RAIRO-Inf. Theor. Appl. **38** (2004) 343-373
DOI: 10.1051/ita:2004017

COMPARING THE SUCCINCTNESS OF MONADIC QUERY LANGUAGES OVER FINITE TREES *

Martin Grohe[1] and Nicole Schweikardt[1]

Abstract. We study the *succinctness* of monadic second-order logic and a variety of monadic fixed point logics on trees. All these languages are known to have the same expressive power on trees, but some can express the same queries much more succinctly than others. For example, we show that, under some complexity theoretic assumption, monadic second-order logic is non-elementarily more succinct than monadic least fixed point logic, which in turn is non-elementarily more succinct than monadic datalog.

Succinctness of the languages is closely related to the combined and parameterised complexity of query evaluation for these languages.

Mathematics Subject Classification. 03B70, 68P15, 68Q45.

Introduction

A central topic in finite model theory has always been a comparison of the expressive power of different logics on finite relational structures. In particular, the expressive power of fragments of monadic second-order logic and various fixed-point logics has already been investigated in some of the earliest papers in finite model theory [5, 9]. One of the main motivations for such studies was an interest in the expressive power of query languages for relational databases.

In recent years, the focus in database theory has shifted from relational to semi-structured data and in particular data stored as XML-documents. A lot of current research in the database community is concerned with the design and implementation of XML query languages (see, for example, [10, 13, 18] or the monograph [1] for a general introduction into semi-structured data and XML). The languages studied in the present paper may be viewed as node-selecting query languages

* *This research was performed while the second author was supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD).*

[1] Institut für Informatik, Humboldt-Universität Berlin, Germany;
e-mail: grohe@informatik.hu-berlin.de; schweika@informatik.hu-berlin.de

for XML. They all contain the core of the language XPath, which is an important building block of several major XML-related technologies. Recently, monadic datalog has been proposed as a node-selecting query language with a nice balance between expressive power and very good algorithmic properties [13, 21].

XML-documents are best modelled by trees, or more precisely, finite labelled ordered unranked trees. It turns out that when studying node-selecting query languages for XML-documents, expressive power is not the central issue. Quite to the contrary: Neven and Schwentick [23] proposed to take the expressive power of monadic second-order logic (MSO) as a benchmark for node-selecting XML-query languages and, in some sense, suggested that such languages should at least have the expressive power of MSO. However, even languages with the same expressive power may have vastly different complexities. For example, monadic datalog and MSO have the same expressive power over trees [13]. However, monadic datalog queries can be evaluated in time linear both in the size of the datalog program and the size of the input tree [13], and thus the combined complexity of monadic datalog is in polynomial time, whereas the evaluation of MSO queries is PSPACE complete. The difference becomes even more obvious if we look at parameterised complexity: Unless PTIME $\neq$ NP, there is no algorithm evaluating a monadic second-order query in time $f(\text{size of query}) \cdot p(\text{size of tree})$ for any elementary function f and polynomial p [11]. Similar statements hold for the complexity of the satisfiability problem for monadic datalog and MSO over trees. The reason for this different behaviour is that even though the languages have the same expressive power on trees, in MSO we can express queries much more *succinctly*. Indeed, there is no elementary translation from a given MSO-formula into an equivalent monadic datalog program. We also say that MSO is *non-elementarily more succinct* than monadic datalog. Just to illustrate the connection between succinctness and complexity, let us point out that if there was an elementary translation from MSO to monadic datalog, then there would be an algorithm evaluating a monadic second-order query in time $f(\text{size of query}) \cdot p(\text{size of tree})$ for an elementary function f and a polynomial p.

In this paper, we study the succinctness of a variety of monadic logics on finite trees. We prove two types of results:

- *Upper bounds on succinctness*, which may also be called *translation* results: formulas φ_1 of a logic L_1 can be translated into equivalent formulas φ_2 of a logic L_2 such that the size $||\varphi_2||$ of φ_2 is bounded by $f(||\varphi_1||)$ for some function f from some class $\mathcal{F}$ of functions.
  In our terminology, such a result will be phrased as: L_1 is $\mathcal{F}$-succinct in L_2.
  For example, we prove that the 2-variable fragment MLFP^2 of monadic least fixed-point logic is $2^{\text{poly}(m)}$-succinct in the full modal μ-calculus FL_μ. That is, every formula φ_1 of MLFP^2 can be translated into an equivalent FL_μ-formula φ_2 such that $||\varphi_2|| \leq 2^{\text{poly}(||\varphi_1||)}$.
  (Definitions of the logics will of course be given later.)

- *Lower bounds on succinctness*, which may also be called *separation* results: formulas φ_1 of a logic L_1 cannot be translated into equivalent formulas φ_2 of a logic L_2 such that $||\varphi_2||$ is bounded by $f(||\varphi_1||)$ for some function f from some class $\mathcal{F}$ of functions.
  In our terminology, such a result will be phrased as: L_1 is not $\mathcal{F}$-succinct in L_2.
  For example, we prove that MLFP^2 is not $2^{o(m)}$-succinct in FL$_\mu$.

Our results are displayed in Figure 1. The left side (upward arrows) is concerned with translations from the lower to the upper logics, the right side (downward arrows) with translations from the upper to the lower logics. Most arrows are labelled with two rows: the first is our best upper bound, the second our best lower bounds. Trivial $o(m)$ lower bounds are not displayed (therefore the $\mathcal{O}(m)$ upper bounds have no counterparts). Question marks indicate the absence of non-trivial lower bounds.

Of course we are a not the first to study the succinctness of logics with the same expressive power. Most known results are about modal and temporal logics. The motivation for these results has not come from database theory, but from automated verification and model-checking. The setting, however, is very similar. For example, Kamp's well-know theorem states that first-order logic and linear time temporal logic have the same expressive power on strings [20], but there is no elementary translation from first-order logic to linear time temporal logic on strings. Even closer to our results, monadic second-order logic and the modal μ-calculus have the same expressive power on (ordered) trees, but again it is well-known that there is no elementary translation from the former to the latter. Both of these results can be proved by simple automata theoretic arguments. More refined results are known for various temporal logics [3, 4, 8, 26]. Adler and Immerman [3] proposed a nice game theoretical approach for proving lower bounds. Building on some of their ideas, we recently established succinctness results on finite variable fragments of first-order logic [17].

In the present paper, however, we mostly rely on automata theoretic arguments. An exception is the, complexity theoretically conditioned, result that MSO is non-elementarily more succinct than MLFP. To prove this result, we are building on a technique introduced in [11].

The paper is organised as follows: in Section 1 we fix the basic notations used throughout the paper. Section 2 concentrates on the translation from MSO to MLFP. In Section 3 we present our results concerning the two-variable fragment of MLFP and the full modal μ-calculus. Section 4 compares MLFP with its extension by simultaneous least fixed point operators. In Section 5 we concentrate on monadic datalog, stratified monadic datalog, and their relations to finite automata and to MLFP. Finally, Section 6 concludes the paper by pointing out several open questions.

The present paper is the full version of the conference contribution [15].

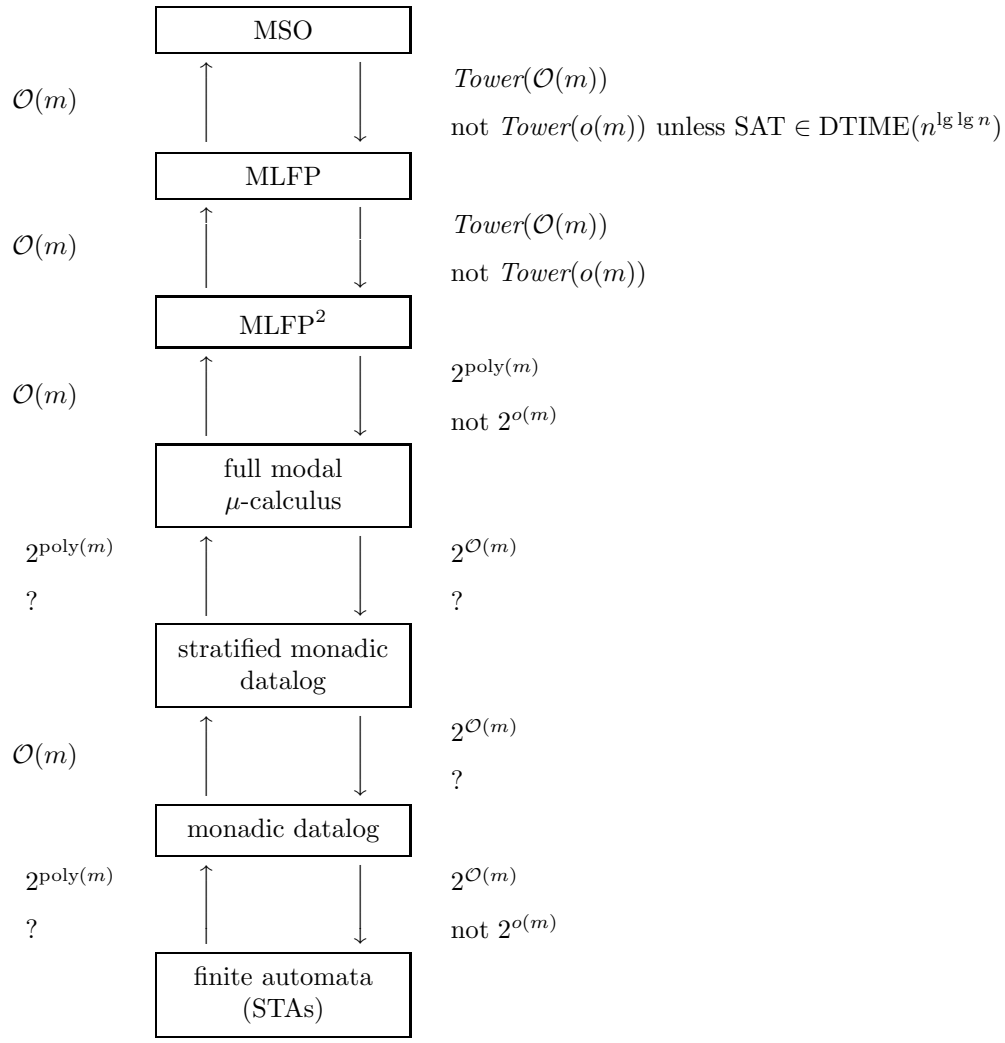

FIGURE 1. Schematic overview over our results.

1. PRELIMINARIES

1.1. BASIC NOTATIONS

Given a set Σ we write Σ^* to denote the set of all finite strings over Σ, and we use ε to denote the empty string. We use $\mathbb{N}$ to denote the set $\{0, 1, 2, ...\}$ of natural numbers. We use lg to denote the logarithm with respect to base 2. With a function f that maps natural numbers to real numbers we associate the

corresponding function from $\mathbb{N}$ to $\mathbb{N}$ defined by $n \mapsto \lceil f(n) \rceil$. For simplicity we often simply write $f(n)$ instead of $\lceil f(n) \rceil$.

The function $\mathit{Tower} : \mathbb{N} \to \mathbb{N}$ is inductively defined *via* $\mathit{Tower}(0) := 1$ and $\mathit{Tower}(h{+}1) = 2^{\mathit{Tower}(h)}$, for all $h \in \mathbb{N}$. *I.e.*, $\mathit{Tower}(h)$ is a tower of 2s of height h.

We say that a function $f : \mathbb{N} \to \mathbb{N}$ has bound $f(m) \leqslant \mathit{Tower}\big(o(h(m))\big)$, for some function $h : \mathbb{N} \to \mathbb{N}$, if there is a function $g \in o(h)$ and a $m_0 \in \mathbb{N}$ such that for all $m \geqslant m_0$ we have $f(m) \leqslant \mathit{Tower}\big(g(m)\big)$.

Note that, in particular, every *elementary* function f has bound $f(m) \leqslant \mathit{Tower}\big(o(m)\big)$. Indeed, for every elementary function f there is a $h \in \mathbb{N}$ such that, for all $n \in \mathbb{N}$, $f(n)$ is less than or equal to the tower of 2s of height h with an n on top.

1.2. Structures

A *signature* τ is a finite set of relation symbols and constant symbols. Each relation symbol $R \in \tau$ has a fixed arity $\mathit{ar}(R)$. A τ-structure $\mathcal{A}$ consists of a set $\mathcal{U}^{\mathcal{A}}$ called the *universe* of $\mathcal{A}$, an interpretation $c^{\mathcal{A}} \in \mathcal{U}^{\mathcal{A}}$ of each constant symbol $c \in \tau$, and an interpretation $R^{\mathcal{A}} \subseteq (\mathcal{U}^{\mathcal{A}})^{\mathit{ar}(R)}$ of each relation symbol $R \in \tau$. *All structures considered in this paper are assumed to have a finite universe.*

The main focus of this paper lies on the class *Trees* of finite binary trees. Precisely, finite binary trees are particular structures over the signature

$$\tau_{\mathit{Trees}} := \{\, \mathit{Root},\ 1\mathrm{st}\mathit{Child},\ 2\mathrm{nd}\mathit{Child},\ \mathit{Has\text{-}No}\text{-}1\mathrm{st}\mathit{Child},\ \mathit{Has\text{-}No}\text{-}2\mathrm{nd}\mathit{Child} \,\},$$

where *Root*, *Has-No*-1st*Child*, *Has-No*-2nd*Child* are unary relation symbols and 1st*Child*, 2nd*Child* are binary relation symbols. We define *Trees* to be the set of all τ_{Trees}-structures T that satisfy the following conditions:

(1) $\mathcal{U}^T \subset \{1,2\}^*$ and for every string $si \in \mathcal{U}^T$ with $i \in \{1,2\}$ we also have $s \in \mathcal{U}^T$.
(2) Root^T consists of the empty string ε.
(3) $1\mathrm{st}\mathit{Child}^T$ consists of the pairs $\big(s, s1\big)$, for all $s1 \in \mathcal{U}^T$.
(4) $2\mathrm{nd}\mathit{Child}^T$ consists of the pairs $\big(s, s2\big)$, for all $s2 \in \mathcal{U}^T$.
(5) $\mathit{Has\text{-}No}\text{-}1\mathrm{st}\mathit{Child}^T$ consists of all strings $s \in \mathcal{U}^T$ with $s1 \notin \mathcal{U}^T$.
(6) $\mathit{Has\text{-}No}\text{-}2\mathrm{nd}\mathit{Child}^T$ consists of all strings $s \in \mathcal{U}^T$ with $s2 \notin \mathcal{U}^T$.

For $T \in \mathit{Trees}$ and $t \in \mathcal{U}^T$ we write T_t to denote the subtree of T with root t.

A *schema* σ is a set of unary relation symbols each of which is distinct from *Has-No*-1st*Child*, *Has-No*-2nd*Child*, *Root*. A σ-labelled tree is a $(\tau_{\mathit{Trees}} \cup \sigma)$-structure consisting of some $T \in \mathit{Trees}$ and additional interpretations $P^T \subseteq \mathcal{U}^T$ for all symbols $P \in \sigma$. We sometimes write $\mathit{label}(t)$ to denote the set $\{P \in \sigma : t \in P^T\}$ of labels at vertex t in T.

We identify a string $w = w_0 \cdots w_{n-1}$ of length $|w| = n \geqslant 1$ over an alphabet Σ with a σ-labelled tree T^w in the following way: We choose σ to consist of a unary relation symbol P_a for each letter $a \in \Sigma$, we choose T^w to be the (unique) element in *Trees* with universe $\mathcal{U}^{T^w} = \{\varepsilon, 1, 11, ..., 1^{n-1}\}$, and we choose

$P_a^{T^w} := \{1^i : w_i = a\}$, for each $a \in \Sigma$. This corresponds to the conventional representation of strings by structures in the sense that $\langle \mathcal{U}^{T^w}, 1\text{st}\mathit{Child}, (P_a^{T^w})_{a\in\Sigma}\rangle$ is isomorphic to the structure $\langle \{0,...,n-1\}, \mathit{Succ}, (P_a^w)_{a\in\Sigma}\rangle$ where *Succ* denotes the binary successor relation on $\{0,...,n-1\}$ and P_a^w consists of all positions of w that carry the letter a. When reasoning about strings in the context of first-order logic, we sometimes also need the linear ordering $<$ on $\{0,...,n-1\}$ (respectively, the transitive closure of the relation 1st*Child*). In these cases we explicitly write FO($<$) rather than FO to indicate that the linear ordering is necessary.

XML-documents are usually modelled as ordered unranked trees and not as binary trees. Here *ordered* refers to the fact that the order of the children of a vertex is given. However, a standard representation of ordered unranked trees as relational structures uses binary relations 1st*Child*, *Next-Sibling* and unary relations *Root*, *Leaf*, *Last-Sibling* (for details, see [13]) and thus essentially represents ordered unranked trees as binary trees. Therefore, all our results also apply to ordered unranked trees.

1.3. Logics and queries

We assume that the reader is familiar with *first-order logic*, for short: FO, and with *monadic second-order logic*, for short: MSO (*cf., e.g.*, the textbooks [7, 19]). We use FO(τ) and MSO(τ), respectively, to denote the class of all first-order formulas and monadic second-order formulas, respectively, of signature τ. We write $\varphi(x_1,...,x_k,X_1,...,X_\ell)$ to indicate that the free first-order variables of the formula φ are $x_1,...,x_k$ and the free set variables are $X_1,...,X_\ell$. Sometimes we use $\overline{x}$ and $\overline{X}$ as abbreviations for sequences $x_1,...,x_k$ and $X_1,...,X_\ell$ of variables.

A formula $\varphi(x)$ of signature τ defines the *unary query* which associates with every τ-structure $\mathcal{A}$ the set of elements $a \in \mathcal{U}^{\mathcal{A}}$ such that $\mathcal{A} \models \varphi(a)$, *i.e.*, $\mathcal{A}$ satisfies φ when interpreting the free occurrences of the variable x by the element a. A *sentence* φ of signature τ (*i.e.*, a formula that has no free variables) defines the *Boolean query* that associates the answer "yes" with all τ-structures that satisfy φ and the answer "no" with all other τ-structures.

Apart from FO and MSO we will also consider *monadic least fixed point logic* MLFP (*cf., e.g.*, [7]) which is the extension of first-order logic by unary least fixed point operators, defined as follows: let τ be a signature and let $\varphi(x,X)$ be a formula of signature τ which is *positive* in the variable X, *i.e.*, every atom of the form $X(z)$ occurs in φ within an *even* number of negation symbols. φ defines for every τ-structure $\mathcal{A}$ a monotone operator[1] $F_{\mathcal{A},\varphi} : 2^{\mathcal{U}^{\mathcal{A}}} \to 2^{\mathcal{U}^{\mathcal{A}}}$ *via* $F_{\mathcal{A},\varphi}(A) := \{a \in \mathcal{U}^{\mathcal{A}} : \mathcal{A} \models \varphi(a,A)\}$, for every $A \subseteq \mathcal{U}^{\mathcal{A}}$. A set A is called a *fixed point* (respectively, *pre fixed point*, respectively, *post fixed point*) of φ in $\mathcal{A}$ iff $F_{\mathcal{A},\varphi}(A) = A$ (respectively, $F_{\mathcal{A},\varphi}(A) \subseteq A$, respectively, $F_{\mathcal{A},\varphi}(A) \supseteq A$).

For all $s \in \mathbb{N}$ we define $L^0_{\mathcal{A},\varphi} := \emptyset$, $L^{s+1}_{\mathcal{A},\varphi} := F_{\mathcal{A},\varphi}(L^s_{\mathcal{A},\varphi})$, and $G^0_{\mathcal{A},\varphi} := \mathcal{U}^{\mathcal{A}}$, $G^{s+1}_{\mathcal{A},\varphi} := F_{\mathcal{A},\varphi}(G^s_{\mathcal{A},\varphi})$. Since $F_{\mathcal{A},\varphi}$ is monotone we have $L^s_{\mathcal{A},\varphi} \subseteq L^{s+1}_{\mathcal{A},\varphi}$ and $G^s_{\mathcal{A},\varphi} \supseteq G^{s+1}_{\mathcal{A},\varphi}$, for all $s \in \mathbb{N}$. Since $\mathcal{U}^{\mathcal{A}}$ is finite, a fixed point will be reached eventually, *i.e.*,

[1] An operator $F : 2^M \to 2^M$ is *monotone* iff $F(A) \subseteq F(B)$ for all $A \subseteq B \subseteq M$.

there is a s_0 such that $L^{s_0}_{\mathcal{A},\varphi} = L^{s_0+1}_{\mathcal{A},\varphi} =: L^{\infty}_{\mathcal{A},\varphi}$ and $G^{s_0}_{\mathcal{A},\varphi} = G^{s_0+1}_{\mathcal{A},\varphi} =: G^{\infty}_{\mathcal{A},\varphi}$. We define $L^{\infty}_{\mathcal{A},\varphi}$ to be the *least fixed point* and $G^{\infty}_{\mathcal{A},\varphi}$ to be the *greatest fixed point* of φ in $\mathcal{A}$. It is straightforward to see that $L^{\infty}_{\mathcal{A},\varphi}$ is indeed contained in every *pre fixed point* of φ in $\mathcal{A}$, and that every *post fixed point* of φ in $\mathcal{A}$ is contained in $G^{\infty}_{\mathcal{A},\varphi}$. The sets $L^{s}_{\mathcal{A},\varphi}$, for $s \in \mathbb{N}$, are called the *stages* of the least fixed point of φ in $\mathcal{A}$.

The logic MLFP(τ) is the extension of FO(τ) by least and greatest fixed point operators. *I.e.*: MLFP(τ) contains FO(τ) and is closed under Boolean connectives and first-order quantifications; and if $\varphi(x, X, \overline{y}, \overline{Y})$ is an MLFP(τ)-formula which is positive in the variable X then $[\mathrm{LFP}_{x,X}\,\varphi](z)$ and $[\mathrm{GFP}_{x,X}\,\varphi](z)$ are MLFP(τ)-formulas such that for every $\big(\tau \cup \{\overline{y}, \overline{Y}\}\big)$-structure $\mathcal{A}$ and every element $a \in \mathcal{U}^{\mathcal{A}}$ we have $\mathcal{A} \models [\mathrm{LFP}_{x,X}\,\varphi](a)$ iff $a \in L^{\infty}_{\mathcal{A},\varphi}$, and $\mathcal{A} \models [\mathrm{GFP}_{x,X}\,\varphi](a)$ iff $a \in G^{\infty}_{\mathcal{A},\varphi}$.

It is well-known that the *greatest* fixed point is the dual of the *least* fixed point, *i.e.*, that the formula $[\mathrm{GFP}_{x,X}\varphi](z)$ is equivalent to the formula $\neg[\mathrm{LFP}_{x,X}\neg\varphi'](z)$, where φ' is obtained from φ by replacing every atom of the form $X(u)$ by the literal $\neg X(u)$. Therefore, every MLFP-formula Ψ can easily be transformed into an equivalent MLFP-formula Ψ' in *negation normal form*, where negation symbols "$\neg$" only occur directly in front of *atomic* subformulas.

1.4. Formula size and succinctness

In a natural way, we view formulas as finite trees, where leaves correspond to the atoms of the formulas and inner vertices correspond to Boolean connectives, quantifiers, and fixed-point operators. We define the *size* $||\varphi||$ of a formula φ to be the number of vertices of the tree that corresponds to φ.

Note that this measure of formula size is a *uniform cost measure* in the sense that it accounts just 1 cost unit for each variable and relation symbol appearing in a formula, no matter what its index is. An alternative is to define the size of a formula as the length of a binary encoding of the formula. Such a *logarithmic cost measure* is, for example, used in [11]. Switching between a uniform and a logarithmic measure usually involves a logarithmic factor.

Definition 1.1 (Succinctness)**.** Let L_1 and L_2 be logics, let F be a class of functions from $\mathbb{N}$ to $\mathbb{N}$, and let $\mathcal{C}$ be a class of structures.
We say that L_1 *is F-succinct in L_2 on $\mathcal{C}$* iff there is a function $f \in F$ such that for every formula $\varphi_1 \in L_1$ there is a formula $\varphi_2 \in L_2$ of size $||\varphi_2|| \leqslant f(||\varphi_1||)$ which is equivalent to φ_1 on all structures in $\mathcal{C}$.

Intuitively, a logic L_1 being F-succinct in a logic L_2 means that F gives an upper bound for the size of L_2-formulas needed to express *all* of L_1. This definition may seem slightly at odds with the common use of the term "succinctness" in statements such as "L_1 is exponentially *more succinct* than L_2" meaning that there is *some* L_1-formula that is not equivalent to any L_2-formula of subexponential size. In our terminology, we would rephrase this last statement as "L_1 is *not* $2^{o(n)}$-succinct in L_2" (here we interpret subexponential as $2^{o(n)}$, but of course this is not the issue). The reason for defining F-succinctness the way we did is that

it makes the formal statements of our results much more convenient. We will continue to use statements such as "L_1 is exponentially more succinct than L_2" in informal discussions.

Example 1.2. MLFP is $\mathcal{O}(m)$-succinct in MSO on the class of all finite structures, because every formula $[\mathrm{LFP}_{x,X}\varphi(x, X, \overline{y}, \overline{Y})](z)$ is equivalent to $\forall X \big(Xz \vee \exists x\, \neg Xx \wedge \varphi(x, X, \overline{y}, \overline{Y})\big)$.

Example 1.3. MLFP is $\mathcal{O}(m)$-succinct in the fragment of MLFP whose formulas are in negation normal form on the class of all structures. This allows us to usually work with formulas in negation normal form.

2. From MSO to MLFP

By the standard translation from MSO-logic to tree automata (*cf., e.g.*, [24]) one knows that every MSO-sentence Φ can be translated into a nondeterministic tree automaton with $\mathit{Tower}\big(\mathcal{O}(||\Phi||)\big)$ states that accepts exactly those labelled trees that satisfy Φ. This leads to

Theorem 2.1 (Folklore)**.** MSO*-sentences are* $\mathit{Tower}\big(\mathcal{O}(m)\big)$*-succinct in* MLFP *on the class of all labelled trees.*

To show that we cannot do essentially better, *i.e.*, that there is no translation from MSO to MLFP of size $\mathit{Tower}\big(o(m)\big)$ we need a complexity theoretic assumption that, however, does not seem to be too far-fetched. Let SAT denote the NP-complete satisfiability problem for propositional formulas in conjunctive normal form. Until now, all known deterministic algorithms that solve SAT have worst-case complexity $2^{\Omega(n)}$ (*cf.*, [6]). Although not answering the P *vs.* NP question, the exposition of a deterministic algorithm for SAT with worst-case complexity $\leqslant n^{\lg n}$ would be a surprising and unexpected breakthrough in the SAT-solving community.

In the following, we write $\lg^{(i)}$ to denote the i times iterated logarithm, inductively defined by $\lg^{(1)}(n) := \lg(n)$ and $\lg^{(i+1)}(n) := \lg(\lg^{(i)}(n))$. Moreover, we write $\lg^*$ to denote the "inverse" of the *Tower* function, that is, the (unique) integer valued function with $\mathit{Tower}(\lg^*(n)-1) < n \leqslant \mathit{Tower}(\lg^*(n))$.

Theorem 2.2. *Unless* SAT *is solvable by a deterministic algorithm that has, for every* $i \in \mathbb{N}$*, time bound* $||\gamma||^{\lg^{(i)}(n)}$ *(where* γ *is the input formula and* n *the number of propositional variables occurring in* γ*),* MSO *is not* $\mathit{Tower}\big(o(m)\big)$*-succinct in* MLFP *on the class of all finite strings.*

The overall proof idea is to assume that the function f specifies the size of the translation from MSO to MLFP and to exhibit a SAT-solving algorithm which

- constructs a string w that represents the SAT-instance γ;
- constructs an MSO-formula $\Phi(z)$ of extremely small size that, when evaluated in w, specifies a canonical satisfying assignment for γ (if γ is satisfiable at all);

- tests, for all MLFP-formulas $\Psi(z)$ of size $\leqslant f(||\Phi||)$, whether Ψ specifies a satisfying assignment for γ.

Before presenting the proof in detail we provide the necessary notations and lemmas:

It is straightforward to see

Lemma 2.3. *There is an algorithm that, given an* MLFP*-formula $\Psi(z)$, a string w, and a position p in w, decides in time $|w|^{\mathcal{O}(||\Psi||)}$ whether $w \models \Psi(p)$.*

Proof. It is straightforward to exhibit an algorithm $\mathcal{A}$ with time bound $|w|^{\mathcal{O}(||\Psi||)}$ such that, given $k, \ell \in \mathbb{N}$, an MLFP-formula $\Psi(x_1, ..., x_k, X_1, ..., X_\ell)$, a string w, a sequence $p_1, ..., p_k$ of positions in w, and sets $P_1, ..., P_\ell$ of positions in w, $\mathcal{A}$ decides whether $w \models \Psi(p_1, ..., p_k, P_1, ..., P_\ell)$:

$\mathcal{A}$ operates by recursion on the construction of Ψ. The only non-trivial case is when Ψ is of the form $[\mathrm{LFP}_{y,Y}\Psi'(y, Y, x_1, ..., x_k, X_1, ..., X_\ell)](x_i)$. In this case, $\mathcal{A}$ computes the stages of the least fixed point of Ψ'. *I.e.*, $\mathcal{A}$ performs the following operations:

(1) Initialise the set $L^0 := \emptyset$.
(2) For $s := 1$ to $|w|$ do
  (a) Initialise the set $L^s := \emptyset$.
  (b) For $q := 0$ to $|w|-1$ do
    check whether $w \models \Psi'(q, L^{s-1}, p_1, ..., p_k, P_1, ..., P_\ell)$;
    if so, then insert q into L^s.
(3) Check whether p_i belongs to $L^{|w|}$;
  if so, then STOP with output "yes", otherwise STOP with output "no".

This computation takes $\mathcal{O}\big(|w|^2 \cdot |w|^{\mathcal{O}(||\Psi'||)}\big) = |w|^{\mathcal{O}(||\Psi||)}$ steps. □

Let us now concentrate on the construction of a string w that represents a SAT-instance γ and of an MSO-formula $\Phi(z)$ that specifies a canonical satisfying assignment of γ (provided that γ is satisfiable at all). Since we want Φ to be extremely short, we cannot choose w to be the straightforward string-representation of γ. Instead, we use the following, more complicated, representation of [11]:

For all $h \geqslant 1$ let $\Sigma_h := \big\{0, 1, \texttt{<1>}, \texttt{</1>}, ..., \texttt{<h>}, \texttt{</h>}\big\}$. The "tags" `<i>` and `</i>` represent single letters of the alphabet and are just chosen to improve readability. For every $n \geqslant 1$ let $L(n)$ be the length of the binary representation of the number $n-1$, *i.e.*, $L(0) = 0$, $L(1) = 1$, and $L(n) = \lfloor \lg(n-1) \rfloor + 1$, for all $n \geqslant 2$. By $\mathrm{bit}(i, n)$ we denote the i-th bit of the binary representation of n, *i.e.,* $\mathrm{bit}(i, n)$ is 1 if $\lfloor \frac{n}{2^i} \rfloor$ is odd, and $\mathrm{bit}(i, n)$ is 0 otherwise.

We encode every number $n \in \mathbb{N}$ by a string $\mu_h(n)$ over the alphabet Σ_h, where $\mu_h(n)$ is inductively defined as follows: $\mu_1(0) := \texttt{<1></1>}$, and

$$\mu_1(n) \;:=\; \texttt{<1>}\,\mathrm{bit}(0, n-1)\,\mathrm{bit}(1, n-1) \cdots \mathrm{bit}(L(n)-1, n-1)\,\texttt{</1>}\,,$$

for $n \geqslant 1$. For $h \geqslant 2$ we let $\mu_h(0) := \texttt{<h></h>}$ and

$$\begin{aligned}\mu_h(n) := \texttt{<h>}\\ &\mu_{h-1}(0)\,\mathrm{bit}(0,n{-}1)\\ &\mu_{h-1}(1)\,\mathrm{bit}(1,n{-}1)\\ &\quad\vdots\\ &\mu_{h-1}(L(n){-}1)\,\mathrm{bit}(L(n){-}1,n{-}1)\\ \texttt{</h>},\end{aligned}$$

for $n \geqslant 1$. Here empty spaces and line breaks are just used to improve readability.

To encode a CNF-formula γ by a string we use the alphabet

$$\begin{aligned}\Sigma'_h \;:=\; &\Sigma_h \cup\\ &\{+,\ -,\ \texttt{<lit>},\ \texttt{</lit>},\ \texttt{<clause>},\ \texttt{</clause>},\ \texttt{<cnf>},\ \texttt{</cnf>}\} \cup\\ &\{\star,\ \texttt{<ass>},\ \texttt{</ass>},\ \texttt{<val>},\ \texttt{</val>}\}.\end{aligned}$$

Let $i \in \mathbb{N}$ and let X_i be a propositional variable. The literal X_i is encoded by the string

$$\mu_h(X_i) \;:=\; \texttt{<lit>}\,\mu_h(i) + \texttt{</lit>},$$

and the literal $\neg X_i$ is encoded by $\mu_h(\neg X_i) \;:=\; \texttt{<lit>}\,\mu_h(i) - \texttt{</lit>}$.

A clause $\delta := \lambda_1 \vee \cdots \vee \lambda_r$ of literals is encoded by

$$\mu_h(\delta) \;:=\; \texttt{<clause>}\,\mu_h(\lambda_1)\cdots\mu_h(\lambda_r)\,\texttt{</clause>}.$$

A CNF-formula $\gamma := \delta_1 \wedge \cdots \wedge \delta_m$ is encoded by the string

$$\mu_h(\gamma) \;:=\; \texttt{<cnf>}\,\mu_h(\delta_1)\cdots\mu_h(\delta_m)\,\texttt{</cnf>}.$$

We write $\mathrm{CNF}(n)$ to denote the class of all CNF-formulas the propositional variables of which are among $X_0, ..., X_{n-1}$. To provide the "infrastructure" for specifying a truth assignment to the variables $X_0, ..., X_{n-1}$, we use the string

$$\begin{aligned}\mu_h(X_0,...,X_{n-1}) \;:=\; &\texttt{<ass>}\\ &\texttt{<val>}\,\mu_h(0)\,\star\,\texttt{</val>}\\ &\texttt{<val>}\,\mu_h(1)\,\star\,\texttt{</val>}\\ &\qquad\vdots\\ &\texttt{<val>}\,\mu_h(n{-}1)\,\star\,\texttt{</val>}\\ &\texttt{</ass>}.\end{aligned}$$

Remark 2.4. There is a 1–1-correspondence between assignments $\alpha : \{X_0, ..., X_{n-1}\} \to \{\mathit{true}, \mathit{false}\}$, on the one hand, and sets P of positions of $\mu_h(X_0, ..., X_{n-1})$ that carry the letter $\star$, on the other hand: such a set P specifies the assignment α^P that, for each $i < n$, maps the variable X_i to the value *true* iff the $\star$-position directly after the substring $\mu_h(i)$ in $\mu_h(X_0, ..., X_{n-1})$ belongs to P. Conversely, a given assignment α specifies the set P^α consisting of exactly those $\star$-positions of $\mu_h(X_0, ..., X_{n-1})$ that occur directly after a substring $\mu_h(i)$ where $\alpha(X_i) = \mathit{true}$.

Finally, we encode a formula $\gamma \in \mathrm{CNF}(n)$ by the string

$$\mu_h(\gamma, \star) \;:=\; \mu_h(\gamma)\, \mu_h(X_0, ..., X_{n-1}).$$

$\mu_h(\gamma, \star)$ is the string w that we will further on use as the representative of a SAT-instance γ. We use the following result of [11]:

Lemma 2.5.

(a) There is an algorithm that, given $h \in \mathbb{N}$ and $\gamma \in$ CNF(n), computes (a binary representation of) the string $\mu_h(\gamma, \star)$ in time $\mathcal{O}\big(h \cdot (\lg h) \cdot (\lg n)^2 \cdot (||\gamma|| + n)\big)$ (cf., [11], Lem. 9).
The string $\mu_h(\gamma, \star)$ has length $|\mu_h(\gamma, \star)| = \mathcal{O}\big(h \cdot (\lg n)^2 \cdot (||\gamma|| + n)\big)$.

(b) There is an algorithm that, given $h \in \mathbb{N}$, computes (the binary representation of) a FO(<)*-formula $\varphi_h(Z)$ in time $\mathcal{O}\big(h \cdot \lg h\big)$, such that for all $n \leqslant$ Tower(h), for all $\gamma \in$ CNF(n), and for all sets P of $\star$-positions in the string $\mu_h(\gamma, \star)$ we have*

$$\mu_h(\gamma, \star) \models \varphi_h(P) \quad \textit{iff} \quad \alpha^P \textit{ is a satisfying assignment for } \gamma$$

(cf., [11], Lem. 10). The formula $\varphi_h(Z)$ has size[2] *$||\varphi_h(Z)|| = \mathcal{O}(h)$.*

Given a CNF(n)-formula γ and its representative $\mu_h(\gamma, \star)$, we now specify a *canonical* satisfying assignment of γ, provided that γ is satisfiable at all. As observed in Remark 2.4, every assignment $\alpha : \{X_0, ..., X_{n-1}\} \to \{true, false\}$ corresponds to a set P^α of positions in $\mu_h(\gamma, \star)$ that carry the letter $\star$. P^α, again, can be identified with the 0-1-string of length $|\mu_h(\gamma, \star)|$ that carries the letter 1 exactly at those positions that belong to P^α. Now, the lexicographic ordering of these strings gives us a linear ordering on the set of all assignments $\alpha : \{X_0, ..., X_{n-1}\} \to \{true, false\}$. As the canonical satisfying assignment of γ we choose the lexicographically smallest satisfying assignment.

Lemma 2.6. *There is an algorithm that, given $h \in \mathbb{N}$, computes (the binary representation of) an* MSO*-formula $\Phi_h(z)$ in time $\mathcal{O}\big(h \cdot \lg h\big)$, such that for all $n \leqslant$ Tower(h), for all $\gamma \in$ CNF(n), and for all positions p of $\mu_h(\gamma, \star)$ that carry the letter $\star$, we have*

$$\mu_h(\gamma, \star) \models \Phi_h(p) \quad \textit{iff} \quad \textit{in the lexicographically smallest satisfying assignment for } \gamma\textit{, the propositional variable corresponding to position } p \textit{ is assigned the value true.}$$

The formula $\Phi_h(z)$ has size $||\Phi_h|| = \mathcal{O}(h)$.

Proof. First, use the algorithm of Lemma 2.5 (b) to construct the FO(<)-formula $\varphi_h(Z)$. It is straightforward to verify that the following formula has the desired

[2] In [11], an additional factor $\lg h$ occurs because there a logarithmic cost measure is used for the formula size, whereas here we use a uniform measure (*cf.*, Sect. 1.4).

properties:

$$\Phi_h(z) \;:=\; \exists\, Z\, \Big((\forall x\; Zx \to P_\star x) \;\wedge\; Zz \;\wedge\; \varphi_h(Z) \;\wedge \\ \forall\, Z'\, \big((\forall x\; Z'x \to P_\star x) \;\wedge\; \varphi_h(Z') \big) \;\to\; Z \leqslant_{\text{lex}} Z' \Big),$$

where $Z \leqslant_{\text{lex}} Z'$ is an abbreviation for the FO(<)-formula

$$(\forall x\; Zx \leftrightarrow Z'x) \;\vee\; \exists y \left(\neg Zy \;\wedge\; Z'y \;\wedge\; \forall x \left(x{<}y \;\to\; (Zx \leftrightarrow Z'x) \right) \right).$$

Afterwards, we replace every occurrence of an atom of the form $x < y$ in $\Phi_h(z)$ by the MSO-formula $\exists Y \big(Yy \;\wedge\; \neg Yx \;\wedge\; \forall z_1 \forall z_2 \, (\mathit{Succ}(z_1, z_2) \wedge Y z_1) \to Y z_2 \big)$. □

Finally, we are ready for the Proof of Theorem 2.2:

Proof of Theorem 2.2. Let $f : \mathbb{N} \to \mathbb{N}$ be a function such that there is, for every MSO-formula $\Phi(z)$, a MLFP-formula $\Psi(z)$ of size $||\Psi|| \leqslant f(||\Phi||)$ which defines the same query as Φ on the class of all finite strings (recall that such an f does indeed exist, because MSO and MLFP have the same expressive power over the class of finite strings).

Consider the algorithm displayed in Figure 2, which decides if the input formula γ is satisfiable.

The correctness of this algorithm directly follows from Lemma 2.6 and from the fact that at least one of the formulas $\Psi(z)$ of size $\leqslant f(||\Phi_h||)$ defines the same query as $\Phi_h(z)$.

It remains to determine the worst-case running time of the algorithm. Let γ be an input CNF-formula for the algorithm, let n be the number of propositional variables of γ, and let $h := \lg^*(n)$.

The steps 1–4 of the algorithm will be performed within a number of steps polynomial in $||\gamma||$, and the MSO-formula $\Phi_h(z)$ produced in step 4 will have size $||\Phi_h|| \leqslant c \cdot h$, for a suitable constant $c \in \mathbb{N}$ (*cf.*, Lems. 2.5 (a) and 2.6).

The loop in step 5 will be performed for at most $2^{c_1 \cdot f(||\Phi_h||) \cdot \lg(f(||\Phi_h||))}$ times, for a suitable constant $c_1 \in \mathbb{N}$. To see this, note that formulas of length $\leqslant f(||\Phi_h||)$ use at most $f(||\Phi_h||)$ different first-order variables and at most $f(||\Phi_h||)$ different set variables. *I.e.,* these formulas can be viewed as strings of length $f(||\Phi_h||)$ over an alphabet of size $c_2 + 2 \cdot f(||\Phi_h||)$, for a suitable constant $c_2 \in \mathbb{N}$. Therefore, the number of such formulas is $\leqslant (c_2 + 2{\cdot}f(||\Phi_h||))^{f(||\Phi_h||)} \leqslant 2^{c_1 \cdot f(||\Phi_h||) \cdot \lg(f(||\Phi_h||))}$.

Each performance of the loop in step 5 will take a number of steps polynomial in

$$|\mu_h(\gamma, \star)|^{\mathcal{O}(f(||\Phi_h||))} \;\leqslant\; \big(c_3 \cdot h \cdot (\lg n)^2 \cdot ||\gamma|| \big)^{c_4 \cdot f(c \cdot h)},$$

for suitable constants $c_3, c_4 \in \mathbb{N}$ (*cf.*, Lems. 2.3 and 2.5 (a)). Altogether, for suitable constants $c, d \in \mathbb{N}$, the algorithm will perform the steps 1–6 within $||\gamma||^{d \cdot f(c \cdot h) \cdot \lg(f(c \cdot h))}$ steps.

Now let us suppose that f has bound $f(m) \leqslant \mathit{Tower}\big(o(m)\big)$. From Lemma 2.7 below we then obtain that our SAT-solving algorithm has, for every $i \in \mathbb{N}$, time bound $||\gamma||^{\lg^{(i)}(n)}$. This finally completes the proof of Theorem 2.2. □

Input: a SAT-instance γ in CNF

(1) Count the number n of propositional variables occurring in γ, and modify γ in such a way that only the propositional variables $X_0, ..., X_{n-1}$ occur in it.
(2) Compute $h := \lg^*(n)$, *i.e.*, choose $h \in \mathbb{N}$ such that $\mathit{Tower}(h{-}1) < n \leqslant \mathit{Tower}(h)$.
(3) Construct the string $\mu_h(\gamma, \star)$ that represents γ (see Lem. 2.5 (a)).
(4) Construct an MSO-formula $\Phi_h(z)$ that has the following property: Whenever p is a position in $\mu_h(\gamma, \star)$ that carries the letter $\star$, we have

$\mu_h(\gamma, \star) \models \Phi_h(p)$ iff in the lexicographically smallest satisfying assignment for γ, the propositional variable corresponding to position p is assigned the value *true*

(*cf.*, Lem. 2.6).
(5) For all MLFP-formulas $\Psi(z)$ of size $||\Psi|| \leqslant f(||\Phi_h||)$ do:
  (a) Initialise the assignment $\alpha := \emptyset$.
  (b) For all positions p in $\mu_h(\gamma, \star)$ that carry the letter $\star$ do
    check whether $\mu_h(\gamma, \star) \models \Psi(p)$;
    if so, then insert the propositional variable corresponding to p into α.
  (c) Check whether α is a *satisfying* assignment for γ;
    if so, then STOP with output "γ is satisfiable *via* assignment α".
(6) STOP with output "γ is not satisfiable".

FIGURE 2. A SAT-solving algorithm.

Lemma 2.7. *Let $f : \mathbb{N} \to \mathbb{N}$ be a function with bound $f(m) \leqslant \mathit{Tower}\big(o(m)\big)$, and let $c, d \in \mathbb{N}$. For every $i \in \mathbb{N}$ there is an $n_0 \in \mathbb{N}$ such that for all $n \geqslant n_0$ we have*

$$d \cdot f\left(c \cdot \lg^*(n)\right) \cdot \lg\left(f(c \cdot \lg^*(n))\right) \quad \leqslant \quad \lg^{(i)}(n).$$

Proof. Since f has bound $f(m) \leqslant \mathit{Tower}\big(o(m)\big)$, we know that there is a $m_0 \in \mathbb{N}$ and a function $g : \mathbb{N} \to \mathbb{N}$ with $g \in \omega(1)$ such that $f(m) \leqslant \mathit{Tower}\big(\frac{m}{g(m)}\big)$, for all $m \geqslant m_0$. Therefore, for $h := \lg^*(n)$ and $m := c \cdot h$ we have

$$d \cdot f(m) \cdot \lg(f(m)) \leqslant d \cdot \mathit{Tower}\left(\frac{m}{g(m)}\right) \cdot \mathit{Tower}\left(\frac{m}{g(m)} - 1\right) \leqslant \mathit{Tower}\left(\frac{m}{g(m)} + 1\right).$$

On the other hand, $n \geqslant \mathit{Tower}(h{-}1)$, and thus $\lg^{(i)}(n) \geqslant \mathit{Tower}(h{-}(i{+}1))$. It therefore suffices to show that $\frac{m}{g(m)} + 1 \leqslant h{-}(i{+}1)$, *i.e.*, that $\frac{m}{g(m)} \leqslant h{-}(i{+}2)$. Since $g \in \omega(1)$, there is some $m_1 \geqslant m_0$ such that for all $m \geqslant m_1$ we have

$g(m) \geqslant c \cdot (i{+}3)$. Hence,

$$\frac{m}{g(m)} = \frac{c \cdot \big(h{-}(i{+}3)\big)}{g(m)} + \frac{c \cdot (i{+}3)}{g(m)} \leqslant h{-}(i{+}3) + 1 = h{-}(i{+}2).$$

This completes the proof of Lemma 2.7. □

3. The two-variable fragment of MLFP and the full modal μ-calculus

Defining the 2-variable fixed-point logics requires some care: MLFP^2 is the fragment of MLFP consisting of all formulas with just 2 first-order variables and no parameters in fixed point operators, *i.e.*, for all subformulas of the form $[\mathrm{LFP}_{x,X}\varphi](y)$, x is the only free first-order variable of φ (note, however, that φ may have several free monadic second-order variables). This is the monadic fragment of the standard 2-variable least fixed-point logic (*cf.* [14]). Without the restriction on free first-order variables in fixed-point operators we obtain full MLFP even with just two individual variables (we prove this in [16]).

We first note that MLFP^2, and actually FO^2, the two variable fragment of first-order logic, is doubly exponentially more succinct than nondeterministic automata on the class of all finite strings:

Example 3.1. Let $\sigma := \{L, R, P_1, ..., P_n\}$ and

$$\varphi_n := \forall x \left(Lx \to \exists y \big(Ry \wedge \bigwedge_{i=1}^{n} (P_i x \leftrightarrow P_i y) \big) \right).$$

We claim that every nondeterministic finite automaton accepting precisely those strings over alphabet 2^σ that satisfy φ has at least 2^{2^n} states. To see this, for every $S \subseteq 2^{\{1,\ldots,n\}}$, we define strings $X_n(S)$ and $Y_n(S)$ such that

- $L^{X_n(S)} = \mathcal{U}^{X_n(S)}$ and $R^{Y_n(S)} = \mathcal{U}^{Y_n(S)}$.
- For all $x \in \mathcal{U}^{X_n(S)}$ we have $\{i \mid x \in P_i^{X_n(S)}\} \in S$, and for all $y \in \mathcal{U}^{Y_n(S)}$ we have $\{i \mid y \in P_i^{Y_n(S)}\} \in S$.
- For all $s \in S$ there exists an $x \in \mathcal{U}^{X_n(S)}$ and an $y \in \mathcal{U}^{Y_n(S)}$ such that $s = \{i \mid x \in P_i^{X_n(S)}\} = \{i \mid y \in P_i^{Y_n(S)}\}$.

For $S, T \subseteq 2^{\{1,\ldots,n\}}$ let $W_n(S,T) := X_n(S)\, Y_n(T)$ be the concatenation of $X_n(S)$ and $Y_n(T)$. Then $W_n(S,T) \models \varphi \iff S \subseteq T$. Clearly, a nondeterministic finite automaton accepting precisely those strings $W_n(S,T)$ with $S \subseteq T$ needs at least 2^{2^n} states.

Let us return to binary trees now. Following Vardi [25], we define the *full modal μ-calculus* FL_μ on binary trees as follows:

Definition 3.2 (FL_μ)**.** For each schema σ, an FL_μ-formula of schema σ is either:

- *true*, *false*, P, or $\neg P$, where $P \in \sigma \cup \{$*Root*, *Has-No-*1st*Child*, *Has-No-*2nd*Child*$\}$;

- $\Phi_1 \wedge \Phi_2$ or $\Phi_1 \vee \Phi_2$, where Φ_1 and Φ_2 are FL_μ-formulas of schema σ;
- X, where X is a propositional variable;
- $\langle R\rangle\,\Phi$ or $[R]\,\Phi$, where $R \in \{1\mathrm{st}\mathit{Child},\ 2\mathrm{nd}\mathit{Child},\ 1\mathrm{st}\mathit{Child}^{-1},\ 2\mathrm{nd}\mathit{Child}^{-1}\}$ and Φ is an FL_μ-formula of schema σ;
- $\mu X.\Phi$ or $\nu X.\Phi$, where X is a propositional variable and Φ is an FL_μ-formula of schema σ.

Instead of formally defining the semantics of FL_μ, we just give a translation of FL_μ into MLFP^2. For a σ-labelled tree T and a node $t \in \mathcal{U}^T$, we write $(T,t) \models \Phi$ to denote that the FL_μ-formula Φ *holds at* t *in* T. We identify propositional variables in FL_μ-formulas and set variables in MLFP^2-formulas. For every FL_μ-formula Φ we define an MLFP^2-formula $\varphi(x)$ in such a way that for all trees T and nodes $t \in \mathcal{U}^T$ we have $(T,t) \models \Phi \iff T \models \varphi(t)$.

- If $\Phi = \mathit{true}$ then $\varphi(x) := (x = x)$. If $\Phi = \mathit{false}$ then $\varphi(x) := \neg(x = x)$. If $\Phi = P$ then $\varphi(x) := P(x)$. If $\Phi = \neg P$ then $\varphi(x) := \neg P(x)$.
- If $\Phi = X$ then $\varphi(x) := X(x)$.
- If $\Phi = \Phi_1 \vee \Phi_2$ then $\varphi(x) := \varphi_1(x) \vee \varphi_2(x)$. If $\Phi = \Phi_1 \wedge \Phi_2$ then $\varphi(x) := \varphi_1(x) \wedge \varphi_2(x)$.
- If $\Phi := \langle R\rangle.\Psi$ for $R \in \{1\mathrm{st}\mathit{Child}, 2\mathrm{nd}\mathit{Child}\}$ then $\varphi(x) := \exists y\big(R(x,y) \wedge \psi(y)\big)$, where $\psi(y)$ is obtained from $\psi(x)$ by simultaneously replacing all occurrences of x by y and *vice versa*. Similarly, if $\Phi := \langle R^{-1}\rangle.\Psi$ for $R \in \{1\mathrm{st}\mathit{Child}, 2\mathrm{nd}\mathit{Child}\}$ then $\varphi(x) := \exists y\big(R(y,x) \wedge \psi(y)\big)$.
- If $\Phi := [R].\Psi$ for $R \in \{1\mathrm{st}\mathit{Child}, 2\mathrm{nd}\mathit{Child}\}$ then $\varphi(x) := \forall y\big(R(x,y) \to \psi(y)\big)$. Similarly, if $\Phi := [R^{-1}].\Psi$ for $R \in \{1\mathrm{st}\mathit{Child}, 2\mathrm{nd}\mathit{Child}\}$ then $\varphi(x) := \forall y\big(R(y,x) \to \psi(y)\big)$.
- If $\Phi := \mu X.\Psi$ then $\varphi(x) := [\mathrm{LFP}_{x,X}\psi(x)](x)$. If $\Phi := \nu X.\Psi$ then $\varphi(x) := [\mathrm{GFP}_{x,X}\psi(x)](x)$.

What we have seen above is that FL_μ is $\mathcal{O}(m)$-succinct in MLFP^2. Our next result is that there also is a reverse translation from MLFP^2 to FL_μ which only incurs an exponential blow-up in size:

Theorem 3.3. MLFP^2 *is* $2^{poly(m)}$*-succinct in* FL_μ *on the class of labelled trees.* More precisely: There is a number $c \in \mathbb{N}$ such that for every MLFP^2-formula $\varphi(x)$ there is an FL_μ-formula Φ of size $||\Phi|| \leqslant 2^{(||\varphi||^c)}$ such that for all labelled trees T and all nodes $t \in \mathcal{U}^T$, $T \models \varphi(t)$ iff $(T,t) \models \Phi$.

Proof. Let the *arity* of a formula be the number of free variables it has.

We say that an MLFP^2-formula $\varphi(x, \overline{X})$ of the form $\exists y\,\psi(x,y,\overline{X})$ is in *normal form* if it is in negation normal form and $\psi(x,y,\overline{X})$ is a disjunction of formulas of the form $\alpha \wedge \theta$ satisfying the following conditions:

(i) θ is a conjunction of at most unary formulas.
(ii) α is one of the following five *basic binary formulas*:

$$\begin{aligned}
\alpha^1 &= 1\text{st}\mathit{Child}(x,y),\\
\alpha^2 &= 2\text{nd}\mathit{Child}(x,y),\\
\alpha^3 &= 1\text{st}\mathit{Child}(y,x),\\
\alpha^4 &= 2\text{nd}\mathit{Child}(y,x),\\
\alpha^5 &= \neg 1\text{st}\mathit{Child}(x,y) \wedge \neg 2\text{nd}\mathit{Child}(x,y) \wedge \neg 1\text{st}\mathit{Child}(y,x) \wedge \neg 2\text{nd}\mathit{Child}(y,x).
\end{aligned}$$

Similarly, a formula of the form $\forall y\, \psi(x, y, \overline{X})$ is in normal form if it is a conjunction of formulas δ of the form $\alpha \to \theta$, where

(i') θ is a disjunction of at most unary formulas.
(ii') α is one of the basic binary formulas $\alpha^1, ..., \alpha^5$.

Finally, an arbitrary MLFP^2-formula is in normal form if all its subformulas of the form $\exists y\, \psi$ and $\forall y\, \psi$ are.

Step 1: For every MLFP^2-formula φ we construct an equivalent MLFP^2-formula φ' in normal form.

The construction is by induction on φ, the only interesting cases being subformulas of the form $\exists y\, \psi$ and $\forall y\, \psi$. We only consider the universal case; the existential case can be treated similarly.

So let $\varphi(x, \overline{X}) = \forall y\, \psi(x, y, \overline{X})$. We may view ψ as a Boolean combination of atomic formulas and at most unary formulas. We bring this Boolean combination into conjunctive normal form; let ψ' be the resulting formula. Let us consider a clause γ of ψ'. Let θ be the disjunction of all at most unary literals in γ, and let α' be the conjunction of the negations of the remaining literals. Then, γ is equivalent to $(\alpha' \to \theta)$, and therefore we may replace γ with $(\alpha' \to \theta)$.

α' is a conjunction of literals of the form $(\neg)R(x, y)$ and $(\neg)R(y, x)$, where $R \in \{1\text{st}\mathit{Child}, 2\text{nd}\mathit{Child}\}$. If α' is not satisfiable in any tree, then we may simply discard it from ψ', because $(\alpha' \to \theta)$ always holds. If this makes ψ' empty, we replace the whole formula φ by $(x{=}x)$. So let us assume that α' is satisfiable in some tree. Then α' is equivalent to a disjunction α'' of some of the five basic binary formulas. Say, α' is equivalent to $\bigvee_{j=1}^{\ell} \alpha^{i_j}$, for some $\ell \leqslant 5$. Then we replace γ by the clauses $(\alpha^{i_1} \to \theta), ..., (\alpha^{i_\ell} \to \theta)$.

We do this for all clauses of ψ' and obtain a formula ψ'' equivalent to ψ that has the desired form. This completes step 1.

Before we translate MLFP^2-formulas in normal form into FL$_\mu$-formulas, we define a few auxiliary MLFP^2-formulas. Recall that for a tree T and a node $t \in \mathcal{U}^T$, by T_t we denote the subtree of T with root t.

Step 2: For every FL$_\mu$-formula Φ we construct FL$_\mu$-formulas $\exists\Phi$, $\exists{\downarrow}\Phi$, and $\exists{\uparrow}\Phi$ of size $\mathcal{O}(||\Phi||)$ such that for all labelled trees T and nodes $t \in \mathcal{U}^T$,

(1) $(T, t) \models \exists\Phi$ if, and only if, there exists a $u \in \mathcal{U}^T$ such that $(T, u) \models \Phi$.
(2) $(T, t) \models \exists{\downarrow}\Phi$ if, and only if, there exists a $u \in \mathcal{U}^{T_t}$ such that $(T, u) \models \Phi$.

(3) $(T,t) \models \exists\uparrow\Phi$ if, and only if, there exists a $u \in \mathcal{U}^T \setminus \mathcal{U}^{T_t}$ such that $(T,u) \models \Phi$.

Similarly, there are formulas $\forall\Phi$, $\forall\downarrow\Phi$, and $\forall\uparrow\Phi$ with the obvious meaning.

It is an easy exercise to construct such formulas. Note, however, that we make crucial use of the fact that we are working on trees.

Step 3: For every MLFP^2-formula φ in normal form we construct an FL_μ-formula Φ such that for all labelled trees T and all nodes $t \in \mathcal{U}^T$, $T \models \varphi(t) \iff (T,t) \models \Phi$.

The construction is by induction on φ.

For all unary subformulas that do not start with an existential or universal quantifier, we can simply revert the translation from FL_μ to MLFP^2.

For 0-ary formulas $\varphi = \exists x\psi$, suppose Ψ is the FL_μ-formula already constructed for ψ. We let $\Phi = \exists\Psi$. Similarly, for 0-ary $\varphi = \forall x\psi$, we let $\Phi = \forall\Psi$.

As the first interesting case, let us assume that $\varphi(x,\overline{X}) = \exists y\,\psi(x,y,\overline{X})$ is in normal form. Suppose that $\psi = \bigvee_{i=1}^{\ell}(\alpha_i \wedge \theta_i)$, where θ_i and α_i satisfy the conditions (i) and (ii) for all i. For every i we define a formula Γ_i equivalent to $\exists y\,(\alpha_i \wedge \theta_i)$, and then we let $\Phi = \bigvee_{i=1}^{\ell}\Gamma_i$.

So let $1 \le i \le \ell$. Let θ_x be the conjunction of all at most unary formulas of $(\alpha_i \wedge \theta_i)$ that either do not have any free first-order variables or whose only free first-order variable is x, and let θ_y be the conjunction of all at most unary subformulas of $(\alpha_i \wedge \theta_i)$ whose only free first-order variable is y. If $\alpha_i = R(x,y)$ for $R \in \{1\mathrm{st}\mathit{Child}, 2\mathrm{nd}\mathit{Child}\}$, we let $\Gamma_i := \Theta_x \wedge \langle R\rangle\Theta_y$, where Θ_x and Θ_y are the FL_μ-formulas already constructed for θ_x and θ_y. Similarly, if $\alpha_i = R(y,x)$, we let $\Gamma_i := \Theta_x \wedge \langle R^{-1}\rangle\Theta_y$. If $\alpha_i = \neg 1\mathrm{st}\mathit{Child}(x,y) \wedge \neg 2\mathrm{nd}\mathit{Child}(x,y) \wedge \neg 1\mathrm{st}\mathit{Child}(y,x) \wedge \neg 2\mathrm{nd}\mathit{Child}(y,x)$, we let

$$\begin{aligned}\Gamma_i := \Theta_x \wedge \big(\ & \langle 1\mathrm{st}\mathit{Child}\rangle\langle 1\mathrm{st}\mathit{Child}\rangle\,\exists\downarrow\Theta_y \ \vee\ \langle 1\mathrm{st}\mathit{Child}\rangle\langle 2\mathrm{nd}\mathit{Child}\rangle\,\exists\downarrow\Theta_y \ \vee \\ & \langle 2\mathrm{nd}\mathit{Child}\rangle\langle 1\mathrm{st}\mathit{Child}\rangle\,\exists\downarrow\Theta_y \ \vee\ \langle 2\mathrm{nd}\mathit{Child}\rangle\langle 2\mathrm{nd}\mathit{Child}\rangle\,\exists\downarrow\Theta_y \ \vee \\ & \langle 1\mathrm{st}\mathit{Child}^{-1}\rangle\,\exists\uparrow\Theta_y \ \vee\ \langle 2\mathrm{nd}\mathit{Child}^{-1}\rangle\,\exists\uparrow\Theta_y.\end{aligned}$$

The case $\varphi(x,\overline{X}) = \forall y\,\psi(x,y,\overline{X})$ can be treated similarly. This completes step 3.

It remains to prove that the size of Φ is at most exponential in the size of φ. The *(operator) depth* of a formula is naturally defined as the height of its parse tree.

By induction on d, we prove that for every at most unary subformula ψ of φ of depth d we have

$$||\Psi|| \in 2^{\mathcal{O}(d\cdot||\psi||)}.$$

Again, the only critical cases are unary subformulas of the form $\exists y\ \chi$ or $\forall y\ \chi$. So suppose that ψ has either of these forms. Let $\xi_1,\dots,\xi_m$ be the at most unary subformulas of χ that are not atoms such that χ can be written as a Boolean combination of $\xi_1,\dots,\xi_m$ and atomic subformulas. Let $\Xi_1,\dots,\Xi_m$ be the FL_μ-formulas corresponding to $\xi_1,\dots,\xi_m$. By induction hypothesis, for $1 \le i \le m$ we have

$$||\Xi_i|| \le 2^{\mathcal{O}((d-1)\cdot||\xi_i||)} \le 2^{\mathcal{O}((d-1)\cdot||\psi||)}.$$

An inspection of our construction shows that

$$||\Psi|| \leq 2^{\mathcal{O}(||\psi||)} \cdot \max_{1\leq i\leq m} ||\Xi_i|| \leq 2^{\mathcal{O}(||\psi||)} \cdot 2^{\mathcal{O}((d-1)\cdot||\psi||)} = 2^{\mathcal{O}(d\cdot||\psi||)}.$$

Finally, this completes the proof of Theorem 3.3. □

Theorem 3.4 (Vardi [25])**.** *For every formula Φ of the full modal μ-calculus FL_μ there is a nondeterministic tree automaton of size $2^{poly(||\Phi||)}$ that accepts exactly those labelled trees in which Φ holds at the root.*

As a matter of fact, Vardi [25] proved a stronger version of this theorem for infinite trees and parity tree automata. But on finite trees, a parity acceptance condition can always be replaced by a normal acceptance for finite tree automata.

The Theorems 3.3 and 3.4 directly imply the following:

Corollary 3.5. *For every MLFP^2-formula $\varphi(x)$ there is a nondeterministic tree automaton of size $2^{2^{poly(||\varphi||)}}$ that accepts exactly those labelled trees in which φ holds at the root.*

On the other hand, Theorem 3.4 and Example 3.1 directly imply the following:

Corollary 3.6. MLFP^2 *is not $2^{o(m)}$-succinct in* FL_μ.

4. Simultaneous least fixed point logic

In this section we consider the *simultaneous least fixed point* operator which is defined as follows (*cf., e.g.*, the textbook [7]):

Let τ be a signature and let $\varphi_1(x_1, X_1, ..., X_n), ..., \varphi_n(x_n, X_1, ..., X_n)$ be formulas over the signature τ, each of which is positive in all the variables $X_1, ..., X_n$. For every τ-structure $\mathcal{A}$, every formula φ_i defines a monotone operator $F_{\mathcal{A},\varphi_i} : \big(2^{\mathcal{U}^\mathcal{A}}\big)^n \to 2^{\mathcal{U}^\mathcal{A}}$ *via* $F_{\mathcal{A},\varphi_i}(A_1, ..., A_n) := \{a \in \mathcal{U}^\mathcal{A} : \mathcal{A} \models \varphi_i(a, A_1, ..., A_n)\}$. A tuple $(A_1, ..., A_n)$ is called a *simultaneous fixed point* of $(\varphi_1, ..., \varphi_n)$ in $\mathcal{A}$ iff, for all $i \leqslant n$, $F_{\mathcal{A},\varphi_i}(A_1, ..., A_n) = A_i$.

For all $i \leqslant n$ and $s \in \mathbb{N}$ we define $L^0_{\mathcal{A},\varphi_i} := \emptyset$ and $L^{s+1}_{\mathcal{A},\varphi_i} := F_{\mathcal{A},\varphi_i}\big(L^s_{\mathcal{A},\varphi_1}, ..., L^s_{\mathcal{A},\varphi_n}\big)$. Since $F_{\mathcal{A},\varphi_i}$ is monotone we have $L^s_{\mathcal{A},\varphi_i} \subseteq L^{s+1}_{\mathcal{A},\varphi_i}$ for all $s \in \mathbb{N}$; and since $\mathcal{A}$ is finite a fixed point will be reached eventually, *i.e.*, there is a s_0 such that $L^{s_0}_{\mathcal{A},\varphi_i} = L^{s_0+1}_{\mathcal{A},\varphi_i} =: L^\infty_{\mathcal{A},\varphi_i}$ for all $i \leqslant n$. We define $\big(L^\infty_{\mathcal{A},\varphi_1}, ..., L^\infty_{\mathcal{A},\varphi_n}\big)$ to be the *simultaneous least fixed point* of $(\varphi_1, ..., \varphi_n)$ in $\mathcal{A}$. It is straightforward to see that the simultaneous least fixed point is included in every simultaneous fixed point $(A_1, ..., A_n)$ of $(\varphi_1, ..., \varphi_n)$ in $\mathcal{A}$, *i.e.*, $L^\infty_{\mathcal{A},\varphi_i} \subseteq A_i$, for all $i \leqslant n$ (*cf., e.g.*, [7], Lem. 8.1.17).

The logic Sim-MLFP(τ) is the extension of MLFP(τ) by simultaneous least fixed point operators. *I.e.*: Sim-MLFP(τ) contains MLFP(τ) and is closed under Boolean connectives and first-order quantifications; and if $n \in \mathbb{N}$ and $\varphi_1(x_1, X_1, ..., X_n, \overline{y}, \overline{Y}), ..., \varphi_n(x_n, X_1, ..., X_n, \overline{y}, \overline{Y})$ are Sim-MLFP(τ)-formulas each of which is

positive in the variables $X_1, ..., X_n$ then, for every $i \leqslant n$,

$$[\text{Sim-LFP}_{x_1,X_1,...,x_n,X_n}\varphi_1, ..., \varphi_n]_{X_i}(x)$$

is a Sim-MLFP(τ)-formula such that for every $\big(\tau \cup \{\overline{y}, \overline{Y}\}\big)$-structure $\mathcal{A}$ and every element $a \in \mathcal{U}^{\mathcal{A}}$ we have $\mathcal{A} \models [\text{Sim-LFP}_{x_1,X_1,...,x_n,X_n}\varphi_1, ..., \varphi_n]_{X_i}(a)$ iff $a \in L^{\infty}_{\mathcal{A},\varphi_i}$.

Remark 4.1. It is straightforward to see that Sim-MLFP is $\mathcal{O}(m)$-succinct in MSO.

Every simultaneous least fixed point can be expressed by an MLFP-formula of exponential size:

Proposition 4.2. *Let τ be a signature, let $n, s \in \mathbb{N}$, and let, for each $i \leqslant n$, $\varphi_i(x_i, X_1, ..., X_n)$ be an* MLFP(τ)*-formula of size $||\varphi_i|| \leqslant s$ and positive in $X_1, ..., X_n$. For every $i \leqslant n$ there is a* MLFP(τ)*-formula $\Phi_{X_i}(x)$ of size $||\Phi_{X_i}|| \leqslant s^n$ such that, for all τ-structures $\mathcal{A}$, $\mathcal{A} \models \forall x\ \Phi_{X_i}(x) \leftrightarrow [\text{Sim-LFP}_{x_1,X_1,...,x_n,X_n}\varphi_1, ..., \varphi_n]_{X_i}(x)$.*

Furthermore, if $\varphi_1, ..., \varphi_n$ are in MLFP^2*, then also Φ_{X_i} is, for every $i \leqslant n$.*

Proof. The proof is by induction on n. The base case n=1 is trivial. For $n > 1$ and a fixed $i \leqslant n$, the induction hypothesis gives us for every $j \in \{1, ..., n\} \setminus \{i\}$ a MLFP($\tau \cup \{X_i\}$)-formula $\tilde{\Phi}_{X_j}(x)$ of size $\leqslant s^{n-1}$ such that, for all $\tau \cup \{X_i\}$-structures $\mathcal{B}$, $\mathcal{B} \models \forall x\ \tilde{\Phi}_{X_j}(x) \leftrightarrow$

$$[\text{Sim-LFP}_{x_1,X_1,...,x_{i-1},X_{i-1},x_{i+1},X_{i+1},...,x_n,X_n}\varphi_1, ..., \varphi_{i-1}, \varphi_{i+1}, ...\varphi_n]_{X_j}(x).$$

Define $\tilde{\varphi}_i(x_i, X_i)$ to be the MLFP-formula obtained from $\varphi_i(x_i, X_1, ..., X_n)$ by replacing every atom of the form $X_j(x)$, for $j \neq i$, with the formula $\tilde{\Phi}_{X_j}(x)$. It should be clear that $||\tilde{\varphi}_i|| < s^n$. To complete the proof of Proposition 4.2 it therefore suffices to show the following

Claim 4.3. *For all τ-structures $\mathcal{A}$ we have*
$$\mathcal{A} \models \forall x\ [\text{LFP}_{x_i,X_i}\tilde{\varphi}_i](x) \leftrightarrow [\text{Sim-LFP}_{x_1,X_1,...,x_n,X_n}\varphi_1, ..., \varphi_n]_{X_i}(x).$$

For proving this claim let, for every $S \subseteq \mathcal{U}^{\mathcal{A}}$ and every $j \neq i$,

$$\begin{aligned}
A_j(S) &:= \{a \in \mathcal{U}^{\mathcal{A}} : \langle \mathcal{A}, S\rangle \models \tilde{\Phi}_{X_j}(a)\},\\
A_i^{(\ell)} &:= \{a \in \mathcal{U}^{\mathcal{A}} : \mathcal{A} \models [\text{LFP}_{x_i,X_i}\tilde{\varphi}_i](a)\},\\
A_i^{(r)} &:= \{a \in \mathcal{U}^{\mathcal{A}} : \mathcal{A} \models [\text{Sim-LFP}_{x_1,X_1,...,x_n,X_n}\varphi_1, ..., \varphi_n]_{X_i}(a)\},\\
A_j^{(r)} &:= \{a \in \mathcal{U}^{\mathcal{A}} : \mathcal{A} \models [\text{Sim-LFP}_{x_1,X_1,...,x_n,X_n}\varphi_1, ..., \varphi_n]_{X_j}(a)\}.
\end{aligned}$$

Our aim is to show that $A_i^{(\ell)} = A_i^{(r)}$.

Since $A_i^{(\ell)}$ is a fixed point of $\tilde{\varphi}_i$, we have $A_i^{(\ell)} =$

$$\Big\{ a \in \mathcal{U}^{\mathcal{A}} : \mathcal{A} \models \varphi_i\Big(a, A_1(A_i^{(\ell)}), ..., A_{i-1}(A_i^{(\ell)}), A_i^{(\ell)}, A_{i+1}(A_i^{(\ell)}), ..., A_n(A_i^{(\ell)})\Big) \Big\}.$$

From the definition of $\tilde{\Phi}_{X_j}$ we furthermore know that $A_j(A_i^{(\ell)}) =$

$$\Big\{ a \in \mathcal{U}^{\mathcal{A}} \,:\, \mathcal{A} \models \varphi_j\Big(a, A_1(A_i^{(\ell)}), ..., A_{i-1}(A_i^{(\ell)}), A_i^{(\ell)}, A_{i+1}(A_i^{(\ell)}), ..., A_n(A_i^{(\ell)})\Big) \Big\}.$$

I.e., $\big(A_1(A_i^{(\ell)}), ..., A_{i-1}(A_i^{(\ell)}), A_i^{(\ell)}, A_{i+1}(A_i^{(\ell)}), ..., A_n(A_i^{(\ell)})\big)$ is a simultaneous fixed point of the formulas $\big(\varphi_1, ..., \varphi_n\big)$ in $\mathcal{A}$, the *least* fixed point of which is $\big(A_1^{(r)}, ..., A_n^{(r)}\big)$. In particular, this implies that $A_i^{(\ell)} \supseteq A_i^{(r)}$.

On the other hand, $\big(A_1^{(r)}, ..., A_{i-1}^{(r)}, A_{i+1}^{(r)}, ..., A_n^{(r)}\big)$ is a simultaneous fixed point of $\big(\varphi_1, ..., \varphi_{i-1}, \varphi_{i+1}, ..., \varphi_n\big)$ in $\langle \mathcal{A}, A_i^{(r)} \rangle$, the *least* fixed point of which is $\big(A_1(A_i^{(r)}), ..., A_{i-1}(A_i^{(r)}), A_{i+1}(A_i^{(r)}), ..., A_n(A_i^{(r)})\big)$.

Therefore, $A_j(A_i^{(r)}) \subseteq A_j^{(r)}$, for all $j \neq i$, and hence

$$\begin{aligned}
&\Big\{ a \in \mathcal{U}^{\mathcal{A}} : \langle \mathcal{A}, A_i^{(r)} \rangle \models \tilde{\varphi}_i(a) \Big\} = \\
&\Big\{ a \in \mathcal{U}^{\mathcal{A}} : \mathcal{A} \models \varphi_i\Big(a, A_1(A_i^{(r)}), ..., A_{i-1}(A_i^{(r)}), A_i^{(r)}, A_{i+1}(A_i^{(r)}), ..., A_n(A_i^{(r)})\Big) \Big\} \subseteq \\
&\Big\{ a \in \mathcal{U}^{\mathcal{A}} \,:\, \mathcal{A} \models \varphi_i\Big(a, A_1^{(r)}, ..., A_{i-1}^{(r)}, A_i^{(r)}, A_{i+1}^{(r)}, ..., A_n^{(r)}\Big) \Big\} = A_i^{(r)}.
\end{aligned}$$

I.e., $A_i^{(r)}$ is a *pre fixed point*[3] of $\tilde{\varphi}_i$ on $\mathcal{A}$, and therefore $A_i^{(\ell)} \subseteq A_i^{(r)}$.
This completes the proof of the above claim and of Proposition 4.2. □

Using Proposition 4.2 one easily obtains the following result by induction on the construction of Sim-MLFP-formulas:

Corollary 4.4. Sim-MLFP *is* $2^{2^{\mathcal{O}(m)}}$*-succinct in* MLFP *on the class of all finite structures.*

Together with Theorem 2.2 this leads to

Corollary 4.5. *Unless* SAT *is solvable by a deterministic algorithm that has, for every* $i \in \mathbb{N}$*, time bound* $||\gamma||^{\lg^{(i)}(n)}$ *(where* γ *is the input formula and* n *the number of propositional variables occurring in* γ*),* MSO *is not Tower*$\big(o(m)\big)$*-succinct in* Sim-MLFP *on the class of all finite strings.*

5. Monadic datalog and stratified monadic datalog

We assume that the reader is familiar with *datalog*, which may be viewed as logic programming without function symbols (*cf., e.g.*, the textbook [2]). A datalog program is *monadic* if all its IDB-predicates (*i.e.*, its intensional predicates that appear in the head of some rule of the program) are unary. In this paper we restrict attention to monadic datalog programs that are interpreted over labelled trees. A monadic datalog program of schema σ may use as EDB-predicates (*i.e.*, extensional predicates which are determined by the structure the program is interpreted over)

[3] Recall that a *pre fixed point* of a monotone operator $F : 2^M \to 2^M$ is a set $A \subseteq M$ with $F(A) \subseteq A$, and that the *least fixed point* of F is the intersection of all pre fixed points of F.

the predicates in τ_{Trees}, the predicates in σ, and a predicate $\neg P$ for every $P \in \sigma$ which is interpreted as the complement of P. We use IDB($\mathcal{P}$) to denote the set of IDB-predicates of $\mathcal{P}$, and we write MonDatalog to denote the class of all monadic datalog programs.

More formally, a monadic datalog program $\mathcal{P}$ of schema σ is a finite set of rules of the form $X(x) \leftarrow \gamma(x, \overline{y})$, where γ is a conjunction of atomic formulas over the signature $\tau_{Trees} \cup \sigma \cup \{\neg P : P \in \sigma\} \cup \text{IDB}(\mathcal{P})$. We define the *size* $||\mathcal{P}||$ of $\mathcal{P}$ in the same way as we defined the size of formulas.

Instead of formally defining the semantics of MonDatalog, we just give a translation of MonDatalog into Sim-MLFP on the class of all labelled trees:

Given a MonDatalog-program $\mathcal{P}$ of schema σ with $\text{IDB}(\mathcal{P}) = \{X_1, ..., X_n\}$, we can assume w.l.o.g. that there is a unique first-order variable x such that every rule in $\mathcal{P}$ is of the form $X_i(x) \leftarrow \gamma(x, \overline{y})$, for some $i \leqslant n$. Let $\gamma_{i,1}(x, \overline{y}), ..., \gamma_{i,\ell_i}(x, \overline{y})$ be a list of all bodies of rules with head $X_i(x)$ in $\mathcal{P}$. Of course, the FO-formula $\varphi_i(x, X_1, ..., X_n) := \bigvee_{j=1}^{\ell_i} \exists \overline{y}\, \gamma_{i,j}(x, \overline{y})$ is positive in all the variables $X_1, ..., X_n$. When evaluated in a σ-labelled tree T, the program $\mathcal{P}$ defines the unary relations $(X_i)^\infty_\mathcal{P}(T)$, for $i \leqslant n$, to be the simultaneous least fixed point of $(\varphi_1, ..., \varphi_n)$ in T.

A MonDatalog-program $\mathcal{P}$ of schema σ, together with a designated *goal* predicate $X \in \text{IDB}(\mathcal{P})$, defines the *unary query* which yields, for every σ-labelled tree T, the set $X^\infty_\mathcal{P}(T)$ of vertices of T. We say that "T belongs to the *Boolean query* defined by $(\mathcal{P}, X)$" iff the root of T belongs to $X^\infty_\mathcal{P}(T)$.

Let us sum up what we have seen above:

Lemma 5.1. *For every* MonDatalog-*program* $\mathcal{P}$ *with IDB*$(\mathcal{P}) = \{X_1, ..., X_n\}$ *there are* FO-*formulas* $\varphi_i(x, X_1, ..., X_n)$, *for every* $i \in \{1, ..., n\}$, *positive in* $X_1, ..., X_n$, *such that* $||\varphi_1|| + \cdots + ||\varphi_n|| \leqslant ||\mathcal{P}||$ *and* [Sim-LFP$_{x,X_1,...,x,X_n}\varphi_1, ...,$ $\varphi_n]_{X_i}(x)$ *defines the same unary query as* $(\mathcal{P}, X_i)$ *on the class of all labelled trees.*

Gottlob and Koch [13] proposed the following useful normal form for monadic datalog:

Definition 5.2 (TMNF)**.** A MonDatalog-program $\mathcal{P}$ of schema σ is in *tree marking normal form* (TMNF, for short) iff each rule is of one of the four forms

1. $X(x) \leftarrow S(x)$,
2. $X(x) \leftarrow X'(x) \wedge X''(x)$,
3. $X(x) \leftarrow R(x, y) \wedge X'(y)$,
4. $X(x) \leftarrow R(y, x) \wedge X'(y)$, where

$X, X', X'' \in \text{IDB}(\mathcal{P})$, $R \in \{1\text{st}\mathit{Child}, 2\text{nd}\mathit{Child}\}$, and
$S \in \sigma \cup \{\neg P : P \in \sigma\} \cup \{\mathit{Root}, \mathit{Has\text{-}No}\text{-}1\text{st}\mathit{Child}, \mathit{Has\text{-}No}\text{-}2\text{nd}\mathit{Child}\}$.

Theorem 5.3 [13] (Th. 4.11)**.** *For every* MonDatalog-*program* $\mathcal{P}$ *and every* $X \in$ *IDB*$(\mathcal{P})$ *there is a TMNF-program* $\mathcal{P}_X$ *of size* $\mathcal{O}(||\mathcal{P}||)$ *with* $X \in$ *IDB*$(\mathcal{P}_X)$ *such that* $(\mathcal{P}_X, X)$ *defines the same unary query as* $(\mathcal{P}, X)$ *on the class of labelled trees.*

One way of adding negation to datalog is to consider stratified datalog (*cf.*, [2]): a *stratified monadic datalog* program $\mathcal{P}$ of schema σ is a finite set of rules of the form $X(x) \leftarrow \gamma(x, \overline{y})$, where γ is a conjunction of atomic formulas over

the signature $\tau_{Trees} \cup \sigma \cup \{\neg P : P \in \sigma\} \cup \mathrm{IDB}(\mathcal{P}) \cup \{\neg X : X \in \mathrm{IDB}(\mathcal{P})\}$ that has the following property: There is a partition of $\mathcal{P}$ into sets $\mathcal{P}_1, ..., \mathcal{P}_n$, for some $n \in \mathbb{N}$, such that each $\mathcal{P}_i$ is a MonDatalog-program of schema $\sigma \cup \bigcup_{j<i} \mathrm{IDB}(\mathcal{P}_j)$. The programs $\mathcal{P}_1, ..., \mathcal{P}_n$ are called the *strata* of $\mathcal{P}$, and $(\mathcal{P}_1, ..., \mathcal{P}_n)$ is called a decomposition of $\mathcal{P}$ into strata.

We write S-MonDatalog to denote the class of all stratified monadic datalog programs.

In [13] it was shown that MonDatalog can define the same unary queries on the class of labelled trees as monadic second-order logic. In the remainder of this section we will compare the succinctness of MonDatalog, S-MonDatalog, FL_μ, MLFP, and a particular kind of tree automaton.

5.1. From MonDatalog to finite automata

Several mechanisms have been proposed in the literature for specifying *unary* queries by finite automata operating on labelled trees (*cf.*, [23]). One such mechanism, introduced in [22] and further investigated in [12, 21], is the *selecting tree automaton*:

Definition 5.4 (STA)**.** Let σ be a schema. A selecting σ-tree automaton (σ-STA, for short) is a tuple $\mathfrak{A} = (Q, 2^\sigma, F, \delta, S)$, where $S \subseteq Q$ is the set of *selecting states* and $(Q, 2^\sigma, F, \delta)$ is a conventional nondeterministic bottom-up tree automaton (*cf., e.g.*, [24]) with finite state space Q, input alphabet 2^σ, accepting states $F \subseteq Q$, and transition function

$$\delta \;:\; 2^\sigma \;\cup\; \big(\{1\} \times Q \times 2^\sigma\big) \;\cup\; \big(\{2\} \times Q \times 2^\sigma\big) \;\cup\; \big(Q \times Q \times 2^\sigma\big) \;\to\; 2^Q\,.$$

A *run* of $\mathfrak{A}$ on a σ-labelled tree T is a mapping $\rho : \mathcal{U}^T \to Q$ that has the following properties, for all vertices $t, t_1, t_2 \in \mathcal{U}^T$: if t has no children then $\rho(t) \in \delta(\mathit{label}(t))$; if $1\mathrm{st}\mathit{Child}(t, t_1) \wedge \mathit{Has\text{-}No}\text{-}2\mathrm{nd}\mathit{Child}(t)$ then $\rho(t) \in \delta\big(1, \rho(t_1), \mathit{label}(t)\big)$; if $2\mathrm{nd}\mathit{Child}(t, t_2) \wedge \mathit{Has\text{-}No}\text{-}1\mathrm{st}\mathit{Child}(t)$ then $\rho(t) \in \delta\big(2, \rho(t_2), \mathit{label}(t)\big)$; if $1\mathrm{st}\mathit{Child}(t, t_1) \wedge 2\mathrm{nd}\mathit{Child}(t, t_2)$ then $\rho(t) \in \delta\big(\rho(t_1), \rho(t_2), \mathit{label}(t)\big)$.

A run ρ of $\mathfrak{A}$ on T is said to be *accepting* if it maps the root of T to a state in F. The *unary query* defined by $\mathfrak{A}$ is the query which maps every σ-labelled tree T to the set of those vertices $t \in \mathcal{U}^T$ that satisfy the following condition: $\rho(t) \in S$ for *every accepting run* ρ of $\mathfrak{A}$ on T.

It was shown in [12, 22] that STAs can define exactly those unary queries on the class of labelled trees that are definable in monadic-second order logic.

Theorem 5.5 [12, 13]**.** MonDatalog *is* $2^{\mathcal{O}(m)}$*-succinct in STAs on the class of labelled trees.* More precisely: let $\mathcal{P}$ be a MonDatalog-program of schema σ, and let $X \in \mathrm{IDB}(\mathcal{P})$. There is a σ-STA $\mathfrak{A}$ with $2^{\mathcal{O}(||\mathcal{P}||)}$ states that defines the same unary query as $(\mathcal{P}, X)$ on the class of labelled trees. Given $\mathcal{P}$, the STA $\mathfrak{A}$ can be computed in time $2^{\mathcal{O}(||\mathcal{P}||)}$.

Proof. From Theorem 5.3 we know that $\mathcal{P}$ can be translated in time $\mathcal{O}(||\mathcal{P}||)$ into a TMNF-program $\mathcal{P}'$ of size $\mathcal{O}(||\mathcal{P}||)$ such that $(\mathcal{P}', X)$ defines the same unary query as $(\mathcal{P}, X)$ on the class of labelled trees. Let ℓ be the number of IDB-predicates of $\mathcal{P}'$. In [12, Ex. 20], a σ-STA with 2^ℓ states was constructed, defining the same unary query as $(\mathcal{P}', X)$ on the class of labelled trees. □

The next example shows that, asymptotically, the above construction is optimal:

Example 5.6. *For every $k \in \mathbb{N}$ there is a* MonDatalog*-program of size $\mathcal{O}(k)$ that defines a query not definable by an STA with less than 2^k states:*

Restricting attention to conventional nondeterministic finite automata (NFAs) that operate on finite strings, it is straightforward to see that there is no NFA $\mathfrak{A}$ with less than 2^k states such that, for all $w_1, w_2 \in \{0,1\}^k$, $\mathfrak{A}$ accepts the string $w_1 2 w_2$ if and only if $w_1 = w_2$. On the other hand, the following MonDatalog-program $\mathcal{P}_k$ has size $\mathcal{O}(k)$ and has the property that, for all $w_1, w_2 \in \{0,1\}^k$, $(X^{ok})^\infty_{\mathcal{P}_k}(w_1 2 w_2)$ contains the maximum position of the string $w_1 2 w_2$ if and only if $w_1 = w_2$. The program $\mathcal{P}_k$ consists of the following rules:

$$
\begin{aligned}
X_0^{transfer}(x) &\leftarrow P_0(y_1) \wedge \mathit{Succ}(y_1, y_2) \wedge \cdots \wedge \mathit{Succ}(y_{k-1}, y_k) \wedge \mathit{Succ}(y_k, x)\\
X_1^{transfer}(x) &\leftarrow P_1(y_1) \wedge \mathit{Succ}(y_1, y_2) \wedge \cdots \wedge \mathit{Succ}(y_{k-1}, y_k) \wedge \mathit{Succ}(y_k, x)\\
X^{compare}(x) &\leftarrow P_0(x) \wedge X_0^{transfer}(x)\\
X^{compare}(x) &\leftarrow P_1(x) \wedge X_1^{transfer}(x)\\
X^{ok}(x) &\leftarrow P_2(x)\\
X^{ok}(x) &\leftarrow X^{ok}(y) \wedge \mathit{Succ}(y, x) \wedge X^{compare}(x)\,.
\end{aligned}
$$

5.2. From S-MonDatalog to MonDatalog

In this section we show that S-MonDatalog-programs can be translated into MonDatalog-programs of at most exponential size. It remains open if the exponential size is indeed necessary or if, on the contrary, for every S-MonDatalog-program $\mathcal{P}$ there exists an equivalent MonDatalog-program $\mathcal{P}'$ of size polynomial in $||\mathcal{P}||$.

Lemma 5.7. *For every σ-STA $\mathfrak{A} = (Q, 2^\sigma, F, \delta, S)$ there is a* MonDatalog*-program $\mathcal{P}$ of size $\mathcal{O}\big(|Q|^3 \cdot |2^\sigma| + |\sigma| \cdot |2^\sigma|\big)$ and a designated goal predicate $\overline{X} \in \mathit{IDB}(\mathcal{P})$ such that $(\mathcal{P}, \overline{X})$ defines the complement of the query defined by $\mathfrak{A}$ on the class of all σ-labelled trees.*

Proof. The program $\mathcal{P}$ operates according to the following evaluation algorithm of [12] (Prop. 21): in a bottom-up pass of the input tree T, the set $\mathit{Reach}(t)$ is computed for every vertex t of T, where $\mathit{Reach}(t)$ consists of all states q such that there is a partial run ρ of $\mathfrak{A}$ on T with $\rho(t) = q$. Afterwards, in a top-down pass the set $\mathit{Acc}(t)$ is computed for every $t \in \mathcal{U}^T$, where $\mathit{Acc}(t)$ consists of all states q such that there is an *accepting* run ρ of $\mathfrak{A}$ on T with $\rho(t) = q$. Clearly, t does *not* belong to the query defined by $\mathfrak{A}$ if and only if $\mathit{Acc}(t)$ contains a state in $Q \setminus S$.

To store the sets $\mathit{Reach}(\cdot)$ and $\mathit{Acc}(\cdot)$, respectively, the program $\mathcal{P}$ will use predicates X_q^{Reach} and X_q^{Acc}, respectively, for every $q \in Q$. The intended meaning

of these predicates is, that for all σ-labelled trees T, $t \in \mathcal{U}^T$, and $q \in Q$ we shall have $t \in (X_q^{Reach})^\infty_{\mathcal{P}}(T)$ iff $q \in \mathit{Reach}(t)$, and $t \in (X_q^{Acc})^\infty_{\mathcal{P}}(T)$ iff $q \in \mathit{Acc}(t)$.

$\mathcal{P}$ has one further predicate, X^{no} for storing the vertices that do *not* belong to the query defined by $\mathfrak{A}$. Now, the definition of $\mathcal{P}$ is straightforward:

(1) For every $a \in 2^\sigma$, $\mathcal{P}$ contains the rule $\mathit{Input}_a(x) \leftarrow \bigwedge_{P \in a} P(x) \wedge \bigwedge_{P \in \sigma \setminus a} \neg P(x)$.

(2) For every $q \in Q \setminus S$, $\mathcal{P}$ contains the rule $X^{no}(x) \leftarrow X_q^{Acc}(x)$.

(3) For all $q \in F$, $\mathcal{P}$ contains the rule $X_q^{Acc}(x) \leftarrow X_q^{Reach}(x) \wedge \mathit{Root}(x)$.

(4) For all $a \in 2^\sigma$ and $p \in \delta(a)$, $\mathcal{P}$ contains the rule $X_p^{Reach}(x) \leftarrow \mathit{Input}_a(x) \wedge \mathit{Has\text{-}No\text{-}}1\mathrm{st}\mathit{Child}(x) \wedge \mathit{Has\text{-}No\text{-}}2\mathrm{nd}\mathit{Child}(x)$.

(5) For all $q \in Q$, $a \in 2^\sigma$, and $p \in \delta(1, q, a)$, $\mathcal{P}$ contains the rules $X_p^{Reach}(x) \leftarrow \mathit{Input}_a(x) \wedge 1\mathrm{st}\mathit{Child}(x, y) \wedge \mathit{Has\text{-}No\text{-}}2\mathrm{nd}\mathit{Child}(x) \wedge X_q^{Reach}(y)$, and $X_q^{Acc}(y) \leftarrow X_q^{Reach}(y) \wedge 1\mathrm{st}\mathit{Child}(x, y) \wedge \mathit{Input}_a(x) \wedge \mathit{Has\text{-}No\text{-}}2\mathrm{nd}\mathit{Child}(x) \wedge X_p^{Acc}(x)$.

(6) For all $q \in Q$, $a \in 2^\sigma$, and $p \in \delta(2, q, a)$, $\mathcal{P}$ contains the rules $X_p^{Reach}(x) \leftarrow \mathit{Input}_a(x) \wedge \mathit{Has\text{-}No\text{-}}1\mathrm{st}\mathit{Child}(x) \wedge 2\mathrm{nd}\mathit{Child}(x, y) \wedge X_q^{Reach}(y)$, and $X_q^{Acc}(y) \leftarrow X_q^{Reach}(y) \wedge 2\mathrm{nd}\mathit{Child}(x, y) \wedge \mathit{Input}_a(x) \wedge \mathit{Has\text{-}No\text{-}}1\mathrm{st}\mathit{Child}(x) \wedge X_p^{Acc}(x)$.

(7) For all $q_1, q_2 \in Q$, $a \in 2^\sigma$ and $p \in \delta(q_1, q_2, a)$, $\mathcal{P}$ contains the rules $X_p^{Reach}(x) \leftarrow \mathit{Input}_a(x) \wedge 1\mathrm{st}\mathit{Child}(x, y_1) \wedge X_{q_1}^{Reach}(y_1) \wedge 2\mathrm{nd}\mathit{Child}(x, y_2) \wedge X_{q_2}^{Reach}(y_2)$, $X_{q_1}^{Acc}(y_1) \leftarrow X_{q_1}^{Reach}(y_1) \wedge 1\mathrm{st}\mathit{Child}(x, y_1) \wedge 2\mathrm{nd}\mathit{Child}(x, y_2) \wedge \mathit{Input}_a(x) \wedge X_p^{Acc}(x) \wedge X_{q_2}^{Reach}(y_2)$,
$X_{q_2}^{Acc}(y_2) \leftarrow X_{q_2}^{Reach}(y_2) \wedge 1\mathrm{st}\mathit{Child}(x, y_1) \wedge 2\mathrm{nd}\mathit{Child}(x, y_2) \wedge \mathit{Input}_a(x) \wedge X_p^{Acc}(x) \wedge X_{q_1}^{Reach}(y_1)$.

One can easily verify that $(\mathcal{P}, X^{no})$ defines the complement of the query defined by $\mathfrak{A}$ on the class of σ-labelled trees. Furthermore, $||\mathcal{P}|| = \mathcal{O}\big(|Q|^3 \cdot |2^\sigma| + |\sigma| \cdot |2^\sigma|\big)$. □

Using Theorem 5.5 and Lemma 5.7 one easily obtains

Proposition 5.8. *For every* MonDatalog*-program $\mathcal{P}$ there is a* MonDatalog*-program $\mathcal{P}'$ of size $2^{\mathcal{O}(||\mathcal{P}||)}$ with $\mathit{IDB}(\mathcal{P}') \supseteq \{X, \overline{X} : X \in \mathit{IDB}(\mathcal{P})\}$ such that, for every $X \in \mathit{IDB}(\mathcal{P})$, $(\mathcal{P}', X)$ defines the same unary query as $(\mathcal{P}, X)$, and $(\mathcal{P}', \overline{X})$ defines the complement of the unary query defined by $(\mathcal{P}, X)$ on the class of labelled trees.*

Proof. Let σ be the schema of $\mathcal{P}$, and let $X \in \mathrm{IDB}(\mathcal{P})$. From Theorem 5.5 we obtain a σ-STA $\mathfrak{A}_X$ with $|Q| = 2^{\mathcal{O}(||\mathcal{P}||)}$ states defining the same query as $(\mathcal{P}, X)$. Lemma 5.7 gives us a MonDatalog-program $\overline{\mathcal{P}_X}$ and a predicate $\overline{X} \in \mathrm{IDB}(\overline{\mathcal{P}_X})$ such that $(\overline{\mathcal{P}_X}, \overline{X})$ defines the complement of the query defined by $\mathfrak{A}_X$; and $||\overline{\mathcal{P}_X}|| = \mathcal{O}\big(|Q|^3 \cdot |2^\sigma| + |\sigma| \cdot |2^\sigma|\big) = 2^{\mathcal{O}(||\mathcal{P}||)}$.

After an appropriate renaming of IDB-predicates we can assume w.l.o.g. that $\mathrm{IDB}(\overline{\mathcal{P}_X}) \cap \mathrm{IDB}(\mathcal{P}) = \emptyset$ and $\mathrm{IDB}(\overline{\mathcal{P}_X}) \cap \mathrm{IDB}(\overline{\mathcal{P}_Y}) = \emptyset$, for all distinct $X, Y \in \mathrm{IDB}(\mathcal{P})$.

Obviously, $\mathcal{P}' := \mathcal{P} \cup \bigcup_{X \in \mathrm{IDB}(\mathcal{P})} \overline{\mathcal{P}_X}$ is the desired MonDatalog-program of size $||\mathcal{P}'|| \leqslant ||\mathcal{P}|| + |\mathrm{IDB}(\mathcal{P})| \cdot 2^{\mathcal{O}(||\mathcal{P}||)} = 2^{\mathcal{O}(||\mathcal{P}||)}$. □

Using the above proposition, it is not difficult to prove

Theorem 5.9. S-MonDatalog *is* $2^{\mathcal{O}(m)}$*-succinct in* MonDatalog *on the class of labelled trees.* More precisely: For every S-MonDatalog-program $\mathcal{P}$ there is a MonDatalog-program $\mathcal{P}'$ of size $2^{\mathcal{O}(||\mathcal{P}||)}$ such that $\mathrm{IDB}(\mathcal{P}') \supseteq \mathrm{IDB}(\mathcal{P})$ and, for every $X \in \mathrm{IDB}(\mathcal{P})$, $(\mathcal{P}', X)$ defines the same unary query as $(\mathcal{P}, X)$ on the class of all labelled trees.

Given $\mathcal{P}$, the program $\mathcal{P}'$ can be computed in time $2^{\mathcal{O}(||\mathcal{P}||)}$.

Proof. Let $\mathcal{P}$ be a S-MonDatalog-program of schema σ. Let $(\mathcal{P}_1, ..., \mathcal{P}_n)$ be a decomposition of $\mathcal{P}$ into strata. *I.e.*, $\mathcal{P} = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_n$, and each $\mathcal{P}_i$ can be viewed as a MonDatalog-program of schema $\sigma_i := \sigma \cup \bigcup_{j<i} \mathrm{IDB}(\mathcal{P}_j)$. For each i, let $\tilde{\mathcal{P}}_i$ be the MonDatalog-program of schema $\tilde{\sigma}_i := \sigma \cup \{X, \overline{X} : X \in \bigcup_{j<i} \mathrm{IDB}(\mathcal{P}_j)\}$, obtained from $\mathcal{P}_i$ by replacing every occurrence of a literal of the form $\neg X(y)$ by the literal $\overline{X}(y)$. Let $\tilde{\mathcal{P}}'_i$ be the MonDatalog-program of size $2^{\mathcal{O}(||\mathcal{P}_i||)}$ that Proposition 5.8 provides for $\tilde{\mathcal{P}}_i$. After an appropriate renaming we can assume w.l.o.g. that, viewed as programs of schema $\tilde{\sigma}_i$ and $\tilde{\sigma}_j$, respectively, $\tilde{\mathcal{P}}'_i$ and $\tilde{\mathcal{P}}'_j$ have no IDB-predicates in common (for distinct $i, j \leqslant n$).

It is straightforward to check that the program $\mathcal{P}' := \bigcup_{i=1}^{n} \tilde{\mathcal{P}}'_i$, viewed as a MonDatalog-program of schema σ, has the desired property that $\mathrm{IDB}(\mathcal{P}') \supseteq \mathrm{IDB}(\mathcal{P})$ and $(\mathcal{P}', X)$ defines the same query as $(\mathcal{P}, X)$, for every $X \in \mathrm{IDB}(\mathcal{P})$.

Furthermore, $||\mathcal{P}'|| \leqslant \sum_{i=1}^{n} 2^{\mathcal{O}(||\mathcal{P}_i||)} \leqslant 2^{\mathcal{O}(||\mathcal{P}||)}$.

It is straightforward to see that $\mathcal{P}'$ can be computed in time $2^{\mathcal{O}(||\mathcal{P}||)}$. □

5.3. S-MonDatalog *vs.* FL_μ

From Theorem 3.4 and Lemma 5.7 one directly obtains

Theorem 5.10. FL_μ *is* $2^{poly(m)}$*-succinct in* S-MonDatalog *on the class of labelled trees.* More precisely: For every FL_μ-formula Φ there is an S-MonDatalog-program $\mathcal{P}$ of size $2^{poly(||\Phi||)}$ and a predicate $X \in \mathrm{IDB}(\mathcal{P})$*, such that, for all labelled trees* T*, the root of* T *belongs to the unary query defined by* $(\mathcal{P}, X)$ *if, and only if, the root of* T *satisfies* Φ*.*

Conversely, using Theorem 5.3, Lemma 5.1, and Proposition 4.2 one can show the following

Theorem 5.11. S-MonDatalog *is* $2^{\mathcal{O}(m \cdot \lg m)}$*-succinct in* FL_μ *on the class of labelled trees.* More precisely: For every S-MonDatalog-program $\mathcal{P}$ and every $X \in \mathrm{IDB}(\mathcal{P})$, there is a FL_μ-formula Φ_X of size $2^{\mathcal{O}(||\mathcal{P}|| \cdot \lg ||\mathcal{P}||)}$ that defines the same unary query as $(\mathcal{P}, X)$ on the class of labelled trees.

Proof. The proof proceeds in 2 steps.

Step 1: Let us first consider the special case where $\mathcal{P}$ is a MonDatalog-program. For every $X \in \mathrm{IDB}(\mathcal{P})$ let $\mathcal{P}_X$ be a TMNF-program of size $\mathcal{O}(||\mathcal{P}||)$ with $X \in \mathrm{IDB}(\mathcal{P}_X)$ such that $(\mathcal{P}_X, X)$ defines the same unary query as $(\mathcal{P}, X)$ on the class of labelled trees (*cf.*, Th. 5.3). Let n be the number of IDB-predicates of $\mathcal{P}_X$. Lemma 5.1 and Proposition 4.2 give us a MLFP^2-formula $\Psi_X(x)$ of size $||\Psi_X|| \leqslant ||\mathcal{P}_X||^n \leqslant ||\mathcal{P}_X||^{||\mathcal{P}_X||} \leqslant 2^{\mathcal{O}(||\mathcal{P}||\cdot\lg||\mathcal{P}||)}$ which defines the same unary query as $(\mathcal{P}, X)$ on the class of labelled trees. Moreover, a close look at the proofs of Lemma 5.1 and Proposition 4.2 for the special case where $\mathcal{P}_X$ is in TMNF shows that $\Psi_X(x)$ is equivalent to a FL$_\mu$-formula Φ_X of size $\mathcal{O}(||\Psi_X||)$.

Step 2: Now consider the case where $\mathcal{P}$ is an S-MonDatalog-program. Let σ be the schema of $\mathcal{P}$, and let $(\mathcal{P}_1, ..., \mathcal{P}_n)$ be a decomposition of $\mathcal{P}$ into strata. *I.e.,* $\mathcal{P} = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_n$, and each $\mathcal{P}_i$ can be viewed as a MonDatalog-program of schema $\sigma_i := \sigma \cup \bigcup_{j<i} \mathrm{IDB}(\mathcal{P}_j)$. Step 1 gives us, for every $i \leqslant n$ and every $X \in \mathrm{IDB}(\mathcal{P}_i)$, a FL$_\mu$-formula Φ_X of size $||\Phi_X|| = 2^{\mathcal{O}(||\mathcal{P}_i||\cdot\lg||\mathcal{P}_i||)}$ that defines the same unary query on the class of σ_i-labelled trees as $(\mathcal{P}_i, X)$.

By induction on i we define, for every $X \in \mathrm{IDB}(\mathcal{P}_i)$, the FL$_\mu$–formula $\hat{\Phi}_X$ of size

$$||\hat{\Phi}_X|| = 2^{\mathcal{O}(||\mathcal{P}_1||\cdot\lg||\mathcal{P}_1||+\cdots+||\mathcal{P}_i||\cdot\lg||\mathcal{P}_i||)}$$

as follows: for every $X \in \mathrm{IDB}(\mathcal{P}_i)$ let $\hat{\Phi}_X$ be the formula obtained from Φ_X by replacing every occurrence of an $Y \in \bigcup_{j<i} \mathrm{IDB}(\mathcal{P}_j)$ by the formula $\hat{\Phi}_Y$. It is straightforward to verify for every $X \in \mathrm{IDB}(\mathcal{P})$ that $\hat{\Phi}_X$ defines the same unary query as $(\mathcal{P}, X)$ on the class of σ-labelled trees. □

It remains open whether the above bounds are optimal.

5.4. From MLFP to S-MonDatalog

Similarly to Theorem 2.1 one easily obtains

Theorem 5.12 (Folklore)**.** MLFP*-sentences are Tower*$\big(\mathcal{O}(m)\big)$*-succinct in* S-MonDatalog *on the class of labelled trees.*

The aim of this section is to show that there are no essentially smaller translations from MLFP to S-MonDatalog. We will use the following well-known observation:

Proposition 5.13 (Folklore)**.** *There is no function $f : \mathbb{N} \to \mathbb{N}$ with bound $f(m) \leqslant$ Tower*$\big(o(m)\big)$ *such that for every* FO(<)*-sentence φ there is a nondeterministic finite automaton $\mathfrak{A}$ with at most $f(||\varphi||)$ states that accepts exactly those strings that satisfy φ.*

Proof. Let the alphabet Σ_h and the strings $\mu_h(n)$ be chosen in the same way as in Section 2. We use the following result of [11]:

Lemma 5.14 [11] (Lem. 8)**.** *For every $h \in \mathbb{N}$ there is a* FO(<)*-formula $\chi_{h,1}$ of size*[4] $||\chi_{h,1}|| = \mathcal{O}(h)$ *such that the following is true for all strings w over alphabet Σ_h, for all positions a, b in w, and for all numbers $m, n \in \{0, ..., Tower(h)\}$: if a is the first position of a substring u of w that is isomorphic to $\mu_h(m)$ and if b is the first position of a substring v of w that is isomorphic to $\mu_h(n)$, then $w \models \chi_{h,1}(a, b)$ if and only if $m = n$.*

For every $h \in \mathbb{N}$ let $H := Tower(h)$. For $a_0 \cdots a_{H-1} \in \{0,1\}^H$ define the string

$$w_{h,a_0\cdots a_{H-1}} := \texttt{<h+1>}\ \ \mu_h(0)\, a_0\ \ \mu_h(1)\, a_1 \cdots \mu_h(H{-}1)\, a_{H-1}\ \ \texttt{</h+1>}.$$

We define the string-language

$$L_h := \left\{\ w_{h,a_0\cdots a_{H-1}}\ :\ a_0 \cdots a_{H-1} \in \{0,1\}^H\ \right\}^*,$$

and we choose a designated string $w_h \in L_h$ *via*

$$w_h := w_{h,\mathrm{BIN}_H(0)}\, w_{h,\mathrm{BIN}_H(1)}\ \cdots\ w_{h,\mathrm{BIN}_H(2^H-1)}\,,$$

where $\mathrm{BIN}_H(n)$ denotes the reverse binary representation of length H of n. For example, $\mathrm{BIN}_4(2) = 0100$ and $\mathrm{BIN}_4(5) = 1010$. It is straightforward to see the following:

Lemma 5.15. *For $h \in \mathbb{N}$ let $H := Tower(h)$. There is no nondeterministic finite automaton $\mathfrak{A}_h$ with less than $2^H = Tower(h{+}1)$ states such that the following is true for every $w \in L_h$: $\mathfrak{A}_h$ accepts w if and only if $w = w_h$.*

Lemma 5.16. *For every $h \in \mathbb{N}$ there is a* FO(<)*-sentence φ_h of size $\mathcal{O}(h)$ such that the following is true for all $w \in L_h$: $w \models \varphi_h$ if and only if $w = w_h$.*

Proof. To simplify notation, we write $Succ(x)$ to denote the successor of a position x in a string w.

Using the formula $\chi_{h,1}$ of Lemma 5.14, it is straightforward to build a FO(<)-sentence φ_h stating for every input string $w \in L_h$ that

- the leftmost substring of w of the form $\texttt{<h+1>} \cdots \texttt{</h+1>}$ contains no position x with $P_{\texttt{</h>}}(x) \wedge P_1(Succ(x))$;
- the rightmost substring of w of the form $\texttt{<h+1>} \cdots \texttt{</h+1>}$ contains no position y with $P_{\texttt{</h>}}(y) \wedge P_0(Succ(x))$;
- for every two successive substrings u and v of w of the form $\texttt{<h+1>} \cdots \texttt{</h+1>}$, there is a position x_1 in u with $P_{\texttt{<h>}}(x_1)$ and a position y_1 in v with $P_{\texttt{<h>}}(y_1)$ such that $\chi_{h,1}(x_1, y_1)$ is true and
  - every position x_0 in u to the left of x_1 satisfies $P_{\texttt{</h>}}(x_0) \rightarrow P_1(Succ(x_0))$;

[4] In [11], an additional factor $\lg h$ occurs because there a logarithmic cost measure is used for the formula size, whereas here we use a uniform measure (*cf.*, Sect. 1.4).

- every position y_0 in v to the left of y_1 satisfies $P_{\texttt{</h>}}(y_0) \rightarrow P_0(Succ(y_0))$;
- the first position x_1' to the right of x_1 in u with $P_{\texttt{</h>}}(x_1')$ satisfies $P_0(Succ(x_1'))$;
- the first position y_1' to the right of y_1 in v with $P_{\texttt{</h>}}(y_1')$ satisfies $P_1(Succ(y_1'))$;
- for all positions x_2 to the right of x_1' in u and for all positions y_2 to the right of y_1' in v such that $P_{\texttt{<h>}}(x_2) \wedge P_{\texttt{<h>}}(y_2) \wedge \chi_{h,1}(x_2, y_2)$ is true, also $P_1(Succ(x_2')) \leftrightarrow P_1(Succ(y_2'))$ is true, where x_2' is the first position to the right of x_2 in u with $P_{\texttt{</h>}}(x_2')$ and y_2' is the first position to the right of y_2 in v with $P_{\texttt{</h>}}(y_2')$.

It is straightforward to check that the string w_h is the unique string in L_h that satisfies φ_h, and that φ_h has size $\mathcal{O}(||\chi_{h,1}||)$. Together with the bound $||\chi_{h,1}|| = \mathcal{O}(h)$ of Lemma 5.14, this completes the proof of Lemma 5.16. □

From Lemmas 5.15 and 5.16 we obtain, for every $h \in \mathbb{N}$, a FO(<)-sentence φ_h of size $\mathcal{O}(h)$ that defines a string-language not definable by a nondeterministic finite automaton with less than $Tower(h{+}1)$ states. This, in particular, completes the proof of Proposition 5.13. □

Using Proposition 5.13 and the results of the Sections 5.1 and 5.2, one obtains the following:

Theorem 5.17. *There is no function $f : \mathbb{N} \rightarrow \mathbb{N}$ with bound $f(m) \leqslant Tower\big(o(m)\big)$ such that for every* FO(<)*-sentence φ there is a* S-MonDatalog*-program $\mathcal{P}$ of size $||\mathcal{P}|| \leqslant f(||\varphi||)$ and a designated goal predicate $X \in IDB(\mathcal{P})$ such that $(\mathcal{P}, X)$ defines the same Boolean query as φ on the class of all finite strings.*

Proof. By contradiction. Assume that the translation from FO(<) to S-MonDatalog can be established by a function f with bound $f(m) \leqslant Tower\big(o(m)\big)$. *I.e.*, there is a $m_0 \in \mathbb{N}$ and a function $g \in \omega(1)$ such that $f(m) \leqslant Tower\big(\frac{m}{g(m)}\big)$, for all $m \geqslant m_0$; and for every FO(<)-sentence φ there is a S-MonDatalog-program $\mathcal{P}$ of size $||\mathcal{P}|| \leqslant f(||\varphi||) \leqslant Tower\big(\frac{||\varphi||}{g(||\varphi||)}\big)$ and an $X \in \mathrm{IDB}(\mathcal{P})$ such that $(\mathcal{P}, X)$ defines the same Boolean query as φ on the class of all finite strings. According to Theorem 5.9, $\mathcal{P}$ is equivalent to a MonDatalog-program $\mathcal{P}'$ of size $||\mathcal{P}'|| = 2^{\mathcal{O}(||\mathcal{P}||)} \leqslant Tower\big(\frac{||\varphi||}{g(||\varphi||)} + c\big)$, for a suitable constant $c \in \mathbb{N}$. From Theorem 5.5 we obtain an STA $\mathfrak{A}$ that defines the same unary query on the class of labelled trees as $(\mathcal{P}', X)$ and that has $|Q| \leqslant 2^{\mathcal{O}(||\mathcal{P}'||)} \leqslant Tower\big(\frac{||\varphi||}{g(||\varphi||)} + d\big)$ states, for a suitable constant $d \in \mathbb{N}$. Restricting attention to strings again, it is straightforward to transform the STA $\mathfrak{A}$ into a nondeterministic finite automaton $\mathfrak{A}'$ that accepts exactly those strings that belong to the Boolean query defined by $\mathfrak{A}$ and that has at most $2^{|Q|}$ different states. *I.e.*, every FO(<)-sentence φ can be translated into a nondeterministic finite automaton that defines the same Boolean query as φ on the class of all finite strings and that has at most $Tower(h(||\varphi||))$ states,

where the function h is defined *via* $h(m) := \frac{m}{g(m)} + d{+}1$. Obviously, $h \in o(m)$, contradicting Proposition 5.13 and completing the proof of Theorem 5.17. □

Since FO(<) is included in MLFP, the above theorem directly implies the following:

Corollary 5.18. MLFP *is not* $\mathit{Tower}\big(o(m)\big)$*-succinct in* S-MonDatalog *on the class of all finite strings.*

It remains open if this result remains valid when replacing MLFP with MLFP^2. Note, however, that for the proof of Proposition 5.13 a small number k of first-order variables suffices. *I.e.,* Proposition 5.13 remains valid when replacing FO(<) with $\mathrm{FO}^k(<)$, and Corollary 5.18 remains valid when replacing MLFP with MLFP^k.

Together with Corollary 3.5 and Lemma 5.7, the above Corollary 5.18 implies

Corollary 5.19. MLFP *is not* $\mathit{Tower}\big(o(m)\big)$*-succinct in* MLFP^2 *on the class of all finite strings.*

6. Conclusion

We studied the succinctness of a number of fixed point logics on trees. We believe that the analysis of succinctness, which may be viewed as a refined, "quantitative" analysis of expressive power, is a very interesting topic that deserves much more attention.

Even though we were able to get a good overall picture of the succinctness of monadic fixed point logics on trees, a number of questions remain open. Let us just mention a few of them:

- The exact relationship between monadic datalog, stratified monadic datalog, and the full modal μ-calculus remains unclear. In particular: Is the class of all queries whose complements can be defined by monadic datalog programs polynomially succinct in monadic datalog, or is there an exponential lower bound? (Recall that in Prop. 5.8 we prove an exponential upper bound.)
- Our proof that MSO is not $\mathit{Tower}(o(m))$-succinct in MLFP relies on a complexity theoretic assumption. Is it possible to prove this result without such an assumption?
- We have only considered the 2-variable fragment of MLFP here. What about the k-variable fragments, for $k \geqslant 3$? Do they form a strict hierarchy with respect to succinctness?

References

[1] S. Abiteboul, P. Buneman and D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann (1999).

[2] S. Abiteboul, R. Hull and V. Vianu, *Foundations of databases*. Addison-Wesley (1995).

[3] M. Adler and N. Immerman, An $n!$ lower bound on formula size. *ACM Trans. Comput. Logic* **4** (2003) 296–314.

[4] N. Alechina and N. Immerman, Reachability logic: An efficient fragment of transitive closure logic. *Logic Journal of the IGPL* **8** (2000) 325–338.

[5] A. Chandra and D. Harel, Structure and complexity of relational queries. *J. Comput. Syst. Sci.* **25** (1982) 99–128.

[6] E. Dantsin, A. Goerdt, E.A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan and U. Schöning, A deterministic $(2-2/(k+1))^n$ algorithm for k-SAT based on local search. *Theor. Comput. Sci.* **289** (2002) 69–83. Revised version of: Deterministic algorithms for k-SAT based on covering codes and local search, ICALP'00. *Lect. Notes Comput. Sci.* **1853**.

[7] H.-D. Ebbinghaus and J. Flum, *Finite Model Theory*. Springer-Verlag, 2nd edition (1999).

[8] K. Etessami, M.Y. Vardi and T. Wilke, First-order logic with two variables and unary temporal logic. *Inform. Comput.* **179** (2002) 279–295.

[9] R. Fagin, Monadic generalized spectra. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* **21** (1975) 89–96.

[10] M.F. Fernandez, J. Siméon and P. Wadler, An algebra for XML query, in *Proc. of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'00)*, edited by S. Kapoor and S. Prasad, Springer-Verlag. *Lect. Notes Comput. Sci.* **1974** (2000) 11–45.

[11] M. Frick and M. Grohe, The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, accepted (2004).

[12] M. Frick, M. Grohe and C. Koch, Query evaluation on compressed trees, in *Proc. of the 18th IEEE Symposium on Logic in Computer Science (LICS'03)* (2003) 188–197.

[13] G. Gottlob and C. Koch, Monadic datalog and the expressive power of web information extraction languages. *J. ACM* **51** (2004) 74–113.

[14] E. Grädel and M. Otto, On Logics with Two Variables. *Theor. Comput. Sci.* **224** (1999) 73–113.

[15] M. Grohe and N. Schweikardt, Comparing the succinctness of monadic query languages over finite trees, in *Proc. of the 17th International Workshop on Computer Science Logic (CSL'03)*, Springer-Verlag. *Lect. Notes Comput. Sci.* **2803** (2003) 226–240.

[16] M. Grohe and N. Schweikardt, *Comparing the succinctness of monadic query languages over finite trees*. Technical Report EDI-INF-RR-0168, School of Informatics, University of Edinburgh, Scotland, UK (2003).

[17] M. Grohe and N. Schweikardt, The succinctness of first-order logic on linear orders, in *Proc. of the 19th IEEE Symposium on Logic in Computer Science (LICS'04)* (2004) 438–447.

[18] H. Hosoya and B.C. Pierce, XDuce: A typed XML processing language (preliminary report), in *International Workshop on the Web and Databases*, edited by D. Suciu and G. Vossen (2000). Reprinted in *The Web and Databases, Selected Papers*, Springer. *Lect. Notes Comput. Sci.* **1997** (2001).

[19] N. Immerman, *Descriptive Complexity*. Springer-Verlag (1999).

[20] H. Kamp, *Tense Logic and the theory of linear order*. Ph.D. Thesis, University of California, Los Angeles (1968).

[21] C. Koch, Efficient processing of expressive node-selecting queries on XML data in secondary storage: A tree-automata based approach, in *VLDB'03: 29th Conference on Very Large Databases*, Berlin, September (2003) 249-260.

[22] F. Neven, *Design and Analysis of Query Languages for Structured Documents – A Formal and Logical Approach*. Ph.D. Thesis, Limburgs Universitair Centrum (1999).

[23] F. Neven and T. Schwentick, Query automata over finite trees. *Theor. Comput. Sci.* **275** (2002) 633–674.
[24] W. Thomas, Languages, automata, and logic, in *Handbook of formal languages* **3** (1996), edited by G. Rozenberg and A. Salomaa, Springer, New York.
[25] M.Y. Vardi, Reasoning about the past with two-way automata, in *25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, edited by K.G. Larsen, S. Skyum and G. Winskel, Springer-Verlag. *Lect. Notes Comput. Sci.* **1443** (1998) 628–641.
[26] T. Wilke, CTL$^+$ is exponentially more succinct than CTL, in *Proc. of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'99)*, Springer-Verlag. *Lect. Notes Comput. Sci.* **1738** (1999) 110–121.