Lindström Quantifiers and Leaf Language Definability

Hans-Jörg Burtschick
Fachbereich Informatik
TU–Berlin
Franklinstr. 28/29
D-10587 Berlin

Heribert Vollmer
Theoretische Informatik
Universität Würzburg
Am Exerzierplatz 3
D-97072 Würzburg

Abstract

We show that examinations of the expressive power of logical formulae enriched by Lindström quantifiers over ordered finite structures have a well-studied complexity-theoretic counterpart: the leaf language approach to define complexity classes. Model classes of formulae with Lindström quantifiers are nothing else than leaf language definable sets. Along the way we tighten the best up to now known leaf language characterization of the classes of the polynomial time hierarchy and give a new model-theoretic characterization of PSPACE.

1 Introduction

Initiated by Fagin's seminal paper [Fag74] characterizing NP as the class of generalized spectra, there has been a long line of research in characterizing complexity classes using notions from finite model theory (see the recent textbook [EF95]). In "classical" complexity theory problems are classified by defining restrictions on different computational resources as e.g. space or time. Using in contrast to this the syntactic complexity of a defining formula it is possible to measure the *descriptive* complexity of problems. By considering e.g. the complexity of quantifier prefixes of prenex formulae it is possible to investigate the descriptive complexity in a unified way. Besides the well-known existential and universal quantifiers recently the expressive power of a certain kind of generalized quantifiers, the so called *Lindström quantifiers*, has been examined (see [Lin66, Ste92, Got95] or Chapter 10 of the textbook [EF95]).

In the field of computational complexity characterizations of complexity classes by so called *leaf languages* have been studied intensively. This approach was introduced by Bovet, Crescenzi, and Silvestri in 1992 [BCS92] as a unified approach to define complexity classes. Consider a polynomial-time nondeterministic Turing machine M that prints a value on every path of its computation on some given input x. By imposing an order upon the machine's nondeterministic choices (e.g. based on the way the Turing program is presented) there is a well-defined sequence of output symbols over all of M's paths, which are ordered based on the order of nondeterministic choices. We call this sequence the *leaf string* of M on input x. Given now a (leaf) language B, a complexity class $\mathrm{Leaf}^{\mathrm{P}}(B)$ is defined as follows: A set L is in $\mathrm{Leaf}^{\mathrm{P}}(B)$ if and only if there is a nondeterministic Turing machine which on input x produces a leaf string which is in B if and only if $x \in L$.

Starting with the somewhat surprising result that the class PSPACE can already be characterized in this way using only regular leaf languages [HLS$^+$93], a lot of characterizations were given (see e.g. [JMT94, HVW95, HVW, CMTV95]), leading to a number of remarkable and unexpected normal forms for such classes as PSPACE, PP, and $\#\mathrm{P}$; and even circuit separations could be proved using leaf languages as a technical vehicle.

It turns out, as we show here, that both concepts Lindström quantifiers and leaf language definability are closely related: Leaf language defined complexity classes can be characterized by second-order Lindström quantifiers that are followed by first-order formulas, and vice versa for every formula of this form there is a leaf language and a nondeterministic polynomially time-bounded Turing machine that define the corresponding model class. Thus Lindström quantifiers and leaf language definability are two points of view of essentially the same concept. Relying on both views one might hope to get new insights into the expressive power of Lindström quantifiers on ordered finite structures and properties of leaf language defined complexity classes.

We give some results in this direction in this paper: The polynomial time hierarchy introduced by Stockmeyer in 1973 [Sto73] is one of the central concepts in complexity theory. Thus of course there have been efforts to give leaf language characterizations of this hierarchy. In [HLS$^+$93] it was shown that using AC^0 as leaf language class, one obtains the union of all classes of the polynomial hierarchy, and using the k-th level of the Brzozowski- (or dot-depth-) hierarchy [Str94] as leaf language class, the boolean closure of the class Σ_k^{p} is obtained. In [JMT94, HVW] it was proved that $\mathrm{Leaf}^{\mathrm{P}}(\Sigma_k\text{-LOGTIME}) = \Sigma_k^{\mathrm{p}}$. Using first-order definable leaf languages we prove a tighter connection: We show that the k-th level from the polynomial time hierarchy can be characterized using leaf languages which can be defined using first-order Σ_k formulae. As another example of how combination of model-theoretic and complexity theoretic arguments can lead to new results, we show how to use a theorem from [HLS$^+$93] to derive a new model-theoretic characterization of PSPACE.

These results are special cases of a general theorem which states that if a leaf language is defined by a formula in prenex normal form starting with one Lindström quantifier and followed by a sequence of universal and existential quantifiers, then the characterized complexity class can be defined in a model-theoretic way by a second-order formula with exactly the same quantifier structure. Thus we see that using methods from finite model theory we are able to tighten existing complexity theoretic results. We think that generally it is a good idea to specify leaf languages in a descriptive way. Leaf language definability was invented to separate the computational aspect in the definition of a class from the way the computation is evaluated. This evaluation scheme should therefore be given in a way free from computational aspects whatsoever. This goal is achieved using finite model theory to specify leaf languages.

2 Preliminaries

The definability of complexity classes via leaf languages was introduced in [BCS92, HLS$^+$93] (see also the recent textbook [Pap94]). Let M be a polynomial time-bounded nondeterministic Turing machine. Given an input x, such a machine produces on every path a symbol from some finite alphabet Γ. Let $\beta_M(x)$ be the string of the so produced symbols (based on the natural order of paths of the machine). Then for $B \subseteq \Gamma^*$, we define $\mathrm{Leaf}^M(B) =_{\mathrm{def}} \{x \mid \beta_M(x) \in B\}$. The class $\mathrm{Leaf}^{\mathrm{P}}(B)$ is the class of all languages $\mathrm{Leaf}^M(B)$ for all nondeterministic polynomial-time machines M. Here, B is the so called *leaf language* defining the class $\mathrm{Leaf}^{\mathrm{P}}(B)$. If $\mathcal{C}$ is a class of languages, then we define $\mathrm{Leaf}^{\mathrm{P}}(\mathcal{C}) =_{\mathrm{def}} \bigcup_{B\in\mathcal{C}} \mathrm{Leaf}^{\mathrm{P}}(B)$.

As it turned out (see [JMT94, HVW]), it sometimes plays a crucial role whether in the above definition we allow arbitrary Turing machines or we require that the computation trees produced have a special shape (e.g. are balanced). As a matter of fact, we will see that this distinction is meaningless in our context. It turns out that for those subclasses $\mathcal{L}$ of regular languages we are going to consider, we can require without loss of generality that all computation trees of Turing machines are *full binary trees*, see [Her95]. The reason is that all those subclasses $\mathcal{L}$ of regular languages which are of interest for the present paper are closed under padding, i.e., if $A \in \mathcal{L}$, $A \subseteq \Gamma^*$ for some alphabet Γ, then there is a letter $\mathbf{1} \notin \Gamma$ and a language $A' \in \mathcal{L}$, $A' \subseteq (\Gamma \cup \{\mathbf{1}\})^*$ such that for every word $w \in \Gamma^*$, we have $w \in A$ if and only if $w \in A'$ if and only if $\mathrm{pad}_{\mathbf{1}}(w) \subseteq A'$, where $\mathrm{pad}_{\mathbf{1}}(w)$ is the regular language $\mathbf{1}^*a_1\mathbf{1}^*a_2\mathbf{1}^*\cdots\mathbf{1}^*a_n\mathbf{1}^*$ for $w = a_1a_2\cdots a_n$ (see [JMT94]). Therefore, when we use the above defined notation $\mathrm{Leaf}^{\mathrm{P}}(\cdot)$ we implicitly assume that we only deal with full binary trees.

In this paper, we consider classes that are characterized via first-order definable leaf languages. Especially, we deal with the following classes of formulae:

A Σ_k-formula (Π_k-formula) is a formula in prenex normal form with k consecutive blocks of quantifiers, where in each block all quantifiers are of the same type, adjacent blocks contain quantifiers of opposite type, and the first block is of existential type (universal type resp.). Let Σ_k-FO (Π_k-FO, Σ_k-SO, and Π_k-SO) be the set of first-order Σ_k-formulae (first-order Π_k-formulae, second-order Σ_k-formulae, and second-order Π_k-formulae resp.).

For a formula ϕ we denote the class of finite ordered structures that are models of ϕ by $\mathrm{Mod}(\phi)$. As abbreviations, we use $\Sigma_k^0 = \mathrm{Mod}(\Sigma_k\text{-FO})$, $\Pi_k^0 = \mathrm{Mod}(\Pi_k\text{-FO})$, $\Sigma_k^1 = \mathrm{Mod}(\Sigma_k\text{-SO})$, and $\Pi_k^1 = \mathrm{Mod}(\Pi_k\text{-SO})$.

Since we are interested in characterizations of complexity classes, we consider in this paper only *ordered finite structures*, which are structures over signatures that do not contain constant or function symbols but a symbol $<$ for a linear order on the universe. We do not use the successor relation.

We use an encoding e of strings into ordered finite structures defined in the usual way. Let A be a finite set of symbols and $l = \lceil \log(|A|) \rceil$. The images of e are ordered finite structures over the signature $\sigma_l = (P_1, \ldots, P_l, <)$ with unary predicates $P_1, \ldots, P_l$. For a fixed enumeration of the symbols in A, let a_i be the i-th symbol in A. If the x-th element in the string $w \in A^*$ is a_i, then $P_j(x) = m$ in $e(w)$, iff the j-th bit in the binary encoding of i is m, for all $1 \leq j \leq l$ and $m \in \{0, 1\}$. Note, that e

is one-one and if $|A|$ is a power of two, the e is an isomorphism. Since the latter case sometimes plays a special role, we denote a finite set of symbols by A_ℓ, iff $|A_\ell| = 2^\ell$. If L is a language, then we write $e(L)$ for $\{ e(w) \mid w \in L \}$. If $\mathcal{C}$ is a class of languages, then $e(\mathcal{C}) =_{\text{def}} \{ e(L) \mid L \in \mathcal{C} \}$.

Furthermore we sometimes have to encode finite ordered structures into strings. Since e is one-one, e^{-1} is a function, too. We define an encoding f from finite ordered structures into strings to be e^{-1} but additionally extended to map relations of arity greater than one into strings. Analogously to the above we write $f(\mathrm{Mod}(\phi))$ for the set $\{ f(\mathcal{A}) \mid \mathcal{A} \models \phi \}$ and $f(\Sigma^0_k)$ ($f(\Pi^0_k)$, $f(\Sigma^1_k)$, and $f(\Pi^1_k)$) for the classes that are characterized by the corresponding logics.

Sometimes we do not mention the encodings e and f explicitly if the meaning will be clear from the context. That is, instead of writing $e(L) = \mathrm{Mod}(\phi)$, $\mathrm{Leaf}^{\mathrm{P}}(f(\Sigma^0_k))$ or $e(\mathrm{NP}) \subseteq \Sigma^1_1$, we simply write $L = \mathrm{Mod}(\phi)$, $\mathrm{Leaf}^{\mathrm{P}}(\Sigma^0_k)$, and $\mathrm{NP} \subseteq \Sigma^1_1$, having in mind that we compare objects of different types.

In a seminal paper Fagin [Fag74] connected the complexity of computations with the syntactic complexity of logical languages. He showed the following: Let M be a nondeterministic polynomial time Turing machine that recognizes a language over some alphabet A. Following the above discussion we assume that on every input, M's computation tree is a full binary tree. Fagin constructed an existential second-order formula Φ_M that describes the computation of M. The formula Φ_M is of the form $\exists\vec{X}\varphi_M(\vec{X})$ where φ_M is a first-order formula that has no free first-order variables and that additionally contains the free second-order variables $\vec{X}$. The arities and the number of the second-order variables are determined by the construction, see [Fag74].

Proposition 2.1 ([Fag74]) $e(\mathrm{NP}) \subseteq \Sigma^1_1$ *and* $\mathrm{NP} \subseteq f(\Sigma^1_1)$.

Let us consider a nondeterministic Turing machine that is n^k time-bounded and that produces a full binary computation tree for all inputs. It is possible to represent a path in a computation tree of depth n^k using a string $\pi \in \{0,1\}^{n^k}$. We enumerate the paths $(\pi_i)_{1 \le i \le 2^{n^k}}$ using the lexicographical ordering on $\{0,1\}^{n^k}$. Analogously, we can enumerate the assignments $(X_i^{\mathcal{A}})_{1 \le i \le 2^{n^k}}$ of a k-ary second-order variable X using the lexicographical ordering on characteristic sequences. In this paper we want to use a direct correspondence between the paths π_i in the computation tree and the assignments $\vec{X}_i^{\mathcal{A}}$ of $\vec{X}$:

Proposition 2.2 *Let M be a n^k time-bounded NTM as above. There exists a Σ_1-SO formula of the form $\exists X \exists \vec{Y} \varphi_M(X, \vec{Y})$, such that for all inputs w of length n and for all $1 \le i \le 2^{n^k}$, we have that M accepts w on path π_i iff $(e(w), X_i^{e(w)}) \models \exists\vec{Y}\varphi_M(\vec{Y})$.*

Proof. We slightly modify the construction in [Fag74] by introducing a k-ary second-order predicate $X^{(k)}$ such that its assignment represents the nondeterministic choice for each step of the computation. That is, for each assignment $X^{\mathcal{A}}$ the assignments $Y_1^{\mathcal{A}}, Y_2^{\mathcal{A}}, \ldots, Y_r^{\mathcal{A}}$ of $\vec{Y}$ are uniquely determined. ❑

3 Results

In order to extend the limited capabilities of first-order predicate logic it has been tried to enhance its expressibility power by adding different sorts of operators or quantifiers. The in our opinion most systematic and formally most elegant approach to this is to consider so called *Lindström quantifiers*.

Consider the classical first-order existential quantifier applied to some quantifier-free formula ψ with free variable x, i.e., consider the formula $\Psi \equiv \exists x \psi(x)$. Given an ordered structure $\mathcal{A}$, we can associate a binary (i.e., 0-1) sequence a_ψ with ψ by evaluating ψ for every possible value of x from $U^{\mathcal{A}}$ and then adding 0 for false and 1 for true to a_ψ. To be more formal: If n is assigned to x then $a_\psi(n) = 1$ iff $\psi(x)$ evaluates to true. The formula Ψ evaluates to true in $\mathcal{A}$ if the above defined sequence a_ψ is such that it has at least one position with the value 1. It is immediate to give a condition for sequences corresponding to a universal quantifier (all positions must be 1), or for the $\exists!$ quantifier (exactly one position must be 1), or for the modular quantifiers $\exists_{\equiv k}$ (the number of 1 positions must be equivalent to 0 mod k).

Thus, it is very natural to define generalized quantifiers by considering arbitrary conditions on binary sequences (which we will call logical acceptance types). Let us give a formal definition.

Let τ be a set of s-tuples of finite binary sequences, i.e., τ consists of tuples $(a_1, \dots, a_s)$ where for every i ($1 \leq i \leq s$), a_i is a mapping from $\{1, \dots, k\}$ to $\{0, 1\}$ for some k. We call such a τ a *logical acceptance type*. The set of all s-tuples of finite binary sequences will in the following be denoted by $\boldsymbol{\tau}(s)$.

Then we denote the Lindström quantifier given by τ by Q_τ. By $Q_\tau\Sigma_k$-FO we denote the set of formulae built as follows: If $\psi_1, \dots, \psi_s$ are Σ_k-FO formulae, each over r free variables $\vec{x} = (x_1, \dots, x_k)$, then $Q_\tau \vec{x}\, [\psi_1(\vec{x}), \dots, \psi_s(\vec{x})]$ is a $Q_\tau\Sigma_k$-FO formula. The semantics of such a formula is defined as follows: Let $\mathcal{A}$ be a finite structure over the corresponding signature. Then $\mathcal{A} \models Q_\tau \vec{x}\, [\psi_1(\vec{x}), \dots, \psi_s(\vec{x})]$ if the tuple $(a_1, \dots, a_s)$ is in τ, where the sequences a_i are defined as follows: For $1 \leq n \leq |U^{\mathcal{A}}|^r$, $a_i(n) = 1$ if and only if $\mathcal{A} \models \psi_i(\vec{x})$ where n is the rank of $\vec{x}$ on the order of r-tuples over $\mathcal{A}$ ($1 \leq i \leq s$). $Q_\tau\Pi_k$-FO is defined analogously. We write $Q^0_\tau\Sigma^0_k$ for $\mathrm{Mod}(Q_\tau\Sigma_k\text{-FO})$ and $Q^0_\tau\Pi^0_k$ for $\mathrm{Mod}(Q_\tau\Pi_k\text{-FO})$.

Given a Lindström quantifier Q_τ, define $\mathbf{Q}^+_\tau$ to be the set of first-order formulae in prenex normal form that starts with one quantifier Q_τ followed by arbitrary first-order formulae (this notation is from [Got95]). Observe that for every $\phi \in \mathbf{Q}^+_\tau$ there is a $k \in \mathbb{N}$ such that ϕ is logically equivalent to a $Q_\tau\Sigma_k$-FO formula.

Considering the set of sequences where at least one value is 1 (all values are 1, exactly one value is 1, the number of 1 values is equivalent to 0 module some k, resp.), we see that the usual quantifiers mentioned above are special cases of the just given definition. Similarly, τ can encode any property, say of a graph-theoretic nature, to capture e.g. the transitive closure operator or the Hamiltonian path operator (see [Imm87, Ste92]).

We remark that our definition resembles very much the definition of *group quantifiers* from [BIS90]. Our definition slightly differs w. r. t. two details from the one given in [EF95, Got95]: First, we only consider Lindström quantifiers defined by ordered structures. Since we are interested in characterizations of complexity classes this is no actual restriction. Second, in the definition from [EF95] the formulae ψ_i may differ in the number of free variables. However, for every Lindström quantifier in the sense of [EF95] it's possible to construct an equivalent Q_τ. To get an overview of what is known about predicate logic with generalized quantifiers, see the excellent report [Got95] or Chapter 10 of the recent textbook [EF95].

Addressing again the group quantifiers from [BIS90] we see that it is possible to view logical acceptance types as languages: Consider for concreteness the group S_5 of all permutations on five positions. S_5 consists of 120 elements. Thus the elements of S_5 can be encoded in binary by 7-bit sequences. Let now τ_{S_5} be the following set of 7-tuples of binary sequences: If $(a_1, \dots, a_7)$ is such a tuple with every sequence defined on $\{1, \dots, m\}$ (for some $m \in \mathbb{N}$), then for every i, $1 \leq i \leq m$, the 7-bit string $a_1(i)a_2(i) \cdots a_7(i)$ encodes an element from S_5 which we denote by $A(i)$. Now $(a_1, \dots, a_7) \in \tau_{S_5}$ if and only if $A(1) \circ A(2) \circ \cdots \circ A(m) = ()$, where $\circ$ denotes multiplication is S_5 and $()$ is the identity permutation. Let Q_{S_5} be the quantifier defined by τ_{S_5}. Then Q_{S_5} is a group quantifier in the sense of [BIS90].

The way we encoded the word problem for S_5 by a suitable $\tau = \tau_{S_5}$ can of course be considered generally: If $\tau \in \boldsymbol{\tau}(s)$ is a set of s-tuples of binary sequences, then every such tuple with sequences defined over $\{1, \dots, m\}$ defines a word over A_s (defined in the Preliminaries) of length m. Thus, every logical acceptance type $\tau \in \boldsymbol{\tau}(s)$ is essentially a language over A_s.

To characterize the power of leaf languages defined with $\mathbf{Q}_\tau^+$ formulae, we define second-order Lindström quantifiers. Our definition will be inspired by an intuition similar to the one for the first-order case given above: Given a second-order formula $\Phi \equiv \exists R \phi(R)$, we can again construct a binary sequence a_ϕ with one bit for every possible assignment of R in a given input structure $\mathcal{A}$. Φ evaluates to *true* in $\mathcal{A}$ if there is at least one 1 in a_ϕ. By allowing general conditions on such sequences we define second-order Lindström quantifiers:

Let $\tau \in \boldsymbol{\tau}(s)$ be a logical acceptance type as above. By $Q_\tau\Sigma_k$-SO we denote the set of formulas built as follows: If $\phi_1, \dots, \phi_s$ are Σ_k-SO formulas, each over q free predicates $\vec{R} = R_1, R_2, \dots, R_r$, then $Q_\tau \vec{R}\left[\phi_1(\vec{R}), \dots, \phi_s(\vec{R})\right]$ is a $Q_\tau\Sigma_k$-SO formula. The semantics of such a formula is defined as follows: Let r be the sum of the arities of all predicate symbols in $\vec{R}$. Then we can identify one possible assignment of $\vec{R}$ over a set $U^{\mathcal{A}}$ with its characteristic sequence $c_{\vec{R}}$ which is a binary string of length $|U^{\mathcal{A}}|^r$. Based on the lexicographic ordering of these strings, we define an ordering on assignments of $\vec{R}$. Let now $\mathcal{A}$ be a finite structure over the corresponding signature. Then $\mathcal{A} \models Q_\tau \vec{R}\left[\phi_1(\vec{R}), \dots, \phi_r(\vec{R})\right]$ if the tuple $(a_1, \dots, a_s)$ is in τ, where the sequences a_i are defined as follows: For $1 \leq n \leq 2^{|U^{\mathcal{A}}|^r}$, $a_i(n) = 1$ if and only if $\mathcal{A} \models \phi_i(\vec{R})$ where n is the rank of $\vec{R}$ in the above-sketched order of assignments of $\vec{R}$ ($1 \leq i \leq s$). Analogously to the first order case, we also define $Q_\tau\Pi_k$-SO, $\mathbf{Q}_\tau^+$-SO, $Q_\tau^1\Sigma_k^1$, and $Q_\tau^1\Pi_k^1$. We use Q_τ-FO and Q_τ-SO as abbreviations for $Q_\tau\Sigma_0$-FO and $Q_\tau\Sigma_0$-SO, resp.

The definition of second-order generalized quantifiers yields the well-known special cases: If we think of existential and universal quantifiers as special first-order Lindström quantifiers, then the corresponding second-order Lindström quantifiers are exactly the familiar second-order existential and universal quantifiers.

Following the proof of Fagin [Fag74] there is a correspondence between leaf language definability and second-order Lindström quantifiers. Recall the formula φ_M from Proposition 2.2.

Proposition 3.1 *Let M be a nondeterministic polynomial time machine. Then we have* $\mathrm{Leaf}^M(\tau) = \mathrm{Mod}(Q_\tau X \left[\exists \vec{Y} \varphi_M(X, \vec{Y})\right])$.

Our next theorem generalizes this connection. As it turns out, in order for our generalization to hold the Lindström quantifier must have a particular property, which can best be visualized in terms of the acceptance type τ. We say that $\tau \in \boldsymbol{\tau}(s)$ is closed under padding if there is a symbol $\mathbf{1} \in A_s$ such that for every word $w \in A_s$ we have that $w \in \tau$ if and only if $\mathrm{pad}_{\mathbf{1}}(w) \subseteq A_s$.

Theorem 3.2 *Let τ be a logical acceptance type which is closed under padding. Then*

$$\mathrm{Leaf}^{\mathrm{P}}(\mathbf{Q}_\tau^+) = \mathrm{Mod}(\mathbf{Q}_\tau^+\text{-SO}).$$

The proof of this theorem essentially requires two constructions. Those constructions reflect the definition of $\mathbf{Q}_\tau^+$. In the first part (Theorem 3.3 below) we address quantifier prefixes consisting only of a Lindström quantifier, and in the second part (Theorem 3.4) we address Σ_k prefixes. The proof of Theorem 3.2 then is a combination of these two proofs.

Theorem 3.3 *Let τ be a logical acceptance type which is closed under padding. Then we have that* $\mathrm{Leaf}^{\mathrm{P}}(Q_\tau\text{-FO}) = \mathrm{Mod}(Q_\tau\text{-SO})$.

Theorem 3.4 $\mathrm{Leaf}^{\mathrm{P}}(\Sigma_k^0) = \mathrm{Mod}(\Sigma_k\text{-SO})$.

Theorem 3.4 follows immediately from the following two lemmas:

Lemma 3.5 $\mathrm{Leaf}^{\mathrm{P}}(\Sigma_k^0) \subseteq \Sigma_k^1$ *for all* $k \geq 1$.

Proof. We construct for a given Σ_k-FO-formula ϕ_{FO} and for a polynomially time-bounded nondeterministic Turing machine M a Σ_k-SO formula ϕ_{SO} and then show for all $w \in \Sigma^*$ that $e(w) \models \phi_{SO}$, iff M accepts w with the leaf language that is defined by ϕ_{FO}.

Let M be a polynomially time-bounded nondeterministic Turing machine such that the computation tree of M is a full binary tree for all inputs. Let us first consider the case, that Γ, the set of symbols that M produces on every computation-path (see Section 2), has two elements. We will discuss the other case at the end of construction.

Let φ_M be the Σ_1-SO-formula which formalizes the computation of M as described in the preliminaries. φ_M contains the second-order variables X and $\vec{Y}$. An assignment of X represents the nondeterministic choices and for a fixed assignment of X there exists exactly one assignment of $\vec{Y}$ that represents a computation path of M. From $\varphi_M(X,\vec{Y})$ we can obtain a formula $\varphi_M^{\text{path}}(X,\vec{Y})$ that for all suitable finite ordered structures $\mathcal{A}$ is true on $X^{\mathcal{A}},\vec{Y}^{\mathcal{A}}$, iff $\vec{Y}^{\mathcal{A}}$ represents a correct computation path of M w. r. t. the nondeterministic choices given by $X^{\mathcal{A}}$.

Let ϕ_{FO} be a first-order formula of the form $\exists\vec{x}_1\forall\vec{x}_2\dots Q_k\vec{x_k}\varphi_{FO}(\vec{x}_1,\vec{x}_2,\dots,\vec{x}_k)$ over $(P^{(1)},<)$. We construct a Σ_k^1 formula ϕ_{SO} by replacing the variables and atomic predicates in φ_{FO} in the following way:

$[\exists x_i]$ We replace all first-order variables x_i in φ_{FO} by tuples of second-order variables $(X_i,\vec{Y_i})$ and thus transform the existential first-order quantifier that bounds x_i into a sequence of second-order quantifiers that bound X_i and $\vec{Y_i}$. To ensure that an assignment of $\vec{Y}$ represents a correct computation path, we replace $\exists x_i\varphi$ by $\exists X_i,\vec{Y_i}\left(\varphi_{M,i}^{\text{path}}(X_i,\vec{Y_i})\wedge\varphi\right)$. In order to obtain a formula in prenex normalform, the formulae $\varphi_{M,i}^{\text{path}}(X_i,\vec{Y_i})$ and $\varphi_{M,j}^{\text{path}}(X_j,\vec{Y_j})$ have no variables in common, for all $i\neq j$.

$[\forall x_i]$ We replace the universal quantifier $\forall x_i:\varphi$ by $\forall X_i,\vec{Y_i}\left(\varphi_{M,i}^{\text{path}}(X_i,\vec{Y_i})\rightarrow\varphi\right)$.

$[P(x_i)]$ We replace each occurrence of $P(x_i)$ by the Σ_k^1-formula $\varphi_{M,i}(X_i,\vec{Y_i})$ which we obtain by renaming the first-order variables in $\varphi_M(X_i\vec{Y_i})$ such that for all $i\neq j$ the formulas $\varphi_{M,i}(X_i,\vec{Y_i})$ and $\varphi_{M,j}(X_j\vec{Y_j})$ have no variables in common.

$[(x_i<x_j)]$ We have to extend the ordering over first-order variables to an ordering over the tuples $X,\vec{Y}$ of second-order variables. This can be achieved by well-known techniques.

For $|\Gamma|>2$, we have to consider first-order formulae over signatures that have $m=\lceil\log|\Gamma|\rceil$ unary predicates. As described in Section 2, it is possible to encode the symbols of Γ in binary. For a fixed encoding it is possible to construct m formulae $\varphi_{M,i}^1(X,\vec{Y_i})\dots\varphi_{M,i}^m(X_i,\vec{Y_i})$ that realize this encoding.

To complete the proof, we have to show for all $w\in\Sigma^*$:

$$w\in\text{Leaf}^{\text{P}}(e^{-1}(\text{Mod}(\phi_{FO}))),\text{ iff } e(w)\in\text{Mod}(\phi_{SO})$$

Let $\mathcal{M}(w)=(U^{\mathcal{M}(w)},R_1^{\mathcal{M}(w)},R_r^{\mathcal{M}(w)},\prec^{\mathcal{M}(w)})$ be the following structure. $U^{\mathcal{M}(w)}$ is the set of assignments of X in $e(w)$, and $R_i^{\mathcal{M}(w)}$ is the set of those assignments $X^{e(w)}$ such that there exist assignments $\vec{Y}^{e(w)}$ and φ_M^i evaluates to true on $(e(w),X^{e(w)},\vec{Y}^{e(w)})$. $\prec^{\mathcal{M}(w)}$ is the above described lexicographical ordering on $U^{\mathcal{M}(w)}$. It follows from Proposition 2.2 that $e(\beta_M(w))$ and $\mathcal{M}(w)$ are isomorphic.

From the construction of ϕ_{SO} it follows that $\mathcal{M}(w)\models\phi_{FO}$, iff $e(w)\models\phi_{SO}$. Therefore, we have that

$$\begin{aligned}w\in\text{Leaf}^{\text{P}}(e^{-1}(\text{Mod}(\phi_{FO}))) &\iff e(\beta_M(w))\models\phi_{FO}\\ &\iff \mathcal{M}(w)\models\phi_{FO}\\ &\iff e(w)\models\phi_{SO}.\end{aligned}$$

❑

Lemma 3.6 $\Sigma_k^1 \subseteq \mathrm{Leaf}^{\mathrm{P}}(\Sigma_k^0)$ *for all* $k \geq 1$.

Proof. Let ϕ_{SO} be a Σ_k-SO formula with m second-order variables. We construct a polynomially time-bounded NTM M and a Σ_k-FO formula ϕ_{FO} defining a leaf-language $e^{-1}(\mathrm{Mod}(\phi FO)) \subseteq \Gamma^*$, such that for all ordered finite structures $\mathcal{A}$ over the corresponding signature: $\mathcal{A} \in \mathrm{Mod}(\phi_{SO})$, iff $f(\mathcal{A}) \in \mathrm{Leaf}^{\mathrm{P}}(e^{-1}(\mathrm{Mod}(\phi_{FO})))$

On input $f(\mathcal{A})$ the NTM M determines nondeterministically the assignments of the second-order variables. That is, on each computation path, M writes the characteristic sequence of one possible assignment on its worktapes. Using these characteristic sequences, M evaluates the first-order part of ϕ_{SO}.

For Σ_1-SO formulas, the corresponding leaf language is defined by $\exists x P(x)$, and for Π_1-SO formulas the corresponding leaf language is defined by $\forall x P(x)$.

For $k > 1$, we have to mark for all second-order variables S_i $(1 \leq i \leq m)$ the paths that share the same assignments $S_i^{\mathcal{A}}$ in the following way: We define Γ to contain the symbols F, T, and i for all $i \in \{1, 2, \ldots, m\}$. For all $1 \leq i \leq m$ M produces nondeterministically the characteristic sequence of an assignment of S_i and an extra bit b_i.
(-) If the assignment is empty (the leftmost path) and b_i is 0, then M produces on all subsequent branches the symbol i.
(-) If the assignment is the full relation (the rightmost path) and b_i is 1, then M produces on all subsequent branches the symbol i.
(-) In all other cases, the bit b_i is ignored and M continues as described above. If the evaluation yields true, then M produces T, and F otherwise. It is easy to see that, if there are two paths marked with i and all paths between are not marked with i, then all the pathes between have the same assignment of S_{i-1} in common, for $1 < i \leq m$.

As described in the preliminaries, the symbols of Γ ($|\Gamma| = m + 2$) can be encoded using $\lceil \log(m + 2) \rceil$ unary predicates. By boolean combinations of these, we construct formulae that express that a path produces F, T, or i, for $1 \leq i \leq m$. We abbreviate these formulae by P_F, P_T, and P_i resp. The fomula $\neg(P_F(x) \vee P_T(x))$ we abbreviate by $P_{\mathrm{marked}}(x)$.

For $1 \leq i < m$ let $\mathrm{Region}_i(x, y)$ be an abbreviation for the formula

$$P_i(x) \wedge P_i(y) \wedge \forall z_i \left(x \leq z_i \leq y \to \neg P_i(z_i)\right).$$

For $k > 1$ we construct inductively a sequence of formulae $\phi_k, \phi_{k-1}, \ldots, \phi_1$. Then the desired formula ϕ_{FO} will be ϕ_1. Let $\phi_k(x_{\mathrm{left}}, x_{\mathrm{right}})$ be the formula

$$\exists x_k \Big((x_{\mathrm{left}} \leq x_k \leq x_{\mathrm{right}} \wedge \neg P_{\mathrm{marked}}(x_k) \wedge P_T(x_k) \Big), \text{ or}$$

$$\forall x_k \Big((x_{\mathrm{left}} \leq x_k \leq x_{\mathrm{right}} \wedge \neg P_{\mathrm{marked}}(x_k)) \to P_T(x_k) \Big),$$

depending on the rightmost quantifier. For the i-th alternation $(1 < i < k)$, let us first consider the case, that $S_{j_1}, \ldots, S_{j_2}$ are existentially quantified and S_{j_1-1} and S_{j_2+1} are universally quantified. We define $\phi_i(x_{\mathrm{left}}, x_{\mathrm{right}})$ to be the formula

$$\exists x_i, y_i \Big(\mathrm{Region}_{j_2}(x_i, y_i) \wedge (x_{\mathrm{left}} \leq x_i) \wedge (y_i \leq x_{\mathrm{right}}) \wedge \phi_{i+1}(x_i, y_i) \Big).$$

Transforming this into prenex normal form we get a quantifier prefix starting with $\exists x_i, y_i \forall z_{j_2} \forall x_{i+1}, y_{i+1} \ldots$.

Now consider the case that $S_{j_1}, \ldots, S_{j_2}$ are universally quantified and S_{j_1-1}, S_{j_2+1} are existentially quantified. We define $\phi_i(x_{\text{left}}, x_{\text{right}})$ to be the formula

$$\forall x_i, y_i \Big(\big(\text{Region}_{j_2}(x_i, y_i) \wedge (x_{\text{left}} \le x_i) \wedge (y_i \le x_{\text{right}})\big) \rightarrow \phi_{i+1}(x_i, y_i)\Big).$$

Transforming this into prenex normal form we get a quantifier prefix starting with $\forall x_i, y_i \exists z_{j_2} \exists x_{i+1}, y_{i+1} \ldots$.

Let S_{j+1} be the leftmost universally quantified second-order variable in ϕ_{SO}. We define $\phi_{FO} = \phi_1$ to be the formula $\exists x_1, y_1 \, \big(\text{Region}_j(x_i, y_i) \wedge \phi_{i+1}(x_i, y_i)\big)$. ❑

Proof. (of Theorem 3.3)
We use the ideas of the proofs of Lemmas 3.5 and 3.6. For the direction from left to right, we are given a polynomial time machine M and a leaf language which is defined by the Q_τ-FO formula $Q_\tau \vec{x}\, [\psi_1(\vec{x}), \ldots, \psi_s(\vec{x})]$. φ_M is the Fagin-formula describing the behaviour of M. φ_M has free relational variables X and $\vec{Y}$. Every first-order formula ψ_i ($1 \le i \le s$) is transformed into a second-order formula Ψ_i which is built from ψ_i as in the proof of Lemma 3.5 by replacing every first-order variable x_j by $(X_j, \vec{Y_j})$ and using the Fagin formula instead of the input predicates. However, we have to consider assignments for $\vec{Y}$ which do not encode valid computations. These assignments do not lead to symbols in the leaf word of machine M, but they appear in the sequence a_Ψ corresponding to Ψ. From a logical point of view, what we need here is the relativization (see [EFT94]) of the quantifier Q_τ by the formula φ_M^{path} which ensures that assignments of $\vec{Y}$ encode valid computation paths. This can be solved as follows: Modify Ψ such that if $\vec{Y}$ does not encode a valid computation path in M then the corresponding letter in a_Ψ is the letter $\mathbf{1}$ by which words in τ can be padded arbitrarily.

For the direction from right to left, suppose we are given a Q_τ-SO formula $Q_\tau \vec{R} \left[\phi_1(\vec{R}), \ldots, \phi_s(\vec{R})\right]$. Construct a Turing machine M which branches by guessing assignments to the second-order variables and evaluates the first-order part in a straightforward manner. Thus this machine can produce a leaf word which is exactly the word corresponding to the sequences $a_{\phi_1}, \ldots, a_{\phi_s}$. If we now take as leaf language $\tau \subseteq A_s$, then we see that M accepts exactly models of the given Q_τ-SO formula. The leaf language can trivially be defined by the Q_τ-FO formula $Q_\tau x\, (P_1(x), \ldots, P_s(x))$. ❑

Proof. (of Theorem 3.2)
We show how to combine Theorem 3.4 and Theorem 3.3 to prove Theorem 3.2.

To show $e(\text{Leaf}^{\text{P}}(Q_\tau^0 \Sigma_k^0)) \subseteq Q_\tau^1 \Sigma_k^1$, we combine the proofs of Lemma 3.5 and Theorem 3.3 in the following way: Given a $Q_\tau \Sigma_k$-FO formula ϕ of the form $Q_\tau \vec{x}\, [\varphi_1(\vec{x}), \ldots, \varphi_s(\vec{x})]$, we first transform the Σ_k-FO formulae $\varphi_1, \ldots, \varphi_s$ into Σ_k-SO formulae as in the proof of Lemma 3.5. The Lindström quantifier then is transformed as in the proof of Theorem 3.3.

For the opposite direction, we show $f(Q_\tau^1 \Pi_k^1) \subseteq \text{Leaf}^{\text{P}}(Q_\tau^0 \Pi_k^0)$. Let $\phi \equiv Q_\tau X\, [\varphi_1(X), \ldots, \varphi_s(X)]$ be a $Q_\tau \Pi_k$-SO formula and let the Turing machine M be constructed as in the proof of Lemma 3.6. Again, we have substrings of the leaf string corresponding to assignments of X. Conceptually, we can evaluate the first-order formulas that are constructed in the proof of Lemma 3.6 for each such substring. An input is accepted if this evaluation yields true or false according to τ. But we have to overcome the following technical difficulty: Every second-order variable bound by Q_τ is transformed into two first-order variables. These are intended to denote the left and right margin of a substring to be evaluated. The first-order formula that follows the Lindström quantifier is evaluated for all pairs of assignments of those variables. If such a pair does not denote the left and right margin of a substring (this is expressible using a universal quantifier) then the evaluation of the first-order formula has to encode the corresponding padding symbol in τ. That is, we again make use of the assumption that τ is closed under padding. The extra universal quantifier to check the margins can be added to the first block of universal quantifierst of the Π_k-SO formulae. ❑

We now state some corollaries of our main result. Observe the proof of Theorem 3.3 shows that on the leaf language level we don't need the full power of Q_τ-FO. Actually, just the leaf language τ itself is sufficient; thus we get the following very close correspondence between leaf languages and logical acceptance types:

Corollary 3.7 $\mathrm{Leaf}^{\mathrm{P}}(\tau) = \mathrm{Mod}(Q_\tau\text{-SO})$.

Another corollary of our main result is the following leaf language characterization of the classes of the polynomial time hierarchy:

Corollary 3.8 *1.* $\mathrm{Leaf}^{\mathrm{P}}(\Sigma_k^0) = \Sigma_k^1 = \Sigma_k^{\mathrm{p}}$.

2. $\mathrm{Leaf}^{\mathrm{P}}(\Pi_k^0) = \Pi_k^1 = \Pi_k^{\mathrm{p}}$.

Proof. Immediately by Lemmas 3.5 and 3.6, their duals for Π-classes, and Stockmeyer's model-theoretic characterization of the classes of the polynomial hierarchy [Sto73]. ❑

Since Σ_k-LOGTIME $\subsetneq \Sigma_k^0$, the just given result tightens the up to now known characterization $\mathrm{Leaf}^{\mathrm{P}}(\Sigma_k\text{-LOGTIME}) = \Sigma_k^{\mathrm{p}}$.

Finally, we want to address an interesting special case: the class PSPACE. Barrington, Immerman, and Straubing showed that first-order logic with group quantifiers defines exactly the regular languages [BIS90]. Hertrampf et al. [HLS$^+$93] who characterized PSPACE by regular leaf languages showed that in fact for this characterization already one single regular language, the word problem for the group S_5, is sufficient. Thus, this leaf language characterization yields the following:

Proposition 3.9 $\mathrm{PSPACE} = \mathrm{Mod}(Q_{S_5}\text{-SO})$.

4 Discussion

As we have seen, Lindström quantifiers which are a well studied logical concept have a complexity theoretic counterpart: the so called leaf language definability, which has been studied intensively in the recent past.

Second-order Lindström quantifiers define (in a model theoretic sense) exactly those languages characterizable by leaf languages for polynomial time machines. If Q_τ is a Lindström quantifier, then the logic Q_τ-SO defines the complexity class $\mathrm{Leaf}^{\mathrm{P}}(\tau)$. Thus it maybe possible that results about leaf languages contribute to the study of the expressive power of second-order Lindström quantifiers on ordered finite structures, and vice versa.

Of course, it would be nice to have a leaf language analogue for first-order Lindström quantifiers. To be able to do this one will have to consider "leaf languages for FO" instead of leaf languages for polynomial time. To be more precise, what is an appropriate restriction of the computation model producing leaf words?

Gottlob in [Got95] showed that under some particular assumptions to τ, first-order formulae with arbitrarily nested Q_τ and existential and universal quantifiers yield superclasses of L (the logspace decidable sets). Thus, in the context of leaf language characterizability, this results in superclasses of PSPACE [HLS$^+$93]. However, more detailed characterizations seem to be an interesting point for future research—of course also in the absence of the assumptions from [Got95].

Furthermore, one should consider leaf languages defined by second-order formulae to investigate the structure of complexity classes above exponential time, e.g. the exponential time hierarchy.

References

[BCS92] D. P. Bovet, P. Crescenzi, and R. Silvestri. A uniform approach to define complexity classes. *Theoretical Computer Science*, 104:263–283, 1992.

[BIS90] David A. Mix Barrington, Neil Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences,* 41:274–306, 1990.

[CMTV95] H. Caussinus, P. McKenzie, D. Thérien, and H. Vollmer. Nondeterministic NC^1 computation. Technical report, Université de Montréal, 1995.

[EF95] H. D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.

[EFT94] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer, 2nd edition, 1994.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial time recognizable sets. In R. Karp, editor, *Complexity of Computations*, pages 43–73, 1974.

[Got95] G. Gottlob. Relativized logspace and generalized quantifiers over finite structures. Technical Report CD-TR-95/76, Institut for Information Systems, Vienna University of Technology, 1995. An extended abstract appeared in the proceedings of the 10th Symposium on Logic in Computer Science, 1995.

[Her95] U. Hertrampf. Regular leaf-languages and (non-) regular tree shapes. Technical Report A-95-21, Institut für Mathematik und Informatik, Medizinische Universität zu Lübeck, 1995.

[HLS$^+$93] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. W. Wagner. On the power of polynomial time bit-reductions. In *8th Ann. Conf. Structure in Complexity Theory*, pages 200–207, 1993.

[HVW] U. Hertrampf, H. Vollmer, and K. W. Wagner. On balanced vs. unbalanced computation trees. *Mathematical Systems Theory*. To appear.

[HVW95] U. Hertrampf, H. Vollmer, and K. W. Wagner. On the power of number-theoretic operations with respect to counting. In *10th Ann. Conf. Structure in Complexity Theory*, pages 299–314, 1995.

[Imm87] N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16:760–778, 1987.

[JMT94] B. Jenner, P. McKenzie, and D. Thérien. Logspace and logtime leaf languages. In *9th Ann. Conf. Structure in Complexity Theory*, pages 242–254, 1994.

[Lin66] P. Lindström. First order predicate logic with generalized quantifiers. *Theoria*, 32:186–195, 1966.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Ste92] I. A. Stewart. Using the Hamilton path operator to capture NP. *Journal of Computer and System Sciences*, 45:127–151, 1992.

[Sto73] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1973.

[Str94] H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, 1994.