Existential Second-order Logic over Graphs: A Complete Complexity-theoretic Classification

Till Tantau

Institute of Theoretical Computer Science
Universität zu Lübeck, Germany
tantau@tcs.uni-luebeck.de

Abstract

Descriptive complexity theory aims at inferring a problem's computational complexity from the syntactic complexity of its description. A cornerstone of this theory is Fagin's Theorem, by which a property is expressible in *existential second-order logic* (ESO logic) if, and only if, it is in NP. A natural question, from the theory's point of view, is which syntactic fragments of ESO logic also still characterize NP. Research on this question has culminated in a dichotomy result by Gottlob, Kolaitis, and Schwentick: for each possible quantifier prefix of an ESO formula, the resulting prefix class over graphs either contains an NP-complete problem or is contained in P. However, the exact complexity of the prefix classes inside P remained elusive. In the present paper, we clear up the picture by showing that for each prefix class of ESO logic, its reduction closure under first-order reductions is either FO, L, NL, or NP. For undirected self-loop-free graphs two containment results are especially challenging to prove: containment in L for the prefix $\exists R_1 \cdots \exists R_n \forall x \exists y$ and containment in FO for the prefix $\exists M \forall x \exists y$ for monadic M. The complex argument by Gottlob et al. concerning polynomial time needs to be carefully reexamined and either combined with the logspace version of Courcelle's Theorem or directly improved to first-order computations. A different challenge is posed by formulas with the prefix $\exists M \forall x \forall y$, which we show to express special constraint satisfaction problems that lie in L.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic

Keywords and phrases existential second-order logic, descriptive complexity, logarithmic space

Digital Object Identifier 10.4230/LIPIcs.STACS.2015.703

1 Introduction

Fagin's Theorem [9] establishes a tight connection between complexity theory and finite model theory: A language lies in NP if, and only if, it is the set of all finite models (coded appropriately as words) of some formula in *existential second-order logic* (ESO logic). This machine-independent characterization of a major complexity class sparked the research area of descriptive complexity theory, which strives to characterize the computational complexity of languages by the syntactic structure of the formulas that can be used to describe them. Nowadays, syntactic logical characterizations have been found for all major complexity classes, see [13] for an overview, although some syntactic extras (like numerical predicates) are often needed for technical reasons.

When looking at subclasses of NP like P, NL, L, or NC^1, one might hope that syntactic restrictions of ESO logic can be used to characterize them; and the most natural way of restricting ESO formulas is to limit the number and types of quantifiers used. All ESO formulas can be rewritten in prenex normal form as $\exists R_1 \cdots \exists R_r \forall x_1 \exists x_2 \cdots \forall x_{n-1} \exists x_n \, \psi$,

© Till Tantau;
licensed under Creative Commons License CC-BY
32nd Symposium on Theoretical Aspects of Computer Science (STACS 2015).
Editors: Ernst W. Mayr and Nicolas Ollinger; pp. 703–715
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

where the R_i are second-order variables, the x_i are first-order variables, and ψ is quantifier-free. Formulas like $\phi_{\text{3-colorable}} = \exists R \exists G \exists B \forall x \forall y \big((R(x) \lor G(x) \lor B(x)) \land \big(E(x,y) \to \neg(R(x) \land R(y)) \land \neg(G(x) \land G(y)) \land \neg(B(x) \land B(y)) \big) \big)$, which describes the NP-complete problem 3-COLORABLE, show that we do not need the full power of ESO logic to capture NP-complete problems: the prefix $\exists R \exists G \exists B \forall x \forall y$ suffices. However, do formulas of the form, say, $\exists R \forall x \exists y \, \psi$ also capture all of NP; or do they characterize exactly, say, P? This question lies at the heart of a detailed study by Gottlob, Kolaitis, and Schwentick [11] entitled *Existential Second-Order Logic Over Graphs: Charting the Tractability Frontier,* where the following dichotomy is shown: For each possible syntactic restriction of the quantifier block of ESO formulas, the resulting *prefix class* either contains an NP-complete problem or is contained in P. For instance, it is shown there that all graph problems expressible by formulas of the form $\exists R \forall x \forall y \, \psi$ lie in P, while some problems expressible by formulas of the form $\exists R \forall x \forall y \forall z \, \psi$ are NP-complete. The dichotomy does not, however, settle the question of whether all of P – or at least some interesting subclass thereof like logarithmic space (L) or nondeterministic logarithmic space (NL) – is described by one of the logical fragments.

1.1 Contributions of This Paper

One cannot really hope to show that the prefix class of, say, the quantifier prefix $\exists R \forall x \forall y$ is exactly P, since $\text{P} \neq \text{NP}$ would follow: This syntactically severely restricted prefix class can be shown [6, Proposition 10.6] to be contained in $\text{NTIME}(n^k)$ for some constant k and is thus provably different from NP by the time hierarchy theorem. The best one can try to prove are statements like "this prefix class is contained in P and contains a problem complete for P" or, phrased more succinctly, "the reduction closure of this prefix class is P." Our main result, Theorem 1.1, consists of such statements: *For each possible* ESO *prefix class, its reduction closure under first-order reductions is either* FO*,* L*,* NL*, or* NP. In particular, no prefix class yields P as its reduction closure (unless, of course, $\text{P} = \text{NP}$ or $\text{NL} = \text{P}$).

It makes a difference which vocabulary we are allowed to use in our formulas and which logical structures we are interested in: Results depend on whether we consider arbitrary graphs, undirected graphs, undirected graphs without self-loops, or just strings. (In this paper, all considered graphs are finite.) The case of strings has been addressed and settled in [6]. In the present paper we consider the same three cases as in [11]: In our vocabulary, we always have just a single binary relational symbol (E), so all models of formulas are graphs. We then differentiate between directed graphs, undirected graphs, and undirected graphs without self-loops (which we call *basic graphs* for brevity). Note that allowing self-loops, whose presence at a vertex x can be tested with the formula $E(x,x)$, is equivalent to considering basic graphs together with an additional monadic input predicate.

To describe the syntactic fragments of ESO logic easily and succinctly, we use the notation of [11]: The uppercase letter E denotes the presence of an existential second-order quantifier, an optional index as in E_2 denotes the arity of the quantifier, and the lowercase letters a and e denote universal and existential first-order quantifiers, respectively. The prefix type of the formula $\phi_{\text{3-colorable}}$ mentioned earlier is $EEEaa$ (or even $E_1E_1E_1aa$ since the predicates are monadic) and we say that $\phi_{\text{3-colorable}}$ *has prefix type* $EEEaa$ (and also $E_1E_1E_1aa$). We will use regular expressions over the alphabet $\{a, e, E, E_1, E_2, E_3, \dots\}$ to denote patterns of prefix types such as E^*aa for "any number of existential second-order quantifiers followed by exactly two universal first-order quantifiers." To define the three kinds of prefix classes that we are interested in, for a formula ϕ let $\text{MODELS}_{\text{directed}}(\phi) = \{G \mid G$ is a directed graph and $G \models \phi\}$, $\text{MODELS}_{\text{undirected}}(\phi) = \{G \mid G$ is an undirected graph and $G \models \phi\}$, and $\text{MODELS}_{\text{basic}}(\phi) = \{G \mid G$ is a basic graph and $G \models \phi\}$. For instance,

$\text{MODELS}_{\text{basic}}(\phi_{\text{3-colorable}}) = \text{3-COLORABLE}$ (ignoring coding issues). Next, for a prefix type pattern P, let $\text{FD}_{\text{directed}}(P) = \{\text{MODELS}_{\text{directed}}(\phi) \mid \phi$ has a prefix type in $P\}$ and define $\text{FD}_{\text{undirected}}(P)$ and $\text{FD}_{\text{basic}}(P)$ similarly for undirected and basic graphs. “FD” stands for “Fagin-definable” and Fagin’s Theorem can be stated succinctly as $\text{FD}_{\text{strings}}(E^*(ae)^*) = \text{NP}$.

As stated earlier, in the context of syntactic fragments of ESO logic it makes sense to consider reduction closures of prefix classes rather than the prefix classes themselves. It will not matter much which particular kind of reductions we use, as long as they are weak enough. All our reductions will be *first-order reductions* [13], which are first-order queries with access to the bit predicate or, equivalently, functions computable by a logarithmic-time-uniform constant-depth circuit family.[1] Let us write $A \leq_{\text{fo}} B$ if A can be reduced to B using first-order reductions. Let us write $\overline{\text{FD}}_{\text{directed}}(P) = \{A \mid A \leq_{\text{fo}} B \in \text{FD}_{\text{directed}}(P)\}$ for the reduction closure of $\text{FD}_{\text{directed}}(P)$ and define $\overline{\text{FD}}_{\text{undirected}}(P)$ and $\overline{\text{FD}}_{\text{basic}}(P)$ similarly.

▶ **Theorem 1.1** (Main Result). *The following table completely classifies all prefix classes of* ESO *logic over basic graphs (upper part) and undirected and directed graphs (lower part):*[2]

If P is at least one of ...	*and at most one of ...,*	*then*
–	$(ae)^*$, E^*e^*a, $\boxed{E_1ae}$	$\overline{\text{FD}}_{\text{basic}}(P) = \text{FO}$
$\boxed{E_1E_1ae}$, $\boxed{E_2ae}$	$\boxed{E^*ae}$	$\overline{\text{FD}}_{\text{basic}}(P) = \text{L}$
$\boxed{E_1aa}$	$\boxed{Eaa}$	$\overline{\text{FD}}_{\text{basic}}(P) = \text{L}$
E_1eaa	E_1e^*aa	$\overline{\text{FD}}_{\text{basic}}(P) = \text{NL}$
E_1aaa, E_1E_1aa, E_2eaa, E_1eae, E_1aee, E_1aea, E_1aae	$E^*(ae)^*$	$\overline{\text{FD}}_{\text{basic}}(P) = \text{NP}$
–	$(ae)^*$, E^*e^*a	$\overline{\text{FD}}_{\text{undirected}}(P) = \overline{\text{FD}}_{\text{directed}}(P) = \text{FO}$
$\boxed{E_1aa}$	E_1e^*aa, $\boxed{Eaa}$	$\overline{\text{FD}}_{\text{undirected}}(P) = \overline{\text{FD}}_{\text{directed}}(P) = \text{NL}$
E_1aaa, E_1E_1aa, E_2eaa, $\boxed{E_1ae}$	$E^*(ae)^*$	$\overline{\text{FD}}_{\text{undirected}}(P) = \overline{\text{FD}}_{\text{directed}}(P) = \text{NP}$

Note that we always have $\overline{\text{FD}}_{\text{undirected}}(P) = \overline{\text{FD}}_{\text{directed}}(P)$, which is not trivial, especially for the prefix E_1aa: On undirected graphs, using only two universally quantified variables, it seems difficult to express “non-symmetric” properties, suggesting $\text{FD}_{\text{undirected}}(E_1aa) \subsetneq \text{L}$. However, using a gadget construction, we will show that $\text{FD}_{\text{undirected}}(E_1aa)$ contains an NL-complete problem.

As an application of the theorem, let us use it to prove EVEN-CYCLE $\in$ L, which is the problem of detecting the presence of a cycle[3] of even length in basic graphs B. The complexity of this problem has been researched for a long time, see [12] for a discussion and variants. The idea is to consider the following ESO formulas:

$$\phi_m = \exists C_1 \cdots \exists C_m \forall x \exists y \Big(E(x,y) \wedge \textstyle\bigvee_{i=1}^m \big(C_i(x) \wedge C_{(i \bmod m)+1}(y) \wedge \bigwedge_{j \neq i} \neg C_j(x)\big)\Big). \quad (1)$$

They “describe” the following situation: The basic graph can be colored with m different colors so that each vertex x is connected to a “next” vertex y with the “next” color (with color C_1 following C_m). For $m > 2$, it is not hard to see that $B \models \phi_m$ if, and only if, every connected component of B contains a cycle whose length is a multiple of m. Since ϕ_m has quantifier prefix E^*ae and the graphs are basic, the second row concerning basic

---

1 As a technicality, since we use first-order reductions with access to the bit predicate, by FO we refer to “first-order logic with access to the bit predicate,” which is the same as logarithmic-time-uniform AC^0.
2 The “interesting” prefixes, where the complexity classes differ between the two parts, are highlighted.
3 A cycle in an undirected graph must, of course, have length at least 3 and consist of distinct vertices.

graphs in Theorem 1.1 tells us that $B \models \phi_m$ can be decided in logarithmic space. The following algorithm now shows EVEN-CYCLE $\in$ L: In a basic input graph B, replace all edges by length-2 paths, then test whether $C \models \phi_4$ holds for some connected component C of B.

1.2 Technical Contributions

The proofs of the statements $\mathrm{FD}_{\mathrm{basic}}(E^*ae) \subseteq \mathrm{L}$ and $\mathrm{FD}_{\mathrm{basic}}(E_1ae) \subseteq \mathrm{FO}$ require a sophisticated technical machinery. In both cases, our proofs follow the ideas of a 35-page proof of $\mathrm{FD}_{\mathrm{basic}}(E^*ae) \subseteq \mathrm{P}$ in [11]. The central observation concerning the first statement is that the *algorithmically* most challenging part in the proof of [11] is the application of Courcelle's Theorem [5] to graphs of bounded tree width. It has been shown in [8] that there is a logspace version of Courcelle's Theorem, which will allow us to lower the complexity from P to L when the input graphs have bounded tree width. For graphs of unbounded tree width, we will explain how the other polynomial time procedures from the proof of [11] can be reimplemented in logarithmic space.

To prove $\mathrm{FD}_{\mathrm{basic}}(E_1ae) \subseteq \mathrm{FO}$, we need to lower the complexity of the involved algorithms further. The idea is to again follow the ideas from [11] for E^*ae. When there is just a single monadic predicate, certain algorithmic aspects of the proof can be simplified so severely that they can actually be expressed in first-order logic. Note, however, that already a second monadic predicate or a single binary predicate makes the complexity jump up to L, that is, $\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1E_1ae) = \overline{\mathrm{FD}}_{\mathrm{basic}}(E_2ae) = \mathrm{L}$.

Concerning the remaining claims from Theorem 1.1 that are not already proved in [11], two cases are noteworthy: Proving that $\mathrm{FD}_{\mathrm{basic}}(E_1eaa)$ contains an NL-complete problem turns out to require a nontrivial gadget construction. Proving $\mathrm{FD}_{\mathrm{basic}}(E_1aa) \subseteq \mathrm{L}$ requires a reformulation of the problems in $\mathrm{FD}_{\mathrm{basic}}(E_1aa)$ as special constraint satisfaction problems and showing that these lie in L.

1.3 Related Work

The study of the expressive power of syntactic fragments of logics dates back decades; the decidability of prefix classes of first-order logic, for instance, has been solved completely in a long sequence of papers, see [2] for an overview. Interestingly, the first-order Ackermann prefix class ae plays a key role in that context and both E_1ae and E^*ae turn out to be the most complicated cases in the context of the present paper, too. The expressive power of monadic second-order logic (MSO logic) has also received a lot of attention, for instance in [3, 5, 7], but emphasis has been on restricted structures rather than on syntactic fragments.

Concerning syntactic fragments of ESO logic, the two papers most closely related to the present paper are [6] by Eiter, Gottlob, and Gurevich and [11] by Gottlob, Kolaitis, and Schwentick. In the first paper, a similar kind of classification is presented as in the present paper, only over *strings* rather than *graphs*. It is shown there that for all prefix patterns P the class $\mathrm{FD}_{\mathrm{strings}}(P)$ is either equal to NP; is not equal to NP but contains an NP-complete problem; is equal to REG; or is a subclass of FO. Interestingly, two classes of special interest are $\mathrm{FD}_{\mathrm{strings}}(E_1^*ae)$ and $\mathrm{FD}_{\mathrm{strings}}(E_1^*aa)$, both of which are the minimal classes equal to the regular languages (by the results of Büchi [3]). In comparison, by the results of the present paper $\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1^*ae) = \overline{\mathrm{FD}}_{\mathrm{basic}}(E_1E_1ae) = \mathrm{L}$, while $\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1ae) = \mathrm{FO}$, and $\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1^*aa) = \overline{\mathrm{FD}}_{\mathrm{basic}}(E_1E_1aa) = \mathrm{NP}$, while $\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1aa) = \mathrm{L}$.

The present paper builds on the paper [11] by Gottlob, Kolaitis, and Schwentick, which contains many of the upper and lower bounds from Theorem 1.1 for the class NP as well as most of the combinatorial and graph-theoretic arguments needed to prove $\mathrm{FD}_{\mathrm{basic}}(E^*ae) \subseteq \mathrm{L}$

and $\mathrm{FD}_{\mathrm{basic}}(E_1ae) \subseteq \mathrm{FO}$. The paper misses, however, the finer classification provided in our Theorem 1.1 and Remark 5.1 of [11] expresses the unclear status of the exact complexity of $\mathrm{FD}_{\mathrm{basic}}(E^*ae)$ at the time of writing, which hinges on a problem called SATU(P): "Note also that for each P, SATU(P) is probably not a PTIME-complete set. [...] This is due to the check for bounded treewidth, which is in LOGCFL (cf. Wanke [1994]) but not known to be in NL." The complexity of the check for bounded tree width was settled only later, namely in a paper by Elberfeld, Jakoby, and the author [8], and shown to lie in L. This does not mean, however, that the proof of [11] immediately yields $\mathrm{FD}_{\mathrm{basic}}(E^*ae) \subseteq \mathrm{L}$ since the application of Courcelle's Theorem is but one of several subprocedures in the proof and since a generalization of tree width rather than normal tree width is used.

1.4 Organization of This Paper

To prove Theorem 1.1, we need to prove the lower bounds implicit in the first column of the theorem's table and the upper bounds implicit in the second column. The lower bounds are proved in Section 2 by presenting reductions from complete problems for L, NL, or NP. The upper bounds are proved in Section 3, where we prove, in order, $\mathrm{FD}_{\mathrm{basic}}(Eaa) \subseteq \mathrm{L}$, $\mathrm{FD}_{\mathrm{basic}}(E^*ae) \subseteq \mathrm{L}$, and $\mathrm{FD}_{\mathrm{basic}}(E_1ae) \subseteq \mathrm{FO}$ using arguments drawn from different areas.

Only the proof *ideas* are given in this conference paper, please see the technical report version for full proofs [16].

2 Lower Bounds: Hardness for L and NL

For each of the prefix patterns listed in the first column of the table in Theorem 1.1 we now show that their prefix classes contain problems that are hard for L, NL, or NP. The problems from which we reduce are listed in Table 1. As can be seen, we only need to prove new results for a minority of the classes since the NP cases have already been settled in [11].

Table 1 The lower bounds in Theorem 1.1 are proved by showing that the problems in this table, which are complete for the classes in the claims, are either expressible in the fragment or are at least reducible to a problem expressible in the fragment. The problem UNREACH asks whether there is *no* path from s to t in a directed graph. The problems A_2 and A_3 are explained below.

Claim		*Hard problem*	*Proved where*
Lower bounds for basic graphs			
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1E_1ae)$	$\supseteq \mathrm{L}$	A_3	Lemma 2.1
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_2ae)$	$\supseteq \mathrm{L}$	A_2	Lemma 2.2
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1aa)$	$\supseteq \mathrm{L}$	2-COLORABLE	[11, Remark 3.1]
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1eaa)$	$\supseteq \mathrm{NL}$	UNREACH	Lemma 2.3
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1aaa)$	$\supseteq \mathrm{NP}$	POSITIVE-ONE-IN-THREE-3SAT	[11, Theorem 2.2]
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1E_1aa)$	$\supseteq \mathrm{NP}$	3-COLORABLE	[11, Theorem 2.3]
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_2eaa)$	$\supseteq \mathrm{NP}$	3-COLORABLE	[11, Theorem 2.4]
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1eae)$	$\supseteq \mathrm{NP}$	3SAT	[11, Theorem 2.5]
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1aee)$	$\supseteq \mathrm{NP}$	NOT-ALL-EQUAL-3SAT	[11, Theorem 2.6]
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1aea)$	$\supseteq \mathrm{NP}$	POSITIVE-ONE-IN-THREE-3SAT	[11, Theorem 2.7]
$\overline{\mathrm{FD}}_{\mathrm{basic}}(E_1aae)$	$\supseteq \mathrm{NP}$	POSITIVE-ONE-IN-THREE-3SAT	[11, Theorem 2.8]
Remaining lower bounds for undirected and, thereby, also for directed graphs			
$\overline{\mathrm{FD}}_{\mathrm{undirected}}(E_1aa)$	$\supseteq \mathrm{NL}$	UNREACH	Lemma 2.3
$\overline{\mathrm{FD}}_{\mathrm{undirected}}(E_1ae)$	$\supseteq \mathrm{NP}$	3SAT	[11, Theorem 2.1]

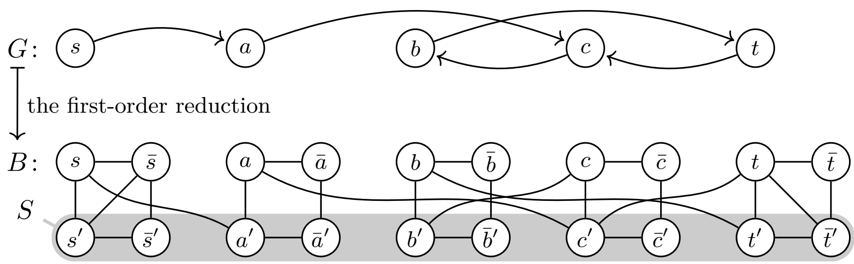

Figure 1 Example of the reduction from Lemma 2.3. The directed graph G on top is reduced to the basic graph at the bottom. The edges from the "squares" result from the first rule given in the full proof in the full paper, the curved edges result from the second rule, and the two diagonal edges result from the last rule.

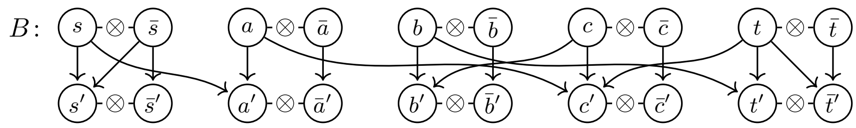

Figure 2 Visualization of the requirements concerning which vertices may lie in M imposed by the formula ψ: For edges with label $\otimes$ exactly one end must lie in M and for directed edges, if the tail of the edge lies in M, the head must also lie in M.

The two special languages A_2 and A_3 in the table are defined as follows: For $m \geq 2$ let $A_m = \{G \mid G$ is an undirected graph in which each connected component contains a cycle whose length is a multiple of $m\}$. These languages are all hard for L: In [4, page 388, remarks for problem UFA] it is shown that the reachability problem for graphs consisting of just two undirected trees is complete for L. Since L is trivially closed under complement, testing whether there is *no* path from a vertex u to a vertex v in a graph consisting of two trees is also complete for L, which in turn is the same as asking whether u and v lie in different trees. To reduce this question to A_m, attach cycles of length $2m$ to both u and v. Then all (namely both) components of the resulting graph contain a cycle whose length is a multiple of m if, and only if, u and v lie in different components. (Using a cycle length of $2m$ rather than m ensures that also for $m = 2$ we attach a proper cycle.)

▶ **Lemma 2.1.** $A_3 \in \mathrm{FD}_{\mathrm{basic}}(E_1E_1ae)$.

Proof idea. Use ϕ_3 from equation (1), but get rid of one of the second-order quantifiers. ◀

▶ **Lemma 2.2.** $A_2 \in \mathrm{FD}_{\mathrm{basic}}(E_2ae)$.

Proof idea. Use $\exists F \forall x \exists y \big(E(x,y) \land F(x,y) \land \neg F(y,x) \land (F(x,x) \leftrightarrow \neg F(y,y))\big)$. ◀

▶ **Lemma 2.3.** UNREACH *reduces to a problem in* $\mathrm{FD}_{\mathrm{basic}}(E_1eaa)$ *and also to a problem in* $\mathrm{FD}_{\mathrm{undirected}}(E_1aa)$.

Proof idea. Undirected graphs are essentially the same as basic graphs with an extra monadic relation S^1 that is part of the input. Similarly, a single existential first-order quantifier such as the one in E_1eaa allows us to pick a vertex and then single out the set of vertices connected to it. Thus, essentially, it suffices to show that UNREACH reduces to $\mathrm{MODELS}_{\mathrm{basic}}(\exists M \forall x \forall y\, \psi)$ where ψ is a formula over the vocabulary (E^2, S^1).

The reduction works as shown in Figure 1: Each vertex of the original directed graph gets replaced by four vertices that are connected in a square. Two of them are in the set S,

Table 2 The upper bounds from Theorem 1.1 and where they are proved. Missing upper bounds for basic and undirected graphs follow from the bounds for directed graphs on the right.

Claims for basic graphs		*Proved where*	*Claims for directed graphs*		*Proved where*
$\mathrm{FD}_{\mathrm{basic}}(E_1ae)$	$\subseteq$ FO	Section 3.3	$\mathrm{FD}_{\mathrm{directed}}((ae)^*)$	$\subseteq$ FO	trivial
$\mathrm{FD}_{\mathrm{basic}}(E^*ae)$	$\subseteq$ L	Section 3.2	$\mathrm{FD}_{\mathrm{directed}}(E^*e^*a)$	$\subseteq$ FO	[11, Theorem 3.1]
$\mathrm{FD}_{\mathrm{basic}}(Eaa)$	$\subseteq$ L	Section 3.1	$\mathrm{FD}_{\mathrm{directed}}(E_1e^*aa)$	$\subseteq$ NL	[11, Theorem 3.2]
			$\mathrm{FD}_{\mathrm{directed}}(Eaa)$	$\subseteq$ NL	[11, Theorem 3.4]
			$\mathrm{FD}_{\mathrm{directed}}(E^*(ae)^*)$	$\subseteq$ NP	Fagin's Theorem

the others are not. Directed edges in the original graph get replaced by undirected edges between one of the four vertices of the tail vertex and one of the four vertices of the head vertex. Additionally, there are edges inside the square of the source and of the target.

The formula ψ expresses that edges inside S and edges outside S correspond to an exclusive or with respect to membership in M, edges between vertices in S and outside S correspond to an implication: If the vertex outside S is in M, so must the vertex inside S. Figure 2 visualizes this situation. One then shows the following: There is some M that makes ϕ true if, and only if, there can be *no* path from s to t in G since we must have $s \in M$, $t \notin M$, and together with s the set M must contain all vertices reachable from s. ◀

3 Upper Bounds: Containment in FO and L

The second column of the table in Theorem 1.1 lists upper bounds that we address in the present section. Table 2 shows the order in which we tackle them.

3.1 *Eaa* Over Basic Graphs: Reformulation as Constraint Satisfaction

Our first upper bound, $\mathrm{FD}_{\mathrm{basic}}(Eaa) \subseteq \mathrm{L}$, is proved in two steps: First, we reformulate the problems in $\mathrm{FD}_{\mathrm{basic}}(Eaa)$ as special constraint satisfaction problems (CSPs) in Lemma 3.1. Second, we show that these CSPs lie in L in Lemma 3.2.

It will not be necessary to formally introduce the whole theory of constraint satisfaction problems since we will only encounter one very specialized form. Furthermore, our CSPs do not quite fit into the standard framework and major results on CSPs like Schaefer's Theorem [15] or the refined version thereof [1] do not settle the complexity of these special CSPs. Nevertheless, we will need some basic terminology: In a binary CSP, we are given a universe U and a set of *constraints*, each of which picks a number of elements from U and specifies one or more possibilities concerning which of these elements may lie in a *solution* $X \subseteq U$. A *constraint language* specifies the types of constraints that we are allowed to use. For instance the constraint language for 3SAT specifies that constraints (which are clauses) must rule out one of the eight possibilities concerning which of the elements (which are the variables) are in X (are set to *true*). We need to deviate from this framework in one important way: we require that there is a constraint for *every* pair of distinct elements of U, not just for some of them. Unfortunately, this deviation inhibits our applying the classification of the complexity of CSPs from [1]; more precisely, the smallest standard CSP classes that are able to express the special CSPs we are interested in are known to contain NL-complete languages – while we wish to prove containment in L.

For sets $C, D \subseteq \{0,1,2\}$ we define a *$\{C,D\}$-constraint satisfaction problem P on a universe U* to be a mapping that maps each size-2 subset $\{x,y\} \subseteq U$ to either C or D. A *solution for P* is a subset $X \subseteq U$ such that for all size-2 subsets $\{x,y\} \subseteq U$ we have

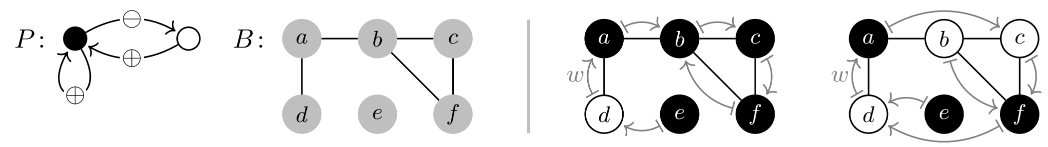

Figure 3 Example of a pattern graph $P = (C, A^{\oplus}, A^{\ominus})$ with two "colors" black and white (so $C = \{black, white\}$, $A^{\oplus} = \{(black, black), (white, black)\}$, and $A^{\ominus} = \{(black, white)\}$) and an uncolored ("gray") example graph B. We have $B \in$ SATURATION(P) as shown by two examples of legal colorings of B together with witness functions w (in gray).

$|\{x, y\} \cap X| \in P(\{x, y\})$. In other words, P fixes for every pair of two vertices x or y one of two possible constraints concerning *how many* elements of $\{x, y\}$ may lie in X. Let CSP$\{C, D\} = \{P \mid P$ is a $\{C, D\}$-CSP that has a solution$\}$. As an example, CSP$\big\{\{1\}, \{0, 1, 2\}\big\}$ is essentially the same as the problem 2-COLORABLE = BIPARTITE since a $\{1\}$-constraint enforces that exactly one of two vertices must lie in X (and, hence, corresponds to an edge), while a $\{0, 1, 2\}$-constraint has no effect (and, hence, corresponds to no edge being present). In Lemma 3.2 we show that all CSP$\{C, D\}$ lie in L, which is fortunate since we reduce to them:

▶ **Lemma 3.1.** *For every Eaa-formula ϕ there are sets $C, D \subseteq \{0, 1, 2\}$ such that the set* MODELS$_{\text{basic}}(\phi)$ *reduces to* CSP$\{C, D\}$.

Proof idea. By [11, Lemma 3.3] we may assume that ϕ has the form $\exists M \forall x \forall y\, \psi$ with a *monadic* quantifier M. Rewrite ψ as $x \neq y \to \big((E(x, y) \to \gamma) \wedge (\neg E(x, y) \to \delta)\big)$ where γ and δ only contain $M(x)$ and $M(y)$ as atomic formulas. Since the graphs are basic and x and y are interchangeable, γ and δ can only make claims concerning $|\{x, y\} \cap M|$. Use C to encode the claim made by γ and D to encode δ. ◀

▶ **Lemma 3.2.** *Let $C, D \subseteq \{0, 1, 2\}$. Then* CSP$\{C, D\} \in$ L.

Proof idea. Argue for each choice of C and D how we can check in logarithmic space whether a $\{C, D\}$-CSP P has a solution $X \subseteq U$. Most cases are quite trivial; the only interesting ones are CSP$\big\{\{1\}, \{0, 1, 2\}\big\}$, which we already saw to be essentially the same as 2-COLORABLE = BIPARTITE $\in$ L, and CSP$\big\{\{0, 1\}, \{1, 2\}\big\}$, which is essentially the same as SPLIT-GRAPH and hence lies even in FO by a characterization of [10]. ◀

3.2 E^*ae Over Basic Graphs: From P to L

Our objective is to show FD$_{\text{basic}}(E^*ae) \subseteq$ L in this section. More precisely, we only need to show FD$_{\text{basic}}(E_1^*ae) \subseteq$ L since [11, Theorem 4.1] states FD$_{\text{basic}}(E^*ae) =$ FD$_{\text{basic}}(E_1^*ae)$.

A proof of the weaker claim FD$_{\text{basic}}(E_1^*ae) \subseteq$ P is spread over the 35 pages of Sections 4, 5, and 6 of the paper [11] and consists of two kinds of arguments: graph-theoretic ones and algorithmic ones. Since the graph-theoretic arguments are independent of complexity-theoretic considerations, our main job is to show how the algorithms described by Gottlob et al. can be implemented in logarithmic space rather than polynomial time.

Similarly to the switch from model checking problems to graphs problems in the previous section, we also wish to reformulate the model checking problems MODELS$_{\text{basic}}(\phi)$ for E_1^*ae-formulas ϕ in a graph-theoretic manner. Gottlob et al. introduce the notion of *pattern graphs* for this: A *pattern graph* $P = (C, A^{\oplus}, A^{\ominus})$ consists of a set of *colors* C, a set $A^{\oplus} \subseteq C \times C$ of $\oplus$-*arcs*, and a set $A^{\ominus} \subseteq C \times C$ of $\ominus$-*arcs* ($A^{\oplus}$ and $A^{\ominus}$ need not be disjoint). Given a basic graph $B = (V, E)$, a *coloring of G with respect to P* is a function $c\colon V \to C$. A mapping $w\colon V \to V$ is called a *witness function for a coloring c* if for all $x \in V$ we have

(1) $x \neq w(x)$, (2) if $\{x, w(x)\} \in E$, then $\bigl(c(x), c(w(x))\bigr) \in A^{\oplus}$, and (3) if $\{x, w(x)\} \notin E$, then $\bigl(c(x), c(w(x))\bigr) \in A^{\ominus}$.[4] If there exists a coloring together with a witness function for B with respect to P, we say that B *can be saturated by* P and the *saturation problem* $\textsc{saturation}(P)$ is the set of all basic graphs that can be saturated by P, see Figure 3 for an example.

The intuition behind these definitions is that a witness function tells us for each x in $\forall x$ which y in $\exists y$ we must pick to make a formula ϕ of the form $\exists M_1 \cdots \exists M_n \, \forall x \exists y \, \psi$ true. The pattern graph encodes the restrictions imposed by ψ and the monadic predicates M_i:

▶ **Fact 3.3** ([11, Theorem 4.6]). For every formula $\phi = \exists M_1 \cdots \exists M_n \, \forall x \exists y \, \psi$, where the M_i are monadic and ψ is quantifier-free, there is a pattern graph P with 2^n vertices such that $\textsc{models}_{\text{basic}}(\phi) = \textsc{saturation}(P)$.

Thus, it remains to show $\textsc{saturation}(P) \in \mathrm{L}$ for all pattern graphs P. Towards this aim, for a fixed pattern graph P we devise logspace algorithms that work for larger and larger classes of basic graphs B, ending with the class of all basic graphs.

Graphs of Bounded Tree Width and Special Graphs We start by considering only graphs of *bounded tree width,* an important class of graphs introduced by Robertson and Seymour in [14]: A *tree decomposition* of a graph B is a tree T together with a mapping that assigns subsets of B's vertices (called *bags*) to the nodes of T. The bags must have two properties: First, for every edge $\{x, y\}$ of B there must be some bag that contains both x and y. Second, the nodes of T whose bags contain a given vertex x must be connected in T. The *width* of a decomposition is the size of its largest bag (minus 1 for technical reasons). The *tree width* of B is the minimal width of any tree decomposition for it. A class of graphs has *bounded tree width* if there is a constant c such that all graphs in the class have tree width at most c. From an algorithmic point of view, many problems that can be solved efficiently on trees can also be solved efficiently on graphs of bounded tree width. Courcelle's Theorem turns this into a precise statement:

▶ **Fact 3.4** (Courcelle's Theorem, [5]). For every $\textsc{mso}$-formula ϕ and $t \geq 1$ we have

$$\textsc{models}_{\text{basic}}(\phi) \cap \{G \mid G \text{ has tree width at most } t\} \in \mathrm{LINTIME}.$$

Gottlob et al. apply this theorem to show that when the input graphs B have bounded tree width, we can decide whether $B \in \textsc{saturation}(P)$ holds in polynomial time: the property $B \in \textsc{saturation}(P)$ is easily described in $\textsc{mso}$ logic. We can lower the complexity from "polynomial time" to "logarithmic space" by using the following logarithmic space version of Courcelle's Theorem:

▶ **Fact 3.5** (Logspace Version of Fact 3.4, [8]). For every $\textsc{mso}$-formula ϕ and $t \geq 1$ we have

$$\textsc{models}_{\text{basic}}(\phi) \cap \{G \mid G \text{ has tree width at most } t\} \in \mathrm{L}.$$

In their graph-theoretic arguments, Gottlob et al. encounter not only graphs of bounded tree width, but also graphs that they call (k,t)-*special* and which are defined as follows: For a basic graph $B = (V, E)$ let us call two vertices u and v *equivalent* if for all $x \in V \setminus \{u, v\}$ we have $\{u, x\} \in E$ if, and only if, $\{v, x\} \in E$. Observe that this defines an easy-to-check

[4] Using $\{u, v\}$ to indicate an undirected edge between u and v in a basic graph and, in not-so-slight abuse of notation, even writing $\{u, v\} \in E$, helps in distinguishing these edges from the directed edges in the pattern graph. Formally, we mean of course $(u, v) \in E$ and $(v, u) \in E$; and $E \subseteq V \times V$ holds.

equivalence relation on the vertices of B and that each equivalence class is either a clique or an independent set of B. A graph is (k,t)-*special* if we can remove (up to) k equivalence classes $A_1, \dots, A_k$ from the graph such that the remaining graph has tree width at most t.

The intuition behind (k,t)-special graphs is that equivalent vertices are "more or less indistinguishable" and, thus, for a large enough equivalence class removing some vertices does not change whether the graph can be saturated or not. Formally, let B be (k,t)-special and let $A_1, \dots, A_k$ be to-be-removed equivalence classes. We obtain an *s-shrink of B* by repeatedly removing vertices from those A_i that have more than s vertices until all of them have at most s vertices. The proof of Lemma 6.4 in [11] implies the following two facts:

▶ Fact 3.6. For every k, t, and pattern graph P there is an s such for every s-shrink B' of a (k,t)-special graph B we have $B \in \textsc{saturation}(P)$ if, and only if, $B' \in \textsc{saturation}(P)$.

▶ Fact 3.7. An s-shrink of a (k,t)-special graph has tree width at most $t + sk$.

In Lemmas 6.3 and 6.4 of [11], Gottlob et al. present polynomial-time algorithms for testing whether a graph is (k,t)-special and for computing an s-shrink when the test is positive. The following lemma shows that we can reimplement these algorithms in a space-efficient manner (which the original algorithms are not):

▶ **Lemma 3.8.** *For every s, k, and t, there is a logspace computable function that maps every (k,t)-special graph B to an s-shrink of B (and all other graphs to "not (k,t)-special").*

Proof idea. Find a tuple $(v_1, \dots, v_k)$ of vertices such that removing all vertices equivalent to some v_i leaves behind a graph of tree width at most t. Then for each v_i leave only the lexicographically first s vertices equivalent to v_i in the graph. ◀

The following lemma sums up the bottom line of the above discussion:

▶ **Lemma 3.9.** *For every pattern graph P and all k and t we have*

$$\textsc{saturation}(P) \cap \{B \mid B \textit{ is } (k,t)\textit{-special}\} \in \mathrm{L}.$$

Proof idea. To decide $\textsc{saturation}(P)$ on (k,t)-special graphs B, compute a shrink B', which has bounded tree width, and apply the logspace version of Courcelle's Theorem. ◀

Graphs With Self-Saturating Mixed Cycles We extend the class of graphs that our logspace machines can handle to graphs that are not necessarily (k,t)-special, but at least contain a *mixed self-saturating cycle*. A *self-saturating cycle of a basic graph $B = (V,E)$ with respect to a pattern graph $P = (C, A^\oplus, A^\ominus)$* is a sequence $(v_1, v_2, \dots, v_{n+1})$ of vertices in V for $n \geq 2$ where the v_i for $i \in \{1, \dots, n\}$ are all different, $v_{n+1} = v_1$, and we can assign colors $c\colon \{v_1, \dots, v_n\} \to C$ such that for all $i \in \{1, \dots, n\}$ we have: if $\{v_i, v_{i+1}\} \in E$, then $(c(v_i), c(v_{i+1})) \in A^\oplus$; and if $\{v_i, v_{i+1}\} \notin E$, then $(c(v_i), c(v_{i+1})) \in A^\ominus$. In other words, B restricted to $\{v_1, \dots, v_n\}$ can be saturated with the "natural" witness function that "moves along" the cycle. The following is an easy observation concerning self-saturating cycles:

▶ **Lemma 3.10.** *For every $B \in \textsc{saturation}(P)$ there is a self-saturating cycle in B for P.*

Proof idea. Just "follow the witness function" until it runs into a cycle. ◀

A self-saturating cycle is *mixed* if for some $i, j \in \{1, \dots, n\}$ we have $\{v_i, v_{i+1}\} \in E$ and $\{v_j, v_{j+1}\} \notin E$, otherwise the cycle is called *pure*. In Figure 3, (b,c,f,b) is a pure self-saturating cycle and (a,c,f,d,a) is a mixed self-saturating cycle as proved by the two example colorings. Two facts concerning mixed self-saturating cycles will be important:

▶ Fact 3.11 ([11, Lemma 6.5]). For every pattern graph P there is a constant d such that every basic graph that has a mixed self-saturating cycle with respect to P also has such a cycle of length at most d.

▶ Fact 3.12 ([11, Section 6.3]). For each pattern graph P there exist k and t such that $B \in \text{SATURATION}(P)$ holds for all graphs B that contain a mixed self-saturating cycle but are not (k,t)-special.

▶ **Lemma 3.13.** *For every pattern graph P, we have*

$$\text{SATURATION}(P) \cap \{B \mid B \textit{ contains a mixed self-saturating cycle}\} \in \mathrm{L}.$$

Proof idea. Fact 3.11 gives a logspace procedure for detecting mixed self-saturating cycles. Combine it with Fact 3.12 and Lemma 3.9. ◀

Arbitrary Basic Graphs The last step is to extend our algorithm to graphs that do not contain mixed self-saturating cycles (and are not (k,t)-special, but this will no longer be important). Clearly, by considering the union of the languages from Lemma 3.13 above and Lemma 3.14 below, we see that $\text{SATURATION}(P) \in \mathrm{L}$ holds for all pattern graphs P.

▶ **Lemma 3.14.** *For every pattern graph P, we have*

$$\text{SATURATION}(P) \cap \{B \mid B \textit{ contains no mixed self-saturating cycle}\} \in \mathrm{L}.$$

Proof idea. Theorem 5.17 of [11] provides a polynomial-time algorithm for deciding $B \in \text{SATURATION}(P)$ when there are no mixed self-saturating cycles in B. The algorithmically relevant operations in the proof are (1) computing complement graphs (exchanging edges and non-edges), (2) computing connected components, and (3) applying Courcelle's Theorem to these components. Clearly, all three operations can also be implemented in logarithmic space using Reingold's Theorem and the logspace version of Courcelle's Theorem. ◀

3.3 E_1ae Over Basic Graphs: From L to FO

Our final task for this paper is showing $\mathrm{FD}_{\text{basic}}(E_1ae) \subseteq \mathrm{FO}$.[5] By Fact 3.3, it suffices to show $\text{SATURATION}(P) \in \mathrm{FO}$ for all pattern graphs with *two* colors (denoted "white" and "black" in the following) and this will be our objective in this section.[6]

In the previous section we proved $\text{SATURATION}(P) \in \mathrm{L}$ for all pattern graphs by developing logspace algorithms that worked for larger and larger classes of graphs. However, this approach is bound to fail for the class FO since properties like "the graph is a tree" (let alone "the graph is (k,t)-special") are not expressible in first-order logic. Instead, in this section we show $\text{SATURATION}(P) \in \mathrm{FO}$ directly for each possible pattern graph with two colors.

The simplest case arises when $P = (C, A^{\oplus}, A^{\ominus})$ is acyclic (meaning that the directed graph $(C, A^{\oplus} \cup A^{\ominus})$ is acyclic): Lemma 3.10 shows that we then have $\text{SATURATION}(P) = \emptyset$ since self-saturating cycles cannot exist for such P. Thus, we only need to consider pattern graphs P with cycles (self-loops are also cycles, here). Since P only has two colors, there are only few ways in which such cycles may arise. The more cycles there are, the easier it will be to color the graph, so we first handle the case that there are cycles both in $A^{\oplus}$ and $A^{\ominus}$, then that there is a cycle in $A^{\oplus}$ or in $A^{\ominus}$, and finally that there is only a cycle in $A^{\oplus} \cup A^{\ominus}$.

---

[5] In contrast, Lemmas 2.1 and 2.2 show that if we have *two* monadic quantifiers or one *binary* quantifier, the prefix class contains an L-complete problem.

[6] In contrast, using three colors we can describe L-complete problems: $\text{SATURATION}(P) = A_3$ where P contains a $\oplus$-labeled 3-cycle and A_3 is the L-complete language from Table 1.

▶ **Lemma 3.15.** *Let $P = (\{black, white\}, A^\oplus, A^\ominus)$ contain cycles both in $A^\oplus$ and $A^\ominus$. Then* $\textsc{saturation}(P)$ *contains all graphs with at least two vertices (and is hence in* FO*).*

Proof idea. The interesting case is a $\oplus$-cycle involving both colors. In each connected component, choose a vertex and color the vertices black or white depending on whether they have odd or even distance from the chosen vertex. The witness of a vertex is a vertex nearer to the chosen vertex (except for the chosen vertex, whose witness is any neighbor). ◀

▶ **Lemma 3.16.** *Let $P = (\{black, white\}, A^\oplus, A^\ominus)$ contain a cycle in $A^\oplus$ or in $A^\ominus$. Then* $\textsc{saturation}(P) \in \mathrm{FO}$.

Proof idea. The most interesting case is exactly the pattern graph shown in Figure 3. We distinguish the cases that the input graph B consists of a matching plus some isolated vertices or has a connected component of size at least three. If the matching is a single edge, $B \notin \textsc{saturation}(P)$; otherwise, $B \in \textsc{saturation}(P)$ since one can devise similar methods as in the previous lemma for coloring the graph and constructing a witness function. ◀

We are left with the case that the set $A^\oplus \cup A^\ominus$ contains a cycle, but neither $A^\oplus$ nor $A^\ominus$ does. This is only possible when P is either $\bullet \underset{\ominus}{\overset{\oplus}{\rightleftarrows}} \circ$ or $\bullet \underset{\oplus}{\overset{\ominus}{\rightleftarrows}} \circ$. For this special kind of cycle, there is an analogue of Fact 3.12 that does not refer to (k, t)-special graphs:

▶ Fact 3.17 ([11, Lemma 6.7]). For every pattern graph P, we have $B \in \textsc{saturation}(P)$ for all B that contain a self-saturating cycle for P on which $\oplus$- and $\ominus$-arcs alternate.

▶ **Lemma 3.18.** *Let $P = (\{black, white\}, A^\oplus, A^\ominus)$ contain a cycle in $A^\oplus \cup A^\ominus$, but none in $A^\oplus$ nor in $A^\ominus$. Then* $\textsc{saturation}(P) \in \mathrm{FO}$.

Proof idea. Use Fact 3.11 to detect a mixed self-saturating cycle using d existential first-order quantifiers. The existence of such a cycle in B is a necessary condition for $B \in \textsc{saturation}(P)$ by Lemma 3.10 and also a sufficient condition by Fact 3.17. ◀

4 Conclusion

In the present paper we have completely classified the first-order reduction closures of prefix classes of ESO logic over directed, undirected, and basic graphs: each one of them is equal to one of the standard classes FO, L, NL, or NP. It turned out that the prefix classes for directed and undirected graphs are always the same, but often differ from the prefix classes for basic graphs. Especially interesting prefixes that mark the border between one complexity class and the next are E_1ae, E^*ae, and Eaa.

A natural question that arises is: Can we find a prefix class whose reduction closure is P? By the results of the present paper, this cannot be an ESO prefix class, unless unlikely collapses occur. However, what about prefix classes of general second-order logic? We may similarly ask whether any class other than L, NL, and the classes of the polynomial hierarchy can be characterized by a prefix class of second-order logic.

Together with the results from [6], we now have a fairly complete picture of the complexity of all ESO prefix classes over directed graphs, undirected graphs, basic graphs, and strings. Concerning arbitrary logical structures, Gottlob et al. [11] already point out that their P-NP-dichotomy for directed graphs generalizes to the collection of all finite structures over any relational vocabulary that contains a relation symbol of arity at least two; and it is not hard to see that our Theorem 1.1 also generalizes in this way (a closer look at the FO and NL upper bounds in [11] shows that they hold for arbitrary structures). The complexity of prefix classes over other special structures is, however, still open, including those of trees, infinite words, and bipartite graphs.

References

1. Eric Allender, Michael Bauland, Neil Immerman, Henning Schnoor, and Heribert Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. *Journal of Computer and System Sciences*, 75(4):245–254, 2009.
2. Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Springer-Verlag, Berlin, 1997.
3. Julius R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.
4. Stephen A. Cook and Pierre McKenzie. Problems complete for deterministic logarithmic space. *Journal of Algorithms*, 8(5):385–394, 1987.
5. Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.
6. Thomas Eiter, Georg Gottlob, and Yuri Gurevich. Existential second order logic over strings. *Journal of the ACM*, 47(1):77–131, 2000.
7. Michael Elberfeld, Martin Grohe, and Till Tantau. Where first-order and monadic second-order logic coincide. In *Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2012)*, pages 265–274. IEEE Computer Society, 2012.
8. Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of Bodlaender and Courcelle. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010)*, pages 143–152, 2010.
9. Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of Computation*, 7:43–74, 1974.
10. Stéphane Földes and Peter L. Hammer. Split graphs. In *Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing*, Congressus Numerantium XIX, pages 311–315. Louisiana State Univeristy, Baton Rouge, Louisiana, 1977.
11. Georg Gottlob, Phokion G. Kolaitis, and Thomas Schwentick. Existential second-order logic over graphs: Charting the tractability frontier. *Journal of the ACM*, 51(2):312–362, 2004.
12. Edith Hemaspaandra, Holger Spakowski, and Mayur Thakur. Complexity of cycle length modularity problems in graphs. In *Proceedings of the 6th Latin American Symposium on Theoretical Informatics (LATIN 2004)*, volume 2976 of *Lecture Notes in Computer Science*, pages 509–518. Springer, 2004.
13. Neil Immerman. *Descriptive Complexity Theory*. Springer-Verlag, New York, 1998.
14. Neil Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.
15. Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Symposium on Theory of Computing (STOC 1978)*, pages 216–226. ACM Press, 1978.
16. Till Tantau. Existential second-order logic over graphs: A complete complexity-theoretic classification. Technical Report arxiv:1412.6396 [cs.LO], ArXiv e-prints, 2014.