Electronic Notes in Theoretical Computer Science 67 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume67.html 20 pages

Model Checking Games

Erich Grädel[1]

Aachen University of Technology

Abstract

We survey evaluation games for first-order logic and least fixed point logics, and discuss their algorithmic complexity.

1 Logic and complexity

In many areas of logic in computer science one ist interested in the *algorithmic complexity of the common reasoning tasks for a logic.* There are numerous such tasks, but most of them can be easily reduced to two (at least for logics with reasonable closure properties), namely *satisfiabiliy testing* and *model checking.* The *satisfiability problem* for a logic L and a domain $\mathcal{D}$ of structures, takes formulae $\psi \in L$ as inputs, and the question to be answered is whether there exists in $\mathcal{D}$ a model for ψ. The *model checking problem* (for L on $\mathcal{D}$) asks, given a structure $\mathfrak{A} \in \mathcal{D}$ and a formula $\psi \in L$, whether $\mathfrak{A} \models \psi$. A closely related problem is *formula evaluation* (or *query evaluation*): Given a structure $\mathfrak{A}$ and a formula $\psi(\overline{x})$ (with free variables $\overline{x}$), compute the relation defined by ψ on $\mathfrak{A}$, i.e., the set $\psi^{\mathfrak{A}} := \{\overline{a} : \mathfrak{A} \models \psi(\overline{a})\}$. Obviously, the evaluation problem for a formula with k free variables on a structure with n elements reduces to n^k model checking problems.

Note that a model checking problem has two inputs: a structure and a formula. We can measure the complexity in terms of both inputs, and this is what is commonly referred to as the *combined complexity* of the model checking problem (for L and $\mathcal{D}$). However, in many cases, one of the two inputs is fixed, and we measure the complexity only in terms of the other. If we fix a structure $\mathfrak{A}$, then the model checking problem for L on this structure amounts to deciding $\mathrm{Th}_L(\mathfrak{A}) := \{\psi \in L : \mathfrak{A} \models \psi\}$, the *$L$-theory of* $\mathfrak{A}$. The complexity of this problem is also called the *expression complexity* of the model checking problem (for L on $\mathfrak{A}$). In particular for first-order logic FO

[1] Mathematische Grundlagen der Informatik, RWTH Aachen, 52056 Aachen, Germany. Email: graedel@informatik.rwth-aachen.de
WWW: www-mgi.informatik.rwth-aachen.de/~graedel

and for monadic second-order logic MSO such problems have a long tradition in logic and numerous applications. Of even greater importance for many fields (including finite model theory and databases) are model checking problems for a fixed formula ψ, which amounts to deciding the *model class of* ψ *inside* $\mathcal{D}$, $\mathrm{Mod}_{\mathcal{D}}(\psi) := \{\mathfrak{A} \in \mathcal{D} : \mathfrak{A} \models \psi\}$. Due to its relevance for databases, the complexity of this problem is also called the *data complexity* of the model checking problem (for ψ on $\mathcal{D}$).

Model checking problems, for almost any logic, can be cast as *strategy problems for appropriate model checking games* (also called Hintikka games). With any formula $\psi(\overline{x})$, any structure $\mathfrak{A}$ (of the same vocabulary as ψ), and any tuple $\overline{a}$ of elements of $\mathfrak{A}$, we associate a *Hintikka game* $\mathcal{G}(\mathfrak{A}, \psi(\overline{a}))$. It is played by two players, *Verifier* and *Falsifier*. Verifier (sometimes also called Player 0, or $\exists$, or Eloise) tries to prove that $\mathfrak{A} \models \psi(\overline{a})$, whereas Falsifier (also called Player 1, or $\forall$, or Abelard) tries to establish that the formula is false. For first-order logic, evaluation games are very simple in the sense that winning conditions are *positional*, and that the games are *well-founded*, i.e., all possible plays are finite (regardless of whether the input structure is finite or infinite). For more powerful logics, notably fixed-point logics, model checking games may have infinite plays and more complicated winning conditions.

2 The model checking game for first-order logic

Let $\mathfrak{A}$ be a finite structure and $\psi(\overline{x})$ be a relational first-order formula, which we assume to be in negation normal form, i.e., built up from atoms and negated atoms by means of the propositional connectives $\wedge, \vee$ and quantifiers $\exists, \forall$. Obviously, any first-order formula can be efficiently converted into an equivalent one in negation normal form. The model checking game $\mathcal{G}(\mathfrak{A}, \psi(\overline{a}))$ has positions (φ, ρ) such that φ is a subformula of ψ, and $\rho : \mathrm{free}(\varphi) \to A$ is an assignment from the free variables of φ to elements of $\mathfrak{A}$. To simplify notation we usually write $\varphi(\overline{b})$ for a position (φ, ρ) where ρ assigns the tuple $\overline{b}$ to the free variables of φ. The initial position of the game is $\psi(\overline{a})$.

Verifier (Player 0) 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. From a position $\exists y \varphi(\overline{b}, y)$ Verifier can move to any position $\varphi(\overline{b}, c)$ where $c \in A$. Dually, Falsifier (Player 1) makes corresponding moves at conjunctions and universal quantifications. At atoms or negated atoms, i.e., positions $\varphi(\overline{b})$ of form $b = b'$, $b \neq b'$, $R\overline{b}$, or $\neg R\overline{b}$, the game is over. Verifier has won the play if $\mathfrak{A} \models \varphi(\overline{b})$, otherwise Falsifier has won.

Model checking games are a way of defining the semantics of a logic. The equivalence with the standard definition can be proved by a simple induction.

Proposition 2.1 $\mathfrak{A} \models \psi(\overline{a})$ *if, and only if, Verifier has a winning strategy for the game* $\mathcal{G}(\mathfrak{A}, \psi(\overline{a}))$.

This suggests a game-based approach to model checking: Given $\mathfrak{A}$ and ψ, construct the game $\mathcal{G}(\mathfrak{A}, \psi)$ and decide whether Verifier has a winning strategy from the initial position. Let us therefore look a bit closer at strategy problems for games.

3 The strategy problem for finite games.

Abstractly, we can describe a two-player game with positional winning conditions by a directed graph $\mathcal{G} = (V, V_0, E)$ whose universe V is the set of positions, where $V_0 \subseteq V$ is the set of positions from which Player 0 moves, and where $E \subseteq V \times V$ is the set of moves. By $V_1 := V - V_0$ we denote the set of positions from which Player 1 moves. For any position $v \in V$, let $vE := \{w : (v, w) \in E\}$ denote the set of successors of v. To describe the winning conditions we adopt the convention that Player i loses at positions $v \in V_i$ where no moves are possible (i.e., $vE = \emptyset$). Alternatively, one could explicitely include into the game description the sets S_0, S_1 of winning terminal positions for each player.

A *play* of $\mathcal{G}$ is a path $v_0 v_1 \dots$ formed by the two players starting from a given position v_0. At positions $v_n \in V_0$, Player 0 chooses a move $(v_n, v_{n+1}) \in E$, at $v_n \in V_1$, 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 has to be established by a winning condition on infinite plays. For the moment, let us say that infinite plays are won by none of the players.

A strategy for a player is a function defining a move for each situation in a play where he has to move. Of particular interest are *positional strategies*, which do not depend on the history of the play, but only on the current position. Hence, 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 $v \in V_i$. A play $v_0 v_1 \dots$ is consistent with a positional strategy f for Player i if $v_{n+1} = f(v_n)$ for all $v_n \in V_i$. A strategy for a player is *winning* from position v_0 if he wins every play starting from v_0 that is consistent with the strategy. We say that a strategy is winning on a set W if it is winning from each position in W. Given a game $\mathcal{G}$, we denote by W_i $(i = 0, 1)$ the sets of positions from which Player i has a winning strategy.

A game is called *well-founded* if all its plays are finite. For instance, the game graph of a model checking game $\mathcal{G}(\mathfrak{A}, \psi(\overline{a}))$ for a first-order formula ψ is finite if, and only if, $\mathfrak{A}$ is finite, but the game is well-founded in any case. In general however, games with finite game graphs need not be well-founded.

A game is called *determined* if from each position one of the players has a winning strategy, i.e., if $W_0 \cup W_1 = V$. Well-founded games are always determined, and so are large classes of more general games (such as games in the Borel hierarchy, see [11,12]).

We denote by GAME the strategy problem for games (with finite game graphs and positional winning conditions),

$$\text{GAME} = \{(\mathcal{G}, v) : \text{Player 0 has a winning strategy in } \mathcal{G} \text{ from position } v\}.$$

It is obvious that the GAME problem can be solved in polynomial time: Denote by W_i^n the set of positions from which Player i has a strategy to win the game in at most n moves. Then $W_i^0 = \{v \in V_{1-i} : vE = \emptyset\}$ is the set of winning terminal positions for Player i, and we can compute the sets W_i^n inductively by

$$W_i^{n+1} := \{v \in V_0 : vE \cap W_i^n \neq \emptyset\} \cup \{v \in V_1 : vE \subseteq W_i^n\}$$

until $W_i^{n+1} = W_i^n$.

To see that GAME can actually be solved in *linear time*, a little more work is necessary. The following algorithm is a variant of depth-first search, and computes the entire winning sets for both players in time $O(|V| + |E|)$.

Theorem 3.1 *Winning sets of finite games can be computed in linear time.*

Proof. We present an algorithm that computes for each position, which player, if any, has a winning strategy for the game starting at this position. During the computation three arrays are used:

- win$[v]$ contains either 0 or 1 indicating which player wins, or $\perp$ if we do not know yet, or if none of the players has a winning strategy from v;
- $P[v]$ contains the predecessors of v; and
- $n[v]$ is the number of those successors for which win$[v] = \perp$.

A linear time algorithm for the GAME problem

Input: A game $\mathcal{G} = (V, V_0, E)$

```
forall v ∈ V do                 (* 1: initialisation *)
    win[v] := ⊥
    P[v] := ∅
    n[v] := 0
enddo

forall (u, v) ∈ E do            (* 2: calculate P and n *)
    P[v] := P[v] ∪ {u}
    n[u] := n[u] + 1
enddo

forall v ∈ V_0                  (* 3: calculate win *)
    if n[v] = 0 then  Propagate(v, 1)
forall v ∈ V − V_0
```

```
    if n[v] = 0 then  Propagate(v, 0)
return win end

procedure Propagate(v, i)
    if win[v] ≠ ⊥ then  return
    win[v] := i                    (* 4: mark v as winning for Player i *)
    forall  u ∈ P[v] do            (* 5: propagate change to predecessors *)
        n[u] := n[u] − 1
        if u ∈ V_i or n[u] = 0 then  Propagate(u, i)
    enddo
end
```

The heart of this algorithm is the procedure Propagate(v, i) which is called any time we have found out that Player i has a winning strategy from position v. Propagate(v, i) records this fact and investigates whether we are now able to determine the winning player for any of the predecessors of v. This is done by applying the following rules:

- If the predecessor u belongs to Player i then this player has a winning strategy by moving to position v.
- If the predecessor u belongs to the opponent, win$[u]$ is undefined, and the winning player has already been determined for all of its successors w, then win$[w] = i$ for all those successors and Player i wins from u regardless of the choice of his opponent.

Since parts (4) and (5) are reached only once for each position v, the inner part of the loop in (5) is executed at most $\sum_v |P[v]| = |E|$ times. Therefore the running time of the algorithm is $O(|V| + |E|)$.

The correctness of the value assigned to win$[v]$ is proved by a straightforward induction on the number of moves in which the corresponding player can ensure that he wins. Note that the positions satisfying $n[v] = 0$ in Part 3 are exactly those without outgoing edges even if $n[v]$ is modified by Propagate.□

Game is known to be a Ptime-complete problem (see [7]). This remains the case for *strictly alternating games*, where $E \subseteq V_0 \times V_1 \cup V_1 \times V_0$. Indeed, any game can be transformed into an equivalent, strictly alternating one by introducing for each move $(u, v) \in V_i \times V_i$ a new node $e \in V_{1-i}$ and by replacing the move (u, v) by two moves (u, e) and (e, u).

The Game problem (sometimes also called *alternating reachability*) is a general combinatorial problem that reappears in different guises in many areas. To illustrate this by an example, we show that the satisfiability problem for propositional Horn formula is essentially the same problem as Game.

Satisfiability for Horn formulae. It is well-known that Sat-Horn, the satisfiability problem for propositional Horn formulae is

- Ptime-complete [7], and

- solvable in linear time [3,8].

Using the GAME problem we obtain very simple proofs for both results. Indeed, GAME and SAT-HORN are equivalent under log-lin reductions, i.e., reductions that are computable with linear time and logarithmic space. The reductions are so simple that we can say that GAME and SAT-HORN are really the same problem.

Theorem 3.2 SAT-HORN *is log-lin equivalent to* GAME.

Proof. GAME $\leq_{\log-\mathrm{lin}}$ SAT-HORN: Given a finite game graph $\mathcal{G} = (V, V_0, E)$ we can construct in time $O(|V| + |E|)$ 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 for Player 0. Hence $v \in W_0$ if the Horn formula $\psi_{\mathcal{G}} \wedge (0 \leftarrow v)$ is unsatisfiable.

SAT-HORN $\leq_{\log-\mathrm{lin}}$ GAME: Given a Horn formula $\psi(X_1, \ldots, X_n) = \bigwedge_{i \in I} C_i$ with propositional variables $X_1, \ldots, X_n$ and Horn clauses C_i of form $H_i \leftarrow X_{i_1} \wedge \cdots \wedge X_{i_m}$ (where the head of the clause, H_i, is either a propositional variable or the constant 0) we define a game $\mathcal{G}_\psi$ as follows. The positions of Player 0 are the initial position 0 and the propositional variables $X_1, \ldots, X_n$, the positions of Player 1 are the clauses of ψ. Player 0 can move from a position X to any clause C with head X, and Player 1 can move from a clause C to any variable occurring in the body of C. Formally $\mathcal{G}_\psi = (V, E)$, $V = V_0 \cup V_1$ with $V_0 = \{0\} \cup \{X_1, \ldots, X_n\}$, $V_1 = \{C_i : i \in I\}$, and

$$E = \{(X, C) \in V_0 \times V_1 : X = \mathrm{head}(C)\} \cup \{(C, X) \in V_1 \times V_0 : X \in \mathrm{body}(C)\}.$$

Player 0 has a winning strategy for $\mathcal{G}_\psi$ from position X if, and only if, $\psi \models X$. In particular, ψ is unsatisfiable if, and only if, Player 0 wins from position 0.$\Box$

4 Complexity of first-order model checking.

Roughly, the size of the model checking game $\mathcal{G}(\mathfrak{A}, \psi)$ is the number of different instantiations of the subformulae of ψ with elements from $\mathfrak{A}$. To measure the size of games, and the resulting time and space bounds for model checking complexity as precisely as possible, we use, besides *formula length* $|\psi|$, the following parameters. The *closure* $\mathrm{cl}(\psi)$ is the set of all subformulae of ψ. Obviously, $|\mathrm{cl}(\psi)| \leq |\psi|$ and in some cases $|\mathrm{cl}(\psi)|$ can be much smaller than $|\psi|$. The *quantifier rank* $\mathrm{qr}(\psi)$ is the maximal nesting depth of quantifiers in ψ, and the *width* of ψ is the maximal number of free variables in subformulae, i.e.,

$$\mathrm{width}(\psi) = \max\{|\mathrm{free}(\varphi)| : \varphi \in \mathrm{cl}(\psi)\}.$$

Instead of considering the width, one can also rewrite formulae with as few variables as possible.

Lemma 4.1 *A first-order formula ψ has width k if, and only if, it is equivalent, via a renaming of bound variables, to a first-order formula with at most k distinct variable symbols.*

Bounded-variable fragments of logics have received a lot of attention in finite model theory. However, we here state results in terms of formula width rather than number of variables to avoid the necessity to economise on the number of variables.

In many cases it is not the best method to construct the full model checking game explicitly and then solve the strategy problem, since many positions of the game will not really be needed. Further, as alternating algorithms are really games, it is not surprising that the best estimates for the complexity of model checking games often are in terms of alternating complexity classes. We present an alternating model checking algorithm for first-order logic that can be viewed as an on-the-fly construction of the model checking game while playing it.

Theorem 4.2 *There is an alternating model checking algorithm that, given a finite structure $\mathfrak{A}$ and a first-order sentence ψ, decides whether $\mathfrak{A} \models \psi$ in time $O(|\psi| + \mathrm{qr}(\psi)\log|A|)$ and space $O(|\psi| + \mathrm{width}(\psi)\log|A|)$ (assuming that atomic statements are evaluated in constant time).*

Proof. We present a recursive alternating procedure **ModelCheck**$(\mathfrak{A}, \rho, \psi)$, which given $\mathfrak{A}$, a formula ψ that may contain free variables, and an assignment $\rho : \mathrm{free}(\psi) \to \mathfrak{A}$, decides whether $\mathfrak{A} \models \psi[\rho]$.

ModelCheck$(\mathfrak{A}, \rho, \psi)$

Input: a first-order formula ψ in negation normal form,
  a finite structure $\mathfrak{A}$ (with universe A),
  an assignment $\rho : \mathrm{free}(\psi) \to A$
if ψ is an atom or negated atom **then**
  if $\mathfrak{A} \models \psi[\rho]$ **accept else reject**
if $\psi = \eta \vee \vartheta$ **then do**
  guess $\varphi \in \{\eta, \vartheta\}$, and let $\rho' := \rho\,|_{\mathrm{free}(\varphi)}$
  ModelCheck$(\mathfrak{A}, \rho', \varphi)$
if $\psi = \eta \wedge \vartheta$ **then do**
  universally choose $\varphi \in \{\eta, \vartheta\}$, and let $\rho' := \rho\,|_{\mathrm{free}(\varphi)}$
  ModelCheck$(\mathfrak{A}, \rho', \varphi)$
if $\psi = \exists x \varphi$ **then do**
  guess an element a of $\mathfrak{A}$
  ModelCheck$(\mathfrak{A}, \rho[x \mapsto a], \varphi)$
if $\psi = \forall x \varphi$ **then do**
  universally choose an element a of $\mathfrak{A}$
  ModelCheck$(\mathfrak{A}, \rho[x \mapsto a], \varphi)$

A straightforward induction shows that the procedure is correct. The time needed by the procedure is the depth of the syntax tree of ψ plus the time needed to produce the variable assignments. On each computation path, at most $\mathrm{qr}(\psi)$ elements of $\mathfrak{A}$ have to be chosen and each element needs $\log|A|$ bits. Hence the time complexity is $O(|\psi| + \mathrm{qr}(\psi)\log|A|)$. During the evaluation the algorithm needs to maintain a pointer to the current position in ψ and to store the current assignment, which requires $\mathrm{free}(\varphi)\log|A|$ bits for the current subformula φ. Hence the space needed by the algorithm is $O(|\psi| + \mathrm{width}(\psi)\log|A|)$. □

Theorem 4.3 *The model checking problem for first-order logic is* PSPACE-*complete. For any fixed $k \geq 2$, the model checking problem for first-order formulae of width at most k is* PTIME-*complete.*

Proof. Membership in these complexity classes follows immediately from Theorem 4.2 via the facts that alternating polynomial time coincides with polynomial space and alternating logarithmic space coincides with polynomial time.

Completeness follows by straightforward reductions from known complete problems. QBF, the evaluation problem for quantified Boolean formulae, reduces to first-order model checking on the fixed structure (A, P) with $A = \{0, 1\}$ and $P = \{1\}$. Given a quantified Boolean formula ψ, without free propositional variables, translate it into a first-order sentence ψ as follows: replace every quantification $\exists X_i$ or $\forall X_i$ over a propositional variable X_i by a corresponding first-order quantification $\exists x_i$ or $\forall x_i$ and replace atomic propositions X_i by atoms Px_i. Obviously ψ evaluates to *true* if, and only if, $(A, P) \models \varphi'$. This proves that the expression complexity and the combined complexity of first-order model checking is PSPACE-complete.

To see that the model-checking problem for first-order formulae of width two is PTIME-complete, we reduce to it the GAME-problem for strictly alternating games, with Player 0 moving first. For $n \in \mathbb{N}$ we construct inductively formulae ψ_n by

$$\begin{aligned} \psi_0(x) &:= \forall y \neg Exy \\ \psi_{2n+1}(x) &:= \exists y(Exy \wedge \psi_{2n}(y)) \\ \psi_{2n+2}(x) &:= \forall y(Exy \rightarrow \psi_{2n+1}(y)). \end{aligned}$$

Obviously ψ_n has width two, and for any strictly alternating game $\mathcal{G} = (V, V_0, E)$, Player 0 can win in n moves from position $v \in V_0$ (for odd n) or $v \in V_1$ (for even n) if, and only if, $\mathcal{G} \models \psi_n(v)$. Now, if Player 0 has a winning strategy from $v \in V_0$, then she also has one for winning in at most n moves, where $n = |V|$, since otherwise the game will be caught in a loop. Hence any instance $\mathcal{G}, v$ for the GAME-problem (for strictly alternating games with $v \in V_0$) can be reduced to the instance $\mathcal{G}, \psi_n(v)$ of the model checking problem for first-order formulae of width two. □

The argument for PTIME-completeness applies in fact already to *propositional modal logic* ML [6]. Instead of the formulae $\psi_n(x)$ constructed above, take the modal formulae defined by $\varphi_0 := \Box \mathit{false}$, $\varphi_{2n+1} = \Diamond\psi_{2n}$, and $\varphi_{2n+2} := \Box\varphi_{2n+1}$.

Corollary 4.4 *The model checking problem for* ML *is* PTIME*-complete.*

If we consider a fixed formula ψ, Theorem 4.2 tells us that the data complexity of first-order logic is much lower than the expression or combined complexity.

Corollary 4.5 *For every first-order sentence ψ, $\{\mathfrak{A} : \mathfrak{A} \text{ finite}, \mathfrak{A} \models \psi\} \in$ ALOGTIME. In particular, the evaluation problem for any fixed first-order sentence can be computed deterministically with logarithmic space.*

5 Least fixed-point logics

Fixed point logics extend a basic logical formalism (like first-order logic, conjunctive queries, or propositional modal logic) by a constructor for forming *fixed points of relational operators.* Note that any formula $\psi(R, \overline{x})$, of vocabulary $\tau \cup \{R\}$ and with a tuple of variables $\overline{x}$ whose length matches the arity of R, defines, for every τ-structure $\mathfrak{A}$, an update operator $F_\psi : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ on the class of k-ary relations on A, namely

$$F_\psi : R \mapsto \{\overline{a} : (\mathfrak{A}, R) \models \psi(R, \overline{a})\}.$$

If R happens to occur only positive in ψ, then the operator F_ψ is monotone, and in that case it has a *least fixed point* $\mathbf{lfp}(F_\psi)$ and a *greatest fixed point* $\mathbf{gfp}(F_\psi)$. Least and greatest fixed points can also be constructed inductively. The *stages* of an operator F are the sets X^α (α an ordinal), defined by

$$\begin{aligned} X^0 &:= \emptyset, \\ X^{\alpha+1} &:= F(X^\alpha), \text{ and} \\ X^\lambda &:= \bigcup_{\alpha<\lambda} X^\alpha \text{ for limit ordinals } \lambda. \end{aligned}$$

The sequence of stages of a monotone operator is increasing and hence eventually reaches a fixed point, which coincides with the least fixed point. The greatest fixed point can be constructed by a dual induction, starting with $Y^0 = A^k$, setting $Y^{\alpha+1} := F(Y^\alpha)$, and $Y^\lambda = \bigcap_{\alpha<\lambda} Y^\alpha$ for limit ordinals. The decreasing sequence of these stages then eventually converges to the greatest fixed point $Y^\infty = \mathbf{gfp}(F)$.

Lemma 5.1 *Let $F : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ be a monotone operator over a finite set A. If F is computable in polynomial time (w.r.t. $|A|$), then so are the fixed points* $\mathbf{lfp}(F)$ *and* $\mathbf{gfp}(F)$.

Least fixed point logic, denoted LFP, is the logic defined by adding to the syntax of first order logic the following *least fixed point formation rule:* If $\psi(R, \overline{x})$ is a formula of vocabulary $\tau \cup \{R\}$ with only positive occurrences of R, if $\overline{x}$ is a tuple of variables and $\overline{t}$ a tuple of terms (such that the lengths of $\overline{x}$ and $\overline{t}$ match the arity of R), then

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

are formulae of vocabulary τ. The free first-order variables of these formulae are those in $(\text{free}(\psi) - \{x : x \text{ in } \overline{x}\}) \cup \text{free}(\overline{t})$.

Semantics. For any τ-structure $\mathfrak{A}$, providing interpretations for all free variables in the formula, we have that $\mathfrak{A} \models [\mathbf{lfp}\, R\overline{x}\ .\ \psi](\overline{t})$ if $\overline{t}^{\mathfrak{A}}$ (the tuple of elements of $\mathfrak{A}$ interpreting $\overline{t}$) is contained in $\mathbf{lfp}(F_\psi)$, where F_ψ is the update operator defined by ψ on $\mathfrak{A}$. Similarly for greatest fixed points.

Example 5.2 Here is a fixed-point formula that defines the transitive closure of the binary predicate E:

$$\text{TC}(u, v) := [\mathbf{lfp}\, Txy\ .\ Exy \vee \exists z(Exz \wedge Tzy)](u, v).$$

Note that in a formula $[\mathbf{lfp}\, R\overline{x}\ .\ \varphi](\overline{t})$ there may be additional free variables in φ besides those in $\overline{x}$, and these remain free in the fixed point formula. They are often called *parameters* of the fixed point formula. For instance, the transitive closure can also be defined by the formula

$$\varphi(u, v) := [\mathbf{lfp}\, Ty\ .\ Euy \vee \exists x(Tx \wedge Exy)](v)$$

which has u as parameter. It is easy to see that every LFP-formula is equivalent to one without parameters (at the expense of increasing the arity of fixed point variables).

Example 5.3 The GAME problem is LFP-definable, by $[\mathbf{lfp}\, Wx\ .\ \varphi](x)$ with

$$\varphi(W, x) := (V_0 x \wedge \exists y(Exy \wedge Wy)) \vee (\neg V_0 \wedge \forall y(Exy \rightarrow Wy)).$$

GAME plays an important role for LFP. It can be shown, that every LFP-definable problem can be reduced to GAME by a quantifier-free translation. Hence GAME is complete for LFP via this notion of reduction, and thus a natural candidate if one tries to separate a weaker logic from LFP.

A more general variant of LFP permits simultaneous inductions over several formulae.

Let $\psi_1(\overline{R}, \overline{x}_1), \ldots, \psi_m(\overline{R}, \overline{x}_m)$ be formulae of vocabulary $\tau \cup \{R_1, \ldots, R_m\}$, with only positive occurrences of $R_1, \ldots, R_m$, and let, for each $i \leq m$, $\overline{x}_i$ be a sequence of variables matching the arity of R_i. Then

$$S := \begin{cases} R_1\overline{x}_1 & := \psi_1 \\ & \vdots \\ R_m\overline{x}_m & := \psi_m \end{cases}$$

is a *system of update rules*, which is used to build formulae $[\mathbf{lfp}\ R_i : S](\overline{t})$ and $[\mathbf{gfp}\ R_i : S](\overline{t})$ (for any tuple $\overline{t}$ of terms whose length matches the arity of R_i).

Semantics: S defines a monotone operator $S^{\mathfrak{A}} = (S_1, \dots, S_m)$ mapping tuples $\overline{R} = (R_1, \dots, R_m)$ of relations on A to $S^{\mathfrak{A}}(\overline{R}) = (S_1(\overline{R}), \dots, S_m(\overline{R}))$ where $S_i(\overline{R}) := \{\overline{a} : (\mathfrak{A}, \overline{R}) \models \psi_i(\overline{R}, \overline{a})\}$. As the operator is monotone, it has a least fixed point $\mathbf{lfp}(S^{\mathfrak{A}}) = (R_1^\infty, \dots, R_m^\infty)$. Now $\mathfrak{A} \models [\mathbf{lfp}\, R_i : S](\overline{a})$ if $\overline{a} \in R_i^\infty$. Similarly for greatest fixed points.

While simultaneous fixed points do not provide more expressive power than simple fixed points, they permit to write formulae in a more modular and more readable form.

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

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

where $\psi[R/\neg R]$ is the formula obtained from ψ by replacing all occurrences of R-atoms by their negations. (As R occurs only positive in ψ, the same is true for $\neg\psi[R/\neg R]$). Because of this duality, greatest fixed points are often omitted in the definition of LFP. On the other hand, it is sometimes convenient to keep the greatest fixed points, and use the duality (and de Morgan's laws) to translate LFP-formulae into *negation normal form*, i.e., to push negations all the way to the atoms.

From the fact that first-order operations are polynomial-time computable, and from Lemma 5.1, we immediately conclude that every LFP-definable property of finite strucures is computable in polynomial time.

Proposition 5.4 *Let ψ be in* LFP. *It is decidable in polynomial time whether a given finite structure $\mathfrak{A}$ is a model of ψ. In short:* LFP $\subseteq$ PTIME.

The modal μ-calculus. A fragment of LFP that is of fundamental importance in many areas of computer science (e.g., in controller synthesis, hardware verification, and knowledge representation) is the modal μ-calculus L_μ. It is obtained by adding least and greatest fixed points to propositional modal logic ML. In other words L_μ relates to ML in the same way as LFP relates to FO.

Definition 5.5 The *modal μ-calculus* L_μ extends ML (including propositional variables $X, Y, \dots$, which can be be viewed as monadic second-order variables) by the following rule for building fixed point formulae:

If ψ is a formula in L_μ and X is a propositional variable that only occurs positively in ψ, then also $\mu X.\psi$ and $\nu X.\psi$ are L_μ-formulae.

The semantics of these fixed point formulae is completely analogous to the one for LFP. On a transition system G (with universe V, and with interpretations for other free second-order variables that ψ may have besides X) ψ defines the monotone operator $F_\psi : \mathcal{P}(V) \to \mathcal{P}(V)$ assigning to every set

$X \subseteq V$ the set $\psi^G(X) := \{v \in V : (G, X), v \models \psi\}$. Now

$$G, v \models \mu X.\psi \text{ iff } v \in \mathbf{lfp}(F_\psi)$$
$$G, v \models \nu X.\psi \text{ iff } v \in \mathbf{gfp}(F_\psi)$$

As for LFP, one can also allow simultaneous fixed point formulae $(\mu X \;.\; S)$ and $(\nu X \;.\; S)$ for systems S of update rules, without changing the expresive power.

Although ML and L_μ are defined as extensions of propositional logic, it is often more convenient to view them as fragments of FO and LFP, respectively. Indeed a modal formula ψ defines a query on transition systems, associating with G that set of nodes $[\![\psi]\!]^G = \{v : G, v \models \psi\}$, and this set can equivalently be defined by an FO- or LFP-formula $\psi^*(x)$. This translation takes an atomic proposition P_b to the atom $P_b x$, it commutes with the Boolean connectives, and it translates the modal operators by quantifiers as follows:

$$(\langle a \rangle \psi)^*(x) := \exists y (E_a xy \wedge \psi^*(y))$$
$$([a]\psi)^*(x) := \forall y (E_a xy \rightarrow \psi^*(y)).$$

Finally, a formula of form $\mu X.\varphi$ is translated to $[\mathbf{lfp}\, Xx \;.\; \varphi^*](x)$, and similarly for greatest fixed points. Note that the resulting formula has width two and can thus be written with only two first-order variables.

Proposition 5.6 *For every formula $\psi \in L_\mu$ there exists a formula $\psi^*(x) \in$ LFP of width two, which is equivalent to ψ in the sense that $G, v \models \psi$ iff $G \models \psi^*(v)$.*

Let us turn to algorithmic issues. The complexity of the model checking problem for L_μ is a major open problem, as far as combined complexity and expression complexity are concerned (see Section 6). However, the data complexity can be settled easily.

Proposition 5.7 (Data complexity of L_μ) *Fix any formula $\psi \in L_\mu$. Given a finite transition system G and a node v it can be decided in polynomial time whether $G, v \models \psi$. Further there exist $\psi \in L_\mu$ for which the model checking problem is* PTIME*-complete.*

Proof. As L_μ is a fragment of LFP, the first claim is obvious. For the second claim, recall that the GAME-problem for strictly alternating games is PTIME-complete. Player 0 has a winning strategy from position $v \in V_0$ in the game $G = (V, V_0, E)$ if, and only if, $G, v \models \mu X.\Diamond\Box X$. $\Box$

For more information on the μ-calculus, see [1,2] and the references there.

6 Model checking games for least fixed point logic

For least fixed point logics, the appropriate evaluation games are *parity games*. These are games of possibly infinite duration 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. Practically competitive model checking algorithms for the modal μ-calculus work by solving the strategy problem for the associated parity game (see, e.g.,[10]).

Definition 6.1 A *parity game* is given by a labelled graph $\mathcal{G} = (V, V_0, E, \Omega)$ where (V, V_0, E) is a game graph as in Section 2 and $\Omega : V \to \mathbb{N}$ assigns to each position a *priority*. The number of different priorities used in the game is the *index* of $\mathcal{G}$. Recall that a finite play of a game is lost by the player who gets stuck, i.e., cannot move. The difference with the games of Section 2 is that we have different winning conditions for infinite plays $v_0 v_1 v_2 \ldots$. If the least number appearing infinitely often in the sequence $\Omega(v_0)\Omega(v_1)\ldots$ of priorities is even, then Player 0 wins the play, otherwise Player 1 wins.

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. For proofs, the reader may consult [16] or [13].

Theorem 6.2 (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.*

Theorem 6.3 *It can be decided in* NP $\cap$ Co-NP *whether a given position in a parity games is a winning position for Player 0.*

Proof. A node v in a parity game $\mathcal{G} = (V, V_0, E, \Omega)$ is a winning position for Player i if there exists a positional strategy $f : V_i \to V$ which is winning from position v. It therefore suffices to show that the question whether a given $f : V_i \to V$ is a winning strategy for Player i from position v can be decided in polynomial time. We prove this for Player 0; the argument for Player 1 is analogous.

Given $\mathcal{G}$ and $f : V_0 \to V$ we obtain a reduced game graph $\mathcal{G}_f = (V, E_f)$ by keeping only the moves that are consistent with f, i.e.,

$$E_f = \{(v, w) : (v \in V_i \wedge w = f(v)) \vee (v \in V_{1-i} \wedge (v, w) \in E)\}.$$

In this reduced game, only the opponent, Player 1, makes non-trivial moves. We call a cycle in (V, E_f) odd if the least priority of its nodes is odd. Clearly, Player 0 wins $\mathcal{G}$ from position v via strategy f if, and only if, in $\mathcal{G}_f$ no odd cycle and no terminating position $w \in V_0$ are reachable from v. Since the reachability problem is solvable in polynomial time, the claim follows. $\square$

In fact, Jurdziński [9] proved that the problem is in UP $\cap$ Co-UP where UP denotes the class of NP-problems with unique witnesses. 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 [10].

Theorem 6.4 *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*

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

The unfolding of a parity game. Let $\mathcal{G} = (V, V_0, E, \Omega)$ be a parity game. We assume that the minimal priority in the range of Ω is even, and that every node v with minimal priority has a unique successor $s(v)$ (i.e., $vE = \{s(v)\}$). This is no loss of generality. We can always tranform a parity game in such a way that all nodes with non-maximimal priority have unique successors (i.e., choices are made only at the least relevant nodes). If the least priority in the game is odd we consider instead the dual game (with switched roles of the players, and priorities decreased by one).

Let T be the set of nodes with minimal priority and let $\mathcal{G}^-$ be the game obtained by deleting from $\mathcal{G}$ all edges $(v, s(v)) \in T \times V$ so that the nodes in T become terminal positions. We will define the *unfolding* of $\mathcal{G}$ as a sequence of games $\mathcal{G}^\alpha$ (where α ranges over the ordinals) which all coincide with $\mathcal{G}^-$ up to the winning conditions for the terminal positions $v \in T$. For every α, we define a decomposition $T = T_0^\alpha \cup T_1^\alpha$ where T_i^α is the set of $v \in T$ where we declare, for the game $\mathcal{G}^\alpha$, Player i to be the winner [2]. Further, for every α, we write W_i^α for the winning set of Player i in the game $\mathcal{G}^\alpha$. Note that W_i^α depends of course on the decomposition $T = T_0^\alpha \cup T_1^\alpha$ (also concerning positions outside T). In turn, the decomposition of T for $\alpha + 1$ depends on the winning sets W_i^α in $\mathcal{G}^\alpha$. We set

$$\begin{aligned}
T_0^0 &:= T \\
T_0^{\alpha+1} &:= \{v \in T : s(v) \in W_0^\alpha\} \\
T_0^\lambda &:= \bigcap_{\alpha<\lambda} T_0^\alpha \text{ for limit ordinals } \lambda.
\end{aligned}$$

By determinacy, $V = W_0^\alpha \cup W_1^\alpha$ for all α, and with increasing α, the winning sets of Player 0 are decreasing and the winning sets of Player 1 are increasing: $W_0^0 \supseteq W_0^1 \supseteq \cdots W_0^\alpha \supseteq W_0^{\alpha+1} \supseteq \cdots$ and $W_1^0 \subseteq W_1^1 \subseteq \cdots W_1^\alpha \subseteq W_1^{\alpha+1} \subseteq \cdots$ Hence there exists an ordinal α (whose cardinality is bounded by the cardinality of V) with $W_0^\alpha = W_0^{\alpha+1} =: W_0^\infty$ and $W_1^\alpha = W_1^{\alpha+1} =: W_1^\infty$. We

[2] Technically this means that in $\mathcal{G}^\alpha$, v is a position in V_{1-i}, so that the opponent of Player i has to move from v, but cannot, and hence loses.

claim that these fixed points coincide with the winning sets W_0 and W_1 for the original game $\mathcal{G}$.

Lemma 6.5 (Unfolding Lemma) $W_0 = W_0^\infty$ *and* $W_1 = W_1^\infty$.

Proof. It suffices to define a strategy f for Player 0 and a strategy g for Player 1 for the game $\mathcal{G}$, by means of which Player i wins from all positions $v \in W_i^\infty$.

Fix first a winning strategy f^α for Player 0 in $\mathcal{G}^\alpha$, with winning set $W_0^\alpha = W_0^\infty$. Note that f^α trivially extends to a strategy f for the game $\mathcal{G}$, since the nodes in T have unique successors in $\mathcal{G}$. We claim that f is in fact a winning strategy in $\mathcal{G}$ from all positions $v \in W_0^\alpha$.

To see this, consider any play $v_0v_1v_2\dots$ in $\mathcal{G}$ from position $v_0 \in W_0^\alpha$ against f. Such a play can never leave W_0^α. Indeed if $v_i \in W_0^\alpha \setminus T$ then $v_{i+1} \in W_0^\alpha$ because f is a winning strategy for $\mathcal{G}^\alpha$; and if $v_i \in W_0^\alpha \cap T = W_0^{\alpha+1} \cap T$, then $v_i \in T_0^{\alpha+1}$ which implies, by definition of $T_0^{\alpha+1}$ that $v_{i+1} = s(v_i) \in W_0^\alpha$. But a play that never leaves W_0^α is necessarily won by Player 0: Either it goes only finitely often through positions in T, and then it coincides from a certain point onwards with a winning play in $\mathcal{G}^\alpha$, or it goes infinitely often through positions in T in which case Player 0 wins because the minimal priority that is hit infinitely often is even.

To construct a winning strategy for Player 1 in the game $\mathcal{G}$, we define, for every node $v \in W_1^\infty$, the ordinal

$$\sigma(v) := \min\{\beta : v \in W_1^\beta\}.$$

Fix, for every ordinal α a winning strategy g^α for Player 1 with winning set W_1^α in the game $\mathcal{G}^\alpha$ and set

$$g(v) := g^{\sigma(v)}(v) \text{ for all } v \in V_1 \setminus T$$

and $g(v) := s(v)$ for $v \in V_1 \cap T$.

Consider any play $v_0v_1v_2\dots$ in $\mathcal{G}$ from position $v_0 \in W_1^\infty$ against g. We claim that whenever $v_i \in W_1^\infty$, then

(1) $v_{i+1} \in W_1^\infty$,

(2) $\sigma(v_{i+1}) \le \sigma(v_i)$, and

(3) if $v_i \in T$, then $\sigma(v_{i+1}) < \sigma(v_i)$.

Indeed, if $v_i \in W_1^\infty \setminus T$ and $\sigma(v_i) = \alpha$, then $v_i \in W_1^\alpha$ and therefore (since Player 1 moves locally according to his winning strategy g^α and Player 0 cannot leave winning sets of her opponent), $v_{i+1} \in W_1^\alpha$. But if $v_i \in W_1^\infty \cap T$ and $\sigma(v_i) = \alpha$, then $v_i \in T_1^\alpha$, $\alpha = \beta + 1$ is a successor ordinal, and $v_{i+1} = s(v_i) \in W_1^\beta$ (by definition of T_1^α). Hence $\sigma(v_{i+1}) \le \beta < \sigma(v_i)$.

Properties (1), (2), and (3) imply that the play stays inside W_1^∞ and that the values $\sigma(v)$ are decreasing. Since there are no infinite strictly descending chains of ordinals, the play eventually remains inside W_1^α, for a fixed α, and outside T (since moves from T would reduce the value of $\sigma(v)$). Hence the play eventually coincides with a play in $\mathcal{G}^\alpha$ where Player 1 plays according to his winning strategy g^α. Thus, Player 1 wins. $\square$

Game semantics for least fixed point formulae. For defining evaluation games for LFP-formulae and analysing the complexity of model checking it is convenient to make the following assumptions. First, the fixed point formulae should not contain parameters (the reason for this will be discussed below). Second, the formula should be in negation normal form, i.e., negations apply to atoms only, and third, it should be *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 $[\mathbf{fp}\, T\overline{x}\,.\,\varphi(T,\overline{x})]$ (where **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 T 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 from T. The *alternation depth* $\mathrm{ad}(\psi)$ of a fixed point sentence ψ is the maximal alternation level of its fixed point variables.

Consider now a structure $\mathfrak{A}$ and an LFP-formula $\psi(\overline{x})$ which we assume to be well-named, in negation normal form, and without parameters. The model checking game $\mathcal{G}(\mathcal{A},\psi(\overline{a}))$ is a parity game. As in the case of first-order logic, the positions of the game are expressions $\varphi(\overline{b})$, i.e., subformulae of ψ that are instantiated by elements of $\mathfrak{A}$. The initial position is $\psi(\overline{a})$. The moves are as in the first-order game, except for the positions associated with fixed point formulae and with fixed point atoms. At such 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 $[\mathbf{fp}\ T\overline{x}\ .\ \varphi(T,\overline{x})](\overline{y})$. From position $[\mathbf{fp}\, T\overline{x}\ .\ \varphi(T,\overline{x})](\overline{b})$ Falsifier moves to $\varphi(T,\overline{b})$, and from any fixed point atom $T\overline{c}$ he moves to the position $\varphi(T,\overline{c})$.

Hence in the case where we do not have fixed points the game is the usual Hintikka game for first-order logic. Next, we consider the case of a formula with just one occurrence of a fixed point operator which is an **lfp**. The intuition is that from position $[\mathbf{lfp}\ T\overline{x}\ .\ \varphi(T,\overline{x})](\overline{b})$ Verifier tries to establish that $\overline{b}$ enters T at some stage α of the fixed point induction that is defined by φ on $\mathfrak{A}$. The game goes to $\varphi(T,\overline{b})$ and from there, as φ is a first-order formula, Verifier can either win the φ-game in a finite number of steps, or she can force it to a position $T\overline{c}$ where $\overline{c}$ enters the fixed point at some stage $\beta < \alpha$. The game then resumes at position $\varphi(\overline{c})$. 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 $T\overline{c}$. 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 $T\overline{c}$, so **gfp**-atoms are assigned priority 0.

In the general case, we have a formula ψ with nested least and greatest fixed points and during an infinite play of $\mathcal{G}(\mathfrak{A}, \psi(\overline{a}))$ 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. To achieve this, we adjust 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**-variable (a **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. The priority labelling Ω on positions of $\mathcal{G}(\mathfrak{A}, \psi)$ is then defined by $\Omega(\varphi, \rho) = \mathrm{al}^*_\psi(T)$, if $\varphi = T\overline{x}$ and T is a fixed point variable, and $\Omega(\varphi, \rho) = \mathrm{ad}^*(\psi)$ otherwise.

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

Theorem 6.6 *Let $\psi(\overline{x})$ be a well-named and parameter-free* LFP*-formula in negation normal form, and let $\mathfrak{A}$ be a relational structure. $\mathfrak{A} \models \psi(\overline{a})$ if, and only if, Player 0 has a winning strategy for the parity game $\mathcal{G}(\mathfrak{A}, \psi(\overline{a}))$.*

Proof. This is proved by induction on ψ. The interesting case concerns fixed point formulae $\psi(\overline{x}) := [\mathbf{gfp}\, T\overline{x} \,.\, \varphi(\overline{x})](\overline{x})$.

In the game $\mathcal{G}(\mathcal{A}, \psi(\overline{a}))$ the positions of minimal priority are the fixed point atoms $T\overline{b}$, which have unique successors $\varphi(\overline{b})$. By induction hypothesis we know that, for every interpretation T_0 of T, $(\mathfrak{A}, T_0) \models \varphi(\overline{a})$ if, and only if, Player 0 has a winning strategy for $\mathcal{G}((\mathfrak{A}, T_0), \varphi(\overline{a}))$. By the unfolding of greatest fixed points, we also know that $\mathfrak{A} \models [\mathbf{gfp}\, T\overline{x} \,.\, \varphi(\overline{x})](\overline{a})$ if $(\mathfrak{A}, T^\alpha) \models \varphi(\overline{a})$ for all approximations T^α.

By ordinal induction one immediately sees that the games $\mathcal{G}((\mathfrak{A}, T^\alpha), \varphi(\overline{a}))$ coincide with the unfolding of the game $\mathcal{G} = \mathcal{G}(\mathfrak{A}, \psi(\overline{a}))$ to the games $\mathcal{G}^\alpha$. By the Unfolding Lemma we conclude that Player 0 wins the game $\mathcal{G}(\mathfrak{A}, \psi(\overline{a}))$ iff she wins all games $\mathcal{G}^\alpha$ iff $(\mathfrak{A}, T^\alpha) \models \varphi(\overline{a})$ for all α iff $\mathfrak{A} \models \psi(\overline{a})$.

For least fixed point formulae we proceed by dualization. □

Clearly, the size of the game $\mathcal{G}(\mathfrak{A}, \psi(\overline{a}))$, (and the time complexity for its construction) is bounded by $|\mathrm{cl}(\psi)| \cdot |A|^{\mathrm{width}(\psi)}$. Hence for LFP-formulae of bounded width, the size of the game is polynomially bounded.

Corollary 6.7 *The model checking problem for* LFP*-formulae of bounded width is in* NP $\cap$ Co-NP, *in fact in* UP $\cap$ Co-UP.

By Theorem 6.4 we obtain the following deterministic complexity bounds for LFP model checking.

Theorem 6.8 *Given a finite structure $\mathfrak{A}$ and a formula $\psi(\overline{a})$ of width k and alternation depth d, it can be decided whether $\mathfrak{A} \models \psi(\overline{a})$ in space $O(d \cdot |\mathrm{cl}(\psi)| \cdot |A|^k)$ and time*

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

Corollary 6.9 *The model checking problem for* LFP*-formulae of bounded width and bounded alternation depth is solvable in polynomial time.*

As formulae of the μ-calculus can be viewed as LFP-formulae of width two, the same bounds apply to L_μ. (For a different approach to this problem, without mentioning games explicitly, see [15].) It is a well-known open problem whether the model checking problem for L_μ can be solved in polynomial time. In any case, we can show that the problem is PTIME-hard, even under severe restrictions.

Proposition 6.10 *The expression complexity for L_μ, even for formulae without modal operators (and hence on trivial structures), is* PTIME*-hard.*

Proof. The proof is (again) by reduction from GAME. Let $\mathcal{G} = (V, E)$ descibe a strictly alternating game, with $V = V_0 \cup V_1$, and $E \subseteq V_0 \times V_1 \cup V_1 \times V_0$. Then Player 0 wins $\mathcal{G}$ from position $u \in V$ iff the trivial structure consisting of a single point is a model of the formula $(\mu X_u \,.\, S)$ where S is the system of rules

$$X_v \leftarrow \bigvee_{(v,w)\in E} \bigwedge_{(w,z)\in E} X_z$$

where we have variables X_v for all $v \in V_0$. □

It is not difficult to see that if the model checking problem for L_μ can be solved in polynomial time, then the same is true for (parameter-free) LFP-formulae of width k, for any fixed $k \in \mathbb{N}$. Given a finite structure $\mathfrak{A} = (A, R_1, \ldots, R_m)$, with relations of R_i of arities $r_i \leq k$, let $G^k(\mathfrak{A})$ be the transition system with universe A^k, unary relations $R_i^* = \{(a_1, \ldots, a_k) : (a_1, \ldots, a_{r_i}) \in R_i\}$ and $I_{ij} = \{(a_1, \ldots, a_k) : a_i = a_j\}$, and binary relations $E_j = \{(\overline{a}, \overline{b}) : a_i = b_i \text{ for } i \neq j\}$ (for $j = 1, \ldots, k$) and $E_\sigma = \{(\overline{a}, \overline{b}) : b_i = a_{\sigma(i)} \text{ for } i = 1, \ldots, k\}$ for each substitution $\sigma : \{1, \ldots, k\} \to \{1, \ldots, k\}$. One can translate formulae $\psi \in$ LFP of width k into formulae $\psi^* \in L_\mu$ such that $\mathfrak{A} \models \psi(\overline{a})$ iff $G^k(\mathfrak{A}), \overline{a} \models \psi^*$. (See [6, pp. 110–111] for details.)

Fixed point formula with parameters. We imposed the condition that the fixed point formulae do not contain parameters. If parameters are allowed, then, at least with a naive definition of width, Corollary 6.7 is no longer true (unless UP = Pspace). The intuitive reason is that parameters allow us to 'hide' first-order variables in fixed-point variables. Indeed Dziembowski [4] proved that QBF, the evaluation problem for quantified Boolean formulae, can be reduced to evaluating LFP-formulae with two first-order variables (but an unbounded number of monadic fixed point variables) on a fixed structure with three elements. Hence the expression complexity of evaluating such formulae is Pspace-complete. A similar argument works for the case where also the number of fixed point variables is bounded, but the structure is not fixed (combined complexity rather than expression complexity). We remark that unwinding such LFP-formulae in infinitary logic produces formulae of unbounded width.

LFP-formulae of unbounded width. For LFP-formulae of unbounded width, Theorem 6.8 only gives an exponential time bound. Indeed this cannot be improved, even for very simple LFP-formulae [14].

Theorem 6.11 (Vardi) *The model checking problem for* LFP*-formulae (of unbounded width) is* Exptime*-complete, even for formulae with only one fixed-point operator, and on a fixed structure with only two elements.*

Acknowledgement. This survey is a preliminary form of a part of a larger text on finite model theory and complexity. I am grateful for comments, corrections, and other contributions to Dietmar Berwanger, Achim Blumensath, Anuj Dawar, Stephan Kreutzer, Antje Nowack, and Mariane Riss.

References

[1] Arnold, A. and D. Niwiński, "Rudiments of μ-calculus," North Holland, 2001.

[2] Bradfield, J. and C. Stirling, *Modal logics and mu-calculi*, in: J. Bergstra, A. Ponse and S. Smolka, editors, *Handbook of Process Algebra*, Elsevier, 2001 pp. 293–332.

[3] Dowling, W. F. and J. H. Gallier, *Linear-time algorithms for testing the satisfiability of propositional horn formulae.*, Journal of Logic Programming **1** (1984), pp. 267–284.

[4] Dziembowski, S., *Bounded-variable fixpoint queries are PSPACE-complete*, in: *10th Annual Conference on Computer Science Logic CSL 96. Selected papers*, Lecture Notes in Computer Science Nr. 1258, Springer, 1996 pp. 89–105.

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

[6] Grädel, E. and M. Otto, *On logics with two variables*, Theoretical Computer Science **224** (1999), pp. 73–113.

[7] Greenlaw, R., J. Hoover and W. Ruzzo, "Limits to Parallel Computation. P-Completeness Theory," Oxford University Press, 1995.

[8] Itai, A. and J. Makowsky, *Unification as a complexity measure for logic programming*, Journal of Logic Programming **4** (1987), pp. 105–117.

[9] Jurdziński, M., *Deciding the winner in parity games is in UP ∩ Co-UP*, Information Processing Letters **68** (1998), pp. 119–124.

[10] Jurdziński, M., *Small progress measures for solving parity games*, in: *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Proceedings*, Lecture Notes in Computer Science **1770** (2000), pp. 290–301.

[11] Martin, D., *Borel determinacy*, Annals of Mathematics **102** (1975), pp. 336–371.

[12] Thomas, W., *On the synthesis of strategies in infinite games*, in: *Proceedings of STACS 95*, Lecture Notes in Computer Science Nr. 900 (1995), pp. 1–13.

[13] Thomas, W., *Languages, automata, and logic*, in: G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages Vol. 3*, Springer, 1997 pp. 389–455.

[14] Vardi, M., *The complexity of relational query languages*, in: *Proceedings of the 14th ACM Symposium on the Theory of Computing*, 1982, pp. 137–146.

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

[16] Zielonka, W., *Infinite games on finitely coloured graphs with applications to automata on infinite trees*, Theoretical Computer Science **200** (1998), pp. 135–183.