ON GUARDED TRANSFORMATION IN THE MODAL MU-CALCULUS

FLORIAN BRUSE, OLIVER FRIEDMANN, AND MARTIN LANGE

Abstract. Guarded normal form requires occurrences of fixpoint variables in a μ-calculus-formula to occur under the scope of a modal operator. The literature contains guarded transformations that effectively bring a μ-calculus-formula into guarded normal form. We show that the known guarded transformations can cause an exponential blowup in formula size, contrary to existing claims of polynomial behaviour. We also show that any polynomial guarded transformation for μ-calculus-formulas in the more relaxed vectorial form gives rise to a polynomial solution algorithm for parity games, the existence of which is an open problem. We also investigate transformations between the μ-calculus, vectorial form and hierarchical equation systems, which are an alternative syntax for alternating parity tree automata.

1. Introduction

The modal μ-calculus $\mathcal{L}_\mu$, as introduced by Kozen [17], is a fundamental modal fixpoint logic which subsumes many other temporal [8, 7] and dynamic logics [17, 10]. The paper at hand is concerned with *guarded normal form* for the μ-calculus, or simply *guarded form*. A formula is guarded if every occurrence of any fixpoint variable is under the scope of a modal operator inside its defining fixpoint formula. For instance, the formula $\nu Y.\Diamond\mu X.p\vee\Diamond X$ is guarded, whereas $\nu Y.\mu X.(p\wedge Y)\vee\Diamond X$ is not: it is possible to pass through the syntax tree of the formula from the quantification for the variable Y down to on occurrence of that variable without traversing through a modal operator $\Diamond$ or $\Box$. This is not possible for the first formula given here.

Intuitively, guarded form ensures that in the evaluation of a formula in a transition system by fixpoint iteration, one proceeds along at least one transition between two iterations of the same variable. Guarded form is also very useful in procedures that check for satisfiability or validity of a set of formulas and handle fixpoint formulas by unfolding: guardedness synchronises the unfolding of all formulas in a set. Many constructions require formulas to be explicitly normalised in guarded form or assume that w.l.o.g., formulas can be brought into guarded form with polynomial overhead [16, 14, 20, 25, 21, 15]. Others can cope with unguarded formulas, but then their constructions require the solving of non-trivial decision problems [9]. Only few deal explicitly with unguarded formulas [11], but they require special tricks in order to handle unguardedness.

It has been known for quite a while that every $\mathcal{L}_\mu$-formula can effectively be transformed into an equivalent guarded formula. The first guarded transformation

This work was supported by the European Research Council under the European Community's Seventh Framework Programme [ERC grant agreement no 259267].

Preprint submitted to: *Logic Journal of the IGPL.*

routine—described by Banieqbal and Barringer, as well as Walukiewicz—explicitly rewrites Boolean subformulas into disjunctive or conjunctive normal form [2, 25]. Clearly, this increases the sizes of formulas exponentially in the worst case. Such a blowup may not be considered harmful for results concerning the expressive power of $\mathcal{L}_\mu$, but it clearly makes a difference for complexity-theoretic results. Kupferman et al. [20, Thm. 2.1] notice that the transformation into Boolean normal form is unnecessary and present an optimised variant of this guarded transformation procedure. They claim that it only involves a linear blowup, but this is not true. The problem lies in the very last statement of their proof: "... by definition $\varphi'(\lambda y.\varphi'(y)) \in \mathrm{Cl}(\lambda y.\varphi'(y)) \ldots$" While this is true, it is not true that if $\lambda y.\varphi'(y)$ is a subformula of φ—and hence in the Fischer-Ladner closure of φ—then also $\varphi[\varphi'(\lambda y.\varphi'(y))/\lambda y.\varphi'(y)]$ is in the closure of φ. Unfolding from the inside out—a principle that forms the core of the guarded transformation—produces exactly this kind of situation. Repeated application of this principle will, in the worst case, result in an exponential growth in the number of distinct subformulas.

Later, another guarded transformation procedure was given by Mateescu [21]. It turns out that it is practically the same algorithm as that given by Kupferman et al. earlier. However, Mateescu estimates it to create an exponential blowup in formula size when used naïvely. On the other hand, he claims that "... each fixpoint subformula ... will be duplicated only once ... and reused ... leading to $\ldots |t(\varphi)| \leq |\varphi|^2$." Again, this is a false observation because the replacement of certain variables by constants can duplicate subformulas. Thus, the formula sharing trick that Mateescu proposes as a solution, which is just a way of saying that the size measured in terms of number of subformulas shall only be quadratic, does not work and the blow-up is not just quadratic.

Neumann and Seidl study guarded transformation in the more general context of hierarchical equation systems [22] with monotone operators. The modal operators $\Box$ and $\Diamond$ are monotone, so hierarchical equation systems are a generalisation of the *vectorial form* that is sometimes used to put multiple nestings of least or greatest fixpoints of the same kind into one block [1]. Hierarchical equation systems can also be seen as an alternative syntax to present alternating parity automata. The semantics of $\mathcal{L}_\mu$ with vectorial form is simply given via simultaneous fixpoint definitions rather than parametric ones. The Bekić Lemma shows that this does not gain additional expressive power [3] but it may gain exponential succinctness because the only known transformations of $\mathcal{L}_\mu$ with vectorial form into $\mathcal{L}_\mu$ without incur an exponential blow-up in formula size. The same holds for hierarchical equation systems.

Neumann and Seidl even give a guarded transformation algorithm for equation systems that result from an $\mathcal{L}_\mu$-formula and claim that it is polynomial. This claim seems to be correct. However, the resulting equation system does not have the nice structure that equation systems corresponding to $\mathcal{L}_\mu$-formulas have. Thus, their guarded transformation for $\mathcal{L}_\mu$-formulas is polynomial, yet it does not produce equivalent guarded $\mathcal{L}_\mu$-formulas but only equivalent guarded equation systems. Translating these back into a guarded $\mathcal{L}_\mu$-formula incurs an exponential blowup, given current knowledge.

Here we study the problem of guarded transformation for the modal μ-calculus with the aim of correcting false claims found in the literature and providing bounds on the complexities of such transformations. The structure of the paper is as follows:

in Section 2 we introduce $\mathcal{L}_\mu$ and its syntactic extensions vectorial form and HES as well as several notions of guardedness. Then we briefly describe and analyse the guarded transformation procedures by Kupferman et al. and Mateescu, respectively that of Neumann and Seidl in Section 3 and show that it can produce formulas of at least exponential size (measured as the number of different subformulas). We also give additional upper bounds on the elimination of ε-transitions in alternating automata and on translations from a syntactic extension to flat $\mathcal{L}_\mu$. In Section 4 we show that guarded transformation for $\mathcal{L}_\mu$-formulas in vectorial form and for HES is hard, namely not easier than solving parity games. As mentioned above, we also show that unfolding vectorial form or an HES into a non-vectorial formula has the same lower complexity bound. This means that any polynomial algorithm for one of these problems would yield a polynomial algorithm for solving parity games, and this would settle a major and long-standing open problem. Finally, in Section 5 we discuss the consequences of this work for previously acclaimed results about $\mathcal{L}_\mu$ that can be found in the literature, we discuss the relation between alternating automata, vectorial form, and $\mathcal{L}_\mu$, and we sketch some open questions with possible routes of attack.

2. The Modal μ-Calculus

2.1. **Syntax and Semantics.** A *labeled transition system* (LTS) over a set of *action names* Σ and a set of *atomic propositions* $\mathcal{P}$ is a tuple $\mathcal{T} = (S, \to, \ell)$ where S is a set of *states*, $\to \subseteq S \times \Sigma \times S$ defines a set of *transitions* between states, labeled with action names, and $\ell : S \to 2^{\mathcal{P}}$ labels each state with the set of atomic propositions that are true in this state.

Let Σ and $\mathcal{P}$ be as above and let $\mathcal{V}$ be a set of variables. Formulas of the modal μ-calculus $\mathcal{L}_\mu$ in positive normal form are those that can be derived from φ in

$$\varphi \ ::= \ q \mid \overline{q} \mid X \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle a \rangle \varphi \mid [a]\varphi \mid \mu X.\varphi \mid \nu X.\varphi,$$

where $X \in \mathcal{V}$, $q \in \mathcal{P}$, and $a \in \Sigma$.

The operators μ and ν act as *binders* for the variables in a formula. A *free occurrence* of a variable X is therefore one that does not occur under the scope of such a binder. A *closed* formula is one that does not have any free variables. We write σ for either μ or ν.

Let $\mathrm{Sub}(\varphi)$ denote the set of *subformulas* of φ. Define the *size* of a formula φ as the number of its distinct subformulas, i.e. $|\varphi| := |\mathrm{Sub}(\varphi)|$. We assume all $\mathcal{L}_\mu$-formulas to be *well-named* in the sense that each variable is bound at most once. Hence, for every $\mathcal{L}_\mu$-formula there is a partial function $\mathrm{fp}_\varphi : \mathcal{V} \to \mathrm{Sub}(\varphi)$ which maps a variable X that is bound in φ by some operator $\sigma X.\psi$ to its *defining fixpoint formula* ψ.

As usual, we use the abbreviations $\mathtt{tt} = q \vee \overline{q}$ and $\mathtt{ff} = q \wedge \overline{q}$ for an arbitrary q. Given a fixpoint binder σ, we write $\hat{\sigma} = \mathtt{ff}$ if $\sigma = \mu$ and $\hat{\sigma} = \mathtt{tt}$ if $\sigma = \nu$.

We write $\varphi[\psi/X]$ to denote the formula that results from φ by replacing every free occurrence of the variable X in it with the formula ψ.

The *modal depth* md is the maximal nesting depth of modal operators in a formula, formally defined as follows.

$$\begin{aligned} \mathrm{md}(q) \;=\; \mathrm{md}(\overline{q}) \;=\; \mathrm{md}(X) \;&:=\; 0 \\ \mathrm{md}(\varphi \vee \psi) \;=\; \mathrm{md}(\varphi \wedge \psi) \;&:=\; \max\{\mathrm{md}(\varphi), \mathrm{md}(\psi)\} \\ \mathrm{md}(\langle a \rangle \varphi) \;=\; \mathrm{md}([a]\varphi) \;&:=\; 1 + \mathrm{md}(\varphi) \\ \mathrm{md}(\mu X.\varphi) \;=\; \mathrm{md}(\nu X.\varphi) \;&:=\; \mathrm{md}(\varphi) \end{aligned}$$

An important fragment of $\mathcal{L}_\mu$ that we consider later is the *propositional μ-calculus* $\mathcal{B}_\mu$. It consists of all $\varphi \in \mathcal{L}_\mu$ such that $\mathrm{md}(\varphi) = 0$. Hence, $\mathcal{B}_\mu$-formulas do not contain any subformulas of the form $\langle a \rangle \psi$ or $[a]\psi$. A formula is called *purely propositional* if it belongs to $\mathcal{B}_\mu$ and does not contain any fixpoint operators.

Formulas of $\mathcal{L}_\mu$ are interpreted in states of an LTS $\mathcal{T} = (S, \to, \ell)$. Let $\rho \colon \mathcal{V} \to 2^S$ be an environment used to interpret free variables. We write $\rho[X \mapsto T]$ to denote the environment which maps X to T and behaves like ρ on all other arguments. The semantics of $\mathcal{L}_\mu$ is given as a function $[\![\cdot]\!]$ mapping a formula to the set of states where it holds w.r.t. the environment.

$$\begin{aligned} [\![q]\!]^{\mathcal{T}}_\rho &= \{s \in S \mid q \in \ell(s)\} \\ [\![\overline{q}]\!]^{\mathcal{T}}_\rho &= \{s \in S \mid q \notin \ell(s)\} \\ [\![X]\!]^{\mathcal{T}}_\rho &= \rho(X) \\ [\![\varphi \vee \psi]\!]^{\mathcal{T}}_\rho &= [\![\varphi]\!]^{\mathcal{T}}_\rho \cup [\![\psi]\!]^{\mathcal{T}}_\rho \\ [\![\varphi \wedge \psi]\!]^{\mathcal{T}}_\rho &= [\![\varphi]\!]^{\mathcal{T}}_\rho \cap [\![\psi]\!]^{\mathcal{T}}_\rho \\ [\![\langle a \rangle \varphi]\!]^{\mathcal{T}}_\rho &= \{s \in S \mid \exists t \in [\![\varphi]\!]^{\mathcal{T}}_\rho \text{ with } s \xrightarrow{a} t\} \\ [\![[a]\varphi]\!]^{\mathcal{T}}_\rho &= \{s \in S \mid \forall t \in S : \text{ if } s \xrightarrow{a} t \text{ then } t \in [\![\varphi]\!]^{\mathcal{T}}_\rho\} \\ [\![\mu X.\varphi]\!]^{\mathcal{T}}_\rho &= \bigcap \{T \subseteq S \mid [\![\varphi]\!]^{\mathcal{T}}_{\rho[X \mapsto T]} \subseteq T\} \\ [\![\nu X.\varphi]\!]^{\mathcal{T}}_\rho &= \bigcup \{T \subseteq S \mid T \subseteq [\![\varphi]\!]^{\mathcal{T}}_{\rho[X \mapsto T]}\} \end{aligned}$$

Two formulas φ and ψ are equivalent, written $\varphi \equiv \psi$, iff for all LTS $\mathcal{T}$ and all environments ρ we have $[\![\varphi]\!]^{\mathcal{T}}_\rho = [\![\psi]\!]^{\mathcal{T}}_\rho$. We may also write $\mathcal{T}, s \models_\rho \varphi$ instead of $s \in [\![\varphi]\!]^{\mathcal{T}}_\rho$.

2.2. **Syntactic Extensions.** Sometimes it is convenient to relax the restrictions on variable dependency. *Vectorial form* allows one to do so. Let $X_1, \ldots, X_m$ be variables and let $\psi_1, \ldots, \psi_m$ be formulas, possibly with free occurrences of the X_i. For both $\sigma \in \{\mu, \nu\}$ and any $1 \leq j \leq m$,

$$\Phi = \sigma X_j . \left\{ \begin{array}{ll} X_1. & \psi_1 \\ & \vdots \\ X_m. & \psi_m \end{array} \right\}$$

is a formula in *m-vectorial form*, and the X_i are considered bound in Φ. We say that a formula is in *vectorial form* if it is in m-vectorial form for some m. Hence, formulas in 1-vectorial form are ordinary $\mathcal{L}_\mu$-formulas as introduced above. The curly brackets are used to indicate that the m defining fixpoint equations are to be seen as a set; there is no implicit order among them with the exception of a

variable X_j marked as *entry variable*. By convention, the entry variable is the variable that occurs first if it is not explicitly given. We consider multiple instances of the same vectorial form, but with different entry variable, to be the essentially the same subformula in the sense that k occurrences of the same block of size k' with different entry variables contribute to the size of the formula with $k+k'$ instead of kk'.

The semantics of a vectorial formula is defined as follows: given a transition system $\mathcal{T} = (S, \to, \ell)$, and an environment ρ, a vector of fixpoint formulas

$$\Psi = \left\{ \begin{array}{ll} X_1. & \psi_1 \\ & \vdots \\ X_m. & \psi_m \end{array} \right\}$$

defines a monotone operator $S^m \to S^m$ via $(S_i)_{i \leq m} \mapsto [\![\varphi_i]\!]^{\mathcal{T}}_{\rho[X_1 \mapsto S_1, \ldots X_m \mapsto S_m]}$. Let $T = (T_1, \ldots, T_m) \subseteq S^m$ be the least fixpoint of this operator, and let X_k be the entry variable. Then

$$[\![\mu X_k.\Psi]\!]^{\mathcal{T}}_{\rho} = T_k$$

and accordingly for greatest fixpoints.

Example 2.1. Consider the following 3-vectorial formula where $\Box\psi$ is used to abbreviate $\bigwedge_{a \in \Sigma}[a]\psi$.

$$\mu X. \left\{ \begin{array}{ll} X. & \Box\mathtt{ff} \vee \langle a \rangle Y \vee Z \\ Y. & \langle b \rangle (Y \vee X) \\ Z. & \langle a \rangle X \vee \langle c \rangle Z \end{array} \right\}$$

It expresses "there is a maximal path labelled with a word from $(ab^+ + c^*a)^*$". It abbreviates the following formula in non-vectorial form.

$$\mu X.\Box\mathtt{ff} \vee \langle a \rangle (\mu Y.\langle b \rangle (Y \vee X)) \vee \mu Z.\langle a \rangle X \vee \langle c \rangle Z$$

Hierarchical equation systems, or HES, generalise $\mathcal{L}_\mu$-formulas by lifting the restriction of tree-like variable dependencies (see below). An equation has the form $Z = e_Z$, where $Z \in \mathcal{V}$ and e_Z can be generated from e in

$$e \ ::= \ q \mid \overline{q} \mid X \mid e \vee e \mid e \wedge e \mid \langle a \rangle e \mid [a]e$$

with $X \in \mathcal{V}$, $q \in \mathcal{P}$, and $a \in \Sigma$ as above. An HES is a finite set $\mathcal{S}$ of equations with disjoint left sides, an ordered partition $\{(S_1, \sigma_1), \ldots, (S_k, \sigma_k)\}$ of the equations with fixpoint qualifications and a designated entry variable from the first partition class. Here, $\sigma_i \in \{\mu, \nu\}$ as per usual. A partition class S_i is also called a block. The size of an HES $\mathcal{S}$ with left hand sides in Z is defined as $\Sigma_{z \in Z}|e_z|$. A variable is *free* in S_i if it is not the left hand side of an equation in some S_j for $j \geq i$. An HES is called closed if no variable is free in S_1. An equation system is called *boolean* if it is built without use of $\langle a \rangle$ and $[a]$ for any $a \in \Sigma$.

Following Neumann and Seidl [22], we define the semantics of HES on the power-set lattice of a given transition system $\mathcal{T} = (S, \to, \ell)$. For a system $\mathcal{S}$ with partition $\{(S_1, \sigma_1), \ldots, (S_k, \sigma_k)\}$, define Z_i as the set of left sides of equations in S_i. For two assignments ρ_1 and ρ_2 with disjoint domains and equal range, define $\rho_1 + \rho_2$ as the assignment that behaves like ρ_1 on the domain of ρ_1 and behaves like ρ_2 on the domain of ρ_2. The semantics of $\mathcal{S}$ is defined inductively over the blocks. Given an

assignment $\rho\colon \mathrm{free}(S_i) \to 2^S$, a block defines a function $f\colon (Z_i \to 2^S) \to (Z_i \to 2^S)$ via

$$(\tau, Z) \mapsto [\![e_z]\!]^{\mathcal{T}}_{\rho+\tau+[\![S_{i+1}]\!]^{\mathcal{T}}_{\rho+\tau}}$$

where $[\![S_{i+1}]\!]^{\mathcal{T}}_{\rho+\tau}$ is the empty assignment if $i = k$. Let $\hat{f}$ denote the least fixpoint of f with respect to τ if $\sigma_i = \mu$, respectively the greatest such fixpoint if $\sigma_i = \nu$. Define $[\![S_i]\!]^{\mathcal{T}}_{\rho}(Z)$ as

$$[\![S_i]\!]^{\mathcal{T}}_{\rho}(Z) = \begin{cases} \hat{f}(Z) & \text{if } Z \in Z_i \\ [\![S_{i+1}]\!]^{\mathcal{T}}_{\rho+\hat{f}}(Z) & \text{if } Z \in Z_j, j > i \end{cases}$$

Finally, given an assignment $\rho\colon \mathrm{free}(S_1) \to 2^S$, set $[\![\mathcal{S}]\!]^{\mathcal{T}} = [\![S_1]\!]^{\mathcal{T}}_{\rho}$ and for closed $\mathcal{S}$ write $\mathcal{T}, s \models \mathcal{S}$ if $s \in [\![\mathcal{S}]\!]^{\mathcal{T}}(Z)$, and the entry variable Z is clear from the context or has been explicitly designated. Two HES $\mathcal{S}$ and $\mathcal{S}'$ are equivalent if they agree on their free variables and their outermost variable and, for every $\mathcal{T} = (S, \to, \ell)$ and every $\rho\colon \mathrm{free}(\mathcal{S}) \to 2^S$, we have $[\![\mathcal{S}]\!]^{\mathcal{T}}_{\rho} = [\![\mathcal{S}']\!]^{\mathcal{T}}_{\rho}$.

It is not hard to see that we can always combine adjacent blocks with the same fixpoint qualification, hence we can consider the fixpoint qualifiers of the blocks to be strictly alternating.

Interestingly, HES and alternating parity tree automata in their symmetric form are different forms of notation for the same thing: states of the automaton correspond to the variables in the equation systen, the block partition corresponds to the priority function over the states of an automaton, and the equations make up the transition function.

2.3. **Equational Form, Variable Order, and Variable Dependencies.** $\mathcal{L}_\mu$-formulas and vectorial $\mathcal{L}_\mu$-formulas can be brought into an alternative syntax that resembles HES. Consider an $\mathcal{L}_\mu$-formula φ with fixpoint variables in $X_1, \ldots, X_n$. Without loss of generality, φ is of the form $\sigma X_1.\psi_1$.[1] Proceeding from an innermost fixpoint formula, convert a subformula of the form $\sigma_i X_i.\psi_i$ to a single-equation block $(\{X_i = e_{X_i}\}, \sigma_i)$ by a translation that abstracts away the difference between fixpoint quantification of the form $\sigma X.\psi$ and occurrences of fixpoint variables. This translation e is defined via

$$\begin{aligned} e(q) &= q & e(\langle a\rangle\psi) &= \langle a\rangle e(\psi) \\ e(\overline{q}) &= \overline{q} & e([a]\psi) &= [a]e(\psi) \\ e(\psi_1 \wedge \psi_2) &= e(\psi_1) \wedge e(\psi_2) & e(\sigma' X'.\psi) &= X' \\ e(\psi_1 \vee \psi_2) &= e(\psi_1) \vee e(\psi_2) & e(X) &= X. \end{aligned}$$

This process yields a set of equation blocks $\{(\{X_1 = e_{X_1}\}, \sigma_1), \ldots, (\{X_n = e_{X_n}\}, \sigma_n)\}$, partially ordered by the subformula relationship of the original formulas. We call the strict, transitive version of this order the *priority order*. The equational syntax generalizes accordingly to formulas in vectorial form: fixpoint variables in the same vector share a block as in the case of HES. As in the case of plain $\mathcal{L}_\mu$-formulas, the subformula relationship induces a strict, transitive partial order. In this case, two variables from the same block are incomparable. Any linearization of this partial order produces a hierarchical equation system.

[1] Otherwise, introduce a vacuous outermost fixpoint quantifier. This can be done without altering alternation depth.

Consider an HES with fixpoint variables $\mathcal{X} = X_n, \ldots, X_0$. Construct a *variable dependency graph* with node set $\mathcal{X}$ as follows: there is an edge from variable X_i to variable X_j if and only if X_j appears in e_{X_i}.

Note that the variable dependency graph for an HES resulting from an $\mathcal{L}_\mu$-formula is always a tree with back edges compatible with the priority order. This means that any edge in the dependency graph either goes from a node to an immediate successor in the priority order, or it is a loop, or it goes to a predecessor in the priority order. However, edges never go from a node to an indirect successor, e.g. to the son of a son. In this sense, priority order and variable dependency almost coincide and can be deduced from the subformula nesting.

The variable dependency graph of a vectorial formula or an HES is generally not a tree with back edges. However, formulas in vectorial form can be characterized in the following way: a formula is in vectorial form if and only if the set of equations can be partitioned into blocks and the blocks can be partially ordered respecting the priority order such that variable dependency edges only stay in a block, go from one block to an immediate successor block, or go to a—not necessarily immediate—predecessor block. In this sense, the block structure forms a tree with back edges. Moreover, all equations in a block are of the same type, i.e., qualified only by μ or only by ν.

2.4. **Guardedness and Weak Guardedness.** We extend the variable dependency graph by annotating edges with the information whether or not a modal operator has been passed between the two variables. More precisely, we call an edge from variable X to variable Y guarded if in the equation $X = e_X$ variable Y occurs under the scope of a modal operator. Note that there can be both a guarded and an unguarded edge from a variable to another. Call the resulting graph the *guardedness graph*.

An occurence of a variable in an equation is called *unguarded* if it is part of an unguarded cycle in the guardedness graph, and it is called *guarded* if it is not unguarded. An equation system is called *guarded* if there are no unguarded occurrences of variables in its guardedness graph, i.e., there are no unguarded cycles. An equation system is called *downwards guarded* if unguarded edges only occur strictly upwards in the partial order induced by variable priority. An equation system is called *ε-free* if there are no unguarded edges in its guardedness graph. Clearly, ε-freeness implies downwards guardedness, which in turn implies guardedness.

Example 2.2. The formula

$$\mu X.\Box\mathtt{ff} \vee \langle a\rangle(\mu Y.\langle b\rangle(Y \vee X)) \vee \mu Z.\langle a\rangle X \vee \langle c\rangle Z$$

from Example 2.1 expresses "there is a maximal path labeled with a word from $(ab^+ + c^*a)^*$" and is guarded. However, consider the following formula which expresses the slightly different property "there is a maximal path labeled with a word from $(ab^+ + c^*)^*$".

$$\mu X.\Box\mathtt{ff} \vee \langle a\rangle(\mu Y.\langle b\rangle(Y \vee X)) \vee \mu Z.X \vee \langle c\rangle Z$$

It is not guarded; in particular, there is an occurrence of X—the latter one—which is not guarded in its defining fixpoint formula, which happens to be the entire formula in this case.

We say that an occurrence of a variable X is *weakly guarded* if it is guarded or if all unguarded cycles for this occurrence of this variable have length at least 2, i.e.

X does not appear unguarded in e_X. For plain $\mathcal{L}_\mu$-formulas, this means that X is either guarded or it occurs under the scope of another fixpoint quantifier in its defining fixpoint subformula $\sigma X.\psi_X$. Note that weak guardedness is indeed weaker than guardedness, hence, not being weakly guarded entails not being guarded.

Consider, for instance, the formula $\mu X.q \vee (\mu Y.(q \wedge X) \vee (\overline{q} \wedge Y) \vee \langle a \rangle Y)$. Then Y has both a guarded and an unguarded occurrence, whereas the only occurrence of X is not guarded but it is weakly guarded.

A *guarded transformation* for $\mathcal{L}_\mu$ or its syntactic extensions is a function such that $\tau(\varphi)$ is guarded and $\tau(\varphi) \equiv \varphi$ for every $\varphi \in \mathcal{L}_\mu$.

3. Upper Bounds and Failure Results

3.1. **Guarded Transformation Without Vectorial Form.** The guarded transformation procedures for non-vectorial formulas by Kupferman et al. and Mateescu rely on two principles. The first principle is the well-known fixpoint unfolding.

Proposition 3.1. *For every $\sigma X.\varphi \in \mathcal{L}_\mu$ we have $\sigma X.\varphi \equiv \varphi[\sigma X.\varphi/X]$.*

The second principle states how occurrences that are not weakly guarded can be eliminated. Remember that $\hat{\sigma}$ is either $\mathtt{tt}$ or $\mathtt{ff}$ depending on σ being ν or μ.

Proposition 3.2 ([20, 21])**.** *Let $\sigma X.\varphi \in \mathcal{L}_\mu$ and let $\sigma X.\varphi'$ result from $\sigma X.\varphi$ by replacing with $\hat{\sigma}$ every occurrence of X that is not weakly guarded. Then $\sigma X.\varphi \equiv \sigma X.\varphi'$.*

These two principles can be combined to a simple guarded transformation procedure. Starting with the innermost fixpoint bindings, one replaces all occurrences of the corresponding variables that are not weakly guarded by $\mathtt{tt}$ or $\mathtt{ff}$ using Proposition 3.2. Note that for the innermost fixpoint subformulas, the concepts of being weakly guarded and being guarded coincide. Thus, the innermost fixpoint subformulas are guarded after this step.

For outer fixpoint formulas this only ensures that all remaining occurrences are weakly guarded. However, by the induction hypothesis, all inner ones are already guarded, and unfolding them using Prop. 3.1 puts all weakly guarded but unguarded occurrences of the outer variable under a $\langle a \rangle$- or $[a]$-modality. Hence, only occurrences that are either guarded or not weakly guarded survive, and the latter can be eliminated using Proposition 3.2 again.

Let τ_0 denote the guarded transformation which works as described above. Kupferman et al. claim that the worst-case blowup in formula size produced by τ_0 is linear, Mateescu claims that it is quadratic. We will show that it is indeed exponential. Consider the family of formulas

$$\Phi_n \;:=\; \mu X_1 \ldots \mu X_n.(X_1 \vee \cdots \vee X_n) \vee \langle a \rangle (X_1 \vee \cdots \vee X_n).$$

Theorem 3.3. *We have $|\Phi_n| = 3n + 1$ and $|\tau_0(\Phi_n)| = \Omega(2^n)$.*

Proof. The first claim about the linear growth of Φ_n is easily verified. We prove that $\tau_0(\Phi_n)$ contains a subformula of modal depth at least 2^{n-1}, which entails exponential size of $\tau_0(\Phi_n)$.

Let $\varphi = (X_1 \vee \cdots \vee X_n)$. Mateescu's guarded transformation transforms a (strict) subformula of the form $\sigma X.\psi$, into $f_X(t'(\psi))[\sigma X.f_X(t'(\psi))/X]$, where t' is the guarded transformation for subformulas and f_X replaces unguarded occurrences of X by $\hat{\sigma}$. Moreover, $t'(\varphi \vee \langle a \rangle \varphi) = \varphi \vee \langle a \rangle \varphi$. For $2 \leq i \leq n$, define $\varphi_i =$

$t'(\mu X_i \cdots X_n.(\varphi \vee \langle a\rangle\varphi)$. Then $\varphi_n = f_{X_n}(\varphi \vee \langle a\rangle\varphi)[\mu X_n.f_{X_n}(\varphi \vee \langle a\rangle\varphi)/X_n]$, and generally, $\varphi_i = f_{X_i}(\varphi_{i+1})[\mu X_i.f_{X_i}(\varphi_{i+1})/X_i]$. We show that φ_i contains φ at modal depth $2^{(n+1-i)}$. Clearly φ_n contains φ at modal depth $2 = 2^1$. Since $\varphi_i = f_{X_i}(\varphi_{i+1})[\mu X_i.f_{X_i}(\varphi_{i+1})/X_i]$, we have that, if φ_{i+1} contains φ at modal depth 2^{n-i}, then φ_i contains φ at double the modal depth, or $2 \cdot 2^{n-i} = 2^{n+1-i}$. Hence, $t'(\mu X_2 \cdots \mu X_n.(\varphi \vee \langle a\rangle\varphi)) = \varphi_2$ contains a formula of modal depth 2^{n-1}. Finally, Mateescu defines $\tau_0(\sigma X.\psi) = \sigma X.f_X(t'(\psi))$, whence $\tau_0(\Phi_n) = \mu X_1.f_X(\varphi_2)$, and the proof is finished. □

However, the blowup for this guarded transformation procedure is never worse than exponential.

Theorem 3.4. *For all every $\mathcal{L}_\mu$-formula φ, the size of $\tau_0(\varphi)$ is in $\mathcal{O}(2^{|\varphi|})$.*

Proof. Let $\varphi \in \mathcal{L}_\mu$. We analyze the size of $\tau_0(\varphi)$ from the inside out. Replacing unguarded occurrences of the innermost fixpoint quantifier with the default values will at most double the size of this formula. Unfolding it will double the size again. For a non-innermost, non-outermost quantifier, replacement of non-weakly guarded occurrences of variables will only affect subformulas created by unfolding. Since these are already accounted for in the blowup, changing some occurrences of variables to default values will not contribute to the blowup. Unfolding a non-innermost, non-outermost subformula will double its size at worst. The outermost formula is not unfolded. This makes at most one doubling of size per fixpoint quantifier, hence the resulting size is bounded from above by $2^{|\varphi|}$. □

Note that this guarded transformation procedure always produces downwards guarded formulas: occurrences of variables are always guarded, and fixpoint quantifiers are also always under the scope of a modal operator.

3.2. **Guarded Transformation With Vectorial Form and for HES.** We do not study guarded transformation for vectorial formulas in particular, because by Lemma 3.5, HES can be converted into vectorial formulas with only polynomial blowup. This transformation keeps guardedness, but the resulting vectorial formula will not be downwards guarded and, hence, not be ε-free.

Neumann and Seidl present guarded transformation in the context of HES over distributive lattices with monotone operators [22]. We stay with to the stipulations from Section 2 and only consider the powerset lattice with operators $\langle a\rangle$ and $[a]$.

Neumann and Seidl give a guarded transformation procedure for a class of equation systems that contains the class of HES obtained from $\mathcal{L}_\mu$-formulas in plain form. This transformation procedure runs in polynomial time and only produces a polynomial blowup. However, the resulting equation system does not correspond to a flat $\mathcal{L}_\mu$-formula. In Thorem 3.6, we see that turning an HES into an $\mathcal{L}_\mu$-formula is likely to incur a blowup. Hence, the guarded transformation by Neumann and Seidl does not constitute a polynomial guarded transformation for $\mathcal{L}_\mu$-formulas.

The following example illustrates the loss in structure of the equation system representing an $\mathcal{L}_\mu$-formula. Consider the following family of formulas:

$$\mu X_1 \ldots \mu X_n.X_1 \vee \cdots \vee X_n \vee \langle a\rangle\Big(\bigvee_{j=1}^{m} \mu Y_j.\langle a\rangle\big(Y_j \vee \bigvee_{i=1}^{n} X_i\big)\Big)$$

The associated equation system has a single block qualified with μ and looks like this:

$$\begin{aligned} X_1 &= X_2 & Y_1 &= \langle a\rangle(Y_1 \vee X_1 \vee \cdots \vee X_n) \\ &\vdots & &\vdots \\ X_{n-1} &= X_n & Y_m &= \langle a\rangle(Y_m \vee X_1 \vee \cdots \vee X_n) \\ X_n &= X_1 \vee \cdots \vee X_n \vee \langle a\rangle(Y_1 \vee \cdots \vee Y_m) \end{aligned}$$

After the guarded transformation, the HES looks like this:

$$\begin{aligned} X_1 &= \langle a\rangle(Y_1 \vee \cdots \vee Y_m) & Y_1 &= \langle a\rangle(Y_1 \vee X_1 \vee \cdots \vee X_n) \\ &\vdots & &\vdots \\ X_n &= \langle a\rangle(Y_1 \vee \cdots \vee Y_m) & Y_m &= \langle a\rangle(Y_m \vee X_1 \vee \cdots \vee X_n) \end{aligned}$$

This HES has a variable dependency graph that is not a tree with back edges.

Note that the transformation of Neumann and Seidl produces downwards guarded formulas.

3.3. **Unraveling of Vectorial Formulas and HES.** We investigate how the different syntactic variants of $\mathcal{L}_\mu$ can be converted into each other.

Lemma 3.5. *Any HES can be converted into an equivalent vectorial formula with only polynomial blowup. This translations keeps guardedness, but the resulting vectorial formula will not be downwards guarded nor ε-free.*

Proof. The desired partition for the equations is already present from the block structure of the HES. It remains to modify the equations such that no variable dependency edges go from a block into a block that is a successor block, but not a direct successor. So assume there is an equation $X = e_X$ in block S_i that mentions a variable Y in a block S_j such that $j \geq i + 2$. Introduce new variables $H_{i+1}, \ldots, H_{j-1}$ with associated equations $H_k = H_{k+1}$ for all $k < j - 1$ and $H_{j-1} = Y$. Moreover, replace all occurences of Y in S_i by H_{i+1}. It is not hard to see that the resulting equation system is equivalent. By repeating this procedure for all offending variables, the equation system can be made a vectorial formula.

Since each variable in a block that is not the first induces at most k intermediate variables, where $k + 2$ is the number of blocks, the blowup is polynomial, namely at most quadratic in the number of variables. Moreover, no new unguarded cycles are introduced, but downwards guardedness is obviously lost. □

Theorem 3.6. *Any HES with n variables can be transformed into an equivalent flat $\mathcal{L}_\mu$ formula with blowup factor 2^{n-1}.*

Proof. Let $\mathcal{S} = \{(S_1, \sigma_1), \ldots, (S_k, \sigma_k)\}$ be an HES. Let $\mathcal{X}$ denote the set of variables in $\mathcal{S}$. Let $<'_p$ denote a topological sorting of the priority order such that the entry variable is the maximal element, and let $>'_p$ denote its converse. For all $X \in \mathcal{X}$, let $\sigma_X = \sigma_i$ if $X = e_X \in S_i$.

Clearly, an equation of the form $X = e_X$ from a σ-block can be converted into an $\mathcal{L}_\mu$-formula of the form $\sigma X.\psi_X$, with variables Y occurring in e_X being either free or another formula ψ_Y being plugged in there. More precisely, for every σ-variable

[2] An earlier version [5] of this article contained an incorrect version of the transformation in Theorem 3.6.

$X \in \mathcal{X}$ and every subset $\mathcal{Y} \subseteq \mathcal{X}$ with $X \in \mathcal{Y}$, we define a formula $\sigma X.\psi_X^{\mathcal{Y}}$ with $\psi_X^{\mathcal{Y}} = t(e_X)$ according to

$$\begin{aligned}
t(q) &= q \\
t(\overline{q}) &= \overline{q} \\
t(\psi_1 \wedge \psi_2) &= t(\psi_1) \wedge t(\psi_2) \\
t(\psi_1 \vee \psi_2) &= t(\psi_1) \vee t(\psi_2) \\
t(\langle a \rangle \psi) &= \langle a \rangle t(\psi) \\
t([a]\psi) &= [a] t(\psi) \\
t(X) &= X \\
t(X') &= X' \text{ if } X' \in \mathcal{Y} \text{ and } X' >'_p X \\
t(X') &= \sigma_{X'} X' \psi_{X'}^{\mathcal{Y} \cup \{X'\}} \text{ if } X' <'_p X \\
t(X') &= \sigma_{X'} X' \psi_{X'}^{\mathcal{Y} \cup \{X'\} \setminus \{Z \colon Z <'_p X'\}} \text{ if } X' \notin \mathcal{Y} \text{ and } X' >'_p X.
\end{aligned}$$

If Z is the entry variable of $\mathcal{S}$, the formula $\varphi_{\mathcal{S}} = \sigma_Z Z.\psi_Z^{\{Z\}}$ is equivalent to $\mathcal{S}$. Since the translation for the non-fixpoint operators is obviously correct, it is enough to show that modal operators are properly nested in order to show this. We observe that no formula of the form $\sigma_Z Z.\psi_Z^{\mathcal{Z}}$ has free variables Z' such that $Z <'_p Z$. Moreover, the nesting of the formulas is finite. Since for each new fixpoint nesting, either a variable is added or a variable is added and all variables below it are cleared from the set $\mathcal{Z}$, the nesting process is finite: in order to remove a variable, a variable that is higher in the priority order has to be added. This can only happen a finite number of times; if Z has k variables above it, then Z is removed no more than $\lfloor 2^{k-2} \rfloor$ times. This leaves the maximal nesting depth at 2^{n-1}, where n is the number of equations. Moreover, since no formula of the form $\sigma_Z Z.\psi_Z^{\mathcal{Z}}$ has free variables Z' such that $Z <'_p Z$, we can reuse formulas such that the formula DAG has exactly 2^{n-1} nodes when restricted to fixpoint formulas. Hence, the total size of the formula can be bounded from above by $2^{n-1} \cdot |e|$, where e is the equation of maximal size. □

Eliminating ε-transitions from an alternating parity tree automaton is treated in the literature in several places. Since an alternating parity tree automaton is just another way of presenting HES, we briefly consider the problem, too. Wilke [27] gives an argument that elimination of ε-procedures can be done with exponential blowup, but keeping a linear number of states. The latter actually has to be replaced with a quadratic blowup in the number of states.

Vardi [24] considers the problem in the more general framework of two-way automata. Finally, the guarded transformation procedure of Neumann and Seidl [22] for general HES can be modified to yield a procedure that eliminates ε-transitions. Since Wilke's proof does not directly present an algorithm and Vardi's proof caters to a much more general framework, we give a variant of Neumann and Seidl's procedure.

Lemma 3.7. *Let $\mathcal{S} = \{(S_1, \sigma_1), \ldots, (S_k, \sigma_k)\}$ be an HES or an alternating parity tree automaton with n equations of total size $\Sigma_{i \leq n} e_i = m$. Then there is an ε-free HES $\mathcal{S}'$ with k blocks, nk equations and of exponential size.*

Proof. For each variable X in block S_i introduce variables X^j in block S_j for all $j > i$. The new variables inherit the old transitions, i.e $e_{X^j} = e_X$.

Note that the right-hand sides of the equations can be seen as elements of the free distributive lattice over the set of atoms $\mathcal{P} \cup \{\overline{p} \colon p \in \mathcal{P}\} \cup \mathcal{X} \cup \big(\bigcup_{a \in \Sigma} \{\langle a \rangle, [a]\} \times \mathcal{X}\big)$. The height of this lattice, i.e. the maximal length of a strictly ascending or descending chain, can be bounded from above by $H = 2|\mathcal{P}| + 3|\mathcal{X}|$.

We can eliminate ε-cycles from an equation $Z = e_Z$ from a block qualified with σ in the following way: let $e_Z^0 = e_Z[\hat{\sigma}/Z]$, and let $e_Z^{i+1} = e_Z[e_Z^i/Z]$. The Knaster-Tarski Theorem yields the existence of a $j \leq H$ such that $e_Z \equiv e_Z^j$. Moreover, e_Z^j does not contain any ε-transitions towards Z. In order to avoid doubly-exponential blowup, it is convenient to convert all intermediate equations into disjunctive normal form. This increases the size of an equation exponentially and takes exponential time. However, since the height of the lattice in question is H, no expression exceeds size 2^H. Moreover, repeating the conversion to normal form H times still takes only time in $\mathcal{O}(H * 2^H)$.

Let $X_1, \ldots, X_{m'}$ be an enumeration of the variables compatible with the priority order, and assume that ε-loops have been removed as per above. Eliminate ε-transitions towards variables X_i the following way, starting with the lowest variable $X_{m'}$ and ending with X_1: in an equation $X_j = e_{X_j}$, replace all unguarded ε-transitions towards X_i with e'_{X_i}, where e'_{X_i} is obtained from e_{X_i} by replacing all Y from blocks with lower priority than X_i with Y^i. After we have done this for X_i, the system does not contain any ε-transitions towards X_i, so after the procedure finishes, the system is ε-free. By a normal form argument as above, the overall blowup in the system does not exceed one exponential.

In order to argue why such a replacement preserves the semantics, consider the framework of a parity automaton. Instead of doing an ε-transition towards a state and then doing more transitions towards another state, we do the transitions to the third state right away. This is correct as long as we record the parity of the state we skipped[3]. The additional copies of the states with low priority server this purpose. Obviously, we can skip a priority if we transition towards a higher priority later. Correctness of the process follows by induction. □

4. Lower Bounds

In this section we show that guarded transformation for vectorial formulas and HES is at least as hard—modulo polynomials—as parity game solving. The problem of whether or not the latter is possible in polynomial time has been open for a long while. We also show that unfolding a formula in vectorial form or an HES into an equivalent non-vectorial formula is at least as hard as parity game solving. The core of the proofs is a product construction similar to that in Kupferman et al. [20].

Theorem 4.1 (Product Construction)**.** *For every LTS $\mathcal{T}$ and every closed $\varphi \in \mathcal{L}_\mu$ there is an LTS $\mathcal{T}'$ with a single state v_0 and such that for every state s_0 in $\mathcal{T}$, there is a vectorial $\varphi'_{s_0} \in \mathcal{B}_\mu$ such that*

(1) *$\mathcal{T}, s_0 \models \varphi$ iff $\mathcal{T}', v_0 \models \varphi'_{s_0}$,*
(2) *$|\mathcal{T}'| = \mathcal{O}(|\varphi| \cdot |\mathcal{T}|)$, and*
(3) *$|\varphi'_{s_0}| = \mathcal{O}(|\varphi| \cdot |\mathcal{T}|)^2$.*

[3]This is the problem with Wilke's construction.

Proof. Let $\mathcal{T} = (S, \to, \ell)$ and let φ be defined over propositions $\mathcal{P}$ and variables $\mathcal{V}$. W.l.o.g. we assume that $S = \{1, \ldots, m\}$ for some $m \in \mathbb{N}$. Define new sets of propositions $\mathcal{P}' := \mathcal{P} \times S$ and variables $\mathcal{V}' := \mathcal{V} \times S$. We write X_s and q_s instead of (X, s) and (q, s).

Let $\mathcal{T}' = (\{v_0\}, \emptyset, \ell')$ consist of a single state with the following labeling: $q_s \in \ell'(v_0)$ iff $q \in \ell(s)$.

Next we give an inductively defined transformation $\mathrm{tr}\colon S \times \mathcal{L}_\mu \to \mathcal{B}_\mu$ which turns an $\mathcal{L}_\mu$ formula over $\mathcal{V}$ and $\mathcal{P}$ into a vectorial $\mathcal{B}_\mu$ formula over $\mathcal{V}'$ and $\mathcal{P}'$.

$$
\begin{aligned}
\mathrm{tr}_s(q) &= q_s \\
\mathrm{tr}_s(\overline{q}) &= \overline{q_s} \\
\mathrm{tr}_s(X) &= X_s \\
\mathrm{tr}_s(\psi_1 \vee \psi_2) &= \mathrm{tr}_s(\psi_1) \vee \mathrm{tr}_s(\psi_2) \\
\mathrm{tr}_s(\psi_1 \wedge \psi_2) &= \mathrm{tr}_s(\psi_1) \wedge \mathrm{tr}_s(\psi_2) \\
\mathrm{tr}_s(\langle a \rangle \psi) &= \bigvee \{ \mathrm{tr}_t(\psi) \mid t \in S \text{ with } s \xrightarrow{a} t \} \\
\mathrm{tr}_s([a]\psi) &= \bigwedge \{ \mathrm{tr}_t(\psi) \mid t \in S \text{ with } s \xrightarrow{a} t \} \\
\mathrm{tr}_s(\sigma X.\psi) &= \sigma X_s. \left\{ \begin{array}{lcl} X_1 & . & \mathrm{tr}_1(\psi) \\ & \vdots & \\ X_m & . & \mathrm{tr}_m(\psi) \end{array} \right\}
\end{aligned}
$$

Set φ'_{s_0} to be $\mathrm{tr}_{s_0}(\varphi)$. It should be clear that φ'_s is indeed a formula of $\mathcal{B}_\mu$. For item 1 of the theorem, consider Stirling's local model-checking game for $\mathcal{L}_\mu$ [23]. It is not hard to see that $(s, \psi) \mapsto (v_0, \psi_s)$ maps positions in the model-checking game for $\mathcal{T}, s_0$ and φ isomorphically to positions in the game for $\mathcal{T}', v_0$ and φ'_{s_0}. Moreover, strategy decisions map in the same manner, hence Verifier has a winning strategy in one game if and only she has one in the other game, and the claim in item 1 follows from that.

The size argument in item 2 follows because $\mathcal{T}'$ has only one state, no transitions and $|\mathcal{P}| \times |S|$ many propositions. It remains to argue for item 3, regarding the size of φ'. Clearly, all steps except the modal operators and the fixpoints do not produce blowup beyond $(|\varphi| \cdot |\mathcal{T}|)$ many subformulas. In the case of the modal operators, each instance of a box or a diamond produces exactly one subformula for each edge in $\mathcal{T}$, and edges with the same target produce the same subformula. A subformula of the form $\sigma X.\psi$ will produce a vectorial fixpoint expression for each state in $\mathcal{T}$, each system of size $|S|$. However, all these systems are isomorphic except in their entry variable. By our stipulation from Section 2, these are considered the same subformula. Moreover, since each of the X_s is available in every subformula of the system, further nested systems are not nested with exponential blowup, but in a linear fashion. The claim for the formula size follows. □

Theorem 4.2. *Guarded transformation for vectorial $\mathcal{L}_\mu$-formulas or vectorial $\mathcal{B}_\mu$-formulas is at least as difficult as solving parity games.*

Proof. We show that any polynomial guarded transformation for $\mathcal{B}_\mu$-formulas in vectorial form yields a polynomial solution algorithm for parity games. The statement for $\mathcal{L}_\mu$-formulas follows from that because every guarded transformation for $\mathcal{L}_\mu$-formulas is also one for $\mathcal{B}_\mu$-formulas.

Assume that there is a polynomial guarded transformation procedure τ for vectorial $\mathcal{B}_\mu$-formulas. Given a parity game, we can treat it as an LTS with one accessibility relation and labellings for ownership and priority of states. Solving the parity game means deciding whether the first player wins from the initial vertex, and this is equivalent to model-checking Walukiewicz' formula [26] for the corresponding priority. This formula is of size linear in the number of priorities, hence it is polynomial in the size of the parity game. Via the product construction from Theorem 4.1, we obtain a vectorial $\mathcal{B}_\mu$-formula φ that is also of size polynomial in the size of the parity game, and a one-state transition system $\mathcal{T}$ such that $\mathcal{T} \models \varphi$ if and only if the first player wins the parity game. Consider $\tau(\varphi)$. Because τ runs in polynomial time, the size of $\tau(\varphi)$ is still polynomial in the size of the parity game. Moreover, since the truth value of the $\mathcal{B}_\mu$-formula φ only depends on the state v_0, this must be true for $\tau(\varphi)$ as well. In effect, all modal operators introduced by τ are vacuous and can be replaced by $\mathtt{tt}$ in case of boxes, and by $\mathtt{ff}$ in case of diamonds. The resulting vectorial formula is again strictly boolean, but also guarded. Hence, it cannot contain any occurrences of fixpoint variables at all, since any such occurrence would be unguarded. Therefore, all fixpoint quantifiers can be removed. The resulting formula is purely propositional and can be solved in polynomial time by a simple bottom-up algorithm. This yields the desired polynomial solution for the parity game. □

The $\mathcal{B}_\mu$ part of this theorem corresponds to Neumann and Seid's observation that "finding equivalent guarded systems in general cannot be easier than computing solutions of hierarchical systems of Boolean equations" [22].

Theorem 4.3. *Transforming a vectorial $\mathcal{L}_\mu$-formula or an HES to a non-vectorial $\mathcal{L}_\mu$-formula is at least as difficult as solving parity games. This holds even if the transformation only accepts ε-free formulas and equation systems and even if it is allowed to produce unguarded formulas.*

Proof. Assume that we have a polynomial transformation τ from ε-free equation systems to $\mathcal{L}_\mu$-formulas, and assume that we are given a parity game and a state in that parity game. We want to use the product construction from Theorem 4.1, i.e. we want to construct a vectorial formula φ' and a one-state transition system $\mathcal{T}', v_0$ such that $\mathcal{T}', v_0 \models \varphi'$ if and only if Verifier wins the parity game from the given state.

Unfortunately, the vectorial formula φ' from this theorem does not contain any modal operators, so it is far from being guarded, let alone ε-free. To remedy this problem, replace the structure $\mathcal{T}', v_0$ by a version $\mathcal{T}'', v_0$ that has a loop at the only vertex. Clearly, a formula or equation system without modal operators is true in the old structure if and only if it is true in the new structure. Moreover, for any $\mathcal{L}_\mu$-formula or equation ψ, we have $\mathcal{T}'', v_0 \models \psi$ if and only if $\mathcal{T}'', v_0 \models \Diamond\psi$. Hence, we can replace every occurrence of a fixpoint variable X or a fixpoint subformula $\sigma X.\psi_X$ in φ' by $\Diamond X$ and $\Diamond\sigma X.\psi_X$ without changing the truth value of the formula. The resulting formula φ'' is now ε-free, and we can apply τ. Since τ produces equivalent formulas, we have

$$\mathcal{T}'', v_0 \models \tau(\varphi'') \;\Leftrightarrow\; \mathcal{T}'', v_0 \models \varphi'' \;\Leftrightarrow\; \mathcal{T}', v_0 \models \varphi' \;.$$

But since $\mathcal{T}'', v_0 \models \psi$ if and only if $\mathcal{T}'', v_0 \models \Diamond\psi$ for all ψ, and similar for modal boxes, we can replace any occurrence of modal operators of the form $\Diamond\psi$ or $\Box\psi$ by ψ.

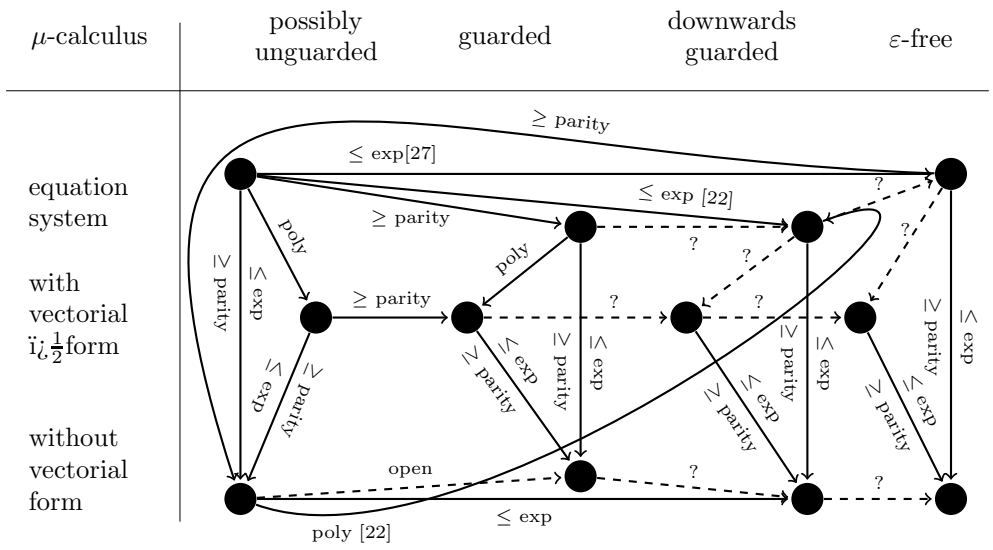

FIGURE 1. State of the art on guarded transformations for the modal μ-calculus.

The resulting formula φ''' does not contain any modal operators, and $\mathcal{T}'', v_0 \models \varphi'''$ if and only if the first player wins the parity game from the given state. Since $\mathcal{L}_\mu$-formulas without modal operators can be model-checked in polynomial time [22], and since all the transformations above are polynomial, the transformation τ gives rise to a polynomial procedure for solving parity games. □

5. Conclusion

In Section 3 we showed that the known guarded transformations produce an exponential blowup in the worst case. We also presented exponential algorithms for the elimination of ε-transitions in alternating parity tree automata and the translation from HES or vectorial formulas into flat $\mathcal{L}_\mu$-formulas.

In Section 4 we showed that a polynomial guarded transformation for vectorial formulas entails a polynomial solution algorithm for parity games, and the same is true for HES. We also proved that polynomial translation from vectorial formulas to non-vectorial formulas, or from HES to $\mathcal{L}_\mu$-formulas, yields the same. The existence of a polynomial guarded transformation for *non-vectorial* formulas is still open, and it is possible that such a transformation exists without yielding a polynomial solution algorithm for parity games. Figure 1 gives an overview over the current state of research.

Consequences and Corrections. Several constructions and procedures that deal with the modal μ-calculus directly or indirectly use guarded transformation. Often they use the (possibly) false claim that guarded transformation can be done at a linear or quadratic blow-up. We examine consequences of the fact that according to current knowledge, guarded transformation is exponential.

As a first step, it is interesting to check whether any of the results from Kupferman/Vardi/Wolper's seminal paper on automata for branching-time temporal logics [20] crucially rely on a polynomial guarded transformation. Fortunately, this is not the case. The product automaton construction [20, Prop. 3.2] also works if the input automaton is not ε-free, and the resulting product automaton has no ε-transitions. This allows the subsequent Thm. 3.1 to be applied, which needs ε-free automata. Hence, all results of [20] that rely on a polynomial guarded transformation remain true.

In [13], the reliance on a polynomial guarded transformation causes problems. It is not immediately obvious that the complexity results for satisfiability of existential and universal $\mathcal{L}_\mu$ and alternation-free $\mathcal{L}_\mu$ (claimed to be NP-complete) as well as derived results should hold for unguarded formulas. A preceding transformation into guarded form will exhaust the complexity limitations, so further investigation on this work is necessary.

Mateescu claims that model checking for $\mathcal{L}_\mu$ on acyclic structures can be done in polynomial time. He observes that, on acyclic structures, least and greatest fixpoints coincide. Thus, the alternation hierarchy collapses to its alternation-free fragment on such structures. It is known that this fragment can be model-checked in linear time [6]. However, least and greatest fixpoints only coincide for guarded formulas: clearly $\mu X.X$ and $\nu X.X$ are not equivalent, not even on acyclic structures. The collapse result is still true but with current technology at hand, we need to assume an exponential blow-up in formula size. Thus, model checking guarded formulas on acyclic structures can be done in polynomial time, arbitrary formulas still require exponential time.

Automata, ε-transitions and Vectorial Form. It is standard practice to construct alternating parity tree automata from guarded $\mathcal{L}_\mu$-formulas, or weak alternating tree automata from guarded alternation-free formulas [9, 28, 12]. The resulting automata are of size linear in the size of the formula. The situation is different for unguarded (alternation-free) $\mathcal{L}_\mu$-formulas because of the absence of a polynomial guarded transformation. Currently, we need to assume a blow-up that is exponential in the size of the formula when translating arbitrary, and therefore possibly unguarded, $\mathcal{L}_\mu$-formulas into alternating parity tree automata. If ε-transitions are allowed in alternating parity tree automata then it is possible to translate arbitrary $\mathcal{L}_\mu$-formulas into such automata at a linear blow-up only; the known constructions can be modified accordingly. Since, by Theorem 4.2, eliminating ε-transitions from alternating automata must be considered exponential, obtaining an ε-free alternating automaton from an unguarded $\mathcal{L}_\mu$-formula incurs an exponential blowup by current state of research.

The above means that the size of an alternating automaton cannot be measured in terms of number of states. Instead, such a notion of size has to include the size of the transition relation, which can easily be exponentially larger. An example of the confusion that the wrong measure can cause is Kupferman/Vardi's work on alternating automata. They show that nonemptiness of weak alternating automata can be checked in linear time when size is measured including the transition relation [20]. This result is then used in a context where the size is measured in the number of states [18]. We do believe that this claim, and subsequent results, are to be questioned.

Another problematic custom is that authors often use vectorial $\mathcal{L}_\mu$ and flat $\mathcal{L}_\mu$ interchangebly, or sometimes even alternating parity automata and $\mathcal{L}_\mu$. Since translating parity tree automata back into $\mathcal{L}_\mu$-formulas is exponential as well, the latter seems inappropriate. An example of the former is [19], where weak alternating automata are translated linearly into vectorial alternation free $\mathcal{L}_\mu$, referred to as $\mathcal{L}_\mu$ only. This seems confusing, since unraveling the vectorial formula obtained will incur exponential blowup. We think that vectorial form has its use, in particular because weak automata translate so easily into vectorial form, but the use of vectorial form should always be clearly labeled as such, in order to avoid hidden exponential gaps.

Further Research. The existence of a polynomial guarded transformation for flat $\mathcal{L}_\mu$ is still open, and it is also not known whether the existence of such a procedure would entail parity games being solvable in polynomial time. Interestingly enough, all known guarded transformation procedures produce downward guarded formulas, and we strongly suspect that any reasonable candidate for a polynomial guarded transformation will do so as well. This is because the notion of guardedness as such seems too weak: aiming for downward guardedness gives enough structure to the formula to establish reasonable induction invariants. On the other hand, showing that plain guarded transformation is subject to some lower bound suffers from the unstructured notion of guardedness. This is taken to an extreme in Berwanger's two-variable version [4] of the Walukiewicz formulas: his formulas are not guarded at all, but are *de facto* guarded. The means that, in a tableau, any two iterations of the same fixpoint variable will have a modal operator in between. Judging from the overall picture, we think that downward guardedness might be the right target to attack.

There are several arrows in Figure 1 where a transformation can be done with exponential blowup, but the lower bound is only that any polynomial procedure would also entail a polynomial solution algorithm for parity games. Closing theses gaps can be done in two ways: giving a polynomial solution algorithm for the problem—and establishing that parity games can be solved in polynomial time in the process—or establishing an exponential lower bound for the transformation.

Finally, the converse questions are also open: does a polynomial solution algorithm for parity games entail a polynomial guarded transformation procedure? Does it entail a polynomial procedure to unravel vectorial formulas or HES into flat $\mathcal{L}_\mu$-formulas? Of course, the ability to solve parity games in polynomial time allows polynomial model-checking for vectorial formulas, but the problems of guarded transformation, respectively of unraveling formulas, seem to be independent of that.

References

[1] A. Arnold and D. Niwiński. *Rudiments of μ-calculus*, volume 146 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 2001.

[2] B. Banieqbal and H. Barringer. Temporal logic with fixed points. In *Proc. Coll. on Temporal Logic in Specification*, volume 398 of *LNCS*, pages 62–73. Springer, 1989.

[3] H. Bekić. *Programming Languages and Their Definition, Selected Papers*, volume 177 of *LNCS*. Springer, 1984.

[4] D. Berwanger. Game logic is strong enough for parity games. *Studia Logica*, 75(2):205–219, 2003.

[5] F. Bruse, O. Friedmann, and M. Lange. Guarded transformation for the modal mu-calculus. *CoRR*, abs/1305.0648, 2013.

[6] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. *Formal Methods in System Design*, 2(2):121–147, 1993.
[7] M. Dam. CTL* and ECTL* as fragments of the modal μ-calculus. *TCS*, 126(1):77–96, 1994.
[8] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 16, pages 996–1072. Elsevier and MIT Press, New York, USA, 1990.
[9] E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd Symp. on Foundations of Computer Science*, pages 368–377, San Juan, Puerto Rico, 1991. IEEE.
[10] E. A. Emerson and C. L. Lei. Efficient model checking in fragments of the propositional μ–calculus. In *Symposion on Logic in Computer Science*, pages 267–278, Washington, D.C., USA, 1986. IEEE.
[11] O. Friedmann and M. Lange. The modal μ-calculus caught off guard. In *20th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX'11*, volume 6793 of *LNCS*, pages 149–163. Springer, 2011.
[12] J. Gutierrez, F. Klaedtke, and M. Lange. The μ-calculus alternation hierarchy collapses over structures with restricted connectivity. In *Proc. 3rd Int. Symp. on Games, Automata, Logics and Formal Verification, GandALF'12*, volume 96 of *Elect. Proc. in Theor. Comp. Sc.*, pages 113–126, 2012.
[13] T. A. Henzinger, O. Kupferman, and R. Majumdar. On the universal and existential fragments of the *mu*-calculus. *Theor. Comput. Sci.*, 354(2):173–186, 2006.
[14] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. In *Proc. 7th Conf. on Concurrency Theory, CONCUR'96*, volume 1119 of *LNCS*, pages 263–277. Springer, 1996.
[15] N. Jungteerapanich. A tableau system for the modal μ-calculus. In *Proc. 18th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX'09*, volume 5607 of *LNCS*, pages 220–234. Springer, 2009.
[16] R. Kaivola. Axiomatising linear time mu-calculus. In *Proc. 6th Int. Conf. on Concurrency Theory*, volume 962 of *LNCS*, pages 423–437. Springer, 1995.
[17] D. Kozen. Results on the propositional μ-calculus. *TCS*, 27:333–354, 1983.
[18] O. Kupferman and M. Y. Vardi. Weak alternating automata and tree automata emptiness. In *STOC*, pages 224–233, 1998.
[19] O. Kupferman and M. Y. Vardi. From linear time to branching time. *ACM Trans. Comput. Log.*, 6(2):273–294, 2005.
[20] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.
[21] R. Mateescu. Local model-checking of modal mu-calculus on acyclic labeled transition systems. In *Proc. 8th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'02*, volume 2280 of *LNCS*, pages 281–295. Springer, 2002.
[22] H. Seidl and A. Neumann. On guarding nested fixpoints. In *Proc. 8th Ann. Conf. on Computer Science Logic, CSL'99*, volume 1683 of *LNCS*, pages 484–498. Springer, 1999.
[23] C. Stirling. Local model checking games. In *Proc. 6th Conf. on Concurrency Theory, CONCUR'95*, volume 962 of *LNCS*, pages 1–11. Springer, 1995.
[24] M. Y. Vardi. Reasoning about the past with two-way automata. In *Proc. 25th Int. Coll. on Automata, Languages and Programming, ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641, 1998.
[25] I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. *Inf. and Comput.*, 157(1–2):142–182, 2000.
[26] I. Walukiewicz. Monadic second-order logic on tree-like structures. *Theoret. Comput. Sci.*, 275(1-2):311–346, 2002.
[27] T. Wilke. CTL$^+$ is exponentially more succinct than CTL. In *Proc. 19th Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'99*, volume 1738 of *LNCS*, pages 110–121. Springer, 1999.
[28] T. Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bull. Belgian Math. Soc.*, 8(2):359–391, 2001.

Florian Bruse, School of Elect. Eng. and Comp. Sc., University of Kassel, Germany
E-mail address: florian.bruse@uni-kassel.de

Oliver Friedmann, Dept. of Comp. Sc., University of Munich, Germany
E-mail address: oliver.friedmann@ifi.lmu.de

Martin Lange, School of Elect. Eng. and Comp. Sc., University of Kassel, Germany
E-mail address: martin.lange@uni-kassel.de