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 $\exists R_1 \cdots \exists R_r \forall x \exists y$ 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
where the $R_i$ are second-order variables, the $x_i$ are first-order variables, and $\psi$ is quantifier-free. Formulas like $\phi$ are common in the literature. For instance, consider the following formula:

$$\exists R \exists x \exists y \psi$$

where $R$ denotes a binary relation, $x$ and $y$ are first-order variables, and $\psi$ is a formula. The prefix type of the formula $\phi$ is given by the patterns of prefix types such as $[a, e, E, E, \ldots]$, where $a$ denotes a universal and existential first-order quantifier, $e$ denotes the presence of an existential second-order quantifier, and $E$ denotes the presence of an existential first-order quantifier.

$\phi$ is 3-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$, where $\mathcal{R}$ is a relational symbol. For instance, $\phi$ is 2-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$. The prefix closure of $\phi$ is given by the patterns of prefix types such as $[a, e, E, E, \ldots]$, where $a$ denotes a universal and existential first-order quantifier, $e$ denotes the presence of an existential second-order quantifier, and $E$ denotes the presence of an existential first-order quantifier.

$\phi$ is 3-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$, where $\mathcal{R}$ is a relational symbol. For instance, $\phi$ is 2-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$.

The prefix closure of $\phi$ is given by the patterns of prefix types such as $[a, e, E, E, \ldots]$, where $a$ denotes a universal and existential first-order quantifier, $e$ denotes the presence of an existential second-order quantifier, and $E$ denotes the presence of an existential first-order quantifier.

$\phi$ is 3-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$, where $\mathcal{R}$ is a relational symbol. For instance, $\phi$ is 2-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$.

The prefix closure of $\phi$ is given by the patterns of prefix types such as $[a, e, E, E, \ldots]$, where $a$ denotes a universal and existential first-order quantifier, $e$ denotes the presence of an existential second-order quantifier, and $E$ denotes the presence of an existential first-order quantifier.

$\phi$ is 3-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$, where $\mathcal{R}$ is a relational symbol. For instance, $\phi$ is 2-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$.

The prefix closure of $\phi$ is given by the patterns of prefix types such as $[a, e, E, E, \ldots]$, where $a$ denotes a universal and existential first-order quantifier, $e$ denotes the presence of an existential second-order quantifier, and $E$ denotes the presence of an existential first-order quantifier.

$\phi$ is 3-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$, where $\mathcal{R}$ is a relational symbol. For instance, $\phi$ is 2-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$.

The prefix closure of $\phi$ is given by the patterns of prefix types such as $[a, e, E, E, \ldots]$, where $a$ denotes a universal and existential first-order quantifier, $e$ denotes the presence of an existential second-order quantifier, and $E$ denotes the presence of an existential first-order quantifier.

$\phi$ is 3-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$, where $\mathcal{R}$ is a relational symbol. For instance, $\phi$ is 2-colorable if and only if $\exists R \exists x \exists y \psi$ is contained in $\exists \exists \mathcal{R}$.
MODELsbasic(\(\phi_3\)-colorable) = 3-COLORABLE (ignoring coding issues). Next, for a prefix type pattern \(P\), let \(FD_{\text{directed}}(P) = \{\text{MODELs}_{\text{directed}}(\phi) \mid \phi\) has a prefix type in \(P)\) and define \(FD_{\text{undirected}}(P)\) and \(FD_{\text{basic}}(P)\) similarly for undirected and basic graphs. “FD” stands for “Fagin-definable” and Fagin’s Theorem can be stated succinctly as \(FD_{\text{strings}}(E^*(ae)^*) = \text{NP}.\)

As stated earlier, in the context of syntactic fragments of \(\text{ESO}\) 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 \textit{first-order reductions} \cite{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 \(A \subseteq B\) if \(A\) can be reduced to \(B\) using first-order reductions. Let us write \(FD_{\text{directed}}(P) = \{A \mid A \subseteq B \in FD_{\text{directed}}(P)\}\) for the reduction closure of \(FD_{\text{directed}}(P)\) and define \(FD_{\text{undirected}}(P)\) and \(FD_{\text{basic}}(P)\) similarly.

\[\begin{array}{c|c|c}
\text{Theorem 1.1 (Main Result).} & \text{The following table completely classifies all prefix classes of ESO logic over basic graphs (upper part) and undirected and directed graphs (lower part).}\(^2\) & \text{If } P \text{ is at least one of . . . } \\
\text{and at most one of . . . , then} & & \text{and most one of . . . , then} \\
\hline
E_1E_1ae, E_2ae & (ae)^*, E^*e^a, E_1ae & FD_{\text{basic}}(P) = \text{FO} \\
E_1aa & E^*ae & FD_{\text{basic}}(P) = L \\
E_1aa, E_1E_1aa, E_2eaa, E_1eae, E_1aee, E_1aea, E_1eaa, E_1aaa & E_1e^*aa & FD_{\text{basic}}(P) = \text{NL} \\
E_1aa, E_1E_1aa, E_2eaa, E_1eae, E_1aee, E_1aea, E_1aae & E^*(ae)^* & FD_{\text{basic}}(P) = \text{NP} \\
\hline
E_1aa & (ae)^*, E^*e^a & FD_{\text{undirected}}(P) = FD_{\text{directed}}(P) = \text{FO} \\
E_1e^*aa, Eaa & FD_{\text{undirected}}(P) = FD_{\text{directed}}(P) = \text{NL} \\
E_1aaa, E_1E_1aa, E_2eaa, E_1eae, E_1aee, E_1aea, E_1eaa, E_1aaa & E^*(ae)^* & FD_{\text{undirected}}(P) = FD_{\text{directed}}(P) = \text{NP} \\
\end{array}\]

Note that we always have \(FD_{\text{undirected}}(P) = FD_{\text{directed}}(P)\), 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 \(FD_{\text{undirected}}(E_1aa) \subseteq \text{L}\). However, using a gadget construction, we will show that \(FD_{\text{undirected}}(E_1aa)\) contains an \(\text{NL}\)-complete problem.

As an application of the theorem, let us use it to prove \textit{even-cycle} \(\in \text{L}\), 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 \cite{12} for a discussion and variants. The idea is to consider the following \(\text{ESO}\) formulas:

\[\phi_m = \exists C_1 \cdots \exists C_m \forall x \forall y \left(E(x,y) \land \bigvee_{i=1}^m \left(C_i(x) \land C_{i \bmod m+1}(y) \land \bigwedge_{j \neq i} \neg C_j(x)\right)\right).\]

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 \(\phi_m\) 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 \(\text{FO}\) we refer to “first-order logic with access to the bit predicate,” which is the same as logarithmic-time-uniform \(\text{AC}^0\).

\(^2\) 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 $\text{FD}_{\text{basic}}(E^*ae) \subseteq L$ and $\text{FD}_{\text{basic}}(E_1ae) \subseteq \text{FO}$ require a sophisticated technical machinery. In both cases, our proofs follow the ideas of a 35-page proof of $\text{FD}_{\text{basic}}(E^*ae) \subseteq \text{P}$ 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 $\text{P}$ 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 $\text{FD}_{\text{basic}}(E_1ae) \subseteq \text{FO}$, 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, $\text{FD}_{\text{basic}}(E_1 E_1 ae) = \text{FD}_{\text{basic}}(E_2 ae) = L$.

Concerning the remaining claims from Theorem 1.1 that are not already proved in [11], two cases are noteworthy: Proving that $\text{FD}_{\text{basic}}(E_1 eaa)$ contains an $\text{NL}$-complete problem turns out to require a nontrivial gadget construction. Proving $\text{FD}_{\text{basic}}(E_1 aa) \subseteq L$ requires a reformulation of the problems in $\text{FD}_{\text{basic}}(E_1 aa)$ 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 $E_1 ae$ 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 $\text{FD}_{\text{strings}}(P)$ is either equal to $\text{NP}$; is not equal to $\text{NP}$ but contains an $\text{NP}$-complete problem; is equal to REG; or is a subclass of FO. Interestingly, two classes of special interest are $\text{FD}_{\text{strings}}(E^*_1 ae)$ and $\text{FD}_{\text{strings}}(E^*_1 aa)$, 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 $\text{FD}_{\text{basic}}(E^*_1 ae) = \text{FD}_{\text{basic}}(E_1 E_1 ae) = L$, while $\text{FD}_{\text{basic}}(E_1 ae) = \text{FO}$, and $\text{FD}_{\text{basic}}(E^*_1 aa) = \text{FD}_{\text{basic}}(E_1 E_1 aa) = \text{NP}$, while $\text{FD}_{\text{basic}}(E_1 aa) = L$.

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 $\text{FD}_{\text{basic}}(E^* ae) \subseteq L$. 

and \( \text{FD}_{\text{basic}}(E_1ae) \subseteq \text{FO} \). 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 \( \text{FD}_{\text{basic}}(E^*ae) \) at the time of writing, which hinges on a problem called \( \text{satu}(P) \): “Note also that for each \( P \), \( \text{satu}(P) \) is probably not a \( \text{PTIME} \)-complete set. […] This is due to the check for bounded treewidth, which is in \( \text{LOGCFL} \) (cf. Wanke [1994]) but not known to be in \( \text{NL} \).” 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 \( \text{L} \). This does not mean, however, that the proof of [11] immediately yields \( \text{FD}_{\text{basic}}(E^*ae) \subseteq \text{L} \) 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 \( \text{L} \), \( \text{NL} \), or \( \text{NP} \). The upper bounds are proved in Section 3, where we prove, in order, \( \text{FD}_{\text{basic}}(Eaa) \subseteq \text{L} \), \( \text{FD}_{\text{basic}}(E^*ae) \subseteq \text{L} \), and \( \text{FD}_{\text{basic}}(E_1ae) \subseteq \text{FO} \) 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 \( \text{L} \) and \( \text{NL} \)

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 \( \text{L} \), \( \text{NL} \), or \( \text{NP} \). 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 \( \text{NP} \) cases have already been settled in [11].

### Table 1

<table>
<thead>
<tr>
<th>Claim</th>
<th>Hard problem</th>
<th>Proved where</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower bounds for basic graphs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_1E_1ae) )</td>
<td>( \supseteq \text{L} )</td>
<td>( A_3 )</td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_2ae) )</td>
<td>( \supseteq \text{L} )</td>
<td>( A_2 )</td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_1aa) )</td>
<td>( \supseteq \text{L} )</td>
<td>2-COLORABLE</td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_1eaa) )</td>
<td>( \supseteq \text{NL} )</td>
<td>UNREACH</td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_1aaa) )</td>
<td>( \supseteq \text{NP} )</td>
<td>POSITIVE-ONE-IN-THREE-3SAT</td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_1E_1aa) )</td>
<td>( \supseteq \text{NP} )</td>
<td>3-COLORABLE</td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_2eaa) )</td>
<td>( \supseteq \text{NP} )</td>
<td>3-COLORABLE</td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_1eae) )</td>
<td>( \supseteq \text{NP} )</td>
<td>3SAT</td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_1ae) )</td>
<td>( \supseteq \text{NP} )</td>
<td>NOT-ALL-EQUAL-3SAT</td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_1aea) )</td>
<td>( \supseteq \text{NP} )</td>
<td>POSITIVE-ONE-IN-THREE-3SAT</td>
</tr>
<tr>
<td>( \text{FD}_{\text{basic}}(E_1aea) )</td>
<td>( \supseteq \text{NP} )</td>
<td>POSITIVE-ONE-IN-THREE-3SAT</td>
</tr>
</tbody>
</table>

Remaining lower bounds for undirected and, thereby, also for directed graphs

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{FD}_{\text{undirected}}(E_1aa) )</td>
<td>( \supseteq \text{NL} )</td>
<td>UNREACH</td>
</tr>
<tr>
<td>( \text{FD}_{\text{undirected}}(E_1ae) )</td>
<td>( \supseteq \text{NP} )</td>
<td>3SAT</td>
</tr>
</tbody>
</table>
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.)

\textbf{Lemma 2.1.} $A_3 \in \text{FD}_{\text{basic}}(E_1 E_1 ae)$.

\textbf{Proof idea.} Use $\phi_3$ from equation (1), but get rid of one of the second-order quantifiers.

\textbf{Lemma 2.2.} $A_2 \in \text{FD}_{\text{basic}}(E_2 ae)$.

\textbf{Proof idea.} Use $\exists F \forall x \exists y (E(x,y) \land F(x,y) \land \neg F(y, x) \land (F(x, x) \leftrightarrow \neg F(y, y)))$.

\textbf{Lemma 2.3.} \textsc{unreach} reduces to a problem in $\text{FD}_{\text{basic}}(E_1 eaa)$ and also to a problem in $\text{FD}_{\text{undirected}}(E_1 aa)$.

\textbf{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 $E_1 eaa$ allows us to pick a vertex and then single out the set of vertices connected to it. Thus, essentially, it suffices to show that \textsc{unreach} reduces to $\text{MODELS}_{\text{basic}}(\exists M \forall x \forall y \psi)$ where $\psi$ 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 $S$, and...
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.

<table>
<thead>
<tr>
<th>Claims for basic graphs</th>
<th>Proved where</th>
<th>Claims for directed graphs</th>
<th>Proved where</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{FD}<em>{\text{basic}}(E</em>{1}ac)$</td>
<td>$\subseteq$ FO</td>
<td>$\text{FD}_{\text{directed}}((ac)^{*})$</td>
<td>$\subseteq$ FO</td>
</tr>
<tr>
<td>$\text{FD}_{\text{basic}}(E^{*}ac)$</td>
<td>$\subseteq$ L</td>
<td>$\text{FD}_{\text{directed}}(E^{<em>}e^{</em>}a)$</td>
<td>$\subseteq$ FO</td>
</tr>
<tr>
<td>$\text{FD}_{\text{basic}}(Eaa)$</td>
<td>$\subseteq$ L</td>
<td>$\text{FD}<em>{\text{directed}}(E</em>{1}e^{*}aa)$</td>
<td>$\subseteq$ NL</td>
</tr>
<tr>
<td>$\text{FD}_{\text{basic}}(Eaa)$</td>
<td>$\subseteq$ L</td>
<td>$\text{FD}_{\text{directed}}(Eaa)$</td>
<td>$\subseteq$ NL</td>
</tr>
<tr>
<td>$\text{FD}_{\text{directed}}(E^{<em>}(ae)^{</em>})$</td>
<td>$\subseteq$ NP</td>
<td>$\text{Fagin’s Theorem}$</td>
<td></td>
</tr>
</tbody>
</table>

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 $\psi$ 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 $\phi$ 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, $\text{FD}_{\text{basic}}(Eaa) \subseteq L$, is proved in two steps: First, we reformulate the problems in $\text{FD}_{\text{basic}}(Eaa)$ 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
B: \begin{align*}
&\text{Figure 3 Example of a pattern graph } P = (C, A^0, A^\oplus) \text{ with two “colors” black and white} \\
\text{(so } C = \{\text{black, white}\}, A^0 = \{(\text{black, black}), (\text{white, black})\}, \text{ and } A^\oplus = \{(\text{black, white})\}) \text{ and an uncolored (“gray”) example graph } B. \text{ We have } B \in \text{saturation}(P) \text{ as shown by two examples of legal colorings of } B \text{ together with witness functions } w \text{ (in gray).}
\end{align*}
(1) \( x \neq w(x) \), (2) if \( \{x, w(x)\} \in E \), then \((c(x), c(w(x))) \in A^0\), and (3) if \( \{x, w(x)\} \notin E \), then \((c(x), c(w(x))) \in A^0\). 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 \( \text{saturation}(P) \) 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 \( \forall x \) we must pick to make a formula \( \phi \) of the form \( \exists M_1 \cdots \exists M_n \forall x \exists y \psi \) true. The pattern graph encodes the restrictions imposed by \( \psi \) 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 \( \psi \) is quantifier-free, there is a pattern graph \( P \) with \( 2^n \) vertices such that \( \text{models}_{\text{basic}}(\phi) = \text{saturation}(P) \).

Thus, it remains to show \( \text{saturation}(P) \in L \) 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 \( \text{mso} \)-formula \( \phi \) and \( t \geq 1 \) we have

\[
\text{models}_{\text{basic}}(\phi) \cap \{G \mid G \text{ has tree width at most } t\} \in \text{LTIME}.
\]

Gottlob et al. apply this theorem to show that when the input graphs \( B \) have bounded tree width, we can decide whether \( B \in \text{saturation}(P) \) holds in polynomial time: the property \( B \in \text{saturation}(P) \) is easily described in \( \text{mso} \) 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 \( \text{mso} \)-formula \( \phi \) and \( t \geq 1 \) we have

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

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 $A_1, \ldots, A_k$ 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 $A_1, \ldots, A_k$ 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 $B \in \text{Saturation}(P)$ if, and only if, $B' \in \text{Saturation}(P)$.

**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 $(v_1, \ldots, v_k)$ 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

$$\text{Saturation}(P) \cap \{B \mid B \text{ is } (k,t)\text{-special}\} \in L.$$

**Proof idea.** To decide $\text{Saturation}(P)$ 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 $P = (C,A^P,A^\circ)$ is a sequence $(v_1,v_2,\ldots,v_n)$ of vertices in $V$ for $n \geq 2$ where the $v_i$ for $i \in \{1,\ldots,n\}$ are all different, $v_{n+1} = v_1$, and we can assign colors $c: \{v_1,\ldots,v_n\} \rightarrow C$ such that for all $i \in \{1,\ldots,n\}$ we have: if $\{v_i,v_{i+1}\} \in E$, then $(c(v_i),c(v_{i+1})) \in A^\circ$; and if $\{v_i,v_{i+1}\} \notin E$, then $(c(v_i),c(v_{i+1})) \in A^\oplus$. In other words, $B$ restricted to the $\{v_1,\ldots,v_n\}$ 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 $B \in \text{Saturation}(P)$ 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 $i,j \in \{1,\ldots,n\}$ 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

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


### 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 $\text{saturation}(P) \in L$ holds for all pattern graphs $P$.

Lemma 3.14. For every pattern graph $P$, we have

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

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 $\text{FD}_{\text{basic}}(E_{1ae}) \subseteq \text{FO}$. By Fact 3.3, it suffices to show $\text{saturation}(P) \in \text{FO}$ for all pattern graphs with two colors (denoted “white” and “black” in the following) and this will be our objective in this section.

In the previous section we proved $\text{saturation}(P) \in L$ 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 $\text{saturation}(P) \in \text{FO}$ 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 $P = (\{\text{black}, \text{white}\}, A^\oplus, A^\ominus)$ contain cycles both in $A^\oplus$ and $A^\ominus$. Then \text{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 $P = (\{\text{black}, \text{white}\}, A^\oplus, A^\ominus)$ contain a cycle in $A^\oplus$ or in $A^\ominus$. Then \text{saturation}(P) \in \text{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, $B \notin \text{saturation}(P)$; otherwise, $B \in \text{saturation}(P)$ 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 $\blacklozenge \circledcirc \lozenge$ or $\blacklozenge \circledcirc \lozenge$. 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 $B \in \text{saturation}(P)$ for all $B$ that contain a self-saturating cycle for $P$ on which $\oplus$- and $\ominus$-arcs alternate.

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

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 $B \in \text{saturation}(P)$ 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 $E_1^{\text{ae}}, E^{*\text{ae}},$ and $E_{\text{aa}}$.

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


