

# On the Expressive Power of Monadic Least Fixed Point Logic

Nicole Schweikardt\*

Institut für Informatik, Humboldt-Universität Berlin  
Unter den Linden 6, D-10099 Berlin, Germany  
Email: [schweika@informatik.hu-berlin.de](mailto:schweika@informatik.hu-berlin.de)  
Url: <http://www.informatik.hu-berlin.de/~schweika>

**Abstract.** Monadic least fixed point logic MLFP is a natural logic whose expressiveness lies between that of first-order logic FO and monadic second-order logic MSO. In this paper we take a closer look at the expressive power of MLFP. Our results are

1. MLFP can describe graph properties beyond any fixed level of the monadic second-order quantifier alternation hierarchy.
2. On strings with built-in addition, MLFP can describe at least all languages that belong to the linear time complexity class DLIN.
3. Settling the question whether addition-invariant MLFP  $\stackrel{?}{=}$  addition-invariant MSO on finite strings would solve open problems in complexity theory: “=” would imply that  $\text{PH} = \text{PTIME}$  whereas “ $\neq$ ” would imply that  $\text{DLIN} \neq \text{LINH}$ .

## 1 Introduction

A central topic in Finite Model Theory has always been the comparison of the expressive power of different logics on finite structures. One of the main motivations for such studies is an interest in the expressive power of query languages for *relational databases* or for *semi-structured data* such as XML-documents. Relational databases can be modeled as finite relational structures, whereas XML-documents can be modeled as finite labeled trees. Since *first-order logic* FO itself is too weak for expressing many interesting queries, various extensions of FO have been considered as query languages.

When restricting attention to strings and labeled trees, *monadic second-order logic* MSO seems to be ‘just right’: it has been proposed as a yardstick for expressiveness of XML query languages [7] and, due to its connection to finite automata (cf., e.g., [25]), the model-checking problem for (Boolean and unary) MSO-queries on strings and labeled trees can be solved with polynomial time data complexity (cf., e.g., [6]). On finite relational structures in general, however, MSO can express complete problems for all levels of the polynomial time hierarchy [1], i.e., MSO can express queries that are believed to be far too difficult to allow efficient model-checking.

---

\* Parts of this work were done while the author was supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD) in order to visit the Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland, U.K.

The main focus of the present paper lies on *monadic least fixed point logic* MLFP, which is an extension of first-order logic by a mechanism that allows to define unary relations *by induction*. Precisely, MLFP is obtained by restricting the least fixed point logic FO(LFP) (cf., e.g., [16, 5]) to formulas in which only *unary* relation variables are allowed. The expressive power of MLFP lies between the expressive power of FO and the expressive power of MSO. On finite relational structures in general, MLFP has the nice properties that (1) the model-checking problem can be solved with polynomial time and linear space data complexity, and (2) MLFP is “on-spot” for the description of many important problems. For example, the transitive closure of a binary relation, or the set of winning positions in the geography game (cf., [5, Exercise 8.1.10]) can be specified by MLFP-formulas. And on strings and labeled trees, MLFP even has exactly the same expressiveness as MSO (with respect to Boolean and unary queries, cf., [25, 7]). But for all that, the logic MLFP has received surprisingly little attention in recent years. Considerably more attention has already been paid to monadic fixed point extensions of *propositional modal logic*, which are used as languages for hardware and process specification and verification. A particularly important example of such a logic is the *modal  $\mu$ -calculus* (cf., e.g., [2]), which can be viewed as the modal analogue of MLFP. *Monadic datalog*, the monadic fixed point extension of *conjunctive queries* (a subclass of FO), has recently been proposed as a database and XML query language that has a good trade-off between the expressive strength, on the one hand, and the complexity of query evaluation, on the other hand [7]. On relational structures in general, however, neither monadic datalog nor the modal  $\mu$ -calculus can express all of FO, whereas all 3 logics are included in MLFP.

As already mentioned, the expressive power of MLFP ranges between that of FO and that of MSO. Dawar [3] has shown that *3-colorability* of finite graphs is not definable in infinitary logic  $L_{\infty\omega}^\omega$ . Since all of MLFP can be expressed in  $L_{\infty\omega}^\omega$ , this implies that the (NP-complete) 3-colorability problem is definable in MSO (even, in *existential* monadic second-order logic  $\text{Mon}\Sigma_1^1$ ), but not in MLFP. Grohe [13] also exposed a *polynomial time solvable* graph problem that is not MLFP-definable, but that is definable in FO(LFP) and, as a closer inspection shows, also in MSO. Both results show that on finite graphs MLFP is strictly less expressive than MSO. The first main result of the present paper states that, nevertheless, MLFP has a certain expressive strength, as it can define graph problems beyond any fixed level of the monadic second-order quantifier alternation hierarchy:

**Theorem 1.1.** *For each  $k \geq 1$  there is an MLFP-definable graph problem that does not belong to the  $k$ -th level of the monadic second-order quantifier alternation hierarchy.*

When shifting attention from finite graphs to finite strings or labeled trees, the picture is entirely different: there, MLFP, MSO, and  $\text{Mon}\Sigma_1^1$  have the same expressive power, namely, of expressing exactly the *regular* languages (cf., [25]). To increase the expressive power of MLFP and MSO on the class of finite strings, one can allow formulas to also use the ternary relation  $+$  which is interpreted as the graph of the addition function. For a logic  $\mathcal{L}$  we write  $\mathcal{L}(+)$  to explicitly indicate that the addition predicate  $+$  may be used in  $\mathcal{L}$ -formulas. In [19] Lynch has shown that  $\text{NTIME}(n) \subseteq \text{Mon}\Sigma_1^1(+)$  on the class of finite strings with built-in addition. I.e., every string-language decidable by a nondeterministic multi-tape Turing machine with linear time bound is de-

definable by a sentence in  $\text{Mon}\Sigma_1^1(+)$ . Building upon this, one can show (cf., [21]) that  $\text{MSO}(+) = \text{LINH}$ , i.e.,  $\text{MSO}(+)$  can define exactly those string-languages that belong to the *linear time hierarchy* (which is the linear time analogue of Stockmeyer’s polynomial time hierarchy). Lynch’s result was strengthened by Grandjean and Olive [11]: They showed that  $\text{Mon}\Sigma_1^1(+)$  can even define all string-languages that belong to the complexity class NLIN. The class NLIN and its deterministic version DLIN are based on linear time random access machines and were introduced by Grandjean in a series of papers [8–10]. As argued in [10], DLIN and NLIN can be seen as “the” adequate mathematical formalizations of linear time complexity. For example, NLIN contains  $\text{NTIME}(n)$  and all 21 NP-complete problems listed by Karp in [17]. The class DLIN contains all problems in  $\text{DTIME}(n)$ , i.e. all problems decidable in linear time by a deterministic multi-tape Turing machine. But DLIN also contains problems such as CHECKSORT (given two lists  $\ell_1 = s_1, \dots, s_n$  and  $\ell_2 = t_1, \dots, t_n$  of strings, decide whether  $\ell_2$  is the lexicographically sorted version of  $\ell_1$ ) which are conjectured not to belong to  $\text{DTIME}(n)$  (see [24]). In the present paper we show the following analogue of the result of [11]:

**Theorem 1.2.** *All string-languages that belong to the linear time complexity class DLIN are definable in  $\text{MLFP}(+)$ .*

One area of research in Finite Model Theory considers extensions of logics which allow *invariant* uses of some auxiliary relations. For example, *order-invariant* formulas may use a linear ordering of a given structure’s universe, but they must not depend on the particular choice of linear ordering. This corresponds to the “real world” situation where the physical representation of a graph or a database, stored in a computer, induces a linear order on the vertices of the graph or the tuples in the database. But this particular order is hidden to the user, because one wants the user’s queries to be independent of the particular physical representation of the data. Therefore, for formulating queries, the user may be allowed to use the fact that *some* order is there, but he cannot make his queries depend on any *particular* order, because he does not know which order the data comes with. Similarly, *successor-* or *addition-invariant* formulas may use a successor-relation or an addition-relation on a structure’s universe, but must be independent of the particular choice of successor- or addition-relation. Such kinds of invariance have been investigated with respect to first-order logic, e.g., in [15, 22, 4]. In the present paper we consider *addition-invariant* formulas on finite strings and show that both, the *equivalence* of addition-invariant MLFP and MSO, as well as a *separation* of addition-invariant MLFP from MSO would solve open problems in complexity theory: Let PH denote Stockmeyer’s *polynomial time hierarchy*, and let LINH be the *linear time hierarchy*, i.e., the linear time analogue of PH.

**Theorem 1.3.** (a) *If addition-invariant MLFP  $\neq$  addition-invariant MSO on the class of finite strings, then DLIN  $\neq$  LINH.*

(b) *If addition-invariant MLFP = addition-invariant MSO on the class of finite strings, then PH = PTIME.*

In other words, it is most likely that addition-invariant MLFP  $\neq$  addition-invariant MSO on strings, but actually *proving* this can be expected to be rather difficult, since it would imply the separation of the complexity class DLIN from the linear time hierarchy LINH.

The paper is structured as follows: Section 2 fixes the basic notations and gives an example of the present paper's use of MLFP-formulas. Theorem 1.1 is proved in Section 3. Section 4 concentrates on the proof of Theorem 1.2. Section 5 deals with the proof of Theorem 1.3. Some open questions are pointed out in Section 6. Due to space limitations, detailed proofs had to be deferred to the full version of this paper [23].

### Acknowledgments.

I want to thank Martin Grohe for valuable discussions on the subject of this paper.

## 2 Preliminaries

For an alphabet  $\mathbb{A}$  we write  $\mathbb{A}^+$  to denote the set of all finite non-empty strings over  $\mathbb{A}$ . For a set  $\mathcal{U}$  we write  $2^{\mathcal{U}}$  to denote the power set of  $\mathcal{U}$ , i.e.,  $2^{\mathcal{U}} := \{X : X \subseteq \mathcal{U}\}$ . We use  $\mathbb{N}$  to denote the set  $\{0, 1, 2, \dots\}$  of natural numbers. For every  $n \in \mathbb{N}$  we write  $[n]$  for the set  $\{0, \dots, n-1\}$ . The logarithm of  $n$  with respect to base 2 is denoted  $\lg n$ .

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

We assume that the reader is familiar with *first-order logic* FO, *monadic second-order logic* MSO, and *least fixed point logic* FO(LFP) (cf., e.g., the textbooks [5, 16]). The  $k$ -th level,  $\text{Mon}\Sigma_k^1$ , of the monadic second-order quantifier alternation hierarchy consists of all MSO-formulas that are in prenex normal form, having a prefix of  $k$  alternating blocks of set quantifiers, starting with an existential block, and followed by a first-order formula.

For a logic  $\mathcal{L}$  we use  $\mathcal{L}(\tau)$  to denote the class of all  $\mathcal{L}$ -formulas of signature  $\tau$ . We write  $\varphi(x_1, \dots, x_k, X_1, \dots, X_\ell)$  to indicate that the free first-order variables of the formula  $\varphi$  are  $x_1, \dots, x_k$  and the free second-order variables are  $X_1, \dots, X_\ell$ . Sometimes we use  $\vec{x}$  and  $\vec{X}$  as abbreviations for sequences  $x_1, \dots, x_k$  and  $X_1, \dots, X_\ell$  of variables. A *sentence*  $\varphi$  of signature  $\tau$  is a formula that has no free variable.

Let  $\tau$  be a signature, let  $\mathcal{C}$  be a class of  $\tau$ -structures, and let  $\mathcal{L}$  be a logic. We say that a set  $L \subseteq \mathcal{C}$  is  *$\mathcal{L}$ -definable in  $\mathcal{C}$*  if there is a sentence  $\varphi \in \mathcal{L}(\tau)$  such that  $L = \{\mathcal{A} \in \mathcal{C} : \mathcal{A} \models \varphi\}$ . Similarly, we say that a set  $L$  of  $\tau$ -structures is  *$\mathcal{L}$ -definable*, iff it is  $\mathcal{L}$ -definable in the class of all (finite)  $\tau$ -structures.

We will mainly consider the *monadic least fixed point logic* MLFP, which is the restriction of least fixed point logic FO(LFP), where fixed point operators are required to be *unary*. For the precise definition of MLFP we refer the reader to the textbook [5] (MLFP is denoted FO(M-LFP) there). *Simultaneous monadic least fixed point logic* S-MLFP is the extension of MLFP by operators that allow to compute the simultaneous least fixed point of several unary operators. In other words: S-MLFP is obtained by restricting simultaneous least fixed point logic FO(S-LFP) to unary fixed point relations. For the formal definition of FO(S-LFP) we, again, refer to [5]. The following example illustrates the present paper's use of S-MLFP-formulas.

*Example 2.1.* Let  $\tau_{<, +}$  be the signature that consists of a binary relation symbol  $<$  and a ternary relation symbol  $+$ . For every  $n \in \mathbb{N}$  let  $\mathcal{A}_n$  be the  $\tau_{<, +}$ -structure with universe

$[n] = \{0, \dots, n-1\}$ , where  $<$  is interpreted by the natural linear ordering and  $+$  is interpreted by the graph of the addition function, i.e.,  $+$  consists of all triples  $(a, b, c)$  over  $[n]$  where  $a+b = c$ . Consider the formulas

$$\begin{aligned}\varphi_S(x, S, P) &:= \text{“}x=0\text{”} \vee \text{“}x=1\text{”} \vee \\ &\quad \exists x_1 \exists x_2 \left( x_1 < x_2 \wedge x_2 < x \wedge S(x_1) \wedge S(x_2) \wedge \right. \\ &\quad \quad \left. \forall z \left( (x_1 < z \wedge z < x_2) \rightarrow P(z) \right) \wedge \text{“}x-x_2 = x_2-x_1+2\text{”} \right) \\ \varphi_P(y, S, P) &:= \exists x_1 \exists x_2 \left( x_1 < x_2 \wedge x_2 < y \wedge S(x_1) \wedge S(x_2) \wedge \right. \\ &\quad \quad \left. \forall z \left( (x_1 < z \wedge z < x_2) \rightarrow P(z) \right) \wedge \text{“}y-x_2 < x_2-x_1+2\text{”} \right).\end{aligned}$$

Of course, the subformulas written in quotation marks “ $\dots$ ” can easily be resolved by proper  $\text{FO}(\tau_{<, +})$ -formulas. In the structure  $\mathcal{A}_n$ , the simultaneous least fixed point  $(S_{\mathcal{A}_n}^{(\infty)}, P_{\mathcal{A}_n}^{(\infty)})$  of  $(\varphi_S, \varphi_P)$  is evaluated as follows: We start with the 0-th stage, where  $S$  and  $P$  are interpreted by the sets  $S_{\mathcal{A}_n}^{(0)} = P_{\mathcal{A}_n}^{(0)} = \emptyset$ . Inductively, for every  $i \in \mathbb{N}$ , the  $(i+1)$ -st stage is obtained via

$$\begin{aligned}S_{\mathcal{A}_n}^{(i+1)} &:= \{ a \in [n] : \mathcal{A}_n \models \varphi_S(a, S_{\mathcal{A}_n}^{(i)}, P_{\mathcal{A}_n}^{(i)}) \} \\ P_{\mathcal{A}_n}^{(i+1)} &:= \{ b \in [n] : \mathcal{A}_n \models \varphi_P(b, S_{\mathcal{A}_n}^{(i)}, P_{\mathcal{A}_n}^{(i)}) \}.\end{aligned}$$

In particular,

$$\begin{aligned}S_{\mathcal{A}_n}^{(1)} &= \{0, 1\}, & S_{\mathcal{A}_n}^{(2)} &= \{0, 1, 4\}, & S_{\mathcal{A}_n}^{(3)} &= \{0, 1, 4, 9\}, & S_{\mathcal{A}_n}^{(4)} &= \{0, 1, 4, 9, 16\}, \\ P_{\mathcal{A}_n}^{(1)} &= \emptyset, & P_{\mathcal{A}_n}^{(2)} &= \{2, 3\}, & P_{\mathcal{A}_n}^{(3)} &= \{2, 3, 5, 6, 7, 8\} & \dots &.\end{aligned}$$

At some stage  $i$  (with  $i \leq n$ ), this process arrives at a fixed point, i.e., at a situation where  $S_{\mathcal{A}_n}^{(i)} = S_{\mathcal{A}_n}^{(i+1)} = S_{\mathcal{A}_n}^{(j)}$  and  $P_{\mathcal{A}_n}^{(i)} = P_{\mathcal{A}_n}^{(i+1)} = P_{\mathcal{A}_n}^{(j)}$ , for every  $j > i$ . This particular tuple  $(S_{\mathcal{A}_n}^{(i)}, P_{\mathcal{A}_n}^{(i)})$  is called the *simultaneous least fixed point*  $(S_{\mathcal{A}_n}^{(\infty)}, P_{\mathcal{A}_n}^{(\infty)})$  of  $(\varphi_S, \varphi_P)$  in  $\mathcal{A}_n$ . It is not difficult to see that for our example formulas  $\varphi_S$  and  $\varphi_P$  we obtain that  $S_{\mathcal{A}_n}^{(\infty)}$  is the set of all square numbers in  $[n]$ , whereas  $P_{\mathcal{A}_n}^{(\infty)}$  is the set of all non-square numbers in  $[n]$ .

Now,  $[\text{S-LFP}_{x,S,y,P\varphi_S,\varphi_P}]_S(u)$  is an S-MLFP-formula that is satisfied by exactly those elements  $u$  in  $\mathcal{A}_n$ 's universe that belong to  $S_{\mathcal{A}_n}^{(\infty)}$ , i.e., that are square numbers. Similarly,  $[\text{S-LFP}_{x,S,y,P\varphi_S,\varphi_P}]_P(u)$  is an S-MLFP-formula that is satisfied by those elements  $u$  in  $\mathcal{A}_n$ 's universe that belong to  $P_{\mathcal{A}_n}^{(\infty)}$ , i.e., that are non-square numbers.

In the above example we have seen that, given the addition relation  $+$ , the set of square numbers is definable in S-MLFP. It is known (cf., e.g., [14, Corollary 5.3]) that MLFP has the same expressive power as S-MLFP. Since S-MLFP-definitions of certain properties or relations are sometimes easier to find and more convenient to read than equivalent MLFP-definitions, we will often present S-MLFP-definitions instead of MLFP-definitions.

### 3 MLFP and the MSO quantifier alternation hierarchy

In this section we show that MLFP can define graph problems beyond any fixed level of the monadic second-order quantifier alternation hierarchy.

Let  $\tau_{\text{graph}}$  be the signature that consists of a binary relation symbol  $E$ . We write  $\mathcal{C}_{\text{graphs}}$  for the class of all finite directed graphs. A graph  $G = \langle V^G, E^G \rangle$  is called *undirected* if the following is true: for every  $v \in V^G$ ,  $(v, v) \notin E^G$ , and for every  $(v, w) \in E^G$ , also  $(w, v) \in E^G$ . We write  $\mathcal{C}_{\text{ugraphs}}$  to denote the class of all finite undirected graphs. Let  $\tau_{\text{grid}} := \{S_1, S_2\}$  be a signature consisting of two binary relation symbols. The *grid* of height  $m$  and width  $n$  is the  $\tau_{\text{grid}}$ -structure

$$\underline{[m, n]} := \langle \{1, \dots, m\} \times \{1, \dots, n\}, S_1^{m,n}, S_2^{m,n} \rangle,$$

where  $S_1^{m,n}$  is the ‘vertical’ successor relation consisting of all tuples  $((i, j), (i+1, j))$  in  $\{1, \dots, m\} \times \{1, \dots, n\}$ , and  $S_2^{m,n}$  is the ‘horizontal’ successor relation consisting of all tuples  $((i, j), (i, j+1))$ . We define  $\mathcal{C}_{\text{grids}} := \{\underline{[m, n]} : m, n \geq 1\}$  to be the class of all finite grids. It was shown in [20] that the monadic second-order quantifier alternation hierarchy is *strict* on the class of finite graphs and the class of finite grids. In the present paper we will use the following result:

**Theorem 3.1 ([20]).** *For every  $k \geq 1$  there is a set  $L_k$  of finite grids such that  $L_k$  is definable in  $\text{Mon}\Sigma_k^1$  but not in  $\text{Mon}\Sigma_{k-1}^1$  (on the class of finite grids).*

Using the construction of [20] and the fact that MLFP is as expressive as S-MLFP, it is an easy (but tedious) exercise to show the following

**Corollary 3.2.** *For every  $k \geq 1$  the set  $L_k$  is definable in MLFP and in  $\text{Mon}\Sigma_k^1$ , but not in  $\text{Mon}\Sigma_{k-1}^1$  (on the class of finite grids).*

Note that the above corollary deals with structures over the signature  $\tau_{\text{grid}}$  that consists of two binary relation symbols. In the remainder of this section we will transfer this to the classes  $\mathcal{C}_{\text{graphs}}$  and  $\mathcal{C}_{\text{ugraphs}}$ . To this end, we need a further result of [20] which uses the notion of *strong first-order reductions*. The precise definition of this notion is of no particular importance for the present paper. What is important is that a strong first-order reduction from a class  $\mathcal{C}$  of  $\tau$ -structures to a class  $\mathcal{C}'$  of  $\tau'$ -structures is an injective mapping  $\Phi : \mathcal{C} \rightarrow \mathcal{C}'$  such that every structure  $\mathcal{A} \in \mathcal{C}$  can be interpreted in the structure  $\Phi(\mathcal{A})$  and, vice versa,  $\Phi(\mathcal{A})$  can be interpreted in  $\mathcal{A}$ . The fundamental use of strong first-order reductions comes from the following result:

**Theorem 3.3 ([20, Theorem 33]).** *Let  $\mathcal{C}$  and  $\mathcal{C}'$  be classes of structures over the relational signatures  $\tau$  and  $\tau'$ , respectively. Let  $\Phi$  be a strong first-order reduction from  $\mathcal{C}$  to  $\mathcal{C}'$ . Let  $\mathcal{L}$  be one of the logics  $\text{Mon}\Sigma_k^1$ , for some  $k \geq 0$ . Let the image  $\Phi(\mathcal{C}) := \{\Phi(\mathcal{A}) : \mathcal{A} \in \mathcal{C}\}$  of  $\Phi$  be  $\mathcal{L}$ -definable in  $\mathcal{C}'$ . Then, the following is true for every  $L \subseteq \mathcal{C}$ :*

$$L \text{ is } \mathcal{L}\text{-definable in } \mathcal{C} \iff \Phi(L) \text{ is } \mathcal{L}\text{-definable in } \mathcal{C}'.$$

In the present paper, the following strong first-order reductions will be used:

**Proposition 3.4 ([20, Proposition 38]).**

- (a) *There exists a strong first-order reduction  $\Phi_1$  from  $\mathcal{C}_{\text{grids}}$  to  $\mathcal{C}_{\text{graphs}}$ , and the image  $\Phi_1(\mathcal{C}_{\text{grids}})$  of  $\Phi_1$  is  $\text{Mon}\Sigma_2^1$ -definable and MLFP-definable in  $\mathcal{C}_{\text{graphs}}$ .*
- (b) *There exists a strong first-order reduction  $\Phi_2$  from  $\mathcal{C}_{\text{graphs}}$  to  $\mathcal{C}_{\text{ugraphs}}$ , and the image  $\Phi_2(\mathcal{C}_{\text{graphs}})$  of  $\Phi_2$  is FO-definable in  $\mathcal{C}_{\text{ugraphs}}$ .*

This directly allows to transfer Theorem 3.1 from finite grids to finite graphs and finite undirected graphs, respectively. To also transfer Corollary 3.2 from  $\mathcal{C}_{\text{grids}}$  to  $\mathcal{C}_{\text{graphs}}$  and  $\mathcal{C}_{\text{ugraphs}}$ , we need the following easy lemma:

**Lemma 3.5.** *Let  $\mathcal{C}$  and  $\mathcal{C}'$  be classes of structures over the relational signatures  $\tau$  and  $\tau'$ , respectively. Let  $\Phi$  be a strong first-order reduction from  $\mathcal{C}$  to  $\mathcal{C}'$ . Every MLFP( $\tau$ )-sentence  $\psi$  can be translated into an MLFP( $\tau'$ )-sentence  $\psi'$  such that, for every  $\mathcal{A} \in \mathcal{C}$ ,  $\mathcal{A} \models \psi \iff \Phi(\mathcal{A}) \models \psi'$ .*

Using this, it is not difficult to prove this section's main result:

**Theorem 3.6.** *For every  $k \geq 2$  there is a set  $D_k$  of finite directed graphs (respectively, a set  $U_k$  of finite undirected graphs) such that  $D_k$  (respectively,  $U_k$ ) is definable in MLFP and  $\text{Mon}\Sigma_k^1$ , but not in  $\text{Mon}\Sigma_{k-1}^1$ .*

## 4 MLFP and linear time complexity

We identify a string  $w = w_0 \cdots w_{n-1}$  of length  $|w| = n \geq 1$  over an alphabet  $\mathbb{A}$  with a structure  $\underline{w}$  in the usual way: We choose  $\tau_{\mathbb{A}}$  to consist of the binary relation symbol  $<$  and a unary relation symbol  $P_a$ , for each letter  $a \in \mathbb{A}$ . We choose  $\underline{w}$  to be the  $\tau_{\mathbb{A}}$ -structure  $\langle \{0, \dots, n-1\}, <, (P_a^w)_{a \in \mathbb{A}} \rangle$ , where  $<$  denotes the natural linear ordering of  $[n] = \{0, \dots, n-1\}$  and  $P_a^w$  consists of all positions of  $w$  that carry the letter  $a$ .

In this section we equip the structure  $\underline{w}$  with an additional ternary addition relation  $+$ . I.e., we identify the string  $w$  with the structure  $\langle \underline{w}, + \rangle := \langle [n], <, +, (P_a^w)_{a \in \mathbb{A}} \rangle$ , where  $+$  consists of all triples  $(a, b, c) \in [n]^3$  with  $a + b = c$ . We identify the set  $\mathbb{A}^+$  of all non-empty strings over alphabet  $\mathbb{A}$  with the set  $\mathcal{C}_{\mathbb{A}} := \{\underline{w} : w \in \mathbb{A}^+\}$ , respectively, with the set  $\mathcal{C}_{\mathbb{A},+} := \{\langle \underline{w}, + \rangle : w \in \mathbb{A}^+\}$ .

To give the precise definition of Grandjean's linear time complexity class DLIN, we need the following notion of *random access machines*, basically taken from [12].

A DLIN-RAM  $\mathcal{R}$  is a random access machine that consists of two *accumulators*  $A$  and  $B$ , a *special register*  $M$ , *registers*  $R_i$ , for every  $i \in \mathbb{N}$ , and a *program* that is a finite sequence  $\mathcal{I}(1), \dots, \mathcal{I}(r)$  of *instructions*, each of which is of one of the following forms:

- $A := 0$ ,      •  $A := A + B$ ,      •  $M := A$ ,      • IF  $A=B$  THEN  $\mathcal{I}(i_0)$  ELSE  $\mathcal{I}(i_1)$ ,
- $A := 1$ ,      •  $A := R_A$ ,      •  $B := A$ ,      • HALT.
- $A := M$ ,      •  $R_A := B$ ,

The meaning of most of these instructions is straightforward. If  $A$  contains a number  $i$ , then the execution of the instruction  $A := R_A$  copies the content of register  $R_i$  into the accumulator  $A$ . Similarly, the execution of the instruction  $R_A := B$  copies the content of accumulator  $B$  into register  $R_i$ . We stipulate that the last instruction,  $\mathcal{I}(r)$ , is the instruction HALT.

The input to  $\mathcal{R}$  is assumed to be present in the first registers of  $\mathcal{R}$  at the beginning of the computation. Precisely, an *input to  $\mathcal{R}$*  is a function  $f : [m] \rightarrow [m]$ , for an arbitrary  $m \in \mathbb{N}$ . The initial content of the special register  $M$  is the number  $m$ , and for every  $i \in \mathbb{N}$ , the initial content of register  $R_i$  is  $f(i)$  if  $i \in [m]$ , and 0 otherwise. The accumulators  $A$  and  $B$  are initialized by 0. The computation of  $\mathcal{R}$  starts with instruction  $\mathcal{I}(1)$  and finishes when it encounters a HALT statement. We say that  $\mathcal{R}$  *accepts* an input

$f$ , if the content of register  $R_0$  is non-zero when  $\mathcal{R}$  reaches a HALT statement.  
 $\mathcal{R}$  recognizes a set  $\mathcal{F} \subseteq \{f : [m] \rightarrow [m] : m \in \mathbb{N}\}$  in time  $\mathcal{O}(m)$ , if

1.  $\mathcal{R}$  accepts an input  $f$  if, and only if,  $f \in \mathcal{F}$ , and
2. there is a number  $d \in \mathbb{N}$  such that  $\mathcal{R}$  is  $d$ -bounded, i.e., for every  $m \in \mathbb{N}$  and every  $f : [m] \rightarrow [m]$  the following is true: when started with input  $f$ ,  $\mathcal{R}$  performs less than  $d \cdot m$  computation steps before reaching a HALT statement, and throughout the computation, each register and each accumulator contains numbers of size  $< d \cdot m$ .

To use DLIN-RAMs for recognizing *string*-languages, one represents strings  $w$  by functions  $f_w$  as follows (cf., [11]). W.l.o.g. we restrict attention to strings over the alphabet  $\mathbb{A} := \{1, 2\}$ . For every  $n \geq 1$  we define  $\ell(n) := \lceil \frac{1}{2} \lg(n+1) \rceil$  and  $m(n) := \lceil \frac{n}{\ell(n)} \rceil$ . A string  $w$  over  $\mathbb{A} = \{1, 2\}$  of length  $n$  can (uniquely) be decomposed into substrings  $w_0, w_1, \dots, w_{m(n)-1}$  such that

- $w$  is the concatenation of the strings  $w_0, \dots, w_{m(n)-1}$ ,
- $w_i$  has length  $\ell(n)$ , for every  $i < m(n)-1$ , and
- $w_{m(n)-1}$  has length at most  $\ell(n)$ .

For each  $i \in [m(n)]$  let  $w_i^{dy}$  be the integer whose dyadic representation is  $w_i$ . I.e., if  $w_i = d_0 \cdots d_{\ell(n)-1}$  with  $d_j \in \{1, 2\}$ , then  $w_i^{dy} = \sum_{j < \ell(n)} d_j \cdot 2^j$ . It is straightforward to see that  $w_i^{dy} < m(n)$ . Now,  $w$  is represented by the function  $f_w : [m(n)] \rightarrow [m(n)]$  with  $f_w(i) := w_i^{dy}$ , for every  $i \in [m(n)]$ .

**Definition 4.1 (DLIN, [10]).** A string-language  $L$  over alphabet  $\mathbb{A} = \{1, 2\}$  belongs to the complexity class DLIN if, and only if, the set of its associated functions  $\{f_w : w \in L\}$  is recognized by a DLIN-RAM in time  $\mathcal{O}(m)$ .

At first sight, the class DLIN may seem a bit artificial: a string  $w$  of length  $n$  is represented by a function  $f_w$  of domain  $[m(n)]$  where  $m(n)$  is of size  $\Theta(\frac{n}{\lg n})$ . A DLIN-RAM with input  $f_w$  is allowed to perform only  $\mathcal{O}(\frac{n}{\lg n})$  computation steps, with register contents of size  $\mathcal{O}(\frac{n}{\lg n})$ . However, as argued in [8–10, 12], DLIN is a very reasonable formalization of the intuitive notion of ‘linear time complexity’. In particular, DLIN contains all string-languages recognizable by a deterministic Turing machine in  $\mathcal{O}(n)$  steps, and, in addition, also some problems (such as CHECKSORT, cf., Section 1) that are conjectured not to be solvable by Turing machines with time bound  $\mathcal{O}(n)$ .

Grandjean and Olive [11] showed that  $\text{Mon}\Sigma_1^1(+)$  can define (at least) all string-languages that belong to the nondeterministic version NLIN of DLIN. In the remainder of this section we show the following analogue of the result of [11]:

**Theorem 4.2 (DLIN  $\subseteq$  MLFP(+)) on finite strings with built-in addition.**

For every finite alphabet  $\mathbb{A}$  and every string-language  $L \subseteq \mathbb{A}^+$  in DLIN there is an MLFP( $\tau_{\mathbb{A}} \cup \{+\}$ )-sentence  $\varphi_L$  such that, for every  $w \in \mathbb{A}^+$ ,  $w \in L$  iff  $\langle \underline{w}, + \rangle \models \varphi_L$ .

The proof of [11]’s result on NLIN and  $\text{Mon}\Sigma_1^1(+)$  uses, as an intermediate step, a characterization of the class NLIN by a logic that existentially quantifies unary functions. There also exists an algebraic characterization of the class DLIN via unary functions

[12]. Unfortunately, this characterization is not suitable for being used as an intermediate step in the proof of Theorem 4.2. What *can* be used for the proof of Theorem 4.2, however, is the following representation, basically taken from [11], of a *run* of a  $d$ -bounded DLIN-RAM  $\mathcal{R}$ . A run of  $\mathcal{R}$  with input  $f : [m] \rightarrow [m]$  is fully described by 6 functions  $I, A, B, M, R_A, R'_A : [d \cdot m] \rightarrow [d \cdot m]$ :

$$\begin{aligned}
I(t) &= \text{the number of the instruction performed in computation step } t+1 \\
A(t) &= \text{content of the accumulator } A \text{ directly } \textit{before} \text{ performing step } t+1 \\
B(t) &= \text{content of the accumulator } B \text{ directly } \textit{before} \text{ performing step } t+1 \\
M(t) &= \text{content of the special register } M \text{ directly } \textit{before} \text{ performing step } t+1 \\
R_A(t) &= \text{content of register } R_{A(t)} \text{ directly } \textit{before} \text{ performing step } t+1 \\
R'_A(t) &= \text{content of register } R_{A(t)} \text{ directly } \textit{after} \text{ performing step } t+1.
\end{aligned}$$

It is not difficult to give inductive definitions of these functions

The *flattening*  $\tilde{G}$  of a function  $G : [d \cdot m] \rightarrow [d \cdot m]$  is the concatenation of the  $\{0, 1\}$ -strings  $\tilde{G}_0, \tilde{G}_1, \dots, \tilde{G}_{d \cdot m - 1}$ , where  $\tilde{G}_i$  is the reverse binary representation of length  $l := \lceil \lg(d \cdot m) + 1 \rceil$  of the number  $G(i)$ . I.e.,  $\tilde{G}_i = b_0 b_1 \dots b_{l-1}$  with  $b_j \in \{0, 1\}$  and  $G(i) = \sum_{j < l} b_j \cdot 2^j$ . It is straightforward to see that for every  $d \in \mathbb{N}$  there is a  $c \in \mathbb{N}$  such that the following is true for every  $n \in \mathbb{N}$  and every function  $G : [d \cdot m(n)] \rightarrow [d \cdot m(n)]$ : The flattening  $\tilde{G}$  of  $G$  is a  $\{0, 1\}$ -string of length  $\leq c \cdot n$ . Consequently,  $\tilde{G}$  can be represented by  $c$  subsets  $\tilde{G}^{(0)}, \dots, \tilde{G}^{(c-1)}$  of  $[n]$  as follows: For every  $p \in [n]$  and  $\gamma \in [c]$ , the  $(\gamma \cdot n + p)$ -th position of  $\tilde{G}$  carries the letter 1 if, and only if,  $p \in \tilde{G}^{(\gamma)}$ .

We write  $\tilde{G}^\bullet$  for the *complement* of  $\tilde{G}$ , i.e., the  $\{0, 1\}$ -string obtained from  $\tilde{G}$  by replacing every 0 by 1 and every 1 by 0. Similarly, for  $\gamma \in [c]$ ,  $\tilde{G}^{\bullet(\gamma)}$  denotes the complement of the set  $\tilde{G}^{(\gamma)}$ .

Clearly, given a string  $w$  of length  $n$  and its functional representation  $f_w : [m(n)] \rightarrow [m(n)]$ , the flattenings (and their complements) of the functions  $I, A, B, M, R_A, R'_A : [d \cdot m(n)] \rightarrow [d \cdot m(n)]$  that describe the computation of  $\mathcal{R}$  on input  $f_w$ , can be represented by a fixed number of subsets of  $[n]$ . Using the inductive definitions of the functions  $I, A, B, M, R_A, R'_A$  mentioned above, we can show the following:

**Lemma 4.3.** *Let  $\mathbb{A} := \{1, 2\}$ , let  $L \subseteq \mathbb{A}^+$ , let  $d \in \mathbb{N}$ , let  $\mathcal{R}$  be a  $d$ -bounded DLIN-RAM that recognizes the set  $\{f_w : w \in L\}$ , and let  $c \in \mathbb{N}$  be such that, for every  $n \geq 1$ , the flattening  $\tilde{G}$  of every function  $G : [d \cdot m(n)] \rightarrow [d \cdot m(n)]$  is a  $\{0, 1\}$ -string of length  $\leq c \cdot n$ . For every symbol  $S \in \mathcal{S} := \{I, A, B, M, R_A, R'_A, I^\bullet, A^\bullet, B^\bullet, M^\bullet, R_A^\bullet, R'_A^\bullet\}$  and every  $\gamma \in [c]$  let  $X_{S, \gamma}$  be a set variable. Let  $\overline{X_{\mathcal{S}, c}}$  be the list of the set variables  $X_{S, \gamma}$  for all  $S \in \mathcal{S}$  and all  $\gamma \in [c]$ .*

*For every  $S \in \mathcal{S}$  and every  $\gamma \in [c]$  there is an MLFP( $\tau_{\mathbb{A}} \cup \{+\}$ )-formula  $\varphi_{S, \gamma}(x, \overline{X_{\mathcal{S}, c}})$  such that the following is true for every string  $w \in \mathbb{A}^+$ :*

*Let  $n$  be the length of  $w$ , let  $f_w : [m(n)] \rightarrow [m(n)]$  be the functional representation of  $w$ , and let  $I, A, B, M, R_A, R'_A : [d \cdot m(n)] \rightarrow [d \cdot m(n)]$  be the functions that describe the computation of  $\mathcal{R}$  on input  $f_w$ . In the structure  $\langle \underline{w}, + \rangle$ , the simultaneous least fixed point of all the formulas  $\varphi_{S, \gamma}$  (for all  $S \in \mathcal{S}$  and  $\gamma \in [c]$ ) consists exactly of the sets  $(\tilde{S}^{(\gamma)})_{S \in \mathcal{S}, \gamma \in [c]}$  that represent the flattenings, and their complements, of the functions  $I, A, B, M, R_A, R'_A$ .*

An important tool for the proof of Lemma 4.3 is the following

**Lemma 4.4 (full arithmetic and counting in MLFP(+)).**

(a) There are MLFP(+)-formulas

$$\varphi_{<}(x, y), \quad \varphi_{\times}(x, y, z), \quad \varphi_{\text{Exp}}(x, y, z), \quad \varphi_{\text{Bit}}(x, y), \quad \varphi_{\text{Dy}_1}(x, y), \quad \varphi_{\text{Dy}_2}(x, y),$$

such that for all  $n \in \mathbb{N}$  and all  $a, b, c \in [n]$ ,  $\langle [n], + \rangle \models \varphi_{<}(a, b)$  (respectively,  $\varphi_{\times}(a, b, c)$ ,  $\varphi_{\text{Exp}}(a, b, c)$ ,  $\varphi_{\text{Bit}}(a, b)$ ,  $\varphi_{\text{Dy}_1}(a, b)$ ,  $\varphi_{\text{Dy}_2}(a, b)$ ) if, and only if,  $a < b$  (resp.,  $a \times b = c$ ,  $a^b = c$ , the  $b$ -th bit in the binary representation of  $a$  is 1, the  $b$ -th bit in the dyadic representation of  $a$  is 1, resp., 2).

(b) Let  $Y$  be a unary relation symbol. There is an MLFP(+)-formula  $\varphi_{\#}(x, Y)$  such that for all  $n \in \mathbb{N}$ , all  $a \in [n]$ , and all  $B \subseteq [n]$  we have that

$$\langle [n], + \rangle \models \varphi_{\#}(a, B) \iff a = |B|.$$

Using Lemma 4.3, Lemma 4.4, and the fact that MLFP has the same expressive power as S-MLFP (cf., Section 2), it is rather straightforward to find an MLFP( $\tau_{\mathbb{A}} \cup \{+\}$ )-sentence  $\varphi_L$  which, for every string  $w \in \mathbb{A}^+$ , is satisfied by  $\langle \underline{w}, + \rangle$  if, and only if,  $\mathcal{R}$  accepts input  $f_w$ . This, finally, will complete the proof of Theorem 4.2.

## 5 Addition-invariant MLFP

In this section we concentrate on *addition invariant* formulas, i.e., on formulas that may use an addition relation on the underlying universe but that are independent of the particular choice of the addition relation.

The notion of “addition relation” is defined as follows: Let  $\mathcal{U}$  be a finite set, let  $n := |\mathcal{U}|$ , and let  $\oplus$  be a ternary relation on  $\mathcal{U}$ .  $\oplus$  is called an *addition relation on  $\mathcal{U}$*  if there is a linear ordering  $\otimes$  of  $\mathcal{U}$  such that  $\mathcal{U} = \{u_0, \dots, u_{n-1}\}$  with  $u_0 \otimes \dots \otimes u_{n-1}$  and  $\oplus = \{(u_i, u_j, u_k) : i + j = k \text{ and } i, j, k \in \{0, \dots, n-1\}\}$ .

We say that  $\oplus$  is the particular addition relation that fits to the linear ordering  $\otimes$ .

**Definition 5.1 (addition-invariance).** Let  $\mathcal{L}$  be a logic, let  $\tau$  be a signature, and let  $\oplus$  be a ternary relation symbol that does not occur in  $\tau$ . An  $\mathcal{L}(\tau \cup \{\oplus\})$ -formula  $\varphi(x_1, \dots, x_k)$  is called *addition-invariant* if the following is true for all finite  $\tau$ -structures  $\mathcal{A}$ : For any two addition relations  $\oplus_1$  and  $\oplus_2$  on  $\mathcal{U}^{\mathcal{A}}$  and all  $a_1, \dots, a_k \in \mathcal{U}^{\mathcal{A}}$  we have  $\langle \mathcal{A}, \oplus_1 \rangle \models \varphi(a_1, \dots, a_k) \iff \langle \mathcal{A}, \oplus_2 \rangle \models \varphi(a_1, \dots, a_k)$ .

Using Lemma 4.4, we can show

**Lemma 5.2.** On linearly ordered structures, addition-invariant MLFP can define the particular addition relation that fits to the given linear ordering of the underlying structure.

From Theorem 4.2 and Lemma 5.2 one directly obtains

**Corollary 5.3 (DLIN  $\subseteq$  addition-invariant MLFP on the class of finite strings).** For every finite alphabet  $\mathbb{A}$  and every string-language  $L \subseteq \mathbb{A}^+$  in DLIN there is an addition-invariant MLFP-sentence  $\varphi$  of signature  $\tau_{\mathbb{A}} \cup \{\oplus\}$  such that, for every string  $w \in \mathbb{A}^+$  and every addition relation  $\oplus$  on  $\underline{w}$ 's universe,  $w \in L$  iff  $\langle \underline{w}, \oplus \rangle \models \varphi$ .

Using this and the well-known result that the satisfiability problem for *quantified Boolean formulas with  $k$  alternations of quantifiers* is complete for the  $k$ -th level of the polynomial time hierarchy, we can show that both, the equivalence of addition-invariant MLFP and MSO, as well as a separation of addition-invariant MLFP from MSO would solve open problems in complexity theory:

- Theorem 5.4.** (a) *If addition-invariant MLFP  $\neq$  addition-invariant MSO on the class of finite strings, then  $DLIN \neq LINH$ .*  
 (b) *If addition-invariant MLFP = addition-invariant MSO on the class of finite strings, then  $PH = PTIME$ .*

## 6 Conclusion

The main results of the present paper are (1) that MLFP can express graph properties beyond any fixed level of the monadic second-order quantifier alternation hierarchy, (2) that *addition-invariant* MLFP can express at least all string-problems that belong to the linear time complexity class DLIN, and (3) that settling the question whether addition-invariant MLFP has the same expressive power as addition-invariant MSO on finite strings would solve open problems in complexity theory.

Many interesting aspects of MLFP remain to be further investigated, for example:

- Is there a natural complexity class that is *exactly* captured by MLFP(+) on strings with built-in addition (analogous to the known result that MSO(+) exactly captures the linear time hierarchy LINH)? A promising candidate might be the time-space complexity class PTIME&LINSPEACE of problems solvable by deterministic polynomial time, linear space bounded Turing machines.
- Is there a hierarchy within MLFP with respect to the alternation of *least* and *greatest* fixed point quantifiers? I.e., does MLFP have a hierarchy analogous to Bradfield's modal  $\mu$ -calculus alternation hierarchy [2]? Note that every level of this MLFP alternation hierarchy is closed under first-order quantification. Therefore, the alternation hierarchy of MLFP might be viewed as a “deterministic” analogue of the *closed monadic hierarchy* of [1] rather than as an analogue of the monadic second-order quantifier alternation hierarchy of [20].
- Investigate the parameterized complexity of the model checking problem for MLFP on various classes of finite structures. E.g., is the model checking problem for MLFP fixed parameter tractable on the class of planar graphs? Partial answers to this question have been obtained by Lindell [18].
- Does Theorem 3.6 still hold when replacing MLFP with the modal  $\mu$ -calculus?

## References

1. M. Ajtai, R. Fagin, and L. Stockmeyer. The closure of Monadic NP. *Journal of Computer and System Sciences*, 60(3):660–716, 2000. Journal version of STOC'98 paper.
2. J. Bradfield. The modal  $\mu$ -calculus alternation hierarchy is strict. *Theoretical Computer Science*, 195(2):133–153, 1998. Journal version of CONCUR'96 paper.

3. A. Dawar. A restricted second order logic for finite structures. *Information and Computation*, 143:154–174, 1998. Journal version of *LCC'94* paper.
4. A. Durand, C. Lautemann, and M. More. Counting results in weak formalisms. Technical Report 1998-14, Université de Caen, France, 1998.
5. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 2nd edition, 1999.
6. J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. *Journal of the ACM*, 49(6):716–752, 2002. Journal version of *ICDT'01* paper.
7. G. Gottlob and C. Koch. Monadic datalog and the expressive power of web information extraction languages. *Journal of the ACM*, 51(1):74–113, 2004. Journal version of *PODS'02* paper.
8. E. Grandjean. Invariance properties of RAMs and linear time. *Computational Complexity*, 4:62–106, 1994.
9. E. Grandjean. Linear time algorithms and NP-complete problems. *SIAM Journal on Computing*, 23(3):573–597, 1994.
10. E. Grandjean. Sorting, linear time, and the satisfiability problem. *Annals of Mathematics and Artificial Intelligence*, 16:183–236, 1996.
11. E. Grandjean and F. Olive. Monadic logical definability of nondeterministic linear time. *Computational Complexity*, 7(1):54–97, 1998. Journal version of *CSL'94* paper.
12. E. Grandjean and T. Schwentick. Machine-independent characterizations and complete problems for deterministic linear time. *SIAM Journal on Computing*, 32(1):196–230, 2002.
13. M. Grohe. *The structure of fixed-point logics*. PhD thesis, Albert-Ludwigs Universität Freiburg, Germany, 1994.
14. 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, U.K., 2003. Full version of *CSL'03* paper.
15. M. Grohe and T. Schwentick. Locality of order-invariant first-order formulas. *ACM Transactions on Computational Logic*, 1:112–130, 2000. Journal version of *MFCS'98* paper.
16. N. Immerman. *Descriptive Complexity*. Springer, 1999.
17. R.M. Karp. Reducibility among combinatorial problems. In *IBM Symposium 1972, Complexity of Computers Computations*. Plenum Press, New York, 1972.
18. S. Lindell. Linear-time algorithms for monadic logic. Short presentation at the 18th IEEE Symposium on Logic in Computer Science (LICS'03), 2003.
19. J. F. Lynch. Complexity classes and theories of finite models. *Mathematical Systems Theory*, 15:127–144, 1982.
20. O. Matz, N. Schweikardt, and W. Thomas. The monadic quantifier alternation hierarchy over grids and graphs. *Information and Computation*, 179(2):356–383, 2002.
21. M. More and F. Olive. Rudimentary languages and second-order logic. Technical Report 1996-1, Université de Caen, France, 1996.
22. B. Rossman. Successor-invariance in the finite. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS'03)*, 2003.
23. N. Schweikardt. On the expressive power of monadic least fixed point logic. Full version of *ICALP'04* paper. Available at <http://www.informatik.hu-berlin.de/~schweika/publications.html>.
24. T. Schwentick. Descriptive complexity, lower bounds and linear time. In *Proceedings of the 12th International Workshop on Computer Science Logic (CSL'98)*, volume 1584 of *Lecture Notes in Computer Science*, pages 9–28. Springer, 1998. Invited paper.
25. W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer, 1996.