# Second-Order Logic over Finite Structures – Report on a Research Programme*

Georg Gottlob

Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Wien, Austria
gottlob@dbai.tuwien.ac.at

**Abstract.** This paper reports about the results achieved so far in the context of a research programme at the cutting point of logic, formal language theory, and complexity theory. The aim of this research programme is to classify the complexity of evaluating formulas from different prefix classes of second-order logic over different types of finite structures, such as strings, graphs, or arbitrary structures. In particular, we report on classifications of second-order logic on strings and of existential second-order logic on graphs.

## 1 Introduction

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 Entscheidungsproblem (cf. [3]), 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 [5,4] and Trakhtenbrot [32] 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 [11] 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 [27,24,7, 16]).

* Most of the material contained in this paper stems, modulo editorial adaptations, from the much longer papers [9,10,12].

D. Basin and M. Rusinowitch (Eds.): IJCAR 2004, LNAI 3097, pp. 229–243, 2004.

While the classical research programme of determining the prefix characterizations of decidable fragments of first-order logic was successfully completed around 1984 (cf. [3]), until recently little was known on analogous problems on finite structures, in particular, on the tractability/intractability frontier of the model checking problem for prefix classes of second-order logic (SO), and in particular, of existential second order logic (ESO) over finite structures. In the late nineties, a number of scientists, including Thomas Eiter, Yuri Gurevich, Phokion Kolaitis, Thomas Schwentick, and the author started to attack this new research programme in a systematic manner.

By *complexity of a prefix class* $C$ we mean the complexity of the following model-checking problem: Given a fixed sentence $\Phi$ in $C$, decide for variable finite structures $A$ whether $A$ is a model of $\Phi$, 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*, *graphs*, or arbitrary finite relational structures (corresponding to relational datavases). 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, three systematic studies were carried out recently, that shed light on the prefix classes of the existential fragment ESO (also denoted by $\Sigma^1_1$) of second-order logic:

- In [9], the ESO prefix-classes over strings are exhaustively classified. In particular, the precise frontier between regular and nonregular classes is traced out, 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 [9] are briefly reviewed in Section 2.
- In [10] this line of research was continued by by systematically investigating the syntactically more complex prefix classes $\Sigma^1_k(\mathcal{Q})$ of second-order logic for each integer $k > 1$ and for each first-order quantifier prefix $\mathcal{Q}$. An exhaustive classification of the regular and nonregular prefix classes of this form was given, and complexity results for the corresponding model-checking problems were derived.
- In [12], 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 4.

### 1.1 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 indi-

vidual 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 $\Sigma_0^1(Q)$ is the set of all prenex first-order formulas (which may contain free variables and constants) with prefix $Q$, and for every $k \geq 0$, $\Sigma_{k+1}^1(Q)$ (resp., $\Pi_{k+1}^1$) is the set of all formulas $\exists \mathbf{R} \Phi$ (resp., $\forall \mathbf{R} \Phi$) where $\Phi$ is from $\Pi_k^1$ (resp., $\Sigma_k^1$). For any prefix set $\mathcal{Q}$, the class $\Sigma_k^1(\mathcal{Q})$ is the union $\Sigma_k^1(\mathcal{Q}) = \bigcup_{Q \in \mathcal{Q}} \Sigma_k^1(\mathcal{Q})$. We write also ESO for $\Sigma_1^1$. For example, $\mathrm{ESO}(\exists^* \forall \exists^*)$ is the class of all formulas $\exists \mathbf{R} \exists \mathbf{y} \forall x \exists \mathbf{z} \varphi$, where $\varphi$ 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 $W = \langle U, C_{a_1}^W, \ldots, C_{a_m}^W, Succ^W, min^W, max^W \rangle$, 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 $C_{a_i}^W$ 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 $C_{a_i}^W$ are pairwise disjoint and $\bigcup_i C_{a_i}^W = U$.
- $Succ^W$ 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 [8]. 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* $\Phi$ over the vocabulary $\sigma_A$ is a second-order formula whose only free variables are the predicate variables of the signature $\sigma_A$, 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 $\Phi$, 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 1.1.* 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:

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

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 $\Phi$ over $\sigma_A$ 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 $\sigma_A$ are regular.

Büchi [5] has shown the following fundamental theorem, which was independently found by Trakhtenbrot [32]. 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 1.1 (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 1.2.** *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 [11], we have the following.

**Proposition 1.3 (Fagin's Theorem).** ESO *captures* NP.

[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.

This theorem can be sharpened to various fragments of ESO. In particular, by Leivant's results [19,8], 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.

Before proceeding with the characterization of the regular languages by nonmonadic fragments of ESO, we would like to note that many papers cover either extensions or restrictions of MSO or REG, and cite some relevant results.

Lynch [20], has studied the logic over strings obtained from existential MSO by adding addition. He proved that model checking for this logic lies in $\mathrm{NTIME}(n)$, i.e., in nondeterministic linear time. Grandjean and Olive [14,22] obtained interesting results related to those of Lynch. They gave logical representations of the class NLIN, i.e., linear time on random access machines, in terms of second-order logic with unary functions instead of relations (in their setting, also the input string is represented by a function).

Lautemann, Schwentick and Thérien [18] proved that the class CFL of context-free languages is characterized by ESO formulas of the form $\exists B\varphi$ where $\varphi$ is first-order, $B$ is a binary predicate symbol, and the range of the second-order quantifier is restricted to the class of *matchings*, i.e., pairing relations without crossover. Note that this is not a purely prefix-syntactic characterization of CFL. From our results and the fact that some languages which are not context-free can be expressed in the minimal nonregular ESO-prefix classes, it follows that a syntactic characterization of CFL by means of ESO-prefix classes is impossible.

Several restricted versions of REG where studied and logically characterized by restricted versions of ESO. McNaughton and Papert [21] showed that first-order logic with a linear ordering precisely characterizes the *star-free* regular languages. This theorem was extended by Thomas [29] to $\omega$-languages, i.e., languages with infinite words. Later several hierarchies of the star-free languages were studied and logically characterized (see, e.g. [29,23,24,25]). Straubing, Thérien and Thomas [28] showed that first-order logic with modular counting quantifiers characterize the regular languages whose syntactic monoids contain only solvable groups. These and many other related results can be found in the books and surveys [27,29,23,24,25].

## 2 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 [9]:

• 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 $\Phi$ 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?

Note that by Gurevich's classifiability theorem (cf. [3]) 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 1.1). It was the aim of [9] 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 $\Phi$ 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 $\Phi$ is true on some finite string ?

Reference [9] 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 [9] are the following findings.

**(1)** The class $\mathrm{ESO}(\exists^*\forall\exists^*)$ is regular. This theorem is the technically most involved result of [9]. 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 [9], 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.

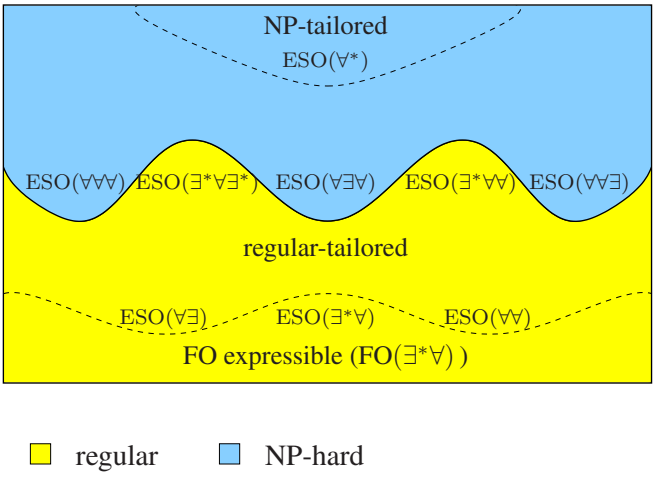


**Fig. 1.** Complete picture of the ESO-prefix classes on finite strings

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 [9], over finite strings satisfiability of a given $\mathrm{ESO}(\mathcal{Q})$ sentence is decidable iff $\mathrm{ESO}(\mathcal{Q})$ is regular.
**(6)** In [9], 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 [9] 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 [19,8] and well-known hierarchy theorems.

The main results of [9] 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($\exists^*\forall$)) [9]. The minimal classes which capture REG are ESO($\forall\exists$) and ESO($\forall\forall$).

*Potential Applications.* Monadic second-order logic over strings is currently used in the verification of hardware, software, and distributed systems. An example of a specific tool for checking specifications based on MSO is the MONA tool developed at the BRICS research lab in Denmark [1,15].

Observe that certain interesting desired properties of systems are most naturally formulated in *nonmonadic* second-order logic. Consider, as an unpretentious example[2], the following property of a ring $P$ of processors of different types, where two types may either be compatible or incompatible with each other. We call $P$ *tolerant*, if for each processor $p$ in $P$ there exist two other distinct processors $backup_1(p) \in P$ and $backup_2(p) \in P$, both compatible to $p$, such that the following conditions are satisfied:

1. for each $p \in P$ and for each $i \in \{1, 2\}$, $backup_i(p)$ is not a neighbor of $p$;
2. for each $i, j \in \{1, 2\}$, $backup_i(backup_j(p)) \notin \{p, backup_1(p), backup_2(p)\}$.

Intuitively, we may imagine that in case $p$ breaks down, the workload of $p$ can be reassigned to $backup_1(p)$ or to $backup_2(p)$. Condition 1 reflects the intuition that if some processor is damaged, there is some likelihood that also its neighbors are (e.g. in case of physical affection such as radiation), thus neighbors should not be used as backup processors. Condition 2 states that the backup processor assignment is antisymmetric and anti-triangular; this ensures, in particular, that the system remains functional, even if two processors of the same type are broken (further processors of incompatible type might be broken, provided that broken processors can be simply bypassed for communication).

Let $T$ be a fixed set of processor types. We represent a ring of $n$ processors numbered from 1 to $n$ where processor $i$ is adjacent to processor $i+1 \ (mod\, n)$ as a string of length $n$ from $T^*$ whose $i$-th position is $\tau$ if the type of the $i$-th processor is $\tau$; logically, $C_\tau(i)$ is then true. The property of $P$ being tolerant is expressed by the following second order sentence $\Phi$:

$$\Phi : \exists R_1, R_2, \forall x \exists y_1, y_2 .\, compat(x, y_1) \wedge compat(x, y_2) \wedge$$

$$\begin{aligned}
&R_1(x, y_1) \wedge R_2(x, y_2) \wedge \\
&\bigwedge_{i=1,2} \bigwedge_{j=1,2} \Big( \neg R_i(y_j, x) \wedge \neg R_1(y_j, y_i) \wedge \neg R_2(y_j, y_i) \Big) \wedge \\
&x \neq y_1 \wedge x \neq y_2 \wedge y_1 \neq y_2 \ \wedge \\
&\neg Succ(x, y_1) \wedge \neg Succ(y_1, x) \wedge \neg Succ(x, y_2) \wedge \neg Succ(y_2, x) \ \wedge \\
&\Big( (x = max) \rightarrow (y_1 \neq min \wedge y_2 \neq min) \Big) \ \wedge
\end{aligned}$$

[2] Our goal here is merely to give the reader some intuition about a possible type of application.

$$\Big((x = min) \rightarrow (y_1 \neq max \wedge y_2 \neq max)\Big),$$

where $compat(x, y)$ is the abbreviation for the formal statement that processor $x$ is compatible to processor $y$ (which can be encoded as a simple boolean formula over $C_\tau$ atoms).

$\Phi$ is the natural second-order formulation of the tolerance property of a ring of processors. This formula is in the fragment $\mathrm{ESO}(\exists^*\forall\exists^*)$; hence, by our results, we can immediately classify tolerance as a regular property, i.e., a property that can be checked by a finite automaton.

In a similar way, one can exhibit examples of $\mathrm{ESO}(\exists^*\forall\forall)$ formulas that naturally express interesting properties whose regularity is not completely obvious *a priori*. We thus hope that our results may find applications in the field of computer aided verification.

## 3 Recent Results on SO over Strings

In [10], we further investigated the extension of the results in the previous section from ESO to full second-order logic over strings. Our focus of attention were the $SO$ prefix classes which are regular vs. those which are not. In particular, we were interested to know how adding second-order variables affects regular fragments.

Generalizing Fagin's theorem, Stockmeyer [26] 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.

We were able to precisely characterize the regular and nonregular standard $\Sigma^1_k$ prefix classes. The maximal standard $\Sigma^1_k$ prefix classes which are regular and the minimal standard $\Sigma^1_k$ prefix classes which are non-regular are summarized in Figure 2.

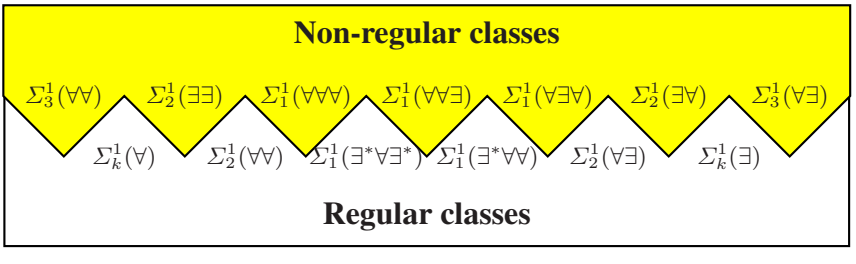


**Fig. 2.** Maximal regular and minimal non-regular SO prefix classes on strings.

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 [13] 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, $\Sigma_k^1(\mathrm{FO}^2)$, for $k \geq 2$, is intractable.

Figure 3 shows inclusion relationship between the classes $\Sigma_2^1(Q)$ where $Q$ contains 2 quantifiers. Similar relationships hold for $\Sigma_k^1(Q)$ classes. Furthermore, as shown in [10], we have that $\Sigma_2^1(\forall\exists) = \Sigma_1^1(\bigwedge \forall\exists)$ and $\Sigma_3^1(\exists\forall) = \Sigma_2^1(\bigvee \exists\forall)$, where $\Sigma_k^1(\bigvee \mathcal{Q})$ (resp., $\Sigma_k^1(\bigwedge \mathcal{Q})$) denote the class of $\Sigma_k^1$ sentences where the first-order part is a finite disjunction (resp., conjunction) of prefix formulas with quantifier in $\mathcal{Q}$.

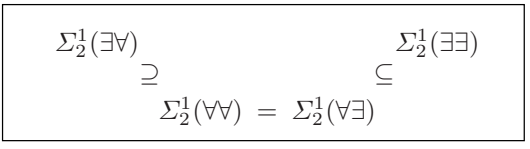


**Fig. 3.** Semantic inclusion relations between $\Sigma_2^1(Q)$ classes over strings, $|Q| = 2$

As for the complexity of model checking, the results from [9] and above imply that deciding whether $W \models \Phi$ for a fixed formula $\Phi$ and a given string $W$ is intractable for all prefix classes $\Sigma_k^1(Q)$ which are (syntactically) not included in the maximal regular prefix classes shown in Figure 2, with the exception $\Sigma_2^1(\exists\forall)$, 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, $\Sigma_2^1(\exists\exists)$ can express $\Sigma_2^p$-complete languages, where $\Sigma_2^p = \mathrm{NP}^{\mathrm{NP}}$ is from the second level of the polynomial hierarchy. Other fragments of $\Sigma_2^1$ have lower complexity; for example, $\Sigma_2^1(\forall^*\exists)$ is merely NP-complete. By adding further second-order variables, we can encode evaluating $\Sigma_k^p$-complete QBFs into $\Sigma_k^1(Q)$, where $Q = \exists\exists$ if $k > 2$ is even and $Q = \forall\forall$ if $k > 1$ is odd.

## 4 Recent Results on ESO over Graphs

In this section, we briefly describe the main results of [12], 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 [12] 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 [12], 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.[3] 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.
- If $\eta$ is a quantification pattern, then $\eta^*$ denotes all patterns obtained by repeating $\eta$ 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 $\varphi$ 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 [12] 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 $\geq 2$. This characterization is obtained by showing (assuming $\mathrm{P} \neq \mathrm{NP}$) 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.

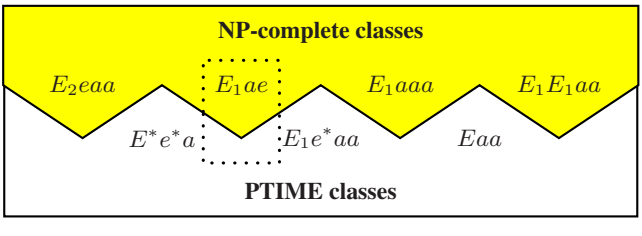


**Fig. 4.** ESO on arbitrary structures, directed graphs and undirected graphs with self-loops.

[3] For ESO over strings [9], the same level of precision was reached with simpler notation.

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 [3]). 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 [12] 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 [30]. $E^*ae$ 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 [12]. 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).

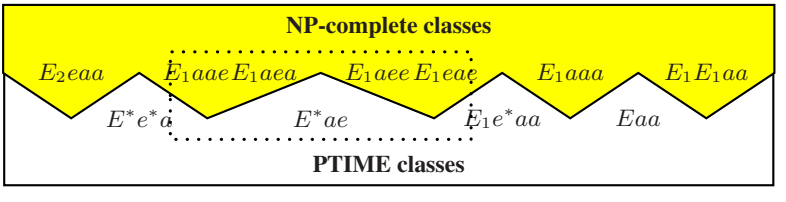


**Fig. 5.** ESO on undirected graphs without self-loops. The dotted boxes in Figures 4 and 5 indicate the difference between the two cases.

Technically, the most difficult result of [12] 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 [30] and using a new graph coloring method, it is shown that if a $E_1^* ae$-sentence $\Phi$ 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 $\Phi$. Second, Courcelle's theorem [6] is used by which model-checking for MSO sentences is polynomial on graphs of bounded tree-width. Third, Bodlaender's result [2] 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 [12] can be summarized in the following theorem.

**Theorem 4.1.** *Figures 4 and 5 provide a complete classification of the complexity of all ESO prefix classes on graphs.*

## 5 Open Problems

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 eiter 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 [9] and of Section 2 in this paper are *finite* strings. However, infinite strings or $\omega$-words are another important area of research. In particular, Büchi has shown that an analogue of his theorem (Proposition 1.1) also holds for $\omega$-words [4]. For an overview of this and many other important results on $\omega$-words, we refer the reader to the excellent survey paper [31]. In this context, it would be interesting to see which of the results established so far survive for $\omega$-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-N04.

## References


1. D. Basin and N. Klarlund Hardware verification using monadic second-order logic. *Proc. 7th Intl. Conf. on Computer Aided Verification (CAV'95)*, LNCS 939, pp. 31–41, 1995.
2. H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.
3. E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.
4. 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.
5. J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.
6. B. Courcelle. The monadic second-order logic of graphs I: recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.
7. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.
8. 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.
9. T. Eiter, G. Gottlob, and Y. Gurevich. Existential second-order logic over strings. *Journal of the ACM*, 47(1):77–131, 2000.
10. T. Eiter, G. Gottlob, and T. Schwentick. Second Order Logic over Strings: Regular and Non-Regular Fragments. *Developments in Language Theory, 5th Intl. Conference (DLT'01)*, Vienna, Austria, July 16-21, 2001, Revised Papers, Springer LNCS 2295, pp.37-56, 2002.
11. R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. M. Karp, editor, *Complexity of Computation*, pp. 43–74. AMS, 1974.
12. G. Gottlob, P. Kolaitis, and T. Schwentick. Existential second-order logic over graphs: charting the tractability frontier. In *Journal of the ACM*, 51(2):312–362, 2004.
13. 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.
14. E. Grandjean. Universal quantifiers and time complexity of random access machines. *Mathematical Systems Theory*, 13:171–187, 1985.
15. J.G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, B. Paige, T. Rauhe, and A. Sandholm Mona: Monadic Second-order Logic in Practice. in Proceedings of *Tools and Algorithms for the Construction and Analysis of Systems, First Intl. Workshop, TACAS'95*, Springer LNCS 1019, pp. 89–110, 1996.
16. N. Immerman. *Descriptive Complexity*. Springer, 1997.
17. P. Kolaitis and C. Papadimitriou. Some computational aspects of circumscription. *Journal of the ACM*, 37(1):1–15, 1990.
18. C. Lautemann, T. Schwentick, and D. Thérien. Logics for context-free languages. In *Proc. 1994 Annual Conference of the EACSL*, pages 205–216, 1995.
19. D. Leivant. Descriptive characterizations of computational complexity. *Journal of Computer and System Sciences*, 39:51–83, 1989.
20. J. F. Lynch. The quantifier structure of sentences that characterize nondeterministic time complexity. *Computational Complexity*, 2:40–66, 1992.
21. R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press, Cambridge, Massachusetts, 1971.
22. F. Olive. A Conjunctive Logical Characterization of Nondeterministic Linear Time. In *Proc. Conference on Computer Science Logic (CSL '97)*, LNCS. Springer, 1998 (to appear).

23. J.-E. Pin. *Varieties of Formal Languages*. North Oxford, London and Plenum, New York, 1986.
24. J.-E. Pin. Logic On Words. *Bulletin of the EATCS*, 54:145–165, 1994.
25. J.-E. Pin. Semigroups and Automata on Words. *Annals of Mathematics and Artificial Intelligence*, 16:343–384, 1996.
26. L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Comp. Sc.*, 3:1–22, 1977.
27. H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, 1994.
28. H. Straubing, D. Thérien, and W. Thomas. Regular Languages Defined with Generalized Quantifiers. *Information and Computation*, 118:289–301, 1995.
29. W. Thomas. Languages, Automata, and Logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, volume III, pages 389–455. Springer, 1996.
30. C. Thomassen. On the presence of disjoint subgraphs of a specified type. *Journal of Graph Theory*, 12:1, 101-111, 1988.
31. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 4. Elsevier Science Pub., 1990.
32. B. Trakhtenbrot. Finite automata and the logic of monadic predicates. *Dokl. Akad. Nauk SSSR*, 140:326–329, 1961.