiteness, any next regeneration of ψ_i must also take place at v. But, at the moment when the thief moves from v to w, detective j is at v and stays there until, on the corresponding path π'' a new fixed point formula ψ_ℓ with the same variable X_j is opened, and a node $v' \neq v$ is reached where $\mathrm{cl}(\psi_\ell)$ holds. Before this has happened, the thief cannot move back to v.

Thus, in order to have a further regeneration of ψ_i the path π'' must go through the following steps:

1. From ψ_i the path proceeds to a fixed point definition $\psi_m = \lambda Y.\varphi_m$ with a different variable $Y \neq X_j$ so that ψ_m depends on ψ_i (i.e., ψ_i is active at ψ_m);
2. from there the path must reach a definition $\psi_\ell = \lambda X_j \varphi_\ell$, so that in the corresponding path on $\mathcal{K}$, the detective j is lured away from v;
3. then the path must regenerate Y to ψ_m, and
4. proceed from ψ_m to X_j where it can finally regenerate ψ_i.

Hence, we have seen that between any two regenerations of ψ_i on π'' we must have a regeneration of a formula ψ_m that depends on ψ_i. As a consequence, all fixed point formulae are regenerated only finitely often on π''. □

At this point we are ready to state our separation theorem.

Theorem 5.1.10. *Let $\mathcal{G}$ be a finite directed graph of entanglement k such that every node of $\mathcal{G}$ is reachable from u. Then, there exists a Kripke structure $\mathcal{K}$ over $\mathcal{G}$ so that the simulation type of $\mathcal{K}, u$ can be described by an existential formula in $\mathrm{L}_\mu[k]$, but not by any existential formula in $\mathrm{L}_\mu[k-1]$.*

Proof. According to Lemma 5.1.2, it is possible to assign transition labels to the edges of $\mathcal{G}$ so that the resulting Kripke structure $\mathcal{K}$ is rigid; no atomic atomic predicates are set.

Since $\mathrm{ent}(\mathcal{K}) = k$, an existential formula $\chi \in \mathrm{L}_\mu[k]$ describing the simulation type of $\mathcal{K}, u$ can be constructed, according to Proposition 4.3.1.

Towards a contradiction, assume that there is an existential formula $\psi \in \mathrm{L}_\mu[k-1]$ defining the simulation type of $\mathcal{K}, u$. According to Lemma 5.1.4, we can assume without loss of generality, that ψ is definite. But then, by Lemma 5.1.9 it would follow that $\mathrm{ent}(\mathcal{K}) \leq k - 1$. □

As a conclusion, this shows that every existential formula describing the simulation type of a k-entangled rigid structure requires at least k variables. However, this does not yet exclude the existence of equivalent L_μ-formula over fewer variables but with universal modalities.

5.2 An existential preservation theorem

The key argument in our proof of the hierarchy theorem consists in the following preservation property, which implies that the formulae we used to separate the hierarchic levels of the existential fragment also witness the strictness of the variable hierarchy in the case of the full μ-calculus.

Theorem 5.2.1. *Let $\mathcal{K}$ be a finite Kripke structure over a strongly connected graph. Then every formula $\psi \in L_\mu[k]$ that defines the simulation type of a state $\mathcal{K}, u$ is equivalent to an existential formula $\psi' \in L_\mu[k]$.*

To show that universal modalities can be safely eliminated from any formula ψ of the considered kind, we take a detour and first show that they can be eliminated from the formula expressing that some node at which ψ holds is reachable. To refer to this formula, we use a shorthand borrowed from temporal logics:

$$\mathsf{F}\psi := \mu X.\psi \vee \bigvee_{a \in \mathrm{Act}} \langle a \rangle X.$$

Lemma 5.2.5 in the second part of this section then states that from any formula equivalent to $\mathsf{F}\psi$, an existential formula equivalent to ψ can be recovered without increasing the number of variables.

Lemma 5.2.2. *Let $\mathcal{K}$ be a finite strongly connected structure with a distinguished state u and let $\psi^{\mathcal{K}}$ be a formula defining the simulation type of $\mathcal{K}, u$. Then, every formula $\chi \equiv \mathsf{F}\psi^{\mathcal{K}}$ can be transformed, without increasing the number of variables, into an equivalent formula χ' with the following properties:*

(i) *no universal modalities occur in χ';*

(ii) *χ' is of shape $\mathsf{F}\psi$, where ψ contains no μ-operators;*

(iii) *every formula $\varphi \in \mathrm{cl}(\chi')$ holds at some state of $\mathcal{K}$.*

Proof. (i) Given an L_μ-formula χ, we say that a subformula $\langle a \rangle \varphi$ starting with a diamond is *vital*, if $\mathrm{cl}_\chi(\varphi)$ implies $\mathsf{F}\psi^{\mathcal{K}}$. Dually, a subformula $[a]\varphi$ starting with a box is vital, if the negation $\neg\mathrm{cl}_\chi(\varphi)$ implies $\mathsf{F}\psi^{\mathcal{K}}$.

Eliminating vital boxes. For $\chi \equiv \mathsf{F}\psi^{\mathcal{K}}$, let χ' be the formula obtained by replacing any occurrence of a vital box-subformula $[a]\varphi$ with $\top$. Then, χ obviously implies χ'. For the converse, let us consider a tree model $\mathcal{T}$ of χ'. If, at all its nodes, $\mathcal{T}, v \vDash [a]\mathrm{cl}_\chi(\varphi)$ holds, then $\mathcal{T} \vDash \chi$. Else, there exists a node $v \in T$ with $\mathcal{T}, v \vDash \langle a\rangle\neg\mathrm{cl}_\chi(\varphi)$. But, since $[a]\varphi$ is vital, this means that $\mathcal{T}, v$ and hence $\mathcal{T}$ verifies $\mathsf{F}\psi^{\mathcal{K}}$. Either way, we obtain $\mathcal{T} \vDash \chi$ and hence $\chi \equiv \chi'$.

Eliminating non-vital modalities. By iterating the above elimination step a finite number of times, we obtain a formula $\chi \equiv \mathsf{F}\psi^{\mathcal{K}}$ without vital box-subformulae. Let now χ' be the formula obtained from χ by substituting simultaneously all remaining (i.e., non-vital) box-subformulae with $\bot$ and all non-vital diamond-subformulae with $\top$.

We will first show that the obtained formula χ' implies χ. Let $\mathcal{T}$ be a tree model of χ' and, for every non-vital subformula $\langle a\rangle\varphi$ of χ, let $\mathcal{T}_\varphi$ be a tree model of $\mathrm{cl}_\chi(\varphi) \wedge \neg\mathsf{F}\psi^{\mathcal{K}}$. Using the latter models, we construct an extension $\mathcal{T}'$ of $\mathcal{T}$ by introducing for every node $v \in T$ and every non-vital subformula $\langle a\rangle\varphi$ of χ, a fresh copy of $\mathcal{T}_\varphi$ to which we connect v via an a-edge.

Since χ' contains no box-subformulae, it is preserved under extensions. Consequently $\mathcal{T}' \vDash \chi'$ and Verifier has a winning strategy σ in the model-checking game $\mathcal{G}(\mathcal{T}', \chi')$. Also, for every tree $\mathcal{T}_\varphi$, Verifier has a winning strategy σ_φ in the game $\mathcal{G}(\mathcal{T}_\varphi, \mathrm{cl}_\chi(\varphi))$. We can combine these strategies, to obtain a winning strategy for Verifier in the game $\mathcal{G}(\mathcal{T}', \chi)$ as follows. Move according to σ unless a position with a non-vital subformula of χ is met; up to that point, the play cannot leave T, otherwise, since $\mathsf{F}\psi^{\mathcal{K}}$ is falsified at any node $w \in T' \setminus T$, any vital subformula $\langle a\rangle\varphi$ would fail at w. Moreover, no subformula $[a]\varphi$ can occur, as it would correspond to a $\bot$ position in $\mathcal{G}(\mathcal{T}', \chi')$. Consequently, σ leads the play to a position $(v, \langle a\rangle\varphi)$ where $v \in T$ and $\langle a\rangle\varphi$ is non-vital. At that event, let the Verifier choose the a-successor at the root of $\mathcal{T}_\varphi$ and proceed with his memoryless winning strategy σ_φ for the remaining game. In this way, Verifier finally wins any play of $\mathcal{G}(\mathcal{T}', \chi)$. Notice that, for all nodes $w \in T' \setminus T$, we have $\mathcal{T}', w \nvDash \mathsf{F}\psi^{\mathcal{K}}$, and hence $\mathcal{T}'$ verifies $\mathsf{F}\psi^{\mathcal{K}}$ (or, equivalently, χ) if, and only if, $\mathcal{T}$ does. Hence, we have the following chain of implications, showing that χ' implies χ:

$$\mathcal{T} \vDash \chi' \implies \mathcal{T}' \vDash \chi' \implies \mathcal{T}' \vDash \chi \implies \mathcal{T} \vDash \chi.$$

For the converse, consider a tree model $\mathcal{T} \vDash \chi$ and, for every (non-vital) subformula $[a]\varphi$ of χ, a tree model $\mathcal{T}_\neg\varphi \vDash \neg\mathrm{cl}_\chi(\varphi) \wedge \neg\mathsf{F}\psi^{\mathcal{K}}$. As in the previous step, we construct an extension $\mathcal{T}'$ of $\mathcal{T}$ by connecting every node $v \in T$ via an a-edge to

a fresh copy of $\mathcal{T}_{\neg\varphi}$, for every subformula $[a]\varphi$ of χ. Since $\chi \equiv \mathsf{F}\psi^{\mathcal{K}}$ is preserved under extensions, $\mathcal{T}'$ is still a model of χ. Let σ be a winning strategy for Verifier in the model-checking game $\mathcal{G}(\mathcal{T}', \chi)$. We will show that σ is also a winning strategy for Verifier in $\mathcal{G}(\mathcal{T}, \chi')$.

Notice that, in $\mathcal{G}(\mathcal{T}', \chi)$ Falsifier has a winning strategy from every position $(v, [a]\varphi)$ with $v \in T$, by moving to the a-successor of v at the root of $\mathcal{T}_{\neg\varphi}$. Consequently, any play according to Verifier's strategy σ will avoid such positions. Besides this, at every position $(v, \langle a\rangle\varphi)$ where $v \in T$ and $\langle a\rangle\varphi$ is a vital subformula of χ, the strategy σ will appoint a successor position (w, φ) with $w \in T$, otherwise, since any a-successor $w' \in T' \setminus T$ falsifies $\mathsf{F}\psi^{\mathcal{K}}$, φ would fail too. Summarising, every play of $\mathcal{G}(\mathcal{T}', \chi)$ according to σ, will avoid universal modalities and meet only nodes $v \in T$, unless a position a non-vital subformula $\langle a\rangle\varphi$ occurs. But under these conditions, we can replicate every play of $\mathcal{G}(\mathcal{T}', \chi)$ according to σ as a play of $\mathcal{G}(\mathcal{T}, \chi')$: in case a non-vital subformula $\langle a\rangle\varphi$ of χ is met in the former game, Verifier immediately wins $\mathcal{G}(\mathcal{T}, \chi')$, since the non-vital diamond-subformulae have been replaced by $\top$. Otherwise, the outcome of the play is the same for both games and Verifier wins as well.

This concludes the proof that $\chi \equiv \chi'$.

(ii) By the above result, we can assume without loss that $\chi \equiv \mathsf{F}\psi^{\mathcal{K}}$ contains no box-modalities. For n being the number of states in $\mathcal{K}$, let ψ be the formula obtained by replacing every occurrence of a least fixed-point subformula $\mu X.\varphi$ in χ by it's n-th approximant φ^n. Then, by definition of the μ-operator, ψ implies χ and thus $\mathsf{F}\psi$ implies $\mathsf{F}\chi$, which is equivalent to χ. Conversely, since $\mathcal{K}, u \vDash \chi$ and $\mathcal{K}$ has n states, we have $\mathcal{K}, u \vDash \psi$. As ψ is preserved under simulation, this means that $\psi^{\mathcal{K}}$ implies ψ. Accordingly $\mathsf{F}\psi^{\mathcal{K}}$, which is equivalent to χ, implies $\mathsf{F}\psi$. Hence, $\chi \equiv \mathsf{F}\psi$.

Note that the transformation of χ into $\mathsf{F}\psi$ does not increase the number of variables, as we can pick any of the variables already occurring in χ to expand the F-notation.

(iii) By the previous argument, we can assume that χ is of shape $\mathsf{F}\psi$ where ψ contains no boxes, i.e., $\chi = \mu X.\psi \vee \bigvee_a \langle a\rangle X$. Clearly, χ itself holds at every node of $\mathcal{K}$ and therefore, for every transition a occurring in $\mathcal{K}$, there is a node $v \in V$ where $\langle a\rangle\chi$, and thus $\mathrm{cl}_\chi\big(\langle a\rangle X\big)$, holds. Hence, any subformula φ of χ, with $\mathcal{K}, v \nvDash \mathrm{cl}_\chi(\varphi)$ for all v, must actually be a subformula of ψ. Let ψ' be the formula obtained by replacing every such occurrence φ in ψ with $\bot$. On the one hand, ψ' then obviously implies ψ. On the other hand, as $\mathcal{K}, u \vDash \mathsf{F}\psi$, there must exists a node v of $\mathcal{K}$ where ψ holds.

At that node we also have $\mathcal{K}, v \models \psi'$ and, because ψ' is preserved under simulation, this means that $\psi_v^{\mathcal{K}}$ implies ψ'. But then $\mathsf{F}\psi^{\mathcal{K}}$ implies $\mathsf{F}\psi'$ and, by $\mathsf{F}\psi \equiv \mathsf{F}\psi^{\mathcal{K}}$, it follows that $\mathsf{F}\psi$ implies $\mathsf{F}\psi'$. □

Radical formulae and crisp models. Before we proceed towards proving the Preservation Theorem, we will introduce some notions which will be useful in the proof of Lemma 5.2.5

Given a formula $\psi \in \mathrm{L}_\mu$, we call a subformula φ *radical*, if it appears directly under a modal quantifier in ψ. We refer to the closure of radicals in ψ by

$$\mathrm{cl}_0(\psi) := \{\psi\} \cup \{\, \varphi \in \mathrm{cl}(\psi) \mid \langle a\rangle\varphi \in \mathrm{cl}(\psi) \text{ or } [a]\varphi \in \mathrm{cl}(\psi) \text{ for some } a \in \textsc{Act} \,\}.$$

Radical formulae are the first to be met when a play of the model-checking game reaches a new node of the Kripke structure. For this reason, we need to care for game positions carrying radical formulae when merging strategies of different games.

Let $\mathcal{K}, u$ be a model of $\psi \in \mathrm{L}_\mu$ and σ a winning strategy for Verifier in $\mathcal{G}(\mathcal{K}, \psi)$. For any node $v \in V$, we define the *strategic type* of v in $\mathcal{K}, u$ under σ as follows:

$$\mathrm{tp}_\sigma^{\mathcal{K}}(v) := \{\, \varphi \in \mathrm{cl}_0(\psi) \mid \text{position } (v, \varphi) \text{ is reachable in } \mathcal{G}_\sigma(\mathcal{K}, \psi) \,\}.$$

In arbitrary games, the type of a node can be rather complex. However, for existential formulae, Verifier has full control over the moves in the Kripke structure. In the ideal case, he can foresee for every node, a single radical formula to be proved there.

Given a Kripke structure $\mathcal{K}, u$ and a formula ψ, we say that a Verifier strategy σ in the model-checking game $\mathcal{G}(\mathcal{K}, \psi)$ is *crisp*, if the strategic type $\mathrm{tp}_\sigma^{\mathcal{K}}(v)$ of any $v \in V$ consists of not more than one radical. Accordingly, we call a model $\mathcal{K}, u$ of ψ crisp (under σ), if Verifier has a crisp winning strategy σ in the associated model-checking game.

The subsequent lemmas, that can be easily proved, provide us with sharp tools for manipulating models of existential formulae.

Lemma 5.2.3. *Given a structure $\mathcal{K}, u$ every existential formula $\psi \in \mathrm{L}_\mu$ with $\mathcal{K}, u \models \psi$ also has a tree model $\mathcal{T}$ bisimilar to $\mathcal{K}, u$ which is crisp. Moreover, if $\mathcal{K}$ is finitely branching, then $\mathcal{T}$ can be chosen so as well.*

Lemma 5.2.4. *Let $\mathcal{T}$ be a crisp tree model of a formula $\psi \in \mathrm{L}_\mu$ under a strategy σ and let $x \in T$ be a node with strategic type $\mathrm{tp}_\sigma^{\mathcal{T}}(x) = \{\varphi\}$. Then, for every crisp tree model*

$\mathcal{S}$ of φ, the tree $\mathcal{T}[x/\mathcal{S}] \vDash \psi$, obtained by replacing the subtree of $\mathcal{T}$ rooted at x with $\mathcal{S}$, is still a crisp model of ψ.

We are now ready for the final step, the elimination of the F-operator.

Lemma 5.2.5. *Let $\mathcal{K}$ be a finite strongly connected structure with a state u and let $\psi^{\mathcal{K}}$ describe the simulation type of $\mathcal{K}, u$. Then, every formula $\psi \in \mathrm{L}_\mu$ so that $\mathsf{F}\psi \equiv \mathsf{F}\psi^{\mathcal{K}}$ can be transformed, without increasing the number of variables, into a formula ψ' without universal modalities, so that $\psi' \equiv \psi^{\mathcal{K}}$.*

Proof. According to Lemma 5.2.2, we can assume that ψ contains no universal modalities or least fixed point operators and that (the closure of) every subformula is true at some node in $\mathcal{K}$.

We will first show that for any node v in $\mathcal{K}$, there is a subformula φ of ψ whose closure $\mathrm{cl}_\psi(\varphi)$ implies $\psi^{\mathcal{K}}_v$. Actually, we always find a radical formula with this property.

Towards a contradiction, let us assume that $\psi^{\mathcal{K}}_v$ is not implied by any radical subformula of ψ. This means that every $\varphi \in \mathrm{cl}_0(\psi)$ has a tree model $\mathcal{T}_\varphi$ which falsifies $\psi^{\mathcal{K}}_v$. According to Corollary 1.2.26, we can choose $\mathcal{T}_\varphi$ to be a finitely branching tree that falsifies already an approximant of $\psi^{\mathcal{K}}_v$ to some finite stage m_φ. Observe that this approximant $(\psi^{\mathcal{K}}_v)[\nu/\nu^{m_\varphi}]$ is a modal formula. Let us denote its modal depth by n_φ. Further, let us fix a number n which is greater than any n_φ for $\varphi \in \mathrm{cl}_0(\psi)$ and co-prime to every number up to the size of the domain V.

By Lemma 5.2.3, we can assume without loss of generality that each $\mathcal{T}_\varphi$ is a crisp model of φ, this being witnessed by a crisp winning strategy for Verifier in the game $\mathcal{G}(\mathcal{T}_\varphi, \varphi)$. In particular, $\mathcal{T}_\psi$ is a crisp model of ψ. Let σ_ψ be a crisp winning strategy for Verifier in the model-checking game $\mathcal{G}(\mathcal{T}_\psi, \psi)$.

With aid of these, we construct a sequence of trees $(\mathcal{T}_i)_{0 \le i < \omega}$, together with crisp Verifier strategies σ_i witnessing that $\mathcal{T}_i \vDash \psi$. To start, we set $\mathcal{T}_0 := \mathcal{T}_\psi$ and $\sigma_0 := \sigma_\psi$. In every step $i > 0$, the tree $\mathcal{T}_{i+1}$ is obtained from $\mathcal{T}_i$ by performing the following manipulations at depth $n(i+1)$. For each subtree of $\mathcal{T}_i$ rooted at a node x of this depth, we check whether $\mathcal{T}_i, x \vDash \psi^{\mathcal{K}}_v$. If this is not the case, the subtree remains unchanged. Else, we look at the strategic type of x under σ_i. If the type is empty, we simply cut all successors of x. Otherwise, $\mathrm{tp}^{\mathcal{T}_i}_{\sigma_i}(x)$ consists of a single radical formula φ, and we replace the subtree $\mathcal{T}_i, x$ with $\mathcal{T}_\varphi$. According to Lemma 5.2.4, the resulting tree $\mathcal{T}_{i+1}$ is a model of ψ, and the composition of the strategy σ_i with the crisp strategies σ_φ on the newly appended subtrees $\mathcal{T}_\varphi$ yields a crisp Verifier strategy σ_{i+1} for the model-checking game $\mathcal{G}(\mathcal{T}_{i+1}, \psi)$.

By construction, each of the trees $\mathcal{T}_i$ is finitely branching and the sequence $(\mathcal{T}_i)_{0\leq i<\omega}$ converges in the prefix topology of finitely branching trees (see [36]). Let $\mathcal{T}_\omega$ be the limit of this sequence. Since no μ-operators occur in ψ, its model class is topologically closed on finitely branching trees, according to [36]. Consequently, $\mathcal{T}_\omega$ is still a model of ψ. By our hypothesis, ψ implies $\mathsf{F}\psi^{\mathcal{K}}$. Thus, at some depth d in $\mathcal{T}_\omega$ a node x with $\mathcal{T}_\omega, x \vDash \psi^{\mathcal{K}}_v$ appears. Since $\mathcal{K}$ is strongly connected, v must lie on a cycle in $\mathcal{K}$. Hence, for $k \leq |V|$ being the length of such a cycle, there exist nodes y with $\mathcal{T}_\omega, y \vDash \psi^{\mathcal{K}}_v$ at every depth $d + jk$. However, our construction eliminated all subtrees carrying the similarity type of v at depths multiple of n. Since n was chosen to be co-prime to any integer up to $|V|$, it follows that $\mathcal{T}_\omega$ cannot satisfy ψ. This is a contradiction which invalidates our assumption that $\psi^{\mathcal{K}}_v$ is not implied by any $\varphi \in \mathrm{cl}_0(\psi)$.

Hence, for every node $v \in V$, there exists a formula $\varphi_v \in \mathrm{cl}_0(\psi)$ implying $\psi^{\mathcal{K}}_v$. We can show that the converse also holds, if v is maximal with respect to the preorder $\lesssim$, in the sense that for every w with $v \lesssim w$ we have $w \lesssim v$. Recall that, by Lemma 5.2.2 (iii), the formula φ_v must be verified at some node w in $\mathcal{K}$. Since φ_v is existential and thus preserved under extension, it follows that $\psi^{\mathcal{K}}_w$ implies φ_v, which further implies $\psi^{\mathcal{K}}_v$. But this means that $v \lesssim w$ and, by maximality of v, that w and v are bisimilar. Hence, $\mathcal{K}, v \vDash \varphi_v$ and consequently $\psi^{\mathcal{K}}_v \equiv \varphi_v$.

This concludes the proof for the case when u is maximal in $\mathcal{K}$ with respect to $\lesssim$. Otherwise, we could not guarantee, of course, that $\varphi_u \equiv \psi^{\mathcal{K}}_u$. But in that case, a formula equivalent to $\psi^{\mathcal{K}}_u$ can be recovered from $\mathrm{cl}_0(\psi)$ without great difficulty. □

5.3 The hierarchy theorem

Up to now we have showed how to construct, for every level k of the variable hierarchy, existential formulae which are not equivalent to any existential formula from a lower hierarchical level. However, this left open the question whether there exist equivalent formulae in $\mathrm{L}_\mu[k-1]$ which use universal quantification. Due to our Preservation Theorem, we are now able to assert that this cannot be the case.

Theorem 5.3.1. *For every k, there exist formulae $\psi \in \mathrm{L}_\mu[k]$ that are not equivalent to any formula in $\mathrm{L}_\mu[k-1]$.*

Proof. Consider a rigid strongly connected Kripke structure $\mathcal{K}$ of entanglement k and let $\psi^{\mathcal{K}} \in \mathrm{L}_\mu$ be a formula describing the simulation type of $\mathcal{K}, u$ for some state u.

Towards a contradiction, assume that there exists a formula $\psi \in \mathrm{L}_\mu[k-1]$ equivalent to $\psi^{\mathcal{K}}$. Since ψ defines the simulation type of $\mathcal{K}$, a finite strongly connected structure, we can apply Theorem 5.2.1 to conclude that there also exists a formula $\psi' \in \mathrm{L}_\mu[k-1]$ using only existential modalities which is equivalent to $\psi^{\mathcal{K}}$. But this contradicts the separation theorem 5.1.10 for the existential fragment. □

5.3.1 Separating formulae with two modalities

The results of the previous sections provide us with a generic technique to construct witnesses for the L_μ-variable hierarchy. The first examples for the strictness of the existential hierarchy, which turn out to be valid witnesses for the unrestricted case too, have been presented in [11]. They rely on rigid k-cliques where every action is labelled differently, leading to formulae over a vocabulary with k^2 modalities.

To show that already over a fixed vocabulary, the variable hierarchy remains strict. we construct rigid Kripke structures over only two modalities leading to formulae that are strict at each level of the variable hierarchy.

Definition 5.3.2. For every $k > 0$, let $\mathcal{C}^k := (V, E_a, E_b)$ be the Kripke structure with state set $V = [k] \times [k]$ and transition relations

$$\begin{aligned} E_a &:= \{ \big((i,j), (i, j-1) \big) \mid i \geq 0, j > 0 \} \\ &\qquad \cup \{ \big((i,0), (i-1, k-1) \big) \mid i > 0 \} \text{ and} \\ E_b &:= \{ \big((i,j), ((i+j) \bmod k, (j-1) \bmod k) \big) \mid 0 \leq i,j < k \}. \end{aligned}$$

Let us first verify that these structures $\mathcal{C}^k$ indeed fulfil the premises formulated in the proof of the separation theorem 5.1.10.

Lemma 5.3.3. *For every k, the structure $\mathcal{C}^k = (V, E_r, E_s)$ satisfies the following conditions:*

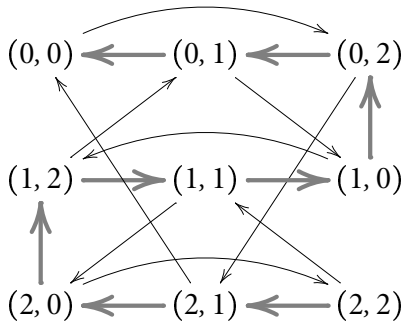
(i) $\mathrm{ent}(\mathcal{C}^k) = k$;

(ii) *$\mathcal{K}$ is deterministic and co-deterministic;*

(iii) *$\mathcal{K}$ is singular with respect to simulation.*

Proof. To prove the first issue, we use our characterisation of entanglement in terms of games, and show that the thief has a winning strategy in any game on $\mathcal{C}^k$ with less than k detectives, but he loses when they come in k or more.

Figure 5.1: The Kripke structure $\mathcal{C}^3$ (a-transitions thicker, b-transitions plain)



We will refer to the rows of the state set $C_i := \{ i, j \mid 0 \leq j < k \}$ as *islands*. Each island induces a cycle, and every two islands are connected by an edge, so that they form a k-clique. Intuitively, if there are less than k cops, at every moment at least one of the islands must be unguarded and the thief can always navigate from his current position to that island without bumping into a detective by pursuing the following strategy: Whenever the current island i is unguarded and, moreover, no detective is on his way to the current position, proceed on i. In the event that a detective is sent to the current position, at least one island j must be left unguarded. Since the current island was previously unguarded, the path from the current position to the safe island j is still free. Hence, set out on this path and follow it until the island j is reached. Upon arrival, j will still be an unguarded island so that the strategy can be reiterated.

In case k or more detectives are available, they can distribute to the different islands, e.g., by following the thief to any position $(i, 0)$ he reaches during the play. Then the robber must move to a fresh island after most $k-1$ steps. But after k times, there are no unguarded islands left, so the thief loses.

It is easily seen that $\mathcal{K}$ is deterministic and co-deterministic.

To verify that it is also singular with respect to $\lesssim$, observe first that the the nodes of $\mathcal{C}^k$ are aligned on the finite r-path from $(k-1, k-1)$ to $(0, 0)$ in descending lexicographic order. Clearly, $(k-1, k-1)$ cannot be simulated by any other node. Let us assume that we have a simulation $u \lesssim v$ between distinct nodes. Since $\mathcal{C}^k$ is strongly connected, there exists a unique path from u to $(k-1, k-1)$. The sequence of actions seen on this path can also be executed starting from v since $u \lesssim v$. By determinism of $\mathcal{C}^k$ the node w reached via this sequence is uniquely

determined. However, since the simulation relation propagates along actions, it follows that $(k-1, k-1) \lesssim w$ which implies $(k-1, k-1) = w$, in contradiction to the co-determinacy of $\mathcal{C}$. □

According to this, the structures $\mathcal{C}^k$ can be used as witnesses in the proof of the separation theorem 5.1.10 yielding strict formulae for each level k of the existential variable hierarchy. Since $\mathcal{C}^k$ is strongly connected, the Existential Preservation Theorem 5.2.1, establishes that these formulae actually witness the strictness of the variable hierarchy of the μ-calculus, already over a language with two modalities only.

Corollary 5.3.4. *For every integer k, there are bimodal existential formulae* $\psi^k \in \mathrm{L}_\mu[k]$ *that are not equivalent to any formula in* $\mathrm{L}_\mu[k-1]$.

We explicitly construct witnessing formulae describing the simulation type of $\mathcal{C}^k, (0,0)$, as in the proof of proof of Proposition 4.3.1. Towards this, we build a sequence of formulae $(\varphi_{i,j})_{0 \le i,j < k}$ over the fixed-point variables $X_0, \ldots, X_{k-1}$ by induction on j, setting for all i simultaneously $\varphi_{i,0} := X_i$ and for every $j > 0$:

$$\varphi_{i,j} := \langle a \rangle \varphi_{i,j-1} \wedge \langle b \rangle \varphi_{i+j,j-1}.$$

Then, we define the system S of rules

$$X_0 := \langle b \rangle \varphi_{0,n-1} \text{ and } X_i := \langle a \rangle \varphi_{i-1,k-1} \wedge \langle b \rangle \varphi_{i,n-1} \text{ for } 0 < i < k.$$

The formula $\nu X_0.S$ obtained as a description for the simulation type of $\mathcal{C}^k$ at state (0.0) is strict for the level k of the variable hierarchy.

Bibliography

[1] A. Arnold, *The mu-calculus alternation-depth is strict on binary trees*, RAIRO Informatique Théorique et Applications, 33 (1999), pp. 329–339.

[2] A. Arnold and D. Niwiski, *Rudiments of μ-calculus*, North Holland, 2001.

[3] R. J. Aumann, *What is game theory trying to accomplish?*, in Frontiers of Economics, K. Arrow and S. Honkapohja, eds., Basil Blackwell, 1985.

[4] J. L. Balcazar, J. Diaz, and J. Gabarro, *Structural complexity 1*, Springer-Verlag, 1988.

[5] D. Berwanger, *Game logic is strong enough for parity games*, Studia Logica, 75 (2003), pp. 205–219. Special issue on Game Logic and Game Algebra edited by M. Pauly and R. Parikh.

[6] D. Berwanger and A. Blumensath, *Automata for guarded fixed point logics*, in Automata, Logics, and Infinite Games, E. Grädel, W. Thomas, and T. Wilke, eds., no. 2500 in LNCS, Springer Verlag, 2002, ch. 19, pp. 343–355.

[7] ——, *The monadic theory of tree-like structures*, in Automata, Logics, and Infinite Games, E. Grädel, W. Thomas, and T. Wilke, eds., no. 2500 in LNCS, Springer Verlag, 2002, ch. 16, pp. 285–301.

[8] D. Berwanger and E. Grädel, *Games and model checking for guarded logics*, in Proceedings of LPAR 2001, Lecture Notes in Computer Science Nr. 2250, Springer, 2001, pp. 70–84.

[9] D. Berwanger and E. Grädel, *Fixed-point logics and solitaire games*, Theory of Computing Systems, 37 (2004), pp. 675 – 694.

[10] D. Berwanger, E. Grädel, and S. Kreutzer, *Once upon a time in the west. Determinacy, complexity and definability of path games*, in Proceedings of the 10th International Conference on Logic for Programming and Automated Reasoning, LPAR 2003, Almaty, M. Vardi and A. Voronkov, eds., vol. 2850 of LNCS, Springer-Verlag, 2003, pp. 226–240.

[11] D. Berwanger, E. Grädel, and G. Lenzi, *On the variable hierarchy of the modal mu-calculus*, in Computer Science Logic, CSL 2002, J. Bradfield, ed., vol. 2471 of LNCS, Springer-Verlag, 2002, pp. 352–366.

[12] D. Berwanger and G. Lenzi, *The variable hierarchy of the μ-calculus is strict*, in STACS 2005, Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science, vol. 3404 of LNCS, Springer-Verlag, 2005, pp. 97–109.

[13] J. Bradfield, *The modal μ-calculus alternation hierarchy is strict*, Theoretical Computer Science, 195 (1998), pp. 133–153.

[14] J. Bradfield and C. Stirling, *Modal logics and mu-calculi: an introduction*, in Handbook of Process Algebra, A. P. J. Bergstra and S. Smolka, eds., Elsevier, North-Holland, 2001.

[15] J. C. Bradfield, *The modal mu-calculus alternation hierarchy is strict*, in Proceedings of the 7th International Conference on Concurrency Theory, CONCUR '96, U. Montanari and V. Sassone, eds., vol. 1119 of Lecture Notes in Computer Science, Springer-Verlag, August 1996, pp. 232–246.

[16] J. C. Bradfield, *Simplifying the modal mu-calculus alternation hierarchy.*, in STACS, 1998, pp. 39–49.

[17] J. Büchi, *On a decision method in restricted second-order arithmetic*, in Proc. 160 Int. Cong. for Logic, Mothodologand Philossophy of Science, Stanford University Press, 1962, pp. 1–11.

[18] A. Ehrenfeucht, *An application of games to the completeness problem for formalized theories*, Fundamenta Mathematicae, 49 (1961), pp. 129–141.

[19] A. Emerson, *Temporal and modal logic*, in Handbook of Theoretical Computer Science, vol B., J. van Leeuwen, ed., Elsevier, 1990, pp. 995–1072.

[20] A. Emerson and C. Jutla, *Tree automata, mu-calculus and determinacy*, in Proc. 32nd IEEE Symp. on Foundations of Computer Science, 1991, pp. 368–377.

[21] E. A. Emerson and C. S. Jutla, *Tree automata, mu-calculus and determinacy (extended abstract)*, in 32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1–4 Oct. 1991, IEEE, pp. 368–377.

[22] M. Fischer and R. Ladner, *Propositional dynamic logic of regular programs*, Journal of Computer and System Sciences, 18 (1979), pp. 194–211.

[23] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi, *On the temporal analysis of fairness*, in POPL '80: Proceedings of the 7th ACM SIGPLAN-SIGACT

symposium on Principles of programming languages, New York, NY, USA, 1980, ACM Press, pp. 163–173.

[24] G. Gottlob, N. Leone, and F. Scarcello, *Robbers, marshals, and guards: Game theoretic and logical characterizations of hypertree width*, in Proc. 20th ACM Symp. on Principles of Database Systems, 2001, pp. 195–201.

[25] E. Grädel, *Why are modal logics so robustly decidable?*, Bulletin of the European Association for Theoretical Computer Science, 68 (1999), pp. 90–103.

[26] ——, *Finite model theory and descriptive complexity*, in Finite Model Theory and Its Applications, Springer-Verlag, 2003. To appear.

[27] E. Grädel, W. Thomas, and T. Wilke, eds., *Automata, Logics, and Infinite Games*, no. 2500 in Lecture Notes in Compter Science, Springer-Verlag, 2002.

[28] E. Grädel and I. Walukiewicz, *Guarded fixed point logic*, in Proc. 14th IEEE Symp. on Logic in Computer Science, 1999, pp. 45–54.

[29] Y. Gurevich and L. Harrington, *Trees, automata, and games*, in Proceedings of the fourteenth annual ACM symposium on theory of computing, 1982, pp. 60–65.

[30] T. Hafer and W. Thomas, *Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree*, in Automata, Languages, and Programming, 14th International Colloquium, ICALP87, vol. 267 of Lecture Notes in Computer Science, Springer, 1987, pp. 269–279.

[31] L. Henkin, *Some remarks on infinitely long formulas*, in Proceedings of the Symposium on Foundations of Mathematics: Infinitistic Methods, Polish Scientific Publishers, 1961, pp. 167–183.

[32] M. C. B. Hennessy and R. Milner, *On observing nondeterminism and concurrency*, in Automata, Languages and Programming, 7th Colloquium, J. W. de Bakker and J. van Leeuwen, eds., vol. 85 of LNCS, Springer-Verlag, 1980.

[33] J. G. Henriksen and P. S. Thiagarajan, *Dynamic linear time temporal logic.*, Ann. Pure Appl. Logic, 96 (1999), pp. 187–207.

[34] J. Hintikka, *Language-games for quantifiers*, in Studies in Logical Theory, N. Rescher, ed., vol. 2 of American Philosophical Quarterly Monograph Series, Basil Blackwell, 1968, pp. 46–72.

[35] W. Hodges, *Logic and games*, in The Stanford Encyclopedia of Philosophy, E. N. Zalta, ed., http://plato.stanford.edu/archives/win2004/entries/logic-games/, Winter 2004.

[36] D. Janin and G. Lenzi, *On the logical definability of topologically closed recognizable languages of infinite trees*, Computing and Informatics, 21 (2002), pp. 185–203.

[37] D. Janin and I. Walukiewicz, *Automata for the modal μ-calculus and related results*, in Proceedings of MFCS 95, Lecture Notes in Computer Science Nr. 969, Springer-Verlag, 1995, pp. 552–562.

[38] ——, *On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic*, in Proceedings of 7th International Conference on Concurrency Theory CONCUR '96, no. 1119 in Lecture Notes in Computer Science, Springer-Verlag, 1996, pp. 263–277.

[39] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas, *Directed treewidth*, J. Comb. Theory Ser. B, 82 (2001), pp. 138–154.

[40] M. Jurdziski, *Small progress measures for solving parity games*, in STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Proceedings, vol. 1770 of Lecture Notes in Computer Science, Springer, 2000, pp. 290–301.

[41] H. W. Kamp, *Tense Logic and the Theory of Linear Order*, PhD thesis, University of California, Los Angeles, 1968.

[42] A. Kanamori, The Higher Infinite, Springer, 1991.

[43] A. Kechris, *Classical Descriptive Set Theory*, Springer, 1995.

[44] D. Kozen, *Results on the propositional μ-calculus*, Theoretical Computer Science, 27 (1983), pp. 333–354.

[45] ——, *A finite model theorem for the propositional μ-calculus*, Studia Logica, 47 (1988), pp. 233–241.

[46] S. A. Kripke, *Semantical analysis of modal logic I: Normal modal propositional calculi*, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, 9 (1963), pp. 67–96.

[47] O. Kupferman, M. Vardi, and P. Wolper, *An automata-theoretic approach to branching-time model checking*, Journal of the ACM, 47 (2000), pp. 312–360.

[48] J. Le Tourneau, *Decision problems related to the concept of operation*, PhD thesis, University of California, Berkeley, 1968.

[49] G. Lenzi, *A hierarchy theorem for the mu-calculus*, in Proceedings of the 23rd International Colloquium on Automata, Languages and Programming,

ICALP '96, F. Meyer auf der Heide and B. Monien, eds., vol. 1099 of Lecture Notes in Computer Science, Springer-Verlag, July 1996, pp. 87–97.

[50] Z. Manna and A. Pnueli, *Temporal Verification of Reactive Systems: Specification*, Springer-Verlag, 1992.

[51] R. Mauldin, ed., *The Scottish Book. Mathematics from the Scottish Café*, Birkhäuser, 1981.

[52] F. Moller and A. M. Rabinovich, *Counting on ctl*: on the expressive power of monadic path logic.*, Information and Computation, 184 (2003), pp. 147–159.

[53] D. Niwinski, *The propositional μ-calculus is more expressive than the propositional dynamic logic of looping.* Unpublished manuscript. 1984.

[54] J. Obdrzalek, *Fast mu-calculus model checking when tree-width is bounded*, in CAV'03, vol. 2725 of LNCS, Springer-Verlag, 2003, pp. 80–92.

[55] R. Parikh, *Propositional game logic*, in IEEE Symposium on Foundations of Computer Science, IEEE, 1983, pp. 195–200.

[56] ——, *The logic of games and its applications*, Annals of discrete mathematics, 24 (1985), pp. 111–140.

[57] M. Pauly, *Game logic for game theorists*, Tech. Rep. INS-R0017, CWI, Amsterdam, September 2000.

[58] ——, *Logic for Social Software*, PhD thesis, University of Amsterdam, 2001.

[59] M. Pistore and M. Vardi, *The planning spectrum – one, two, three, infinity*, in Proc. 18th IEEE Symp. on Logic in Computer Science, 2003.

[60] A. Pnueli, *The temporal logic of programs*, in Proceedings of the 18th Annual IEEE Symposium on the Foundations of Computer Science, 1977, pp. 46–57.

[61] M. Rabin, *Decidability of second-order theories and automata on infinite trees*, Transactions of the AMS, 141 (1969), pp. 1–35.

[62] A. L. Semenov, *Decidability of monadic theories*, in Proceedings of the 11th International Symposium on Mathematical Foundations of Computer Science, MFCS '84, vol. 176 of LNCS, Springer-Verlag, 1984, pp. 162–175.

[63] P. D. Seymour and R. Thomas, *Graph searching and a min-max theorem for tree-width*, J. Comb. Theory Ser. B, 58 (1993), pp. 22–33.

[64] S. Shelah, *The monadic theory of order*, Annals of Mathematics, 102 (1975), pp. 379–419.

[65] C. Stirling, *Bisimulation, modal logic and model checking games*, Logic Journal of the IGPL, 7 (1999), pp. 103–124.

[66] R. S. Streett, *Propositional dynamic logic of looping and converse is elementary decidable*, Information and Control, 54 (1982), pp. 121–141.

[67] A. Tarski, *Contributions to the theory of models I, II*, Indagationes Mathematicae, 16 (1954), pp. 572–588.

[68] W. Thomas, *A combinatorial approach to the theory of omega-automata*, Information and Control, 48 (1981), pp. 261–283.

[69] J. van Benthem, *When are two games the same?* available at http://turing.wins.uva.nl/ johan/TORINO.ps.

[70] J. van Benthem, *Modal Correspondence Theory*, PhD thesis, University of Amsterdam, 1976.

[71] ——, *Exploring Logical Dynamics*, CSLI Publications, Stanford, 1996.

[72] S. Vannucci, *Effectivity functions and stable governance structures*, Annals of Operations Research, 109 (2002), pp. 99–127.

[73] M. Vardi, *Why is modal logic so robustly decidable?*, in Descriptive Complexity and Finite Models, N. Immerman and P. Kolaitis, eds., vol. 31 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS, 1997, pp. 149–184.

[74] J. von Neumann and O. Morgenstern, *The Theory of Games and Economic Behavior*, Princeton Univ. Pres, 1944.

[75] I. Walukiewicz, *Pushdown processes: Games and model checking*, Information and Computation, 164 (2001), pp. 234–263.

[76] ——, *Monadic second-order logic on tree-like structures*, TCS, 275 (2002), pp. 311–346.

[77] P. Wolper, *A translation from full branching time temporal logic to one letter propositional dynamic logic with looping.* Unpublished manuscript, 1982.

[78] ——, *Temporal logic can be more expressive*, Information and Control, 56 (1983), pp. 72–99.

[79] E. Zermelo, *Über eine anwendung der mengenlehre auf die theorie des schachspiels*, in Proceedings of the Fifth Congress of Mathematicians, 1912, pp. 501 – 504.

Symbol Index

Index

Lebenslauf

Dietmar Wilhelm Berwanger

19. Mai 1972	geboren in Lugoj, Rumänien
1978 – 1986	Deutsche Grundschule in Lugoj
1986 – 1990	Lyzeum für Mathematik und Physik in Timişoara
Juni 1990	Bacalaureat (rumänische Hochschulreife)
1991	Geschwister-Scholl-Gymnasium in Mannheim
Dez. 1991	Abitur
1993 – 2000	Studium der Informatik an der RWTH Aachen
Mai 2000	Diplom Informatik
2000 – 2005	Wissenschaftlicher Angestellter am Lehr- und Forschungsgebiet Mathematische Grundlagen der Informatik, RWTH Aachen