Games and Model Checking for Guarded Logics

Dietmar Berwanger and Erich Grädel

Mathematische Grundlagen der Informatik
RWTH Aachen

Abstract. We investigate the model checking problems for guarded first-order and fixed point logics by reducing them to parity games. This approach is known to provide good results for the modal μ-calculus and is very closely related to automata-based methods. To obtain good results also for guarded logics, optimized constructions of games have to be provided.
Further, we study the structure of parity games, isolate 'easy' cases that admit efficient algorithmic solutions, and determine their relationship to specific fragments of guarded fixed point logics.

1 Introduction

Guarded logics are fragments of first-order logic, second-order logic, or fixed point logics defined by restricting quantification so that, semantically speaking, each subformula can simultaneously refer only to elements that are 'very close together' or 'guarded'. The main motivation for the investigation of guarded logics was to explain the good algorithmic and model-theoretic properties of propositional *modal* logics (in a broad sense, including verification logics like CTL and the modal μ-calculus) and to generalize them to a richer setting. The goal was to define natural and expressive logics that could be used on relational structures of arbitrary vocabulary and still would retain convenient features of modal logic, such as the characterization via an appropriate notion of bisimulation, the applicability of automata-based methods, and the good balance between expressiveness and algorithmic manageability (see [13]).

Syntactically, guarded logics are based on a restriction of first-order quantification to the form $\exists \boldsymbol{y}(\alpha(\boldsymbol{x},\boldsymbol{y}) \wedge \psi(\boldsymbol{x},\boldsymbol{y}))$ or $\forall \boldsymbol{y}(\alpha(\boldsymbol{x},\boldsymbol{y}) \rightarrow \psi(\boldsymbol{x},\boldsymbol{y}))$ where quantifiers may range over a tuple $\boldsymbol{y}$ of variables, but are 'guarded' by a formula α that must contain all the free variables of the formula ψ that is quantified over. The guard formulae are of a simple syntactic form (in the basic version, they are just atoms). Depending on the conditions imposed on guard formulae, one has logics with different levels of 'guardedness'. In this paper we consider guarded fragments of first-order logic and least fixed point logic with two notions of guardedness.

While model-theoretic properties and satisfiability algorithms for guarded logics have already been studied rather extensively (see, e.g., [1, 8, 11, 12, 15]), the model checking problem has not yet received as much attention. In [9] a guarded variant of Datalog, called Datalog LITE, has been introduced which is shown to admit efficient query evaluation (linear time in the query length and the size of the database). Datalog LITE is equivalent, via efficient translations, to the alternation-free portion of the guarded fixed point logic μGF so we have efficient model checking for an interesting part of μGF. Guarded logics have also been related to the complexity of query evaluation, in particular for conjunctive queries (see [7, 10]). However, a systematic and comprehensive study of the complexity of model checking problems for guarded logics has not been done yet. In this paper, we attack this problem by developing the game approach to model checking for these logics.

It is well-known that model checking problems for almost any logic can be cast as strategy problems for the appropriate evaluation games (also called Hintikka games). That is, a sentence ψ is true in a structure $\mathfrak{A}$ if and only if Verifier (alias Player 0, alias Eloise) has a winning strategy in the associated Hintikka game $\mathcal{G}(\mathfrak{A}, \psi)$. For *first-order logic*, evaluation games are well-founded (i.e., all plays are finite) and the strategy problem can be solved in linear time in the size of the game. For fixed point logics, the appropriate evaluation games are *parity games*. These are infinite games where each position is assigned a natural number, called its priority, and the winner of an infinite play is determined according to whether the least priority seen infinitely often during the play is even or odd. It is open whether winning sets and winning strategies for parity games can be computed in polynomial time. The best algorithms known today are polynomial in the size of the game, but exponential with respect to the number of priorities. Competitive model checking algorithms for the modal μ-calculus work by solving the strategy problem for the associated parity game (see e.g. [18]).

The reason for the good model checking properties of *guarded logics* is that the associated evaluation games remain small. Indeed, guarded quantification limits the number of possible moves in the evaluation games and thus leads to smaller game graphs. We analyze the translation from guarded formulae to evaluation games and determine the complexity of the resulting games in terms of structural parameters of formulae and input structures. As a consequence *guarded fixed point logics* admit efficient model checking if and only if parity games can be efficiently solved. While we do not know whether this is possible in general, we analyze the structure of parity games and isolate 'easy' cases that admit efficient solutions. With this analysis, we also try to make precise some of the game theoretic intuitions that underly algorithmic approaches to automata and model checking problems. We link these 'easy games' to logic and thus obtain efficient model checking algorithms for fragments of guarded fixed point logic.

2 Guarded logics

Definition 1. The *guarded fragment* GF of first-order logic is defined inductively as the closure of atomic formulae (in a relational vocabulary, with equality) under Boolean connectives and the following quantification rule: For every formula $\psi(\boldsymbol{x}, \boldsymbol{y}) \in \mathrm{GF}$ and every atom $\alpha(\boldsymbol{x}, \boldsymbol{y})$ such that $\{y : y \text{ in } \boldsymbol{y}\} \cup \mathrm{free}(\psi) \subseteq$

free(α), the formulae $\exists \boldsymbol{y}(\alpha(\boldsymbol{x},\boldsymbol{y}) \wedge \psi(\boldsymbol{x},\boldsymbol{y}))$ and $\forall \boldsymbol{y}(\alpha(\boldsymbol{x},\boldsymbol{y}) \rightarrow \psi(\boldsymbol{x},\boldsymbol{y}))$ also belong to GF. The semantics is the usual one for first-order logic.

Here, free(ψ) means the set of free variables of ψ. An atom $\alpha(\boldsymbol{x},\boldsymbol{y})$ that relativizes a quantifier as in the quantification rule for GF is the *guard* of the quantifier. We sometimes use the notation $(\exists \boldsymbol{y} \,.\, \alpha)\psi$ and $(\forall \boldsymbol{y} \,.\, \alpha)\psi$ for guarded formulae.

Guarded fixed point logic. The natural fixed point extension of GF is μGF and was introduced in [15]. It relates to GF in the same way as the modal μ-calculus relates to propositional modal logic and as least fixed point logic LFP (popular in finite model theory) relates to first-order logic FO.

Definition 2. μGF extends GF by the following rules for constructing fixed point formulae: Let T be a k-ary relation symbol, $\boldsymbol{x} = x_1, \ldots, x_k$ a k-tuple of distinct variables, and $\psi(T,\boldsymbol{x})$ a formula that contains only positive occurrences of T, no free first-order variables other than $x_1, \ldots, x_k$, and where T is not used in guards. Then we can build the formulae

$$[\mathrm{LFP}\ T\boldsymbol{x} \,.\, \psi](\boldsymbol{x}) \qquad \text{and} \qquad [\mathrm{GFP}\ T\boldsymbol{x} \,.\, \psi](\boldsymbol{x}).$$

The semantics of the fixed point formulae is the usual one: Given a structure $\mathfrak{A}$ providing interpretations for all free second-order variables in ψ, except T, we have $\mathfrak{A} \models [\mathrm{LFP}\ T\boldsymbol{x} \,.\, \psi(T,\boldsymbol{x})](\boldsymbol{a})$ iff $\boldsymbol{a}$ is contained in the least fixed point of the monotone operator mapping each $T \subseteq A^k$ to $\psi^{\mathfrak{A}}(T) := \{\boldsymbol{a} \in A^k : \mathfrak{A} \models \psi(T,\boldsymbol{a})\}$, and similarly for GFP and greatest fixed points.

It is clear that μGF generalizes the modal μ-calculus L_μ and also the μ-calculus with inverse modalities. Hence the algorithmic problems for μGF, even for formulae with two variables, are at least as hard as for L_μ.

Note that there are two important syntactic restrictions for μGF: Fixed point variables may not be used in guards and fixed point formulae may not contain parameters, i.e., other free variables than those used for constructing the fixed point. These restrictions on μGF are essential. Relaxing either of them would make the logic lose its desirable properties (decidability, closure under guarded bisimulation, tree model property). Hence μGF is contained in the *parameter-free fragment* of the least fixed point logic, which behaves somewhat differently than full LFP. For instance, while the expression complexity and combined complexity of LFP formulae of bounded width (even for width 2) is PSPACE-complete if parameters are allowed, the same problems for parameter-free formulae can be solved in NP $\cap$ co-NP (see [4, 14, 22]).

Clique guarded logics. In GF and μGF only atomic formulae can be used as guards. For some applications this is too restrictive (for instance, temporal operators like **until** cannot be expressed). There exist more general notions of guardedness that lead to more expressive guarded logics but preserve most of their desirable model-theoretic and algorithmic properties. The most powerful and arguably also the most natural of these extensions are the clique guarded logics.

Definition 3. Let $\mathfrak{A} = (A, R_1, \dots, R_m)$ be a relational structure. A set $X \subseteq B$ is *guarded* in $\mathfrak{A}$ if $X = \{a\}$ or if there is a tuple $\boldsymbol{a} \in R_i$ (for some $i \leq m$) such that $X = \{a : a \text{ in } \boldsymbol{a}\}$. A set $X \subseteq A$ is *clique guarded* in $\mathfrak{A}$ if for any two elements a, a' of X there exists a guarded set containing both a and a'. (To put it differently, X induces a clique in the Gaifman graph of $\mathfrak{A}$). A tuple $\boldsymbol{a} \in A^k$ is (clique) guarded if $\boldsymbol{a} \in X^k$ for some clique guarded set X.

Note that for each finite vocabulary τ and each $k \in \mathbb{N}$, there is a positive, existential first-order formula $\mathit{clique}(x_1, \dots, x_k)$ such that, for every τ-structure $\mathfrak{A}$ and every k-tuple $\boldsymbol{a} \in A^k$, $\mathfrak{A} \models \mathit{clique}(\boldsymbol{a}) \iff \boldsymbol{a}$ is clique guarded in $\mathfrak{A}$.

Definition 4. The *clique guarded fragment* CGF of first-order logic and the *clique guarded fixed point logic* μCGF are defined in the same way as GF and μGF, but with the *clique*-formulae as guards. Hence, the quantification rule for CGF and μCGF is the following: If $\psi(\boldsymbol{x}, \boldsymbol{y})$ is a formula of CFG or μCGF, then

$$\exists \boldsymbol{y}(\mathit{clique}(\boldsymbol{x}, \boldsymbol{y}) \land \psi(\boldsymbol{x}, \boldsymbol{y})) \quad \text{and} \quad \forall \boldsymbol{y}(\mathit{clique}(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \psi(\boldsymbol{x}, \boldsymbol{y}))$$

belong to CGF, provided that $\mathrm{free}(\psi) \cup \{y : y \text{ in } \boldsymbol{y}\} \subseteq \mathrm{free}(\mathit{clique})$.

In practice, one will not want to spell out the clique-formulae explicitly. One possibility is not to write them down at all, i.e., to take the usual (unguarded) first-order syntax and to change the semantics of quantifiers so that only clique guarded tuples are considered. Another common option is to use as guards any formula implying a clique formula, i.e., any of the form $\gamma(\boldsymbol{x}, \boldsymbol{y}) := \exists \boldsymbol{z} \beta(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ where β is a conjunction of atoms such that each pair of variables from $\mathrm{free}(\gamma)$ occurs together in at least one conjunct of β. As we want our complexity results to be as powerful as possible, we do not take into account the length of clique guards at all. See [12] for background on clique guarded logics and their relations to other notions such as the *loosely guarded* or *packed* fragments.

Normal forms and alternation depth. We will always assume that fixed point formulae are in *negation normal form*, i.e., that negations apply to atoms only and that formulae are *well-named*, i.e., every fixed point variable is bound only once and the free second-order variables are distinct from fixed point variables. We write $D_\psi(T)$ for the unique subformula in ψ of form $[\mathrm{FP}\ T\boldsymbol{x} \,.\, \varphi(T, \boldsymbol{x})]$ (here and in the following FP means either LFP or GFP). For technical reasons, we finally assume that each fixed point variable T occurs in $D_\psi(T)$ only inside the scope of a quantifier. This is a common assumption that does not affect the expressive power.

We say that T depends on T', if it occurs free in $D_\psi(T')$. The transitive closure of this dependency relation is called the *dependency order*, denoted by $\sqsubset_\psi$. The *alternation level* $\mathrm{al}_\psi(T)$ of T in ψ is the maximal number of alternations between least and greatest fixed point variables on the $\sqsubset_\psi$-paths descending from T. The *alternation depth* $\mathrm{ad}(\psi)$ of a fixed point sentence ψ is the maximal alternation level of its fixed point variables.

3 Evaluation games for guarded logics

Parity games are two-player games of possibly infinite duration. We describe a parity game formally by a labelled graph $\mathcal{G} = (V, V_0, E, \Omega)$ over a finite set of *positions* V with a designated subset V_0, an irreflexive edge relation E representing the possible *moves*, and a labelling function $\Omega : V \to \mathbb{N}$ that assigns to each position a *priority*. The number of different priorities used in the game is the *index* of $\mathcal{G}$.

A *play* of $\mathcal{G}$ is a path $v_0, v_1, \ldots$ formed by the two players starting from a given position v_0. If the current position v belongs to V_0, Player 0 chooses a move $(v, w) \in E$ and the play proceeds from w. Otherwise, her opponent, Player 1, chooses the move. When no moves are available at the current position, the player who has to choose loses. In case this never occurs the play goes on infinitely and the winner is established by looking at the sequence $\Omega(v_0), \Omega(v_1), \ldots$ If the least priority appearing infinitely often in this sequence is even, Player 0 wins the play, otherwise Player 1 wins.

Let $V_1 := V \setminus V_0$ be the set of positions where Player 1 moves. A *positional strategy* for Player i in $\mathcal{G}$ is a function $f : V_i \to V$ which indicates a choice $(v, f(v)) \in E$ for every position $v \in V_i$. (It is called positional, because it does not depend on the history of the play, but only on the current position.) A strategy for a player is *winning* from position v_0 if the indicated choices allow him to win every play starting from v_0. We say a strategy is winning on a set W if it is winning from each position in W. The Forgetful Determinacy Theorem for parity games [5] states that these games are always determined (i.e. from each position one of the players has a winning strategy) and in fact, positional strategies always suffice.

Theorem 1 (Forgetful Determinacy). *In any parity game the set of positions can be partitioned into two sets W_0 and W_1 such that Player 0 has a positional winning strategy on W_0 and Player 1 has a positional winning strategy on W_1.*

We call W_0 and W_1 the *winning sets* of Player 0 and, respectively, Player 1 and the pair (W_0, W_1) the *winning partition* or *solution* of $\mathcal{G}$. Since positional strategies are small objects and since it can be efficiently checked whether a strategy is winning, it can be decided in NP $\cap$ co-NP whether a given position in a parity games is a winning position for Player 0. In fact, it is known [17] that the problem is in UP $\cap$ co-UP. The best known deterministic algorithms to compute winning partitions of parity games have running times that are polynomial with respect to the size of the game graph, but exponential with respect to the index of the game [18].

Theorem 2. *The winning partition of a parity game $\mathcal{G} = (V, V_0, E, \Omega)$ of index d can be computed in space $O(d \cdot |E|)$ and time*

$$O\left(d \cdot |E| \cdot \left(\frac{|V|}{\lfloor d/2 \rfloor}\right)^{\lfloor d/2 \rfloor}\right).$$

Game semantics for fixed point formulae. Consider a finite structure $\mathfrak{A}$ and a guarded fixed point sentence ψ which we assume to be well-named and in negation normal form.

The model checking game $\mathcal{G}(\mathcal{A}, \psi)$ is a parity game whose positions are pairs (φ, ρ) such that φ is a subformula of ψ, and ρ is an assignment from the free first-order variables of φ to elements of $\mathfrak{A}$. The initial position is the pair $(\psi, \emptyset)$.

Verifier (Player 0) moves at positions associated to disjunctions and to formulae starting with an existential quantifier. ¿From a position $(\varphi \vee \vartheta, \rho)$ she moves to either (φ, ρ) or (ϑ, ρ). ¿From a position $((\exists \boldsymbol{y} \,.\, \alpha)\varphi, \rho)$ Verifier can move to any position (φ, ρ') such that ρ' is the restriction to free(φ) of an assignment $\rho^+ : \mathrm{free}(\alpha) \to A$ with $\rho^+ \supseteq \rho$ and $\mathfrak{A} \models \alpha[\rho^+]$. In addition, Verifier is supposed to move at atomic false positions, i.e., at positions (φ, ρ) such that φ is a literal $x = y$, $x \neq y$, $R\boldsymbol{x}$, or $\neg R\boldsymbol{x}$ (where R is *not* a fixed point variable) and $\mathfrak{A} \models \neg\varphi[\rho]$. However, positions associated with literals do have no successors, so Verifier loses at atomic false positions.

Dually, Falsifier (Player 1) moves at conjunctions and universal quantifications, and loses at atomic true positions. In addition there are positions associated with fixed point formulae and with fixed points atoms. At these positions there is a unique move (by Falsifier, say) to the formula defining the fixed point. For a more formal definition, recall that as ψ is well-named there is for any fixed point variable T in ψ a unique subformula $[\mathrm{FP}\ T\boldsymbol{x} \,.\, \varphi(T, \boldsymbol{x})](\boldsymbol{x})$. ¿From position $([\mathrm{FP}\ T\boldsymbol{x} \,.\, \varphi(T, \boldsymbol{x})](\boldsymbol{x}), \rho)$ Falsifier moves to the pair (φ, ρ), and from $(T\boldsymbol{y}, \rho)$ he moves to the position $(\varphi(T, \boldsymbol{x}), \rho')$ with $\rho'(\boldsymbol{x}) = \rho(\boldsymbol{y})$.

In the simple case where we do not have fixed points (i.e., we deal with formulae from GF or CGF), the game is just the guarded version of the usual Hintikka game for first-order logic. In particular, it is well-founded, and it should be obvious that Verifier has a winning strategy for $\mathcal{G}(\mathfrak{A}, \psi)$ iff $\mathfrak{A} \models \psi$. Next, we consider the case of a formula with just one least fixed point. From a position $([\mathrm{LFP}\ T\boldsymbol{x} \,.\, \varphi(T, \boldsymbol{x})](\boldsymbol{x}), \rho)$ the Verifier tries to establish that $\rho(\boldsymbol{x})$ enters T at stage α of the fixed point induction defined by φ. The game goes to (φ, ρ) and from there, as φ is a guarded first-order formula, Verifier can either win the φ-game in a finite number of steps, or she can force it to a position $(T\boldsymbol{y}, \rho')$ where $\rho'(\boldsymbol{y})$ enters the fixed point at some stage $\beta < \alpha$. The game then resumes at a position associated with φ. As any descending sequence of ordinals is finite, Verifier will win the game in a finite number of steps. If the formula is not true, then Falsifier can either win in a finite number of steps or force the play to go through infinitely many positions of form $T\boldsymbol{y}$. Hence, these positions should be assigned priority 1 (and all other positions higher priorities) so that such a play will be won by Falsifier. For GFP-formulae the situation is reversed. Verifier wants to force an infinite play, going infinitely often through positions $T\boldsymbol{y}$, so GFP-atoms are assigned priority 0.

In the general case, we have a formula ψ with nested least and greatest fixed points and on an infinite play of $\mathcal{G}(\mathfrak{A}, \psi)$ one may see different fixed point variables infinitely often. But then one of these variables is the smallest with respect to the dependency order $\sqsubset_\psi$. It can be shown that $\mathfrak{A} \models \psi$ iff this smallest variable is a GFP-variable (provided players play optimally).

Hence, the priority labelling should assign even priorities to GFP-atoms and odd priorities to LFP-atoms. Further, if $T \sqsubset_\psi T'$ and T, T' are fixed point variables of different kind, then T-atoms should get lower priority than T'-atoms.

As the index of a parity game is the main source of difficulty in computing winning sets, the number of different priorities should be kept as small as possible. We avoid the factor 2 appearing in common constructions of this kind by adjusting the definition of alternation level and alternation depth, setting $\mathrm{al}^*_\psi(T) := \mathrm{al}_\psi(T) + 1$ if $\mathrm{al}_\psi(T)$ is even (odd) and T is an LFP (GFP) variable. In the other cases, $\mathrm{al}^*_\psi(T) = \mathrm{al}_\psi(T)$. Finally let $\mathrm{ad}^*(\psi)$ be the maximal value of $\mathrm{ad}^*_\psi(T)$ for the fixed point variables in ψ.

Definition 5. The priority labelling Ω on positions of $\mathcal{G}(\mathfrak{A}, \psi)$ is defined by

$$\Omega(\psi, \rho) = \begin{cases} \mathrm{al}^*_\psi(T) & \text{if } \varphi = T\boldsymbol{x} \text{ and } T \text{ is a fixed point variable} \\ \mathrm{ad}^*_\psi & \text{otherwise.} \end{cases}$$

This completes the definition of the game $\mathcal{G}(\mathcal{A}, \psi)$. Note that the priority labelling satisfies the properties explained above, and that the index of $\mathcal{G}(\mathcal{A}, \psi)$ is at most $\mathrm{ad}(\psi) + 1$.

The proof that the game is correct is similar to the proofs for model checking games for the μ-calculus (see e.g. [16, 21]). For details taking into account the optimizations made here, see [2].

Proposition 1. *Let ψ be a guarded fixed point sentence from* μGF *or* μCGF*, and let $\mathfrak{A}$ be a relational structure. $\mathfrak{A} \models \psi$ if and only if Player 0 has a winning strategy for the parity game $\mathcal{G}(\mathcal{A}, \psi)$.*

4 The complexity of model checking

The reduction scheme in the previous section provides a model checking technique for guarded fixed point logics: Given $(\mathfrak{A}, \psi)$, construct the corresponding game $\mathcal{G}(\mathfrak{A}, \psi)$ and check whether Player 0 has a winning strategy. The complexity of this algorithm is determined by the complexity of the reduction and the complexity of solving the resulting parity game. In this section we discuss the construction of the game $\mathcal{G}(\mathcal{A}, \psi)$ for different kinds of guarded formulae and determine the size of the game. The general algorithmic results on parity games, yield complexity results for our model checking problems. In the last section, we will then look more closely at the structure of games.

We can measure the complexity in terms of different parameters of the given formula and the given finite structure. Relevant notions, besides formula length, are the closure $\mathrm{cl}(\psi)$ of ψ, which is just the set of its subformulae, and the *width*, which is the maximal number of free variables in subformulae, i.e. $\mathrm{width}(\psi) := \max\{|\mathrm{free}(\varphi)| : \varphi \in \mathrm{cl}(\psi)\}$. The size of a finite relational structure $\mathfrak{A} = (A, R_1, \ldots, R_m)$ is defined as $\|\mathfrak{A}\| = |A| + \sum_{i=1}^{m} r_i |R_i|$ where r_i is the arity of R_i (i.e., the number of structure elements plus the sum of the length of all tuples in its relations).

Model checking GF and μGF. It is obvious that GF generalizes propositional modal logic ML. The model checking problem for ML is PTIME-complete [14] and can be solved in time $O(\|\mathfrak{A}\| \cdot |\psi|)$. Both results extend to GF.

Proposition 2. *Given a sentence $\psi \in \mu$GF and a finite structure $\mathfrak{A}$, the associated parity game $\mathcal{G}(\mathfrak{A}, \psi)$ can be constructed in time $O(\|\mathfrak{A}\| \cdot |\psi|)$.*

Proof. In the construction of the parity game $\mathcal{G}(\mathfrak{A}, \psi) = (V, V_0, E, \Omega)$ we can restrict V to the positions (φ, ρ) that are reachable from the initial position $(\psi, \emptyset)$. The construction of the game is straightforward, we just have to estimate its size. It suffices to prove that $|E| = O(\|\mathfrak{A}\| \cdot |\psi|)$.

Let $H(\psi)$ be the syntax tree of ψ, with back edges from fixed point atoms $T\boldsymbol{x}$ to the defining formula $\varphi(T, \boldsymbol{x})$. Obviously, $H(\psi)$ has less than $|\psi|$ nodes and edges.

We claim that for every edge $\varphi \to \varphi'$ of $H(\psi)$ there exist at most $\|\mathfrak{A}\|$ edges of form $(\varphi, \rho) \to (\varphi', \rho')$ in the game graph $\mathcal{G}(\mathfrak{A}, \psi)$. We consider several cases.

First, let $\varphi = (Q\boldsymbol{x} \,.\, \alpha)\varphi'$. In that case an edge $(\varphi, \rho) \to (\varphi, \rho')$ can exist only if there exists an assignment ρ^+ such that $\mathfrak{A} \models \alpha[\rho^+]$ and ρ, ρ' are the restrictions of ρ^+ to the free variables of φ and φ', respectively. As guards are atomic formulae the number of assignments satisfying a guard is bounded by $\|\mathfrak{A}\|$.

In all other cases, i.e. if φ is not a quantified formula, then for any fixed φ and ρ there are at most two edges $(\varphi, \rho) \to (\varphi', \rho')$. Hence it suffices to show that for each such $\varphi \in H(\psi)$ there exist at most $\|\mathfrak{A}\|$ reachable positions (φ, ρ) in the game graph. Recall that fixed point variables occur inside their defining formulae only under the scope of a quantifier. If φ is not inside a quantifier, only the position $(\varphi, \emptyset)$ is reachable. Otherwise there is a uniquely determined least subformula of ψ that strictly contains φ and has the form $(Q\boldsymbol{y} \,.\, \alpha)\vartheta$. Then a position (φ, ρ) is reachable if and only if (ϑ, ρ) is reachable. Note that ϑ and α are uniquely determined by φ, and the position (ϑ, ρ) is reachable only if $\mathfrak{A} \models \alpha[\rho^+]$ where $\rho^+ \supseteq \rho$. By the same argument as above it follows that the number of reachable positions (φ, ρ) is bound by $\|\mathfrak{A}\|$. This completes the proof.

According to Theorem 2, we obtain the following complexity bounds for model checking μGF via the associated parity game.

Theorem 3. *Given a structure $\mathfrak{A}$ and a μGF-sentence ψ of alternation depth d the model checking problem $\mathfrak{A} \models \psi$ can be solved in space $O(d \cdot \|\mathfrak{A}\| \cdot |\psi|)$ and time*

$$O\left(d^2 \cdot \left(\frac{\|\mathfrak{A}\| \cdot |\psi|}{\lfloor (d+1)/2 \rfloor}\right)^{\lfloor (d+3)/2 \rfloor}\right).$$

It is instructive to compare these results to the case of unguarded formulae. Note that even though there is no explicit restriction on the width (or equivalently, the number of first-order variables) in GF or μGF, the width is implicitly bounded by the arity of the relation symbols. (On graphs, for instance, GF and

μGF are really two-variable logics). For a meaningful comparison of the size of evaluation games and the model checking complexity for GF/μGF with FO/LFP (or with clique guarded logics, see below) one should bound the width of these formulae.

On a coarse level, if one just considers membership and completeness in major complexity classes, then modal logic ML, bounded-variable first-order logics, and GF are on the same level, with model checking problems that are PTIME-complete. For fixed-point logics, a similar picture emerges: L_μ, bounded-width parameter-free LFP, and μGF have model checking problems that are in UP $\cap$ co-UP and hard for PTIME.

However, a more detailed analysis reveals differences that are quite relevant for practical algorithms. Even for bounded-variable fragments FO^k of first-order logic the size of the corresponding model checking game is $O(|A|^k|\psi|)$, as in general, all possible assignments $\rho : \{x_1, \ldots, x_k\} \to A$ need to be taken into account. Hence, the model checking games (and the complexity) of bounded-variable logics are often quite substantially larger than for guarded logics. We will see that the complexity of clique guarded logic is between these two.

Model checking CGF **and** μCGF**.** By the definition of clique guardedness, for a tuple $\boldsymbol{x}$ of variables appearing free in a subformula of ψ, the value of any assignment $\rho(\boldsymbol{x})$ induces a clique in the Gaifman graph of $\mathfrak{A}$. The number and size of cliques in this graph can be bound by parameters derived from its tree decompositions.

Definition 6. A *tree decomposition* of width l of some structure $\mathcal{B}$ is a tree labelled with subsets of at most $l+1$ elements of B, called *blocks*, such that (1) every strictly guarded set in $\mathcal{B}$ is included in some block and (2) for any element $a \in B$ the set of blocks which contain b is connected. The *tree width* of $\mathcal{B}$ is the minimal width of a tree decomposition of $\mathcal{B}$.

Lemma 1. *Given a structure $\mathfrak{A}$ of tree width l, the number of clique guarded assignments in $\mathfrak{A}$ for a tuple of k variables is bounded by $c_k(\mathfrak{A}) := (l+1)^k \cdot |\mathfrak{A}|$.*

Proof. Let T be a tree decomposition of width l of the structure $\mathfrak{A}$ and thus also of its Gaifman graph. A simple graph theoretic argument [12] shows that cliques are not disbanded by tree decompositions, that is, every clique of the Gaifman graph is contained in some decomposition block. Consequently, every clique guarded set in $\mathfrak{A}$, in particular $\rho(\boldsymbol{x})$, is contained in some block of T. Since we can assume without loss that $|T| \le |A|$, the number of clique guarded k-tuples in $\mathfrak{A}$, and with it the number of clique guarded assignments, is bounded by $(l+1)^k \cdot |A|$.

By a similar analysis as in the case of μGF we obtain the following estimates.

Proposition 3. *Given a μCGF-sentence ψ of width k and a finite structure $\mathfrak{A}$ of tree width l, the associated parity game $\mathcal{G}(\mathfrak{A}, \psi)$ can be constructed in time $O(c_k(\mathfrak{A}) \cdot |\mathrm{cl}(\psi)|)$.*

Theorem 4. *For a structure $\mathfrak{A}$ and a* μCGF*sentence ψ of width k and alternation depth d the model checking problem can be solved in space $O(d \cdot c_k(\mathfrak{A}) \cdot |\mathrm{cl}(\psi)|)$ and time*

$$O\left(d^2 \cdot \left(\frac{c_k(\mathfrak{A}) \cdot |\mathrm{cl}(\psi)|}{\lfloor (d+1)/2 \rfloor}\right)^{\lfloor (d+3)/2 \rfloor}\right).$$

For unguarded sentences, corresponding complexity expressions are obtained by replacing $c_k(\mathfrak{A})$ with the number of possible assignments in that case, $|\mathfrak{A}|^k$. If the tree width l of $\mathfrak{A}$ is small compared to $|\mathfrak{A}|$ this value is be much higher than $c_k(\mathfrak{A}) = (l+1)^k \cdot |\mathfrak{A}|$. Especially, for a fixed clique guarded sentence ψ, the size of the game $\mathcal{G}(\mathfrak{A}, \psi)$ for structures $\mathfrak{A}$ of bounded tree width grows linearly with the size of $\mathfrak{A}$, while it grows polynomially with degree k when ψ is unguarded.

However, in the case of unbounded width the model checking problem for CGF and μCGF ar as hard as for FO and LFP. To prove this, one takes input structures with a complete binary guard relation, so that all tuples in the structure become clique guarded. Similar observations apply to expression complexity and to other guarded logics (like loosely guarded or packed fragments).

Proposition 4. *The model checking problems for* CGF *and* μCGF *of unbounded width are* PSPACE*-complete and* EXPTIME*-complete, respectively.*

5 Easy games and tractable fixed point formulae

In the previous section we established complexity results for model checking problems based on complexity bounds for the associated parity games as given by Theorem 2. These complexity bounds take into account worst-case scenarios as, for example, that the underlying game graph is strongly connected and that each player can force the play to reach almost any priority. However, for model checking games resulting from specific classes of formulae or structures such assumptions may be overly pessimistic and better heuristics apply. By looking more closely at the structure of parity games, we can isolate easy cases of games and obtain classes of formulae for which the associated parity games can be solved efficiently, in subquadratic time with regard to their size.

De-tangling the game. An obvious approach to complex problems is by decomposition into independent simpler subproblems. In the context of games subproblems correspond to subgames. A *subgame* of a parity game $\mathcal{G}$ is a game $\mathcal{G}\restriction_U$ obtained by restricting $\mathcal{G}$ to the positions of some subset $U \subseteq V$. The subgame induced by the set of all positions reachable from a node v in $\mathcal{G}$ is called *rooted* at v and denoted by $\mathcal{G}\restriction_v$. We say a subgame $\mathcal{H}$ of $\mathcal{G}$ is *stable*, if the winning set of each player in $\mathcal{H}$ is a subset of its winning set in $\mathcal{G}$.

Clearly, in any game the rooted subgames are stable. Given a parity game $\mathcal{G}$, let $\mathcal{H}$ be a rooted subgame. Note that there may be different positions v, w such that $\mathcal{H} = \mathcal{G}\restriction_v = \mathcal{G}\restriction_w$. In that case we say that v and w are *entangled*. The entanglement relation is an equivalence on game positions. Let us call its classes the *tangles* of $\mathcal{G}$. In particular, we call the class of v with $\mathcal{G}\!\restriction_v = \mathcal{H}$ the *root tangle* of $\mathcal{H}$. Via their root tangle the rooted subgames of $\mathcal{G}$ are in one-to-one correspondence to the tangles of $\mathcal{G}$. We call a tangle that induces a rooted subgame in $\mathcal{G}$ a *leaf* tangle. Observe that the tangles of a game are precisely the strongly connected components of the underlying graph. Over these the reachability relation in $\mathcal{G}$ induces a partial ordering with the root tangle of $\mathcal{G}$ as the greatest element and the leaf tangles as least elements.

Towards a divide-and-conquer approach, we are interested in subproblems which are independent, so that the global solution can easily be recomposed from the partial solutions. Suppose that, in order to solve $\mathcal{G}$, some rooted strict subgame $\mathcal{G}\!\restriction_U$ was already solved. At this point it would not help to solve the subgame induced by the unresolved positions $\mathcal{G}\!\restriction_{V \setminus U}$, which is it not necessarily stable. Instead, we can first propagate the solutions found for U into $V \setminus U$ by including in the winning set W_0 of Player 0 every position from V_0 with some successor already in W_0 and those positions from V_1 with all successors already in W_0; for Player 1 we can proceed dually. Let P be the set of positions assigned to winning sets by iterating the above process. Then the subgame $\mathcal{G}\!\restriction_{V \setminus (U \cup P)}$ induced by the unresolved positions is stable.

This suggests a heuristics for solving a parity game that starts by solving the leaf tangles and then, after propagating their solutions, applies recursively to the subgame induced by the unresolved positions. In a concrete implementation, this heuristics relies on three algorithms.

(1) A *decomposition procedure* which computes and updates the strongly connected components of the game graph to provide a list of its leaf tangles.
(2) A *tangle-solver* for solving the games induced by the leaf tangles.
(3) A *propagator* that evaluates the obtained partial solutions and passes a new stable subgame to the next recursion step.

For an efficient implementation, the game graph can initially be decomposed into its strongly connected components by using Tarjan's algorithm which works in time linear in the number of edges. Subsequently, this decomposition can be updated dynamically by removing a component after each call of the tangle-solver and refining the decomposition order for every batch of positions removed from the original game by the propagator. By adapting an algorithm from [3] the time for this operation can be linearly bounded by the number of removed edges, so the global algorithm will not spend more than linear time in the decomposition procedure. Also the propagator can be written such that it visits every edge of the game graph at most once. Thus, the above heuristics leads to an algorithm which spends only linear time in the decomposition and the propagation procedures.

Proposition 5. *The winning partition of a parity game can be computed efficiently if an efficient tangle-solver is available.*

Well-founded games. A tangle is *trivial*, if it consists of a single position. If all tangles in a game are trivial, the game graph is acyclic and the solution of the leaf tangles propagate all the way up to the root. Such games are called *well-founded*, and it is folklore that they can be solved efficiently.

Proposition 6. *Any well-founded game can be solved in linear time.*

For instance, given a well-founded game $\mathcal{G} = (V, V_0, E)$ we can write in time $O(|E|)$ a propositional Horn formula $\psi_{\mathcal{G}}$ consisting of the clauses $u \leftarrow v$ for all edges $(u, v) \in E$ with $u \in V_0$, and the clauses $u \leftarrow v_1 \wedge \cdots \wedge v_m$ for all nodes $u \in V - V_0$ where $uE = \{v_1, \ldots, v_m\}$. The minimal model of $\psi_{\mathcal{G}}$ is precisely the winning set W_0. In fact for well-founded games, the above heuristics yields a linear time decision procedure which coincides with the known linear time algorithm for solving propositional Horn formulae.

Corollary 1. *For a structure $\mathfrak{A}$ and a* CGF*-sentence ψ of width k, the model checking problem can be solved in time $O(c_k(\mathfrak{A}) \cdot |\mathrm{cl}(\psi)|)$. If $\psi \in$* GF *the problem can be solved in time $O(\|\mathfrak{A}\| \cdot |\psi|)$.*

Dull versus lively games. Given a game, an i-loop is a simple cycle in the game graph with the least occurring priority i. Let the *range* of a tangle T be the set of priorities i for which T contains an i-loop. We say a tangle is *dull* if all priorities in its range are of the same parity. A game is dull if all its tangles are so, otherwise it is called *lively*. Note that whenever a tangle is not trivial, then its range contains the least priority occurring in it. Thus, any nontrivial tangle T which is dull in a game $\mathcal{G}$ can be decided by checking whether the least occurring priority is even. If this is the case, Player 0 wins from each position in $\mathcal{G}\restriction_T$, otherwise he loses. Using this tangle-solver with our heuristics we can solve dull games in linear time.

Theorem 5. *Any dull game $\mathcal{G} = (V, V_0, E, \Omega)$ can be solved in time $O(|V| + |E|)$.*

The problem of establishing the winner of a parity game is algorithmically equivalent to the emptiness problem for nondeterministic parity automata over *trees*. In [20] Kupferman, Vardi, and Wolper show that the latter problem reduces to the 1-letter emptiness problem of alternating automata over *words*. In this framework, dull games correspond to so-called weak automata. Kupferman, Vardi, and Wolper show that the 1-letter emptiness problem for weak alternating word automata can be solved in linear time, thus proving that model checking for alternation-free L_μ-formulae can be performed in linear time. Here, we will generalize this result to alternation-free guarded formulae, i.e., formulae of alternation depth 1. Towards this, let us establish a connection between the alternation depth of a formula and the range of the tangles in the associated game. Let T be a tangle in a model checking game $\mathcal{G}(\mathfrak{A}, \psi)$. We say that a fixed point variable X is *alive* in T if the tangle contains some position $(X\boldsymbol{x}, \beta)$.

Lemma 2. *In any nontrivial tangle of $\mathcal{G}(\mathfrak{A}, \psi)$ the set of alive fixed point variables has a unique minimal element w.r.t. $\sqsubset_\psi$.*

Proof. Consider a play π cycling in T from some position $v = (X\boldsymbol{x}, \beta)$ back to itself. By a straightforward induction, we can verify that for each fixed point variable Y seen along π, $D_\psi(Y)$ is a subformula of $D_\psi(X)$ and X appears free in $D_\psi(Y)$. Hence, $X \sqsubset_\psi Y$ for all fixed point variables Y on π. As T is strongly connected, for any variable $Y \in T$ there is a play cycling through X and Y.

Observe that in a model checking game the range of a tangle T consists of the priorities of all alive fixed point variables. Thus, if all these variables depend on some fixed point variable X, the range of T is included in $\{\Omega(Y) : X \sqsubset_\psi Y\}$. By definition, in an alternation-free formula ψ there are no dependencies between variables of different kind. Accordingly, the priorities associated to any two alive variables are of the same parity. It follows that there are no alternating priorities in the range of any tangle of a model checking game $\mathcal{G}(\mathfrak{A}, \psi)$.

Proposition 7. *Alternation-free fixed point formulae lead to dull games.*

Corollary 2. *For a structure* $\mathfrak{A}$ *and an alternation-free* μCGF*-sentence* ψ *of width* k*, model checking can be performed in time* $O(c_k(\mathfrak{A}) \cdot |\mathrm{cl}(\psi)|)$*. In particular, if* $\psi \in \mu$GF *the problem can be solved in time* $O(\|\mathfrak{A}\| \cdot |\psi|)$*.*

The result for alternation-free μGF has been proved by a different method in [9].

Solitaire games. For arbitrary games, the number of alternating priorities in the range of a tangle or, more generally, its index, is not bounded. To our present knowledge, this index appears as a significant source of complexity in all decision procedures for parity games. Typically, the worst-case running time of such a procedure grows exponentially with (a constant fraction of) the index. One reason for this difficulty may be that even if a player has a winning strategy in a game, he cannot tell in which priority the play will finally be trapped. In general, the opponent may be able to choose at any point between different, though hostile, loops and thus continually trade against the first player's strategy. The costs of a search for a trap where all this trading is ineffective can be quite high in terms of complexity.

In the case of dull games considered above, this difficulty is circumvented by restricting the trading range as much as possible. Thus, each player knows that the only way to win is by attracting its opponent into a loop of the right priority in which the game can be kept infinitely. Another approach to avoid the difficulties caused by trading priorities consists in allowing only one player to move in the cyclic parts of the game.

Definition 7. A game is *solitaire* if all nontrivial moves are performed by the same player. A game is *nested solitaire* if all its tangles are solitaire games.

Notice that a positional strategy can be presented as a solitaire game. In the automata-theoretic view a solitaire game corresponds to a *deterministic* parity tree automaton whose emptiness problem is linear time reducible to the nonemptiness problem of a one-letter nondeterministic parity word automaton. This problem was known to be solvable in time $O((n+m)d)$ for an automaton with n states, m transitions and d even priorities. Recently, King, Kupferman, and Vardi [19] presented an algorithm which solves this problem in time $O((n+m)\log d)$. Via the aforementioned equivalence their result applies to solitaire games also.

Proposition 8. *The winner of a solitaire game* $\mathcal{G} = (V, V_0, E, \Omega)$ *of index* d *can be established in time* $O\big(\log d \cdot (|V| + |E|)\big)$*.*

However, for deciding the winner of a *nested* solitaire game, our heuristics relies on the computation of the entire winning partition of leaf tangles. It is not clear whether the algorithm of King, Kupferman, and Vardi can be extended to a fast solver for games that consist of solitaire tangles but are not solitaire themselves. To solve the leaf tangles of such games we will therefore use another approach.

Let $\mathcal{H}$ be a solitaire game on a tangle with all positions belonging to Player 0. She wins from a position if she can reach an even loop. Let the even priorities in the range of $\mathcal{H}$ be $\{i_1, \ldots, i_d\}$. To determine the winning partition of $\mathcal{H}$ we can proceed as follows.

Decompose $\mathcal{H}$ into d copies $\mathcal{H}_1, \ldots, \mathcal{H}_d$ where $\mathcal{H}_j$ is the subgame of $\mathcal{H}$ induced by the positions of priority at least i_j. Transform $\mathcal{H}_j$ into a dull game by assigning to all positions of priority greater than i_j some higher, odd priority. Now solve every $\mathcal{H}_j$ as a dull game. Note that, if the copy of a position v in $\mathcal{H}_j$ is winning for Player 0, then v is also winning in $\mathcal{H}$. Thus, the union of winning sets in the copies $\mathcal{H}_j$ yields, by propagation, the winning set of Player 0 in $\mathcal{H}$.

This procedure reduces the solution of a solitaire tangle game to the solution of d dull games. By embedding the algorithm into our decomposition and propagation scheme we obtain a solver for nested solitaire games.

Theorem 6. *A nested solitaire parity game $\mathcal{G} = (V, V_0, E, \Omega)$ of index d can be solved in time $O\big(\, d \cdot (|V| + |E|)\,\big)$.*

Solitaire formulae. Given that nested solitaire games can be treated efficiently, the question arises whether these games correspond to a natural fragment of fixed point logic. Note that in a model checking game, Player 0 makes choices at positions corresponding to disjunctions or existential quantifications, whereas Player 1 makes nontrivial choices at conjunctions and universal quantifications. To make sure that all tangles in a game are (nested) solitaire we thus have to restrict the use of one of these pairs of operators. The radical approach would be to remove $\wedge$ and $\forall$ (or, equivalently, $\vee$ and $\exists$). We thus would obtain a fragment whose model checking games are solitaire which, however, is not very expressive. A more liberal approach is to restrict the syntax as follows.

Definition 8. The *solitaire fragment* of μCGF consist of those formulae where negation and universal quantification apply to closed formulae only and conjunctions to pairs of formulae of which at least one is closed.

Recall that a fixed point formula is *closed* if it contains no free fixed point variables. By Lemma 2, positions with closed subformulae are not entangled with any other positions. Consequently, the model checking games of solitaire μCGF-formulae are nested solitaire games. We remark that the solitaire fragment of the modal μ-calculus has already been studied under the name L_2 in [6].

Proposition 9. *The model checking problem for a structure $\mathfrak{A}$ and a solitaire μCGF-sentence ψ of width k and alternation depth d, can be solved in time $O\left(d \cdot c_k(\mathfrak{A}) \cdot |\mathrm{cl}(\psi)|\right)$. In particular, for $\varphi \in \mu$GF the problem can be solved in time $O\left(d \cdot \|\mathfrak{A}\| \cdot |\psi|\right)$.*

References

[1] H. ANDRÉKA, J. VAN BENTHEM, AND I. NÉMETI, *Modal languages and bounded fragments of predicate logic*, Journal of Philosophical Logic, 27 (1998), 217–274.

[2] D. BERWANGER, *Games and model checking for guarded logics.* Diploma thesis, RWTH Aachen, 2000.

[3] R. BLOEM, H. GABOW, AND F. SOMENZI, *An algorithm for strongly connected component analysis in $n \log n$ symbolic steps*, in Formal Methods in Computer Aided Design, LNCS Nr. 1954 (2000), 37–54.

[4] S. DZIEMBOWSKI, *Bounded-variable fixpoint queries are PSPACE-complete*, Computer Science Logic CSL 96. LNCS Nr. 1258 (1996), 89–105.

[5] A. EMERSON AND C. JUTLA, *Tree automata, mu-calculus and determinacy*, in Proc. 32nd IEEE Symp. on Foundations of Computer Science, 1991, 368–377.

[6] A. EMERSON AND C. JUTLA AND A. P. SISTLA, *On model checking for the μ-calculus and its fragments*, Theoretical Computer Science 258 (2001), 491–522.

[7] J. FLUM, M. FRICK, AND M. GROHE, *Query evaluation via tree-decompositions*, in Database Theory – ICDT 2001, LNCS Nr. 1973 (2001), 22–38.

[8] H. GANZINGER, C. MEYER, AND M. VEANES, *The two-variable guarded fragment with transitive relations*, in Proc. 14th IEEE Symp. on Logic in Computer Science, 1999, 24–34.

[9] G. GOTTLOB, E. GRÄDEL, AND H. VEITH, *Datalog LITE: A deductive query language with linear time model checking*, ACM Transactions on Computational Logic, (to appear).

[10] G. GOTTLOB, N. LEONE, AND F. SCARCELLO, *Robbers, marshals, and guards: Game theoretic and logical characterizations of hypertree width*, in Proc. 20th ACM Symp. on Principles of Database Systems, 2001, 195–201.

[11] E. GRÄDEL, *On the restraining power of guards*, Journal of Symbolic Logic, 64 (1999), 1719–1742.

[12] E. GRÄDEL, *Guarded fixed point logics and the monadic theory of countable trees*, to appear in Theoretical Computer Science, (2001).

[13] E. GRÄDEL, *Why are modal logics so robustly decidable?*, in Current Trends in Theoretical Computer Science. Entering the 21st Century, G. Paun, G. Rozenberg, and A. Salomaa, eds., World Scientific, 2001, 393–498.

[14] E. GRÄDEL AND M. OTTO, *On logics with two variables*, Theoretical Computer Science, 224 (1999), 73–113.

[15] E. GRÄDEL AND I. WALUKIEWICZ, *Guarded fixed point logic*, in Proc. 14th IEEE Symp. on Logic in Computer Science, 1999, 45–54.

[16] B. HERWIG, *Zur Modelltheorie von L_μ.* Dissertation, Universität Freiburg, 1989.

[17] M. JURDZINSKI, *Deciding the winner in parity games is in UP $\cap$ Co-UP.*, Information Processing Letters, 68 (1998), 119–124.

[18] M. JURDZIŃSKI, *Small progress measures for solving parity games*, Proceedings of STACS 2000, LNCS Nr. 1770 (2000), 290–301.

[19] V. KING, O. KUPFERMAN, AND M. VARDI, *On the complexity of parity word automata*, in Proceedings of FOSSACS 2001, LNCS Nr. 2030 (2001), 276–286.

[20] O. KUPFERMAN, M. VARDI, AND P. WOLPER, *An automata-theoretic approach to branching-time model checking*, Journal of the ACM, 47 (2000), 312–360.

[21] C. STIRLING, *Bisimulation, model checking and other games.* Notes for the Mathfit instructional meeting on games and computation. Edinburgh, 1997.

[22] M. VARDI, *On the complexity of bounded-variable queries*, in Proc. 14th ACM Symp. on Principles of Database Systems, 1995, 266–267.