In: Proc. 5th International Conf. on Developments in Language Theory (DLT 2001),

Second-Order Logic over Strings: Regular and Non-Regular Fragments

Thomas Eiter[1], Georg Gottlob[1], and Thomas Schwentick[2]

[1] Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Wien, Austria
eiter@kr.tuwien.ac.at, gottlob@dbai.tuwien.ac.at
[2] Institut für Informatik, Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 1-4, D-07743 Jena, Germany
tick@minet.uni-jena.de

Abstract. By a well-known result due to Büchi and Trakhtenbrot, all monadic second-order sentences over words describe regular languages. In this paper, we investigate prefix classes of general second-order logic. Such a prefix class is called *regular*, if each of its sentences describes a regular language, and *nonregular* otherwise. Recently, the regular and nonregular prefix classes of existential second-order logic (Σ_1^1) were exhaustively determined. We briefly recall these results and continue this line of research by systematically investigating the syntactically more complex prefix classes $\Sigma_k^1(\mathcal{Q})$ of second-order logic for each integer $k > 1$ and for each first-order quantifier prefix $\mathcal{Q}$. We give an exhaustive classification of the regular and nonregular prefix classes of this form, and derive of complexity results for the corresponding model checking problems. We also give a brief survey of recent results on the complexity of evaluating existential second-order logic over graphs, and a list of interesting open problems.

1 Introduction

Ever since the embryogenesis of computer science, the following scientific method has been successfully used: Represent computational problems by means of logical formulas, and derive insights into the structure (and thus the solvability and complexity) of a problem from the analysis of the structure of the corresponding formula. In particular, logicians and computer scientists have been studying for a long time the relationship between fragments of predicate logic and the solvability and complexity of decision problems that can be expressed within such fragments. Among the studied fragments, quantifier prefix classes play a predominant role. This can be explained by the syntactical simplicity of such prefix classes and by the fact that they form a natural hierarchy of increasingly complex fragments of logic that appears to be deeply related to core issues of decidability and complexity. In fact, one of the most fruitful research programs that kept logicians and computer scientists busy for decades was the exhaustive solution of Hilbert's classical decision problem (cf. [2]), i.e., of the problem of determining those prefix classes of first-order logic for which formula-satisfiability (resp. finite satisfiability of formulas) is decidable.

Quantifier prefixes emerged not only in the context of decidability theory (a common branch of recursion theory and theoretical computer science), but also in core areas of computer science such as formal language and automata theory, and later in complexity theory. In automata theory, Büchi [4, 3] and Trakhtenbrot [20] independently proved that a language is regular iff it can be described by a sentence of monadic second-order logic, in particular, by a sentence of monadic existential second-order logic. In complexity theory, Fagin [9] showed that a problem on finite structures is in NP iff it can be described by a sentence of existential second-order logic (ESO). These fundamental results have engendered a large number of further investigations and results on characterizing language and complexity classes by fragments of logic (see, e.g. the monographs [17, 15, 6, 12]).

What has not been achieved so far is the exhaustive classification of the complexity of second-order prefix classes over finite structures. By complexity of a prefix class C we mean the complexity of the following model checking problem: Given a fixed sentence Φ in C, decide for variable finite structures A whether A is a model of Φ, which we denote by $A \models \Phi$. Determining the complexity of all prefix classes is an ambitious research programme, in particular the analysis of various types of finite structures such as *strings*, i.e., finite word structures with successor, *trees*, or *graphs*. Over strings and trees, one of the main goals of this classification is to determine the *regular* prefix classes, i.e., those whose formulas express regular languages only; note that by Büchi's theorem, regular fragments over strings are (semantically) included in monadic second-order logic.

In the context of this research programme, two systematic studies were carried out recently, that shed light on the prefix classes of the existential fragment ESO (also denoted by Σ^1_1) of second-order logic:

- In [8], the ESO prefix-classes over strings are exhaustively classified. In particular, the precise frontier between regular and nonregular classes is traced, and it is shown that every class that expresses some nonregular language also expresses some NP-complete language. There is thus a huge complexity gap in ESO: some prefix classes can express only regular languages (which are well-known to have extremely low complexity), while all others are intractable. The results of [8] are briefly reviewed in Section 3.
- In [10], the complexity of all ESO prefix-classes over graphs and arbitrary relational structures is analyzed, and the tractability/intractability frontier is completely delineated. Unsurprisingly, several classes that are regular over strings become NP-hard over graphs. Interestingly, the analysis shows that one of the NP-hard classes becomes polynomial for the restriction to undirected graphs without self-loops. A brief account of these results is given in Section 5.

In the present paper, we continue the research on strings initiated in [8]. But rather than considering fragments of ESO $= \Sigma^1_1$, we will venture into spheres of the higher prefix classes Σ^1_k, where $k > 1$. For a first-order quantifier pattern $\mathcal{Q}$, we denote by $\Sigma^1_k(\mathcal{Q})$ the class of all second-order formulas over strings of the form

$$\exists \mathbf{R}_1 \forall \mathbf{R}_2 \cdots Q_k \mathbf{R}_k \ \mathcal{Q}[\mathbf{x}] \ \varphi(\mathbf{R}_1, \ldots, \mathbf{R}_k, \mathbf{x}),$$

where, for $1 \leq i \leq k$, $\mathbf{R}_i$ is a list of relations, and $\mathcal{Q}([\mathbf{x}])$ denotes a quantifier prefix according to pattern $\mathcal{Q}$ quantifying over first-order variables from the list $\mathbf{x}$.

We give an exhaustive classification of the regular and nonregular second-order prefix classes described by the notation $\Sigma^1_k(\mathcal{Q})$, for all integers $k > 1$ and all first-order quantification patterns $\mathcal{Q}$. We also present complexity results for the corresponding model checking problems. However, at this point, we do not completely describe the tractability frontier but leave a few complexity problems for further work (see Section 6).

The rest of this paper is organized as follows. In Section 2, we state the relevant definitions and recall some classical results. In Section 3, we briefly survey the results of [8] on the classification of ESO prefix-classes over strings. Section 4, in which we present our new results, is the core of this paper. In Section 5, we give a brief account on the classification of ESO over graphs from [10]. Finally, we conclude the paper with a list of interesting open problems in Section 6.

2 Preliminaries and Classical Results

We consider second-order logic with equality (unless explicitly stated otherwise) and without function symbols of positive arity. Predicates are denoted by capitals and individual variables by lower case letters; a bold face version of a letter denotes a tuple of corresponding symbols.

A *prefix* is any string over the alphabet $\{\exists, \forall\}$, and a *prefix set* is any language $\mathcal{Q} \subseteq \{\exists, \forall\}^*$ of prefixes. A prefix set $\mathcal{Q}$ is *trivial*, if $\mathcal{Q} = \emptyset$ or $\mathcal{Q} = \{\lambda\}$, i.e., it consists of the empty prefix. In the rest of this paper, we focus on nontrivial prefix sets. We often view a prefix Q as the prefix class $\{Q\}$. A *generalized prefix* is any string over the extended prefix alphabet $\{\exists, \forall, \exists^*, \forall^*\}$. A prefix set $\mathcal{Q}$ is *standard*, if either $\mathcal{Q} = \{\exists, \forall\}^*$ or $\mathcal{Q}$ can be given by some generalized prefix.

For any prefix Q, the class $\Sigma^1_0(Q)$ is the set of all prenex first-order formulas (which may contain free variables and constants) with prefix Q, and for every $k \geq 0$, $\Sigma^1_{k+1}(Q)$ (resp., Π^1_{k+1}) is the set of all formulas $\exists \mathbf{R} \Phi$ (resp., $\forall \mathbf{R} \Phi$) where Φ is from Π^1_k (resp., Σ^1_k). For any prefix set $\mathcal{Q}$, the class $\Sigma^1_k(\mathcal{Q})$ is the union $\Sigma^1_k(\mathcal{Q}) = \bigcup_{Q \in \mathcal{Q}} \Sigma^1_k(\mathcal{Q})$. We write also ESO for Σ^1_1. For example, ESO$(\exists^*\forall\exists^*)$ is the class of all formulas $\exists \mathbf{R} \exists \mathbf{y} \forall x \exists \mathbf{z} \varphi$, where φ is quantifier-free; this is the class of ESO-prefix formulas, whose first-order part is in the well-known Ackermann class with equality.

Let $A = \{a_1, \ldots, a_m\}$ be a finite alphabet. A *string* over A is a finite first-order structure $W = \langle U, C^W_{a_1}, \ldots, C^W_{a_m}, Succ^W, min^W, max^W \rangle$, for the vocabulary $\sigma_A = \{C_{a_1}, \ldots, C_{a_m}, Succ, min, max\}$, where

- U is a nonempty finite initial segment $\{1, 2, \ldots, n\}$ of the positive integers;
- each $C^W_{a_i}$ is a unary relation over U (i.e., a subset of U) for the unary predicate C_{a_i}, for $i = 1, \ldots, m$, such that the $C^W_{a_i}$ are pairwise disjoint and $\bigcup_i C^W_{a_i} = U$.
- $Succ^W$ is the usual successor relation on U and min^W and max^W are the first and the last element in U, respectively.

Observe that this representation of a string is a *successor structure* as discussed e.g. in [7]. An alternative representation uses a standard linear order $<$ on U instead of the successor *Succ*. In full ESO or second-order logic, $<$ is tantamount to *Succ* since either predicate can be defined in terms of the other.

The strings W for A correspond to the nonempty finite words over A in the obvious way; in abuse of notation, we often use W in place of the corresponding word from A^* and vice versa.

A *SO sentence* Φ over the vocabulary σ_A is a second-order formula whose only free variables are the predicate variables of the signature σ_A, and in which no constant symbols except min and max occur. Such a sentence defines a language over A, denoted $\mathcal{L}(\Phi)$, given by $\mathcal{L}(\Phi) = \{W \in A^* \mid W \models \Phi\}$. We say that a language $L \subseteq A^*$ is *expressed* by Φ, if $\mathcal{L}(\Phi) = L \cap A^+$ (thus, for technical reasons, without loss of generality we disregard the empty string); L is *expressed by a set* S of sentences, if L is expressed by some $\Phi \in S$. We say that S *captures* a class C of languages, if S expresses all and only the languages in C.

Example 2.1. Let us consider some languages over the alphabet $A = \{a, b\}$, and how they can be expressed using logical sentences.

- $L_1 = \{a, b\}^* b \{a, b\}^*$: This language is expressed by the simple sentence

$$\exists x. C_b(x).$$

- $L_2 = a^* b$: This language is expressed by the sentence

$$C_b(max) \wedge \forall x \neq max. C_a(x).$$

- $L_3 = (ab)^*$: Using the successor predicate, we can express this language by

$$C_a(min) \wedge C_b(max) \wedge \forall x, y. Succ(x, y) \rightarrow (C_a(x) \leftrightarrow \neg C_a(y)).$$

- $L_4 = \{w \in \{a, b\}^* \mid |w| = 2n, n \geq 1\}$: We express this language by the sentence

$$\exists E \, \forall x, y. \neg E(min) \wedge E(max) \wedge Succ(x, y) \rightarrow (E(x) \leftrightarrow \neg E(y)).$$

  Note that this a monadic ESO sentence. It postulates the existence of a monadic predicate E, i.e., a "coloring" of the string such that neighbored positions have different color, and the first and last position are uncolored and colored, respectively.
- $L_5 = \{a^n b^n \mid n \geq 1\}$: Expressing this language is more involved:

$$\begin{aligned}\exists R \, \forall x, x^+, y, y^-. R(min, max) \wedge [R(x, y) \rightarrow (C_a(x) \wedge C_b(y))] \wedge \\ [(Succ(x, x^+) \wedge Succ(y^-, y) \wedge R(x, y)) \rightarrow R(x^+, y^-)].\end{aligned}$$

  Observe that this sentence is not monadic. Informally, it postulates the existence of an arc from the first to the last position of the string W, which must be an a and a b, respectively, and recursively arcs from the i-th to the $(|W| - i + 1)$-th position.

Let A be a finite alphabet. A sentence Φ over σ_A is called *regular*, if $\mathcal{L}(\Phi)$ is a regular language. A set of sentences S (in particular, any ESO-prefix class) is *regular*, if for every finite alphabet A, all sentences $\Phi \in S$ over σ_A are regular.

Büchi [4] has shown the following fundamental theorem, which was independently found by Trakhtenbrot [20]. Denote by MSO the fragment of second-order logic in which all predicate variables have arity at most one,[1] and let REG denote the class of regular languages.

Proposition 2.1 (Büchi's Theorem). MSO *captures* REG.

That MSO can express all regular languages is easy to see, since it is straightforward to describe runs of a finite state automaton by an existential MSO sentence. In fact, this is easily possible in monadic $\mathrm{ESO}(\forall\exists)$ as well as in monadic $\mathrm{ESO}(\forall\forall)$. Thus, we have the following lower expressiveness bound on ESO-prefix classes over strings.

Proposition 2.2. *Let $\mathcal{Q}$ be any prefix set. If $\mathcal{Q} \cap \{\exists,\forall\}^*\forall\{\exists,\forall\}^+ \neq \emptyset$, then* $\mathrm{ESO}(\mathcal{Q})$ *expresses all languages in* REG.

On the other hand, with non-monadic predicates allowed ESO has much higher expressivity. In particular, by Fagin's result [9], we have the following.

Proposition 2.3 (Fagin's Theorem). ESO *captures* NP.

This theorem can be sharpened to various fragments of ESO. In particular, by Leivant's results [14, 7], in the presence of a successor and constants min and max, the fragment $\mathrm{ESO}(\forall^*)$ captures NP; thus, $\mathrm{ESO}(\forall^*)$ expresses all languages in NP.

3 Recent Results on ESO over Strings

Combining and extending the results of Büchi and Fagin, it is natural to ask: What about (nonmonadic) prefix classes $\mathrm{ESO}(\mathcal{Q})$ over finite strings? We know by Fagin's theorem that all these classes describe languages in NP. But there is a large spectrum of languages contained in NP ranging from regular languages (at the bottom) to NP-hard languages at the top. What can be said about the languages expressed by a given prefix class $\mathrm{ESO}(\mathcal{Q})$? Can the expressive power of these fragments be characterized? In order to clarify these issues, the following particular problems where investigated in [8]:

• Which classes $\mathrm{ESO}(\mathcal{Q})$ express only regular languages? In other terms, for which fragments $\mathrm{ESO}(\mathcal{Q})$ is it true that for any sentence $\Phi \in \mathrm{ESO}(\mathcal{Q})$ the set $Mod(\Phi) = \{W \in A^* \mid W \models \Phi\}$ of all finite strings (over a given finite alphabet A) satisfying Φ constitutes a regular language? By Büchi's Theorem, this question is identical to the following: Which prefix classes of ESO are (semantically) included in MSO?

[1] Observe that we assume MSO allows one to use nullary predicate variables (i.e., propositional variables) along with unary predicate variables. Obviously, Büchi's Theorem survives.

Note that by Gurevich's classifiability theorem (cf. [2]) and by elementary closure properties of regular languages, it follows that there is *a finite number of maximal regular prefix classes* $\mathrm{ESO}(\mathcal{Q})$, and similarly, of minimal nonregular prefix classes; the latter are, moreover, standard prefix classes (cf. Section 2). It was the aim of [8] to determine the maximal regular prefix classes and the minimal nonregular prefix classes.

- What is the complexity of model checking (over strings) for the *nonregular* classes $\mathrm{ESO}(\mathcal{Q})$, i.e., deciding whether $W \models \Phi$ for a given W (where Φ is fixed)?

Model checking for regular classes $\mathrm{ESO}(\mathcal{Q})$ is easy: it is feasible by a finite state automaton. We also know (e.g. by Fagin's Theorem) that some classes $\mathrm{ESO}(\mathcal{Q})$ allow us to express NP-complete languages. It is therefore important to know (i) which classes $\mathrm{ESO}(\mathcal{Q})$ can express NP-complete languages, and (ii) whether there are prefix classes $\mathrm{ESO}(\mathcal{Q})$ of intermediate complexity between regular and NP-complete classes.

- Which classes $\mathrm{ESO}(\mathcal{Q})$ *capture* the class REG? By Büchi's Theorem, this question is equivalent to the question of which classes $\mathrm{ESO}(\mathcal{Q})$ have exactly the expressive power of MSO over strings.
- For which classes $\mathrm{ESO}(\mathcal{Q})$ is finite satisfiability decidable, i.e., given a formula $\Phi \in \mathrm{ESO}(\mathcal{Q})$, decide whether Φ is true on some finite string ?

Reference [8] answers all the above questions exhaustively. Some of the results are rather unexpected. In particular, a surprising dichotomy theorem is proven, which sharply classifies all $\mathrm{ESO}(\mathcal{Q})$ classes as either regular or intractable. Among the main results of [8] are the following findings.

(1) The class $\mathrm{ESO}(\exists^*\forall\exists^*)$ is regular. This theorem is the technically most involved result of [8]. Since this class is nonmonadic, it was not possible to exploit any of the ideas underlying Büchi's proof for proving it regular. The main difficulty consists in the fact that relations of higher arity may connect elements of a string that are very distant from one another; it was not *a priori* clear how a finite state automaton could guess such connections and check their global consistency. To solve this problem, new combinatorial methods (related to hypergraph transversals) were developed.

Interestingly, model checking for the fragment $\mathrm{ESO}(\exists^*\forall\exists^*)$ is NP-complete over *graphs*. For example, the well-known set-splitting problem can be expressed in it. Thus the fact that our input structures are monadic *strings* is essential (just as for MSO).

(2) The class $\mathrm{ESO}(\exists^*\forall\forall)$ is regular. The regularity proof for this fragment is easier but also required the development of new techniques (more of logical than of combinatorial nature). Note that model checking for this class, too, is NP-complete over graphs.

(3) Any class $\mathrm{ESO}(\mathcal{Q})$ not contained in the union of $\mathrm{ESO}(\exists^*\forall\exists^*)$ and $\mathrm{ESO}(\exists^*\forall\forall)$ is not regular.

Thus $\mathrm{ESO}(\exists^*\forall\exists^*)$ and $\mathrm{ESO}(\exists^*\forall\forall)$ are the *maximal regular standard prefix classes*. The unique maximal (general) regular ESO-prefix class is the union of these two classes, i.e, $\mathrm{ESO}(\exists^*\forall\exists^*) \cup \mathrm{ESO}(\exists^*\forall\forall) = \mathrm{ESO}(\exists^*\forall(\forall \cup \exists^*))$.

As shown in [8], it turns out that there are three minimal nonregular ESO-prefix classes, namely the standard prefix classes $\mathrm{ESO}(\forall\forall\forall)$, $\mathrm{ESO}(\forall\forall\exists)$, and $\mathrm{ESO}(\forall\exists\forall)$. All these classes express nonregular languages by sentences whose list of second-order variables consists of a single binary predicate variable.

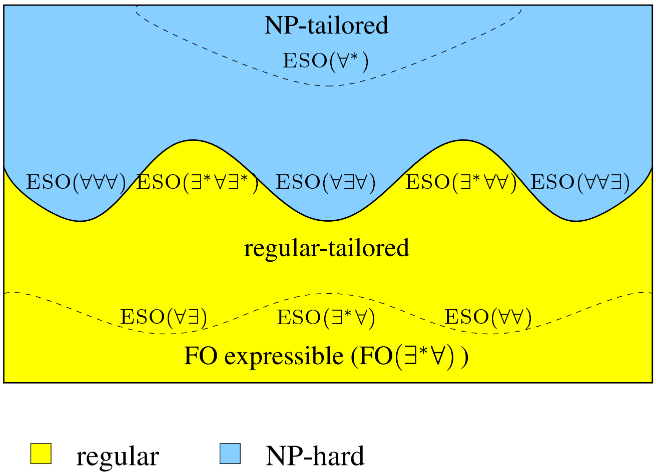

Fig. 1. Complete picture of the ESO-prefix classes on finite strings

Thus, 1.-3. give a *complete characterization* of the regular $\mathrm{ESO}(\mathcal{Q})$ classes.
(4) The following dichotomy theorem is derived: Let $\mathrm{ESO}(\mathcal{Q})$ be any prefix class. Then, either $\mathrm{ESO}(\mathcal{Q})$ is regular, or $\mathrm{ESO}(\mathcal{Q})$ expresses some NP-complete language. This means that model checking for $\mathrm{ESO}(\mathcal{Q})$ is either possible by a deterministic finite automaton (and thus in constant space and linear time) or it is already NP-complete. Moreover, for all NP-complete classes $\mathrm{ESO}(\mathcal{Q})$, NP-hardness holds already for sentences whose list of second-order variables consists of a single binary predicate variable. There are no fragments of intermediate difficulty between REG and NP.
(5) The above dichotomy theorem is paralleled by the solvability of the finite satisfiability problem for ESO (and thus FO) over strings. As shown in [8], over finite strings satisfiability of a given $\mathrm{ESO}(\mathcal{Q})$ sentence is decidable iff $\mathrm{ESO}(\mathcal{Q})$ is regular.
(6) In [8], a precise characterization is given of those prefix classes of ESO which are equivalent to MSO over strings, i.e. of those prefix fragments that *capture* the class REG of regular languages. This provides new logical characterizations of REG. Moreover, in [8] it is established that any regular ESO-prefix class is over strings either equivalent to full MSO, or is contained in first-order logic, in fact, in $\mathrm{FO}(\exists^*\forall)$.

It is further shown that $\mathrm{ESO}(\forall^*)$ is the unique minimal ESO prefix class which captures NP. The proof uses results in [14, 7] and well-known hierarchy theorems.

The main results of [8] are summarized in Figure 1. In this figure, the ESO-prefix classes are divided into four regions. The upper two contain all classes that express nonregular languages, and thus also NP-complete languages. The uppermost region contains those classes which capture NP; these classes are called NP-*tailored*. The region next below, separated by a dashed line, contains those classes which can express some NP-hard languages, but not all languages in NP. Its bottom is constituted by the minimal nonregular classes, ESO($\forall\forall\forall$), ESO($\forall\exists\forall$), and ESO($\forall\forall\exists$). The lower two regions contain all regular classes. The maximal regular standard prefix classes are ESO($\exists^*\forall\exists^*$) and ESO($\exists^*\forall\forall$). The dashed line separates the classes which capture REG(called *regular-tailored*), from those which do not; the expressive capability of the latter classes is restricted to first-order logic (in fact, to FO($\exists^*\forall$)) [8]. The minimal classes which capture REG are ESO($\forall\exists$) and ESO($\forall\forall$).

As an example for an NP-complete language, we consider a natural string encoding of the classical satisfiability problem (SAT) and show how it can be expressed by a sentence from the minimal non-regular class ESO($\forall\forall\forall$).

Example 3.1 ([8]). For expressing SAT, we encode instances $\mathcal{C} = \{C^1, \dots C^m\}$, where the C^i are clauses on propositional variables $p_1, \dots, p_n$, to strings $enc(\mathcal{C})$ over the alphabet $A = \{\texttt{0}, \texttt{1}, \texttt{+}, \texttt{-}, \texttt{[}, \texttt{]}, \texttt{(}, \texttt{)}\}$ as follows. The variables p_i, $1 \leq i \leq n$ are encoded by binary strings of length $\lceil \log n \rceil$. Each string encoding p_i is enclosed by parentheses '`(`','`)`'. The polarity of a literal $p_i/\neg p_i$ is represented by the letter '`+`' or '`-`', respectively, which immediately follows the closing parenthesis '`)`' of the encoding of p_i. A clause is encoded as a sequence of literals which is enclosed in square brackets '`[`','`]`'. Without loss of generality, we assume that $\mathcal{C} \neq \emptyset$ and that each clause $C^i \in \mathcal{C}$ contains at least one literal.

For example, $\mathcal{C} = \{\, \{p, q, \neg r\},\, \{\neg p, \neg q, r\}\,\}$ is encoded by the following string:

$$enc(\mathcal{C}) = \texttt{[(00)+(01)+(10)-][(00)-(01)-(10)+]} \ .$$

Here, p, q, r are encoded by the binary strings 00, 01, 10, respectively. Clearly, $enc(\mathcal{C})$ is obtainable from any standard representation of $\mathcal{C}$ in logspace.

In what follows, we will use the formulas

$$eqcol(x, y) = \bigvee_{\ell \in A} (C_\ell(x) \wedge C_\ell(y)), \tag{1}$$

$$varenc(x) = C_{\texttt{(}}(x) \vee C_{\texttt{0}}(x) \vee C_{\texttt{1}}(x) \vee C_{\texttt{)}}(x) \tag{2}$$

which state that the string has at positions x and y the same letter from A and that x is a letter of a variable encoding, respectively.

Then, satisfiability of $\mathcal{C}$ is expressed through the following $\Sigma^1_1(\forall\forall\forall)$ sentence:

$$\Phi = \exists V \exists G \exists R \exists R' \forall x \forall y \forall z.\, \varphi(x, y, z),$$

where G and V are unary, R' and R are binary, and $\varphi(x, y, z)$ is the conjunction of the following quantifier-free formulas φ_G, φ_V, φ_R, and $\varphi_{R'}$:

$$\varphi_G = \varphi_{G,1} \wedge \varphi_{G,2} \wedge \varphi_{G,3},$$

where

$$\begin{aligned}
\varphi_{G,1} &= \Big(C_{\texttt{[}}(x) \to \neg G(x)\Big) \wedge \Big(C_{\texttt{]}}(x) \to G(x)\Big),\\
\varphi_{G,2} &= \Big(Succ(x, y) \wedge \neg C_{\texttt{[}}(y) \wedge \neg C_{\texttt{)}}(y)\Big) \to \Big(G(y) \leftrightarrow G(x)\Big),\\
\varphi_{G,3} &= \Big(C_{\texttt{)}}(y) \wedge Succ(x, y) \wedge Succ(y, z)\Big) \to\\
&\qquad \Big(G(y) \leftrightarrow \Big[G(x) \vee (V(y) \wedge C_{\texttt{+}}(z)) \vee (\neg V(y) \wedge C_{\texttt{-}}(z))\Big]\Big);
\end{aligned}$$

next,

$$
\begin{aligned}
\varphi_V &= \Big(C_{\texttt{)}}(x) \wedge C_{\texttt{)}}(y) \wedge R(x,y)\Big) \rightarrow \Big(V(x) \leftrightarrow V(y)\Big),\\
\varphi_R &= \Big[R(x,y) \rightarrow (\mathit{eqcol}(x,y) \wedge \mathit{varenc}(x))\Big] \wedge\\
&\quad \Big[(C_{\texttt{(}}(x) \wedge C_{\texttt{(}}(y)) \rightarrow R(x,y)\Big] \wedge\\
&\quad \Big[\Big(\neg C_{\texttt{(}}(x) \wedge \mathit{Succ}(z,x)\Big) \rightarrow \Big(R(x,y) \leftrightarrow (R'(z,y) \wedge \mathit{eqcol}(x,y))\Big)\Big], \quad (3)
\end{aligned}
$$

and

$$
\varphi_{R'} = \mathit{Succ}(z,y) \rightarrow \Big[R'(x,y) \leftrightarrow \Big(R(x,z) \wedge \neg C_{\texttt{)}}(z)\Big)\Big]
$$

Informally, the predicate V assigns a truth value to each occurrence of a variable p_i in $\mathcal{C}$, which is given by the value of V at the closing parenthesis of this occurrence. $\mathcal{C}$ is satisfiable, precisely if there exists such a V which assigns all occurrences of p_i the same truth value and satisfies every clause. This is checked using G, R, and R'.

The predicate G is used for checking whether each clause $C \in \mathcal{C}$ is satisfied by the assignment V. To this end, G is set to false at the '$\texttt{[}$' marking the beginning of C, and set to true at the '$\texttt{]}$' marking the end of C by formula $\varphi_{G,1}$; the formulas $\varphi_{G,2}$ and $\varphi_{G,3}$ propagate the value of G from a position x in the clause representation to the successor position y, where the value switches from false to true if y marks the sign of a literal which is satisfied by V; the conjunct $\neg C_{\texttt{[}}$ in $\varphi_{G,}$ prohibits the transfer of G from the end of C to the beginning of the next clause, for which G must be initialized to false.

The predicate R is used to identify the closing parentheses '$\texttt{)}$' of the representations of occurrences of the same variables. For positions x and y at which the string W has letter '$\texttt{)}$', the predicate $R(x,y)$ is true precisely if x and y mark the end of the same variable name. This is used in the formula φ_V, which then simply states that V assigns every occurrence of a variable p in $\mathcal{C}$ the same truth value.

The purpose of φ_R and $\varphi_{R'}$ is to ensure that $R(x,y)$ has for positions x and y which mark the ends of occurrences of the same variable the desired meaning. This is accomplished inductively. Intuitively, $R(x,y)$ expresses that x and y have the same color, must be part of the encodings $o_x = \texttt{(} \cdots \texttt{)}$ and $o_y = \texttt{(} \cdots \texttt{)}$ of variable occurrences in $\mathcal{C}$, and are at the same distance from the beginnings of these encodings. By reference to the predicate R', $R(x,y)$ furthermore expresses that this also holds for predecessors x^- and y^- of x in o_x and y in o_y, respectively. The predicate R' is needed since we may use only three first-order variables, while the natural statement of the inductive property of R uses four first-order variables.

As shown in [8], $\mathit{enc}(\mathcal{C}) \models \Phi$ iff $\mathcal{C}$ is satisfiable. Furthermore, using packing techniques, Φ can be replaced by an equivalent ESO($\forall\forall\forall$) sentence of the form $\Phi' = \exists R \forall x \forall y \forall z \varphi(x,y,z)$, where R is binary.

4 SO over Strings

In this section, we investigate an extension of the results in the previous section from ESO to full second-order logic over strings. Our focus of attention will be the *SO* prefix classes which are regular vs. those which are not. In particular, we are interested to know how adding second-order variables affects regular fragments.

Generalizing Fagin's theorem, Stockmeyer [16] has shown that full SO captures the polynomial hierarchy (PH). Second-order variables turn out to be quite powerful. In fact, already two first-order variables, a single binary predicate variable, and further monadic predicate variables are sufficient to express languages that are complete for the levels of PH.

4.1 Regular fragments

Let us first consider possible generalizations of regular ESO prefix classes. It is easy to see that every ESO sentence in which only a single first-order variable occurs is equivalent to a monadic second-order sentence. This can be shown by eliminating predicate variables of arity > 1 through introducing new monadic predicate variables. The result generalizes to sentences in full SO with the same first-order part, which can be shown using the same technique. Thus, the class $\Sigma^1_k(\{\exists, \forall\})$ is regular for every $k \geq 0$.

Similarly, it can be shown that every ESO sentence in which only two first-order variables occur is equivalent to a MSO sentence [11]. However, as follows from the results of the next section, this does not generalize beyond ESO. Nonetheless, there are still higher-rank Σ^1_k fragments with two first-order variables, in particular Σ^1_2 prefix classes, which are regular. Here, similar elimination techniques as in the case of a single first-order variable are applicable. The following lemmas are useful for this purpose.

Lemma 4.1 ([13]). *Every formula $\exists \mathbf{R} \exists y_1 \ldots \exists y_k.\varphi$, where φ is quantifier-free, is equivalent to a formula $\exists y_1 \ldots \exists y_k.\psi$, where ψ is quantifier-free.*

Proposition 4.1. *Every formula in $\Sigma^1_k(\exists^j)$, where $k \geq 1$ is odd, is equivalent to some formula in $\Sigma^1_{k-1}(\exists^j)$, and every formula in $\Sigma^1_k(\forall^j)$, where $k \geq 2$ is even, is equivalent to some formula in $\Sigma^1_{k-1}(\forall^j)$.*

A generalization of the proof of Theorem 9.1 in [8] yields the following.

Theorem 4.1. *Over strings, $\Sigma^1_2(\forall\forall) = \mathrm{MSO}$.*

Proof. By the previous proposition, every formula $\exists \mathbf{S} \forall \mathbf{R} \forall x \forall y \varphi$ in $\Sigma^1_2(\forall\forall)$ is equivalent to some formula in $\Sigma^1_1(\forall\forall)$ (rewrite $\exists \mathbf{S} \forall \mathbf{R} \forall x \forall y \varphi$ to $\exists \mathbf{S} \neg(\exists \mathbf{R} \exists x \exists y \neg\varphi)$). Thus, $\Sigma^1_2(\forall\forall) \subseteq \mathrm{MSO}$. On the other hand, over strings $\mathrm{MSO} = \Sigma^1_1(\forall\forall)$, which is trivially included in $\Sigma^1_2(\forall\forall)$. □

A similar elimination of second-order variables is possible in a slightly more general case.

Lemma 4.2 ([8]). *Every formula $\Phi = \exists \mathbf{R} \exists y_1 \ldots \exists y_k \forall x.\varphi$, where φ is quantifier-free, is equivalent to a finite disjunction of first-order formulas $\exists y_1 \ldots \exists y_k \forall x.\psi$, where ψ is quantifier-free.*

The lemma was proven in [8] for strings, but the proof easily generalizes to arbitrary structures.

Let $\Sigma^1_k(\bigvee \mathcal{Q})$ (resp., $\Sigma^1_k(\bigwedge \mathcal{Q})$) denote the class of Σ^1_k sentences where the first-order part is a finite disjunction (resp., conjunction) of prefix formulas with quantifier in $\mathcal{Q}$. Then we have the following.

Proposition 4.2. *Every formula in $\Sigma^1_k(\exists^j\forall)$, where $k \geq 1$ is odd and $j \geq 0$, is equivalent to some formula in $\Sigma^1_{k-1}(\bigvee \exists^j\forall)$, and every formula in $\Sigma^1_k(\forall^j\exists)$, where $k \geq 2$ is even and $j \geq 1$, is equivalent to some formula in $\Sigma^1_{k-1}(\bigwedge \forall^j\exists)$.*

Proof. The first part follows immediately from Lemma 4.2, by which also the second is easily shown. Let $\Phi = \exists \mathbf{S} \cdots \forall \mathbf{R} \forall \mathbf{y} \exists x.\varphi(\mathbf{y}, x)$ be a sentence in $\Sigma^1_k(\forall^j\exists)$ for even $k \geq 2$. Then, Φ is equivalent to

$$\exists \mathbf{S} \cdots \neg(\exists \mathbf{R} \exists \mathbf{x} \forall x.\neg\varphi(\mathbf{y}, x))$$

By Lemma 4.2, this formula is equivalent to

$$\exists \mathbf{S} \cdots \neg(\bigvee_{i=1}^{n} \exists \mathbf{x} \forall x.\psi_i(\mathbf{y}, x))$$

where each $\psi_i(\mathbf{y}, x)$ is quantifier-free. By moving negation inside, we obtain the result. $\square$

Corollary 4.1. *Over strings, $\Sigma^1_2(\forall\exists) = \mathrm{MSO}$.*

Proof. By the previous proposition, every sentence Φ in $\Sigma^1_2(\forall\exists)$ is equivalent to some sentence Φ' in $\Sigma^1_1(\bigwedge \forall\exists)$. We may rewrite Φ' as a sentence Φ'' of form $\exists \mathbf{R} \forall x \exists \mathbf{y} \varphi$, where φ is quantifier-free. Recall that $\mathbf{y}$ denotes a tuples of variables. By Corollary 8.4 in [8] , Φ'' is in MSO. Thus $\Sigma^1_2(\forall\exists) \subseteq \mathrm{MSO}$. The reverse inclusion holds since, as well-known, over strings $\mathrm{MSO} = \Sigma^1_1(\forall\exists)$. $\square$

4.2 Non-regular fragments

$\Sigma^1_2(Q)$ where $|Q| \leq 2$. While for the first-order prefixes $Q = \forall\exists$ and $\mathcal{Q} = \forall\forall$, regularity of $\mathrm{ESO}(Q)$ generalizes to Σ^1_2, this is not the case for the other two-variables prefixes $\exists\forall$ and $\exists\exists$.

Theorem 4.2. $\Sigma^1_2(\exists\forall)$ *is nonregular.*

For a proof of this result, we consider the following example.

Example 4.1. Let $A = \{a, b\}$ and consider the following sentence:

$$\Phi = \exists R \forall X \exists x \forall y.[C_a(y) \vee (C_a(x) \wedge (X(y) \leftrightarrow R(x, y))]$$

Informally, this sentence is true for a string W, just if the number of b's in W (denoted $\#b(W)$) is at most logarithmic in the number of a's in W (denoted $\#a(W)$). More formally, $L(\Phi) = \{w \in \{a,b\}^* \mid \#b(W) \leq \log \#a(W)\}$; by well-known results, this language is not regular.

Indeed, for any assignment to the monadic variable X, there must be a position i colored with C_a in the string such that, for every position j colored with C_b, an arc goes from i to j just if X is true for j. For assignments to X which are different on the positions colored with C_b, different positions i must be chosen. If $\#b(W) \leq \log \#a(W)$, this choice can be satisfied; otherwise, there must exist two different assignments to X which require that, for some position i colored with C_a and some position j colored with C_b, both $R(i,j)$ and $\neg R(i,j)$ hold, which is not possible.

Lemma 4.3. *Every first-order formula $\forall x \forall y \varphi(x,y)$, where φ is quantifier-free, is equivalent to some first-order formula $\forall X \exists x \forall y \psi(x,y)$ where X is a fresh monadic variable and φ is quantifier-free.*

Proof. (Sketch) Define

$$\psi(x,y) = (X(x) \wedge \varphi(x,y)) \vee (\neg X(x) \wedge \neg X(y))$$

Then, for every structure and assignment to the free variables, $\forall x \forall y \varphi(x,y)$ (resp., $\forall x \exists y \varphi(x,y)$) is equivalent to $\forall X \exists x \forall y \psi(x,y)$. $\square$

Using the lemma, we establish the following result about the expressiveness of $\Sigma^1_2(\exists\forall)$.

Theorem 4.3. *Every $\Phi \in \Sigma^1_2(\bigwedge\{\exists\forall, \forall\forall\})$ is equivalent to some $\Sigma^1_2(\exists\forall)$ formula.*

Proof. (Sketch) Let $\Phi = \exists \mathbf{R} \forall \mathbf{T} \varphi$, where $\varphi = \bigwedge_{i=1}^{n} \varphi_i$ is a conjunction of prefix first-order formulas φ_i with prefix from $\{\exists\forall, \forall\exists, \forall\forall\}$. By Lemma 4.3, each φ_i of the form $\forall x \forall y \varphi'_i(x,y)$ (resp. $\forall x \exists y \varphi(x,y)$), can be equivalently replaced by a formula $\exists x \forall y \psi_i(x,y,X_i)$ and adding $\forall X_i$ to the universal second-order part of Φ, where X_i is a fresh monadic variable. Then, rewrite the resulting conjunction $\bigwedge_{i=1}^{n} \exists x \forall y \alpha_i(x,y)$ in the first-order part to $\forall Z_1 \cdots \forall Z_n \exists x \forall y (\beta \rightarrow (\bigwedge_{i=1}^{n} Z_i \rightarrow \alpha(x,y)))$, where the Z_i are fresh Boolean variables, and β states that exactly one out of $Z_1, \ldots, Z_n$ is true. The resulting formula Φ', which is in $\Sigma^1_2(\exists\forall)$, is equivalent to Φ. $\square$

For the class $\Sigma^1_2(\exists\exists)$, our proof of non-regularity is more involved. The proof of the following lemma shows how to emulate universal first-order quantifiers using universal second-order quantifiers and existential quantifiers over strings.

Lemma 4.4. *Over strings, every universal first-order formula $\forall \mathbf{x} \varphi(\mathbf{x})$ which contains no predicates of arity > 2 is equivalent to some $\Pi^1_1(\exists\exists)$ formula.*

Proof. (Sketch) The idea is to emulate the universal quantifier $\forall x_i$ for every x_i from $\mathbf{x} = x_1, \ldots, x_k$ using a universally quantified variable S_i which ranges over singletons, and express "x_i" by "$\exists x_i . S_i(X_i)$." Then, $\forall \mathbf{x} \varphi(\mathbf{x})$ is equivalent to $\forall \mathbf{S} \exists \mathbf{x} \bigwedge_{i=1}^{k} S_i(x_k) \wedge \varphi(\mathbf{x})$.

We can eliminate all existential variables $\mathbf{x}$ but two in this formula as follows. Rewrite the quantifier-free part in CNF $\bigwedge_{i=1}^{\ell} \delta_i(\mathbf{x})$, where each $\delta_i(\mathbf{x})$ is a disjunction of literals. Denote by $\delta_i^{j,j'}(x_j, x_{j'})$ the clause obtained from $\delta_i(\mathbf{x})$ by removing every literal which contains some variable from $\mathbf{x}$ different from x_j and $x_{j'}$. Since no predicate in φ has arity > 2, formula $\forall \mathbf{x} \varphi(\mathbf{x})$ is equivalent to the formula

$$\forall \mathbf{S} \exists \mathbf{x} \bigwedge_{i=1}^{\ell} \left(\bigvee_{j \neq j'} \exists x \exists y \delta_i^{j,j'}(x, y) \right).$$

The conjunction $\bigwedge_{i=1}^{\ell}$ can be simulated similarly as in the proof of Theorem 4.3, by using universally quantified Boolean variables $Z_1, \ldots, Z_\ell$ and a control formula β which states that exactly one out of $Z_1, \ldots, Z_n$ is true. By pulling existential quantifiers, we thus obtain

$$\forall \mathbf{S} \exists x \exists y \gamma,$$

where

$$\gamma = \beta \rightarrow \bigwedge_{i=1}^{\ell} \left(Z_i \rightarrow \bigvee_{j \neq j'} \delta_i^{j,j'}(x, y) \right).$$

Thus, it remains to express the variables S_i ranging over singletons. For this, we use a technique to express S_i as the difference $X_{i,1} \setminus X_{i,2}$ of two monadic predicates $X_{i,1}$ and $X_{i,2}$ which describe initial segments of the string. Fortunately, the fact that $X_{i,1}$ and $X_{i,2}$ are *not* initial segments or their difference is *not* a singleton can be expressed by a first-order formula $\exists x \exists y \psi_i(x, y)$, where ψ_i is quantifier-free, by using the successor predicate. Thus, we obtain

$$\forall \mathbf{X}_1 \mathbf{X}_2 \left(\bigvee_{i=1}^{k} \exists x \exists y \psi_i(x, y) \right) \vee \exists x \exists y \gamma^*,$$

where γ^* results from γ by replacing each S_i with $X_{i,1}$ and $X_{i,2}$. By pulling existential quantifiers, we obtain a $\Sigma_2^1(\exists\exists)$ formula, as desired. $\square$

For example, the lemma is applicable to the sentence Φ in Example 3.1. Thus,

Theorem 4.4. $\Sigma_2^1(\exists\exists)$ *is nonregular.*

We remark that a similar argument as in the proof of Lemma 4.4 can be used to show that over strings, every universal first-order formula $\forall \mathbf{x} \varphi(\mathbf{x})$ which contains no predicates of arity > 2 is equivalent to some $\Pi_1^1(\bigvee \exists\forall)$ formula. Consequently, the SAT encoding in Example 3.1 can be expressed in $\Sigma_2^1(\bigvee \exists\forall)$.

$\Sigma_2^1(Q)$ where $|Q| > 2$. By the result of the previous subsection, and the results in Section 3, we can derive that there is no $\Sigma_2^1(Q)$ prefix class where Q contains more than two variables which is regular.

Indeed, Theorem 4.2 implies this for every prefix Q which contains $\exists$ followed by $\forall$, and Theorem 4.4 implies this for every prefix Q which contains at least two existential quantifiers. For the remaining minimal prefixes $Q \in \{\forall\forall\forall, \forall\forall\exists\}$, non-regularity of $\Sigma_2^1(Q)$ follows from the results summarized in Figure 1. Thus,

Theorem 4.5. *$\Sigma_2^1(Q)$ is nonregular for any prefix Q such that $|Q| > 2$.*

$\Sigma_k^1(Q)$ where $k > 2$. Let us now consider the higher fragments of SO over strings. The question is whether any of the regular two-variable prefixes $Q \in \{\forall\forall, \forall\exists\}$ for $\Sigma_2^1(Q)$ survives. However, as we shall see this is not the case.

Since $\Pi_2^1(\forall\forall)$ is contained in $\Sigma_3^1(\forall\forall)$, it follows from Theorem 4.4 that $\Sigma_3^1(\forall\forall)$ is nonregular. For the remaining class $\Sigma_3^1(\forall\exists)$, we use the following result, which simulates an existential first-order quantifier using an existential second-order quantifier and a first-order universal quantifier.

Lemma 4.5. *Suppose a signature contains constants. Then every formula $\exists x \exists y \varphi$ in which some constant c occurs is equivalent to some formula $\exists X \forall x \exists y \psi$, where X is a fresh monadic variable.*

Proof. Write ψ as $(x = c \to \varphi(x, y) \vee X(y)) \wedge (X(x) \to \varphi(x, y))$. □

We thus easily obtain the following result.

Theorem 4.6. *Over strings, $\Sigma_k^1(\exists\exists) \subseteq \Sigma_k^1(\forall\exists)$ for every odd k, and $\Sigma_k^1(\forall\forall) \subseteq \Sigma_k^1(\exists\forall)$ for every even $k \geq 2$.*

Thus, combined with Theorem 4.4, we obtain that $\Sigma_3^1(\forall\exists)$ is nonregular.

We note that under very general conditions a universal first-order quantifier can be replaced by a universal second-order quantifier and an existential first-order quantifier.

Lemma 4.6. *Let φ be a prenex first-order formula (possibly with free first and second order variables) of the form $\varphi = \forall y\, Q \forall z \alpha(y, \ldots, z)$, where Q is a (possible empty) block of first-order quantifiers. Then, φ is equivalent to*

$$\psi = \forall Y \exists y\, Q \forall z [(Y(y) \wedge \alpha) \vee (\neg Y(y) \wedge \neg Y(z))],$$

where Y is a fresh monadic SO variable.

Using the lemma, we obtain the following result.

Theorem 4.7. *Let $P_1 \in \{\forall\}^*$ and $P_2 \in \{\exists, \forall\}^* \forall \{\exists, \forall\}^*$ be first-order prefixes. Then,*

- *for any odd $k \geq 1$, $\Sigma_k^1(P_1 \forall P_2) \subseteq \Sigma_{k+1}^1(P_1 \exists P_2)$ and $\Pi_k^1(P_1 \forall P_2) \subseteq \Pi_k^1(P_1 \exists P_2)$,*
- *for any even $k \geq 2$, $\Sigma_k^1(P_1 \forall P_2) \subseteq \Sigma_k^1(P_1 \exists P_2)$, $\Pi_k^1(P_1 \forall P_2) \subseteq \Pi_{k+1}^1(P_1 \exists P_2)$.*

Thus, for example we obtain $\Sigma_2^1(\forall\forall\forall) \subseteq \Sigma_2^1(\exists\forall\forall)$, and by repeated application $\Sigma_2^1(\forall\forall\forall) \subseteq \Sigma_2^1(\exists\exists\forall)$. Since by the results in Figure 1, $\Sigma_2^1(\forall\forall\forall)$ is intractable, these fragments inherit intractability.

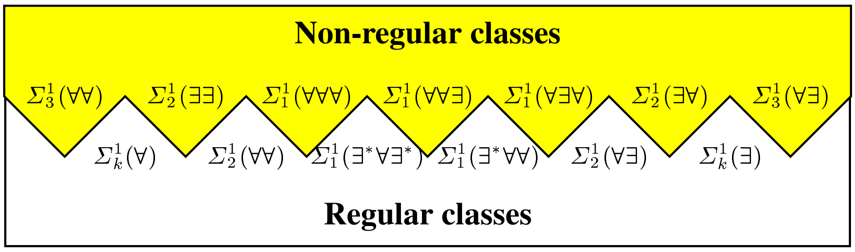

Fig. 2. Maximal regular and minimal non-regular SO prefix classes on strings.

4.3 Summary of new results

The maximal standard Σ^1_k prefix classes which are regular and the minimal standard Σ^1_k prefix classes which are non-regular are summarized in Figure 2.

Thus, no $\Sigma^1_k(Q)$ fragment, where $k \geq 3$ and Q contains at least two variables, is regular, and for $k = 2$, only two such fragments ($Q = \exists\forall$ or $Q = \forall\exists$) are regular. Note that Grädel and Rosen have shown [11] that $\Sigma^1_1(\mathrm{FO}^2)$, i.e., existential second order logic with two first-order variables, is over strings regular. By our results, $\Sigma^1_k(\mathrm{FO}^2)$, for $k \geq 2$, is intractable.

Figure 3 shows inclusion relationship between the classes $\Sigma^1_2(Q)$ where Q contains 2 quantifiers. Similar relationships hold for $\Sigma^1_k(Q)$ classes. Furthermore, by Proposition 4.2 we have that $\Sigma^1_2(\forall\exists) = \Sigma^1_1(\bigwedge\forall\exists)$ and $\Sigma^1_3(\exists\forall) = \Sigma^1_2(\bigvee\exists\forall)$.

$$
\begin{array}{ccccc}
\Sigma^1_2(\exists\forall) & & & & \Sigma^1_2(\exists\exists) \\
& \supseteq & & \subseteq & \\
& \Sigma^1_2(\forall\forall) & = & \Sigma^1_2(\forall\exists) &
\end{array}
$$

Fig. 3. Semantic inclusion relations between $\Sigma^1_2(Q)$ classes over strings, $|Q| = 2$

As for the complexity of model checking, the results from [8] and above imply that deciding whether $W \models \Phi$ for a fixed formula Φ and a given string W is intractable for all prefix classes $\Sigma^1_k(Q)$ which are (syntactically) not included in the maximal regular prefix classes shown in Figure 2, with the exception $\Sigma^1_2(\exists\forall)$, for which the tractability is currently open.

In more detail, the complexity of SO over strings increases with the number of SO quantifier alternations. In particular, $\Sigma^1_2(\exists\exists)$ can express Σ^p_2-complete languages, where $\Sigma^p_2 = \mathrm{NP}^{\mathrm{NP}}$ is from the second level of the polynomial hierarchy. This can be shown by encoding, generalizing the SAT encoding in Example 3.1, quantified boolean formulas (QBFs) of the form $\exists p_1 \cdots \exists p_n \forall q_1 \cdots \forall q_m \neg\varphi$, where φ is a propositional CNF over the atom $p_1, \ldots, p_n, q_1, \ldots, q_m$, to model checking for $\Sigma^1_2(\exists\exists)$. Roughly, we can mark in the string the occurrences of the variables q_i with an additional predicate, and represent truth assignments to them by a monadic variable V'. The sentence Φ in Example 3.1 is rewritten to

$$\Psi = \exists R \exists R' \exists V \forall V' [\alpha_1 \wedge (\alpha_2 \vee \alpha_3)]$$

where α_1 is a universal first-order formula which defines proper R, R' and V using φ_R, $\varphi_{R'}$, and φ_V; α_2 is a $\exists\exists$-prenex first-order formula which states that V' assigns two different occurrences of some universally quantified atom q_i different truth values; and α_3 states that the assignment to $p_1, \ldots, p_n$, $q_1, \ldots, q_m$ given by V and V' violates φ. The latter can be easily checked by a finite state automaton, and thus is expressible as a monadic $\Pi^1_1(\exists\exists)$ sentence. Since Ψ contains no predicate of arity > 2, by applying the techniques in the previous section we can rewrite it to an equivalent $\Sigma^1_2(\exists\exists)$ sentence.

Thus, model checking for the fragment $\Sigma^1_2(\exists\exists)$ is Σ^p_2-complete. Other fragments of Σ^1_2 have lower complexity; for example, Proposition 4.2 and Figure 1 imply that $\Sigma^1_2(\forall^*\exists)$ is NP-complete. By adding further second-order variables, we can encode evaluating Σ^p_k-complete QBFs into $\Sigma^1_k(Q)$, where $Q = \exists\exists$ if $k > 2$ is even and $Q = \forall\forall$ if $k > 1$ is odd. Note that, after eliminating R' from the sentence Φ in Example 3.1 as described in [8], the sentence contains only a single binary predicate variable.

5 ESO over Graphs: A Short Account

In this section, we briefly describe the main results of [10], where the computational complexity of ESO-prefix classes of is investigated and completely characterized in three contexts: over (1) directed graphs, (2) undirected graphs with self-loops, and (3) undirected graphs without self-loops. A main theorem of [10] is that a *dichotomy* holds in these contexts, that is to say, each prefix class of ESO either contains sentences that can express NP-complete problems or each of its sentences expresses a polynomial-time solvable problem. Although the boundary of the dichotomy coincides for 1. and 2. (which we refer ot as *general graphs* from now on), it changes if one moves to 3. The key difference is that a certain prefix class, based on the well-known *Ackermann class*, contains sentences that can express NP-complete problems over general graphs, but becomes tractable over undirected graphs without self-loops. Moreover, establishing the dichotomy in case 3. turned out to be technically challenging, and required the use of sophisticated machinery from graph theory and combinatorics, including results about graphs of bounded tree-width and Ramsey's theorem.

In [10], a special notation for ESO-prefix classes was used in order to describe the results with the tightest possible precision involving both the number of SO quantifiers and their arities.[2] Expressions in this notation are built according to the following rules:

- E (resp., E_i) denotes the existential quantification over a single predicate of arbitrary arity (arity $\leq i$).
- a (resp., e) denotes the universal (existential) quantification of a single first-order variable.

[2] For ESO over strings [8], the same level of precision was reached with simpler notation.

• If η is a quantification pattern, then η^* denotes all patterns obtained by repeating η zero or more times.

An expression $\mathcal{E}$ in the special notation consists of a string of ESO quantification patterns (E-patterns) followed by a string of first-order quantification patterns (a or e patterns); such an expression represents the class of all prenex ESO-formulas whose quantifier prefix corresponds to a (not-necessarily contiguous) substring of $\mathcal{E}$.

For example, $E_1^* eaa$ denotes the class of formulas $\exists P_1 \cdots \exists P_r \exists x \forall y \forall z \varphi$, where each P_i is monadic, x, y, and z are first-order variables, and φ is quantifier-free.

A prefix class C is NP-*hard* on a class $\mathcal{K}$ of relational structures, if some sentence in C expresses an NP-hard property on $\mathcal{K}$, and C is *polynomial-time* (PTIME) on $\mathcal{K}$, if for each sentence $\Phi \in C$, model checking is polynomial. Furthermore, C is called *first-order* (FO), if every $\Phi \in C$ is equivalent to a first-order formula.

The first result of [10] completely characterizes the computational complexity of ESO-prefix classes on general graphs. In fact, the same characterization holds on the collection of all finite structures over any relational vocabulary that contains a relation symbol of arity ≥ 2. This characterization is obtained by showing (assuming P $\neq$ NP) that there are four *minimal* NP-hard and three *maximal* PTIME prefix classes, and that these seven classes combine to give complete information about all other prefix classes. This means that every other prefix either contains one of the minimal NP-hard prefix classes as a substring (and, hence, is NP-hard) or is a substring of a maximal PTIME prefix class (and, hence, is in PTIME). Figure 4 depicts the characterization of the NP-hard and PTIME prefix classes of ESO on general graphs.

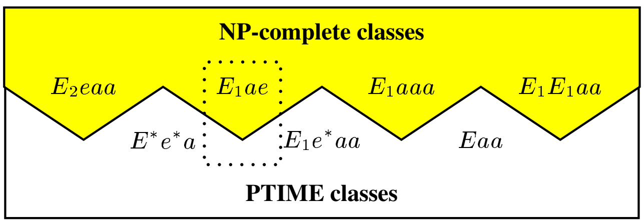

Fig. 4. ESO on arbitrary structures, directed graphs and undirected graphs with self-loops.

As seen in Figure 4, the four minimal NP-hard classes are E_2eaa, E_1ae, E_1aaa, and E_1E_1aa, while the three maximal PTIME classes are E^*e^*a, E_1e^*aa, and Eaa. The NP-hardness results are established by showing that each of the four minimal prefix classes contains ESO-sentences expressing NP-complete problems. For example, a SAT encoding on general graphs can be expressed by an E_1ae sentence. Note that the first-order prefix class ae played a key role in the study of the classical decision problem for fragments of first-order logic (see [2]). As regards the maximal PTIME classes, E^*e^*a is actually FO, while the model checking problem for fixed sentences in E_1e^*aa and Eaa is reducible to 2SAT and, thus, is in PTIME (in fact, in NL).

The second result of [10] completely characterizes the computational complexity of prefix classes of ESO on undirected graphs without self-loops. As mentioned earlier, it was shown that a dichotomy still holds, but its boundary changes. The key difference is that E^*ae turns out to be PTIME on undirected graphs without self-loops, while its subclass E_1ae is NP-hard on general graphs. It can be seen that interesting properties of graphs are expressible by E^*ae-sentences. Specifically, for each integer $m > 0$, there is a E^*ae-sentence expressing that a connected graph contains a cycle whose length is divisible by m. This was shown to be decidable in polynomial time by Thomassen [19]. E^*ae constitutes a maximal PTIME class, because all four extensions of E_1ae by any single first-order quantifier are NP-hard on undirected graphs without self-loops [10]. The other minimal NP-hard prefixes on general graphs remain NP-hard also on undirected graphs without self-loops. Consequently, over such graphs, there are seven minimal NP-hard and four maximal PTIME prefix classes that determine the computational complexity of all other ESO-prefix classes (see Figure 5).

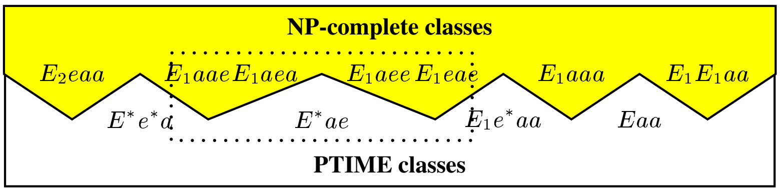

Fig. 5. ESO on undirected graphs without self-loops. The dotted boxes in Figures 4 and 5 indicate the difference between the two cases.

Technically, the most difficult result of [10] is the proof that E^*ae is PTIME on undirected graphs without self-loops. First, using syntactic methods, it is shown that each E^*ae-sentence is equivalent to some E_1^*ae-sentence. After this, it is shown that for each E_1^*ae-sentence the model checking problem over undirected graphs without self-loops is is equivalent to a natural coloring problem called the *saturation problem*. This problem asks whether there is a particular mapping from a given undirected graph without self-loops to a fixed, directed *pattern graph* P which is extracted from the E_1^*ae-formula under consideration. Depending on the labelings of cycles in P, two cases of the saturation problem are distinguished, namely *pure pattern graphs* and *mixed pattern graphs*. For each case, a polynomial-time algorithm is designed. In simplified terms and focussed on the case of connected graphs, the one for pure pattern graphs has three main ingredients. First, adapting results by Thomassen [19] and using a new graph coloring method, it is shown that if a E_1^*ae-sentence Φ gives rise to a pure pattern graph, then a fixed integer k can be found such that every undirected graph without self-loops and having tree-width bigger than k satisfies Φ. Second, Courcelle's theorem [5] is used by which model checking for MSO sentences is polynomial on graphs of bounded tree-width. Third, Bodlaender's result [1] is used that, for each fixed k, there is a polynomial-time algorithm to check if a given graph has tree-width at most k.

The polynomial-time algorithm for mixed pattern graphs has a similar architecture, but requires the development of substantial additional technical machinery, including a generalization of the concept of graphs of bounded tree-width. The results of [10] can be summarized in the following theorem.

Theorem 5.1. *Figures 4 and 5 provide a complete classification of the complexity of all ESO prefix classes on graphs.*

6 Conclusion and Open Problems

The aim of this paper was twofold, namely, to give a status report on the ambitious research programme of determining the complexity of prefix classes over finite structures, and to present new results consisting of a complete characterization of those prefix classes $\Sigma_k^1(\mathcal{Q})$ which over strings express regular languages only.

As for the first goal, we hope to have convinced the reader that this is an important and interesting research programme. In particular, let us stress that this programme is anything but boring. Many of the prefix classes we have analyzed so far represented mathematical challenges and required novel solution methods. Some of them could be solved with automata-theoretic techniques, others with techniques of purely combinatorial nature, and yet others required graph-theoretic arguments.

While we have traced the exact "regularity frontier" for the $\Sigma_k^1(\mathcal{Q})$ classes (see Fig. 2), we have not yet completely determined their tractability frontier with respect to model checking. A single result is missing, namely, the complexity of the nonregular class $\Sigma_2^1(\exists\forall)$. If model checking turns out to be NP-hard for it, then the tractability frontier coincides with the regularity frontier (just as for ESO, cf. [8]). If, on the other hand, model checking for $\Sigma_2^1(\exists\forall)$ turns out to be tractable, then the picture is slightly different. We plan to clarify this issue in the future. Moreover, it would be interesting to sharpen our analysis of the Σ_k^1 fragments over strings by studying the second-order quantification patterns at a finer level, taking into account the number of the second-order variables and their arities, as done for ESO over graphs in [10] (cf. Section 5).

Let us conclude this paper by pointing out a few interesting (and in our opinion important) issues that should eventually be settled.

- While the work on word structures concentrated so far on strings with a successor relation $Succ$, one should also consider the case where in addition a predefined linear order $<$ is available on the word structures, and the case where the successor relation $Succ$ is replaced by such a linear order. While for full ESO or SO, $Succ$ and $<$ are freely interchangeable, because either predicate can be defined in terms of the other, this is not so for many of the limited ESO-prefix classes. Preliminary results suggest that most of the results in this paper carry over to the $<$ case.
- Delineate the tractability/intractability frontier for all SO prefix classes over graphs.
- Study SO prefix classes over trees and other interesting classes of structures (e.g. planar graphs).
- The scope of [8] and of Section 3 in this paper are *finite* strings. However, infinite strings or ω-words are another important area of research. In particular, Büchi has shown that an analogue of his theorem (Proposition 2.1) also holds for ω-words [3]. For

an overview of this and many other important results on ω-words, we refer the reader to the excellent survey paper [18]. In this context, it would be interesting to see which of the results established so far survive for ω-words. For some results, such as the regularity of $\mathrm{ESO}(\exists^*\forall\forall)$ this is obviously the case since no finiteness assumption on the input word structures was made in the proof. For determining the regularity or nonregularity of some other clases such as $\mathrm{ESO}(\exists^*\forall\exists^*)$, further research is needed.

Acknowledgments. This work was supported by the Austrian Science Fund (FWF) Project Z29-INF.

References

1. H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.
2. E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.
3. J. R. Büchi. On a decision method in restriced second-order arithmetic. In E. Nagel et al., editor, *Proc. International Congress on Logic, Methodology and Philosophy of Science*, pp. 1–11, Stanford, CA, 1960. Stanford University Press.
4. J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.
5. B. Courcelle. The monadic second-order logic of graphs I: recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.
6. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.
7. T. Eiter, G. Gottlob, and Y. Gurevich. Normal forms for second-order logic over finite structures, and classification of NP optimization problems. *Annals of Pure and Applied Logic*, 78:111–125, 1996.
8. T. Eiter, G. Gottlob, and Y. Gurevich. Existential second-order logic over strings. *Journal of the ACM*, 47(1):77–131, 2000.
9. R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. M. Karp, editor, *Complexity of Computation*, pp. 43–74. AMS, 1974.
10. G. Gottlob, P. Kolaitis, and T. Schwentick. Existential second-order logic over graphs: charting the tractability frontier. In *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS 2000)* , pp. 664–674. IEEE Computer Society Press, 2000.
11. E. Grädel and E. Rosen. Two-variable descriptions of regularity. In *Proc. 14th Annual Symposium on Logic in Computer Science (LICS-99)* , pp. 14–23. IEEE CS Press, 1999.
12. N. Immerman. *Descriptive Complexity*. Springer, 1997.
13. P. Kolaitis and C. Papadimitriou. Some computational aspects of circumscription. *Journal of the ACM*, 37(1):1–15, 1990.
14. D. Leivant. Descriptive characterizations of computational complexity. *Journal of Computer and System Sciences*, 39:51–83, 1989.
15. J.-E. Pin. Logic on words. *Bulletin of the EATCS*, 54:145–165, 1994.
16. L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Comp. Sc.*, 3:1–22, 1977.
17. H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, 1994.
18. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 4. Elsevier Science Pub., 1990.
19. C. Thomassen. On the presence of disjoint subgraphs of a specified type. *Journal of Graph Theory*, 12:1, 101-111, 1988.
20. B. Trakhtenbrot. Finite automata and the logic of monadic predicates. *Dokl. Akad. Nauk SSSR*, 140:326–329, 1961.