Theory Comput. Systems **37**, 675–694 (2004)
DOI: 10.1007/s00224-004-1147-5

Theory of Computing Systems

Fixed-Point Logics and Solitaire Games*

Dietmar Berwanger and Erich Grädel

Mathematical Foundations of Computer Science,
Aachen University of Technology,
D-52056 Aachen, Germany
{berwanger,graedel}@ cs.rwth-aachen.de

Abstract. The model-checking games associated with fixed-point logics are parity games, and it is currently not known whether the strategy problem for parity games can be solved in polynomial time. We study Solitaire-LFP, a fragment of least fixed-point logic, whose evaluation games are nested soltaire games. This means that on each strongly connected component of the game, only one player can make nontrivial moves. Winning sets of nested solitaire games can be computed efficiently.

The model-checking problem for Solitaire-LFP is PSPACE-complete in general and PTIME-complete for formulae of bounded width. On finite structures (but not on infinite ones), Solitaire-LFP is equivalent to transitive closure logic.

We also consider the solitaire fragment of guarded fixed-point logics. Due to the restricted quantification pattern of these logics, the associated games are small and therefore admit more efficient model-checking algorithms.

1. Introduction

Fixed-point logics play an important role in many areas of logic. LFP, the extension of first-order logic by least and greatest fixed points, is of fundamental importance in finite model theory, descriptive complexity, and databases. The modal μ-calculus L_μ is a similar extension of propositional modal logic. It relates to LFP in much the same way as multimodal logic relates to first-order logic. In terms of expressive power, it subsumes a variety of logics used in verification, in particular, LTL, CTL, CTL*, PDL, and also many

* This research has been partially supported by the European Community Research Training Network "Games and Automata for Synthesis and Validation" (GAMES) (Contract HPRN-CT-2002-00283), see www.games.rwth-aachen.de.

logics used in other areas of computer science, for instance, game logic and description logics. Both LFP and L_μ have a rich theory, and are well-behaved in model-theoretic and also, to a large extent, in algorithmic terms.

Nevertheless, there are still important open problems concerning their complexity. The most prominent one is whether the model-checking problem for the modal μ-calculus or for LFP-formulae of bounded width can be solved in polynomial time. This problem is equivalent to an algorithmic problem in the theory of infinite games, namely, the question whether winning sets in *parity games* can be computed in polynomial time. Parity games are two-person games on finite or infinite game graphs that admit infinite plays and where each position is assigned a natural number, called its priority. The winner of an infinite play is determined according to whether the least priority seen infinitely often during the play is even or odd. Parity games arise as the natural model-checking games for fixed-point logics. Priorities of game positions correspond to the alternation level of fixed-point formulae. 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.

In this article we have a closer look at a class of parity games that can be solved efficiently, and at the fixed-point formulae that are associated with them. *Nested solitaire games* are parity games where on each strongly connected component, only one player can make nontrivial choices. As we will show, the winning sets of a nested solitaire game can be computed in time that is linear in the product of the number of priorities with the size of the game graph.

We define Solitaire-LFP, a fragment of least fixed-point logic, whose model-checking games are nested solitaire games, and we analyse the algorithmic properties and the expressive power of this fragment. A corresponding fragment of the modal μ-calculus has already been studied in [8], and has been shown to be equivalent to the logic ECTL*. For Solitaire-LFP, it turns out that the model-checking problem is PSPACE-complete in the general case (for formulae of unbounded width) and PTIME-complete for formulae of bounded width. We further prove that on finite structures, Solitaire-LFP is equivalent to transitive closure logic (TC). To establish this result we exploit the solitaire-structure of the model-checking game and the fact that TC-formulae are equivalent to stratified linear Datalog programs. We construct for every formula in Solitaire-LFP a stratified linear Datalog program which defines the winning positions in the associated model-checking game. A further consequence of this proof is that every formula in Solitaire-LFP of width k (of arbitrary alternation level) is equivalent, on finite structures, to an alternation-free fixed-point formula of width at most $2k$.

The tractability of model checking via games depends not only on the complexity of solving the games but also on their size. Typically, the evaluation game for a formula contains distinct positions for every possible variable assignment to each of its subformulae. As a consequence, the size of a game associated to a formula ψ of width k and a structure with n elements may be of the order $|\psi| \cdot n^k$. In applications where the structures are big, this is inconvenient. One is therefore interested in fragments of fixed-point logics for which the size of the games is considerably smaller, ideally linear in both the size of the formula and the size of the structure.

A reasonably expressive family of logics for which this is the case is the family of *guarded logics*, see Section 4. Guarded fragments of first-order logic, second-order logic, or fixed-point logics are defined by restricting quantification so that, semantically speaking, each subformula can simultaneously refer only to elements that are "very close together" or "guarded". In this way guarded logics generalise propositional *modal* logics, where quantification proceeds only along edges, to relational structures of arbitrary vocabulary, still retaining the good balance between expressiveness and algorithmic manageability (see [10]).

In game-theoretic terms, guarded quantification limits the number of possible moves in the evaluation games and thus leads to smaller game graphs. Depending on the conditions imposed on guard formulae, one obtains logics with different levels of "guardedness". We consider guarded fragments of least fixed-point logic with two notions of guardedness. For the simplest variant, where the guards are just atoms, we obtain games that grow linearly with the size of the structure. Consequently, model checking for the solitaire fragment of this guarded fixed-point logic can be performed in linear time. For a more liberal variant, clique-guarded fixed-point logic, the size of the game depends on the size of cliques in the Gaifman graph of the structure which is itself bounded by the tree width of the structure.

2. Least Fixed-Point Logic

Leastfixed-point logic, denoted LFP, extends first-order logic by least and greatest fixed points of definable relational operators. We briefly recall some basic definitions here. For a more extensive introduction to LFP, refer to [12].

Every formula $\psi(R, \boldsymbol{x})$, where R is a relation symbol of arity k and $\boldsymbol{x}$ is a tuple of k variables, defines, for any structure $\mathfrak{A}$ of appropriate vocabulary, an update operator $F : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ on the class of k-ary relations over the universe A of $\mathfrak{A}$, namely, $F : R \mapsto \{\boldsymbol{a} : (\mathfrak{A}, R) \models \psi(R, \boldsymbol{a})\}$. If ψ is positive in R, that is, if every occurrence of R falls under an even number of negations, this operator is monotone in the sense that $R \subseteq R'$ implies $F(R) \subseteq F(R')$. It is well known that every monotone operatore has a least fixed point and a greatest fixed point, which can be defined as the intersection, respectively the union, of all fixed points, but which can also be constructed by transfinite induction. For the least fixed point, the *stages* are defined by $X^0 := \emptyset$, $X^{\alpha+1} := F(X^\alpha)$, and $X^\lambda := \bigcup_{\alpha<\lambda} X^\alpha$ for limit ordinals λ. By the monotonicity of F, the sequence of stages increases until it reaches the least fixed point. The *greatest fixed point* is constructed in a dual way, starting with A^k as the initial stage and taking intersections at limit ordinals.

Formally, LFP is defined by adding to the syntax of first-order logic the following *fixed-point formation rule*: If $\psi(R, \boldsymbol{x})$ is a formula with a relational variable R occurring only positively and a tuple of first-order variables $\boldsymbol{x}$, and if $\boldsymbol{t}$ is a tuple of terms (such that the lengths of $\boldsymbol{x}$ and $\boldsymbol{t}$ match the arity of R), then

$$[\mathbf{lfp}\, R\boldsymbol{x} \,.\, \psi](\boldsymbol{t}) \quad \text{and} \quad [\mathbf{gfp}\, R\boldsymbol{x} \,.\, \psi](\boldsymbol{t})$$

are also formulae, binding the variables R and $\boldsymbol{x}$.

The semantics of least fixed-point formulae in a structure $\mathfrak{A}$, providing interpretations for all free variables in the formula, is the following: $\mathfrak{A} \models [\mathbf{lfp}\, R\boldsymbol{x} \,.\, \psi](\boldsymbol{t})$ if $\boldsymbol{t}^{\mathfrak{A}}$ belongs to the least fixed point of the update operator defined by ψ on $\mathfrak{A}$. Similarly for greatest fixed points.

Note that in formulae $[\mathbf{lfp}\, R\boldsymbol{x} \,.\, \psi](\boldsymbol{t})$ one may allow ψ to have other free variables besides $\boldsymbol{x}$, which are called *parameters*. However, every LFP-formula can easily be transformed into an equivalent one without parameters, at the expense of increasing the arity of fixed-point variables. In this article we only consider fixed-point formulae without parameters.

The duality between least and greatest fixed point implies that for any formula ψ,

$$[\mathbf{gfp}\, R\boldsymbol{x} \,.\, \psi](\boldsymbol{t}) \equiv \neg[\mathbf{lfp}\, R\boldsymbol{x} \,.\, \neg\psi[R/\neg R]](\boldsymbol{t}).$$

Using this duality together with de Morgan's laws, every LFP-formula can be brought into *negation normal form*, where negation applies to atoms only.

The model checking problem for a logic $\mathcal{L}$ is to establish for a given formula $\psi \in \mathcal{L}$ and an appropriate finite structure $\mathfrak{A}$, whether $\mathfrak{A} \models \psi$. The complexity of model checking problems can be measured in different ways. In general, both the structure and the formula are considered as inputs and we speak about the *combined complexity* of $\mathcal{L}$. However, in many instances it makes sense to fix either the formula or the structure and measure the complexity in terms of the other input, thus obtaining the notions of *data complexity* and *expression complexity*. We say that the data complexity of $\mathcal{L}$ is complete for a complexity class $\mathcal{C}$ if the model-checking problem is in $\mathcal{C}$ for *every* fixed formula $\psi \in \mathcal{L}$, and if it is $\mathcal{C}$-hard for *some* fixed formula $\psi \in \mathcal{L}$ (and similarly for the expression complexity).

For general LFP-formulae the model-checking complexity is well known [14], [21].

Theorem 2.1. *The combined complexity and the expression complexity of* LFP *is* EXPTIME*-complete, and the data complexity is* PTIME*-complete.*

Note that the bounds for expression and combined complexity take into account only the length $|\psi|$ of the input formula. It turns out that the critical parameter responsible for the EXPTIME-completeness is actually the *width* of a formula, i.e., the maximal number of free first-order variables in its subformulae. Fortunately, in many applications we only need formulae of small width. In particular, the modal μ-calculus can be translated to LFP-formulae of width two. For LFP-formulae of bounded width, better complexity bounds apply.

Proposition 2.2. *The model-checking problem for* LFP*-formulae of bounded width* (*and for the modal* μ*-calculus*) *is contained in* NP $\cap$ co-NP *and* PTIME*-hard.*

It is open whether this problem can be solved in polynomial time. Positive results have been obtained for fragments of LFP. One such partial result involves the *alternation depth*, i.e., the number of genuine alternations between the least and greatest fixed points in a formula; in the special case when no such alternations occur, the formula is called *alternation free*. For LFP-formulae of bounded width and bounded alternation depth the model-checking problem can be solved in polynomial time.

The fragments of bounded alternation depth in LFP induce a strict semantical hierarchy [4], [19]. On finite structures, this remains true for the modal μ-calculus (since it has the finite model property) but not for LFP. Every LFP-formula is equivalent, over finite structures, to an alternation-free one, indeed to a formula with a single application of an **lfp**-operator to a first-order formula [14]. However, this result does not help to improve the model-checking complexity, since the proof collapses d nested k-ary fixed points to one of width dk, and the complexity of LFP is exponential in the formula width.

2.1. *Model-Checking Games*

Model-checking problems, for almost any logic, can be formulated as the problem of deciding winning positions in the appropriate evaluation games. For fixed-point logics like LFP or L_μ, the evaluation games are *parity games*. A parity game is given by a transition system $\mathcal{G} = (V, V_0, E, \Omega)$, where V is a set of positions with a designated subset V_0, $E \subseteq V \times V$ is a transition relation, and $\Omega : V \to \mathbb{N}$ assigns to every position a *priority*. The number of priorities in the range of Ω is called 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, the player in turn loses. In case this never happens 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.) Given a starting position v_0, a strategy f for Player i is a *winning* strategy if he wins every play from v_0 in which he moves according to f. More generally, a strategy is winning on a set W if it is winning from every position $v_0 \in W$. The Forgetful Determinacy Theorem for parity games [7] 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 2.3 (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 checked efficiently whether a strategy is winning, the question whether a given position is winning for Player 0 can be decided in NP $\cap$ co-NP. In fact, it is known [16] 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 [17].

Theorem 2.4. *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*

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

Consider a structure $\mathfrak{A}$ and an LFP-sentence ψ which we may assume to be in negation normal form, without parameters, and *well-named*, in the sense that every fixed-point variable is bound only once.

The model-checking game $\mathcal{G}(\mathfrak{A}, \psi)$ is a parity game whose positions are formulae $\varphi(\boldsymbol{a})$ such that $\varphi(\boldsymbol{x})$ is a subformula of ψ, and $\boldsymbol{a}$ is a tuple of elements of $\mathfrak{A}$, interpreting the free variables of φ. The initial position is ψ.

Player 0 (Verifier) moves at positions associated to disjunctions and to formulae starting with an existential quantifier. From a position $\varphi \vee \vartheta$ she moves to either φ or ϑ and from a position $\exists y\, \varphi(\boldsymbol{a}, y)$ she can move to any position $\varphi(\boldsymbol{a}, b)$ for $b \in A$. In addition, Verifier is supposed to move at atomic false positions, i.e., at positions φ of form $a = a'$, $a \neq a'$, $R\boldsymbol{a}$, or $\neg R\boldsymbol{a}$ (where R is *not* a fixed-point variable) such that $\mathfrak{A} \models \neg\varphi$. However, positions associated with these atoms do not have successors, so Verifier loses at atomic false positions. Dually, Player 1 (Falsifier) moves at conjunctions and universal quantifications, and loses at atomic true positions. In addition, there are positions associated with fixed-point formulae $[\mathbf{fp} T\boldsymbol{x} \,.\, \varphi(T, \boldsymbol{x})](\boldsymbol{a})$ and with fixed-points atoms $T\boldsymbol{x}$, for fixed-point variables T. 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, for any fixed-point variable T in ψ there is a unique subformula $[\mathbf{fp} T\boldsymbol{x} \,.\, \varphi(T, \boldsymbol{x})](\boldsymbol{a})$. From position $[\mathbf{fp} T\boldsymbol{x} \,.\, \varphi(T, \boldsymbol{x})](\boldsymbol{a})$ Falsifier moves to $\varphi(T, \boldsymbol{a})$, and from $T\boldsymbol{b}$ he moves to $\varphi(T, \boldsymbol{b})$.

The priority labelling assigns even priorities to **gfp**-atoms and odd priorities to **lfp**-atoms. Further, if T, T' are fixed-point variables of different kind with T' depending on T (which means that T occurs free in the formula defining T'), then T-atoms get lower priority than T'-atoms. All remaining positions, not associated with fixed-point variables, receive highest priority. As a result, the number of priorities in the model-checking games equals the alternation depth of the fixed-point formula plus one. For more details and explanations, and for the proof that the construction is correct, see, e.g., [12] and [20].

Theorem 2.5. *Let ψ be an* LFP*-sentence and let $\mathfrak{A}$ be a relational structure. Then $\mathfrak{A} \models \psi$ if, and only if, Player* 0 *has a winning strategy for the parity game $\mathcal{G}(\mathfrak{A}, \psi)$.*

For sentences ψ of width k, the game $\mathcal{G}(\mathfrak{A}, \psi)$ can be constructed in linear time with regard to its size $O(|A|^k \cdot |\psi|)$. According to Theorem 2.4, we obtain the following complexity bounds for model checking LFP via the associated parity game.

Proposition 2.6. *For a finite structure $\mathfrak{A}$ and an* LFP*-sentence ψ of width k and alternation depth d, the model-checking problem can be solved in space $O(d \cdot |A|^k \cdot |\psi|)$*

and time

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

Note that, if both the alternation depth and the width of the formulae are bounded, the algorithm runs in polynomial time. As mentioned above, the model-checking problem is EXPTIME-complete for formulae of unbounded width, even if there is only one application of an LFP-operator. The important unresolved case concerns LFP-formulae with bounded width, but unbounded alternation depth. This includes the μ-calculus, since every formula of L_μ can be translated into an equivalent LFP-formula of width two. In fact, the following three problems are algorithmically equivalent, in the sense that if one of them admits a polynomial-time algorithm, then all of them do:

(1) Computing winning sets in parity games.
(2) The model-checking problem for LFP-formulae of width at most k, for any $k \geq 2$.
(3) The model-checking problem for the modal μ-calculus.

3. A Tractable Case

The so far unresolved question whether fixed-point logics admit efficient model-checking algorithms, and the correspondence between parity games and fixed-point logics suggests that one may identify algorithmically simple fragments of fixed-point logics by studying games of restricted shape that can be solved efficiently. A promising example for the effectivity of this direction is the correspondence between alternation-free formulae and dull games. To define these games we call a cycle in a game graph even (or odd) if the least priority occurring on it is thus.

Definition 3.1. A parity game is *dull* if even and odd cycles are disjoint. A game is called *weak* if priorities cannot decrease along transitions.

Weak games and dull games are closely related notions. Observe that every weak game is also dull. Conversely, any dull game can be transformed in linear time into an equivalent weak game, by changing only the priorities, not the game graph. Kupferman et al. [18] established (using different terminology) that dull games can be solved in linear time and that they emerge as model-checking games for alternation-free L_μ-formulae. Actually, dull games correspond in general to the alternation-free fragment of LFP (see [2]). As a consequence, the problem of checking a model $\mathfrak{A}$ against an alternation-free LFP-formula ψ of width k can be solved in time $O(|\psi| \cdot |A|^k)$. If ψ is a formula of the modal μ-calculus or guarded fixed-point logic, the complexity depends only linearly on the size of the structure.

3.1. *Solitaire Games*

Instead of restricting reachable priorities, we can take a different approach to render games easy, namely, by restricting the interaction between the players. The simplest case is given by solitaire games.

Definition 3.2. A parity game is called *solitaire* if all nontrivial moves are performed by the same player.

In a soltaire game where only Player 0 makes nontrivial moves, his winning set consists of those positions from which a terminal position in V_1 or an even cycle is reachable. Consequently, each strongly connected component of a solitaire game is completely included in the winning set of one of the two players.

The positions from which terminal positions in V_1 are reachable, can be computed in linear time using depth-first search. Hence, we may restrict our attention to games $\mathcal{G}$ without terminal positions. Without loss of generality, we can assume that all positions belong to Player 0.

In the simplest setting, when only priorities 0 and 1 occur in $\mathcal{G}$, the winning set of Player 0 is the set of nodes from which a nontrivial strongly connected component containing at least one position of priority 0 is reachable. By partitioning the game graph into its strongly connected components the solution of $\mathcal{G}$ can be computed in linear time.

Games of higher index can be solved by reduction to several instances of games of the above kind. For every even priority i occurring in the game $\mathcal{G}$, let $\mathcal{G}_i$ be the restriction of $\mathcal{G}$ to positions of priority $j \geq i$ where the priority i is replaced by 0 and all positions of priority $j > i$ receive priority 1. Note that if Player 0 wins from a position v in some game $\mathcal{G}_i$, than he also wins from v in the original game $\mathcal{G}$. Conversely, Player 0 wins in $\mathcal{G}$ only if he can reach a winning position v in some $\mathcal{G}_i$. Hence, we can solve $\mathcal{G}$ by first computing the winning positions of Player 0 for each $\mathcal{G}_i$. The winning set of Player 0 in $\mathcal{G}$ comprises all strongly connected components from which one of these winning positions is reachable.

To summarise, our method involves two reachability tests, one at the beginning, to handle terminal positions, and one at the end; between these, for every even priority, the solution of a solitaire game with only two priorities is computed. Each of these steps requires only linear time with respect to the size of the game graph.

Proposition 3.3. *The solution of a solitaire game $\mathcal{G} = (V, V_0, E, \Omega)$ of index d can be computed in time $O(d \cdot (|V| + |E|))$.*

A significant feature of parity games in general is that their main complexity resides in strongly connected components. Indeed, as pointed out in [2], the partial solutions of subgames induced in a game by strongly connected components can be propagated to obtain the global solution with only linear overhead.

Definition 3.4. A parity game is called *nested solitaire* if each strongly connected component induces a solitaire game.

Observe that in a nested solitaire game both players may perform nontrivial moves.

To solve a nested solitaire game $\mathcal{G}$ we can proceed as follows. First, we decompose the game graph into its strongly connected components. Note that in any terminal component C, that is, a strongly connected component with no outgoing edges, a position is winning in $\mathcal{G}$ iff it is also winning in the subgame induced by C. As a solitaire game, this subgame can be solved efficiently, providing a partial solution for the winning sets W_0 and W_1 in $\mathcal{G}$.

Next, we extend the obtained partial solution by assigning to W_0 the positions $v \in V_0$ with some successor already in W_0, and $v \in V_1$ with all successors already in W_0; the partial solution for W_1 propagates dually. Let $\mathcal{G}'$ be the subgame induced by the positions that remained unassigned after the propagation process. In the case where there are no such positions, we are done. Otherwise, we reiterate the procedure for $\mathcal{G}'$, which is again a nested solitaire game.

By adapting an algorithm from [3], the strongly connected components of the game graph can be maintained during the propagation process in linear time. For details, see also [2].

Theorem 3.5. *A nested solitaire game $\mathcal{G} = (V, V_0, E, \Omega)$ of index d can be solved in time $O(d \cdot (|V| + |E|))$.*

Notice that any positional strategy can be presented as a solitaire game. In the automata-theoretic view, a solitaire game where all nontrivial moves are perfomed by Player 1 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.

3.2. *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. Hence we obtain solitaire model-checking games for formulae where either $\wedge$ and $\forall$ (or, equivalently, $\vee$ and $\exists$) do not appear, and negations are only applied to atomic formulae. However, these formulae are of very limited expressive power.

In order to understand which formulae lead to nested solitaire games, observe that all cycles in model-checking games arise by regeneration of fixed-point variables. Thus, to guarantee that a nontrivial move $\varphi \to \varphi'$ leaves the current strongly connected component, we have to ensure that φ and φ' do not depend on a common fixed-point variable. However, according to the rules of the game, a fixed-point variable that is free in φ' must be free already in φ. In contrast, if φ' has no free fixed-point variables, then no position with φ will be reachable from positions with φ'.

We say that an LFP-formula is *closed* if it does not contain free fixed-point variables.

Definition 3.6. The *solitaire fragment* of LFP, denoted Solitaire-LFP, consists 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.

As an example, the following Solitaire-LFP formula on graphs defines the set of positions that lay on a cycle:

$$\varphi(x) = [\mathbf{lfp}\ Ruv\,.\,Euv \vee \exists w(Euw \wedge Rwv)](x, x).$$

Using this formula, we may define the set of positions from which there is a path passing through infinitely many positions that are not located on a cycle:

$$[\mathbf{gfp}\ Tx\,.\,[\mathbf{lfp}\ Sx\,.\,\exists y(Exy \wedge (Sy \vee (\neg\varphi(y) \wedge Ty)))](x)](x).$$

Note that the definition of Solitaire-LFP is not closed under transformation of formulae into negation normal form. However, when speaking of a solitaire formula ψ we tacitly assume that ψ is the presentation in negation normal form of a formula complying with the above definition. Under this proviso, a straightforward induction over closed subformulae shows that Solitaire-LFP indeed corresponds to nested solitaire games.

Proposition 3.7. *The model-checking games associated with* Solitaire-LFP *formulae are nested solitaire games.*

We remark that the solitaire fragment of L_μ has been studied under the name L_2 in [8].

3.3. *Complexity*

As the model-checking games of Solitaire-LFP are nested solitaire, we obtain the following deterministic complexity bound as a direct consequence of Theorem 3.5.

Proposition 3.8. *The model-checking problem for a structure $\mathfrak{A}$ and a solitaire* LFP*-sentence ψ of width k and alternation depth d can be solved in time $O(d \cdot |\psi| \cdot |A|^k)$.*

In terms of major complexity classes, the following results can be established.

Theorem 3.9. *The expression and combined complexity of* Solitaire-LFP *is* PSPACE*-complete, and the data complexity is* NLOGSPACE*-complete.*

Proof. For the upper bounds, we present a recursive nondeterministic procedure Eval($\mathfrak{A}$, ψ) which, given a structure $\mathfrak{A}$ and a closed (instantiated) Solitaire-LFP-formula ψ, decides whether $\mathfrak{A} \models \psi$. For convenience, we assume that in conjunctions the first formula is always closed. Further, let $G(\psi)$ denote the set of **gfp**-variables in ψ.

In the given form, the algorithm below may produce nonterminating computations. To prevent this, we can enforce rejection when the iteration count of the main loop exceeds $2 \cdot |\psi| \cdot |A|^k$, where k is the width of ψ. Maintaining such a counter requires space $O(\log|\psi| + k \cdot \log|A|)$ for each recursion level.

Up to the handling of fixed points, this algorithm is a variant of a common method for first-order evaluation. Essentially, the procedure Eval assumes the role of Verifier

function Eval($\mathfrak{A}$, ψ)
guess a formula witness $\in \{R\boldsymbol{a} : R \in G(\psi)\} \cup \{\bot\}$
seen_witness := *false*
do
  if ψ is an atom **then**
    return $\mathfrak{A} \models \psi$
  if $\psi = \neg\vartheta$ **then**
    return $\neg$Eval($\mathfrak{A}$, ϑ) (* ϑ is a closed formula *)
  if $\psi = \varphi_1 \vee \varphi_2$ **then**
    guess $i \in \{1, 2\}$; $\psi := \varphi_i$
  if $\psi = \vartheta \wedge \varphi$ **then**
    if $\neg$Eval($\mathfrak{A}$, ϑ) **then return** *false* **else** $\psi := \varphi$
  if $\psi = \exists x\varphi$ **then**
    guess $b \in A$; $\psi := \varphi[x \mapsto b]$
  if $\psi = \forall x\varphi$ **then**
    $r := true$
    for all $b \in A$ **do** $r := r \wedge$ Eval($\mathfrak{A}$, $\varphi[x \mapsto b]$)
    return r
  if $\psi = [\mathbf{fp}T\boldsymbol{x}.\varphi](\boldsymbol{c})$ **then**
    $\psi := \varphi(\boldsymbol{c})$
  if $\psi = T\boldsymbol{a}$ (a fixed-point atom) **then**
    if (seen_witness $\wedge$ $T\boldsymbol{a}$ = witness) **then return** *true* (* even cycle found *)
    if (seen_witness $\wedge$ T does not depend on witness) **then return** *false*
    if ($\neg$seen_witness $\wedge$ $T\boldsymbol{a}$ = witness) **then** seen_witness := *true*
    $\psi := \varphi_T(\boldsymbol{c})$ (where φ_T is the formula defining T)
repeat

in the model-checking game $\mathcal{G}(\mathfrak{A}, \psi)$, in the sense that it nondeterministically guesses her moves at disjunctions and existential quantifications. The correctness is proved by induction over the closed subformulae of ψ; the only interesting cases are fixed-point variables.

We can argue in terms of the model-checking game. Assume that Verifier has a strategy to prove $\mathfrak{A} \models \psi$. Then, at the starting position of the game, $\mathcal{G}(\mathfrak{A}, \psi)$ she can ensure that the play either reaches an even cycle or it descends into a subgame $\mathcal{G}(\mathfrak{A}, \varphi)$, with φ a closed subformula of ψ. The latter case is covered by the induction hypothesis. For the former, let $T\boldsymbol{a}$ be a position of lowest priority on the cycle. Guessing this position as a witness at the first step of the algorithm will lead to acceptance (even cycle found) within less than $2 \cdot |\psi| \cdot |A|^k$ iterations, which are enough to see any game position twice.

For the other direction, if $\mathfrak{A} \not\models \psi$, then any recursive call Eval($\mathfrak{A}$, φ) would lead to rejection by induction hypothesis, as φ must be a closed subformulae of ψ. On the other hand, the program cannot accept at the top level either, since for any guess of a greatest fixed-point atom $T\boldsymbol{a}$ as a witness, each cycle in $\mathcal{G}(\mathcal{A}, \psi)$ that contains $T\boldsymbol{a}$ also contains positions of lower priority, i.e., associated to variables that do not depend on T. Thus, $T\boldsymbol{a}$ cannot be regenerated without regenerating one of these first, at which point the procedure rejects.

Now we consider the space requirement. During evaluation of a formula ψ of width k, the recursion depth does not exceed the nesting depth of a closed subformulae in ψ, which is itself bounded by $|\psi|$. At each recusion level a pointer to ψ, together with an assignment consisting of k pointers to elements of A, is stored. Hence, overall the algorithm requires space $O(|\psi| \cdot (\log|\psi| + k \log|A|)$.

The lower bounds follow immediately from the efficient translation of transitive closure logic into Solitaire-LFP and Proposition 3.10 in the following section. □

3.4. *Expressive Power*

Transitive Closure Logic. A semantic fragment of LFP that is well-behaved in terms of complexity is *transitive closure logic*, TC, which extends first-order logic by a constructor for forming the transitive closure of definable relations. Syntactically, if $\varphi(\boldsymbol{x}, \boldsymbol{y})$ is a formula in variables $\boldsymbol{x}, \boldsymbol{y}$, and $\boldsymbol{s}, \boldsymbol{t}$ are terms, the tuples $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}$, and $\boldsymbol{t}$ being all of the same length, then

$$[\mathbf{tc}_{\boldsymbol{x},\boldsymbol{y}}\varphi(\boldsymbol{x}, \boldsymbol{y})](\boldsymbol{s}, \boldsymbol{t})$$

is a also a TC-formula. Its meaning can be expressed in terms of LFP as

$$[\mathbf{lfp}T\boldsymbol{x}\boldsymbol{y} \,.\, \varphi(\boldsymbol{x}, \boldsymbol{y}) \vee \exists \boldsymbol{z}(T\boldsymbol{x}\boldsymbol{z} \wedge \varphi(\boldsymbol{z}, \boldsymbol{y})](\boldsymbol{s}, \boldsymbol{t}).$$

Observe that any TC-formula translates into an alternation-free LFP-formula of the same width. The model-checking complexity of TC is well-understood [13], [15], [21].

Proposition 3.10. *The model-checking problem for* TC *is* PSPACE*-complete in the general case and* PTIME*-complete for formulae of bounded width. The data complexity is* NLOGSPACE*-complete.*

Linear Stratified Datalog. Transitive closure logic can be naturally characterised in terms of the database query language Datalog. A *Datalog rule* is an expression of the form $H \leftarrow B_1, \ldots, B_r$ where H, the *head* of the rule, is an atomic formula $Ru_1 \cdots u_s$, and $B_1, \ldots, B_r$, the *body* of the rule, is a collection of literals (i.e., atoms or negated atoms). The relation symbol R is called the *head predicate* of the rule. A *basic Datalog program* Π is a finite collection of rules such that none of its head predicates occurs negated in the body of any rule. The predicates which appear only in the bodies of the rules are called *input* predicates. Given a relational structure $\mathfrak{A}$ over the vocabulary of the input predicates, the program computes, via the usual fixed-point semantics, an interpretation for the head predicates.

A *stratified Datalog program* is a sequence $\Pi = (\Pi_0, \ldots, \Pi_r)$ of basic Datalog programs, called the *strata* of Π, such that each of the head predicates of Π is a head predicate in precisely one stratum Π_i and is used as a body predicate only in higher strata Π_j for $j \geq i$. More precisely,

(i) if a head predicate of stratum Π_j occurs *positively* in the body of a rule of stratum Π_i, then $j \leq i$, and
(ii) if a head predicate of stratum Π_j occurs *negatively* in the body a rule of stratum Π_i, then $j < i$.

The semantics of a stratified program is defined stratum per stratum. The body predicates of a stratum Π_i are either input predicates of the entire program Π or they are head predicates of lower strata. Hence, once the lower strata are evaluated, we can compute the interpretation of the head predicates of Π_i as in the case of basic Datalog. For details, consult [1].

A stratified Datalog program is *linear* if in the body of each rule there is at most one occurrence of a head predicate of the same stratum (but there may be arbitrarily many occurrences of head predicates from lower strata). Linear programs suffice to define transitive closures, so it follows by a straightforward induction that TC $\subseteq$ Linear Stratified Datalog. The converse is also true (see [6] and [9]).

Proposition 3.11. *Linear Stratified Datalog is equivalent to* TC.

Observe that the translation from transitive closure logic into LFP given in Section 2 involves only solitaire formulae. Consequently, Solitaire-LFP subsumes TC.

Lemma 3.12. TC $\subseteq$ Solitaire-LFP.

It follows from well-known results that the converse is not true in general. A simple example is the solitaire formula $[\mathbf{gfp} Tx\,.\,\exists y(Exy \wedge Ty)](x)$ expressing that there is an infinite path from x. It is known that this query is not even expressible in the infinitary logic $L_{\infty\omega}$ (otherwise, well-founded linear orders would be axiomatizable in $L_{\infty\omega}$). Even restricted to countable structures this query is not expressible in TC. However, on finite structures the converse does hold.

Theorem 3.13. *On finite structures*, Solitaire-LFP $\equiv$ TC.

To prove this, we exploit the solitaire-structure of the model-checking game and the fact that TC-formulae are equivalent to stratified linear Datalog programs. For every formula $\psi \in$ Solitaire-LFP, we construct a stratified linear Datalog program Π_ψ which defines the winning positions in the associated model-checking game.

More precisely, the construction proceeds by induction along the following lines:

(a) For every subformula $\varphi(\boldsymbol{x})$, we introduce a head predicate W_φ. On any finite structure $\mathfrak{A}$, the program evaluates the atom $W_\varphi(\boldsymbol{a})$ to true if, and only if, Verifier has a winning strategy from position $\varphi(\boldsymbol{a})$.

(b) Further, the program contains auxiliary head predicates $R_{\varphi T}$, for each **gfp**-formula $[\mathbf{gfp}\ T\boldsymbol{x}\,.\,\vartheta](\boldsymbol{x})$ in ψ and every subformula φ depending on T. On any finite $\mathfrak{A}$, the program evaluates $R_{\varphi T}(\boldsymbol{a}, \boldsymbol{b})$ to true if, and only if, in the game $\mathcal{G}(\mathfrak{A}, \psi)$ there is a path π from position $\varphi(\boldsymbol{a})$ to position $\vartheta(\boldsymbol{b})$ which does not pass through any position of priority less than $\Omega(T)$ and, moreover, π is reliable for Verifier in the following sense: at every position $v \in \pi$ at which Falsifier has a choice (this can only be at positions corresponding to conjunctions), the single move which does not follow π leads to a subgame in which Verifier wins. Note that $R_{\vartheta T}(\boldsymbol{a}, \boldsymbol{a})$ implies that Verifier has a strategy to reach a cycle of minimal priority $\Omega(T)$, unless she wins by other means.

We remark that the following construction is standard up to the treatment of the **gfp**-formulae via the reachability predicates $R_{\varphi T}$.

(1) If $\varphi(\boldsymbol{x})$ is a literal $(\neg)P\boldsymbol{x}$, the program Π_φ consists of the single rule

$$W_\varphi(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x}).$$

(2) For $\varphi = \neg\eta$, Π_φ is obtained by adding to Π_ϑ a new stratum with the rule

$$W_\varphi(\boldsymbol{x}) \leftarrow \neg W_\eta(\boldsymbol{x}).$$

Note that this is well-defined since η does not contain free fixed-point variables.

(3) For $\varphi = \eta \vee \vartheta$, the program Π_φ consists of $\Pi_\eta \cup \Pi_\vartheta$ together with the rules

$$W_\varphi(\boldsymbol{x}) \leftarrow W_\eta(\boldsymbol{x}), \qquad W_\varphi(\boldsymbol{x}) \leftarrow W_\vartheta(\boldsymbol{x})$$

and, if applicable,

$$R_{\varphi T}(\boldsymbol{x}, \boldsymbol{y}) \leftarrow R_{\eta T}(\boldsymbol{x}, \boldsymbol{y}), \qquad R_{\varphi T}(\boldsymbol{x}, \boldsymbol{y}) \leftarrow R_{\vartheta T}(\boldsymbol{x}, \boldsymbol{y}).$$

(4) For $\varphi = \vartheta \wedge \eta$, we can assume that ϑ does not contain free fixed-point variables. Now Π_φ is $\Pi_\vartheta \cup \Pi_\eta$ augmented with the rules

$$W_\varphi(\boldsymbol{x}) \leftarrow W_\vartheta(\boldsymbol{x}), W_\eta(\boldsymbol{x})$$

and, if applicable,

$$R_{\varphi T}(\boldsymbol{x}, \boldsymbol{y}) \leftarrow W_\vartheta(\boldsymbol{x}), R_{\eta T}(\boldsymbol{x}, \boldsymbol{y}).$$

Note that W_ϑ does not depend on W_φ, so the program is indeed linear.

(5) For $\varphi = \exists z \eta(\boldsymbol{x}, z)$, we obtain Π_φ by adding to Π_η the rules

$$W_\varphi(\boldsymbol{x}) \leftarrow W_\eta(\boldsymbol{x}, z), \quad \text{and} \quad R_{\varphi T}(\boldsymbol{x}, \boldsymbol{y}) \leftarrow R_{\eta T}(\boldsymbol{x}z, \boldsymbol{y})$$

for the appropriate **gfp**-variables T.

(6) For $\varphi = \forall z \eta(\boldsymbol{x}, z)$, the subformula η does not contain free fixed-point variables. We construct Π_φ by adding to Π_η the rule

$$L_\varphi(\boldsymbol{x}) \leftarrow \neg W_\eta(\boldsymbol{x}, z)$$

and, in a new stratum,

$$W_\varphi(\boldsymbol{x}) \leftarrow \neg L_\varphi(\boldsymbol{x}).$$

(7) For fixed-point definitions $\varphi = [\mathbf{fp}\ T\boldsymbol{x}\,.\,\vartheta](\boldsymbol{x})$, and likewise for the predicates $\varphi = T\boldsymbol{x}$, we construct Π_φ by adding to Π_ϑ the rule

$$W_\varphi(\boldsymbol{x}) \leftarrow W_\vartheta(\boldsymbol{x})$$

and, for all **gfp**-variables T' on which T depends (hence, $\Omega(T') \leq \Omega(T)$),

$$R_{\varphi T'}(\boldsymbol{x}, \boldsymbol{y}) \leftarrow R_{\vartheta T'}(\boldsymbol{x}, \boldsymbol{y}).$$

In case φ is a **gfp**-definition $\varphi = [\mathbf{gfp}\ T\boldsymbol{x}\,.\,\vartheta](\boldsymbol{x})$, the program Π_φ additionally contains the rule

$$W_\varphi(\boldsymbol{x}) \leftarrow R_{\vartheta T}(\boldsymbol{x}, \boldsymbol{x});$$

if it is a **gfp**-atom $\varphi = T\boldsymbol{x}$, we add

$R_{\varphi T}(\boldsymbol{x}, \boldsymbol{x})$.

It is readily seen that the solitaire structure of ψ implies that Π_ψ is indeed a linear stratified program. It remains to prove the following.

Lemma 3.14. *For every solitaire formula $\psi(\boldsymbol{x})$ and every finite structure $\mathfrak{A}$ we have that $\mathfrak{A} \models \psi(\boldsymbol{a})$ iff Π_ψ evaluates on $\mathfrak{A}$ the atom $W_\psi(\boldsymbol{a})$ to true.*

Proof. We show that the truth values for $W_\varphi(\boldsymbol{a})$ and $R_{\varphi T}(\boldsymbol{a}, \boldsymbol{b})$ defined by Π_ψ on $\mathfrak{A}$ indeed have the game theoretic meaning described by items (a) and (b) above.

A winning play for Verifier in $\mathcal{G}(\psi, \mathfrak{A})$ from a position $\varphi(\boldsymbol{a})$ must either lead in finitely many steps to a literal $(\neg)P\boldsymbol{b}$ that is true in $\mathfrak{A}$, or it must lead to a **gfp**-atom $T\boldsymbol{b}$ from which it cycles without hitting any priority smaller than $\Omega(T)$. It is not difficult to see that the rules of Π_ψ ensure a strategy for precisely this.

The rules for cases (1)–(6) reduce the winning conditions W_φ and the reachability conditions $R_{\varphi T}$ in the obvious way to the immediate subformulae of φ (i.e., to the positions after the next move).

In case (7) the rules do the same for $\varphi = [\mathbf{fp}T\boldsymbol{x} \,.\, \vartheta](\boldsymbol{x})$ and, in addition, they take into account the moves from $T\boldsymbol{a}$ back to $\vartheta(\boldsymbol{a})$. To win from $T\boldsymbol{a}$, Verifier must win from $\vartheta(\boldsymbol{a})$.

If a least-fixed point atom $T\boldsymbol{a}$ is regenerated infinitely often, then the play is lost (unless a **gfp**-variable of smaller priority is also regenerated infinitely often). Most importantly, the additional rules for **gfp**-formulae take care of the possibility to win by forcing an appropriate cycle. For $\varphi(\boldsymbol{x}) = [\mathbf{gfp}T\boldsymbol{x} \,.\, \vartheta](\boldsymbol{x})$, the rule $W_\varphi(\boldsymbol{x}) \leftarrow R_{\vartheta T}(\boldsymbol{x}, \boldsymbol{x})$ ensures that $W_\varphi(\boldsymbol{b})$ is evaluated to true if Verifier can force a reliable cycle that contains $\vartheta(\boldsymbol{b})$ and on which no priority smaller than $\Omega(T)$ is seen. Together with the other rules this further implies that Verifier also wins from positions where she can force a play that eventually hits such a cycle. □

This completes the proof of Theorem 3.13.

4. Small Model Checking Games for Guarded Logics

As we observed in Section 2, the parameter responsible for the high complexity of LFP model checking in the general case is the formula width. This parameter also has a major influence on the size of evaluation games. Already for first-order formulae of width k, the size of the corresponding model-checking game is $O(|A|^k|\psi|)$, as typically all possible assignments $\rho : \{x_1, \ldots, x_k\} \to A$ need to be taken into account. In this section we consider guarded logics that have model-checking games of much smaller size.

Syntactically, guarded logics are based on a restriction of first-order quantification to the form $\exists\mathbf{y}(\alpha(\mathbf{x}, \mathbf{y}) \wedge \psi(\mathbf{x}, \mathbf{y}))$ or $\forall\mathbf{y}(\alpha(\mathbf{x}, \mathbf{y}) \to \psi(\mathbf{x}, \mathbf{y}))$ where quantifiers may range over a tuple $\mathbf{y}$ of variables, but are "guarded" by a formula α that contains all the free variables of the formula ψ that is quantified over. These guard formulae are of a simple syntactic form. In the following we consider guarded fragments of LFP where the guards are either atoms or clique-formulae.

The *guarded-fragment* GF of first-order logic is defined inductively as the closure of atomic formulae under Boolean connectives and the following quantification rule: For every formula $\psi(\boldsymbol{x}, \boldsymbol{y}) \in \mathrm{GF}$ and every atom $\alpha(\mathbf{x}, \mathbf{y})$ such that all variables that are free in ψ are also free in α, the formulae $\exists \mathbf{y}(\alpha(\mathbf{x}, \mathbf{y}) \wedge \psi(\mathbf{x}, \mathbf{y}))$ and $\forall \mathbf{y}(\alpha(\mathbf{x}, \mathbf{y}) \rightarrow \psi(\mathbf{x}, \mathbf{y}))$ belong to GF.

Guarded fixed-point logic μGF extends GF by the following fixed-point formation rule: Let T be a k-ary relation symbol, let $\boldsymbol{x}$ be a k-tuple of distinct variables, and let $\boldsymbol{t}$ be a k-tuple of terms; if $\psi(T, \boldsymbol{x})$ is a formula with no other free first-order variables than those in $\boldsymbol{x}$, in which T occurs only positively and is not used in guards, then we can build the formulae

$$[\mathbf{lfp} T\boldsymbol{x} \,.\, \psi](\boldsymbol{t}) \quad \text{and} \quad [\mathbf{gfp} T\boldsymbol{x} \,.\, \psi](\boldsymbol{t}).$$

The semantics of guarded fixed-point logic is the same as for LFP. We sometimes use the notation $(\exists \mathbf{y} \,.\, \alpha)\psi$ and $(\forall \mathbf{y} \,.\, \alpha)\psi$ for guarded formulae. Obviously, 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.

For some applications it is too restrictive to allow only atoms as guards (for instance, temporal operators like **until** cannot be expressed in GF). A natural and very powerful generalisation of guarded logics is based on the notion of clique guardedness.

Let $\mathfrak{A} = (A, R_1, \ldots, R_m)$ be a relational structure. A set $X \subseteq A$ 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 $\mathit{clique}(x_1, \ldots, x_k)$ such that, for every τ-structure $\mathfrak{A}$ and every k-tuple $\boldsymbol{a} \in A^k$, we have $\mathfrak{A} \models \mathit{clique}(\boldsymbol{a}) \iff \boldsymbol{a}$ is clique guarded in $\mathfrak{A}$. 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

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

belong to CGF, provided that all free variables in ψ are also free in *clique*.

In practice, one will not 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 guards which *imply* a clique-formula, i.e., any formula of the form $\gamma(\boldsymbol{x}, \boldsymbol{y}) := \exists \mathbf{z} \beta(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ 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 to keep our complexity results independent of these coding issues, we do not take into account the length of clique guards at all. See [11] for background on clique-guarded logics and their relations to other notions such as the *loosely guarded* or *packed* fragments.

Let $\mathfrak{A}$ be a structure and let ψ be a guarded formula. Then, at any position $(\exists \boldsymbol{y} \,.\, \alpha(\boldsymbol{a}, \boldsymbol{y}))\varphi(\boldsymbol{a}, \boldsymbol{y})$ in the model-checking game $\mathcal{G}(\mathfrak{A}, \psi)$, Verifier should move to a position $\varphi(\boldsymbol{a}, \boldsymbol{b})$ where $\boldsymbol{b}$ satisfies $\mathfrak{A} \models \alpha(\boldsymbol{a}, \boldsymbol{b})$, otherwise she would lose immediately. The same hold for Falsifier at positions $\big(\forall \boldsymbol{y} \,.\, \alpha(\boldsymbol{a}, \boldsymbol{y})\big)\varphi(\boldsymbol{a}, \boldsymbol{y})$. On that account we can narrow the possible moves at quantifications to cover only positions where the guards are satisfied. Further, the game can be restricted to the set of positions reachable from the initial position ψ. The construction of this game is straightforward and can be done in linear time with respect to its size.

Towards an accurate estimation of game sizes, besides the length and the width of the given formula ψ, a relevant notion is the closure $\mathrm{cl}(\psi)$, that is, the set of its subformulae. 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).

Proposition 4.1. *Given a sentence $\psi \in \mu\mathrm{GF}$ and a finite structure $\mathfrak{A}$, the associated parity game $\mathcal{G}(\mathfrak{A}, \psi)$ is of size $O(\|\mathfrak{A}\| \cdot |\psi|)$.*

Proof. 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 not more than $|\psi|$ edges. We claim that for every edge $\varphi \to \varphi'$ of $H(\psi)$ there exist at most $\|\mathfrak{A}\|$ edges of form $\varphi(\boldsymbol{a}) \to \varphi'(\boldsymbol{a}')$ in the game graph $\mathcal{G}(\mathfrak{A}, \psi)$.

We consider several cases. First, let $\varphi = (Q\boldsymbol{x} \,.\, \alpha)\varphi'$ where Q stand for a quantifier. In that case an edge $\varphi(\boldsymbol{a}) \to \varphi(\boldsymbol{a}')$ can exist only if there exists an assignment ρ such that $\mathfrak{A} \models \alpha[\rho]$ which maps the free variables of φ and φ' respectively to $\boldsymbol{a}$ and $\boldsymbol{a}'$. Since 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 any fixed position $\varphi(\boldsymbol{a})$ has at most two outgoing edges $\varphi(\boldsymbol{a}) \to \varphi'(a')$. Hence, it suffices to show that for each such $\varphi \in H(\psi)$ there exist at most $\|\mathfrak{A}\|$ reachable positions $\varphi(\boldsymbol{a})$ in the game graph. Without loss, we may assume that fixed-point variables occur inside their defining formulae only under the scope of a quantifier. Formulae φ that are not inside a quantifier are sentences and occur only once. For all other formulae $\varphi(\boldsymbol{x})$ there is a uniquely determined least subformula of ψ that strictly contains φ and has the form $(Q\boldsymbol{y} \,.\, \alpha)\vartheta(\boldsymbol{z})$. Moreover, the free variables $\boldsymbol{x}$ of φ are contained in $\boldsymbol{z}$. Then a position $\varphi(\boldsymbol{a})$ is reachable if, and only if, a position $\vartheta(\boldsymbol{c})$ is reachable, such that the assignment $\boldsymbol{z} \mapsto \boldsymbol{c}$ extends the assignment $\boldsymbol{x} \mapsto \boldsymbol{a}$. Note that ϑ and α are uniquely determined by φ, and the position $\vartheta(\boldsymbol{c})$ is reachable only if $\mathfrak{A} \models \alpha[\rho]$ for some ρ that extends the assignment $\boldsymbol{z} \mapsto \boldsymbol{c}$. By the same argument as above it follows that the number of reachable positions $\varphi(\boldsymbol{a})$ is bounded by $\|\mathfrak{A}\|$. This completes the proof. □

According to Theorem 2.4, we obtain the following complexity bounds for model checking μGF via the associated parity game.

Proposition 4.2. *Given a structure $\mathfrak{A}$ and a μGF-sentence ψ of alternation depth d*

the model-checking problem can be solved in space $O(d \cdot \|\mathfrak{A}\| \cdot |\psi|)$ *and time*

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

Note that in μGF, the formula width is implicitly bounded by the arity of the relation symbols. (On graphs, for instance, GF and μGF are actually two-variable logics.) However, even when we compare the obtained complexity for μGF with the corresponding result for LFP-formulae of bounded width in Proposition 2.6, the difference is significant.

In contrast to μGF, we can write formulae of unbounded width in μCGF. As a consequence, the model-checking problem for μCGF is as hard as for LFP in terms of major complexity classes. To prove this, one takes input structures with a complete binary guard relation, so that all tuples in the structure become clique guarded.

Proposition 4.3. *The combined complexity and the expression complexity of* μCGF *is* EXPTIME*-complete*, *and the data complexity is* PTIME*-complete*.

However, as we will see, for formulae of bounded width, the complexity of μCGF levels off between μGF and LFP.

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 decomposition.

Definition 4.4. 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 guarded set in $\mathcal{B}$ is included in some block and (2) for any element $a \in B$ the set of blocks which contain a is connected. The *tree width* of $\mathcal{B}$ is the minimal width of a tree decomposition of $\mathcal{B}$.

For a comprehensive presentation of the notions of tree decomposition and width see, e.g., Chapter 12 of [5].

Lemma 4.5. *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* $(l+1)^k \cdot |\mathfrak{A}|$.

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 [11] 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 $|T| \leq |A|$, 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 4.6. *Given a μCGF-sentence ψ of width k and a finite structure $\mathfrak{A}$ of tree width l, the size of the associated parity game $\mathcal{G}(\mathfrak{A}, \psi)$ is bounded by $O(l^k|A| \cdot |\mathrm{cl}(\psi)|)$.*

This means that for structures of small tree width μCGF-games are considerably smaller than LFP-games. 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 an unguarded LFP-formula of width k. This difference is also reflected in the model-checking complexity.

Proposition 4.7. *For a structure $\mathfrak{A}$ of tree width l and a μCGFsentence ψ of width k and alternation depth d, the model-checking problem can be solved in space $O(d \cdot l^k|A| \cdot |\mathrm{cl}(\psi)|)$ and time*

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

Finally, we interpret these results for the guarded solitaire formulae, that is, formulae of μGF and μCGF formed according to Definition 3.6.

Proposition 4.8. *The model-checking problem for a structure $\mathfrak{A}$ of tree width l and a solitaire μCGF-sentence ψ of width k and alternation depth d, can be solved in time $O(d \cdot l^k|A| \cdot |\mathrm{cl}(\psi)|)$. In particular, for $\varphi \in \mu\mathrm{GF}$ the problem can be solved in time $O(d \cdot \|\mathfrak{A}\| \cdot |\psi|)$.*

References

[1] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*, Addison-Wesley, Reading, MA, 1995.

[2] D. Berwanger and E. Grädel, Games and model checking for guarded logics, in *Proceedings of LPAR* 2001, Lecture Notes in Computer Science vol. 2250, Springer-Verlag, Berlin 2001, pp. 70–84.

[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*, Lecture Notes in Computer Science, vol. 1954, Springer-Verlag, Berlin, 2000, pp. 37–54.

[4] J. Bradfield, The modal μ-calculus alternation hierarchy is strict, *Theoretical Computer Science*, **195** (1998), 133–153.

[5] R. Diestel, *Graph Theory*, Springer-Verlag, Berlin 1997.

[6] H.-D. Ebbinghaus and J. Flum, *Finite Model Theory*, 2nd edn., Springer-Verlag, Berlin, 1999.

[7] A. Emerson and C. Jutla, Tree automata, mu-calculus and determinacy, in *Proceedings of the* 32*nd IEEE Symposium on Foundations of Computer Science*, 1991, pp. 368–377.

[8] A. Emerson, C. Jutla, and P. Sistla, On model checking for the μ-calculus and its fragments, *Theoretical Computer Science*, **258** (2001), 491–522.

[9] E. Grädel, On transitive closure logic, in *Proceedings of the* 5*th Workshop on Computer Science Logic*, *CSL* 91, *Bern* 1991, Lecture Notes in Computer Science, vol. 626, Springer-Verlag, Berlin, 1991, pp. 149–163.

[10] E. Grädel, Why are modal logics so robustly decidable?, in *Current Trends in Theoretical Computer Science*, World Scientific, Singapore, 2001, pp. 393–498.

[11] E. Grädel, Guarded fixed point logic and the monadic theory of trees, *Theoretical Computer Science*, **288** (2002), 129–152.

[12] E. Grädel, Finite model theory and descriptive complexity, In *Finite Model Theory and Its Applications*, Springer-Verlag, Forthcoming.
[13] E. GRÄDEL AND M. OTTO, On logics with two variables, *Theoretical Computer Science*, **224** (1999), 73–113.
[14] N. Immerman, Relational queries computable in polynomial time, *Information and Control*, **68** (1986), 86–104.
[15] N. Immerman, Languages that capture complexity classes, *SIAM Journal on Computing*, **16** (1987), 760–778.
[16] M. Jurdziński, Deciding the winner in parity games is in UP ∩ Co-UP., *Information Processing Letters*, **68** (1998), 119–124.
[17] M. Jurdziński, Small progress measures for solving parity games, in *STACS* 2000, 17*th Annual Symposium on Theoretical Aspects of Computer Science*, *Proceedings*, Lecture Notes in Computer Science, vol. 1770, Springer-Verlag, Berlin, 2000, pp. 290–301.
[18] O. Kupferman, M. Vardi, and P. Wolper, An automata-theoretic approach to branching-time model checking, *Journal of the ACM*, **47** (2000), 312–360.
[19] Y. Moschovakis, *Elementary Induction on Abstract Structures*, North-Holland, Amsterdam, 1974.
[20] C. Stirling, Bisimulation, model checking and other games, *Notes for the Mathfit Instructional Meeting on Games and Computation*, Edinburgh, 1997.
[21] M. Vardi, The complexity of relational query languages, in *Proceedings of the* 14*th ACM Symposium on the Theory of Computing*, 1982, pp. 137–146.

Received November 5, 2002, *and in revised form March* 21, 2003, *and in final form March* 31, 2003.
Online publication September 13, 2004.