Games and Model Checking for Guarded Logics

Dietmar W. Berwanger

Diplomarbeit

vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät der
Rheinisch-Westfälischen Technischen Hochschule Aachen

Angefertigt am
Lehr- und Forschungsgebiet
Mathematische Grundlagen der Informatik
bei

Professor Dr. Erich Grädel

Mai 2000

Diplomarbeit ausgegeben und betreut von Professor Dr. Erich Grädel,
Zweitgutachter Professor Dr. Wolfgang Thomas.

Titel: "Games and Model Checking for Guarded Logics"

Dietmar Wilhem Berwanger
Matrikelnummer 197319
Geboren am 19. Mai 1972
E-Mail: berwanger@informatik.rwth-aachen.de

Textsatz: $\LaTeX\,2_\varepsilon$ & XEmacs unter Linux

Erklärung
Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Aachen, den 29. Mai 2000

(Dietmar W. Berwanger)

Contents

Chapter 1

Introduction

In computer science we use mathematical structures to represent facts about the world. To access our representation we interpret formal languages over these structures, or models. The connection between formal languages, which we call logics, and their models is the subject of model theory.

When choosing a logic to speak about models we are concerned about several issues. Of course, we wish our language to be as *expressive* as possible. It may happen that our language is able to express things we don't consider to make sense, e. g. requests which could never be satisfied or questions whose answer is always the same. So it is desirable to have a criterion of *satisfiability*, telling us which expressions are reasonable. Finally, once an assertion is formulated it is important to have tractable means to verify whether it holds or not. The latter problem is called the *model checking* problem and constitutes the subject of our work.

For both satisfiability and model checking we would like to have algorithms that are as efficient as possible. In other words, we want our logics to be *algorithmically manageable*. The demands for expressiveness and manageability turn out to be contradictory. The more expressive a logic is, the harder it is to decide which sentences are satisfiable or to evaluate them. So we find ourselves in the situation of tailoring logics towards a good balance between expressiveness and feasibility.

From modal logic to guarded fixed point logics. The guarded logics underlying to our work are located on the crossroads between expressive power and reasonably efficient decidability. The *guarded fragment of first-order logic* introduced by Andréka, van Benthem, and Németi in [AvBN98] can be understood as an extension of the propositional *modal logic*, ML. This logic extends propositional logic by the modal operators $\Diamond$ and $\Box$, such that for any formula φ the formulae $\Diamond\varphi$ (possibly φ) and $\Box\varphi$ (necessarily φ) can be built. The meaning of this formulae is given by transition systems, i. e. directed graphs labelled with propositional constants. At some distinguished node v the formula $\Diamond\varphi$ is true, if v has some successor where φ holds whereas $\Box\varphi$ is true at v, if φ holds at all successors of v.

Every formula $\psi \in \mathsf{ML}$ can be translated into a first-order formula $\psi^*(x)$ with one free variable, by translating propositional constants P to atoms Px, passing over the Boolean operators and translating the modal operators as follows, with E being the edge relation of the transition system.

$$\begin{aligned}\Diamond\psi &\rightsquigarrow \exists y(Exy \wedge \psi^*(y)),\\ \Box\psi &\rightsquigarrow \forall y(Exy \rightarrow \psi^*(y)).\end{aligned}$$

One can observe that in the image of this translation quantifiers appear only relativized by atomic formulae, i. e. in the form

$$\begin{aligned}&\exists \boldsymbol{y}(\alpha(\boldsymbol{x},\boldsymbol{y}) \wedge \psi(\boldsymbol{x},\boldsymbol{y})) \text{ and}\\ &\forall \boldsymbol{y}(\alpha(\boldsymbol{x},\boldsymbol{y}) \rightarrow \psi(\boldsymbol{x},\boldsymbol{y})),\end{aligned}$$

where α is an atom containing all free variables of ψ. The guarded fragment GF of first-order logic consists of all first-order formulae where quantification is restricted in this way.

For this extension of ML very favorable model-theoretic properties have been proven [AvBN98, Grä]:

1. The satisfiability problem for GF is decidable.
2. GF has the finite model property, meaning that any satisfiable formula has a finite model.
3. GF has (a generalized variant of) the tree model property.
4. The notion of equivalence between formulae in GF can be characterized by a generalization of bisimulation.

The enumerated properties already present the guarded fragment as a promising candidate for application. Especially the tree model property can be seen as a indication for algorithmic tractability, as it makes the logic amenable to automata-theoretic methods. Further, bisimulation invariance, which in essence means that a logic does not distinguish between models showing the same behavior, grants GF natural prominence in the description of behavioral systems.

Taking GF as a basis, fragments of increased expressive power have been defined. The loosely guarded fragment, LGF [AvBN98], and the clique guarded fragment, CGF [Grä99], loosen the constraint of guarding variables by atoms allowing more relaxed quantification without loosing the good model-theoretic properties of GF.

In [Koz83] Kozen defines the $\mu - calculus$, or $\mathsf{L}\mu$, extending ML by least and greatest fixed point operators. With the ability to express recursion, which is a fundamental device in computation, this logic became extremely useful for the description and verification of complex systems.

The extension of GF with least and greatest fixed-point was introduced by Grädel and Walukiewicz in [GW99]. The obtained logic $\mu\mathsf{GF}$ was shown to preserve most of the good model-theoretic properties of GF. Moreover, the extensions μLGF and μCGF of the more expressive first-order fragments LGF and CGF defined in [GW99] and [Grä99], respectively, have been proven to be model-theoretically robust under these extensions.

All this results motivate further research in the framework of guarded logics.

Outline. In this work we will study the model checking problem for fixed point extensions of guarded logics. A model checking problem is essentially given by a structure $\mathfrak{A}$ and sentence ψ and consists in verifying whether ψ holds in the structure $\mathfrak{A}$ or not.

In Chapter 2 we will formally define the guarded fragments GF, LGF, and CGF of first-order logics and their fixed point extensions μGF, μLGF, and μCGF.

Our approach to the model checking problem will be via the framework of parity games. To this framework Chapter 3 is dedicated. After introducing the game terminology we present an algorithm of Jurdziński [Jur00] for the problem of deciding the winner of a parity game and the result of Emerson, Jutla, and Sistla [EJS93] stating that this problem belongs to NP $\cap$ Co-NP.

In Chapter 4 we present a reduction of the model checking problem for guarded fixed point logics to parity games, generalizing the model checking games for Lμ constructed by Herwig [Her89] and Stirling [Sti95]. In our approach the connection between model checking and games will be mediated by the notions of signatures for fixed point formulae defined by Grädel and Walukiewicz [GW99] and progress measures for parity games introduced by Jurdziński [Jur00], which turn out to be closely related.

Departing from this reduction we will investigate the complexity of the model checking problems for guarded fixed point logics in the general case. As Vardi points out in [Var95] the *width* of a sentence, i. e. the number of free variables in subformulae of the sentence may have a crucial impact to the complexity of model checking. For this reason we will consider model checking for the bounded-variable fragments separately. Since the number of free variables of μGF-formulae is trivially bound by the vocabulary this separation does apply only to μLGF and μCGF.

In [Dzi96] Dziembowski proves that the model checking problems for extensions of first-order logic with fixed point operators are PSPACE-complete, even when the number of first and second order-variables is bounded. On the other hand the model checking problem for Lμ, which can easily be embedded into μGF, belongs to NP $\cap$ Co-NP as shown by Emerson and Jutla [EJ91]. The question arises, how the complexity of model checking for μGF and the bounded variable fragments of μCGF respectively μCGF is related to this bounds.

By reduction to parity games we will show that model checking for μGF and the bounded-variable fragments of μLGF and μCGF belong to NP $\cap$ Co-NP. Hence, model checking for these logics is structurally not harder than for Lμ, despite their higher expressiveness. Via the algorithm for solving parity games presented in Chapter 2 we obtain deterministic bounds for model checking μGF and the restricted fragments of μLGF and μCGF.

Finally, considering unrestricted μLGF and μCGF we will prove that model checking for these logics is EXPTIME-complete.

Once the model checking complexity is established for the general case, we are interested in delimitating fragments of μGF where model checking can be performed tractably, i. e. in linear time with respect to the size of the structure and polynomially in the length of the formula. To this end, in Chapter 5 we identify classes of simple games, which can be tractably solved. Transferring the restrictions imposed on these games back to μGF we obtain tractable fragments of μGF generalizing the tractable fragments known for Lμ.

Acknowledgement. I would like to thank Erich Grädel for his insight and guidance while I wrote this thesis, and to Marcin Jurdziński, Stephan Kreutzer, and Jens Vöge for many inspiring discussions on the subject.

Chapter 2

Guarded Logics

2.1 Preliminaries

Throughout this paper we consider only finite structures over finite relational vocabularies. We will use fraktur letters for structures $\mathfrak{A}, \mathfrak{B}, \ldots$ and the corresponding roman letters for their universe $A, B, \ldots$

By first-order logic (FO) we mean relational first-order logic with equality, i. e. we admit relation symbols of any arity and constant symbols but no function symbols of positive arity.

Given a formula ψ, we denote by $|\psi|$ the length of ψ viewed as a string. We say a formula φ is a *subformula* of ψ if φ is represented by a substring of ψ. By $\mathrm{sf}(\psi)$ we denote the set of all subformulae of ψ

We use boldface letters $\boldsymbol{t}$ to denote tuples of variables and constants $t_1, \ldots, t_n$. In certain contexts we may use $\boldsymbol{t}$ to designate the set $\{t_1, \ldots, t_n\}$. We write $\varphi(\boldsymbol{x})$ to express that all the free first-order variables of φ are among $\boldsymbol{x}$. Herewith we do not exclude the presence of free second-order variables in φ.

For a first or second-order formula we denote by $\mathrm{free}(\varphi)$ the set of free variables of φ defined in the standard way. If φ is second-order, we write $\mathrm{free}_1(\varphi)$ to mean free first-order variables and $\mathrm{free}_2(\varphi)$ to mean free second-order variables.

Given a structure $\mathfrak{A}$ any formula $\varphi(y_1, \ldots y_k)$ defines a set

$$\varphi^{\mathfrak{A}}(\_\_) := \{\boldsymbol{b} \in A^k \ : \ \mathfrak{A} \models \psi(\boldsymbol{b})\}.$$

For $\varphi(\boldsymbol{x}, \boldsymbol{y})$ and an assignment χ of the variables $\boldsymbol{x}$ we define

$$\varphi^{\mathfrak{A},\chi}(\boldsymbol{x}, \_\_) := \{\boldsymbol{b} \in A^k \ : \ \mathfrak{A}, \chi \models \psi(\boldsymbol{x}, \boldsymbol{b})\}.$$

2.2 Guarded fragments of first-order logic

We consider subsets of first-order formulae where quantification is restricted by predicates called *guards*.

Definition 2.1. (Guardedness) Given a tuple of variables $\boldsymbol{y}$ and a first-order formula α we say

(1) $\boldsymbol{y}$ is *strictly guarded* by α if α is a positive atomic formula and $\boldsymbol{y} \subseteq \text{free}(\alpha)$.

(2) $\boldsymbol{y}$ is *loosely guarded* by α if α is a conjunction of atoms $\alpha_1 \wedge \cdots \wedge \alpha_m$ such that $\boldsymbol{y} \subseteq \text{free}(\alpha)$ and for any two variables $z \in \boldsymbol{y}$ and $z' \in \text{free}(\alpha)$ there is at least one atom α_j that contains both z and z'.

Definition 2.2. The *(strictly) guarded fragment* GF of first-order logic is defined by the following calculus.

(A) $\quad \dfrac{\ }{\varphi}$, $\quad$ where φ is a relational atomic formula,

$(\neg)$ $\quad \dfrac{\varphi}{\neg\varphi}$, $\quad (\wedge)$ $\quad \dfrac{\varphi, \eta}{\varphi \wedge \eta}$, $(\vee)$ $\quad \dfrac{\varphi, \eta}{\varphi \vee \eta}$, $(\rightarrow)$ $\quad \dfrac{\varphi, \eta}{\varphi \rightarrow \eta}$,

$(\exists)$ $\quad \dfrac{\varphi}{\exists \boldsymbol{y}(\alpha \wedge \varphi)}$, $\quad$ where $\boldsymbol{y}$ is a tuple of variables strictly guarded by α and $\text{free}(\varphi) \subseteq \text{free}(\alpha)$,

$(\forall)$ $\quad \dfrac{\varphi}{\forall \boldsymbol{y}(\alpha \rightarrow \varphi)}$, $\quad$ where $\boldsymbol{y}$, α and φ are as above.

The *loosely guarded fragment* LGF is defined in the same way as GF, except that the quantifier rules $(\exists)$ and $(\forall)$ require $\boldsymbol{y}$ to be loosely guarded by α.

In [Grä99] Grädel introduces another, more liberal, variant of guarded quantification, leading to a fragment that is strictly more expressive than LGF showing the same nice decidability and model-theoretic properties. It is called the *clique guarded fragment*, CGF.

Above, we defined the notion of guardedness via syntactic restriction. To introduce clique guarded quantification we will take a semantic viewpoint. Considering a fixed vocabulary τ and a τ-structure $\mathfrak{A}$ with universe A, let us look to the meaning of guardedness.

Definition 2.3. Given a guard $\alpha \in \mathsf{FO}[\tau]$, we say a tuple $\boldsymbol{b} \subseteq A$ is (strictly or loosely) guarded by α if there is a tuple of variables $\boldsymbol{y}$ (strictly or loosely) guarded by α such that we have $\mathfrak{A} \models \alpha(\boldsymbol{a}, \boldsymbol{b})$ for some tuple $\boldsymbol{a} \subseteq A$.

More general, we say a tuple $\boldsymbol{b}$ is (strictly or loosely) guarded in $\mathfrak{A}$ if it is guarded by some α. When we view the tuple $\boldsymbol{b}$ as a set we speak about a *guarded set.*

Taking the elements of the universe as nodes and linking two nodes if they form a strictly guarded pair we obtain what in model theory is called the locality or Gaifman graph of the structure.

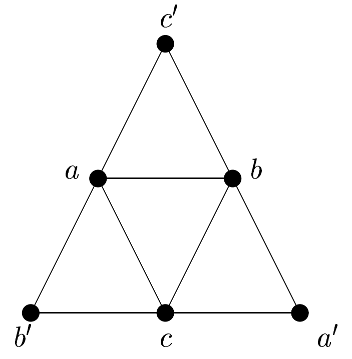
Definition 2.4. The *Gaifman graph* of a the structure $\mathfrak{A}$ is the graph $G(\mathfrak{A}) = (A, E^G)$ with edge relation

$$E^G := \{(b, b') \ : \ b \neq b' \text{ and } (b, b') \text{ is strictly guarded in } \mathfrak{A}\}.$$

It follows immediatly that any strictly guarded set in $\mathfrak{A}$ induces a clique in $G(\mathfrak{A})$. Moreover, we can easily see that this holds for loosely guarded sets as well:

Lemma 2.5. *Any guarded set in* $\mathfrak{A}$ *induces a clique in* $G(\mathfrak{A})$.

Observe that the converse does not hold: consider a structure $\mathfrak{B} = (B, R^{\mathfrak{B}})$ with $B = \{a, b, c, a', b', c'\}$ and $R^{\mathfrak{B}} = \{(a, b, c'), (a, b', c), (a', b, c)\}$. The set $\{a, b, c\}$ is neither strictly nor loosely guarded in $\mathfrak{B}$, but it induces a clique in $G(\mathfrak{B})$:



The idea of guardedness can be extended to cover *all* the cliques of the Gaifman graph. Obviously, for any fixed vocabulary τ the requirement that some tuple $\boldsymbol{b}$ induces a clique in the Gaifman graph of some τ-structure $\mathfrak{A}$ can be formulated in first-order logic.

Lemma 2.6. *For each finite vocabulary* τ *and any natural number* k *there is a positive existential formula* α *such that in all* τ*-structures* $\mathfrak{A}$ *we have*

$$\mathfrak{A} \models \alpha(\boldsymbol{b}) \text{ iff } \boldsymbol{b} \text{ induces a clique in } G(\mathfrak{A})$$

for any k*-tuple* $\boldsymbol{b} \subseteq A$.

We shall give a name to formulae of this type.

Definition 2.7. (Clique formula) Given a vocabulary τ, we say a first-order formula α over τ is a *clique-formula* if for every τ-structure $\mathfrak{A}$ the set $\alpha^{\mathfrak{A}}(\_\_)$ induces a clique in $G(\mathfrak{A})$.

Finally, let us insert the clique formulae into the category of guards.

Definition 1.1. (Guardedness, continued) Given a tuple of variables $\boldsymbol{y}$ and a first-order formula α we say

(3) $\boldsymbol{y}$ is *clique guarded* by α if α is a clique formula and $\boldsymbol{y} \subseteq \text{free}(\alpha)$.

The *clique guarded fragment* CGF is defined in the same way as GF, except that the quantifier rules ($\exists$) and ($\forall$) require $\boldsymbol{y}$ to be clique guarded by α.

Notation. We will use the notation $(\exists \boldsymbol{y}.\alpha)$ and $(\forall \boldsymbol{y}.\alpha)$ for restricted quantifiers, i. e. we write guarded formulae in the form $(\exists \boldsymbol{y}.\alpha)\varphi(\boldsymbol{x}, \boldsymbol{y})$ and $(\forall \boldsymbol{y}.\alpha)\varphi(\boldsymbol{x}, \boldsymbol{y})$. When this notation is used, it is always understood that α is a proper guard as specified by Definition 2.1

2.3 Guarded fixed point logics

Definition 2.8. The *guarded fixed point logics* $\mu\mathsf{GF}$, $\mu\mathsf{LGF}$ and $\mu\mathsf{CGF}$ are obtained by adding to GF, LGF, and CGF respectively the following rules for constructing fixed point formulae:

(μ) $\dfrac{[\mathrm{LFP}\ W\boldsymbol{x}.\varphi](\boldsymbol{x})}{\varphi}$, where W is a k-ary relation symbol occuring only positively in φ and $\boldsymbol{x}$ is a k-tuple of distinct first-order variables containing all of $\mathrm{free}_1(\varphi)$.

(ν) $\dfrac{[\mathrm{GFP}\ W\boldsymbol{x}.\varphi](\boldsymbol{x})}{\varphi}$, where W, $\boldsymbol{x}$ and φ are as above.

The parts in square brackets, i. e. $[\mathrm{LFP}\ W\boldsymbol{x}.\varphi]$ and $[\mathrm{LFP}\ W\boldsymbol{x}.\varphi]$, are called *fixed point predicates.*

For analytic purposes we additionally define the *unguarded fixed point logic* $\mu\mathsf{F}$ to be the extension of full first-order logic by the fixed point rules (μ) and (ν). Please note that $\mu\mathsf{F}$ is still a strict subset of the least fixed point logic LFP (see [EF95]), as the set of free first-order variables of φ is required to be contained in $\boldsymbol{x}$. In other words, $\mu\mathsf{F}$ does not admit parameters in fixed points.

We will use the notation $[\mathrm{FP}\ W\boldsymbol{x}.\psi]$ in statements that apply equally to the greatest and the least fixed point. The set of free variables of fixed point formulae is defined by adding the following clause to the standard definition:

$$\mathrm{free}\big([\mathrm{FP}\ W\boldsymbol{x}.\varphi](\boldsymbol{x})\big) = \mathrm{free}(\varphi) \setminus \{W\}.$$

A fixed point formula is called *independent* if it has no free second-order variables.

The semantics of fixed point formulae is defined as follows: Given a structure $\mathfrak{A}$, a formula $\varphi(W,\boldsymbol{x})$ with $W \in \mathrm{free}_2(\varphi)$, and an assignment χ for all the free second-order variables in φ except W, the formula $\varphi(W,\boldsymbol{x})$ defines an operator on k-ary relations $U \subseteq A^k$, namely

$$\varphi^{\mathfrak{A},\chi} : U \mapsto \varphi^{\mathfrak{A},\chi}(U, \_\_).$$

Since W occurs only positively in ψ, this operator is syntactically monotone (i. e. $U \subseteq U'$ implies $\varphi^{\mathfrak{A},\chi}(U) \subseteq \varphi^{\mathfrak{A},\chi}(U')$) and therefore has a least fixed point $\mathrm{LFP}(\varphi^{\mathfrak{A},\chi})$ and a greatest fixed point $\mathrm{GFP}(\varphi^{\mathfrak{A},\chi})$. The meaning of least fixed point formulae is now defined by

$$\mathfrak{A},\chi \models [\mathrm{LFP}\ W\boldsymbol{x}.\varphi](\boldsymbol{a}) \quad \text{iff} \quad \boldsymbol{a} \in \mathrm{LFP}(\varphi^{\mathfrak{A},\chi}).$$

and similarly for the greatest fixed point:

$$\mathfrak{A},\chi \models [\mathrm{GFP}\ W\boldsymbol{x}.\varphi](\boldsymbol{a}) \quad \text{iff} \quad \boldsymbol{a} \in \mathrm{GFP}(\varphi^{\mathfrak{A},\chi}).$$

Normal form

We say a fixed point formula φ is in *negation normal form* if all the negations apply to atoms only. Note that every formula can be put in negation normal form by pushing negation inwards using De Morgan's Laws and the dualities:

$$\begin{aligned}\neg[\mathrm{LFP}\ W\boldsymbol{x}.\varphi(W,\boldsymbol{x})] &\equiv [\mathrm{GFP}\ W\boldsymbol{x}.\neg\varphi(\neg W,\boldsymbol{x})],\\ \neg[\mathrm{GFP}\ W\boldsymbol{x}.\varphi(W,\boldsymbol{x})] &\equiv [\mathrm{LFP}\ W\boldsymbol{x}.\neg\varphi(\neg W,\boldsymbol{x})].\end{aligned}$$

This normalization can at most double the length of the formula. Further we say φ is *well-named* if every fixed point variable is bounded only once in φ and the free second-order variables are distinct from fixed point variables. Obviously, every formula can be transformed into a well-named one in linear time.

2.4 Model theoretic properties

In the following we will give a short overview of some model theoretic properties of guarded logics. Thereby we will refer to the *width* of a guarded sentence ψ, which denotes the maximal number of free variables in subformulae of ψ.

Please note that for a fixed vocabulary τ, the width of every sentence ψ in GF and CGF respectively $\mu\mathsf{GF}$ and $\mu\mathsf{CGF}$ is bound by the maximal arity of the relation symbols in τ. For loosely or clique guarded sentences however, the width may exceed the arity of the relation symbols.

1. **Decidability.** The satisfiability problems for guarded logics are decidable. For all the considered logics the satisfiability problem is 2EXPTIME-complete in the general case. Moreover, for sentences of bounded width the satisfiability problem is EXPTIME-complete. This has been proved in [Grä] for GF and LGF and in [Grä99] for CGF and $\mu\mathsf{CGF}$ and in [GW99] for $\mu\mathsf{GF}$ and $\mu\mathsf{LGF}$.

2. **Finite model property of GF.** Every satisfiable sentence in GF has a finite model ([Grä, HT]).

3. **Tree model property.** If a guarded sentence ψ of width k is satisfiable, then it has a countable model of tree width at most $k-1$ and branching degree at most $O(|\psi|^k)$. The generalized tree model property was proven in [Grä99].

4. **Bisimulation invariance.** The notion of equivalence under striclty guarded formulae can be characterized by a generalization of bisimulation. For GF this was shown in [AvBN98]. The result for $\mu\mathsf{GF}$ appears in [GHO00].

Chapter 3

Parity Games

3.1 Definition

Definition 3.1. A *parity game* is a tuple $\mathcal{G} = (V, E, \Omega)$ where

$V = V_\diamond \dot\cup V_\Box$ is a finite set of *positions*,
$E \subseteq V \times V$ is an irreflexive edge relation, i. e. $(v, v) \notin E$ for all $v \in V$, and
$\Omega : V \to \mathbb{N}$ is a function labelling every position with a nonzero *priority*.

The *index* of $\mathcal{G}$ is the highest priority occurring in the game. We denote by Ω_i the partition of vertices of priority i in $\mathcal{G}$. Given a position $v \in V$ we write vE for the set of successors $\{w : (v, w) \in E\}$.

The game is played between two players, $\diamond$ and $\Box$ (sometimes called Even and Odd). A play consists of moving a token from one position to another along an edge of $\mathcal{G}$. If the token is on a vertex from $V_\diamond$ then player $\diamond$ moves, otherwise player $\Box$.

The result of a play from an initial position v_0 is a finite or infinite E-path $\pi = v_0, v_1, \ldots$. This path can be finite only because one of the players gets stuck. In this case he looses. If the play is infinite, we look at the sequence of priorities $\Omega(v_0), \Omega(v_1), \ldots$. If the least priority that appears infinitely often in this sequence is even, player $\diamond$ wins, otherwise she looses. [1]

A *strategy* σ for player $\diamond$ in $\mathcal{G}$ is a partial function defined on E-paths that end with a position from $V_\diamond$. It assigns to any path $\pi \in V^* \times V_\diamond$ in the domain of σ a prolongation $\sigma(\pi) \in V$ along an E-edge.

We say the strategy σ is *winning* from a position v_0 if player $\diamond$ wins every play from v_0 by moving the token according to σ. If a player has a winning strategy in $\mathcal{G}$ from v_0 we will sometimes say he wins the game from v_0.

A strategy is called *memoryless* if $\sigma(\pi) = \sigma(\pi')$ whenever π and π' end in the same position.

[1] This acceptance condition is also called Mostowski or chain Rabin condition.

Remark 3.2. We can alternatively view a memoryless strategy σ for player $\Diamond$ in $\mathcal{G}$ as a game $\mathcal{G}_\Diamond = (V, E_\Diamond, \Omega)$ with

$$E_\Diamond = \{(v, \sigma(v)) \ : \ v \in \mathrm{dom}(\sigma)\} \cup (E \cap V_\Box \times V).$$

In this view the strategy is winning from position $v_0 \in V$ iff player $\Diamond$ wins every play from v_0 in $\mathcal{G}_\Diamond$.

Parity games have been introduced in a different phrasing by A. W. Mostowski [Mos84, Mos91] and independently by Emerson and Jutla [EJ91]. The following essential results are due to them.

Theorem 3.3. *[Determinacy] From any position in $\mathcal{G}$ either player $\Diamond$ or player $\Box$ has a winning strategy.*

Theorem 3.4. *[Memoryless strategy] If a player has a winning strategy in $\mathcal{G}$ then he has a memoryless winning strategy.*

3.2 Flowers and progress measures

The remainder of this chapter is dedicated to the problem of deciding the winner of a parity game starting at a given position. In this section we will introduce some notions to set the foundations for the game solving algorithm of Jurdziński [Jur00], which we will present in the next section.

When we view a game $\mathcal{G} = (V, E, \Omega)$ as a labeled directed graph, we refer to it as the *game graph.* Given a subset of positions $Q \subseteq V$, we will denote the subgraph induced by Q with $\mathcal{G}{\restriction_Q}$:

$$\mathcal{G}{\restriction_Q} := (Q, E \cap Q \times Q, \Omega{\restriction_Q}).$$

Please note that $\mathcal{G}{\restriction_Q}$ is a game itself; we will call it the *subgame induced by Q*. For Q being the set of all nodes reachable from some $v \in V$ we call $\mathcal{G}{\restriction_Q}$ the *subgame rooted at v* and denote it by $\mathcal{G}{\restriction_v}$.

Flower decomposition. It is easy to see that any infinite play in $\mathcal{G}$ will eventually stay within some strongly connected component of the game graph, i. e. a subset of V, such that for any two positions v, w in the subset, there is a path from v to w and from w to v. If $Q \subseteq V$ is a maximal subset with this property, we call the subgraph $\mathcal{G}{\restriction_Q}$ a *flower* of $\mathcal{G}$. Since flowers are disjoint by maximality, they form a partitioning of the playground. Moreover, as there are no cycles visiting different flowers, the edge relation induces a topological ordering on the set of flowers. We call the set of flowers equipped with a topological ordering a *flower decomposition* of $\mathcal{G}$.

A *loop* in $\mathcal{G}$ is a simple cycle, i. e. a path $v_0, \ldots, v_l$ such that $v_0 = v_l$ and $v_i \neq v_0$ for all $0 < i < l$. If the least priority seen on the path is k we speak about a *k-loop.* When k is even (odd) we also call it an *even loop* (*odd loop*).

Obviously loops live in flowers. If all loops in a flower are even, we say the flower is even. Dually, if all loops in a flower are odd, we say the flower is odd. A flower consisting of a single node, hence containing no loop, is *trivial.* In this terminology, a (memoryless) strategy $\mathcal{G}_\Diamond$ in the sense of Remark 3.2 is a game where all non-trivial flowers are even.

Progress measure. Our algorithmic approach essentially relies on the concept of progress measures introduced by Klarlund in [Kla90]. In our setting, a progress measure is a mapping on the positions of a game, which tells how far from a win a position is. From the viewpoint of player $\Diamond$, the progress measure assigned to a $\Box$-node with no outgoing edges should be low (she already won); on an odd cycle, where all her exits lead again to odd cycles it might be higher, and it will be infinite on odd cycles with no exit at all. The duty of a progress measure is to record the change of winning potential implied by a possible move. Thus, the strength of this device rests on the fact that it requires local consistency to provide a certificate for the existence of a winning strategy, which is a global property.

When used with tuples of natural numbers the comparison symbol $<$ denotes the lexicographical ordering with least significant position last. With a subscript $i \in \mathbb{N}$, the notations $\boldsymbol{a} <_i \boldsymbol{b}$ and $\boldsymbol{a} =_i \boldsymbol{b}$ refer to the ordering of the arguments truncated to their first i components. For technical convenience, we extend this ordering by a top element, $\top$, that satisfies $\top >_i \boldsymbol{a}$ for all tuples $\boldsymbol{a}$ of natural numbers and $\top =_i \top$ for every $i \in \mathbb{N}$.

Definition 3.5. Let $\mathcal{G} = (V, E, \Omega)$ be a parity game of index d. A *progress measure* for player $\Diamond$ in $\mathcal{G}$ is a function $p : V \to \mathbb{N}^d \cup \{\top\}$ such that for any $v \in V$ with $p(v) < \top$

- if $v \in V_\Diamond$ then there is some $w \in vE$ such that $p(w) \leq_{\Omega(v)} p(v)$ and
- if $v \in V_\Box$ then for all $w \in vE$ we have $p(w) \leq_{\Omega(v)} p(v)$.

If in the above conditions $\Omega(v)$ is odd the inequalities are required to be strict.

There is a direct correspondence between progress measures and winning strategies.

Proposition 3.6. *If for some $v_0 \in V$ player $\Diamond$ has a progress measure p with $p(v_0) < \top$ then she wins the game from v_0.*

Proof. Departing from p we set up a strategy $\mathcal{G}_p = (V, E_p, \Omega)$ as follows:

- for $v \in V_\Diamond$, if $p(v) < \top$ then choose $w \in vE$ with $p(w) \leq_{\Omega(v)} p(v)$ and set $vE_p := \{w\}$ else set $vE_p := \emptyset$,
- for $v \in V_\Box$ set $vE_p := vE$.

Claim: The strategy $\mathcal{G}_p$ is winning for player $\Diamond$ from position v_0.

By definition of E_p, if a position $v \in V$ has finite measure (i. e. $p(v) < \top$) then all successors $w \in vE_p$ have finite measure as well. Since $p(v_0) < \top$, any play in $\mathcal{G}_p$ starting at v_0 will proceed only through positions with finite measure. Further, any position in $V_\Diamond$ with finite measure has at least one successor in E_p. Therefore player $\Diamond$ can never get stuck and any finite play is a win.

Now let $(v_0, v_1, \dots)$ be an infinite play in $\mathcal{G}_\Diamond$. Towards a contradiction assume that player $\Box$ wins, i. e. the least priority i seen infinitely often in the sequence $\Omega(v_0), \Omega(v_1), \dots$ is odd. By the definition of $\mathcal{G}_p$, the values of p never increase along edges. Moreover, they decrease on some position $\leq i$ whenever a vertex v with $\Omega(v) = i$ is met. Since the lexicographical ordering of tuples in $\mathbb{N}^d$ is wellfounded this can not happen infinitely often. Hence the least priority occurring infinitely often is even and player $\Diamond$ wins. $\Box$

Proposition 3.7. *If player $\Diamond$ wins from position v_0 in $\mathcal{G}$ then she also has a progress measure p with $p(v) < \top$.*

Proof. Let $\mathcal{G}_\Diamond = (V, E_\Diamond, \Omega)$ be a winning strategy for player $\Diamond$ from v_0. We will set up a progress measure by applying timestamps to the positions of V recursively. The stamp will be maintained in a global variable $\boldsymbol{p} \in \mathbb{N}^d$ which is $\mathbf{0}$ in the start.

The following procedure `Stamp`$(\mathcal{H})$ stamps the vertices of a subgame $\mathcal{H} \subseteq \mathcal{G}_\Diamond$.

- If $\mathcal{H}$ has no vertices, nothing has to be done.
- Otherwise, let $\mathcal{Q} = (Q_1, \dots, Q_l)$ be a flower decomposition of $\mathcal{H}$. The flowers Q_k, $k \in \{1, \dots, l\}$, are processed in reverse topological order, from $k = l$ down to 1, as follows.

  Erode. If Q_k is a trivial flower, i. e. consisting of a single node v, look at the priority $i = \Omega(v)$. If i is odd then increase the i-th position of the timestamp $\boldsymbol{p}$, else leave $\boldsymbol{p}$ unchanged. Apply the timestamp $\boldsymbol{p}$ to v (set $p(v) = \boldsymbol{p}$) and proceed.

  Attack. If Q_k is not a trivial flower, let i be the least priority seen in Q_k. Apply the stamp $\boldsymbol{p}$ to every position in $Q_k \cap \Omega_i$. Then recursively call `Stamp`$(\mathcal{H}\restriction_{Q_k \setminus \Omega_i})$ and proceed.

During the execution of the above algorithm on the subgame $\mathcal{G}_\Diamond\restriction_{v_0}$ any node v reachable from v_0 receives a timestamp in $p(v)$. For any $v \in V$ which is not reachable from v_0 set $p(v) := \top$.

In the following discussion, we will always refer to flowers and topological orderings with regard to the subgame $\mathcal{H}$ under recursion. About a loop or a flower we say it is eroded (attacked) if the first stamp on its nodes is applied by erosion (attack).

To see that p is indeed a progress measure let us first establish that the algorithm has the following policy:

1. No loop is eroded without being previously attacked.
2. When a flower is being attacked, none of its nodes have unstamped successors.
3. If a flower Q with minimal priority i is being attacked at timestamp $\boldsymbol{p}$ then $p(v) =_i \boldsymbol{p}$ for every $v \in Q$.
4. A node v is eroded only after all its successors have been processed.

Proof. Principle 1: As games are self-loop-free any loop is contained in some non-trivial flower. So, on the top level of the recursion it will not be eroded. After an attack, any loop is either being attacked itself or left untouched. In the latter case, if the flower protecting it was attacked, the loop enters the next recursion level unbroken.

Principle 2: A flower Q is being attacked only when all flowers topologically succeeding it have previously been processed. Therefore, any unstamped successor of its nodes must be contained in Q as well.

Principle 3: According to principle 1, the first nodes of a flower are always stamped by attack. This way the nodes of minimal priority receive timestamp $\boldsymbol{p}$. Every other node $v \in Q$ will be processed later and receives a timestamp lexicographically greater or equal to $\boldsymbol{p}$. Assuming there is some $w \in Q$ with $p(v) >_i \boldsymbol{p}$ leads to a contradiction, since positions below i are incremented only when nodes of odd priority less than i are met but i is minimal.

Principle 4: Let v be the first node eroded while its successor w was not yet processed. This means, that w does not topologically succeed v (in the game under recursion). Hence, there is a loop containing v and w which was not yet attacked, contradicting principle 1. □

We are ready now to check that p is consistent.

As the algorithm visits every node of the subgame it is processing, all positions reachable from v_0 are stamped, especially v_0 itself. Therefore $p(v_0) < \top$. Since $\mathcal{G}_\diamond$ is a winning strategy for $\diamond$, all positions in $V_\diamond$ reachable from v_0 have an outgoing edge.

For any node v reachable from v_0 and every successor $w \in vE_\diamond$ we have $p(v) \geq_{\Omega(v)} w$. For v being eroded this follows directly by Principle 4. For v being stamped by attack, let Q be the attacked flower. If $w \notin Q$ then the property holds by Principle 1, otherwise, it holds by Principle 3.

Additionally, if $\Omega(v) = i$ is odd, the algorithm increments the i-th position of the stamp before applying it. Note that no position of odd priority is ever attacked. So principle 4 applies and we get $p(v) >_i p(w)$ for all successors $w \in vE_\diamond$.

□

Tracking the progress of the timestamp $\boldsymbol{p}$ in the execution of the above algorithm, please note that $\boldsymbol{p}$ changes only when a node of odd priority i is visited whereby its i-th position is increased by 1. As $\boldsymbol{p}$ is $\mathbf{0}$ at the start and every node of V is visited at most once this implies that on any even position the value of $\boldsymbol{p}$ always stays 0 while on any odd position i it never exceeds $|\Omega_i|$.

For any natural n we denote by $[n]$ the set $\{0,\dots,n\}$. In extension to this notation let $[\Omega] \subseteq \mathbb{N}^d$ be defined as

$$[\Omega] := [|\Omega_1|] \times [0] \times [|\Omega_2|] \times [0] \times \cdots \times [|\Omega_{d-1}|] \times [0]$$

if d is even and $[\Omega] := [|\Omega_1|] \times [0] \times \cdots \times [0] \times [|\Omega_d|]$ if d is odd.

Obviously all timestamps $\boldsymbol{p}$ can be retained in the set $[\Omega]$. This leads to the following criterion for the existence of a winning strategy.

Theorem 3.8. *Player $\Diamond$ has a winning strategy from some node v_0 in $\mathcal{G}$ iff she has a progress measure $p : V \to [\Omega] \cup \{\top\}$ with $p(v_0) < \top$.*

3.3 A blueprint for winning

Departing from the above criterion we build an algorithm for establishing the winner of a parity game.

Let $\mathrm{mip}(i, \boldsymbol{q})$ with $1 \le i \le d$ and $\boldsymbol{q} \in [\Omega] \cup \{\top\}$ be a shorthand for the value of minimal progress over $\boldsymbol{q}$ up to position i:

$$\mathrm{mip}(i,\boldsymbol{q}) := \begin{cases} \top & \text{if } \boldsymbol{q} = \top, \\ \min\{\boldsymbol{p} \in [\Omega] \ : \ \boldsymbol{q} \le_i \boldsymbol{p}\} & \text{if } \boldsymbol{q} < \top \text{ and } i \text{ is even }, \\ \min\{\boldsymbol{p} \in [\Omega] \cup \{\top\} \ : \ \boldsymbol{q} <_i \boldsymbol{p}\} & \text{if } \boldsymbol{q} < \top \text{ and } i \text{ is odd.} \end{cases}$$

Consider the set of functions $P := \big\{p \ : \ p : V \to [\Omega] \cup \{\top\}\big\}$ containing all candidates for progress measures. Given two functions $q, p \in P$ we define the relation $p \sqsubseteq q$ to hold if $q(v) \le p(v)$ for all $v \in V$. As $\sqsubseteq$ partially orders the elements of P, the structure $(P, \sqsubseteq)$ is a complete lattice.

On this lattice we define for every $v \in V$ an operator Lift_v.

$$\mathrm{Lift}_v(p)(u) = \begin{cases} p(u) & \text{if } u \ne v, \\ \min_{w \in vE} \mathrm{mip}\big(\Omega(v), p(w)\big) & \text{if } u = v \in V_\Diamond, \\ \max_{w \in vE} \mathrm{mip}\big(\Omega(v), p(w)\big) & \text{if } u = v \in V_\Box. \end{cases}$$

for all $u \in V$ and for every $p \in P$.

Since $\mathrm{Lift}_v(p)$ changes the value of its argument p only at node v we can conclude with the definition of mip in mind:

Lemma 3.9. *For any $v \in V$ the operator Lift_v is monotone.*

Further, if some $p \in P$ is not a progress measure then there is some node at which p can be (strictly) lifted, i. e. there is some $v \in V$ such that $p \sqsubset \mathrm{Lift}_v(p)$. So any simultaneous fixed point of the $|V|$-ary operator $\mathrm{Lift} := (\mathrm{Lift}_v)_{v \in V}$ is a progress measure. In the complete lattice $(P, \sqsubseteq)$ this operator, being (componentwise) monotonous, has a fixed point by the Knaster-Tarski Theorem. Moreover, the least of its fixed points can be iterated starting from the bottom element.

There may be progress measures in $p \in P$ which are not fixed points of Lift. In this case we have $\mathrm{Lift}(p) \sqsubset p$, by the definition of progress measures. The iteration of Lift then leads to a fixed point, i. e. progress measure, strictly less than p. So the least progress measure in P is always the least fixed point of Lift.

This gives rise to the following algorithm for computing the least progress measure for a given parity game:

> **procedure** `MinProgressMeasure`($\mathcal{G}$)
> $p := (v \mapsto \mathbf{0}$ for all $v \in V)$
>   **while** $p \sqsubset \mathrm{Lift}_v(p)$ for some $v \in V$ **do**
> $p := \mathrm{Lift}_v(p)$ **od**
>   **output** p
> **end.**

What do we get by computing the least progress measure p of a game?

By Theorem 3.8 player $\Diamond$ wins from some node $v_0 \in V$ iff there is a progress measure $q \in P$ with $q(v_0) \leq \top$. In this case, by minimality of p, we have $p(v_0) \leq q(v_0) < \top$. In other words, the set $W_\Diamond := p^{-1}[\Omega]$ consists of those nodes in $\mathcal{G}$ from which player $\Diamond$ wins. We will call this subset of V the *winning set* of player $\Diamond$. By determinacy, the complement of this set, $W_\Box := V \setminus W_\Diamond$, is the winning set of player $\Box$.

Moreover, departing from p, we can construct a strategy $E_\Diamond$ for $\Diamond$ as in the proof of Proposition 3.6, which will be a winning strategy from all positions in $W_\Diamond$.

Thus, the minimal progress measure for player $\Diamond$ provides the winning set together with a winning strategy that justifies each node in this set. We may call the pair $(W_\Diamond, E_\Diamond)$ a *blueprint* of the game for player $\Diamond$.

3.3.1 Complexity

In the following we are going to estimate the cost of a blueprint in terms of complexity. For a game $\mathcal{G} = (V, E, \Omega)$ of index d, let $n := |V|$ be the number of positions and $k = |E|$ be the number of edges.

The procedure `MinProgressMeasure` requires space only to store the function $p \in P$, i. e. a d-tuple of natural numbers for any vertex $v \in V$. Thus, the space complexity is $O(nd)$.

Regarding time complexity, for any vertex v the function $\mathrm{Lift}_v(p)$ works in time $O(d \cdot |vE|)$ being called at most $\big|[\Omega]\big|$ times. So the time required by

`MinProgressMeasure` is bounded by

$$O\Big(\sum_{v\in V} d\cdot|vE|\cdot\big|[\Omega]\big|\Big) = O(dk\cdot\big|[\Omega]\big|).$$

For $\big|[\Omega]\big|$ we have

$$\big|[\Omega]\big| = \prod_{i=1}^{\lceil d/2\rceil}(|\Omega_{2i-1}|+1) \leq \Big(\frac{n}{\lceil d/2\rceil}\Big)^{\lceil d/2\rceil}.$$

Hence, the worst-case time complexity of `ProgressMeasureLifting` is

$$O\left(k\Big(\frac{n}{\lceil d/2\rceil}\Big)^{\lceil d/2\rceil}\right).$$

With a little effort we can discard the factor 1/2 in the denominator (see [Jur00] for this) getting the following result as a reference for complexity.

Theorem 3.10. *There is an algorithm determining the winner of a parity game which works in in linear space and in time bounded by*

$$O\left(k\Big(\frac{n}{d}\Big)^{\lceil d/2\rceil}\right).$$

3.3.2 Nondeterminism.

Allowing nondeterminism, we can do much better. We will first consider the original problem, of determining the winner from a position $v_0 \in V$. However, The algorithm which we are going to develop will work towards a complete blueprint.

A winning strategy for player $\Diamond$ is a labelling of the edges. Such a labelling $E_\Diamond \subseteq E$ can be guessed in polynomial time. Let U contain the nodes reachable from v_0 by $E_\Diamond$ and set $\mathcal{G}_\Diamond = (U, E_\Diamond, \Omega\restriction_U)$. The next step is to verify that $\mathcal{G}_\Diamond$ is a winning strategy for $\Diamond$.

A way to do this is to adapt the procedure in the proof of Proposition 3.7 such that it rejects whenever an attack is intended on a cycle which is not even or when a node in $V_\Diamond$ with no successors is met. With this modification the progress measure labelling accepts $\mathcal{G}_\Diamond$ iff it is a winning strategy. In this case, it will additionally return a (non-minimal) progress measure on U.

In Chapter 4 we will present an algorithm to verify a claimed winning strategy in linear time. Nevertheless the above algorithm already works in polynomial (at most quadratic) time, showing that our problem is in NP.

On the other hand, parity games can be easily complemented. Consider the game $\mathcal{G}^\neg := (V, E, \Omega + 1)$ obtained by interchanging $V_\Diamond$ and $V_\Box$ and increasing $\Omega(v)$ by 1 for all vertices $v \in V$. It is easy to see that from any starting position player $\Box$ wins in $\mathcal{G}^\neg$ iff player $\Diamond$ wins in $\mathcal{G}$.

Finally, we can interleave the runs of the above algorithm on $\mathcal{G}$ and $\mathcal{G}^{\neg}$. Then, by determinacy, the resulting algorithm accepts, returning a winning strategy for $\Diamond$ if the thread on $\mathcal{G}$ was successful and a winning strategy for $\Box$ otherwise.

This leads us to a result which was first formulated by Emerson, Jutla and Sistla in [EJS93].

Theorem 3.11. *The problem of establishing the winner of a parity game starting at a given position is in* NP $\cap$ Co-NP.

The above algorithm can be easily extended to return blueprints for both players as follows: Guess a labelling $W_\Diamond$ of V, set $W_\Box := V \setminus W_\Diamond$ and interleave the (adapted) progress measure labelling algorithms for $\mathcal{G}\restriction_{W_\Box}$ and $\mathcal{G}\restriction_{W_\Box}$.

Corollary 3.12. *The problem of computing blueprints for both players of a parity game is in* NP $\cap$ Co-NP.

Actually, more than this is known. In [Jur98] Jurdziński showed that the problem of deciding the winner of a parity game belongs to UP $\cap$ CoUP, i. e. the class of problems with a *unique* certificate of polynomial length. In this class the need of guessing seems very implausible and it is hard to believe that this problems may be NP-complete. Nevertheless, the question whether there is a polynomial algorithm for solving parity games is a long standing open problem.

Chapter 4

Model Checking

Let L be a logic and $\mathcal{C}$ a class of structures. The model checking problem (for L over $\mathcal{C}$) is the following: Given a formula $\varphi \in \mathsf{L}$, a structure $\mathfrak{A} \in \mathcal{C}$ and an assignment χ of all free variables of φ, check whether $\mathfrak{A}, \chi$ is a model of φ.

In the present chapter we consider the model checking problems for the introduced guarded logics over finite relational structures and we show that all these problems can be reduced to the problem of determining the winner of a parity game.

4.1 Model checking as a game

We first give a general reduction scheme for the model checking problem over guarded fixed point logics to parity games that we will later specialize for $\mu\mathsf{GF}$, $\mu\mathsf{CGF}$ and $\mu\mathsf{LGF}$.

Consider a finite structure $\mathfrak{A} = (A, \tau)$ and a guarded fixed point formula ψ together with an assignment χ of all free (first- and second-order) variables of ψ.

Without loss of generality we may assume that the formulae to be checked are well-named and in negation normal form. Moreover, if ψ is not a sentence we can expand $\mathfrak{A}$ with fresh constants and relation symbols corresponding to the free variables in ψ and interpreted according to χ, obtaining a problem where ψ is a sentence and χ is the empty assignment $\emptyset$.

Let $\mathrm{VA}(\psi, A)$ stand for the (finite) set of all assignments of first-order variables from ψ with elements of A.

Definition 4.1. (Context)

- We say a pair $(\varphi, \varrho) \in \mathrm{sf}(\psi) \times \mathrm{VA}(\psi, A)$ is a *context* if all free first-order variables of φ belong to the domain of ϱ.
- A context (φ, ϱ) is *atomic true* if

$$\begin{array}{ll} \varphi = R\boldsymbol{x} & \text{and } \varrho(\boldsymbol{x}) \in R^{\mathfrak{A}} \text{ for some relation symbol } R \in \tau \text{ or} \\ \varphi = \neg R\boldsymbol{x} & \text{and } \varrho(\boldsymbol{x}) \notin R^{\mathfrak{A}} \text{ or} \\ \varphi = (\boldsymbol{x} = \boldsymbol{y}) & \text{and } \varrho(\boldsymbol{x}) = \varrho(\boldsymbol{y}). \end{array}$$

- Conversely, a context (φ, ϱ) is *atomic false* if

$$\begin{array}{ll} \varphi = R\boldsymbol{x} & \text{and } \varrho(\boldsymbol{x}) \notin R^{\mathfrak{A}} \text{ or} \\ \varphi = \neg R\boldsymbol{x} & \text{and } \varrho(\boldsymbol{x}) \in R^{\mathfrak{A}} \text{ or} \\ \varphi = (\boldsymbol{x} = \boldsymbol{y}) & \text{and } \varrho(\boldsymbol{x}) \neq \varrho(\boldsymbol{y}). \end{array}$$

To the model checking problem given by $\mathfrak{A}$ and ψ we will now construct an associated parity game $\mathcal{G}(\mathfrak{A}, \psi) = (V, E, \Omega)$ as follows.

The playground V consists of contexts $(\varphi, \varrho) \in \mathrm{sf}(\psi) \times \mathrm{VA}(\psi, A)$; later on we will properly restrict this set for complexity considerations. There are two partitions:

$$\begin{array}{rcl} V_\Diamond & = & \{(\varphi, \varrho) \in V \ : \ \varphi = \eta \vee \delta \text{ or } \varphi = \exists \boldsymbol{y}.\eta \text{ or } (\varphi, \varrho) \text{ is atomic false}\}, \\ V_\Box & = & V \setminus V_\Diamond. \end{array}$$

To define the **edge relation** we first introduce some technical definitions from [GW99].

Definition 4.2. (Binding)

- If a second-order variable T is bound in ψ then the *binding definition* of T in ψ is the (unique) fixed point predicate in ψ of the form $[\mathrm{FP}\ T\boldsymbol{x}.\varphi(T, \boldsymbol{x})]$.
- The *definition list* for ψ is the function D_ψ assigning to each fixed point variable in ψ its binding definition.
- A variable T is called an LFP-*variable* if $D_\Psi(T)$ is an LFP-predicate; similarly we define GFP-variables.

We sometimes refer to the *kind* of a fixed point variable, meaning its being an LFP or GFP-variable.

Let $\varrho{\restriction}\boldsymbol{x}$ be the restriction of the assignment function ϱ to the set $\boldsymbol{x}$ of variables and let $\varrho[\boldsymbol{x}/\boldsymbol{a}]$ be the assignment ϱ' that agrees with ϱ everywhere except $\boldsymbol{x}$ where $\varrho(\boldsymbol{x}) = \boldsymbol{a}$.

Given a position $(\varphi, \varrho) \in V$ we define the set of its successors $(\varphi, \varrho)E$.

- If $\varphi = \eta \vee \delta$ or $\varphi = \eta \wedge \delta$ then

$$(\varphi, \varrho)E = \{(\eta, \varrho), (\delta, \varrho)\}.$$

- If $\varphi = (\exists \boldsymbol{y}.\alpha).\eta(\boldsymbol{x}, \boldsymbol{y})$ or $\varphi = (\forall \boldsymbol{y}.\alpha).\eta(\boldsymbol{x}, \boldsymbol{y})$ then

$$(\varphi, \varrho)E = \Big\{(\eta, \varrho') \ : \ \varrho' = (\varrho{\restriction}\boldsymbol{x})\,[\boldsymbol{y}/\boldsymbol{b}] \ \text{ for some } \boldsymbol{b} \in \alpha^{\mathfrak{A},\varrho}(\boldsymbol{x}, \underline{\ \ })\Big\}.$$

- If $\varphi = [\text{FP } T\boldsymbol{x}.\eta](\boldsymbol{x})$ then

$$(\varphi, \varrho)E = \{(\eta, \varrho)\}.$$

- If $\varphi = T\boldsymbol{x}$ for a fixed point variable T with $D_\psi(T) = [\text{FP } T\boldsymbol{x}.\eta(T, \boldsymbol{x})]$ then

$$(\varphi, \varrho)E = \{(\eta, \varrho)\}.$$

Finally, E is the set of all edges going out from game positions.

$$E = \bigcup \big\{ (v, w) \ : \ v \in V,\ w \in vE \big\}.$$

The **priority labelling** is based on an ordering of the fixed point variables that are bound in ψ.

Definition 4.3. (**Syntactic order**)

Let $<_\psi$ be a binary relation over $\text{dom}(D_\psi)$ defined by $T <_\psi T'$ if the symbol T occurs in ψ (viewed as a string) before any occurence of T'. Obviously $<_\psi$ is a linear order. We will call it *syntactic order.*

Remark 4.4. Whenever T occurs free in $D_\psi(T')$ then $D_\psi(T')$ is properly included in $D_\psi(T)$ and therefore $T <_\psi T'$.

Let $T_1 <_\psi \cdots <_\psi T_d$ be the enumeration of the fixed point variables in ψ with respect to $<_\psi$. We define Ω by

$$\Omega(\varphi, \varrho) = \begin{cases} 2i - 1 & \text{if } \varphi = T_i\boldsymbol{x} \text{ and } T_i \text{ is an LFP-variable,} \\ 2i & \text{if } \varphi = T_i\boldsymbol{x} \text{ and } T_i \text{ is a GFP-variable,} \\ 2d + 2 & \text{otherwise.} \end{cases}$$

4.2 The reduction theorem

Theorem 4.5. *Player $\Diamond$ wins the game $\mathcal{G}(\mathfrak{A}, \psi)$ starting at position $(\psi, \emptyset)$ iff $\mathfrak{A} \models \psi$ holds.*

Approximations and signatures

For the proof of the above theorem we will adapt a technique introduced by Grädel and Walukiewicz in [GW99] involving signatures. The notion of a signature was defined by Streett and Emerson in [SE89] in connection with infinite models. As we consider only finite models we will here use the concept in a simplified form.

Given a formula φ with a free k-ary second-order variable T and a formula $\gamma(\boldsymbol{x})$ with free variables $\boldsymbol{x} = x_1, \ldots, x_k$, we write $\varphi[T/\gamma]$ for the formula obtained by replacing all occurences of atoms $T\boldsymbol{y}$ in φ by $\gamma(\boldsymbol{y})$.

Definition 4.6. (**Approximants**) For all $j \in \mathbb{N}$ we define *approximants* $\eta^j(\boldsymbol{x})$ of an LFP-predicate $[\mathrm{LFP}\ T\boldsymbol{x}.\eta(T,\boldsymbol{x})]$ inductively as follows:

$\eta^0(\boldsymbol{x}) = \mathsf{ff}$
$\eta^{j+1}(\boldsymbol{x}) = \eta[T/\eta^j](\boldsymbol{x})$

By the Knaster-Tarski Theorem the least fixed point of a monotonic operator over subsets of a finite set can be obtained inductively, i. e., for a formula η with no free second-order variables other than T we have

$$\mathfrak{A}, \varrho \models [\mathrm{LFP}\ T\boldsymbol{x}.\eta(T,\boldsymbol{x})](\boldsymbol{x}) \quad \text{iff} \quad \mathfrak{A}, \varrho \models \eta^k(\boldsymbol{x}) \text{ for some } k \in \mathbb{N}.$$

Let (φ, ϱ) be a context and $T_1 <_\psi \cdots <_\psi T_n$ the enumeration of the fixed point variables in ψ. For a tuple of natural numbers $\boldsymbol{s} = (s_1, \dots, s_n)$ we define the sentence

$$\varphi^{\boldsymbol{s}} := \varphi[T_n/D^\dagger(T_n)][T_{n-1}/D^\dagger(T_{n-1})]\dots[T_1/D^\dagger(T_1)]$$

where

$$D^\dagger(T_i) = \begin{cases} D_\psi(T_i) & \text{if } T_i \text{ is a GFP-variable} \\ \eta^{s_i} & \text{if } T_i \text{ is an LFP-variable with } D_\psi(T_i) = [\mathrm{LFP}\ T_i\boldsymbol{x}.\eta]. \end{cases}$$

Definition 4.7. (**Signature**) The LFP-*signature* of a context (φ, ϱ) in $\mathfrak{A}$, denoted by $\mathrm{Sig}_\mu^{\mathfrak{A}}(\varphi, \varrho)$, is the lexicographically smallest tuple of natural numbers $\boldsymbol{s}$ such that $\mathfrak{A}, \varrho \models \varphi^{\boldsymbol{s}}$ holds.

Remark 4.8. As the definition of $\varphi^{\boldsymbol{s}}$ involves s_i only on positions i corresponding to LFP-variables, the definition of $\mathrm{Sig}_\mu^{\mathfrak{A}}(\varphi, \varrho)$ implies $s_i = 0$ on all positions corresponding to GFP-variables. Note further that the signature $\mathrm{Sig}_\mu^{\mathfrak{A}}(\varphi, \varrho)$ is defined if and only if $\mathfrak{A}, \varrho \models \varphi[T_n/D_\psi(T_n)][T_{n-1}/D_\psi(T_{n-1})]\dots[T_1/D_\psi(T_1)]$, i. e. on contexts which are "true within ψ". Consequently, atomic false contexts do not have a signature.

Lemma 4.9. *The following rules hold for signatures:*

(1) If $(\eta \lor \delta, \varrho)$ has signature $\boldsymbol{s}$ in $\mathfrak{A}$ then at least one of (η, ϱ) and (δ, ϱ) has signature $\boldsymbol{s}$ in $\mathfrak{A}$.

(2) If $(\eta \land \delta, \varrho)$ has signature $\boldsymbol{s}$ in $\mathfrak{A}$ then both (η, ϱ) and (δ, ϱ) have signature not bigger than $\boldsymbol{s}$ in $\mathfrak{A}$.

(3) If $((\forall \boldsymbol{y}.\alpha)\eta(\boldsymbol{x},\boldsymbol{y}), \varrho)$ has signature $\boldsymbol{s}$ in $\mathfrak{A}$ then for every tuple $\boldsymbol{b} \in \alpha^{\mathfrak{A},\varrho}(\boldsymbol{x}, \_\_)$ the context $(\eta(\boldsymbol{x},\boldsymbol{y}), \varrho{\restriction}_{\boldsymbol{x}}\ [\boldsymbol{y}/\boldsymbol{b}])$ has signature not bigger than $\boldsymbol{s}$ in $\mathfrak{A}$.

(4) If $((\exists \boldsymbol{x}.\alpha)\eta(\boldsymbol{x},\boldsymbol{y}), \varrho)$ has signature $\boldsymbol{s}$ in $\mathfrak{A}$ then there is a tuple $\boldsymbol{b} \in \alpha^{\mathfrak{A},\varrho}(\boldsymbol{x}, \_\_)$ such that the context $(\eta(\boldsymbol{x},\boldsymbol{y}), \varrho{\restriction}_{\boldsymbol{x}}\ [\boldsymbol{y}/\boldsymbol{b}])$ has signature $\boldsymbol{s}$ in $\mathfrak{A}$.

(5) If $([\mathrm{FP}\ T_i\boldsymbol{x}.\eta](\boldsymbol{x}), \varrho)$ has signature $\boldsymbol{s}$ then $\mathrm{Sig}_\mu^{\mathfrak{A}}(\eta, \varrho) =_{i-1} \boldsymbol{s}$.

(6) If $(T_i\boldsymbol{x}, \varrho)$ has signature $\boldsymbol{s}$ and T_i is an LFP-variable with $D_\psi(T_i) = [\mathrm{LFP}\ T_i\boldsymbol{x}.\eta]$ then $\mathrm{Sig}_\mu^{\mathfrak{A}}(\eta, \varrho) <_i \boldsymbol{s}$.

(7) *If* $(T_i\boldsymbol{x}, \varrho)$ *has signature* $\boldsymbol{s}$ *and* T_i *is an* GFP*-variable with* $D_\psi(T_i) = [\mathrm{GFP}\ T_i\boldsymbol{x}.\eta]$ *then* $\mathrm{Sig}^{\mathfrak{A}}_\mu(\eta, \varrho) = \boldsymbol{s}$.

Proof. Let us observe first that, any guarded fixed point formula φ being monotonous, $\mathfrak{A}, \varrho \models \varphi^{\boldsymbol{s}}$ implies $\mathfrak{A}, \varrho \models \varphi^{\boldsymbol{t}}$ for every $\boldsymbol{t} \geq \boldsymbol{s}$.

(1) By definition of $\varphi^{\boldsymbol{s}}$ we have $\mathfrak{A}, \varrho \models (\eta \vee \delta)^{\boldsymbol{s}}$ iff $\mathfrak{A}, \varrho \models \eta^{\boldsymbol{s}} \vee \delta^{\boldsymbol{s}}$ which means that either $\mathfrak{A}, \varrho \models \eta^{\boldsymbol{s}}$ or $\mathfrak{A}, \varrho \models \delta^{\boldsymbol{s}}$. Hence, at least one of η and δ have some signature $\boldsymbol{t} \leq \boldsymbol{s}$ and therefore $\mathfrak{A} \models (\eta \vee \delta)^{\boldsymbol{t}}$ must hold. As $\boldsymbol{s}$ is minimal it follows that $\boldsymbol{s} = \boldsymbol{t}$. Property (2) follows in the same way.

(3) As the step from φ to $\varphi^{\boldsymbol{s}}$ touches only second-order variables we have $\mathfrak{A}, \varrho \models \big((\forall \boldsymbol{y}.\alpha)\eta(\boldsymbol{x}, \boldsymbol{y})\big)^{\boldsymbol{s}}$ iff $\mathfrak{A}, \varrho \models (\forall \boldsymbol{y}.\alpha)(\eta(\boldsymbol{x}, \boldsymbol{y})^{\boldsymbol{s}})$. By semantics of guarded logics, this holds iff for every tuple $\boldsymbol{b} \in \alpha^{\mathfrak{A},\varrho}(\boldsymbol{x}, \_\_)$ we have $\mathfrak{A}, \varrho[\boldsymbol{y}/\boldsymbol{b}] \models \eta(\boldsymbol{x}, \boldsymbol{y})^{\boldsymbol{s}}$. As η, and thus $\eta^{\boldsymbol{s}}$, containes no free first-order variables besides $\boldsymbol{x} \cup \boldsymbol{y}$ this is equivalent to $\mathfrak{A}, \varrho\restriction_x [\boldsymbol{y}/\boldsymbol{b}] \models \eta(\boldsymbol{x}, \boldsymbol{y})^{\boldsymbol{s}}$. Hence the signatures of $(\eta(\boldsymbol{x}, \boldsymbol{y}), \varrho\restriction_{\boldsymbol{x}} [\boldsymbol{y}/\boldsymbol{b}])$ are defined for any appropriate $\boldsymbol{b}$ and all lexicographically below $\boldsymbol{s}$. The existential case (4) is analogous.

(5) By Remark 4.4, none of the variables T_j with $j > i$ appear free neither in $[\mathrm{FP}\ T_i\boldsymbol{x}.\eta](\boldsymbol{x})$ nor in η. Therefore $\mathfrak{A}, \varrho \models [\mathrm{FP}\ T_i\boldsymbol{x}.\eta]^{\boldsymbol{s}}$ is equivalent to

$$\mathfrak{A}, \varrho \models [\mathrm{FP}\ T_i\boldsymbol{x}.\eta][T_{i-1}/D^\dagger(T_{i-1})] \ldots [T_1/D^\dagger(T_1)], \tag{4.1}$$

Let $\eta^\dagger$ denote the formula $\eta[T_{i-1}/D^\dagger(T_{i-1})] \ldots [T_1/D^\dagger(T_1)]$. Please note that the only free second-order variable left in $\eta^\dagger$ is T_i. By the rules of substitution, we can now write relation 4.1 as $\mathfrak{A}, \varrho \models [\mathrm{FP}\ T_i\boldsymbol{x}.\eta^\dagger]$ which, by fixed point semantics, is equivalent to $\mathfrak{A}, \varrho \models \eta^\dagger\big[T_i/[\mathrm{FP}\ T_i\boldsymbol{x}.\eta^\dagger]\big]$.

Unrolling the outer occurence of $\eta^\dagger$ we get

$$\mathfrak{A}, \varrho \models \eta[T_{i-1}/D^\dagger(T_{i-1})] \ldots [T_1/D^\dagger(T_1)]\big[T_i/[\mathrm{FP}\ T_i\boldsymbol{x}.\eta^\dagger]\big].$$

As none of $T_1 \ldots T_{i-1}$ is free in $\eta^\dagger$ we can change the order of substitution to obtain

$$\mathfrak{A}, \varrho \models \eta\big[T_i/[\mathrm{FP}\ T_i\boldsymbol{x}.\eta^\dagger]\big][T_{i-1}/D^\dagger(T_{i-1})] \ldots [T_1/D^\dagger(T_1)].$$

Hence, the first $i-1$ positions of the signature of $([\mathrm{FP}\ T_i\boldsymbol{x}.\eta](\boldsymbol{x}), \varrho)$ are equal to the corresponding positions in the signature of (η, ϱ).

(6) Again, none of the variables T_j with $j > i$ appear free in $D_\psi(T_i)$, such that $\mathfrak{A}, \varrho \models (T_i\boldsymbol{x})^{\boldsymbol{s}}$ means

$$\mathfrak{A}, \varrho \models \eta^{s_i}[T_{i-1}/D^\dagger(T_{i-1})] \ldots [T_1/D^\dagger(T_1)].$$

By the definition of approximants, this is equivalent to

$$\mathfrak{A}, \varrho \models \eta[T_i/\eta^{s_i - 1}][T_{i-1}/D^\dagger(T_{i-1})] \ldots [T_1/D^\dagger(T_1)],$$

implying that the first i positions of $\mathrm{Sig}^{\mathfrak{A}}_\mu(\eta, \varrho)$ are $(s_1, \ldots, s_{i-1}, s_i - 1)$.

(7) If T is a GFP-variable, $\mathfrak{A}, \varrho \models (T_i\boldsymbol{x})^{\boldsymbol{S}}$ means by definition

$$\mathfrak{A}, \varrho \models \eta[T_i/D^\dagger(T_i)][T_{i-1}/D^\dagger(T_{i-1})]\dots[T_1/D^\dagger(T_1)].$$

and consequently the signatures of $(T_i\boldsymbol{x}, \varrho)$ and $\eta., \varrho)$ are equal.

□

Let us now return to the reduction theorem.

($\Longrightarrow$) The above lemma almost states that signatures are progress measures. Consider the following modification of the priority function.

$$\Omega'(\varphi, \varrho) = \begin{cases} 2i-2 & \text{if } \varphi = [\text{FP } T_i\boldsymbol{x}.\eta](\boldsymbol{x}), \\ \Omega(\varphi, \varrho) & \text{otherwise.} \end{cases}$$

Allowing zero priorities in parity games for the extent of this proof, consider the game $\mathcal{G}' = (V, E, \Omega')$. It is easy to see that the modified priorities will never decide a game, as the definition of T_i, $[\text{FP } T_i\boldsymbol{x}.\eta](\boldsymbol{x})$ may be visited infinitely often only if some fixed point predicate $T_j\boldsymbol{x}$ with $j \leq i-1$ is visited infinitely often. Consequently, player $\Diamond$ will win precisely the same games in $\mathcal{G}'$ and in $\mathcal{G}$ such that $\mathcal{G}'$ and $\mathcal{G}$ can be identified whithout loss.

To close the dimensional gap between progress measures and signatures, we define $p : V \to \mathbb{N}^{2n+2} \cup \{\top\}$ for every $v \in V$ as follows.

- If v has no LFP-signature then $p(v) = \top$.
- Otherwise, for any i such that T_i is an LFP-variable, copy the i-th position of $\mathrm{Sig}_\mu^{\mathfrak{A}}(v)$ to position $2i+1$ in $p(v)$ and set all the remaining positions to 0.

As a direct consequence of the lemma about signatures the mapping p is a progress measure for player $\Diamond$ in $\mathcal{G}(\mathfrak{A}, \psi)$. Since the context $(\psi, \emptyset)$ has no free second-order variables and $\mathfrak{A}, \emptyset \models \psi$ we have $\mathrm{Sig}_\mu^{\mathfrak{A}}(\psi, \emptyset) = \mathbf{0}$ and hence $p(\psi, \emptyset) = \mathbf{0} < \top$. By Proposition 3.6 this implies that player $\Diamond$ wins the game $\mathcal{G}(\mathfrak{A}, \psi)$ from $(\psi, \emptyset)$ concluding the first part of the proof.

($\Longleftarrow$) To prove the reverse statement, suppose that ψ is *not* true in $\mathfrak{A}$.

For any formula $\varphi \in \mathrm{sf}(\psi)$, let $\varphi^\neg$ be the sentence obtained by putting $\neg\varphi$ into negated normal form. Note that, if ψ is well-named, $\varphi^\neg$ will be well-named too. If $\mathfrak{A}, \varrho \not\models \varphi$ for some assignment ϱ of $\mathrm{free}(\varphi)$, then $\mathfrak{A}, \varrho \models \varphi^\neg$ and the LFP-signature $\mathrm{Sig}_\mu^{\mathfrak{A}}(\varphi^\neg, \varrho)$ is defined. We can call this signature the GFP-*signature* of (φ, ϱ) in $\mathfrak{A}$ and denote it by $\mathrm{Sig}_\nu^{\mathfrak{A}}(\varphi, \varrho)$. The name is suggested by the fact that normalization of a negated formula changes every LFP−operator into an GFP−operator and vice versa.

Our aim is now to use the GFP-signature as a winning certificate for □. By dualizing the definition of progress measures for player $\Diamond$ we obtain progress measures for player □. Then, the dual of Proposition 3.6 is straightforward:

Lemma 4.10. *If for some $v_0 \in V$ player $\Box$ has a progress measure q with $p(v_0) < \top$ then he wins the game from v_0.*

For GFP−signatures the dual of lemma 4.9 holds. To spread a GFP-signature in order to obtain a progress measure for $\Box$ we proceed as above defining $q : V \to \mathbb{N}^{2n+2} \cup \{\top\}$ such that for every $v \in V$:

- If v has no GFP-signature then $p(v) = \top$.
- Otherwise, for any i such that T_i is a GFP-variable, copy the i-th position of $\mathrm{Sig}^{\mathfrak{A}}_{\nu}(v)$ to position $2i$ in $p(v)$ and set all the remaining positions to 0.

As $\mathfrak{A} \models \psi^{\neg}$ the GFP-signature $\mathrm{Sig}^{\mathfrak{A}}_{\nu}(\psi, \emptyset)$ is defined (actually $= \mathbf{0}$) and we get $q(\psi, \emptyset) < \top$. Hence, player $\Box$ wins the game $\mathcal{G}(\mathfrak{A}, \psi)$ from $(\psi, \emptyset)$.

$\Box$

Reduction algorithm

In the following we adopt the convention that the structure $\mathfrak{A}$ is given in *list representation*, i. e. each relation is described by the list of tuples it contains. Then the *size* $|\mathfrak{A}|$ of $\mathfrak{A}$ is defined as the number of tuples contained in the relations of $\mathfrak{A}$ plus the number of elements in the universe.

The following algorithm accepts (an encoding of) a model checking problem $(\mathfrak{A}, \psi)$ as an input and constructs the associated parity game $\mathcal{G} = (V, E, \Omega)$.

To this end it performs a breadth-first search to collect the set of all reachable game positions in V. Starting with the initial position $(\psi, \emptyset)$, all positions reachable along some edge are iteratively added to the playground. Thereby the edge relation is set up. A list U is maintained to contain the front positions that have still to be followed. The function $\mathrm{S}ucc(v)$ returns the set vE of successors of v.

```
Input: (𝔄, ψ)
V := ∅; E := ∅
U := (ψ, ∅)    % front position
for each  v ∈ U do
    V := V ∪ {v}
    Compute Ω(v)
    W := Succ(v)
    for each  w ∈ W do E := E ∪ {(v, w)} od
    U := (U ∪ W) \ V       % move front discarding processed positions
od
Output: (V, E, Ω)
```

The syntactic order can be stored in a table while scanning the formula. This way the labelling with Ω can be done in constant time for each position. For any node $v \in V$ the function $\mathrm{S}ucc(v)$ works in time $O(|vE|)$ and every execution of the outer for-loop adds a new position to V. Thus we get the following complexity bound.

Lemma 4.11. *The reduction of a model checking problem to a parity game can be done in linear time in the size of the resulting game.*

It might look like we got the wrong end, relating the complexity of this algorithm to the output rather than to the input, but we shall see in the next section that in this context the approach makes sense.

4.3 Complexity issues

The computational complexity of model checking problems has been extensively studied in connection with query evaluation over relational databases. There are three distinct notions of complexity.

1. We can fix the structure $\mathfrak{A}$ and check different sentences of a logic against this structure. In this case we measure the complexity as a function of the length of the sentence and call it the *expression complexity* of the logic.
2. We can have a fixed sentence that is checked against different structures. In this case we measure the complexity as a function of the size of the structure and call it the *data complexity* of the logic.
3. We can evaluate different sentences of a logic against different structures and measure the complexity as a function of the combined size of the structure and the length of the sentence. We will call this the *combined complexity* of the considered logic.

The reduction scheme in the previous section provides a model checking algorithm for every guarded fixed point logic: given $(\mathfrak{A},\ \psi)$, construct the corresponding game $\mathcal{G}(\mathfrak{A}, \psi)$ and check if player $\Diamond$ has a winning strategy. The complexity of this algorithm is determined by the complexity of the reduction and the complexity of the resulting parity game.

As the question of whether there is a polynomial algorithm for solving parity games is still open, we have to get along with time bounds growing exponentially with the index.

4.3.1 Alternation depth

The algorithm we presented in the previous section determines the winner of a parity game $\mathcal{G} = (V, E, \Omega)$ in time growing exponentially with the index: $O\big(k\left(\frac{n}{d}\right)^{\frac{d}{2}}\big)$, where $n = |V|$, $k = |E|$ and d is the index of $\mathcal{G}$.

As every algorithm for solving parity games known to us displays this behaviour, we cannot exclude the possibility that the index of a parity game has a strong impact on complexity of its solution. It is therefore worth trying to reduce unnecessary priorities. To this end we introduce the notions of dependency order and alternation level.

Definition 4.12. (**Dependency order**) Let $\sqsubset\!\!\cdot_\psi$ be a binary relation over the variables bound in ψ, defined by $T \sqsubset\!\!\cdot_\psi T'$ if T occurs free in $D_\psi(T')$. We denote the transitive closure of $\sqsubset\!\!\cdot_\psi$ by $\sqsubset_\psi$ and call it *dependency order.*

Lemma 4.13.

(1) *The dependency order $\sqsubset_\psi$ is a partial order.*
(2) *The syntactic order $<_\psi$ extends $\sqsubset_\psi$.*

Proof. (1) By definition, $\sqsubset_\psi$ is transitive. To see that it is assymetric note that whenever $T \sqsubset\!\!\cdot\, T'$ it follows that $T \neq T'$ and $D_\Psi(T)$ is properly included in $D_\Psi(T')$. (2) follows directly from remark 4.4. □

The alternation level is defined on fixed point variables appearing in some sentence ψ. Intuitively, it records the number of alternations between least and greatest fixed point variables on the longest $\sqsubset\!\!\cdot_\psi$-path descending from a variable. [1]

Definition 4.14. (Alternation level) Let ψ be a guarded fixed point sentence and T a fixed point variable appearing in ψ. Then the *alternation level* of T in ψ, $\mathrm{al}_\psi(T)$, is defined as follows.

- If $D_\psi(T)$ is independent then $\mathrm{al}_\psi(T) = 1$.
- Else, let W be the greatest free variable of $D_\psi(T)$ with respect to $<_\psi$. If W and T are both of the same kind (LFP or GFP) then $\mathrm{al}_\psi(T) = \mathrm{al}_\psi(W)$, otherwise $\mathrm{al}_\psi(T) = \mathrm{al}_\psi(W) + 1$.

We use the alternation level to introduce a measure of the alternation depth.

Definition 4.15. (Alternation depth) The *alternation depth* of a sentence ψ, is $\mathrm{ad}(\psi) = \max\{\mathrm{al}_\psi(T) \ : \ T \in \mathrm{dom}(D_\psi)\}$ if ψ contains fixed point variables. Otherwise $\mathrm{ad}(\psi) = 1$.

The following lemma characterizes the connection between the alternation order and the alternation level.

Lemma 4.16. *For any fixed point variables X and Y in $\mathrm{dom}(D_\psi)$ we have*

$$X \sqsubset_\psi Y \implies \mathrm{al}_\psi(X) \leq \mathrm{al}_\psi(Y).$$

Equality holds only if X and Y are both of the same kind.

Proof. We proceed by induction on the set of variables between X and Y in the syntactic order, $\mathcal{V} = \{T \ : \ X <_\psi T <_\psi Y\}$. If $\mathcal{V} = \emptyset$ we can conclude from $X \sqsubset_\psi Y$ and lemma 4.13 (2) that $X \sqsubset\!\!\cdot_\psi Y$. In this case X is syntactically the greatest free variable in D_Ψ and, by definition of the alternation level, $\mathrm{al}_\psi(X) \leq \mathrm{al}_\psi(Y)$, with equality only if X and Y are both FP-variables of the

[1] It is legitimate to speak about *the* longest path, as we will later show that $\sqsubset\!\!\cdot_\psi$ induces a linear order on the set $\{Y \ : \ Y \sqsubset_\psi X\}$ for every variable X.

same kind. For the induction step, we choose the greatest variable $Z \sqsubseteq Y$ in the syntactic order. The presumption $Z \notin \mathcal{V}$ fails, as it implies that all free variables of $D_\Psi(Y)$ appear before X, in contradiction to $X \sqsubset_\psi Y$. Hence, $Z \in \mathcal{V}$ and using the induction hypothesis for the set $\{T \;:\; X <_\psi T <_\psi Z\} \subset \mathcal{V}$ we obtain $\mathrm{al}(X) \leq \mathrm{al}(Z) \leq \mathrm{al}(Y)$, with equality only if X, Z, and Y are all of the same kind. □

For reasons that will soon become clear, we adjust the parity of the alternation level for every FP-variable, setting $\mathrm{al}^*_\psi(T) := \mathrm{al}_\psi(T) + 1$ if $\mathrm{al}_\psi(\varphi)$ is even (odd) and T is an LFP (GFP) variable and $\mathrm{al}^*_\psi(T) := \mathrm{al}_\psi(T)$ otherwise. Please observe that by this adjustment every LFP (GFP) variable receives an odd (even) level. Extending the correction to the alternation depth we set $\mathrm{ad}^*(\psi) := \max\{\mathrm{al}^*_\psi(T) \;:\; T \in \mathrm{dom}(D_\psi)\}$ for every fixed point formula ψ.

Consider now a model checking game $\mathcal{G}(\mathfrak{A}, \psi) = (V, E, \Omega)$ and let $\Omega' : V \mapsto \mathbb{N}$ be the priority function defined by

$$\Omega'(\varphi, \varrho) = \begin{cases} \mathrm{al}^*_\psi(T) & \text{if } \varphi = T\boldsymbol{x} \text{ and } T \text{ is a FP-variable and} \\ \mathrm{ad}^*(\psi) + 1 & \text{otherwise.} \end{cases}$$

Obviously any play of $\mathcal{G}$ is a play of the parity game $\mathcal{G}' := (V, E, \Omega')$ and viceversa.

Proposition 4.17. *Player $\Diamond$ wins a play in $\mathcal{G}$ iff she wins the same play in $\mathcal{G}'$.*

Proof. Consider a play π in $\mathcal{G}$. In case π is finite there is no question that $\Diamond$ will win in $\mathcal{G}$ iff she wins in $\mathcal{G}'$, as the winning condition involves priorities for infinite plays only.

If π is infinite, let $\mathrm{Inf}(\pi)$ denote the set of fixed point variables appearing infinitely often in π. We will first show the following.

Lemma 4.18. *The set $\mathrm{Inf}(\pi)$ has a unique least element with respect to $\sqsubset_\psi$.*

Proof. As a partial order $\sqsubset_\psi$ is wellfounded. Hence, there is an $X \in \mathrm{Inf}(\pi)$ such that there is no $Y \in \mathrm{Inf}(\pi)$ with $Y \sqsubset_\psi X$, i. e. all variables in $\mathrm{Inf}(\pi)$ are bounded in X. Let ε be an index in π at which X occurs, such that all variables appearing on indices greater than ε belong to $\mathrm{Inf}(\pi)$.

By induction we show that for any formula φ occuring in π at some index after ε, we have $\varphi \in \mathrm{sf}(D_\psi(X))$.

Let φ_i denote the formula in the context π_i. For $i = \varepsilon$ the assertion $X \in \mathrm{sf}(D_\psi(X))$ follows trivially. At indices $i > \varepsilon$, if φ_i is not an fixed point variable then, first by game rules and then by induction hypothesis, we have $\varphi_{i+1} \in \mathrm{sf}(\varphi_i) \subseteq \mathrm{sf}(D_\psi(X))$. Else, if $\varphi_i = Z\boldsymbol{x}$ for some $Z \in \mathrm{Inf}(\pi)$ then, by induction hypothesis, Z appears in $D_\psi(X)$. By the way X is choosen, Z cannot be free there and $\varphi_{i+1} = D_\psi(Z) \in \mathrm{sf}(D_\psi(X))$.

Suppose now X is not unique and let $Y \neq X$ be another least element of $\sqsubset_\psi$ in $\mathrm{Inf}(\pi)$. As Y occurs in π at some position δ after ε, we have $D_\psi(Y) \in \mathrm{sf}(D_\psi(X))$. On the other hand, X will occur again at some position after δ, implying $D_\psi(X) \in \mathrm{sf}(D_\psi(Y))$, which leads to $X = Y$, a contradiction. □

This result, together with lemma 4.16 leads us to the insight that on the set of variables occuring infinitely often the minima of dependency order, syntactic order, and alternation level are significantly related.

Corollary 4.19. *On the set* $\mathrm{Inf}(\pi)$ *the following properties hold.*

(1) The minimal elements with respect to $\sqsubset_\psi$ *and* $<_\psi$ *coincide.*

(2) The minimum with respect to $\sqsubset_\psi$ *has minimal alternation level.*

(3) All variables with minimal alternation level are of the same kind.

Proof. (1) As the syntactic order $<_\psi$ extends the alternation order $\sqsubset_\psi$ it is obvious that in any set the minimum w. r. t. $<_\psi$ is a least element w. r. t. $\sqsubset_\psi$. But as in $\mathrm{Inf}(\pi)$ the least element w. r. t. the alternation order is unique and the assertion follows.

The statement (2) is a direct consequence of lemma 4.16.

To prove statement (3), let X be the minimum of $\mathrm{Inf}(\pi)$ with respect to $\sqsubset_\psi$. Then for any variable Y with minimal alternation level, i. e. with $\mathrm{al}_\psi(Y) = \mathrm{al}_\psi(X)$ by (2), we have $X \sqsubset_\psi Y$. As in lemma 4.16 equality holds only if X and Y are FP-variables of the same kind and the assertion follows. □

With the fruit of this corollary we return to the proof of our proposition. Assume that player $\Diamond$ wins the play π in $\mathcal{G}$. Due to (1), the minimum of $\mathrm{Inf}(\pi)$ with respect to $<_\psi$ as well as $\sqsubset_\psi$ is an GFP variable. By (2), the alternation level l of this variable will be minimal among the alternation levels of $\mathrm{Inf}(\pi)$. Further, by (3), the variables of level l played infinitely often are exclusively GFP-variables. Therefore, if l is even, it will coincide with the minimal value of Ω' in the play. Otherwise, the minimal value of Ω' is $l+1$. Either way $\min\{\Omega'(T) \ : \ T \in \mathrm{Inf}(\pi)\}$ is even and $\Diamond$ wins the play π in $\mathcal{G}'$.

In the case that player $\Box$ wins the play in $\mathcal{G}$ the dual argumentation can be applied. □

In conclusion we obtain the following

Proposition 4.20. *Any guarded fixed point formula* ψ *has a model checking game of index at most* $\mathrm{ad}(\psi) + 1$.

Proof. Proposition 4.17 establishes that $\mathrm{ad}^*(\psi)$ priorities are sufficient. As any infinite game proceeds infinitely often through fixed point variables, i. e. positions of priority less than $\mathrm{ad}^*(\psi)$, in the definition of $\mathcal{G}'$ we can replace $\Omega'(v) := \mathrm{ad}^*(\psi) + 1$ by $\Omega'(v) := \mathrm{ad}^*(\psi)$ (which is at most $\mathrm{ad}(\psi) + 1$) whithout effectively changing the winning condition of the model checking game. □

Convention. By "flattening" the maximal priority to the (adjusted) alternation depth as described, we lose some of the nice properties of model checking games, e. g. that positions of non-maximal priority have unique successors. As the gain in terms of complexity is minor in general, we will primarily work with model checking games built as for proposition 4.17.

4.3.2 Reduction of $\mu\mathsf{GF}$

Given a fixed vocabulary τ, let $\psi \in \mu\mathsf{GF}$ be a guarded fixed point sentence over τ and $\mathfrak{A} = (A, \tau)$ a finite structure.

In the construction of the parity game $\mathcal{G}(\mathfrak{A}, \psi) = (V, E, \Omega)$ we can restrict V to the contexts $(\varphi, \varrho) \in \mathrm{VA}(\psi, A)$ which are reachable from $(\psi, \emptyset)$.

Please observe that from a context (φ, ϱ), the assignment ϱ is passed over along the edges of E, except when φ is existential or universal. In the latter case a new assignment, which satisfies the guard of φ, is introduced.

We slightly extend the notion of guardedness, saying that a subformula φ of ψ is *guarded in* ψ, if it lies in the scope of some quantifier. Note that ψ itself is not guarded (in ψ) and that in every context (φ, ϱ) reachable from $(\psi, \emptyset)$ where φ is not guarded, the assignment ϱ is empty. Hence, the number of reachable contexts where the formula is not guarded is bound by $|\operatorname{sf}(\psi)|$.

Fix now a subformula φ that is guarded in ψ and let $\widehat{\varphi}$ denote the least existential or universal subformula of ψ containing φ, w. l. o. g. $\widehat{\varphi} = (\exists \boldsymbol{y}.\alpha)\eta(\boldsymbol{x}, \boldsymbol{y})$. For some assignment ϱ, the context (φ, ϱ) is reachable from the initial position $(\psi, \emptyset)$ only if ϱ was previously introduced by $\widehat{\varphi}$, i. e. if (η, ϱ) is reachable too. As η is determined, this implies that there are not more reachable (φ, ϱ)'s than (η, ϱ)'s.

Be aware that η and α are fixed with φ. By the definition of E, a context (η, ϱ) is reachable only if the assignment ϱ maps the tuple of variables $(\boldsymbol{x}, \boldsymbol{y})$ to a tuple from $\alpha^{\mathfrak{A}}(\_\_)$. As the guard α is positive atomic, say $\alpha = R\boldsymbol{z}$ for some $R \in \tau$, there are $|R^{\mathfrak{A}}|$ assignments satisfying this condition. So there are at most $|R^{\mathfrak{A}}|$ reachable positions (η, ϱ). Consequently the number of reachable contexts (φ, ϱ) for a fixed φ is bound by the size of the structure, $|\mathfrak{A}|$.

Taking into consideration that there are at most $|\operatorname{sf}(\psi)| - 1$ guarded subformulae in ψ we conclude that the number of contexts reachable in $\mathcal{G}$ from $(\psi, \emptyset)$ is bound by

$$|\operatorname{sf}(\psi)| + (|\operatorname{sf}(\psi)| - 1) \cdot |\mathfrak{A}| \leq |\operatorname{sf}(\psi)| \cdot |\mathfrak{A}|.$$

The game can now be constructed by starting from position $(\psi, \emptyset)$ and adding to the playground V all positions reachable along some edge. Thereby the edge relation E can be set up.

Putting it all together, with $|\operatorname{sf}(\psi)| = O(|\psi|)$ we get

Proposition 4.21. *The reduction from the model checking problem for $\mu\mathsf{GF}$ to parity games can be done in time $O(kn)$ where k denotes length of the formula* *and n the size of the structure. The size of the resulting game is bound by the product of the length of the formula and the size of the structure.*

As a corollary, model checking via parity games for $\mu\mathsf{GF}$ has the following time complexity bounds, where $n = |\mathfrak{A}|$, $m = |\psi|$ and $d = \mathrm{ad}(\psi) + 1$ (which is $O(m)$ in the general case).

Complexity		
Combined	NP ∩ Co-NP	$O(nm\left(\frac{nm}{d}\right)^{\frac{d}{2}})$
Expression	NP ∩ Co-NP	$O(m\left(\frac{m}{d}\right)^{\frac{d}{2}})$
Data	PTIME	$O(n^c)$, $c = \frac{d}{2}$

4.3.3 Reduction of $\mu\mathsf{LGF}$ and $\mu\mathsf{CGF}$

When we step to the loosely and clique guarded fragments, things get more complicated. With strictly guarded quantification the number of free variables in a subformula is implicitly bounded by the arity of its guard. As the vocabulary is fixed, the width of a formula may never exceed the maximal arity of the relation symbols in τ. But LGF and CGF imposee no constant bound on the number of variables. In the following we will show that this leads to EXPTIME-completeness of the model checking problem for these logics.

A straightforward approach to reduce the model checking problem for $\mu\mathsf{LGF}$ and $\mu\mathsf{CGF}$ to parity games leads to assignments of $O(|\psi|)$ variables and a resulting playground of size exponential in the length of the formula.

In [Var95] Vardi formulates the intuition that, for a logic where the number of variables used in a formula is not uniformly bounded, the expression and combined complexity of the corresponding model checking problem is in general exponentially higher than the data complexity. On the other hand, restricting the number of variables in a formula narrows the gap between combined complexity and data complexity.

In our case we see this intuition confirmed.

Proposition 4.22. *With respect to expression complexity model checking for $\mu\mathsf{LGF}$ and $\mu\mathsf{CGF}$ is* EXPTIME*-complete.*

Proof. Membership follows from the straightforward reduction approach. To show hardness we will use a method inspired from [GGV], relying on results from [GLV97]. The idea is to polynomially encode the Succint Monotone Circuit Value problem into $\mu\mathsf{LGF}$. As this problem is known to be EXPTIME-complete the encoding will lead to an EXPTIME-hard model checking problem. Since $\mu\mathsf{CGF}$ containes $\mu\mathsf{LGF}$ this will prove our assertion.

Let us first recall some basic notions. A *monotone boolean circuit* is a finite relational structure $\mathcal{K} = (K, \xi)$ with signature $\xi = (\sqcap, \sqcup, \top, \bot, \odot)$ where

- $K = \{0, \dots, 2^n - 1\}$ for some $n \in \mathbb{N}$ is a set of *gates*,
- $\sqcap$ and $\sqcup$ are ternary relations, with $\sqcap xyz$ and $\sqcup xyz$ denoting that gate x is computed as the conjunction (disjunction) of gates y and z,
- $\top$, $\bot$ and $\odot$ are unary relations denoting the $\mathtt{tt}$ and $\mathtt{ff}$ (high and low) input gates and the output gate, respectively.

Moreover, it is requested that the output gate is unique and each non-input gate is computed from other gates, such that there are no cycles.

The Monotone Circuit Value problem, MCV, consists of deciding for a given circuit $\mathcal{K}$, if the value computed on its output gate is $\mathtt{tt}$. This is well-known to be PTIME-complete. We can formulate the problem in $\mu\mathsf{LGF}$ (even $\mu\mathsf{GF}$) as follows:

$$\begin{aligned}(\exists x.\odot x)[\text{LFP } Hx.\top x \vee (\exists yz.\sqcap xyz)(Hy \wedge Hz)&\\ \vee (\exists yz.\sqcup xyz)(Hy \vee Hz)](x)&\end{aligned} \tag{4.2}$$

A *variable monotone circuit* is a structure $\mathcal{K}(V) := (K, \sqcap, \sqcup, \top, \bot, \odot, V)$ with unary $V = \{1, \dots, k\}$, such that for any partition of $V = V_\top \dot\cup V_\bot$ the structure

$$\mathcal{K}(V_\top, V_\bot) := (K, \sqcap, \sqcup, \top \dot\cup V_\top, \bot \dot\cup V_\bot, \odot)$$

is a monotone boolean circuit.

Towards a binary encoding of circuits, for a given bit vector $\boldsymbol{v}$ of size k we will write $\mathcal{K}(\boldsymbol{v})$ to denote the monotone boolean circuit $\mathcal{K}(V_\top, V_\bot)$ with $V_\top := \{i \in V \ : \ v_i = 1\}$ and $V_\bot := \{i \in V \ : \ v_i = 0\}$. A gate from $\{0, \dots, 2^n - 1\}$ will be encoded by an n bit vector. For the *sort* of a gate we fix the following encoding: $\odot \mapsto 00$, $\top \mapsto 01$, $\sqcap \mapsto 10$, $\sqcup \mapsto 11$. We don't fix any encoding for $\bot$. Finally, any gate may have two predecessors, from which its value is computed. We will say the least of them is predecessor number 0 and the other one number 1. We will consider gate 0 to be fictitious and defer all (minor) notational ambiguities to it.

A *succint representation* of a monotone boolean circuit $\mathcal{K} = (\{0, \dots, 2^n - 1\}, \xi)$ is a variable monotone circuit $\mathcal{C}_\mathcal{K}(V)$, with

- $V = \{1, \dots, 2n + 3\}$, [2 gates + predecessor code (1) + sort code (2)]
- $\mathcal{C}_\mathcal{K}(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{i}, \boldsymbol{y}) \in$ MCV iff the gate denoted by $\boldsymbol{x}$ has sort s and its ith predecessor is the gate $\boldsymbol{y}$.

Our aim now is to construct a structure $\mathfrak{A}$ and a sentence $\psi_{\mathcal{C}_\mathcal{K}} \in \mu\mathsf{LGF}$ for every succint representation $\mathcal{C}_\mathcal{K}$ of a monotone boolean circuit, such that

$$\mathfrak{A} \models \psi_{\mathcal{C}_\mathcal{K}} \quad \text{iff} \quad \mathcal{K} \in \text{MCV}. \tag{4.3}$$

For the moment it looks like we know how to solve the (MCV) problem, namely by applying formula 4.2 and all we need is to get an adequate description of it.

Let us take a closer look at formula 4.2. The trouble with it is, that variables refer to gates, which are now *encoded* in the setting of the problem. So we have to rewrite the subformulae $\odot x$, $\sqcap xyz$ and $\sqcup xyz$ in such a way that they refer to encodings of gates, additionally making sure they remain loose guards.

As a basis for the binary representation, we fix a structure $\mathfrak{B}$ over $A = \{0, 1\}$ with a unary relation $T = \{1\}$ and a binary relation $E = A \times A$. Given a variable vector $\boldsymbol{x} = x_1, \ldots, x_l$ and a constant c binary representable as $c_1, \ldots, c_l$ we will use the expression $x \doteq c$ as a shorthand for $\bigwedge_{c_i=1} Tx_i \wedge \bigwedge_{c_i=0} \neg Tx_i$. Consequently, the size of $x \doteq c$ is $O(l)$.

To recover the circuit $\mathcal{K}$ from its succint representation $\mathcal{C}_{\mathcal{K}}(V)$ we need a μLGF-formula $\psi_{\mathcal{C}}(\boldsymbol{v})$, such that for a (binary) interpretation ϱ of the variable vector $\boldsymbol{v}$ we get

$$\mathfrak{A}, \varrho \models \psi_{\mathcal{C}}(\boldsymbol{v}) \quad \text{iff} \quad \mathcal{C}_{\mathcal{K}}(\varrho(\boldsymbol{v})) \in \text{MCV}. \tag{4.4}$$

This formula can be obtained from 4.2 by replacing each individual variable by a variable vector. Proceeding bottom-up we rewrite $\sqcap xyz$ as

$$\sqcap^\star(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \quad := \quad \bigvee_{(a,b,c) \in \sqcap^{\mathcal{C}}} (\boldsymbol{x} \doteq a \wedge \boldsymbol{y} \doteq b \wedge \boldsymbol{z} \doteq c)$$

Similarily we define $\sqcup^\star(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. To describe the set of high gates we have to take into consideration the free variables $\boldsymbol{v}$.

$$\top^\star(\boldsymbol{x}, \boldsymbol{v}) \quad := \quad \bigvee_{a \in \top^{\mathcal{C}}} (\boldsymbol{x} \doteq a) \vee \bigvee_{b \in V^{\mathcal{C}}} \Big(Tv_b \wedge (\boldsymbol{x} \doteq b) \Big).$$

Finally,

$$\odot^\star(\boldsymbol{x}) \quad := \quad (\boldsymbol{x} \doteq a) \qquad \text{for (the unique) } a \in \odot^{\mathcal{C}}.$$

To secure the guardedness of the rewritten formula we introduce a *blind guard* for any variable vector $\boldsymbol{x} = x_1, \ldots, x_l$ setting

$$\gamma(\boldsymbol{x}) = \bigwedge_{1 \le i < j \le l} Ex_i x_j.$$

Note that the size of $\gamma(\boldsymbol{x})$ is $O(l^2)$. (By the way, the only reason to introduce the relation E was, of course, to trick out the guard.)

Putting it all together, we obtain the following μLGF-formula satisfying the requirements of 4.4:

$$\begin{aligned}\psi_{\mathcal{C}}(\boldsymbol{v}) := (\exists \boldsymbol{x}.\gamma(\boldsymbol{vx}))\big(\odot^\star(\boldsymbol{x}) \wedge \big[\text{LFP } H\boldsymbol{vx}.&\top^\star(\boldsymbol{x}, \boldsymbol{v}) \\ &\vee (\exists \boldsymbol{yz}.\gamma(\boldsymbol{xyz}))(\sqcap^\star(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \wedge (H\boldsymbol{y} \wedge H\boldsymbol{z})) \\ &\vee (\exists \boldsymbol{yz}.\gamma(\boldsymbol{xyz}))(\sqcup^\star(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \wedge (H\boldsymbol{y} \vee H\boldsymbol{z}))\big](\boldsymbol{v}, \boldsymbol{x})\big).\end{aligned} \tag{4.5}$$

In order to formulate the MCV-problem for $\mathcal{K}$, in the next step we recover $\sqcap^{\mathcal{K}}$, $\sqcup^{\mathcal{K}}$, $\top^{\mathcal{K}}$ and $\odot^{\mathcal{K}}$ by querying the succint representation $\mathcal{C}_{\mathcal{K}}$ via $\psi_{\mathcal{C}}$. Therefore we define:

$$\begin{aligned}
\sqcap^{\mathcal{K}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}) := & \big(\exists \boldsymbol{si}.\gamma(\boldsymbol{usiv})\big)\big(\boldsymbol{s} \doteq 10 \land \boldsymbol{i} \doteq 0 \land \psi_{\mathcal{C}}(\boldsymbol{usiv})\big) \land \\
& \big(\exists \boldsymbol{si}.\gamma(\boldsymbol{usiw})\big)\big(\boldsymbol{s} \doteq 10 \land \boldsymbol{i} \doteq 1 \land \psi_{\mathcal{C}}(\boldsymbol{usiw})\big), \\
\sqcup^{\mathcal{K}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}) := & \big(\exists \boldsymbol{si}.\gamma(\boldsymbol{usiv})\big)\big(\boldsymbol{s} \doteq 11 \land \boldsymbol{i} \doteq 0 \land \psi_{\mathcal{C}}(\boldsymbol{usiv})\big) \land \\
& \big(\exists \boldsymbol{si}.\gamma(\boldsymbol{usiw})\big)\big(\boldsymbol{s} \doteq 11 \land \boldsymbol{i} \doteq 1 \land \psi_{\mathcal{C}}(\boldsymbol{usiw})\big), \\
\top^{\mathcal{K}}(\boldsymbol{u}) := & \big(\exists \boldsymbol{s}.\gamma(\boldsymbol{usiv})\big)\big(\boldsymbol{s} \doteq 01 \land \psi_{\mathcal{C}}(\boldsymbol{usiv})\big), \\
\odot^{\mathcal{K}}(\boldsymbol{u}) := & \big(\exists \boldsymbol{s}.\gamma(\boldsymbol{usiv})\big)\big(\boldsymbol{s} \doteq 00 \land \psi_{\mathcal{C}}(\boldsymbol{usiv})\big).
\end{aligned}$$

At this point we are ready to express the MCV-problem for $\mathcal{K}$ in $\mu\mathsf{LGF}$ by

$$\begin{aligned}
\psi_{\mathcal{C}_{\mathcal{K}}} = \big(\exists \boldsymbol{u}.\gamma(\boldsymbol{u})\big)\big(\odot^{\mathcal{K}}(\boldsymbol{u}) \land \big[\mathrm{LFP}\ & \boldsymbol{Tu}.\top^{\mathcal{K}}(\boldsymbol{u}) \\
& \lor (\exists \boldsymbol{vw}.\gamma(\boldsymbol{uvw}))(\sqcap^{\mathcal{K}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}) \land (T\boldsymbol{v} \land T\boldsymbol{w})) \\
& \lor (\exists \boldsymbol{vw}.\gamma(\boldsymbol{uvw}))(\sqcup^{\mathcal{K}}(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}) \land (T\boldsymbol{v} \lor T\boldsymbol{w}))\big](\boldsymbol{u})\big).
\end{aligned}$$

Let us check now that the reduction is polynomial, i. e. the compution time for $\psi_{\mathcal{K}}$ is polynomially bounded by the size of the succint representation, $\mathcal{C}_{\mathcal{K}}$.

By the size of a circuit we will mean here the number of gates. Let m be the size of the input $\mathcal{C}_{\mathcal{K}}$. Obviously $m > 2n+3$. Consider the formulae $\sqcap^{\star}$, $\sqcup^{\star}$, $\top^{\star}$ and $\odot^{\star}$ first. All of them can be set up by a scan of the input, so their cost is $O(m)$. To build $\psi_{\mathcal{C}}$ we need to add the guards γ. This takes at most $O(m^2)$ steps leading to a complexity of $O(m^2)$ for the construction of $\sqcap^{\mathcal{K}}$, $\sqcup^{\mathcal{K}}$, $\top^{\mathcal{K}}$ and $\odot^{\mathcal{K}}$ (their guards included). Overall the reduction to $\psi_{\mathcal{C}_{\mathcal{K}}}$ can be done in time $O(m^2)$.

□

From the fact that the monotone circuit value problem is expressible in strictly guarded fixed point logic and therewith in all the guarded logics that we are considering, we can derive the following

Corollary 4.23. *With respect to data complexity $\mu\mathsf{GF}$, $\mu\mathsf{LGF}$, $\mu\mathsf{CGF}$ and $\mu\mathsf{F}$ are* PTIME*-complete.*

We obtain the following time complexities for $\mu\mathsf{LGF}$ and $\mu\mathsf{CGF}$:

Complexity	
Combined	EXPTIME-complete
Expression	EXPTIME-complete
Data	PTIME

The high expression complexity of $\mu\mathsf{LGF}$ and $\mu\mathsf{CGF}$ will determine us to consider formulae of uniformly bounded width,

4.3.4 Bounded-variable fragments

In [Dzi96] Dziembowski studies the model checking complexity for different bounded-variable fragments of first order fixed point logic. He points out that for fragments with a restricted number of first and second order-variables the combined complexity is PSPACE-complete. Yet, he proposes a syntactic restriction that defines a fragment where the combined complexity of model checking is in NP $\cap$ Co-NP. This fragment coincides with the bounded-variable fragment of the unguarded fixed point logic $\mu\mathsf{F}$, i. e. the subset of formulae in $\mu\mathsf{F}$ of bounded width.

Dziembowskis proof is founded on a polynomial translation of bounded variable $\mu\mathsf{F}$-formulae into the μ-calculus. Our approach confirms its result via a polynomial reduction to parity games.

For a fixed vocabulary τ let $\psi \in \mu\mathsf{F}$ be an unguarded sentence of width at most k to be checked against a structure $\mathfrak{A} = (A, \tau)$. The playground V of the associated parity game $\mathcal{G} = (V, E, \Omega)$ will consist of contexts $(\varphi, \varrho) \in VA(\psi, A)$, where the assignment ϱ extends over a domain of at most k variables. Therefore the size of V is bound by $|\operatorname{sf}(\psi)| \cdot |A|^k$. This guarantees that our reduction algorithm will run in time at most $O(|\psi| \cdot |A|^k)$.

Proposition 4.24. *The reduction from the model checking problem to parity games for a formula in $\mu\mathsf{F}$ of whidth at most k can be done in time polynomial in the length of the formula and the size of the structure universe. The size of the resulting game is bound by the product of the length of the formula and the number of k-tuples of structure elements.*

This results in the following time complexity bounds for model checking of $\mu\mathsf{F}$ formulae of width at most k, where $n = |A|$, $m = |\psi|$ and $d = \operatorname{ad}(\psi) + 1$.

Complexity		
Combined	NP $\cap$ Co-NP	$O(n^k m \left(\frac{n^k m}{d}\right)^{\frac{d}{2}})$
Expression	NP $\cap$ Co-NP	$O(m \left(\frac{m}{d}\right)^{\frac{d}{2}})$
Data	PTIME	$O(n^c)$, $c = k\frac{d}{2}$

Of course, this complexity bounds are valid for the bounded variable fragments of $\mu\mathsf{LGF}$ and $\mu\mathsf{CGF}$ as well. However, given the strong restrictions imposed by the latter fragments to the structure of the formula a better model checking complexity seems plausible. This issue remains subject to further research.

Chapter 5

Tractable Fragments

In the previous chapter we presented a model checking algorithm for $\mu\mathsf{GF}$ that requires exponential time in both the size of the structure and the length of the formula. For practical application this is surely inadequate. Particularly as the areas in which we see a direct applicability of guarded fixed point logics, automated verification and database systems, the structures are typically huge. Of course, the membership in NP $\cap$ Co-NP entitles us to hope that there might be a polynomial algorithm, but as we will see below this would imply that strategies for parity games can be computed in PTIME, which is a long-standing problem.

At this point one could argue that we might restrict the alternation depth of the formulae considered in practice, thus obtaining a model checker working polynomially, i. e. *efficient* in the classic complexity-theoretic sense. However, even if the expressive power of the logics restricted in such a manner would suffice for real-world applications, an algorithm running in time polynomial with a degree given by the alternation depth might be unsatisfactory already for formulae of alternation depth 3.

The notion of *tractability* of an algorithm is not strictly defined in complexity theory as it depends strongly on the nature of the problem to be solved. In the setting of our problem we will say a model checking algorithm is *tractable* if it runs in time at most quadratic in the size of the of the structure and polynomial in the length of the formula. In extension, we say a logic $\mathcal{F}$ is tractable, if there is an tractable algorithm for the model checking problem for $\mathcal{F}$ over arbitrary finite structures.

In order to delimit tractable fragments of guarded fixed point logics we will consider the propositional μ-calculus of Kozen [Koz83]. This logic is interesting for several reasons. First, there is a intimate connection to parity games following from [EJ91]. Then, guarded fixed point logics can be viewed as extensions of the μ-calculus, and different tractable fragments that have been studied for this logic can be transfered to the scope of guarded fixed point logics.

5.1 On the μ-calculus

Since its introduction in 1983 a number of different variants of Kozen's μ-calculus have been proposed. In the following we will refer to the temporal version of propositional μ-calculus as introduced by Emerson and Lei in [EL86]. It can be easily shown that this formalism is equally expressive to Kozen's calculus and the versions of Cleaveland [Cle90] and Stirling and Walker [SW91].

Definition 5.1. Consider a set $\mathcal{P}$ of propositional constants and a set $\mathcal{V}$ of propositional variables fixed. The syntax of the propositional μ-calculus, $\mathsf{L}\mu$, is given by the following grammar:

$$(A) \quad \frac{}{P}\,, \qquad (\neg) \quad \frac{}{\neg P} \qquad \text{where } P \in \mathcal{P} \cup \mathcal{V} \text{ is propositional atomic;}$$

$$(\wedge) \quad \frac{\varphi, \eta}{\varphi \wedge \eta}\,, \qquad (\vee) \quad \frac{\varphi, \eta}{\varphi \vee \eta}\,, \qquad (\Box) \quad \frac{\varphi}{\Box\varphi}\,, \qquad (\Diamond) \quad \frac{\varphi}{\Diamond\varphi}\,;$$

$$(\mu) \quad \frac{\varphi}{\mu X.\varphi}\,, \qquad (\nu) \quad \frac{\varphi}{\nu X.\varphi} \qquad \begin{array}{l}\text{where } X \in \mathcal{V} \text{ is a propositional variable}\\ \text{occurring only positively in } \varphi.\end{array}$$

The meaning of $\mathsf{L}\mu$-formulae is provided by *transition systems*, i. e. structures consisting of a set of *states* with a binary relation E, the *accessibility* relation, and with unary relations corresponding to the propositional constants.

We define the semantics via an embedding of the μ-calculus into the guarded fixed point logic. To any formula $\psi \in \mathsf{L}\mu$ the *standard translation* $\psi \rightsquigarrow \psi^*(x)$ associates a formula $\psi^* \in \mu\mathsf{GF}$ with one free first-order variable. In this translation, every propositional constant is considered a unary predicate of $\mu\mathsf{GF}$. Let $\psi^*(y)$ stand for $\psi^*(x)$ where x is replaced by y at all occurences and vice versa.

$$\begin{array}{rcl}
\psi & \rightsquigarrow & \psi(x) \quad \text{if } \psi \text{ is propositional atomic;}\\
\neg\psi & \rightsquigarrow & \neg\psi^*;\\
\psi \vee \delta & \rightsquigarrow & \psi^* \vee \delta^*;\\
\psi \wedge \delta & \rightsquigarrow & \psi^* \wedge \delta^*;\\
\Diamond\psi & \rightsquigarrow & (\exists y.Exy)\psi^*(y);\\
\Box\psi & \rightsquigarrow & (\forall y.Exy)\psi^*(y);\\
\mu X.\psi & \rightsquigarrow & [\text{LFP } Xx.\psi^*(x)](x);\\
\nu X.\psi & \rightsquigarrow & [\text{GFP } Xx.\psi^*(x)](x).
\end{array}$$

Given a formula φ, a transition system $\mathcal{K}$, and an assignment χ of propositional variables, the set of states $\|\varphi\|^{\mathcal{K}}_{\chi}$ in which φ is true is now defined by

$$\|\varphi\|^{\mathcal{K}}_{\chi} := \{w \ : \ \mathcal{K}, \chi \models \varphi^*(w)\}.$$

Definition 5.2. The alternation depth of a $\mathsf{L}\mu$-formula φ is defined to be the alternation depth of its standard translation φ^*.

The embedding of $\mathsf{L}\mu$ into $\mu\mathsf{GF}$ allows the reduction scheme given in the previous chapter to be applied to model checking for the μ-calculus over transition systems. This results in a game-theoretic interpretation that resembles the model checking games for $\mathsf{L}\mu$ given by Herwig [Her89] and Stirling [Sti95].

But the connection between $\mathsf{L}\mu$ and parity games goes further. As Emerson and Jutla have shown in [EJ91], the positions of a parity game from which some player has a strategy can be characterized by a $\mathsf{L}\mu$-formula. In [Wal96] Walukiewicz employs such a characterization, which we will introduce in a slightly adapted form.

Consider a parity game $\mathcal{G} = (V, E, \Omega)$ of index n, with $V = V_\Diamond \dot\cup V_\Box$ and $\Omega : V \mapsto \{1, \ldots, n\}$. The game $\mathcal{G}$ can be represented as a transition system

$$\mathcal{G} := (V, E, V_\Diamond, V_\Box, \Omega_1, \ldots, \Omega_n),$$

where V is now considered to be a set of states and $V_\Diamond$, $V_\Box$, $\Omega_1, \ldots \Omega_n$ are unary relations, $\Omega_i^{\mathcal{G}}$ denoting the set of positions with priority i, i. e. the set $\{v \ : \ \Omega(v) = i\}$.

We may assume w. l. o. g. that n is odd. Consider the formula

$$\varphi_\Diamond(Z_1, \ldots, Z_n) := \big(V_\Diamond \land \Diamond \bigvee_{i=1}^{n} (\Omega_i \land Z_i)\big) \lor \big(V_\Box \land \Box \bigvee_{i=1}^{n} (\Omega_i \land Z_i)\big),$$

and the sentence

$$\psi_\Diamond := \mu Z_1.\nu Z_2.\ldots\mu Z_n.\varphi_\Diamond(Z_1, \ldots, Z_n).$$

For $W_\Diamond$ the set of states in $\mathcal{G}$ from which player $\Diamond$ has a winning strategy, we claim:

Proposition 5.3 (Characterization). $W_\Diamond = \|\psi_\Diamond\|^{\mathcal{G}}$.

Proof. We define the *cascade* game associated to $\mathcal{G}$ as a parity game

$$\widetilde{\mathcal{G}} = (\widetilde{V}, \widetilde{E}, \widetilde{V}_\Diamond, \widetilde{V}_\Box, \widetilde{\Omega}_1, \ldots, \widetilde{\Omega}_n),$$

where

- The set of states $\widetilde{V} := V \times \{1, \ldots, n+2\}$ consisting of $n+2$ copies of V which we call *layers*. Copies 1 to n are intended as *definition* layers, copy $n+1$ as the *choice* layer, and copy $n+2$ as the *priority* layer.
- Player $\Diamond$ receives those states on the choice layer that stem from $\Diamond$-states in the original game: $\widetilde{V}_\Diamond := V_\Diamond \times \{n+1\}$ and her opponent takes the rest $\widetilde{V}_\Box := \widetilde{V} \setminus \widetilde{V}_\Diamond$.
- For a state $(v, i) \in \widetilde{V}$ the set of successors is defined by

$$(v,i)\widetilde{E} := \begin{cases} \{(v, i+1)\} & \text{if } 1 \le i \le n, \\ \{(w, n+2) \ : \ (v, w) \in E\} & \text{if } i = n+1, \\ \{(v, j)\} & \text{if } i = n+2 \text{ and } v \in \Omega_j. \end{cases}$$

Please observe that only on the choice layer might there be states with non-unique successors. For this reason we will call states on the non-choice layers *choiceless*.

- The priority of any state v in $\mathcal{G}$ is copied to its correspondents $(v, n+1)$ on the priority layer in $\widetilde{\mathcal{G}}$. All the states of $\widetilde{\mathcal{G}}$ not containd in the priority layer recieve the highest priority, n:

$$\begin{aligned} \widetilde{\Omega}_i &:= \Omega_i \times \{n+1\} \quad \text{for } 1 \leq i < n, \text{ and} \\ \widetilde{\Omega}_n &:= \widetilde{V} \setminus \bigcup_{i=1}^{n-1} \widetilde{\Omega}_i. \end{aligned}$$

Example. Figure 5.1 illustrates the construction of the cascade game for a game $\mathcal{G}$ with states $\{a, b, c, d\}$. States on the choice layer belonging to player $\Diamond$ ($\Box$) are depicted as diamonds (boxes), choiceless states as circles. States are labeled with their priority; empty labels mean maximal priority.

Not surprisingly any winning strategy for $\mathcal{G}$ works for $\widetilde{\mathcal{G}}$ as well and vice versa.

Lemma 5.4. *Player $\Diamond$ has a winning strategy from some state v_0 in $\mathcal{G}$, iff she has a winning strategy from $(v_0, 1)$ in $\widetilde{\mathcal{G}}$.*

Proof. Any play π in $\mathcal{G}$ starting from v_0 uniquely determines a play $\widetilde{\pi}$ starting from $(v_0, 1)$ in the associated cascade game, where the players move like in π whenever the token is on the choice layer. This will always work, as the transitions from choiceless states only shift between copies of the current state. Conversely, given a play $\widetilde{\sigma}$ from $(v_0, 1)$ in $\widetilde{\mathcal{G}}$ we obtain a play σ starting from v_0 in $\mathcal{G}$ by eclipsing the choiceless states in $\widetilde{\sigma}$.

Thus, the mapping $\pi \mapsto \widetilde{\pi}$ establishes a one-to-one correspondence between plays of $\mathcal{G}$ from v_0 and plays of $\widetilde{\mathcal{G}}$ from $(v_0, 1)$ and we can easily check that player $\Diamond$ wins the play π in $\mathcal{G}$ if and only if she wins the play $\widetilde{\pi}$ in $\widetilde{\mathcal{G}}$.

For the left-to-right implication of the proof, please recall that player $\Diamond$ has a winning strategy in the game $\mathcal{G}$ from state v_0 iff $\mathcal{G}$ can be transformed by cutting edges that leave $\Diamond$-states into a game $\mathcal{G}_\Diamond$, where $\Diamond$ wins every play from v_0 (see remark 3.2).

Assuming that player $\Diamond$ has a winning strategy $\mathcal{G}_\Diamond$ from v_0 in $\mathcal{G}$, let $\widetilde{\mathcal{G}}_\Diamond$ be the cascade game associated to $\mathcal{G}_\Diamond$. Towards a contradiction suppose $\widetilde{\mathcal{G}}_\Diamond$ is not a winning strategy for player $\Diamond$ in $\widetilde{\mathcal{G}}$ from $(v_0, 1)$. In this case there is a play $\widetilde{\sigma}$ in $\widetilde{\mathcal{G}}_\Diamond$ won by player $\Box$. By the above observation player $\Box$ then wins the play σ in $\mathcal{G}_\Diamond$, in contradiction to our assumption.

For the proof of the other direction, assume that player $\Diamond$ has a winning strategy $\widetilde{E}_\Diamond$ from $(v_0, 1)$ in $\widetilde{\mathcal{G}}$ obtained by restricting the edge set $\widetilde{E}$. For every $v \in V$ set

$$vE_\Diamond := \begin{cases} vE & \text{if } v \in V_\Box \\ \{w\} & \text{if } v \in V_\Diamond, \text{ and } (v, n+1)\widetilde{E}_\Diamond = \{(w, n+2)\}. \end{cases}$$

In the same way as above it turns out that $E_\Diamond$ is a winning strategy for player $\Diamond$ in the game $\mathcal{G}$ from v_0. $\Box$

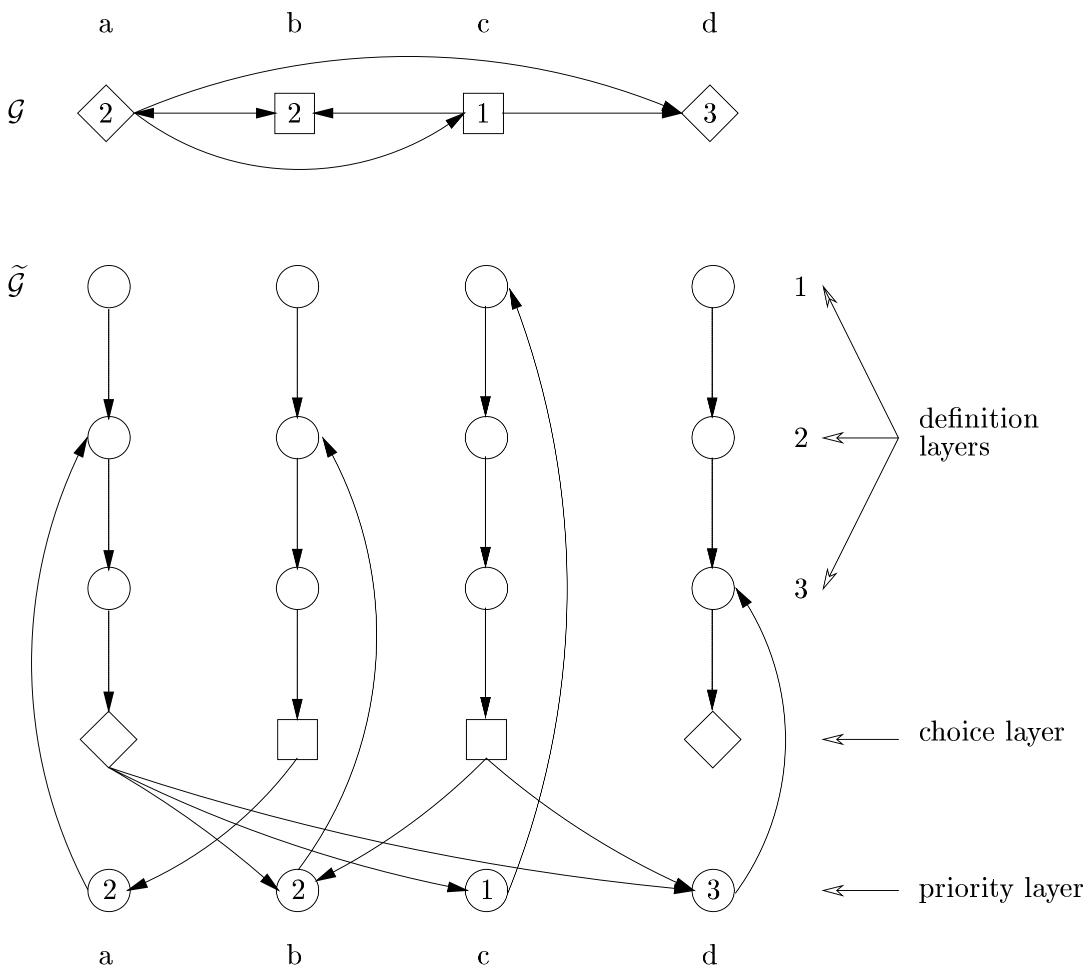


Figure 5.1: a cascade construction

Returning to the model checking game, let us rename the states of $\widetilde{\mathcal{G}}$ as pairs of $\mathsf{L}\mu$-formulae and states from V as follows. We write σZ_i for μZ_i (νZ_i), if i is odd (even). If σ stands for μ (ν), then $\bar{\sigma}$ denotes ν (μ).

- $(v, i) \leadsto \big(\sigma Z_i \bar{\sigma} Z_{i-1} \dots \mu Z_1.\varphi_\diamond(Z_1, \dots, Z_i), v\big)$ for $1 \leq i \leq n$.
- $(v, n+1) \leadsto \big(\diamond \bigvee_{i=1}^{n}(\Omega_i \wedge Z_i), v\big)$ if $v \in V_\diamond$.
- $(v, n+1) \leadsto \big(\square \bigvee_{i=1}^{n}(\Omega_i \wedge Z_i), v\big)$ if $v \in V_\square$.
- $(v, n+2) \leadsto (Z_i, v)$ where $i = \Omega(v)$.

We may now see the cascade game as the core of the model checking game for $\mathcal{G}$ and $\psi_\diamond$. Consider $\mathcal{G}(\mathcal{G}, \psi_\diamond)$ constructed as in the previous chapter, where for notational convenience formulae are written in $\mathsf{L}\mu$ and assignments $\varrho : x \mapsto v$

are denoted by v. The playground contains the *old* states of $\widetilde{\mathcal{G}}$ and additionally, for each $v \in V$ the *new* states:

(1) $\big((V_\diamond \wedge \diamond \bigvee_{i=1}^n (\Omega_i \wedge Z_i)) \vee (V_\Box \wedge \Box \bigvee_{i=1}^n (\Omega_i \wedge Z_i)), v\big)$.

(2) $\big(V_\diamond \wedge \diamond \bigvee_{i=1}^n (\Omega_i \wedge Z_i), v\big)$.

(3) $\big(V_\Box \wedge \Box \bigvee_{i=1}^n (\Omega_i \wedge Z_i), v\big)$.

(4) $\big((\Omega_k \wedge Z_k) \vee \bigvee_{i=k+1}^n (\Omega_i \wedge Z_i), v\big)$ for $1 \leq k < n$.

(5) $(\Omega_k \wedge Z_k, v)$ for $1 \leq k < n$.

(6) (Ω_k, v) for $1 \leq k < n$.

(7) $(V_\diamond, v)$ and $(V_\Box, v)$.

We will be interested in games starting from states $(\psi_\diamond, v)$ for $v \in V$, which are old.

Assume player $\diamond$ has a winning strategy $\widetilde{\mathcal{G}}_\diamond$ from some state $(\psi_\diamond, v_0)$ in $\widetilde{\mathcal{G}}$. Then by the following rules a strategy for $\diamond$ in the game $\mathcal{G}(\mathcal{G}, \psi_\diamond)$ is defined. We write simply $\bigvee_a^b$ for $\bigvee_{i=a}^b (\Omega_i \wedge Z_i)$. The rules are for every $v \in V$.

- On $(\varphi_\diamond, v)$ choose $\big(V_\diamond \wedge \diamond \bigvee_1^n, v\big)$ iff $v \in V_\diamond$.
- On $\big(\diamond \bigvee_1^n, v\big)$, if $\widetilde{\mathcal{G}}_\diamond$ chooses (Z_k, w) with $w \in \Omega_k$, then choose $\big(\bigvee_1^n, w\big)$.
- For every k: on $\big((\Omega_k \wedge Z_k) \vee \bigvee_{k+1}^n, v\big)$ choose $(\Omega_k \wedge Z_k, v)$ iff $v \in \Omega_k$.

This strategy supports the intention of player $\diamond$ to conduct the game in such a way that the sequence of old states encountered during a play of $\mathcal{G}(\mathcal{G}, \psi_\diamond)$ corresponds to a play of $\widetilde{\mathcal{G}}_\diamond$, which is winning for $\diamond$. If she succeeds herewith, she wins the model checking game too, as all the new states being played are of maximal priority, unable to affect the parity winning condition and she can never get stuck on a new state. Player $\Box$ can hinder her intention only by moving adversely on states of type (2), (3) or (5), but then he gets stuck in the next move. Hence, player $\diamond$ has a winning strategy in $\mathcal{G}(\mathcal{G}, \psi_\diamond)$ starting from $(\psi_\diamond, v_0)$.

Similarly a winning strategy for player $\Box$ can be transfered from the cascade to the model checking game. Let $\widetilde{\mathcal{G}}_\Box$ be his winning strategy in $\widetilde{\mathcal{G}}$ starting at position $(\psi_\diamond, v_0)$. We define a strategy for the game $\mathcal{G}(\mathcal{G}, \psi_\diamond)$ as follows:[1]

- On $\big(V_\diamond \wedge \diamond \bigvee_1^n, v\big)$ choose $(V_\diamond, v)$ iff $v \notin V_\diamond$.
  On $\big(V_\Box \wedge \Box \bigvee_1^n, v\big)$ choose $(V_\Box, v)$ iff $v \notin V_\Box$.
- On $\big(\Box \bigvee_1^n, v\big)$, if $\widetilde{\mathcal{G}}_\diamond$ chooses (Z_k, w) with $w \in \Omega_k$, then choose $\big(\bigvee_1^n, w\big)$.
- On $(\Omega_k \wedge Z_k, v)$ choose (Ω_k, v) iff $v \notin \Omega_k$.

[1] we specify rules only for states with possibly more than one successor

Herein the technique of $\Box$ is to choose with $\widetilde{\mathcal{G}}_\Box$ whenever an old state from the choice layer is reached, as long as the sequence of old states met in the play of $\mathcal{G}(\mathcal{G}, \psi_\Diamond)$ form a play of $\widetilde{\mathcal{G}}_\Box$. If player $\Diamond$ moves such that the new play deviates from the old one, it can only be on states of type (1) or (4). In the first case she will get stuck already in the next move by the first rule of $\Box$, while at states of type (4) she will get stuck by the last rule. Either way, the strategy is winning for player $\Box$ in $\mathcal{G}(\mathcal{G}, \psi_\Diamond)$ from state $(\psi_\Diamond, v_0)$.

With theorem 4.5 this concludes the proof of proposition 5.3. $\Box$

Notation. Due to the stated property we will call $\psi_\Diamond$ the *characteristic* formula of game $\mathcal{G}$.

Applying this above result to model checking games for guarded fixed point logics we get.

Proposition 5.5. *The model checking game for a fixed point formula ψ has a characteristic formula of alternation depth at most* $\mathrm{ad}(\psi) + 1$.

Proof. The assertion follows directly from proposition 4.20 and 5.3.

After proving that on classes of finite trees least and greatest fixed points coincide we can transfer the technique of “flattening the maximal priority” to $\psi_\Diamond$ evacuating the innermost fixed point.

$$\begin{aligned}
\psi_\Diamond &= \mu Z_1.\nu Z_2.\dots\nu Z_{n-1}\mu Z_n.\varphi_\Diamond(Z_1,\dots,Z_{n-1},Z_n)\\
&\equiv \mu Z_1.\nu Z_2.\dots\nu Z_{n-1}\nu Z_n.\varphi_\Diamond(Z_1,\dots,Z_{n-1},Z_n)\\
&\equiv \mu Z_1.\nu Z_2.\dots\nu Z_{n-1}.\varphi'_\Diamond(Z_1,\dots,Z_{n-1}\vee Z_n),
\end{aligned}$$

with

$$\varphi'_\Diamond(Z_1,\dots,Z_{n-1}) := \big(V_\Diamond \wedge \Diamond \bigvee_{i=1}^{n-1}(\Omega_i \wedge Z_i)\big) \vee \big(V_\Box \wedge \Box \bigvee_{i=1}^{n-1}(\Omega_i \wedge Z_i)\big).$$

$\Box$

As the translation from μGF and more generally from guarded fixed point logics of uniformly bounded width to parity games is polynomial the above Lμ-characterization leads us to the following

Corollary 5.6. *If there is a* PTIME *model checking algorithm for* Lμ*, then model checking for* μGF *and the bounded-variable fragments of* μLGF*,* μCGF *and* μF *can be done in polynomial time too.*

In its most general form, the above result follows already from the reduction of the bounded-variable fragment of μF [2] to Lμ given by Dziembowski in [Dzi96]. However, in a rather restricted form, we obtain a syntactic criterion for the tractability of fragments of μGF.

[2]In the quoted paper this fragment of fixed point logic is called $\widehat{\mathrm{F}P}^k$.

Corollary 5.7. *A sublogic* F *of* $\mu\mathsf{GF}$ *is tractable if the characteristic formulae of the model checking games for* F *(over arbitrary structures) are in a tractable fragment of* $\mathsf{L}\mu$.

Proof. For a sentence $\varphi \in \mathsf{F} \subseteq \mu\mathsf{GF}$ and a structure $\mathfrak{A}$ the parity game $\mathcal{G}(\mathfrak{A}, \varphi)$ can be constructed in time $O(|\mathfrak{A}| \cdot |\varphi|)$, by proposition 4.21. Checking if $\mathfrak{A} \models \varphi$ reduces to establishing if player $\Diamond$ has a winning strategy from $(\varphi, \emptyset)$ in $\mathcal{G}(\mathfrak{A}, \varphi)$, which is equivalent to checking if $(\varphi, \emptyset) \in \|\psi_\Diamond\|^{\mathcal{G}(\mathfrak{A},\varphi)}$. In the latter model checking game, the structure $\mathcal{G}(\mathfrak{A}, \varphi)$ has size $n = O(|\mathfrak{A}| \cdot |\varphi|)$ and the length of the formula $\psi_\Diamond$ depends linearly of the number of priorities in $\mathcal{G}(\mathfrak{A}, \varphi)$, say $n = O(|\varphi|)$. Finally, the assumption that $\psi_\Diamond$ is in a tractable fragment, i. e. model checking can be done in time at most $O(n^2 m^k)$ for some k, results in time at most $O\big(|\mathfrak{A}|^2 \cdot |\varphi|^{k+2}\big)$ for checking F. □

5.1.1 Tractable fragments of $\mathsf{L}\mu$

There are two fragments of $\mathsf{L}\mu$ known to admit linear time model checking both in the size of the structure and the length of the formula. The *alternation-free* fragment, identified by Emerson and Lei in [EL86], is obtained by disallowing nesting of least and greatest fixed points. The fragment L_2, defined by Emerson, Jutla, and Sistla in [EJS93] which allows arbitrary nesting of least and greatest fixed points, imposes more subtle restrictions.

The alternation-free fragment

In our formalism, the alternation-free fragment of $\mathsf{L}\mu$ consists of the formulae of alternation depth 1. An equivalent, yet more expressive, characterization was given in [BVW94] saying that a formula is alternation-free if, for all $y \in \mathcal{V}$ there are respectively no occurrences of ν (μ) on any syntactic path from an occurrence of μy (νy) to an occurrence of y.

Linear time algorithms for this fragment have been given by Cleaveland and Steffen [CS92], Stirling and Walker [SW91], Andersen [And92], and Kupferman, Vardi and Wolper [BVW94].

The approaches are different, but they all rely on a depth-first search of witnesses for membership in the winning set. The algorithms of Andersen and Stirling and Walker involve And-Or-graphs, which can be easily adapted to parity games.

The main reason for the tractability of alternation-free L_μ seems to be the possibility of decomposing the set of states into a hierarchical system of partition, such that that, once a partition is reached, the game cannot turn back to a lower partition, such that any play gets finally stuck in one of them.

The fragment L_2

Definition 5.8. The sublogic L_1 of $\mathsf{L}\mu$ as introduced by Emerson, Jutla and Sistla in [EJS93] is defined by replacing the rules ($\wedge$) and ($\Box$) in the definition of $\mathsf{L}\mu$ (Definition 5.1) by the following rules respectively:

$$(\wedge') \quad \frac{\varphi}{\varphi \wedge P}\,, \qquad (\Box') \quad \frac{}{\Box P}\,, \qquad \text{where } P \in \mathcal{P} \text{ is a propositional constant.}$$

The sublogic L_2 is defined by modifying the same rules of $\mathsf{L}\mu$ to admit closed L_2-formulae, wherever propositional constants are allowed in L_1.

$$(\wedge'') \quad \frac{\varphi, \eta}{\varphi \wedge \eta}\,, \qquad (\Box'') \quad \frac{\eta}{\Box \eta}\,, \qquad \text{where } \eta \text{ is a closed formula.}$$

Obviously L_1 is a subset of L_2. It is useful to observe that any model checking algorithm A_1 for L_1 can be extended to a model checking algorithm A_2 for L_2, that recursively evaluates closed subformulae until an L_1-subformula is reached, which is checked with A_1 and replaced by a fresh propositional constant. Thus, the complexity of A_2 will be essentially that of A_1.

In [EJS93] the authors proposed a model checking algorithm for L_1 and L_2 working in time $O(m^2 n)$ (where m is the length of the formula and n the size of the structure). This bound was later improved by Bhat and Cleveland [BC96] to $O(mnd)$ where d is the alternation depth of the formula. In [Sei] Seidl shows that the alternation depth hierarchy of L_2 collapses at level 2, presenting a reduction of model checking for L_2 to the alternation-depth-2 fragment which leads to a model checking algorithm of complexity $O(mn)$.

5.2 Tractable fragments of μGF

In the following section we will extend the presented fragments of $\mathsf{L}\mu$ to the guarded fixed point logic.

Definition 5.9. The fragment *alternation-free* μGF is the set of μGF-sentences with alternation depth 1.

Note. This fragment was previously defined by Gottlob, Grädel, and Veith in [GGV], where its tractability has been shown by reduction to Datalog LITE.

Definition 5.10. The fragment GF_1 of μGF is defined by replacing the rules $(\wedge)$ and $(\forall)$ in the definition of μGF (Definitions 2.2, 2.8) respectively by the following rules:

$$(\neg') \quad \frac{\eta}{\neg \eta}\,, \qquad \text{where } \eta \text{ is a relational atomic formula;}$$

$$(\wedge') \quad \frac{\varphi, \eta}{\varphi \wedge \eta}\,, \qquad \text{where } \eta \text{ is a relational atomic formula;}$$

$$(\forall') \quad \frac{\eta}{\forall \boldsymbol{y}(\alpha \to \eta)}\,, \qquad \text{where } \boldsymbol{y} \text{ is a tuple of variables strictly guarded by } \alpha, \ \mathrm{free}(\psi) \subseteq \mathrm{free}(\alpha) \text{ and } \eta \text{ is a relational atomic formula.}$$

In extending GF_1 to GF_2 the role of closed $\mathsf{L}\mu$-formulae in L_2 will be played by independent fixed point formulae.

Definition 5.11. The fragment GF_2 of $\mu\mathsf{GF}$ is defined by replacing the rules $(\wedge)$ and $(\forall)$ in the definition of $\mu\mathsf{GF}$ to admit independent GF_2-formulae wherever relational atomic formulae are allowed in GF_1:

$(\neg'')$ $\dfrac{\eta}{\neg\eta}$, where η is an independent formula;

$(\wedge'')$ $\dfrac{\varphi, \eta}{\varphi \wedge \eta}$, where η is independent;

$(\forall'')$ $\dfrac{\eta}{\forall \boldsymbol{y}(\alpha \to \eta)}$, where $\boldsymbol{y}$ is a tuple of variables strictly guarded by α, $\mathrm{free}(\psi) \subseteq \mathrm{free}(\alpha)$ and η is independent.

It is easy to observe that the standard translation embeds L_1 into GF_1 and L_2 into GF_2. As we will show by different approaches in the next subsections, the tractability results for the fragments of $\mathsf{L}\mu$ can be transfered to $\mu\mathsf{GF}$ as well.

Proposition 5.12. *The following sublogics of the guarded fixed point logic admit tractable model checking algorithms:*

(1) *Model checking of Alternation-free $\mu\mathsf{GF}$ cane be done in linear time in the product of the size of the structure and in the length of the formula.*

(2) *Model checking of the fragments GF_1 and GF_2 can be done in time linear in the product of the size of the structure, the length of the formula and its alternation depth.*

5.3 A syntactic approach

In a first instance we will attempt to prove proposition 5.12 via the mechanics of the characteristic formula of model checking games (see Corollary 5.7).

Proof. (5.12(1)) Let $\mathfrak{A}$ be a structure and ψ an alternation-free $\mu\mathsf{GF}$-sentence. According to proposition 4.20 the model checking game $\mathcal{G}(\mathfrak{A}, \psi)$ has index at most 2. To preserve the property of not seeing more than one active priority infinitely often, we choose not to flatten the game, but the characteristic formula. By 5.5 this formula is w. l. o. g.

$$\psi_\Diamond = \mu X.\nu Y\big((V_\Diamond \wedge \Diamond\varphi) \vee (V_\Box \wedge \Box\varphi)\big)$$

with

$$\varphi = (\Omega_1 \wedge X) \vee \big((\Omega_2 \vee \Omega_3) \wedge Y\big),$$

expressing that however player $\Box$ moves, player $\Diamond$ has a way to move such that she never gets stuck and Ω_1 is met only finitely often, while Ω_2 and Ω_3 can be met infinitely often.

As the $\mu\mathsf{GF}$ formula to be checked is alternation-free, there are no cycles containing both Ω_1 and Ω_2 states in the model checking game. Therefore $\psi_\Diamond$ is equivalent to

$$\nu Y.\mu X\big((V_\Diamond \wedge \Diamond\varphi) \vee (V_\Box \wedge \Box\varphi)\big).$$

In [EJS93] Emerson, Jutla, and Sistla state the following.

Proposition 5.13. *For a formula $\psi \in \mathsf{L}\mu$ of the form $\mu X\varphi$ where φ is in negation normal form with no further μs appearing in it, the model checking problem for ψ and some structure $\mathfrak{A}$ can be reduced in linear time to the non-emptiness problem of a Büchi tree automaton of size $O(|\mathfrak{A}| \cdot |\psi|)$.*

For the latter problem an algorithm running quadratic time in the size of the automaton is known but it is open, whether algorithms of better complexity exist.

Via the above result, we obtain a complexity bound of $O(m^2n^2)$ for model checking Alternation-free $\mu\mathsf{GF}$ which matches our condition of tractability although it does not meet the bound stated in Proposition 5.12. We will return to this problem in Section 5.5 with a game approach. $\Box$

Proof. (5.12(2)) Once the assumption is proved for GF_1, the generalization to GF_2 is straightforward.

According to 5.5, for a structure $\mathfrak{A}$ and a sentence $\psi \in \mathsf{L}_1$ the model checking game $\mathcal{G}(\mathfrak{A}, \psi)$ is characterized by

$$\psi_\Diamond = \mu Z_1.\nu Z_2.\dots\mu Z_n.\varphi_\Diamond(Z_1,\dots,Z_n),$$

with

$$\varphi_\Diamond(Z_1,\dots,Z_n) := \big(V_\Diamond \wedge \Diamond \bigvee_{i=1}^{n}(\Omega_i \wedge Z_i)\big) \vee \big(V_\Box \wedge \Box \bigvee_{i=1}^{n}(\Omega_i \wedge Z_i)\big),$$

for $n = \mathrm{ad}(\psi)$, odd w. l. o. g.

We are looking for a characteristic in L_2. Towards this we have to do some surgery on the subformula $V_\Box \wedge \Box \bigvee_{i=1}^{n}(\Omega_i \wedge Z_i)$.

For any $\eta \in \mathsf{L}\mu$ we have

$$\begin{aligned}\Box\eta &\equiv \Box\big((\Box\mathsf{ff} \wedge \Diamond\mathsf{tt}) \vee \eta\big)\\ &\equiv \Box\big((\Box\mathsf{ff} \vee \eta) \wedge (\Diamond\mathsf{tt} \vee \eta)\big) \qquad (5.1)\\ &\equiv \Box(\Box\mathsf{ff} \vee \eta) \wedge \Box(\Diamond\mathsf{tt} \vee \eta).\end{aligned}$$

Now, model checking GF_1-formulae does not result in arbitrary parity games. The syntactic restrictions of this fragment impose that any state of $V_\Box$ has at most one successor which is not a terminal state. Therefore all non-terminal successors of a $V_\Box$-state have some property η iff there is a non-terminal successor with η or, trivially, all successors are terminal. This fact can can be formalized by the following equivalence:

$$V_\Box \wedge \Box(\Box \mathsf{ff} \vee \eta) \equiv V_\Box \wedge \big(\Box\Box \mathsf{ff} \vee \Diamond(\Diamond \mathsf{tt} \wedge \eta)\big). \tag{5.2}$$

On the other hand, for any game we have

$$V_\Box \wedge \Box(\Diamond \mathsf{tt} \vee \eta) \equiv V_\Box \wedge \Box(\Diamond \mathsf{tt} \vee V_\Box) \tag{5.3}$$

if η is of the form $(V_\Diamond \wedge \Diamond\delta) \vee (V_\Box \wedge \Box\delta')$ for some $\delta, \delta' \in \mathsf{L}\mu$. To see this, we can write

$$\begin{aligned}
\Diamond \mathsf{tt} \vee \eta &= \Diamond \mathsf{tt} \vee (V_\Diamond \wedge \Diamond\delta) \vee (V_\Box \wedge \Box\delta')) \\
&\equiv \Diamond \mathsf{tt} \vee \Big(\Box \mathsf{ff} \wedge \big((V_\Diamond \wedge \Diamond\delta) \vee (V_\Box \wedge \Box\delta')\big)\Big), \text{as } \Diamond \mathsf{tt} \vee \Box \mathsf{ff} \equiv \mathsf{tt} \\
&\equiv \Diamond \mathsf{tt} \vee (V_\Diamond \wedge \underbrace{\Box \mathsf{ff} \wedge \Diamond\delta)}_{\mathsf{ff}} \vee (V_\Box \wedge \underbrace{\Box \mathsf{ff} \wedge \Box\delta')}_{\mathsf{tt}} \\
&\equiv \Diamond \mathsf{tt} \vee V_\Box.
\end{aligned}$$

Unrolling the fixed points we can prove by induction that $\eta := \bigvee_{i=1}^{n}(\Omega_i \wedge Z_i)$ is of the form required in equation 5.3.

Therefore we can write

$$\begin{aligned}
V_\Box \wedge \Box\eta &\equiv V_\Box \wedge \Box(\Box \mathsf{ff} \vee \eta) \wedge \Box(\Diamond \mathsf{tt} \vee \eta) && \text{by 5.1} \\
&\equiv V_\Box \wedge \Box(\Box \mathsf{ff} \vee \eta) \wedge \Box(\Diamond \mathsf{tt} \vee V_\Box) && \text{by 5.3} \\
&\equiv V_\Box \wedge \big(\Box\Box \mathsf{ff} \vee \Diamond(\Diamond \mathsf{tt} \wedge \eta)\big) \wedge \Box(\Diamond \mathsf{tt} \vee V_\Box) && \text{by 5.2}
\end{aligned}$$

With $\eta \in \mathsf{L}_2$ the above expression is in L_2 and therewith $\psi_\Diamond \in \mathsf{L}_2$, which concludes the proof. □

5.4 Simple games

In Section 3.3 (Nondeterminism) we already presented an algorithm deciding a parity game in quadratic time. Admittedly, the game under consideration was of a rather simple structure, namely a strategy. In the following section we will examine classes of games which are in different ways related to strategies presenting tractable algorithms for deciding the winner from a given position. In the cases where classes are defined by restrictions applied to one of the players we will always choose player $\Box$ to be restricted, tacitly assuming that by restricting player $\Diamond$ the complexity bounds still hold by duality, such that all considered classes are actually closed under complementation.

Algorithmics. All the algorithms we are going to present proceed in the reverse topological order of flowers. So every algorithm will start with a flower decomposition. Using Tarjans algorithm for finding strongly connected components [Tar72] flower decomposition can be performed time linear in the size of the game.

Towards a uniform approach the algorithms share a basic data structure consisting of a labelling of the game graph with triples $(\pi, k, *)$. For some node $v \in V$ the intended meaning of the labelling is as follows.

- π is the set of (pointers to) the predecessors, Ev. This item is initialized during the flower decomposition of the game graph and it is not changed afterwards.
- k is the number of unvisited successors. The same way as π, this item is initialized by the flower decomposition. If e. g. $v \in V_\diamond$ then $v.k$ decreases whenever a successor of v is found to be winning for $\Box$. If instead some successor is found to be winning for $\diamond$ then v is winning for $\diamond$ too and $v.k$ will not be used further. In case that $v.k$ is decreased to 0 this means that all successors are won by the opponent and position v is lost too.
- $*$ is a tag which is void until the algorithm decides whether v belongs to $W_\diamond$ or $W_\Box$, in which case it is accordingly set to $\diamond$ or $\Box$.

Suppose some node v is decided to be in the winning set of player $\diamond$. As mentioned above, this will change the label of every predecessor of v. Moreover, if there are predecessors of v in $V_\diamond$ they are winning for $\diamond$ too and the process reiterates. For a set of nodes which are already decided, given as a stack S, the propagation of decisions from S is handled by the following procedure.

```
procedure Erode( S )
  while S ≠ ∅ do
      v := pop(S)
    for each w ∈ v.π with w.* = void do
      w.k := w.k − 1
      case v.*
      ◇:  if w ∈ V◇  or  w.k = 0 then
            w.* := ◇
            push(S, v)
          fi
      □:  if w ∈ V□  or  w.k = 0 then
            w.* := □
            push(S, v)
          fi
    repeat
  od
end.
```

Regarding complexity, we can already observe that handling of a node popped from the stack involves a constant number of operations to each of its predecessors.

Please note that this procedure corresponds to the erosion step in the stamping algorithm of Section 3.3 if we let S contain the set of trivial flowers. As the game considered there was a winning strategy for $\Diamond$, all these positions were winning for $\Diamond$. Erosion stopped in front of cycles, when every predecessor of a decided node had at least one successor which was not been decided yet.

5.4.1 Solitaire

The category of solitaire games is defined by restrictions on the moves of one player. We start by considering a most radical restriction.

Definition 5.14. We say a parity game is *(strictly) solitaire* if all positions belong to the same player.

Let $\mathcal{G} = (V, E, \Omega)$ be solitaire. We consider the case when $V = V_\Diamond$. Then the question if player $\Diamond$ wins a game from some v reduces to the problem of checking if there is an even loop reachable from v. Hence the seeds of erosion in this case are the even loops.

Let d be the index of $\mathcal{G}$. For some number $j \leq d$ we denote by $\mathcal{G}_{\geq j}$ the subgame of $\mathcal{G}$ induced by the nodes of priority at least j. The following algorithm decides the winning sets of $\mathcal{G}$.

1. For each even index j from 2 to d perform a flower decomposition of $\mathcal{G}_{\geq j}$.
2. For every flower Q in the decomposition that contains a j-loop mark all the nodes $v \in Q$ as winning for $\Diamond$, $v.* := \Diamond$ and erode $\mathcal{G}$ starting from Q.
3. When all even indices are processed, set $v.* := \Box$ for all unmarked v.

The procedure performs $d/2$ flower decomposition, each of which can be done in linear time. A flower Q in $\mathcal{G}_{\geq j}$ contains a j-loop precisely if it is non-trivial and there is some node $v \in Q$ with $\Omega(v) = j$. This flowers can be found in linear time too. Finally, every node of $\mathcal{G}$ is pushed on the stack at most once. This leads to the following overall complexity.

Lemma 5.15. *Deciding the winning sets for a solitaire game of size n and index d can be done in time $O(dn)$.*

Strategies

In the sense of Remark 3.2 a strategy is a game where one of the players is restricted at any position to at most one move. Considering e. g. $\mathcal{G} = (V, E, \Omega)$ a strategy of player $\Box$, this means $|vE| \leq 1$ for any $v \in V_\Box$. But under these circumstances it makes no difference if player $\Box$ moves from some position in $V_\Box$ or player $\Diamond$. Either way, the outcome is the same, i. e. the same winning sets are determined. Thus, we can view a strategy for some player as a solitaire game by assigning all the nodes to the opponent. This leads to the following result.

Corollary 5.16. *Given a strategy of size n and index d, the set of positions from which it is winning can be decided in time $O(dn)$.*

1-solitaire

In the next steps we are gradually loosening the restrictions applied to player $\Box$. A position in a game is called *terminal* if it has no successors.

Definition 5.17. A parity game is 1-*solitaire* if every position belonging to one of the players has at most one non-terminal successor.

In other words, a game is 1-solitaire if the subgame induced by its non-terminal nodes is solitaire.

This class of games does still not grant too much freedom to player $\Box$. Certainly, if some position $v \in V_\Box$ has a terminal successor $w \in V_\Diamond$, player $\Box$ is free to move from v to w and wins. On the other hand, if none of the non-terminal successors are in $V_\Diamond$ his only hope will be to win via the non-terminal successor. This way, it is very easy for player $\Diamond$ to anticipate the moves of her opponent, such that the game she is playing is almost solitaire.

The following algorithm decides the winning sets of a 1-solitaire game $\mathcal{G}$ (restricted for $\Box$) by starting erosion at the terminal nodes and then solving the remaining game, which turns out to be a strategy, i. e. solitaire.

1. Perform a flower decomposition of $\mathcal{G}$. During this operation the labeling with π and k is done. Create an empty stack S.
2. For each $v \in V$ with $v.k = 0$, i e. terminal position, if $v \in V_\Box$ set $v.* := \Diamond$, otherwise $v.* := \Box$. Either way push v onto S.
3. Call $\texttt{Erode}(S)$ and let $\mathcal{G}'$ be the subgame of $\mathcal{G}$ induced by the positions that remain unmarked.
4. Apply the algorithm for solving strategies to $\mathcal{G}'$.

There may be no doubt that the algorithm works correctly while marking the wreath of terminal nodes and performing erosion starting from there. As every terminal node is marked from the beginning any position of $V_\Box$ appearing in $\mathcal{G}'$ will have at most one successor in $\mathcal{G}'$. So $\mathcal{G}'$ forms a strategy for $\Box$. The only way to mark some node v of $\mathcal{G}'$ incorrectly would be if the player owning v could win from v by leaving the $\mathcal{G}'$ but he loses when he stays in $\mathcal{G}'$ - which is what the algorithm for solving strategies assumes. But this cannot happen, as v would have been eroded in the beginning.

Obviously, 1-solitaire games have still the same complexity as pure solitaire games.

Lemma 5.18. *Deciding the winning sets for a 1-solitaire game of size n and index d can be done in time $O(dn)$.*

Nested solitaire

The following lemma states that the solutions of rooted subgames are partial solutions of the entire game.

Lemma 5.19. *Let v be a position in a parity game $\mathcal{G} = (V, E, \Omega)$. Player $\Diamond$ wins from some position v_0 in $\mathcal{G}\restriction_v$ iff she wins from v_0 in $\mathcal{G}$.*

Proof. The assertion follows immediately from the observation that once a play enters a subgame it can never leave it, i. e. for the set of Q of positions reachable from v, there is no edge $(w, w') \in E$ with $w \in Q$ and $w \in V \setminus Q$. $\square$

Let v be a node in $\mathcal{G}$ and w some successor of v. Then the subgame $\mathcal{G}_w$ is obviously an induced subgame of $\mathcal{G}_v$. The games rooted at v and w coincide iff v and w belong to the same flower in $\mathcal{G}$. Algorithmically, strict subgames are favorable, as they allow to calculate partial solutions. If w is a terminal node, it is also a trivial flower and $\mathcal{G}\restriction_w$ is a *strict* subgame of $\mathcal{G}v$.

The algorithm for solving 1-solitaire games relies on the fact that every node $v \in V_\Box$ has at most one successor which does not root a strict subgame. When flowers are processed in reverse topological order all this strict subgames are already decided when v is processed. We can generalize the scheme of 1-solitaire as follows.

Definition 5.20. A parity game is nested solitaire if every subgame induced by its flowers is solitaire.

Please note that the restriction is not applied globally to one of the players, but locally to some player in each flower. Assume player $\Diamond$ is restricted in some non-trivial flower Q of a nested solitaire game. Then the above definition claims that for every position $v \in V_\Diamond$ there is at most one successor which does not root a strict subgame of $\mathcal{G}_v$.

To decide the winning set of such games we can proceed bottom-up the topological order. This way, whenever a node of the player restricted in the current flower is processed, it has at most one unmarked successor and hence the condition of solitaire games is given for every flower guaranteeing correctness of the algorithm. The procedure is the following.

- Let $(Q_1, \ldots, Q_l)$ be a flower decomposition of $\mathcal{G}$. During the decomposition perform the labelling with π and k and set up a table pointing to the restricted player of each flower. Create an empty stack S.
- Process the flowers Q_i starting with $i = l$.
- Solve $\mathcal{G}\restriction_{Q_i}$ as a solitaire game for the locally non-restricted player.
- Call `Erode`(S).
- Proceed with $i = i - 1$.

To see that the algorithm works correctly, we proceed by induction over the flower index. The first flower to be processed is certainly a solitaire, since no node of the restricted player has more than one successor in the same flower. When a flower is solved, all the decisions which have not been propagated yet are on the stack. Assume player $\Box$ is restricted in the next flower. Then the execution of `Erode` will leave undecided at most one successor for each node $v \in V_\Box$ in the next flower. So the condition of solitaire games is propagated as well.

We can view the above algorithm as a linear time reduction from nested to pure solitaire games.

Lemma 5.21. *Deciding the winning sets for a nested solitaire game of size n and index d can be done in time $O(dn)$.*

5.4.2 Flat flowers

The results of the foregoing subsection lead to the intuition that games are simple if they are build of simple flowers. We will now release the restrictions imposed to one of the players, requiring instead a simple flower structure. Please recall that any play finally stays in some flower. Therefore it might be easy to decide games where all flowers are either good or bad for one of the players.

Definition 5.22. A parity game is *flat* if all its flowers are either even or odd.

Let us look to plays of flat games. Once a play reaches a node of some flower Q there are at most two ways to proceed. Either it leaves the flower Q and reaches another flower, which comes after Q in the topological order or it stays in Q forever. In the latter case, as the flower is flat its parity determines the winner. Particularly, if Q has no successors, this means that the play is already decided. Thus, we can iteratively solve the game by processing the flowers in reverse topological order, first propagating the decisions of the successor flowers and then deciding the remaining nodes according to the parity of the current flower.

1. Let $(Q_1, \dots, Q_l)$ be a flower decomposition of $\mathcal{G}$. During the decomposition perform the labelling with π and k and set up a table containing the parity of each flower. Create an empty stack S.

2. For each $v \in V$ with $v.k = 0$, i e. terminal position, if $v \in V_\Box$ set $v.* := \Diamond$, otherwise $v.* := \Box$. Either way, push v onto S.

3. Process the flowers Q_i starting with $i = l$ as follows.

   - `Erode`(S).
   - In case Q_i is not completely eroded assign the winner of every unmarked position v according to the parity of Q_i:
     - if Q_i is a even flower, set $v.* := \Diamond$,

   - otherwise, if Q_i is odd, set $v.* := \Box$;
   - finally, push v onto S.

4. Proceed with $i := i - 1$ until $i = 0$.

In this algorithm each node is pushed into the stack precisely once. Recalling that in `Erode` the handling of a node popped from the stack needs constant time for each of its predecessors we can conclude that this procedure works in linear time.

Proposition 5.23. *Deciding the winning sets for a flat game is possible in linear time.*

The above algorithm goes back to Dowling and Gallier [DG84] who developed it for testing satisfiability of propositional Horn formulae. It is employed by Bernholz, Vardi, and Wolper in [BVW94] for checking the 1-letter nonemptiness problem for weak alternating word automata to which model checking for alternation-free $\mathsf{L}\mu$ is reduced.

5.5 Putting it all together

In the following section we will prove the assertions of Proposition 5.12 by showing that the model checking problems for formulae in the considered fragments reduce to simple games.

The alternation-free fragment of $\mu\mathsf{GF}$ can also be characterized as the set of formulae ψ with the property that for any subformula φ of ψ all fixed point variables free in φ are of the same kind.

Proposition 5.24. *Given an alternation-free formula $\psi \in \mu\mathsf{GF}$ the game $\mathcal{G}(\mathfrak{A}, \psi)$ is flat for any structure $\mathfrak{A}$.*

Proof. Towards a contradiction, assume that in $\mathcal{G}(\mathfrak{A}, \psi)$ there is some flower Q containing both odd and even loops. Let Y and Z be the least GFP-variable respectively LFP-variable (w. r. t. the dependency order) seen on some even respectively odd loop in Q. As Q and Z belong to the same flower there is some play in $\mathcal{G}$ with $\{Y, Z\} \subseteq \mathrm{Inf}(\pi)$. By the argument of Lemma 4.18 it follows that either $Y \in D_\psi(Z)$ or $Z \in D_\psi(Y)$. But this implies that ψ has a subformula with free variables of different kind, which is a contradiction. $\Box$

As shown in the previous section, flat games can be solved in time linear in their size. Since the model checking game for a structure $\mathfrak{A}$ of size n and a formula φ of length n is bound by mn in size, we obtain a time complexity bound of O(mn) for checking the Alternation-free $\mu\mathsf{GF}$, which proves the first part of Proposition 5.12.

By definition of the model checking games, a formula results in a purely solitaire game if it does not contain conjunctions and universal quantification.

Further, the way the edge relation is defined, only atomic contexts have no successors in the model checking game, regardless of the structure to be checked. Consequently, 1-solitaire games reflect the fragment consisting of formulae in which conjunctions are allowed only if one conjunct is atomic and universal quantification $\forall \boldsymbol{y}(\alpha \to \eta)$ is allowed only when η is atomic, i. e. the fragment $\mu\mathsf{GF}_1$. For some structure $\mathfrak{A}$ of size n and a formula $\psi \in \mu\mathsf{GF}_1$ of length m and alternation depth d the model checking game has size mn and index $d+1$. As this game is solitaire it can be solved in time $O(mnd)$, according to Lemma 5.18. This proves the first part of Proposition 5.12 (2).

Turning our attention to nested solitaire games, let us first observe the following.

Lemma 5.25. *In a model checking game $\mathcal{G}(\mathfrak{A}, \psi) = (V, E, \Omega)$ the subgames rooted in adjacent nodes v and $w \in vE$ coincide only if the formula in w has a free fixed point variable.*

Proof. If the subgames $\mathcal{G}_v$ and $\mathcal{G}_w$ coincide the node v is reachable from w. As w is a successor of v there must be loop containing both v and w. Let $\varphi_0, \ldots, \varphi_{k-1}, \varphi_k$ be the sequence of subformulae seen on this loop where $\varphi_k = \varphi_0$ is the subformula of the context w.

By definition of the game, the size of the subformulae always decreases along edges except when some fixed point predicate is met. As $\varphi_0 = \varphi_k$ there must be such a predicate among $\varphi_0 \ldots \varphi_k$. Let $Z\boldsymbol{x}$ be the fixed point predicate met on the loop after φ_0. Then Z is free in φ_0. □

The above lemma immediately leads to the following.

Proposition 5.26. *For any $\psi \in \mu\mathsf{GF}_2$ and every structure $\mathfrak{A}$ the model checking game $\mathcal{G}(\mathfrak{A}, \psi)$ is nested solitaire.*

Therefrom we can infer that the corresponding model checking problem for a can be solved in time $O(mnd)$, with $m = |\mathfrak{A}|$, $n = |\psi|$ and $d = \mathrm{ad}(\psi)$, concluding the proof of Proposition 5.12.

Bibliography

[And92] H. R. Andersen. Model checking and Boolean graphs. In Bernd Krieg-Brückner, editor, *ESOP'92, 4th European Symposium on Programming*, volume 582 of *Lecture Notes in Computer Science*, pages 1–19, Rennes, France, 26–28 February 1992. Springer.

[AvBN98] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27:217–274, 1998.

[BC96] G. Bhat and R. Cleaveland. Efficient local model-checking for fragments of the modal μ-calculus. *Lecture Notes in Computer Science*, 1055:107–126, 1996.

[BVW94] O. Bernholtz, M.Y. Vardi, and P. Wolper. An automata–theoretic approach to branching–time model checking. In D.L. Dill, editor, *Proceedings of the 6th International Conference on Computer–Aided Verification (CAV'94)*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155. Springer–Verlag, 1994.

[Cle90] R. Cleaveland. Tableau-based model checking in the propositional mu-calculus. *Acta Informatica*, 27(8):725–748, 1990.

[CS92] R. Cleaveland and B. Steffen. A linear–time model–checking algorithm for the alternation–free modal mu–calculus. In Kim G. Larsen and Arne Skou, editors, *Proceedings of Computer Aided Verification (CAV '91)*, volume 575 of *LNCS*, pages 48–58, Berlin, Germany, July 1992. Springer.

[DG84] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *LOGIC PROGRAM. (USA) ISSN: 0743-1066*, 1(3):267–84, October 1984.

[Dzi96] S. Dziembowski. The fixpoint bounded-variable queries are PSPACE-complete. In *Computer Science Logic '96, Utrecht*, volume 1258 of *LNCS*, pages 89–105, Berlin, Germany, 1996. Springer.

[EF95] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1995.

[EJ91] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science*, pages 368–377, San Juan, Puerto Rico, 1–4 October 1991. IEEE.

[EJS93] A Emerson, C. Jutla, and A. P. Sistla. Model checking in fragments of mu-calculus. In *International Conference on Computer Aided Verification*, volume 697 of *LNCS*, pages 385–396, Berlin, Germany, June 1993. Springer.

[EL86] E.A. Emerson and C.L. Lei. Efficient model checking in fragments of the propositional μ–calculus. In *Symposion on Logic in Computer Science*, pages 267–278, Washington, D.C., USA, June 1986. IEEE Computer Society Press.

[GGV] G. Gottlob, E. Grädel, and H. Veith. Linear Time Datalog. submitted.

[GHO00] E. Grädel, C. Hirsch, and M. Otto. Back and Forth Between Guarded and Modal Logics. In *Proceedings of 15th IEEE Symposium on Logic in Computer Science LICS 2000*, 2000. To appear.

[GLV97] G. Gottlob, N. Leone, and H. Veith. Succintness as a source of complexity in logical formalisms. Technical Report DBAI-TR-97-13, TU Wien, November 1997.

[Grä] E. Grädel. On the restraining power of guards. (24 pages), to appear in Journal of Symbolic Logic.

[Grä99] E. Grädel. Decision procedures for guarded logics. In *Automated Deduction - CADE16. Proceedings of 16th International Conference on Automated Deduction, Trento, 1999*, volume 1632 of *LNCS*. Springer-Verlag, 1999.

[GW99] E. Grädel and I. Walukiewicz. Guarded Fixed Point Logic. In *Proceedings of 14th IEEE Symposium on Logic in Computer Science LICS '99, Trento*, 1999.

[Her89] B. Herwig. Zur Modelltheorie von $\mathsf{L}\mu$. Dissertation, Universität Freiburg, 1989.

[HT] C. Hirsch and S. Tobies. Tableaux algorithms and finite model property for guarded logics. Communication at the Aachen-Freiburg-Mainz Seminar, May 2000.

[Jur98] M. Jurdziński. Deciding the winner in parity games is in UP intersect co-UP. *IPL: Information Processing Letters*, 68, 1998.

[Jur00] M. Jurdziński. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Pro-*

ceedings, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301, Lille, France, February 2000. Springer.

[Kla90] N. Klarlund. Progress measures and finite arguments for infinite computations. Technical Report 90-1153, Department of Computer Science, Cornell University, Ithaca, New York, September 1990.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, December 1983.

[Mos84] A. W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In Andrzej Skowron, editor, *Proceedings of the 5th Symposium on Computation Theory*, volume 208 of *LNCS*, pages 157–168, Zaborów, Poland, December 1984. Springer.

[Mos91] A. W. Mostowski. Games with forbidden positions. Technical Report 78, University of Gdańsk, 1991.

[SE89] R. S. Streett and E. A. Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81(3):249–264, June 1989.

[Sei] H. Seidl. Model-checking for L_2. Unpublished.

[Sti95] C. Stirling. Local model checking games. In Insup Lee and Scott A. Smolka, editors, *Proceedings of the 6th International Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *LNCS*, pages 1–11, Berlin, GER, August 1995. Springer.

[SW91] C. Stirling and D. Walker. Local model checking in the modal mu–calculus. *Theoretical Computer Science*, 89(1):161–177, 1991.

[Tar72] R. E. Tarjan. Deep-first search and linear graph algorithms. *SIAM J. Computing*, 1:146–160, 1972.

[Var95] M. Y. Vardi. On the complexity of bounded-variable queries (extended abstract). In *Proceedings of the 14th ACM Symposium on Principles of Database Systems*, pages 266–276. ACM, 1995.

[Wal96] I. Walukiewicz. Pushdown processes: games and model checking. In *CAV '96*, volume 1102 of *LNCS*, pages 62–74. Springer, 1996.