A Generalization of the Büchi-Elgot-Trakhtenbrot Theorem

Matthias Galota and Heribert Vollmer

Theoretische Informatik, Universität Würzburg,
Am Hubland, 97074 Würzburg, Germany.

Abstract. We consider the power of nondeterministic finite automata with generalized acceptance criteria and the corresponding logics. In particular, we examine the expressive power of monadic second-order logic enriched with monadic second-order generalized quantifiers for algebraic word-problems. Extending a well-known result by Büchi, Elgot, and Trakhtenbrot, we show that considering monoidal quantifiers, the obtained logic captures the class of regular languages. We also consider monadic second-order groupoidal quantifiers and show that these are powerful enough to define every language in LOGCFL.

1 Introduction

Nondeterministic finite automata with generalized acceptance criteria were introduced in [PV01]. Usually, an NFA M is said to accept its input word w if it has at least one accepting computation on w, or, in other words, if in the computation tree of $M(w)$ there is at least one accepting path. One might ask what class of languages one obtains when an input is defined to be accepted if another condition holds, e.g., the number of accepting paths is divisible by some prime number p. Even more generally, let us suppose that every path in the computation tree produces an output symbol (which can be 0, 1 or something else), and say that the *leafstring* of M on w is the sequence of these symbols over all paths read from left to right (in an order defined formally below in Sect. 2). A generalized acceptance criterion in the sense of [PV01] now is a leaf language A, i.e., a set of leafstrings. Automaton M with leaf language A *by definition* accepts an input w if the leafstring of $M(w)$ is an element of A. Clearly, the leaf language for usual nondeterministic acceptance is the language $L_{\exists}$ of all binary words with at least one occurrence of the letter "1", while that for modulo-p acceptance is the language $L_{\text{mod } p}$ of all words with a number of "1"s that is a multiple of p. Leaf languages drawn from various complexity and formal language classes were examined in [PV01], and the power of NFAs using these to define acceptance was clarified.

In this paper, we address the question, to which logical framework finite automata with leaf languages correspond. We show that a suitable way to characterize NFAs with leaf languages in a logical way is to consider monadic second-order logic enhanced with monadic second-order Lindström quantifiers (a.k.a. generalized quantifiers, see, e.g., [EF95, Chap. 10.1]). We prove the following general theorem: The languages accepted by finite automata with leaf language A are exactly those definable by formulas with one second-order Lindström quantifier given by A and no further second-order quantifiers. For example, the above condition “divisible by p” corresponds to a monadic modular counting quantifier.

Our result gives a comparison between automata and logic which we hope will prove useful in many contexts. Here, we apply our general theorem to obtain consequences in two directions.

First we consider monadic *monoidal* quantifiers. These are defined by monoid word-problems, i.e., regular languages. Thus, by our general theorem, these correspond to regular leaf languages. Using a result of Peichl and Vollmer [PV01] about the power of finite automata with regular leaf languages, we conclude that formulas with monadic second-order monoidal quantifiers cannot define non-regular languages.

Second, we consider monadic *groupoidal* quantifiers, that is, quantifiers given by groupoid word-problems. (A groupoid is a finite multiplication table with an identity, i.e., informally, a monoid without the associativity requirement.) Such word-problems are context-free languages; hence, by the above, groupoid quantifiers correspond to context-free leaf languages. Using a result of Lautemann et al. [LMSV01] giving a model-theoretic characterization of LOGCFL, we conclude that finite automata with context-free leaf language can accept every language in LOGCFL (the class of languages logspace-reducible to context-free languages; it is known that these are exactly those languages accepted by auxiliary pushdown-automata operating simultaneously in logarithmic space and polynomial time [Coo71, Sud78]). Hence, in this case the situation is different from the monoidal case: There, “regular quantifiers” could not define non-regular languages, while now “context-free quantifiers” can define a strict superclass of CFL.

2 Finite Automata and Leaf Languages

In this section we will define *finite automata with generalized acceptance criterion*, as introduced in [PV01].

The basic model we use is that of nondeterministic finite automata. On an input word w, such a device defines a tree of possible computations. We want to consider this tree, but with a natural order on the leaves. Therefore we make the following definition:

A *finite leaf automaton* is a tuple $M = (Q, \Sigma, \delta, s, \Gamma, \beta)$ where Q is the finite set of *states*, Σ is an alphabet, the *input alphabet*, $\delta\colon Q \times \Sigma \to Q^+$ is the *transition function*, $s \in Q$ is the *initial state*, Γ is an alphabet, the *leaf alphabet*, and $\beta\colon Q \to \Gamma$ is a function that associates a state q with its *value* $\beta(q)$. The sequence $\delta(q,a)$, for $q \in Q$ and $a \in \Sigma$, contains all possible successor states of M when reading letter a while in state q, and the order of letters in that sequence defines a *total order on these successor states*. This definition allows the same state to appear more than once as a successor in $\delta(q,a)$.

Let M be as above. The computation tree $T_M(w)$ of M on input w is a labeled directed rooted tree defined as follows:

– The root of $T_M(w)$ is labeled (s, w).
– Let v be a node in $T_M(w)$ labeled by (q, x), where $x \neq \epsilon$ (the empty word), $x = ay$ for $a \in \Sigma$, $y \in \Sigma^*$. Let $\delta(q, a) = q_1 q_2 \cdots q_k$. Then v has k children in $T_M(w)$, and these are labeled by $(q_1, y), (q_2, y), \ldots, (q_k, y)$ in this order.

If we look at the tree $T_M(w)$ and attach the symbol $\beta(q)$ to a leaf in this tree with label (q, ε), then leafstring$^M(w)$ is defined to be the string of symbols attached to the leaves, read from left to right in the order induced by δ.

Definition 1. For $A \subseteq \Gamma^*$, the class $\mathrm{Leaf}^{\mathrm{FA}}(A)$ consists of all languages $B \subseteq \Sigma^*$, for which there is a leaf automaton M as just defined, with input alphabet Σ and leaf alphabet Γ such that for all $w \in \Sigma^*$, $w \in B$ iff leafstring$^M(w) \in A$.

The following result from [PV01] about the power of regular leaf languages for finite automata will be central for one of our results below:

Proposition 2. $\mathrm{Leaf}^{\mathrm{FA}}(\mathrm{REG}) = \mathrm{REG}$.

3 The Logical Framework

We follow standard notation for monadic second-order logic with linear order, see, e.g., [Str94]. We restrict our attention to *string signatures*, i.e., signatures of the form $\langle P_{a_1}, \ldots, P_{a_s} \rangle$, where all the predicates P_{a_i} are unary, and in every structure $\mathcal{A}$, $\mathcal{A} \models P_{a_i}(j)$ iff the jth symbol in the input is the letter a_i. Such structures are thus words over the alphabet $\{a_1, \ldots, a_s\}$; first-order variables range over positions within such a word, i.e., from 1 to the word length n; second-order variables range over subsets of $\{1, \ldots, n\}$. The logic's linear order symbol refers to numerical order on $\{1, \ldots, n\}$. For technical reasons to be motivated shortly, we also assume that every alphabet has a built-in linear order, and we write alphabets as sequences of symbols to indicate that order, e.g., in the above case we write $(a_1, \ldots, a_s)$.

Our basic formulas are built from first- and second-order variables in the usual way, using the Boolean connectives $\{\wedge, \vee, \neg\}$, the relevant predicates P_{a_i} together with $\{=, <\}$, the constants min and max, the first- and second-order quantifiers $\{\exists, \forall\}$, and parentheses. For a formula φ, L_φ denotes the language defined by φ, i.e., the set of all models of ϕ; that is: L_φ is a set of words. We use $\triangleq$ to name formulas, e.g.: $\varphi \triangleq \exists x P_1(x)$.

SOM is the class of all languages definable using formulas as just described. (The letters SOM stand for second order monadic logic; in the literature, this logic is sometimes denoted by MSO.) FO is the subclass of SOM restricted to languages definable by first-order formulas. It is known [MP71] that FO is equal to the class of star-free regular languages. In this paper we are mainly interested in the following earlier result (see [BE58, Büc62, Tra61]):

Proposition 3 (Büchi-Elgot-Trakhtenbrot Theorem). *The class* SOM *is equal to the class* REG *of regular languages.*

Next, we extend the logical language allowing generalized quantifiers.

Definition 4. Consider a language L over an alphabet $\Sigma = (a_1, a_2, \ldots, a_s)$. Such a language gives rise to a Lindström quantifier Q_L, that may be applied to any sequence of $s-1$ formulas as follows:

Let $\overline{x}$ be a k-tuple of variables (each of which ranges from 1 to the "input length" n, as we have seen). We assume the lexical ordering on $\{1, 2, \ldots, n\}^k$, and we write $\overline{x}^{(1)} < \overline{x}^{(2)} < \cdots < \overline{x}^{(n^k)}$ for the sequence of potential values taken on by $\overline{x}$. The k-ary *Lindström quantifier* Q_L binding $\overline{x}$ takes a meaning if $s-1$ formulas, each having as free variables the variables in $\overline{x}$ (and possibly others), are available. Let $\varphi_1(\overline{x})$, $\varphi_2(\overline{x})$, ..., $\varphi_{s-1}(\overline{x})$ be these $s-1$ formulas. Then $Q_L\overline{x}\big[\varphi_1(\overline{x}), \varphi_2(\overline{x}), \ldots, \varphi_{s-1}(\overline{x})\big]$ holds on a string $w = w_1 \cdots w_n$, iff the word of length n^k whose ith letter, $1 \leq i \leq n^k$, is

$$\begin{cases} a_1 \text{ if } w \models \varphi_1(\overline{x}^{(i)}), \\ a_2 \text{ if } w \models \neg\varphi_1(\overline{x}^{(i)}) \wedge \varphi_2(\overline{x}^{(i)}), \\ \quad\vdots \\ a_s \text{ if } w \models \neg\varphi_1(\overline{x}^{(i)}) \wedge \neg\varphi_2(\overline{x}^{(i)}) \wedge \cdots \wedge \neg\varphi_{s-1}(\overline{x}^{(i)}), \end{cases}$$

belongs to L. We denote this word by $f^{\overline{x}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$.

As an example, take $s = 2$ and consider $L_\exists =_{\text{def}} 0^*1(0+1)^*$; then $Q_{L_\exists}$ is the usual first-order existential quantifier. Similarly, the universal quantifier can be expressed using the language $L_\forall =_{\text{def}} 1^*$. The quantifiers $Q_{L_{\text{mod } p}}$ for $p > 1$ are known as modular counting quantifiers [Str94].

The Lindström quantifiers of Def. 4 are precisely what has been referred to as "Lindström quantifiers on strings" [BV98]. The original more general definition [Lin66] uses transformations to arbitrary structures, not necessarily of string signature. However, in the context of this paper, only reductions to (mostly regular or context-free) languages or algebraic word-problems will be important, and hence the above definition seems to be the most natural here.

Fix a finite monoid M. Each $S \subseteq M$ defines an M-word-problem, i.e., a language $\mathcal{W}(S, M)$ composed of all words w, over the alphabet M, that "multiply out" to an element of S.

The following definition is due to Barrington, Immerman, and Straubing [BIS90].

Definition 5. A *monoidal quantifier* is a Lindström quantifier Q_L where L is a word-problem of some finite monoid.

It is well-known that any regular language is a homomorphic pre-image of (i.e., reduces via a homomorphism to) a word-problem over some monoid, and, vice-versa, every word-problem of a finite monoid is regular. Hence, a monoidal quantifier is nothing other than a Lindström quantifier Q_L where L is a regular language. Coming back for a moment to the classical definition of Lindström quantifiers, we thus see that a Lindström quantifier on strings defined by a regular language is nothing else than a Lindström quantifier (in the classical sense) defined by a structure that is a finite monoid multiplication table.

The class $Q_{\mathrm{Mon}}\mathrm{FO}$ is the class of all languages definable by applying a single monoidal quantifier to an appropriate tuple of FO formulas. The class $\mathrm{FO}(Q_{\mathrm{Mon}})$ is defined analogously, but allowing monoidal quantifiers to be used as any other quantifier would (i.e., allowing arbitrary nesting).

It was known [BIS90] that first-order logic with unnested unary monoidal quantifiers characterizes the class of regular languages. Recently this has been extended in [LMSV01] as follows:

Theorem 6. $\mathrm{FO}(Q_{\mathrm{Mon}}) = Q_{\mathrm{Mon}}\mathrm{FO} = \mathrm{REG}$.

One of our results below will be that replacing in this statement the first-order quantifiers by monadic second-order quantifiers does not enlarge the expressive power.

A *groupoid* is a finite multiplication table with an identity element. For a fixed groupoid G, each $S \subseteq G$ defines a G-word-problem, i.e., a language $\mathcal{W}(S,G)$ composed of all words w, over the alphabet G, that can be bracketed in such a way that w multiplies out to an element of S. Groupoid word-problems relate to context-free languages in the same way as monoid word-problems relate to regular languages: Every such word-problem is context-free, and every context-free language is a homomorphic pre-image of a groupoid word-problem (this result is credited to Valiant in [BLM93]).

The following definition is due to Bédard, Lemieux, and McKenzie [BLM93]:

Definition 7. A *groupoidal quantifier* is a Lindström quantifier Q_L where L is a word-problem of some finite groupoid.

Usage of groupoidal quantifiers in our logical language is signalled by Q_{Grp}, used in the same way as described for Q_{Mon} above.

Second-order Lindström quantifiers on strings were introduced in [BV98]. Here, we are mainly interested in those binding only set variables, so called *monadic quantifiers*.

Definition 8. Consider a language L over an alphabet $\Sigma = (a_1, a_2, \ldots, a_s)$. Let $\overline{X} = (X_1, \ldots, X_k)$ be a k-tuple of unary second-order variables, i.e., set variables. There are 2^{nk} different instances (assignments) of $\overline{X}$. We assume the following ordering on those instances: Let each instance of a single X_i be encoded by a bit string $s_1^i \cdots s_n^i$ with the meaning $s_j^i = 1 \iff j \in X_i$. Then we encode an instance of $\overline{X}$ by the bit string $s_1^1 s_1^2 \cdots s_1^k s_2^1 s_2^2 \cdots s_2^k \cdots s_n^1 s_n^2 \cdots s_n^k$ and order the instances lexicographically by their codes. We write $\overline{X}^{(1)} < \overline{X}^{(2)} < \cdots < \overline{X}^{(2^{nk})}$ for the sequence of all instances in that order. The *monadic second-order Lindström quantifier* Q_L binding $\overline{X}$ takes a meaning if $s-1$ formulas, each having free variables $\overline{X}$, are available. Let $\varphi_1(\overline{X})$, $\varphi_2(\overline{X})$, $\ldots$, $\varphi_{s-1}(\overline{X})$ be these

$s-1$ formulas. Then $\varphi = Q_L\overline{X}\big[\varphi_1(\overline{X}), \varphi_2(\overline{X}), \ldots, \varphi_{s-1}(\overline{X})\big]$ holds on a string $w = w_1 \cdots w_n$, iff the word of length 2^{nk} whose ith letter, $1 \leq i \leq 2^{nk}$, is

$$\begin{cases} a_1 \text{ if } w \models \varphi_1(\overline{X}^{(i)}), \\ a_2 \text{ if } w \models \neg\varphi_1(\overline{X}^{(i)}) \wedge \varphi_2(\overline{X}^{(i)}), \\ \quad \vdots \\ a_s \text{ if } w \models \neg\varphi_1(\overline{X}^{(i)}) \wedge \neg\varphi_2(\overline{X}^{(i)}) \wedge \cdots \wedge \neg\varphi_{s-1}(\overline{X}^{(i)}), \end{cases}$$

belongs to L. We denote this word by $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$.

Again, taking as examples the languages $L_\exists$ and $L_\forall$, we obtain the usual second-order existential and universal quantifiers.

If L above is a monoid word-problem, then we say that Q_L is a monadic second-order monoidal quantifier; if L is a groupoid word-problem then Q_L is called monadic second-order groupoidal quantifier.

The class mon-Q^1_LFO is the class of all languages describable by applying a specific monadic second-order monoidal quantifier Q_L to an appropriate tuple of formulas without further occurrences of second-order quantifiers. The class mon-Q^1_{Mon}FO is defined analogously using arbitrary monadic second-order monoidal quantifiers. The class mon-Q^1_{Mon}SOM is defined analogously using tuples of SOM formulas. The class SOM(mon-Q^1_{Mon}) is defined analogously, but allowing monoidal quantifiers to be used as any other quantifier would (i. e., allowing arbitrary nesting). Analogous notation with mon-Q^1_{Grp} will be used.

4 Leaf Languages vs. Generalized Quantifiers

We start by proving a technical lemma which we will need later on. We follow the treatment of formulas with free variables developed in detail in [Str94, pp. 14ff], i.e., such formulas define languages of words with (first- and second-order) variables attached to its letters, so called $(\mathcal{V}_1, \mathcal{V}_2)$-*structures* for a set $\mathcal{V}_1$ of first-order and a set $\mathcal{V}_2$ of second-order variables; formally a $(\mathcal{V}_1, \mathcal{V}_2)$-structure over Σ is a word over the alphabet $\Sigma \times 2^{\mathcal{V}_1} \times 2^{\mathcal{V}_2}$ where Σ is the original (letter) alphabet.

Lemma 9. *Let Σ be an alphabet and $\mathcal{V}_1$ and $\mathcal{V}_2$ be sets of first- and second-order variables. Let $\overline{X} = (X_1, \ldots, X_k)$ be a vector of monadic second-order variables. Let $\varphi_1(\overline{X}), \varphi_2(\overline{X}), \ldots, \varphi_{s-1}(\overline{X})$ be formulas (from any calculus) that define regular sets of $(\mathcal{V}_1, \mathcal{V}_2 \cup \{X_1, \ldots, X_k\})$-structures over Σ.*

Let $\Gamma = (a_1, a_2, \ldots, a_s)$ be an ordered alphabet, and let $L \subseteq \Gamma^*$. Define $\varphi \triangleq Q_L\overline{X}\big[\varphi_1(\overline{X}), \varphi_2(\overline{X}), \ldots, \varphi_{s-1}(\overline{X})\big]$; hence φ is a formula which defines a set of $(\mathcal{V}_1, \mathcal{V}_2)$-structures over Σ. Then there exists a leaf automaton M that accepts the language L_φ when working with leaf language L.

Proof. Let $M_1, \ldots, M_{s-1}$ be the deterministic automata that accept $L_{\varphi_1}, \ldots, L_{\varphi_{s-1}}$. For $1 \leq i < s$, let $M_i = (Q_i, \Sigma \times 2^{\mathcal{V}_1} \times 2^{\mathcal{V}_2 \cup \{X_1, \ldots, X_k\}}, \delta_i, s_i, A_i)$. We consider the usual parallel product of the M_i, i.e., $M_\times =_{\text{def}} \left(\times_{i=1}^{s-1} Q_i, \Sigma \times 2^{\mathcal{V}_1} \times 2^{\mathcal{V}_2 \cup \{X_1,\ldots,X_k\}}, \delta_\times, (s_1, s_2, \ldots, s_{s-1}), \times_{i=1}^{s-1} A_i\right)$ with a transition function given by $\delta_\times((q_1, q_2, \ldots, q_{s-1}), (v, S, T)) = (\delta_1(q_1, (v, S, T)), \ldots, \delta_{s-1}(q_{s-1}, (v, S, T)))$.

We define our leaf automaton

$$M =_{\text{def}} \left(\times_{i=1}^{s-1} Q_i, \Sigma \times 2^{\mathcal{V}_1} \times 2^{\mathcal{V}_2}, \delta, (s_1, s_2, \ldots, s_{s-1}), \Gamma, \beta\right)$$

as follows:

We want M to accept only $(\mathcal{V}_1, \mathcal{V}_2)$-structures as input. Certainly we have to simulate the behavior of $M_\times$ on every possible instance of $\overline{X} = (X_1, \ldots, X_k)$. Each position of an input of $M_\times$ may or may not be an element of each (set) variable X_i—in other words: If w is an input of M, then for each read letter $(w_j, \mathcal{V}_1^j, \mathcal{V}_2^j)$ of w, where $w_j \in \Sigma$, $\mathcal{V}_1^j \subseteq \mathcal{V}_1$, $\mathcal{V}_2^j \subseteq \mathcal{V}_2$, we have to simulate $M_\times$ for the cases $X_i \in \mathcal{V}_2^j$ and $X_i \notin \mathcal{V}_2^j$. So there are 2^k possibilities for simulated inputs for each letter of w. We encode the members of $\mathcal{P}(\{X_1, \ldots, X_k\})$ by S_i, $1 \leq i \leq 2^k$ where each set S_i is defined by $X_j \in S_i$ iff the jth bit of $\text{bin}_k(i-1)$ (the length k binary encoding of the natural number $i-1$) is 1.

The transition function of M is now defined as (recall that the transition function for leaf automata maps into *sequences* of states)

$$\begin{aligned}\delta_\varphi((q_1, q_2, \ldots, q_{s-1}), (x, S, T)) = {} & \delta_\times((q_1, q_2, \ldots, q_{s-1}), (x, S, T \cup \{S_1\})) \\ & \delta_\times((q_1, q_2, \ldots, q_{s-1}), (x, S, T \cup \{S_2\})) \\ & \quad \vdots \\ & \delta_\times((q_1, q_2, \ldots, q_{s-1}), (x, S, T \cup \{S_{2^k}\}))\end{aligned}$$

Finally we define

$$\beta(q_1, q_2, \ldots, q_{s-1}) =_{\text{def}} \begin{cases} a_1 \text{ if } q_1 \in A_1, \\ a_2 \text{ if } q_1 \notin A_1 \wedge q_2 \in A_2, \\ \quad \vdots \\ a_s \text{ if } q_1 \notin A_1 \wedge q_2 \notin A_2 \wedge \cdots \wedge q_{s-1} \notin A_{s-1} \end{cases}$$

Now M on input $w = w_1 \ldots w_n$ spans a computation tree $T_M(w)$ of depth n and branching width 2^k. Such a tree has $(2^k)^n = 2^{nk}$ leaves. We will now prove that the leaf string on $T_M(w)$ is identical to the word $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$:

Each of the 2^k branches on an input letter corresponds to a set S_i, $1 \leq i \leq 2^k$. Thus we can encode the branch with the bit string $\text{bin}_k(i-1)$ defining S_i as above. When ordering the leaves on $T_M(w)$, the branching on earlier input letters is of higher importance than the branching on later input letters. So we can encode a path in $T_M(w)$ by the bit string $\text{bin}_k(i_1-1)\text{bin}_k(i_2-1)\cdots\text{bin}_k(i_n-1)$ of length nk, meaning: on input letter j the set S_{i_j} was chosen. If path l has a lexicographically smaller encoding than path l', then the leaf of l has a position in the leaf string of $T_M(w)$ to the left of the leaf on path l'. Since there are exactly 2^k different bit strings $\text{bin}_k(i)$, each of those strings corresponds to one set S_i, and as such it corresponds to the branch i in a forking of the computation tree. Thus the space of all 2^{nk} binary strings of length nk is completely filled with encodings of computation paths. From this follows, that path number l in the computation tree is encoded by the bit string of length nk which is the binary representation of $l-1$.

Now we want to show that on path l the tuple of sets $\overline{X}^{(l)}$ (using the notation from Def. 8) is simulated. The encoding of path l is the binary string z representing the number $l-1$. We can split z in n parts of length k: $z = z_1 z_2 \ldots z_n$. As said before, the bit string $z_j = z_j^1 z_j^2 \ldots z_j^k$ encodes the simulation of sets X_i in this way: $z_j^i = 1$ iff the set X_i was chosen on input letter j. So the value of the variable X_i depends on the values of $z_1^i, z_2^i, \ldots, z_n^i$ of the string $z = z_1^1 z_1^2 \ldots z_1^k z_2^1 z_2^2 \ldots z_2^k \ldots z_n^1 z_n^2 \ldots z_n^k$. But since z represents the number $l-1$, this is exactly the definition of the tuple $\overline{X}^{(l)}$.

Thus we conclude that on path l of the computation tree we simulate the behavior of the machines $M_1, \ldots, M_n$ on input w supplemented by $\overline{X}^{(l)}$. Each M_k simulates the formula φ_k. So, if on reading the last input letter the state q_k is reached on machine M_k then it holds $q_k \in A_k$ iff $w \models \varphi_k(\overline{X}^{(l)})$.

From the definition of the function β we see that the leaf letter on path l is

$$\begin{cases} a_1 \text{ if } w \models \varphi_1(\overline{X}^{(l)}), \\ a_2 \text{ if } w \models \neg\varphi_1(\overline{X}^{(l)}) \wedge \varphi_2(\overline{X}^{(l)}), \\ \quad \vdots \\ a_s \text{ if } w \models \neg\varphi_1(\overline{X}^{(l)}) \wedge \neg\varphi_2(\overline{X}^{(l)}) \wedge \cdots \wedge \neg\varphi_{s-1}(\overline{X}^{(l)}). \end{cases}$$

But this is the letter l of the word $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$ (see Def. 8). This completes the proof that the leaf string on computation tree $T_M(w)$ is identical to the word $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$.

This implies $w \models Q_L \overline{X}\big[\varphi_1(\overline{X}), \varphi_2(\overline{X}), \ldots, \varphi_{s-1}(\overline{X})\big]$ iff $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w) \in L$ iff $\text{leafstring}^M(w) \in L$. Thus M with leaf language L accepts the language L_φ. $\square$

Let $\mathcal{N}$ be the class of all languages that have a neutral letter; i.e., $L \subseteq \Gamma^*$ is in $\mathcal{N}$ if there is a letter $\not\epsilon \in \Gamma$ such that, for all $u, v \in \Gamma^*$, we have $uv \in L \iff u\not\epsilon v \in L$.

Our main result in this section states that, for all languages $L \in \mathcal{N}$, finite automata with leaf language L accept exactly those languages definable with a monadic Q_L^1 quantifier.

Theorem 10. *For any* $L \in \mathcal{N}$, $\text{Leaf}^{\text{FA}}(L) = \text{mon-}Q_L^1\text{FO}$.

Proof. We first consider the inclusion $\text{Leaf}^{\text{FA}}(L) \supseteq \text{mon-}Q_L^1\text{FO}$.

Let φ be a mon-Q_L^1FO-formula which defines a language $L_\varphi \subseteq \Sigma^*$. Then φ will have the form $Q_L \overline{X}\big[\varphi_1(\overline{X}), \varphi_2(\overline{X}), \ldots, \varphi_{s-1}(\overline{X})\big]$ for some formulas $\varphi_i(\overline{X})$, $1 \le i \le s-1$, which are first order except of the occurrence of the monadic variables in $\overline{X}$. Hence we conclude by Lemma 9 that there must exist a leaf automaton M_φ which accepts the language L_φ.

It remains to prove the opposite inclusion $\mathrm{Leaf}^{\mathrm{FA}}(L) \subseteq$ mon-Q_L^1FO.

Let $\Gamma = \{a_1, a_2, \ldots, a_{s-1}, \epsilon\}$, let $L \in \mathcal{N} \cap \mathcal{P}(\Gamma^*)$ where ϵ is neutral, and let $M = (Q, \Sigma, \delta, q_1, \Gamma, \beta)$ be a leaf automaton. Suppose that M working with leaf language L accepts a certain language $A \subseteq \Sigma^*$. We have to construct a mon-Q_L^1FO-formula φ such that $L_\varphi = A$.

First, order the leaf alphabet in such a way that ϵ is the last symbol in Γ, e.g., $\Gamma = (a_1, a_2, \ldots, a_{s-1}, \epsilon)$. Let $Q = \{q_1, \ldots, q_l\}$ and let $m =_{\mathrm{def}} \max\{\, |\delta(q,a)| \mid q \in Q, a \in \Sigma \,\}$. Formula φ will be of the form $Q_L \overline{X}\big[\varphi_1(\overline{X}), \varphi_2(\overline{X}), \ldots, \varphi_{s-1}(\overline{X})\big]$ where $\overline{X} =_{\mathrm{def}} (X_m, \ldots, X_1, Y_l, \ldots, Y_1)$.

We want to encode the number of a computation path of M on input w in the variables X_i. This will be done in the following way: $j \in X_i$ iff M reading the j-th symbol of w chooses the i-th alternative of the successor function.

In the variables Y_i we want to encode the sequence of states of M on input w on a certain computation path. We shall do this in the following form: $j \in Y_i$ iff M (on that computation path whose number is given by variables X_l) while reading the j-th symbol of the input is in state q_i.

The formulas φ_k on input w and an instance of $\overline{X}$ thus have to express the following:

1. All φ_k have to test whether the X_i encode a computation path of M on w. If this is not the case (e.g. if there exists a j and $i_1 \neq i_2$ with $j \in X_{i_1}$ and $j \in X_{i_2}$) then $w \not\models \varphi_k(\overline{X})$ for all $k \in \{1, \ldots, s-1\}$. (A look at the definition of $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$ shows that the letter of $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$ yielded by $\overline{X}$ will then be ϵ.)
2. All φ_k have to test whether the Y_i encode the correct sequence of states of M for the computation path encoded by the X_i. If this is not the case then $w \not\models \varphi_k(\overline{X})$ for all $k \in \{1, \ldots, s-1\}$. (The letter of $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$ yielded by $\overline{X}$ thus again will be ϵ.)
3. If $\overline{X}$ encodes a computation of M—i. e., if the X_i encode the number of a path of M and the Y_i encode the correct sequence of states of M on that path—then $w \models \varphi_k(\overline{X})$ iff M on the computation path encoded by the X_i produces the leaf letter a_k. (The letter of $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$ yielded by $\overline{X}$ will then be identical to the leaf letter produced by M on the computation path encoded by the X_i.)

It is easy to construct FO-formulas with this behavior.

We now show that the word $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$ is essentially equal to the word $\mathrm{leafstring}^M(w)$, more precisely: that both words are identical once all occurrences of the letter ϵ are deleted from them; hence they are equivalent w.r.t. membership in L.

Since each computation of M is encoded by a certain instance of $\overline{X}$, each letter in $\mathrm{leafstring}^M(w)$ is contained in $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$, also. And since each instance of $\overline{X}$ not encoding a sensible computation of M yields the letter ϵ in $f^{\overline{X}}_{\varphi_1 \cdots \varphi_{s-1}}(w)$, we know that the amount of each letter different from ϵ in both words is identical. All there is left to do is showing that their sequences are identical in both words.

Let p_1 and p_2 be computation paths in M, u and v their respective leaf letters and $\overline{X_1}$ and $\overline{X_2}$ the encoding of the sensible computations associated with these paths (it was shown above that then u and v are also the letters of $f^{\overline{X}}_{\varphi_1\cdots\varphi_{s-1}}(w)$ produced by $\overline{X_1}$ and $\overline{X_2}$). We show that p_1 is positioned before p_2 iff the encoding of $\overline{X_1}$ is of lower order than the encoding of $\overline{X_2}$.

The encoding of an instance $\overline{X_k}$, according to the definition of $f^{\overline{X}}_{\varphi_1\cdots\varphi_{s-1}}(w)$, can be separated into n bit strings of length $m+l$. Each of those strings is of the form $x_m \ldots, x_1, y_l, \ldots, y_1$. The j-th such string corresponds to the input letter j with the meaning $x_i = 1 \leftrightarrow j \in X_i$ (there will always be only one $x_i = 1$, since the $\overline{X_k}$ are correct encodings) and $y_i = 1 \leftrightarrow j \in Y_i$ (there will also always be only one $y_i = 1$).

Two computation paths that differ have a word position on which the first difference occurs. We call it d. Up to position d not only the sequences of successor function choices are identical for both paths but also the sequence of states M assumes while reading letters of w; this holds because the computations so far are identical.

But from this it follows that the encodings of the $\overline{X_k}$ are identical up to the bit string number d. So, this string decides which encoding is of higher order. Let the d-th string of $\overline{X_1}$ be $x^1_m \ldots x^1_1 y^1_l \ldots y^1_1$ and of $\overline{X_2}$ be $x^2_m \ldots x^2_1 y^2_l \ldots y^2_1$.

The unique $x^1_a = 1$ is the choice for the successor function on computation path p_1 and input letter d while the $x^2_b = 1$ is the corresponding choice for path p_2. Then $a < b$ iff p_1 is ordered before p_2 in $T_M(w)$ (this is due to the ordering which the successor function imposes on the computation tree) and the string $x^1_m \ldots x^1_b \ldots x^1_a \ldots x^1_1 y^1_l \ldots y^1_1 = 0 \ldots 0 \ldots 1 \ldots 0 y^1_l \ldots y^1_1$ is smaller than $x^2_m \ldots x^2_b \ldots x^2_a \ldots x^2_1 y^2_l \ldots y^2_1 = 0 \ldots 1 \ldots 0 \ldots 0 y^2_l \ldots y^2_1$. Thus the ordering of $\overline{X_1}$ and $\overline{X_2}$ is identical to the ordering of p_1 and p_2.

This concludes the proof that the words $f^{\overline{X}}_{\varphi_1\cdots\varphi_{s-1}}(w)$ and $\mathrm{leafstring}^M(w)$ are identical if the letters ϵ are not considered. Thus $w \in A$ iff $\mathrm{leafstring}^M(w) \in L$ iff $f^{\overline{X}}_{\varphi_1\cdots\varphi_{s-1}}(w) \in L$ iff $w \in L_\varphi$, hence we conclude that φ defines the language A accepted by M. ☐

5 Monoidal Quantifiers

We can now give an extension of Proposition 3: if we extend the monadic second-order formalism by monoidal quantifiers, we do not gain expressive power.

Theorem 11. $\text{mon-}Q^1_{\text{Mon}}\text{FO} = \text{FO}(\text{mon-}Q^1_{\text{Mon}}) = \text{mon-}Q^1_{\text{Mon}}\text{SOM} = \text{SOM}(\text{mon-}Q^1_{\text{Mon}}) = \text{REG}.$

Proof. The inclusions $\text{mon-}Q^1_{\text{Mon}}\text{FO} \subseteq \text{mon-}Q^1_{\text{Mon}}\text{SOM} \subseteq \text{SOM}(\text{mon-}Q^1_{\text{Mon}})$ and $\text{mon-}Q^1_{\text{Mon}}\text{FO} \subseteq \text{FO}(\text{mon-}Q^1_{\text{Mon}}) \subseteq \text{SOM}(\text{mon-}Q^1_{\text{Mon}})$ are trivial.

For the inclusion $\text{REG} \subseteq \text{mon-}Q^1_{\text{Mon}}\text{FO}$ we use the fact that every regular language can be defined by an SOM formula with only one second-order existential quantifier preceding a first-order formula [Tho82], and an existential quantifier is one particular monoidal quantifier—as already mentioned, it is equivalent to $Q_{L_\exists}$ with $L_\exists = 0^*1(0+1)^*$.

Now all there is left to do is proving that $\mathrm{SOM}(\text{mon-}Q^1_{\mathrm{Mon}}) \subseteq \mathrm{REG}$. We do this closely following the proof of Theorem III.1.1 in [Str94]. There, $\mathrm{SOM} \subseteq \mathrm{REG}$ is proven proceeding by induction on the construction of second order monadic formulas. Since here we have SOM-formulas with one additional constructive element—the Lindström quantifier—we have to extend the inductive proof by one step:

Let Σ be an alphabet and $\mathcal{V}_1$ and $\mathcal{V}_2$ be sets of first- and second-order variables. Then a formula $\varphi \in \mathrm{SOM}(\text{mon-}Q^1_{\mathrm{Mon}})$ with free variables $\mathcal{V}_1 \cup \mathcal{V}_2$ accepts a language $L_\varphi \in (\Sigma \times 2^{\mathcal{V}_1} \times 2^{\mathcal{V}_2})^*$. We have to prove that L_φ is a regular language. As mentioned, the proof proceeds by induction, see [Str94, pp. 21ff]. The additional case in the induction step for us here is that of $\varphi \triangleq Q_L\overline{X}\big[\varphi_1(\overline{X}), \varphi_2(\overline{X}), \ldots, \varphi_{s-1}(\overline{X})\big]$: By induction hypothesis there exist deterministic automata $M_1, M_2, \ldots, M_{s-1}$ that accept the $(\mathcal{V}_1, \mathcal{V}_2 \cup \{X_1, \ldots, X_k\})$-structures that model $\varphi_1, \varphi_2, \ldots, \varphi_{s-1}$. Hence we can use Lemma 9 to conclude that there is a leaf automaton M_φ that accepts L_φ when working w.r.t. leaf language L.

But since L is regular, we can use Proposition 2 to see that then L_φ is regular, too. This concludes the induction and the proof of the theorem. $\square$

This result, together with the result of Thomas [Tho82] mentioned in the above proof, leads immediately to the following *normal form for monadic second-order monoidal logic*:

Corollary 12. *Every* $\mathrm{SOM}(\text{mon-}Q^1_{\mathrm{Mon}})$*-formula is equivalent to a formula of the form* $\exists X \varphi(X)$*, where* X *is a set variable and* φ *is a formula without second-order quantifiers.*

6 Groupoidal Quantifiers

Finally, we examine the power of monadic second-order groupoidal quantifiers. These are powerful enough to define every language in LOGCFL, as we prove next.

Theorem 13. $\mathrm{LOGCFL} \subseteq \text{mon-}Q^1_{\mathrm{Grp}}\mathrm{FO}$.

Proof. Lautemann et al. in [LMSV01] showed that $\mathrm{LOGCFL} = Q_{\mathrm{Grp}}\mathrm{FO}$. This means that for every language $L \in \mathrm{LOGCFL}$ there is a sentence φ in first-order logic prefixed with a generalized quantifier for a context-free language such that $L = L_\varphi$.

However, such a CFL-quantifier can easily be simulated by a monadic second-order Lindström quantifier defined by another context-free language as follows: Consider the formula $\varphi \triangleq Q_L x_1 \cdots x_k \big[\psi(x_1 \cdots x_k)\big]$ for first-order formula ψ and context-free language L over $(0,1)$ (the case of larger alphabets proceeds completely analogously). Let L' be obtained from L by adding a neutral letter, i.e., L' is defined over alphabet $(0, 1, \not\epsilon)$ where $\not\epsilon$ is neutral. Define

$$\varphi' \triangleq Q_{L'} X_1 \cdots X_k \big[\psi_1(X_1 \cdots X_k), \psi_2(X_1 \cdots X_k)\big],$$

where

$$\psi_1(X_1 \cdots X_k) \triangleq \exists x_1 \cdots x_k \Big(\textstyle\bigwedge_{i=1}^k \big(x_i \in X_i \land \forall y(y \in X_i \to x_i = y)\big) \land \psi(x)\Big),$$
$$\psi_2(X_1 \cdots X_k) \triangleq \exists x_1 \cdots x_k \Big(\textstyle\bigwedge_{i=1}^k \big(x_i \in X_i \land \forall y(y \in X_i \to x_i = y)\big) \land \neg\psi(x)\Big).$$

Then the only assignments of a set variable X_i that do not lead to output symbol $\not\epsilon$ in $f_{\psi_1\psi_2}^{(X_1,\ldots,X_k)}$ are the singleton sets, and these correspond (in the same relative order) to assignments of x_i in ψ. Hence, for every input w, $f_{\psi_1\psi_2}^{(X_1,\ldots,X_k)}(w) \in L' \iff f_\psi^{(x_1,\ldots,x_k)}(w) \in L$, and thus, $L_\varphi = L_{\varphi'}$.

This proves $\mathrm{LOGCFL} = Q_{\mathrm{Grp}}\mathrm{FO} \subseteq \text{mon-}Q^1_{\mathrm{Grp}}\mathrm{FO}$. □

Peichl and Vollmer, after introducing finite leaf automata in [PV01], examined the power of various classes of leaf languages and thus obtained a number of new characterizations of complexity classes and formal language classes. One particular case, left open in that paper, is the following: What is the power of finite leaf automata with context-free leaf languages, i.e., how does the class $\mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL})$ relate to other well-known classes? The only upper and lower bounds known were $\mathrm{CFL} \subsetneq \mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL}) \subseteq \mathrm{DSPACE}(n^2) \cap \mathrm{DTIME}\big(2^{O(n)}\big)$. Here, we improve the lower bound.

Corollary 14. $\mathrm{LOGCFL} \subseteq \mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL})$.

Proof. From Theorem 13, we obtain $\mathrm{LOGCFL} \subseteq \text{mon-}Q^1_{\mathrm{Grp}}\mathrm{FO} = \text{mon-}Q^1_{\mathrm{CFL}}\mathrm{FO}$. By Theorem 10, however, the latter class is equal to $\mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL})$, yielding $\mathrm{LOGCFL} \subseteq \mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL})$. □

7 Conclusion

We considered monadic second-order logic extended with generalized quantifiers. We contributed to the study of the expressive power of such logics as well as to the study of the power of finite automata with generalized acceptance criteria. So far, we did not exploit the close connection between logical reductions and generalized quantifiers. This might lead to further interesting consequences.

One of the results of this paper is that by adding *arbitrary* monadic second-order monoidal quantifiers to SOM does not give us more power. On the other hand one might ask what specific regular languages L besides $L_\exists$ (for nondeterministic acceptance, see Theorem 11 and Corollary 12) there are, for which $Q^1_L\mathrm{FO} = \mathrm{REG}$. Even for a very "easy" language obtained from deterministic acceptance of regular languages, namely $L_{\det} = 1(0+1)^*$, we have $Q^1_{L_{\det}}\mathrm{FO} =$

REG. (This implies easily that also $Q^1_{L_{\mathrm{mod}\ p}}\mathrm{FO} = \mathrm{REG}$.) Can interesting subclasses of REG be defined with restricted quantifiers?

Still open remains a complete clarification of the power of $\mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL})$. We proved $\mathrm{LOGCFL} \subseteq \mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL})$. It is relatively easy to see that every language in $\mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL})$ can be accepted by auxiliary pushdown automata with linear space and simultaneous exponential time bound; hence $\mathrm{LOGCFL} = \text{NAuxPDA-SPACE-TIME}(\log n, n^{O(1)}) \subseteq \mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL}) \subseteq \text{NAuxPDA-SPACE-TIME}(n, 2^{O(n)})$. Is one of these inclusions actually an equality? And how do the logics in the range from mon-$Q^1_{\mathrm{Grp}}\mathrm{FO}$ to SOM(mon-Q^1_{Grp}) relate to these classes?

Acknowledgment

We thank Gerhard Buntrock (Lübeck), Klaus-Jörn Lange (Tübingen), Klaus Reinhard (Tübingen), and, in particular, Sven Kosub (Würzburg) for helpful discussions. We also thank the anonymous referees for hints that helped to improve the presentation of the paper.

References

[BE58] J. R. Büchi and C. C. Elgot. Decision problems of weak second order arithmetics and finite automata, Part I. *Notices of the American Mathematical Society*, 5:834, 1958.

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

[BLM93] F. Bédard, F. Lemieux, and P. McKenzie. Extensions to Barrington's M-program model. *Theoretical Computer Science*, 107:31–61, 1993.

[Büc62] J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Proceedings Logic, Methodology and Philosophy of Sciences 1960*, Stanford, CA, 1962. Stanford University Press.

[BV98] H.-J. Burtschick and H. Vollmer. Lindström quantifiers and leaf language definability. *International Journal of Foundations of Computer Science*, 9:277–294, 1998.

[Coo71] S. A. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *Journal of the Association for Computing Machinery*, 18:4–18, 1971.

[EF95] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer Verlag, Berlin Heidelberg, 1995.

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

[LMSV01] C. Lautemann, P. McKenzie, T. Schwentick, and H. Vollmer. The descriptive complexity approach to LOGCFL. *Journal of Computer and Systems Sciences*, 2001. To appear. A preliminary version appeared in the *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science Vol. 1563, pp. 444–454, Springer Verlag, 1999.

[MP71] R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press, 1971.

[PV01] T. Peichl and H. Vollmer. Finite automata with generalized acceptance criteria. *Discrete Mathematics and Theoretical Computer Science*, 2001. To appear. A preliminary version appeared in the *Proceedings of the 26th International Colloqium on Automata, Languages, and Programming*, Lecture Notes in Computer Science Vol. 1644, pp. 605–614, Springer Verlag, 1999.

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

[Sud78] I. H. Sudborough. On the tape complexity of deterministic context-free languages. *Journal of the Association for Computing Machinery*, 25:405–414, 1978.

[Tho82] W. Thomas. Classifying regular events in symbolic logic. *Journal of Computer and Systems Sciences*, 25:360–376, 1982.

[Tra61] B. A. Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329, 1961. In Russian.