On Existential MSO and Its Relation to ETH

ROBERT GANIAN, Algorithms and Complexity Group, TU Wien
RONALD DE HAAN, Institute for Logic, Language and Computation, University of Amsterdam
IYAD KANJ, School of Computing, DePaul University
STEFAN SZEIDER, Algorithms and Complexity Group, TU Wien

Impagliazzo et al. proposed a framework, based on the logic fragment defining the complexity class SNP, to identify problems that are equivalent to $k$-CNF-Sat modulo subexponential-time reducibility (serf-reducibility). The subexponential-time solvability of any of these problems implies the failure of the Exponential Time Hypothesis (ETH). In this article, we extend the framework of Impagliazzo et al. and identify a larger set of problems that are equivalent to $k$-CNF-Sat modulo serf-reducibility. We propose a complexity class, referred to as Linear Monadic NP, that consists of all problems expressible in existential monadic second-order logic whose expressions have a linear measure in terms of a complexity parameter, which is usually the universe size of the problem.

This research direction can be traced back to Fagin’s celebrated theorem stating that NP coincides with the class of problems expressible in existential second-order logic. Monadic NP, a well-studied class in the literature, is the restriction of the aforementioned logic fragment to existential monadic second-order logic. The proposed class Linear Monadic NP is then the restriction of Monadic NP to problems whose expressions have linear measure in the complexity parameter.

We show that Linear Monadic NP includes many natural complete problems such as the satisfiability of linear-size circuits, dominating set, independent dominating set, and perfect code. Therefore, for any of these problems, its subexponential-time solvability is equivalent to the failure of ETH. We prove, using logic games, that the aforementioned problems are inexpressible in the monadic fragment of SNP, and hence, are not captured by the framework of Impagliazzo et al. Finally, we show that Feedback Vertex Set is inexpressible in existential monadic second-order logic, and hence is not in Linear Monadic NP, and investigate the existence of certain reductions between Feedback Vertex Set (and variants of it) and 3-CNF-Sat.

CCS Concepts: • Theory of computation → Complexity theory and logic;

Additional Key Words and Phrases: Exponential time hypothesis (ETH), monadic second-order logic, subexponential time complexity, serf-reducibility, logic games

Kanj was partially supported by DFG, project NI 369/12. Ganian, de Haan and Szeider were partially supported by the Austrian Science Fund (FWF), projects P26200 and P26696. Ganian acknowledges support by FWF project P31336. Szeider acknowledges support by FWF project 32441 and by the WWTF project ICT19-065.

Authors’ addresses: R. Ganian and S. Szeider, Algorithms and Complexity Group, TU Wien, Institute of Logic and Computation, Favoritenstraße 9–11, E192-01, 1040 Wien, Austria; emails: rganian@gmail.com, stefan@szeider.net; R. de Haan, Institute for Logic, Language and Computation, University of Amsterdam, Postbus 94242, 1090 GE, Amsterdam, the Netherlands; email: me@ronalddehaan.eu; I. Kanj, School of Computing, DePaul University, 243 S. Wabas Avenue, 4th floor, Chicago, IL 60604, USA; email: ikanj@cdm.depaul.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.
1942-3454/2020/09-ART22 $15.00
https://doi.org/10.1145/3417759
1 INTRODUCTION

Motivation and related work. The area of exact algorithms seeks moderately exponential-time algorithms for NP-hard problems that improve on the trivial brute-force algorithms. This area of research has been gaining a lot of traction in the past two decades. Nowadays, there is a large set of important NP-complete problems such that for each problem there is a long list of exact algorithms, each improving slightly on the running time of the preceding one; we refer the interested reader to the literature \[31\] for exposure to the area of exact algorithms. Perhaps the most well-known—and important—NP-hard problem is the satisfiability of Boolean formulas in conjunctive normal form, abbreviated CNF-SAT. Its importance stems from its practical relevance as well as from the canonical role it plays for the complexity class NP. Despite all the exercised efforts over the past several decades, no significant improvement to the trivial brute-force algorithm that solves CNF-SAT in time $O^*(2^n)$ has been made, where $n$ is the number of Boolean variables in the input formula. It has become a common belief among a large number of researchers in the area of exact algorithms that no $O^*(2^{cn})$-time algorithm, for any constant $c < 1$, exists for CNF-SAT. In particular, the existence of such an algorithm for CNF-SAT would lead to the falsification of the Strong Exponential-Time Hypothesis (SETH), which has been used for proving many lower-bound results recently, as shall be discussed at the end of this section. The SETH states that the limit of $c_k$ as $k$ goes to infinity, where $c_k \leq 1$ is the smallest constant such that $k$-CNF-SAT is solvable in time $O(2^{c_k n})$, is 1. (The $k$-CNF-SAT problem, for $k \geq 3$, is the restriction of CNF-SAT to instances in which the clause-width is at most $k$.) We mention that for the $k$-CNF-SAT problem, there is a long list of moderately exponential-time algorithms culminating in the currently best deterministic algorithm (for a general value of $k$) running in time $O((2 - 2/(k + 1))^n)$ \[20\].

The sequence of improvements in the running time of moderately exponential-time algorithms for $k$-CNF-SAT, as well as for other NP-complete problems, led researchers to ask whether we can “indefinitely” keep improving the exponential running time of exact algorithms for these problems. More formally, it triggered the question of whether an algorithm for CNF-SAT exists in time $O(2^{o(n)})$, or equivalently in time $O(2^{\varepsilon n})$ for every $\varepsilon > 0$, exists; an algorithm with such running time is referred to as a subexponential-time algorithm. In their seminal paper, Impagliazzo et al. \[35\] investigated this question. They proved that the existence of a subexponential-time algorithm for $k$-CNF-SAT, for any integer-constant $k \geq 3$, is related to its membership in the class SNP, introduced by Papadimitriou and Yannakakis \[44\]. The class SNP consists of all search problems expressible by second-order existential formulas whose first-order part is universal; we shall refer to the aforementioned logic fragment as SNP logic. For a problem in SNP, they defined a complexity parameter based on its SNP logic expression. They introduced the notion of completeness for the class SNP under serf-reductions, which are subexponential-time preserving reductions with respect to the aforementioned complexity parameter, and identified a class of problems that are complete for SNP under serf-reducibility; the subexponential-time solvability of any of these problems with respect to its corresponding complexity parameter implies the subexponential-time solvability of all problems in SNP. They proved that many well-known NP-hard problems are SNP-complete under serf-reducibility with respect to a complexity parameter that is linear in the natural

\[1\]The $O^*$ notation suppresses polynomial factors in the input length.
universe size of the problem, including \( k \)-CNF-Sat (the universe is the Boolean variables in the formula), Vertex Cover, Independent Set, and \( c \)-Colorability (\( c \geq 3 \)) (universe is the vertex set of the graph), for which extensive efforts to develop subexponential-time algorithms have been made in the past three decades with no success. This led them to formulate the Exponential Time Hypothesis (ETH), stating that \( k \)-CNF-Sat is not solvable in subexponential-time, which is equivalent to the statement that not all SNP problems are solvable in subexponential time. ETH has become a standard hypothesis in complexity theory for proving hardness results that is closely related to the computational intractability of a large class of well-known NP-hard problems, measured from a number of different angles, such as subexponential-time complexity, fixed-parameter tractability, and approximation [11, 12, 18, 42, 43].

Our results. The SNP logic framework developed by Impagliazzo et al. [35] captures some well-known NP-complete problems that are serf-reducible to \( k \)-CNF-Sat with respect to their universe size, but fails to capture many others. Using logic games, we prove in Section 3 that there are many natural NP-complete problems that are not captured by this SNP logic framework, and yet are serf-reducible to \( k \)-CNF-Sat. Examples of such problems include the satisfiability problem for linear-size circuits (Linear Circuit-Sat), Dominating Set, Independent Dominating set, and Perfect Code. The restrictedness of the SNP-based framework, due to the restrictedness of SNP logic to allowing only universal quantifiers in the first-order part, prevents the expressibility of problems such as the aforementioned ones, as we show in this article that none of these problems can be expressed in SNP logic with a complexity parameter that is linear in the natural universe size of the problem.

We propose a complexity class, referred to as Linear Monadic NP, that captures more problems than the SNP-based framework of Impagliazzo et al. [35]. Fagin’s celebrated theorem [24] states that NP coincides with the class of properties expressible in existential second-order logic. The class monadic NP, introduced by Fagin et al. [27], is the restriction of NP to problems expressible in monadic second-order logic in which the second-order part is existential, that is, all second-order variables have arity at most one (i.e., set variables), and no universal quantification is allowed over these relations. The class monadic NP is a well-studied complexity class, and several properties have been shown to be inexpressible in the logic fragment defining this class [3, 4, 27, 36, 39]. We show in this article that the class monadic NP plays an important role as well in identifying problems that are equivalent to \( k \)-CNF-Sat under serf-reducibility. We define a measure (for expressions) and use it to define a logic fragment consisting of the restriction of existential monadic second-order logic to expressions whose measure is linear in the complexity parameter, where the complexity parameter is defined as in Impagliazzo et al. [35]. The class Linear Monadic NP consists of all search problems expressible in the aforementioned logic fragment. All natural problems described therein [35], including \( k \)-CNF-Sat, Independent Set, Vertex Cover, and \( c \)-Colorability, which are complete for SNP under serf-reducibility with respect to their natural universe size (number of variables/vertices), are also complete for Linear Monadic NP under serf-reducibility. In fact, each problem in the class of problems defined by Impagliazzo et al. [35] consisting of all problems expressible in the SNP logic with a linear complexity parameter (as defined therein [35]) is in Linear Monadic NP. We prove that problems such as Linear Circuit-Sat, Dominating Set, and Independent Dominating set are in Linear Monadic NP (and are actually complete for Linear Monadic NP), but are not expressible in the SNP logic, thus showing that the set of problems expressible in the SNP logic with a linear complexity parameter is a proper subset of Linear Monadic NP. This implies that the subexponential-time solvability of any of the aforementioned problems is equivalent to the failure of ETH. Our inexpressibility proofs use the framework of logic games, namely, the Ajtai-Fagin game and the Ehrenfeucht-Fraïssé game.
Finally, we show using logic games that Feedback Vertex Set is inexpressible in existential monadic second-order logic, and hence is not in Linear Monadic NP. Whereas it can be easily shown that 3-CNF-Sat is serf-reducible to Feedback Vertex Set (by composing the two folklore polynomial-time reductions, which are also serf-reductions, from 3-CNF-Sat to Vertex Cover and from Vertex Cover to Feedback Vertex Set), it is open whether a serf-reduction exists in the other direction. We show that there is a polynomial-time reduction from Feedback Vertex Set to 3-CNF-Sat with a quasi-linear increase in the universe size and define a variant of Feedback Vertex Set that is equivalent to 3-CNF-Sat under serf-reducibility.

One can, in principle, also extend our approach of considering expressions with a linear value of our measure to SNP—rather than to Monadic SNP. To make this work, one would need to consider restrictions to the size of the domains over which the non-unary second-order variables range, as otherwise one would need a superlinear number of variables to express the problem with a Boolean circuit (for details, see Section 3.1). We are not aware of any results proving the existence of a natural problem that is in SNP, yet not in Monadic SNP. We leave this as a direction for future research.

We mention that SETH—alluded to earlier in this section—is a stronger related exponential-time hardness hypothesis than ETH. Assuming SETH, several sub-quadratic (polynomial) time lower-bound results have been recently shown for well-known problems, such as the Longest Common Subsequence problem and the problem of computing the Fréchet distance [1, 5–7]. There has also been some work by Dantsin and Wolpert on extending SNP to a class of problems such that the existence of a moderately exponential-time algorithm for CNF-Sat implies the same for all members of this class. This extension of SNP is based on allowing existential first-order quantifiers after universal first-order quantifiers in the second-order logic formulation of SNP (see Section 2.2), and then closing the resulting class under polynomial-time reductions. The work of Dantsin and Wolpert differs from the work in this article in that they consider a relaxation of the second-order logic formulas that define SNP, whereas we consider a restriction on this class of second-order logic formulas.

Roadmap. We begin in Section 2 by briefly reviewing some notions and terminologies from Boolean satisfiability, subexponential-time complexity, and graph theory. In Section 3, we define the class Linear Monadic NP, and we show that LINEAR CIRCUIT-Sat is complete for this class. We demonstrate how this class can be used to show that several natural optimization problems are serf-reducible to 3-CNF-Sat. We also show in this section that there are natural problems in Linear Monadic NP that are inexpressible in Monadic SNP. In Section 4, we show that Feedback Vertex Set is not in Linear Monadic NP (by showing that it is inexpressible in EMSO), and we investigate the existence of certain reductions between Feedback Vertex Set (and variants of it) and 3-CNF-Sat. We conclude in Section 5. Finally, in Appendix A, we provide a list of well-known problems that are serf-equivalent to 3-CNF-Sat.

2 PRELIMINARIES

2.1 Satisfiability

A propositional formula $F$ over a set $\{x_1, \ldots, x_n\}$ of variables is in conjunctive normal form (CNF) if it is the conjunction of a set of clauses $\{C_1, \ldots, C_m\}$, where each clause $C_i$, $i = 1, \ldots, m$, is the disjunction of literals (i.e., variables or negations of variables). We say that a propositional formula $F$ is satisfiable if there exists a truth assignment $\tau$ to the variables in $F$ that assigns at least one literal in each clause of $F$ the value 1 (true); we also say in this case that $\tau$ satisfies $F$. The CNF-Satisfiability problem, shortly CNF-Sat, is given a formula $F$ in CNF and decides whether or not $F$ is satisfiable. The width of a clause in a CNF formula $F$ is the number of literals in the clause. The
k-CNF-SAT problem, where \( k \geq 2 \), is the restriction of the CNF-SAT problem to instances in which the width of each clause is at most \( k \). It is well known that the \( k \)-CNF-SAT problem for \( k \geq 3 \) is \( \text{NP} \)-complete [32]. \( \text{LINEAR CNF-SAT} \) is the restriction of CNF-SAT to formulas with a linear number of clauses with respect to the number of variables.

A circuit is a directed acyclic graph. The vertices of indegree 0 are called the variables or input gates and are labeled either by positive literals \( x_i \) or by negative literals \( \overline{x}_i \). The vertices of indegree larger than 0 are called the gates and are labeled with Boolean operators \( \land \) or \( \lor \). A special gate of outdegree 0 is designated as the output gate. We do not allow negation gates in the circuit, since, by De Morgan’s laws, a general circuit can be efficiently converted into the above circuit model. The size of a circuit \( C \), denoted \( |C| \), is the number of gates in it. A circuit \( C \) is satisfiable if there is a truth assignment to the input variables of \( C \) that makes \( C \) evaluate to 1. The \text{Circuit-Satisfiability} problem, shortly \text{Circuit-Sat}, is given a circuit \( C \) and decides whether or not \( C \) is satisfiable. \text{LINEAR CIRCUIT-SAT} is the restriction of \text{CIRCUIT-SAT} to circuits of linear size with respect to the number of variables.

### 2.2 Subexponential Time and SNP

For any \( n \in \mathbb{N} \), by \([n]\), we denote the set \{1, 2, \ldots, \( n \)\}. The time complexity functions used in this article are assumed to be proper complexity functions, where by a proper complexity function \( f : \mathbb{N} \rightarrow \mathbb{N} \), we mean a nondecreasing function that, on input of length \( n \), is computable in time \( O(n + f(n)) \) and space \( O(f(n)) \). For any two proper complexity functions \( f, g : \mathbb{N} \rightarrow \mathbb{N} \), by writing \( f(n) = o(g(n)) \), we mean that there exists a proper complexity function \( \mu(n) : \mathbb{N} \rightarrow \mathbb{N} \), and \( n_0 \in \mathbb{N} \), such that \( f(n) \leq \mu(n)/\mu(n_0) \) for all \( n \geq n_0 \).

It is clear that \text{CNF-SAT} is solvable in time \( 2^n|F|^{O(1)} \), where \( F \) is the input instance and \( n \) is the number of variables in \( F \). We say that \text{CNF-SAT} is solvable in subexponential time if there exists an algorithm that solves the problem in time \( 2^{o(n)}|F|^{O(1)} \). Using the results of Chen et al. [13] and Flum and Grohe [29], the above definition is equivalent to the following: \text{CNF-SAT} is solvable in subexponential time if there exists an algorithm that for all \( \epsilon = 1/\ell \), where \( \ell \) is a positive integer, solves the problem in time \( 2^{\epsilon n}|I|^{O(1)} \).

Let \( Q \) and \( Q' \) be two problems, and let \( \mu \) and \( \mu' \) be two parameter functions defined on instances of \( Q \) and \( Q' \), respectively. In the case of \text{CNF-SAT}, \( \mu \) and \( \mu' \) will be the number of variables in the instances of these problems. A subexponential-time Turing reduction family [29, 35], shortly a serf-reduction,\(^3\) is an algorithm \( A \) with an oracle to \( Q' \) such that there are computable functions \( f, g : \mathbb{N} \rightarrow \mathbb{N} \) satisfying: (1) given a pair \((I, \epsilon)\) where \( I \in Q \) and \( \epsilon = 1/\ell \) (\( \ell \) is a positive integer), \( A \) decides \( I \) in time \( f(1/\epsilon)2^{\epsilon\mu(I)}|I|^{O(1)} \); and (2) for all oracle queries of the form “\( I' \in Q' \)” posed by \( A \) on input \((I, \epsilon)\), we have \( \mu'(I') \leq g(1/\epsilon)(\mu(I) + \log |I|) \).

The class \text{SNP} consists of all search problems expressible by second-order existential formulas whose first-order part is universal [44]; that is, search problems expressible by second-order formulas of the form \( \exists R_1 \ldots \exists R_q \forall z_1 \ldots \forall z_r \ \Phi(S, R_1, \ldots, R_q, z_1, \ldots, z_r) \), where \( S \) is the input structure, \( R_1, \ldots, R_q \) are (bound) relations, \( z_1, \ldots, z_r \) are first-order variables, and \( \Phi \) is a quantifier-free Boolean formula. We will refer to the preceding logic fragment as the \text{SNP logic}. Impagliazzo et al. [35] defined a complexity parameter for each such expression equal to \( \sum_{i=1}^q |R_i|^{\alpha_i} \), where \( |R_i| \) and \( \alpha_i, i = 1, \ldots, q \), are the size of the domain of \( R_i \) and the arity of \( R_i \), respectively.

\(^2\)As complexity parameter for the problems \text{CIRCUIT-SAT} and \text{LINEAR CIRCUIT-SAT}, we take the number of variables of the circuit. For the case of \text{LINEAR CIRCUIT-SAT}, one can equivalently take the number of gates of the circuit, as this is required to be linear in the number of variables.

\(^3\)Serf-reductions were introduced by Impagliazzo et al. [35]. Here, we use the definition given by Flum and Grohe [29]. The latter of the two definitions is as general as possible while still yielding that subexponential-time computability is closed under serf-reductions, and is more flexible for our purposes.
monadic relations, the number of bits needed to describe $R_i$ is the size of the universe on which $R_i$ is interpreted, and hence [35] the complexity parameter in this case is linear in the universe size. Impagliazzo et al. [35] formulated the Exponential Time Hypothesis (ETH) stating that $k$-CNF-SAT (for any $k \geq 3$) cannot be solved in subexponential time $2^{o(n)}$, where $n$ is the number of variables in the input formula. Therefore, there exists $c > 0$ such that $k$-CNF-SAT cannot be solved in time $O(2^{cn})$. ETH is equivalent to the statement that not all SNP problems are solvable in subexponential time. A fragment of SNP logic that is of special interest to us in this article—as will become clear in Section 3—is the following:

**Definition 2.1.** Define monadic SNP logic to be the restriction of SNP logic to expressions in which the second-order relations are monadic.

Lemma 2.2 below follows from the standard technique of renaming variables and the proof of the Sparsification Lemma [35]; see also the book by Flum and Grohe [29].

**Lemma 2.2.** $k$-CNF-SAT ($k \geq 3$) is solvable in $2^{o(n)}$ time if and only if $k$-CNF-SAT with a linear number of clauses and in which the number of occurrences of each variable is at most 3 is solvable in time $2^{o(n)}$, where $n$ is the number of variables in the formula (note that the size of an instance of $k$-CNF-SAT is polynomial in $n$). In particular, choosing $k = 3$, we get: 3-CNF-SAT in which every variable occurs at most 3 times, denoted 3-3-Sat, is not solvable in $2^{o(n)}$ time unless ETH fails.

### 2.3 Graphs

For a graph $G$, $V(G)$ and $E(G)$ are the sets of vertices and edges of $G$; $n(G) = |V(G)|$ and $e(G) = |E(G)|$ are the number of vertices and edges in $G$, respectively. A graph with one vertex is trivial. For a vertex $v$, we let $N(v)$ be the set of vertices adjacent to $v$, and $N[v] = N(v) \cup \{v\}$. The degree of a vertex $v$, $\deg(v)$, is the number of edges incident to $v$ in $G$; $\deg_H(v)$ is the degree of $v$ in a subgraph $H$ of $G$. For any integer $\ell \geq 3$, an $\ell$-cycle is a cycle of length $\ell$.

## 3 LINEAR MONADIC NP AND SERF-REDUCIBILITY TO LINEAR CIRCUIT-SAT

In this section, we present a fragment of existential second-order logic that extends the SNP logic introduced by Impagliazzo et al. [35] in the sense that it captures a larger set of problems that are serf-reducible to 3-CNF-Sat. The logic fragment we propose is a restriction of the well-studied existential monadic second-order logic (EMSO) that defines the complexity class Monadic NP (modulo standard syntactic augmentations to allow the expression of optimization problems)—for more details, see, e.g., References [14, 15, 34]. To define this logic fragment, we introduce a measure/function that associates with each expression in EMSO a value in terms of the complexity parameter of the expression, and we define Linear Monadic NP to be the restriction of Monadic NP to those expressions whose measure is linear in the complexity parameter. The complexity parameter of the expression we use is the same parameter defined by Impagliazzo et al. [35] for expressions in SNP.

**Definition 3.1 (Complexity Parameter).** If $R_1, \ldots, R_q$ are the existentially quantified second-order relations in an EMSO expression, where $R_i$ has arity $\alpha_i = 1$, for $i = 1, \ldots, q$, then the complexity parameter of the expression is $\sum_{i=1}^{q} |R_i|^{\alpha_i} = \sum_{i=1}^{q} |R_i|$. We note that the SNP logic does not restrict the existentially quantified relations to be monadic, and we could have opted to do the same in this article (i.e., not restrict ourselves to monadic relations and Monadic NP), and the results in this article would not have been affected. However, observe that the aforementioned complexity parameter defined by Impagliazzo et al. [35] and also used by us, when interpreted on a given structure (graph, CNF-formula, etc.) yields a parameter.
that is equal to $\sum_{i=1}^{q} |U|^{\alpha_i}$, where $U$ is the natural universe on which the relations $R_1, \ldots, R_q$ are interpreted. Hence, this parameter can only be linear in the universe size (number of vertices in a graph, number of variables in a formula, etc.) if all relations in the expression are monadic; otherwise, the complexity parameter will be at least quadratic. To be able to use serf-reductions to claim that a subexponential-time algorithm for 3-CNF-Sat yields a subexponential-time algorithm for the search problem under consideration with respect to a complexity parameter that is linear in the universe size of the problem, the existentially quantified relations in the SNP logic expression of the problem need to be monadic. Therefore, the fragment of interest in the framework of Impagliazzo et al. [35], when studying the equivalence between natural problems modulo serf-reducibility, is monadic SNP logic. We refer to the corresponding complexity class as Monadic SNP. By the same token, and without loss of generality, in this article, we restrict ourselves to Monadic NP.

3.1 Linear Monadic NP

The fragment of existential second-order logic that we consider is based on an inductively defined measure. Intuitively, this measure captures the size of the Boolean circuit that can be constructed for a concrete instance of the problem that is satisfiable if and only if there is an assignment to the second-order variables that makes the first-order part of the formula true.

We consider EMSO formulas with a single free monadic second-order variable to be able to express optimization problems where solutions of a given (exact) size are sought.

Definition 3.2. Let $\varphi(S) = \exists R_1 \ldots \exists R_q \Phi(S, R_1, \ldots, R_q)$ be an existential second-order logic formula, where $S$ is a free monadic second-order variable, where $R_1, \ldots, R_q$ are (bound) monadic second-order variables, and where $\Phi$ is a first-order logic formula that contains no free first-order variables and that contains no second-order variables besides $S, R_1, \ldots, R_q$. Moreover, let $A$ be an optimization problem consisting of inputs $(I, N)$, where $I$ is a relational structure over the same relational vocabulary as $\Phi$ and where $N$ is a positive integer. We say that $\varphi(S)$ expresses the problem $A$ if for each input $(I, N)$ it holds that $(I, N) \in A$ if and only if there is an interpretation $S_0$ of $S$ in $I$ of size exactly $N$ such that $I \models \varphi(S_0)$.

To keep the presentation of the technical machinery in this section readable, we assume that the formulas are in negation normal form, that is, that negations only occur directly before atoms. Because we can efficiently transform every formula to negation normal form (by De Morgan’s laws), we do not lose generality by making this assumption. For search problems that do not involve an optimization component (e.g., 3-CNF-Sat), we can simply omit the free variable $S$ in the logic expression and the integer $N$ in the problem input. All definitions and results extend straightforwardly to this setting.

For optimization problems where solutions of a given minimum or maximum size are sought, it suffices to investigate the variant of the problem involving solutions of exactly the given size. Since the number of possible choices for the size bound $m$ is linear in the size of the domain of the relational structure $I$, one can easily give a serf-reduction from, say, the problem where solutions of size at most $m$ are sought to the problem where solutions of size exactly $m$ are sought (serf-reductions are Turing reductions, and thus one can query the latter problem for each value $N' \leq N$).

That being said, all definitions and results in this section can straightforwardly be adapted to work also for optimization problems where solutions of a given minimum or maximum size are sought.

Our size measure $s$ is based on inductively defined measures, $s_{\forall}$ and $s_{\exists}$, that we define below. All measures take as input a second-order logic formula and return an arithmetic expression over the single variable $n$. For any formula $\varphi(S)$ that expresses a search problem $A$, intuitively, the
expression \( s(\varphi(S)) = f(n) \) denotes an upper bound on the size of the Boolean circuit that expresses a given instance in terms of the input size \( n \).

We inductively define these measures as follows: Here, we let \( Q \) range over \( \{\forall, \exists\} \), and we let \( \overline{Q} \) denote the unique quantifier in \( \{\forall, \exists\}\backslash\{Q\} \). Moreover, we let \( \circ \) range over \( \{\land, \lor\} \), and we let \( a \) denote any atom.

\[
\begin{align*}
  s(\exists R_i \psi) &= s(\psi); & (1) \\
  s(Q z_i \psi) &= \begin{cases} 
  n & \text{if } \psi \text{ does not contain any occurrence of } \overline{Q}; \\
  1 + s_Q(Q z_i \psi) & \text{otherwise}; 
  \end{cases} & (3) \\
  s_Q(Q z_i \psi) &= 1 + n \cdot s_Q(\psi); & (4) \\
  s_Q(\psi_1 \circ \psi_2) &= 1 + s_Q(\psi_1) + s_Q(\psi_2); & (5) \\
  s_Q(\exists \psi) &= s_Q(\psi); & (6) \\
  s_Q(\forall \psi) &= s_Q(\psi); & (7) \\
  s_Q(\psi_1 \circ \psi_2) &= s_Q(\psi_1) \quad \text{if } \psi_1 \text{ contains no occurrences of second-order variables}; \\
  s_Q(\psi_1 \circ \psi_2) &= s_Q(\psi_2) \quad \text{if } \psi_2 \text{ contains no occurrences of second-order variables}; \quad (8)
\end{align*}
\]

for the case where both \( \psi_1 \) and \( \psi_2 \) contain second-order variables, we let:

\[
\begin{align*}
  s_3(\psi_1 \lor \psi_2) &= s_3(\psi_1) + s_3(\psi_2); & (9) \\
  s_4(\psi_1 \land \psi_2) &= s_4(\psi_1) + s_4(\psi_2); & (10) \\
  s_3(\psi_1 \lor \psi_2) &= 1 + s_3(\psi_1) + s_3(\psi_2); & (11) \\
  s_4(\psi_1 \land \psi_2) &= 1 + s_4(\psi_1) + s_4(\psi_2). & (12)
\end{align*}
\]

The intuition behind the definition of the measure is that \( s(\varphi(S)) \) measures the size of the Boolean circuit \( C \) that is needed to express the formula \( \varphi(S) \) when interpreted over the input \( I \); \( s(\varphi(S)) \) returns an arithmetic expression with a single symbol, \( n \), that is interpreted as the size of the universe \( U \) of the relational structure \( I \). The variables of \( C \) are of the form \( x_T,e \), representing whether an element \( e \) in the domain of \( I \) is chosen to be part of the interpretation of the monadic second-order variable \( T \). The circuit \( C \) then encodes the first-order part of \( \varphi \).

The computation of \( s(\varphi) \) works as follows: The existential second-order quantifiers are disregarded, and the outermost Boolean connectives are dealt with in a straightforward inductive manner. For every maximal subformula \( \psi \) that starts with a first-order quantifier \( Q \), there are two options. Either \( \psi \) contains only first-order quantifiers of a single type—in which case the measure \( s \) returns \( n \) for this subformula—or \( \psi \) contains quantifiers of both types. In the latter case, the size of the (sub)circuit to represent \( \psi \) is measured inductively using \( s_Q \). The measure \( s_Q \) keeps track of what logic gate is the parent of the subcircuit representing the subformula \( \psi \) (i.e., \( Q = \forall \) corresponds to an \( \land \)-gate as parent and \( Q = \exists \) corresponds to a \( \lor \)-gate as parent) and increases the size only if the subcircuit cannot be integrated/merged with its parent gate. For example, if the output gate of the subcircuit is an \( \land \)-gate, and its parent is also an \( \land \)-gate, these two gates can be merged into a single large \( \land \)-gate.

**Definition 3.3.** We define **Linear Monadic NP** to be the class of NP search problems that are expressible using an EMSO formula \( \varphi(S) \) with \( s(\varphi(S)) = O(n) \).

The following theorem shows that **LINEAR CIRCUIT-SAT** can serve as the canonical satisfiability problem for Linear Monadic NP:

**Theorem 3.4.** Let \( A \) be a problem in Linear Monadic NP, and suppose that \( A \) is expressible using an EMSO formula \( \varphi(S) \), where \( s(\varphi(S)) = O(n) \). Then \( A \) is serf-reducible to **LINEAR CIRCUIT-SAT**, where
the parameter of $A$ is the size of the universe of the input structure, and the parameter of Linear Circuit-Sat is the number of variables of the circuit.

**Proof.** Let $A$ be a search problem with input of the form $(I, N)$, where $I$ is a relational structure and $N \in \mathbb{N}$, such that $A$ is expressed by the EMSO formula $\varphi(S) = \exists R_1, \ldots, R_q \ \Phi(R_1, \ldots, R_q, S)$, where $\Phi(R_1, \ldots, R_q, S)$ is a first-order formula in negation normal form and $s(\varphi(S)) = O(n)$. Since $s(\varphi(S)) = O(n)$, we know that $\Phi$ consists of first-order logic formulas $\psi_1, \ldots, \psi_k$—each starting with a first-order quantifier—that are combined (in an arbitrary way) using the Boolean connectives $\land$ and $\lor$. Moreover, by Equations (1)–(3), we know that $s(\psi_i) = O(n)$ for each $i \in [k]$. We show that for any instance $(I, N)$ of $A$, we can decide whether $(I, N) \in A$ in subexponential time by using an oracle that decides the satisfiability of linear-size circuits. Let $U$ be the universe of the relational structure $I$. The input gates of each circuit $C$ are the Boolean variables $x_{T,u}$, for $u \in U$, and their negations, where for each $T \in \{S, R_1, \ldots, R_q\}$, the variables $x_{T,u}$ encode the interpretation $T_0 \subseteq U$ given to the second-order variable $T$. That is, $x_{T,u} = 1$ if and only if $u \in T_0$. Note that the number of variables $x_{T,u}$ is linear in $|U|$.

Recall that $n$ is a variable symbol in $s(\varphi(S))$ that, intuitively, represents the size of the universe $U$ of the relational structure $I$. The fact that $s(\varphi(S)) = O(n)$—and thus that $s(\psi_i) = O(n)$ for each $\psi_i$—will then allow us to show that all circuits $C$ that we construct in the remainder of this proof are of size linear in $|U|$.

We first construct an auxiliary circuit $C_{size}$. The circuit $C_{size}$ ensures that exactly $N$ variables $x_{S,u}$ are set to true in each satisfying assignment. This can be done with a circuit of linear size, and even with a linear number of 3-CNF-Sat clauses [48]. Further, we will show how for each $\psi_i, i \in [k]$, we can construct a sequence of circuits $C_{i,1}, \ldots, C_{i,b_i}$ such that:

- (a) $b_i \leq 2^{O(n)}$;
- (b) for each $\ell \in [b_i], C_{i,\ell}$ is of size $O(|U|)$; and
- (c) for each assignment $\alpha: \{S, R_1, \ldots, R_q\} \to 2^U$, it holds that $(I, \alpha) \models \psi_i$ if and only if there is some $\ell \in [b_i]$ such that $C_{i,\ell}$ is satisfied by the truth assignment to the variables $x_{T,u}$ that corresponds to $\alpha$.

Once we have constructed such a family of circuits $C_{i,\ell}$ for each formula $\psi_i$, we can decide whether $(I, N) \in A$ as follows. We iterate over all ways of satisfying $\Phi$ by satisfying the formulas $\psi_i$—that is, all sets of formulas $F \subseteq \{\psi_1, \ldots, \psi_k\}$ that have the property that if all formulas in $F$ are satisfied, then $\Phi$ is satisfied. Let $\mathcal{F}$ denote the set of all sets $F \subseteq \{\psi_1, \ldots, \psi_k\}$ that have this property. For each set $F \in \mathcal{F}$, we know that all $\psi_i \in F$ are simultaneously satisfied if there exists some $\alpha: \{S, R_1, \ldots, R_q\} \to 2^U$ and some $\ell_i \in [b_i]$ for each $\psi_i \in F$ such that $(I, \alpha) \models \bigwedge_{\psi_i \in F} C_{i,\ell_i}$. Thus, for each set $F \in \mathcal{F}$, we iterate over all possible choices for $C_{i,\ell_i}$, for each $\psi_i \in F$, and we check whether $C_{size} \land \bigwedge_{\psi_i \in F} C_{i,\ell_i}$ is satisfiable by querying the oracle. Clearly, each such conjunction can be expressed by a linear size circuit, because both $C_{size}$ and all $C_{i,\ell_i}$ are circuits of linear size. Now $(I, N) \in A$ if and only if there exists an $F \in \mathcal{F}$ such that $C_{size} \land \bigwedge_{\psi_i \in F} C_{i,\ell_i}$ is satisfiable. Since $k$ is constant and $b_i \leq 2^{O(n)}$ for each $i \in [k]$, there are at most $2^k \cdot (2^{O(n)})^k = 2^{o(n)}$ many different combinations we have to iterate over. Therefore, the above reduction is a serf-reduction to Linear Circuit-Sat.

All that remains is to show that, for each $\psi_i$, we can construct a family of circuits $C_{i,1}, \ldots, C_{i,b_i}$ satisfying properties (a)–(c) above. Take an arbitrary $\psi_i$. Because $s(\Phi) = O(n)$, we have $s(\psi_i) = O(n)$ as well. We distinguish several cases:

- (i) $\psi_i$ contains only universal first-order quantifiers;
- (ii) $\psi_i$ contains only existential first-order quantifiers; or
- (iii) $\psi_i$ contains both universal and existential first-order quantifiers.
We will treat these cases in the listed order. For cases (i) and (ii), the construction of the circuits $C_{i,1}, \ldots, C_{i,b_i}$ can be explained concisely. For case (iii), the construction requires a lengthier explanation that involves another case distinction.

Case (i) We can rewrite $\psi_i$ into prenex form without introducing any existential first-order quantifiers, because $\psi_i$ is in negation normal form. Then, by a known result [35, Theorem 2], we can express $\psi_i$ (when interpreted over $I$) as an $r$-CNF formula $\chi_i$ (possibly of super-linear size), for some constant $r$. By the Sparsification Lemma [35] [29, Lemma 16.17], we can transform $\chi_i$ into a suitable sequence $C_{i,1}, \ldots, C_{i,b_i}$ of linear-size circuits, satisfying properties (a)–(c).

Case (ii) We can use a known result [35, Theorem 2] to express $\psi_i$ (when interpreted over $I$) as an $r$-DNF formula $\chi_i$. This DNF formula $\chi_i$ directly gives rise to a suitable sequence $C_{i,1}, \ldots, C_{i,b_i}$ of linear-size circuits, satisfying properties (a)–(c). Each such circuit $C_{i,\ell}$ expresses one of the terms of $\chi_i$.

Case (iii) Let $Q$ denote the first quantifier appearing in $\psi_i$. We know that $s(\psi_i) = s_Q(\psi_i) = O(n)$. We construct a single linear-size circuit $C_i$ such that for each assignment $\alpha: \{S, R_1, \ldots, R_q\} \rightarrow 2^U$ it holds that $(I, \alpha) \models \psi_i$ if and only if $C_i$ is satisfied by the truth assignment to the variables $x_{i,u}$ that correspond to $\alpha$ (that is, we let $b_1 = 1$). Intuitively, we do so by unfolding $\psi_i$ using the input structure $I$—translating atoms containing second-order variables into input variables for the circuit. Concretely, for each subformula $\chi$ of $\psi_i$ with free first-order variables $x_1, \ldots, x_b$, and for each instantiation $\beta: \{x_1, \ldots, x_b\} \rightarrow U$ for the free first-order variables of $\chi$, we construct a circuit $C_{X,\beta}$. We construct these circuits to define $C_i = C_{\psi_i,0}$. The construction of the circuits $C_{X,\beta}$ is recursive—that is, the construction uses circuits $C_{X',\beta'}$, where $X'$ is a subformula of $\chi$ and $\beta'$ is an instantiation of the free first-order variables of $X'$. Moreover, we ensure that each such circuit $C_{X,\beta}$ is either: a Boolean constant ($\top$ or $\bot$), a literal, or an $\land$-gate or an $\lor$-gate as output gate. Additionally, we ensure that for each $\chi$, and for every $\beta, \beta': \text{Free}(\chi) \rightarrow U$, if both $C_{X,\beta}$ and $C_{X',\beta'}$ are not a Boolean constant or a literal, then they have the same output-gate type (i.e., either both have an $\land$-gate, or both have an $\lor$-gate, as output gates). We inductively prove that this construction yields a circuit $C_{\psi_i,0}$ that is of linear size, due to the fact that $s(\psi_i) = O(n)$ and to Equations (1)–(12).

Let $\chi$ be an arbitrary subformula of $\psi_i$ and let $\beta: \text{Free}(\chi) \rightarrow U$ be an arbitrary instantiation of the free first-order variables of $\chi$. We construct the circuit $C_{X,\beta}$ as follows: We distinguish several cases:

(A) $\chi = \forall z \chi'$;
(B) $\chi = \exists z \chi'$;
(C) $\chi = x_1 \land x_2$;
(D) $\chi = x_1 \lor x_2$; or
(E) $\chi$ is a literal.

We treat the cases (A)–(E) in the listed order. For each of these cases, we distinguish further subcases.

Case (A). We know that $\chi = \forall z \chi'$. We base the construction of $C_{X,\beta}$ on the circuits $C_{X',\beta_u}$ for $u \in U$, where $\beta_u = \beta \cup \{z \mapsto u\}$. If (A.1) any of these circuits is the Boolean constant $\bot$, we let $C_{X,\beta} = \bot$. If (A.2) all of these circuits are the Boolean constant $\top$, we let $C_{X,\beta} = \top$. We distinguish two remaining subcases: Either (A.3) all circuits $C_{X',\beta_u}$ that are not Boolean constants or literals have an $\land$-gate as output gate, or (A.4) all circuits $C_{X',\beta_u}$ that are not Boolean constants or literals have an $\lor$-gate as output gate. (Due
to the fact that no two circuits $C_X,\beta$ and $C_{X',\beta'}$ have output gates of different types, this case distinction is exhaustive.) In both of these latter cases, we let the output gate of $C_X,\beta$ be an $\land$-gate. First, consider case (A.3). We let the children of the output $\land$-gate of $C_X,\beta$ be the collective set of children of the output gates of the circuits $C_{X',\beta_u}$ that have an output $\land$-gate, together with the circuits $C_{X',\beta_u}$ that are literals. That is, $C_X,\beta$ is then equivalent to the conjunction $\bigwedge_{u \in U} C_{X',\beta_u}$. Next, consider case (A.4). We let the children of the output $\land$-gate of $C_X,\beta$ be the output $\lor$-gates of the circuits $C_{X',\beta_u}$ that have an output $\lor$-gate, together with the circuits $C_{X',\beta_u}$ that are literals. Then clearly, also in this case, $C_X,\beta$ is equivalent to the conjunction $\bigwedge_{u \in U} C_{X',\beta_u}$.

Case (B). We know that $\chi = \exists z \chi'$. This case is entirely dual to case (A). Again, we base the construction of $C_X,\beta$ on the circuits $C_{X',\beta_u}$, where $\beta_u = \beta \cup \{z \mapsto \psi\}$. If (B.1) any of these circuits is the Boolean constant $\top$, we let $C_X,\beta = \top$. If (B.2) all of these circuits are the Boolean constant $\bot$, we let $C_X,\beta = \bot$. We distinguish two remaining subcases: Either (B.3) all circuits $C_{X',\beta_u}$ that are not Boolean constants or literals have an $\lor$-gate as output gate, or (B.4) all circuits $C_{X',\beta_u}$ that are not Boolean constants or literals have an $\land$-gate as output gate. In both of these latter cases, we let the output gate of $C_X,\beta$ be an $\land$-gate. First, consider case (B.3). We let the children of the output $\land$-gate of $C_X,\beta$ be the collective set of children of the output gates of the circuits $C_{X',\beta_u}$ that have an output $\land$-gate, together with the circuits $C_{X',\beta_u}$ that are literals. That is, $C_X,\beta$ is then equivalent to the disjunction $\bigvee_{u \in U} C_{X',\beta_u}$. Next, consider case (B.4). We let the children of the output $\land$-gate of $C_X,\beta$ be the output $\lor$-gates of the circuits $C_{X',\beta_u}$ that have an output $\land$-gate, together with the circuits $C_{X',\beta_u}$ that are literals. Then clearly, also in this case, $C_X,\beta$ is equivalent to the disjunction $\bigvee_{u \in U} C_{X',\beta_u}$.

Case (C). We know that $\chi = \chi_1 \land \chi_2$. We distinguish several subcases: (C.1) $\chi_1$ is a Boolean constant, (C.2) $\chi_2$ is a Boolean constant, or (C.3) neither $\chi_1$ nor $\chi_2$ is a Boolean constant. In case (C.1), if $\chi_1 = \top$, then we let $C_X,\beta = \top$, and if $\chi_1 = \bot$, then we let $C_X,\beta = C_{X_2,\beta}$. Similarly, in case (C.1), if $\chi_2 = \top$, then we let $C_X,\beta = \bot$, and if $\chi_2 = \bot$, then we let $C_X,\beta = C_{X_1,\beta}$. In case (C.3), we let the output gate of $C_X,\beta$ be an $\land$-gate, with its only two children the output gates of $C_{X_1,\beta}$ and $C_{X_2,\beta}$.

Case (D). This case is entirely analogous to case (C). We know that $\chi = \chi_1 \lor \chi_2$. We distinguish several subcases: (D.1) $\chi_1$ is a Boolean constant, (D.2) $\chi_2$ is a Boolean constant, or (D.3) neither $\chi_1$ nor $\chi_2$ is a Boolean constant. In case (D.1), if $\chi_1 = \top$, then we let $C_X,\beta = \top$, and if $\chi_1 = \bot$, then we let $C_X,\beta = C_{X_2,\beta}$. Similarly, in case (D.2), if $\chi_2 = \top$, then we let $C_X,\beta = \top$, and if $\chi_2 = \bot$, then we let $C_X,\beta = C_{X_1,\beta}$. In case (D.3), we let the output gate of $C_X,\beta$ be an $\lor$-gate, with as only two children the output gates of $C_{X_1,\beta}$ and $C_{X_2,\beta}$.

Case (E). We know that $\chi$ is a literal. Again, we distinguish two cases: Either (E.1) $\chi$ contains no second-order variable, or (E.2) $\chi$ contains some second-order variable. In case (E.1), we know whether or not $(I, \alpha, \beta) \models \chi$, regardless of the interpretation $\alpha$ of the second-order variables. We let $C_X,\beta = \top$ if $(I, \alpha, \beta) \models \chi$, and we let $C_X,\beta = \bot$ if $(I, \alpha, \beta) \not\models \chi$. In case (E.2), we know that either $\chi = T(z)$ or $\chi = \neg T(z)$, for some monadic second-order variable $T$ and some first-order variable $z$. If $\chi = T(z)$, then we let $C_X,\beta$ consist of the input variable $x_{T,u}$, where $u = \beta(z)$. Similarly, if $\chi = \neg T(z)$, then we let $C_X,\beta$ consist of the negation $\neg x_{T,u}$ of the input variable $x_{T,u}$.

After having described the construction of the circuits $C_X,\beta$, we will now argue that $C_{\psi_i,\theta}$ is of size linear in $|U|$. We do so by inductively showing that for each subformula $\chi$ of $\psi_i$ and for each instantiation $\beta : \text{Free}(\chi) \to U$ of the free first-order variables of $\chi$, it holds that (a) if $s_1(\chi)$ and $s_2(\chi)$ are constants, then $|C_X,\beta|$ is also a constant, and (b) if $s_1(\chi)$ and $s_2(\chi)$ are linear functions in $n$, then $|C_X,\beta| = \mathcal{O}(|U|)$. In the base case, where $\chi$ is a
literal, this statement holds trivially, since for any \( \beta \), the circuit \( C_{\chi, \beta} \) contains no internal gates, and \( s_{\forall}(\chi) = s_{\exists}(\chi) = 0 \) by Equation (5). In the inductive case where \( \chi = \chi_1 \circ \chi_2 \), for some \( \circ \in \{\land, \lor\} \), the statement follows directly from the induction hypothesis, the construction of \( C_{\chi, \beta} \), and Equations (7)–(12).

The only two remaining inductive cases are those where \( \chi = \forall z \chi' \) or \( \chi = \exists z \chi' \). We begin with the case where \( \chi = \forall z \chi' \), and we distinguish several subcases:

1. \( \chi' \) is a literal;
2. \( \chi' = \forall z' \chi'' \) or \( \chi' = \chi_1 \land \chi_2 \); or
3. \( \chi' = \exists z' \chi'' \) or \( \chi' = \chi_1 \lor \chi_2 \).

We treat these cases in the listed order.

Case (1). We know that \( C_{\chi, \beta} \) is either a conjunction of literals or a Boolean constant. Therefore, the size of \( C_{\chi, \beta} \) is at most 1, and thus the statement holds trivially.

Case (2). We know that for each \( \beta_u = \beta \cup \{z \mapsto u\} \), \( u \in U \), the circuit \( C_{\chi', \beta_u} \) is either a literal or a Boolean constant or has an \( \land \)-gate as output gate. Then, since by case (A.3) in the construction of \( C_{\chi, \beta} \), we know that \( C_{\chi, \beta} \) has an \( \land \)-gate as output gate, whose children are the children of the output gates of the circuits \( C_{\chi', \beta_u} \) that have an output \( \land \)-gate, together with the circuits \( C_{\chi', \beta_u} \) that are literals. First, to show (a), suppose that \( s_{\forall}(\chi) = O(1) \). By Equation (6), this can only be the case if \( s_{\forall}(\chi') = 0 \). Then, by the induction hypothesis, we know that each circuit \( C_{\chi', \beta_u} \) is either a literal or a Boolean constant. Then, \( |C_{\chi, \beta}| = O(1) \), and thus statement (a) holds. Next, to show (b), suppose that \( s_{\forall}(\chi) = O(n) \). By Equation (6), this can only be the case if \( s_{\forall}(\chi') = O(1) \). Then, by the induction hypothesis, we know that each circuit \( C_{\chi', \beta_u} \) is of constant size. Therefore, \( |C_{\chi, \beta}| = O(|U|) \), and thus statement (b) holds.

Case (3). We know that for each \( \beta_u = \beta \cup \{z \mapsto u\} \), \( u \in U \), the circuit \( C_{\chi', \beta_u} \) is either a literal or a Boolean constant, or has an \( \lor \)-gate as output gate. Then, since by case (A.4) in the construction of \( C_{\chi, \beta} \), we know that \( C_{\chi, \beta} \) has an \( \land \)-gate as output gate, whose children are the output gates of the circuits \( C_{\chi', \beta_u} \) that have an output \( \lor \)-gate, together with the circuits \( C_{\chi', \beta_u} \) that are literals. To show (a), suppose that \( s_{\forall}(\chi) = O(1) \). By Equation (4), this can only be the case if \( s_{\forall}(\chi') = 0 \). Then, by the induction hypothesis, we know that each circuit \( C_{\chi', \beta_u} \) is either a literal or a Boolean constant. Then, \( |C_{\chi, \beta}| = O(1) \), and thus statement (a) holds. Next, to show (b), suppose that \( s_{\forall}(\chi) = O(n) \). By Equation (4), this can only be the case if \( s_{\forall}(\chi') = O(1) \). Then, by the induction hypothesis, we know that each circuit \( C_{\chi', \beta_u} \) is of constant size. Therefore, \( |C_{\chi, \beta}| = O(|U|) \), and thus statement (b) holds.

The case where \( \chi = \exists z \chi' \) is entirely analogous (and dual) to the case where \( \chi = \forall z \chi' \). As the arguments for the case where \( \chi = \forall z \chi' \) can straightforwardly be adapted to the case where \( \chi = \exists z \chi' \), we omit a detailed treatment of this latter case. This completes our proof of the statements (a) and (b). Then, since \( C_i = C_{\psi_i, \emptyset} \) and by assumption we have that \( s(\psi_i) = O(n) \), by statement (b), we can conclude that \( |C_i| = O(|U|) \).

This concludes our treatment of the cases (i)–(iii), showing that for each \( \psi_i \), we can construct a family of circuits \( C_{i, 1}, \ldots, C_{i, b} \), satisfying properties (a)–(c), and thereby concludes our description of the serf-reduction to LINEAR CIRCUIT-SAT.

\[ \square \]

Definition 3.5. A problem \( A \) is Linear Monadic NP-complete under serf-reducibility if (1) it is in Linear Monadic NP and (2) every problem in Linear Monadic NP is serf-reducible to \( A \).
**Corollary 3.6.** **Linear Circuit-Sat is Linear Monadic NP-complete under serf-reducibility.**

**Proof.** It is well-known (see, e.g., the handbook of Flum and Grohe [29]) that circuit satisfiability can be expressed by the following EMSO formula $\varphi_{\text{Circ-SAT}}(S)$, where the structure representing the input has the set of gates in the circuit (including the input gates) as its universe:

\[
\varphi_{\text{Circ-SAT}}(S) = \forall y \ (\text{AND}(y) \rightarrow (S(y) \leftrightarrow \forall z(E(z, y) \rightarrow S(z)))) \\
\quad \land (\text{OR}(y) \rightarrow (S(y) \leftrightarrow \exists z(E(z, y) \land S(z)))) \\
\quad \land (\text{NEG}(y) \rightarrow (S(y) \leftrightarrow \exists z(E(z, y) \land \neg S(z)))) \\
\quad \land (\text{OUT}(y) \rightarrow S(y))).
\]

Here the unary predicates AND, OR, and NEG encode the types of the gates in the circuit, the unary predicate OUT encodes which gate is the output gate, and the binary predicate E encodes the set of (directed edges) between the gates in the circuit.

It is not difficult to verify based on Equations (1)–(12) that $s(\varphi_{\text{Circ-SAT}}(S)) = 9n + 1.$\(^4\) We use the following abbreviations:

\[
\begin{align*}
\varphi_{\text{AND}} &= (\text{AND}(y) \rightarrow (S(y) \leftrightarrow \forall z(E(z, y) \rightarrow S(z)))), \\
\varphi_{\text{OR}} &= (\text{OR}(y) \rightarrow (S(y) \leftrightarrow \exists z(E(z, y) \land S(z))))), \\
\varphi_{\text{NEG}} &= (\text{NEG}(y) \rightarrow (S(y) \leftrightarrow \exists z(E(z, y) \land \neg S(z)))), \text{ and} \\
\varphi_{\text{OUT}} &= (\text{OUT}(y) \rightarrow S(y))).
\end{align*}
\]

We have that:

\[
s(\varphi_{\text{Circ-SAT}}(S)) = 1 + s_4(\forall y \varphi_{\text{AND}} \land \varphi_{\text{OR}} \land \varphi_{\text{NEG}} \land \varphi_{\text{OUT}}) = 1 + n \cdot s_4(\varphi_{\text{AND}} \land \varphi_{\text{OR}} \land \varphi_{\text{NEG}} \land \varphi_{\text{OUT}}) = 1 + n \cdot (s_4(\varphi_{\text{AND}}) + s_4(\varphi_{\text{OR}}) + s_4(\varphi_{\text{NEG}}) + s_4(\varphi_{\text{OUT}})).
\]

We also have that:

\[
s_4(\varphi_{\text{AND}}) = s_4(\text{AND}(y) \rightarrow (S(y) \leftrightarrow \forall z(E(z, y) \rightarrow S(z)))) = s_4(S(y) \leftrightarrow \forall z(E(z, y) \rightarrow S(z))) = s_4(\forall z(E(z, y) \rightarrow S(y))) \land (\forall z(E(z, y) \rightarrow S(z)) \rightarrow S(y)) = s_4(\neg S(y) \lor \forall z(E(z, y) \rightarrow S(z))) + s_4(\neg \forall z(E(z, y) \rightarrow S(z))) = s_4(\forall z(E(z, y) \rightarrow S(z))) + s_4(\exists z(E(z, y) \land \neg S(z))) = 2 + s_3(\neg S(y)) + s_3(\exists z(E(z, y) \land \neg S(z))) + s_3(\exists z(E(z, y) \land \neg S(z))) + s_3(S(y)) = 3 + 0 + n \cdot s_4(\neg E(z, y) \lor S(z)) + n \cdot s_3(E(z, y) \land \neg S(z)) + 0 = 3 + n \cdot S(z) + n \cdot s_3(\neg S(z)) = 3 + n \cdot 1 = 3.
\]

Moreover, by entirely similar derivations, we have that $s_4(\varphi_{\text{OR}}) = s_4(\varphi_{\text{NEG}}) = 3.$ Finally, we have:

\[
s_4(\varphi_{\text{OUT}}) = s_4(\text{OUT}(y) \rightarrow S(y))) = s_4(S(y)) = 0.
\]

Therefore, we have that $s(\varphi_{\text{Circ-SAT}}(S)) = 1 + n \cdot (3 + 3 + 3) = 9n + 1.$

---

\(^4\)Here, we express $(a \leftrightarrow b)$ as $(a \rightarrow b) \land (b \rightarrow a)$ and $(a \rightarrow b)$ as $(\neg a \lor b)$. In general, the value of the measure $s$ can differ when one uses different (but logically equivalent) propositional logic expressions, but all of these values will be linearly related.
It follows that \textsc{Linear Circuit-Sat} is in the class Linear Monadic NP with respect to the size of the circuit, and hence with respect to the number of input variables in the circuit, as the complexity parameter. Completeness for Linear Monadic NP follows from Theorem 3.4. □

Note that all problems in Monadic SNP [35] are in Linear Monadic NP. This is because every EMSO expression (in negation normal form) that contains no first-order quantifier alternations has a linear measure—due to Equation (3). Because \textsc{Linear Circuit-Sat} is serf-reducible to 3-CNFSat [38], Corollary 3.6 implies the following:

**Corollary 3.7.** Every Monadic SNP-complete problem is Linear Monadic NP-complete under serf-reducibility. Therefore, a Linear Monadic NP-complete problem is solvable in subexponential time if and only if ETH fails.

The above implies that the well-known SNP-complete problems \(k\)-CNF-Sat \((k \geq 3)\), c-Colorability, Independent Set, Clique, and Vertex Cover, among others, are Linear Monadic NP-complete. Note that every problem that is Linear Monadic NP-complete under serf-reducibility is clearly hard for SNP under serf-reducibility. This follows from the fact that \(k\)-CNF-Sat \((k \geq 3)\) is SNP complete under serf-reducibility [35] and is reducible to \textsc{Linear Circuit-Sat} (see Lemma 2.2).

### 3.2 Applications: Expressing Natural Optimization Problems

We saw in the previous subsection that \textsc{Linear Circuit-Sat} is in Linear Monadic NP. We prove in Subsection 3.3 that \textsc{Linear Circuit-Sat} is not in monadic SNP. The same can be shown for problems that are serf-equivalent to \textsc{Linear Circuit-Sat}, such as \textsc{Linear Hitting Set} and \textsc{Linear Set Cover}. In this subsection, we give several examples of natural graph problems that are in Linear Monadic NP, but are inexpressible in monadic SNP logic, as will be shown in Section 3.3.

Our first example is \textsc{Dominating Set}, with the number of vertices as the complexity parameter: Given a graph \(G\) and \(N \in \mathbb{N}\), decide if \(G\) contains a set \(S\) of \(N\) vertices such that every vertex in \(V(G) \setminus S\) has a neighbor in \(S\). This problem can be expressed using the following formula \(\varphi_{DS}(S)\) given below; the structure representing the input contains the vertices of the graph as elements in its universe, and it contains a binary predicate \(E\) that captures the edge-set:

\[
\varphi_{DS}(S) = \forall x \; (S(x) \lor \exists y (S(y) \land E(x, y))).
\]

From Equations (1)–(12), we get that \(s(\varphi_{DS}(S)) = n + 1\). In effect:

\[
s(\varphi_{DS}(S)) = s[\forall x \; (S(x) \lor \exists y (S(y) \land E(x, y)))] = 1 + n \cdot s[\forall x \; (S(x) \lor \exists y (S(y) \land E(x, y)))]
\]

\[= 1 + n \cdot (1 + 0 + n \cdot s[\exists y (S(y) \land E(x, y))])
= 1 + n \cdot (1 + 0 + n \cdot s[S(y)])
= 1 + n \cdot (1 + 0 + n \cdot 0)
= n + 1.
\]

It follows that \textsc{Dominating Set} is in Linear Monadic NP. Our second example is the \textsc{Red-Blue Dominating Set} problem [19], with the number of vertices as the complexity parameter: Given a graph \(G\) in which \(V(G)\) is partitioned in a set \(R\) of red vertices and a set \(B\) of blue vertices, and \(N \in \mathbb{N}\), decide if there is a set of \(N\) red vertices that dominate all blue vertices. The expression is \(\varphi_{RB-DS}(S)\):

\[
\varphi_{RB-DS}(S) = \forall x \; (\neg S(x) \lor R(x)) \land \forall y \; (\neg B(y) \lor \exists z \; (S(z) \land E(z, y))),
\]

for which it can be easily verified that \(s(\varphi_{RB-DS}(S)) = 2n + 2\).
The **Non-Blocker** problem [23] is: Given a graph \( G \) and \( N \in \mathbb{N} \), decide if \( G \) contains a set of \( N \) vertices \( S \) such that every vertex in \( S \) has a neighbor not in \( S \). Similarly, we can express the problem **Non-Blocker**, with the number of vertices as complexity parameter, using the following formula \( \varphi_{NB}(S) \):

\[
\varphi_{NB}(S) = \forall x \ (\neg \exists y (S(x) \lor S(y) \land E(x,y))).
\]

It can be easily verified that \( s(\varphi_{NB}(S)) = n + 1 \), and hence **Non-Blocker** is in Linear Monadic NP.

The **Independent Dominating Set** problem [19] is defined as follows: Given a graph \( G \) and \( N \in \mathbb{N} \), decide if there is an independent set \( S \) of \( N \) vertices that dominates \( V(G) \). This problem is in Linear Monadic NP, because it can be expressed using the following formula \( \varphi_{IDS}(S) \) (a conjunction of the two expressions for **Dominating Set** and **Independent Set**), for which \( s(\varphi_{IDS}(S)) = 2n + 2 \):

\[
\varphi_{IDS}(S) = \varphi_{DS}(S) \land [\forall x \forall y (E(x,y) \lor \neg S(x) \lor \neg S(y))].
\]

Similarly to **Independent Dominating Set**, the **Dominating Clique** problem [16] asks, given a graph \( G \) and \( N \in \mathbb{N} \), to decide if there is a subset \( S \) of cardinality \( N \) that is a clique and that dominates \( V(G) \). The problem is in Linear Monadic NP, because it can be expressed using the formula \( \varphi_{DC}(S) \), for which it can be easily verified that \( s(\varphi_{DC}(S)) = 2n + 2 \):

\[
\varphi_{DC}(S) = [\forall x \ (S(x) \lor \exists y (S(y) \land E(x,y)))] \land [\forall x \forall y (E(x,y) \lor \neg S(x) \lor \neg S(y))].
\]

Further examples of problems that are in Linear Monadic NP are **r-Dominating Set** [33], **r-Threshold Dominating Set** [23], and **r-Domatic Partition** [46] for any integer-constant \( r \). The **r-Dominating Set** problem is to decide whether a given graph \( G \) contains a set \( S \) of \( N \) vertices such that each vertex in \( G \) is within distance \( r \) to some vertex \( s \in S \). The problem **r-Threshold Dominating Set** problem is to decide whether a given graph \( G \) contains a set \( S \) of \( N \) vertices such that for each vertex \( v \in V(G) \), we have \( |N[v] \cap S| \geq r \). The problem **r-Domatic Partition** consists of deciding whether the vertices of a given graph \( G \) can be partitioned into \( r \) sets \( S_1, \ldots, S_r \) such that each \( S_i \) is a dominating set for \( G \).

**r-Dominating Set** can be expressed using the following formula \( \varphi_{r-DS}(S) \), for which it holds that \( s(\varphi_{r-DS}(S)) = O(n) \):

\[
\varphi_{r-DS}(S) = \forall x \exists y_1 \exists y_2 \ldots \exists y_r \ [((x = y_1) \lor E(x,y_1)) \land \cdots \land ((y_{r-1} = y_r) \lor E(y_{r-1},y_r)) \land S(y_r)],
\]

**r-Threshold Dominating Set** can be expressed using the following formula \( \varphi_{r-TDS}(S) \), for which \( s(\varphi_{r-TDS}(S)) = O(n) \):

\[
\varphi_{r-TDS}(S) = \forall x \exists y_1 \exists y_2 \ldots \exists y_r \ [S(y_1) \land \cdots \land S(y_r) \land ((x = y_1) \lor E(x,y_1)) \land E(x,y_2) \land \cdots \land E(x,y_r)].
\]

**r-Domatic Partition** can be expressed using the following formula \( \varphi_{r-DP}(S) \) for which \( s(\varphi_{r-DP}(S)) = O(n) \):

\[
\varphi_{r-DP} = \exists S_1 \ldots \exists S_r \ [\forall x \ (\bigwedge_{1 \leq i < j \leq r} (\neg S_i(x) \lor \neg S_j(x))) \land \bigvee_{1 \leq i \leq r} S_i(x)] \land \varphi_{DS}(S_1) \land \cdots \land \varphi_{DS}(S_r).
\]

Finally, the **Perfect Code** problem [19] is defined as follows: Given a graph \( G \) and \( N \in \mathbb{N} \), decide if there exists a subset \( S \) of \( N \) vertices such that, for every vertex \( v \in V \), we have \( |N[v] \cap S| = 1 \). The problem is in Linear Monadic NP, because it can be expressed using the following formula \( \varphi_{PC}(S) \), for which it can be verified that \( s(\varphi_{PC}(S)) = 3n + 3 \):

\[
\varphi_{PC}(S) = [\forall x \forall y (\neg E(x,y) \lor \neg S(x) \lor \neg S(y))] \land [\forall z (S(z) \lor \exists y (S(y) \land E(z,y)))]
\]

\[
\land [\forall z \forall x \forall y (\neg S(x) \lor \neg S(y) \lor \neg E(x,z) \lor \neg E(y,z))].
\]
Via standard reductions from 3-CNF-Sat, it can be easily shown that all the problems discussed above are complete for Linear Monadic NP under serf-reducibility. Therefore, we have the following:

**Corollary 3.8.** For any of the problems Dominating Set, Red-Blue Dominating Set, Independent Dominating Set, Dominating Clique, $r$-Threshold Dominating Set, Distance-$r$ Dominating Set, $r$-Domatic Partition, and Perfect Code, the following holds: the problem is solvable in subexponential time if and only if ETH fails.

We compiled in Section A a list of natural problems that are equivalent to $k$-CNF-Sat modulo serf-reducibility. This list is not at all comprehensive, but it includes some of the well-known and widely used NP-hard problems in the literature.

### 3.3 Inexpressibility in Monadic SNP Logic

We prove in this subsection that Dominating Set is inexpressible in monadic SNP logic and show how the proof can be straightforwardly adapted to other considered problems. In particular, we show that there exists no formula $\phi$ in this logic with one free set variable such that $(G, S) \models \phi(S)$ if and only if $S$ is a dominating set in $G$. This shows that the greater freedom in quantifier alternation offered by the proposed logic fragment is necessary to express this problem.

The key tool we use to prove inexpressibility is the so-called Ajtai-Fagin game [2, 40], which is closely related to the adaptation of the well-known Ehrenfeucht-Fraïssé game to Monadic NP (see, e.g., the work of Fagin [26] or Schwentick [47]). It is known that these two games both characterize Monadic NP [2, 26]. The first application of the Ehrenfeucht-Fraïssé game in this setting was a proof that connectivity is not in Monadic NP [25], while the Ajtai-Fagin game was first used to show that directed reachability is not in Monadic NP [2].

Before we give a formal proof of the result, we will need several notions from the area of logic [40]. The quantifier rank of a formula is the maximum nesting depth of quantifiers in the formula. A graph equipped with sets $(U_1, \ldots, U_\ell)$ is a graph with additional unary relations $U_1, \ldots, U_\ell$ over its variable set. This constitutes a special case of a relational structure, and atoms of first-order formulas over such graphs can then naturally not only probe adjacency of a pair of vertices (the standard “edge” relation of the graph) but also inclusion of a specific vertex in a specific set (the $\ell$-many unary “set” relations).

Let $\Gamma$ be a fragment of first-order logic, and let $\Gamma[k]$ denote the restriction of $\Gamma$ to formulas of quantifier rank at most $k$. Given two equipped graphs $A$ and $B$, we say that $A$ and $B$ agree on $\Gamma[k]$ if and only if the following holds for each formula $\phi \in \Gamma[k]$:

$$A \models \phi \iff B \models \phi.$$  

**Definition 3.9.** Let $P$ be a property of graphs equipped with a single set, let $\ell, k \in \mathbb{N}$. The $(P, \ell, k, \Gamma)$-Ajtai-Fagin game is a two-player game between the duplicator and the spoiler that proceeds in the following four steps:

1. The duplicator selects a graph $G$ equipped with a set $S$ such that $(G, S) \in P$.
2. The spoiler selects $\ell$ subsets $U_1, \ldots, U_\ell$ of $V(G)$.
3. The duplicator selects a graph $G'$ equipped with a set $S'$ such that $(G', S') \notin P$, and also $\ell$ subsets $U'_1, \ldots, U'_\ell$ of $V(G')$.
4. The duplicator wins if and only if he can prove that $(G, S, U_1, \ldots, U_\ell)$ and $(G', S', U'_1, \ldots, U'_\ell)$ agree on $\Gamma[k]$.
On Existential MSO and Its Relation to ETH

Proposition 3.10 ([2, 40]). The duplicator has a winning strategy in the \((P, \ell, k, \Gamma)\)-Ajtai-Fagin game if and only if \(P\) is not definable by any formula with a single free set variable \(S\) of the form \(\exists X_1, \ldots, \exists X_\ell \chi(S, X_1, \ldots, X_\ell)\), where \(\chi\) is a formula in \(\Gamma\).

We remark that Definition 3.9 and Proposition 3.10 are streamlined versions of their counterparts in Libkin’s book [40]. Specifically, Proposition 3.10 gives more details on the relation of \(\ell, k\) to the inexpressibility of \(P\). Moreover, Definition 3.9 restricts itself to equipped graphs instead of general \(\delta\)-structures (as this is sufficient for our needs) but includes a specific handle on the fragment of FO logic that must be used in the formula (Step 4 of the game in the book [40] only considers the full FO logic, and hence equivalently states the condition in terms of the standard Ehrenfeucht-Fraïssé game).

The final ingredient we need is a result that links the number of quantifier alternations in an FO formula to the moves of the Ehrenfeucht-Fraïssé game. The alternation number of an FO formula \(\varphi\) is the number of quantifier alternations in \(\varphi\) [45]; in particular, universal first-order formulas have alternation number 0. The \(j\)-alternations \(k\)-round Ehrenfeucht-Fraïssé game is then identical to the classical Ehrenfeucht-Fraïssé game, however, the spoiler is only allowed to change the structure he plays on at most \(j\) times. For completeness, we provide a formal definition below. We refer to Libkin’s book [40] for a definition of partial isomorphism.

Definition 3.11. The \(j\)-alternations \(k\)-round Ehrenfeucht-Fraïssé game is a two-player game played in \(k\) rounds on a pair \(\mathcal{A}, \mathcal{B}\) of relational structures. The players are called the spoiler and the duplicator. At the beginning of the game, the spoiler picks one of the structures, which becomes the active structure, and we set the alternation counter \(z\) to 0. Each round of the game consists of the following steps:

1. The spoiler picks one of the relational structure. If the picked structure is not active, then \(z := z + 1\) and the newly picked structure is set to active instead of the old one. If \(z > j\), then the duplicator immediately wins.

2. The spoiler makes a move by picking an element of the picked structure (either \(a \in \mathcal{A}\) or \(b \in \mathcal{B}\)).

3. The duplicator responds by picking an element in the other structure.

After \(k\) rounds have passed (and assuming the duplicator has not already won due to Step 1), the duplicator wins if and only if the sequence of moves \((a_1, \ldots, a_k)\) on \(\mathcal{A}\) defines a partial isomorphism with the sequence of moves \((b_1, \ldots, b_k)\) on \(\mathcal{B}\).

Theorem 3.12 ([45]). Let \(\Gamma\) be the restriction of first-order logic to formulas of alternation number at most \(j\). The duplicator wins the \(j\)-alternations \(k\)-round Ehrenfeucht-Fraïssé game on \((\mathcal{A}, \mathcal{B})\) if and only if \(\mathcal{A}\) and \(\mathcal{B}\) agree on \(\Gamma[k]\).

As an immediate corollary of Theorem 3.12, we obtain that the duplicator wins the \(0\)-alternations \(k\)-round Ehrenfeucht-Fraïssé game on \((\mathcal{A}, \mathcal{B})\) if and only if \(\mathcal{A}\) and \(\mathcal{B}\) agree on all universal first-order formulas of quantifier rank at most \(k\). We are now ready to prove our inexpressibility result.

Proposition 3.13. Dominating Set is inexpressible in monadic SNP logic.

Proof. We use the Ajtai-Fagin game and describe a winning strategy for the duplicator for every fixed \(\ell, k \in \mathbb{N}\). In Step 1, the duplicator selects a graph \(G\) that consists of \(2^{2\ell} \cdot k + 1\) copies of \(K_2\), and a set \(S\) that contains a single vertex from each copy of \(K_2\); observe that \(S\) is a dominating set of \(G\). In Step 2, the spoiler arbitrarily selects his subsets \(U_1, \ldots, U_\ell\) of vertices of \(G\).

Before proceeding to Step 3, we need to find a “sufficiently frequent” configuration in \((G, U_1, \ldots, U_\ell)\) for the duplicator to exploit. Let the type \(T(v)\) of a vertex \(v \in G\) be defined as
$T(v) = \{ i \mid v \in U_i \}$. Let $a, b$ be vertices of $G$ that form a $K_2$ such that $b \in S$. Then the configuration $C(a, b)$ of $a, b$ is the tuple $(T(a), T(b))$. Since the number of distinct types is upper-bounded by $2^\ell$, the number of distinct configurations is upper-bounded by $2^{2\ell}$. By construction, there must exist some configuration that occurs at least $k + 1$ times in $G$; let us now fix an arbitrary such configuration $(T_1, T_2)$.

In Step 3, the duplicator selects a graph $H$ equipped with a set $Q$. The graph $H$ consists of a copy of $G$ and one isolated vertex $p$. Let $f$ be a bijection witnessing the isomorphism from $H − p$ to $G$; then the duplicator selects $Q$ and also the vertex-subsets $W_1, \ldots, W_\ell$ of $H$ as follows:

1. for each vertex $w \in H − p$, $w \in Q$ if and only if $f(w) \in S$;
2. for each vertex $v \in H − p$ and each $i \in [\ell]$, $v \in W_i$ if and only if $f(v) \in U_i$; and
3. $p \in W_i$ if and only if $i \in T_1$.

In Step 4, it suffices to prove that $(G, S, U_1, \ldots, U_\ell)$ and $(H, Q, W_1, \ldots, W_\ell)$ agree on all first-order formulas with alternation number 0 and quantifier rank at most $k$. By Theorem 3.12, this is equivalent to giving a winning strategy for the duplicator in the $0$-alternations $k$-round Ehrenfeucht-Fraïssé game.

So, assume the spoiler chooses to play on $G$. Then after each vertex $x$ picked by the spoiler in $G$, the duplicator responds by picking $f^{-1}(x)$ in $H$. Let $X = \{ x_1, \ldots, x_k \}$ be the $k$ moves of the spoiler in $G$ and $Y = \{ y_1, \ldots, y_k \}$ be the moves of the duplicator in $H$. Since $f$ is a bijection between $G$ and $H − p$ that furthermore preserves inclusion in the selected sets, it follows that there is a partial isomorphism between $(G[X], S, U_1 \cap X, \ldots, U_\ell \cap X)$ and $(H[Y], Q, W_1 \cap Y, \ldots, W_\ell \cap Y)$. In other words, the duplicator has a winning strategy in this case.

However, assume the spoiler chooses to play on $H$. Let $R = \{ a, b \mid ab \in E(H) \land C(f(a), f(b)) = (T_1, T_2) \}$ and let $R' = \{ f(z) \mid z \in R \}$. Then after each vertex $y$ picked by the spoiler in $H \setminus (p \cup R)$, the duplicator simply responds by picking $f(y)$ in $G$. However, for the spoiler’s moves in $(p \cup R)$, the duplicator needs to exploit that $G$ contains a sufficient number of $K_2$ graphs with the same configuration rather than blindly using $f$. Formally, we will describe the duplicator’s responses by defining a partial function $f'$ (i.e., a binary relation) from $R \cup \{ p \}$ to $R'$. At the beginning, let $f' = \emptyset$ and let all $K_2$ graphs in $G[R']$ be marked as free. Whenever the spoiler moves on a vertex $y \in R \cup \{ p \}$, the duplicator checks whether $f'(y)$ is already defined, and if so, then the duplicator responds by picking $f'(y)$. If $f'(y)$ is not defined, then the duplicator proceeds as follows:

1. choose an arbitrary free $K_2$ with vertex set $\{v, w\}$ in $G[R']$ such that $v \not\in S$, and mark $G[\{v, w\}]$ as not free;
2. if $y = p$, then set $f'(y) = v$;
3. if $y \in Q$, then set $f'(y) = w$, and for the unique neighbor $y_0$ of $y$ set $f'(y_0) = v$; and
4. if $y \not\in Q$, then set $f'(y) = v$, and for the unique neighbor $y_0$ of $y$ set $f'(y_0) = w$;
5. finally, the duplicator picks $f'(y)$.

To complete the proof, it suffices to verify that $(G[X], S, U_1 \cap X, \ldots, U_\ell \cap X)$ and $(H[Y], Q, W_1 \cap Y, \ldots, W_\ell \cap Y)$ are partially isomorphic. Since $f$ is a bijection between $G \setminus R'$ and $H \setminus (\{p\} \cup R)$ that furthermore preserves inclusion in the selected sets, it suffices to verify that $(G[X \cup R'], S, U_1 \cap X, \ldots, U_\ell \cap X)$ and $(H[Y \cup R \cup \{p\}], Q, W_1 \cap Y, \ldots, W_\ell \cap Y)$ are partially isomorphic. By construction, for each move $y$ of the spoiler, the duplicator’s response $f'(y)$ will be contained in the same sets. Furthermore, it is easy to verify that if a pair of moves $y_1, y_2$ of the spoiler are adjacent, non-adjacent, or placed on the same vertex, then by construction the responses of the duplicator will also be adjacent, non-adjacent, or placed on the same vertex, respectively. Hence, we conclude that these structures indeed admit a partial isomorphism and in particular the duplicator has a winning strategy regardless of which structure the spoiler starts on.

\[\square\]
The same general technique can be applied to prove the inexpressibility of other problems considered in Subsection 3.2. For instance, an identical construction can be used to prove that Independent Dominating Set and Perfect Code are inexpressible in SNP logic. For Red-Blue Dominating Set, it suffices to very slightly adjust the construction: In Step 1 each $K_2$ contains one red vertex $a$ and one blue vertex $b$, and $S$ contains all the "$a$" vertices; and in Step 3 the new vertex $p$ is blue. For Non-Blocker, it is also sufficient to make a simple adjustment to the above proof: In Step 3, the duplicator also adds $p$ to $Q$. Below, we explain how to use the framework to prove the inexpressibility of Linear Circuit-Sat.

**Proposition 3.14.** Linear Circuit-Sat is inexpressible in monadic SNP logic.

**Proof.** We use the Ajtai-Fagin game similarly as in the case of Dominating Set, however, we need to adjust the construction used by the duplicator. Our construction uses the gadget $\alpha$, which contains two unique variables $a, b$, three literals $a, \overline{a}, b$, and two $\lor$-gates: $\lor_1$ with input $a$ and $\lor_2$ with input $\overline{a}, b$. Observe that both $\lor$-gates in $\alpha$ are satisfiable and that $\alpha$ contains 5 vertices (3 literals and 2 gates). Let $\alpha'$ be obtained from $\alpha$ by deleting the $b$ literal, and observe that $\alpha'$ is not satisfiable. All $\lor$-gates will later be connected to a single output $\land$-gate.

Let us now fix arbitrary numbers $k, \ell \in \mathbb{N}$ and give a winning strategy for the duplicator in the Ajtai-Fagin game. In Step 1, the duplicator selects a circuit $G$ that consists of $2^{5\ell} \cdot k + 1$ gadgets $\alpha$, with all the $\lor$-gates connected to a single output $\land$-gate $z$. It is easy to see that $G$ is satisfiable. In Step 2, the spoiler selects some subsets $U_1, \ldots, U_\ell$ of vertices of $G$.

Before we proceed to Step 3, we will once again need to find a sufficiently large set of gadgets $\alpha$ that are indistinguishable w.r.t. the spoiler’s sets; recall that the type $T(v)$ of a vertex $v$ is the set $\{i \mid v \in U_i\}$. We define the configuration $C(\alpha)$ of a gadget $\alpha$ as the tuple $(T(a), T(\overline{a}), T(b), T(\lor_1), T(\lor_2))$. Since the number of distinct types is upper-bounded by $2^\ell$, the number of distinct configurations is upper-bounded by $2^{5\ell}$. By construction, there must exist some configuration that occurs at least $k + 1$ times in $G$; let us now fix an arbitrary such configuration $C$. For brevity, we will slightly abuse notation and use $C[z]$ to denote the type of $z$ in $C$.

In Step 3, the duplicator selects a circuit $H$ that consists of a copy of $G$ and one gadget $\alpha'$, whose $\lor$-gates are also connected to the single output $\land$-gate $z$. Observe that $H$ is not satisfiable. Let $f$ be a bijection witnessing the isomorphism from $H - \alpha'$ to $G$; then the duplicator selects the vertex-subsets $W_1, \ldots, W_\ell$ of $H$ as follows:

1. for each vertex $w \in H - V(\alpha')$ and each $i \in [\ell]$, $w \in W_i$ if and only if $f(w) \in U_i$; and
2. for each vertex $v \in \{a, \overline{a}, \lor_1, \lor_2\}$ of $\alpha'$, we set $v \in W_i$ if and only if $i \in C[v]$.

In Step 4, it suffices to prove that $(G, U_1, \ldots, U_\ell)$ and $(H, W_1, \ldots, W_\ell)$ agree on all first-order formulas with alternation number 0 and quantifier rank at most $k$. By Theorem 3.12, this is equivalent to giving a winning strategy for the duplicator in the $0$-alternations $k$-round Ehrenfeucht-Fraïssé game. Here, we proceed analogously as in the proof of Proposition 3.13.

In particular, assume the spoiler chooses to play on $G$. Then after each vertex $x$ picked by the spoiler in $G$, the duplicator responds by picking $f^{-1}(x)$ in $H$. Let $X = \{x_1, \ldots, x_k\}$ be the $k$ moves of the spoiler in $G$ and $Y = \{y_1, \ldots, y_k\}$ be the moves of the duplicator in $H$. Since $f$ is a bijection between $G$ and $H - \alpha'$ that furthermore preserves inclusion in the selected sets, it follows that there is a partial isomorphism between $(G[X], U_1 \cap X, \ldots, U_\ell \cap X)$ and $(H[Y], W_1 \cap Y, \ldots, W_\ell \cap Y)$. In other words, the duplicator has a winning strategy in this case.

However, assume the spoiler chooses to play on $H$. Let $R$ be the set of all vertices of $H$ that occur in some gadget $\alpha$ whose configuration is $C$, and let $R' = \{f(z) \mid z \in R\}$. Then after each vertex $y$ picked by the spoiler in $H \setminus (\alpha' \cup R)$, the duplicator simply responds by picking $f(y)$ in $G$. However, for the spoiler’s moves in $V(\alpha') \cup R$, the duplicator needs to exploit that $G$ contains...
a sufficient number of gadgets $\alpha$ with configuration $C$. Formally, we will describe the duplicator’s responses by defining a partial function $f’$ from $R \cup V(\alpha')$ to $R'$. At the beginning, let $f’ = \emptyset$ and let all gadgets $\alpha$ in $G[R']$ be marked as free. Whenever the spoiler moves on a vertex $y \in R \cup V(\alpha')$, the duplicator checks whether $f’(y)$ is already defined, and if so, the duplicator responds by picking $f’(y)$. If $f’(y)$ is not defined, the duplicator proceeds as follows:

1. choose an arbitrary free gadget $\alpha$ in $G[R’]$, say $\tau$, and mark $\tau$ as not free;
2. for each vertex $v \in \{a, \overline{a}, b, \overline{v}_1, \overline{v}_2\}$ in the gadget containing $y$ (note that $b$ might not be present if $y$ is in $\alpha’$), set $f’(v)$ to be the corresponding vertex in $\tau$;
3. finally, the duplicator picks $f’(y)$.

To complete the proof, it suffices to verify that $(G[X], U_1 \cap X, \ldots, U_\ell \cap X)$ and $(H[Y], W_1 \cap Y, \ldots, W_\ell \cap Y)$ are partially isomorphic. Since $f$ is a bijection between $G \setminus R’$ and $H \setminus (V(\alpha’) \cup R)$, which furthermore preserves inclusion in the selected sets, it suffices to verify that $(G[X] \cup R’], U_1 \cap X, \ldots, U_\ell \cap X)$ and $(H[Y] \cup R \cup V(\alpha’)], W_1 \cap Y, \ldots, W_\ell \cap Y)$ are partially isomorphic. By construction, for each move of the spoiler $y$ the duplicator’s response $f’(y)$ will be contained in the same sets. Furthermore, it is easy to verify that if a pair of moves of the spoiler $y_1, y_2$ are adjacent, non-adjacent, or placed on the same vertex, then by construction the responses of the duplicator will also be adjacent, non-adjacent, or placed on the same vertex, respectively. Hence, we conclude that these structures indeed admit a partial isomorphism and in particular the duplicator has a winning strategy regardless of which structure the spoiler starts on.

## 4 FEEDBACK VERTEX SET

In the previous section, we gave a logic fragment such that every problem that is expressible in this logic fragment is serf-reducible to 3-CNF-Sat. A natural question to ask is whether there exist NP-complete problems that are not serf-reducible to 3-CNF-Sat, under some plausible complexity-theoretic hypothesis. This question has been answered by several works. For instance, Calabro et al. [9] showed that, unless ETH fails, the restriction of CNF-Sat to instances in which the number of clauses is super-linear in the number of variables is not serf-reducible to 3-CNF-Sat. More unlikely complexity-theoretic consequences befall if we replace CNF-Sat restricted to instances with super-linear number of clauses with “harder” satisfiability problems (e.g., general CNF-Sat or even Circuit-Sat). While the aforementioned problems are all expressible in EMSO, their expressions yield super-linear measures, as defined in the previous section. Other examples of problems that are expressible in EMSO with a super-linear measure in the parameter, but, unless ETH fails, are not serf-reducible to 3-CNF-Sat, include CNF-Sat and Hitting Set.

We also saw in the previous section that many natural graph problems, such as $c$-Colorability (for any integer-constant $c > 0$), Independent Set, Vertex Cover, and Dominating Set are serf-reducible to 3-CNF-Sat, which raises the question of whether there is any natural graph problem that is not serf-reducible to 3-CNF-Sat. While it is not difficult to define a graph problem that, unless ETH fails, is not serf-reducible to 3-CNF-Sat, we do not know of any natural graph problem for which the aforementioned statement can be proved. In this section, we propose Feedback Vertex Set as a possible such candidate problem. The reason we believe that Feedback Vertex Set might be such a problem is that Feedback Vertex Set implicitly embodies a Hitting Set problem [10] (or a satisfiability problem) with possibly exponentially many cycles to hit. Whereas in CNF-Sat and in Hitting Set the sets/clauses are given explicitly (i.e., are part of the input), which allows us to quantify over the set of all sets/clauses, and hence, express these problems

---

5It is easy to see that, unless ETH fails, Hitting Set is not serf-reducible to 3-CNF-Sat because it is equivalent to general CNF-Sat under serf-reductions.
in monadic NP, the cycles in an instance of Feedback Vertex Set are implicitly encoded in the input graph. Moreover, enumerating all the cycles in a graph may require exponential time, which surpasses the allowed time in a serf-reduction.

While we are unable to prove that Feedback Vertex Set is not serf-reducible to 3-CNF-Sat (assuming ETH does not fail), we could prove that Feedback Vertex Set is inexpressible in EMSO, which rules out the possibility of using the proposed framework in this article to show that Feedback Vertex Set is serf-reducible to 3-CNF-Sat. We leave it as an open problem whether Feedback Vertex Set is serf reducible to 3-CNF-Sat. We also show in this section that there is a polynomial-time reduction from Feedback Vertex Set to 3-CNF-Sat that maps an instance of Feedback Vertex Set with $n$ vertices to an equivalent instance of 3-CNF-Sat with $O(n \log n)$ variables, and we define a variant of Feedback Vertex Set that is equivalent to 3-CNF-Sat under serf-reducibility.

4.1 Inexpressibility of Feedback Vertex Set in EMSO

Recall that, by Theorem 3.4, we could obtain a serf-reduction from Feedback Vertex Set to 3-CNF-Sat if there existed an EMSO formula $\varphi(X)$ with linear measure such that $(G, S) \models \varphi(S)$ if and only if $S$ is a feedback vertex set in $G$. In this subsection, we will prove that this approach cannot work; in particular, we show that feedback vertex set is not even expressible in EMSO.

Our first step lies in showing that Acyclicity, the problem of deciding whether a given undirected graph is acyclic, is inexpressible by an EMSO sentence. However, instead of using the Ajtai-Fagin game (which would require a highly technical specification of a strategy for the duplicator), our proof will rely on the classical notion of Hanf-locality. We provide a brief introduction below; a more in-depth overview is given, for instance, in Libkin’s book [40].

Let $d, k \in \mathbb{N}$, $G$ be a graph and $\bar{u} = (u_1, \ldots, u_k) \in V(G)^k$. The radius-$d$ ball around $\bar{u}$ is the set of vertices of $G$ that have distance at most $d$ to at least one vertex in $\bar{u}$. Then the $d$-neighborhood of $\bar{u}$ in $G$, denoted $N_d(\bar{u})$, is the subgraph of $G$ induced on the radius-$d$ ball around $\bar{u}$ with $k$ additional constants interpreted as $u_1, \ldots, u_k$. We say that a graph $G$ is equipped with sets $\bar{A} = (A_1, \ldots, A_k)$ if $G$ has $k$ designated subsets of $V(G)$, denoted $A_1, \ldots, A_k$; similarly, an equipment is the enrichment of a graph by vertex-subsets.

**Definition 4.1.** Let $G, H$ be graphs equipped with sets $\bar{U} = (U_1, \ldots, U_r)$ and $\bar{W} = (W_1, \ldots, W_r)$, respectively. Let $\bar{u} = (u_1, \ldots, u_k) \in V(G)^k$ and $\bar{w} = (w_1, \ldots, w_k) \in V(H)^k$. We write

$$(G, \bar{U}, \bar{u}) \equiv_d (H, \bar{W}, \bar{w})$$

if there exists a bijection $f : V(G) \to V(H)$ such that for every $c \in V(G)$,

$$N_d^{(G, U_1, \ldots, U_r)}(\bar{u} \circ c) \text{ is isomorphic to } N_d^{(H, W_1, \ldots, W_r)}(\bar{w} \circ f(c)).$$

We remark that isomorphism here refers to the isomorphism of labeled graphs, i.e., preserving inclusion of vertices in subsets.

**Theorem 4.2 (Hanf-locality of first-order formulas).** Two graphs $\mathcal{A}$ and $\mathcal{B}$ equipped with $\ell$ sets agree on a first-order formula $\varphi$ with $k$ free variables if and only if there exists an integer $d \geq 0$ such that for all $\bar{a}(a_1, \ldots, a_k) \in V(\mathcal{A})^k$ and $\bar{b} = (b_1, \ldots, b_k) \in V(\mathcal{B})^k$, we have:

$$(\mathcal{A}, \bar{a}) \equiv_d (\mathcal{B}, \bar{b}).$$

The smallest $d$ that satisfies the above is called the Hanf-locality rank of $\varphi$. Observe that in the case of first-order sentences, the tuples $\bar{a}$ and $\bar{b}$ will be empty. We now proceed to prove the desired claim.
Lemma 4.3. Acyclicity is inexpressible in EMSO. Specifically, there exists no EMSO sentence \( \phi \) such that \( G \models \phi \) if and only if \( G \) is a forest.

Proof. Suppose for a contradiction that Acyclicity is definable by an EMSO sentence \( \Psi = \exists Z_1 \ldots Z_\ell \psi \), and assume w.l.o.g. that \( \ell > 0 \). Since \( \psi \) is a first-order sentence (over a graph equipped with an assignment of the variables \( Z_1, \ldots, Z_\ell \) to vertex sets), it is Hanf-local. Let \( d \) be the Hanf-locality rank of \( \psi \).

Let \( t = \nu(2d+1) \) and \( r = (2d + 3)t + 2d + 2 \). We first claim the following: For every path of length at least \( r \) that is equipped with \( \ell \) sets, there exist two vertices \( a \) and \( b \) such that the distance between them is at least \( 2d + 2 \) and their \( d \)-neighborhoods are isomorphic. Indeed, for each vertex \( a \) of distance at least \( d + 1 \) from an endpoint of the path \( G \), the \( d \)-neighborhood of \( a \) is a path of length \( 2d + 1 \) with \( a \) being the middle vertex. Each vertex on the chain can belong to a subset of the \( \ell \) equipped sets, and there are \( 2^\ell \) such subsets. Hence, there are at most \( t \) different possible choices of \( d \)-neighborhoods (up to isomorphism) for vertices at distance at least \( d + 1 \) from each endpoint of the path. If the length of the path is at least \( r = (2d + 3)t + 2d + 2 \), then there is one \( d \)-neighborhood (up to isomorphism) that is realized by at least \( 2d + 3 \) vertices of distance at least \( d + 1 \) from any endpoint, and hence at least two of these vertices will be at distance at least \( 2d + 2 \) from each other.

Now let \( G \) be a path of length at least \( r \). Since \( G \) is acyclic, we have \( G \models \Psi \). Let \( U_1, \ldots, U_m \) witness this fact, that is, \( (G, U_1, \ldots, U_m) \models \Psi \). Let \( a, b \) be vertices of distance at least \( 2d + 2 \) from each other such that \( N_d^{(G, U_1, \ldots, U_m)}(a) \) is isomorphic to \( N_d^{(G, U_1, \ldots, U_m)}(b) \); the existence of such a pair of vertices has been established above. Let \( p \) be the unique endpoint of the path that is closer to \( b \) than to \( a \), and let \( a' \) be the unique neighbor of \( a \) that is closer to \( p \) (than the other neighbor of \( a \)), and let \( b' \) be that of \( b \). We construct a new graph \( G' \) by removing edges \( aa' \) and \( bb' \) from \( G \) and adding edges \( ab' \) and \( ba' \) into \( G \). We claim that for every vertex \( c \),

\[
N_d^{(G, U_1, \ldots, U_m)}(c) \approx N_d^{(G', U_1, \ldots, U_m)}(c).
\]

First, observe that the \( d \)-neighborhood of any vertex \( c \) in \( G \) of distance at least \( d + 1 \) from either \( a, b, a', b' \) is the same as the \( d \)-neighborhood of \( c \) in \( G' \). Let us fix an ordering of the vertices of \( G \): \( v \leq w \) if and only if the distance between \( v \) and \( p \) is greater than that between \( w \) and \( p \). Consider a vertex \( c \leq a \) such that the distance \( d_0 \) between \( c \) and \( a \) is at most \( d \). Then the \( d \)-predecessors of \( c \) as well as the \( d_0 \) successors of \( c \) are clearly the same in both graphs. Furthermore, since \( N_d^{(G, U_1, \ldots, U_m)}(a) \) is isomorphic to \( N_d^{(G, U_1, \ldots, U_m)}(b) \), the subgraphs induced by the \( d \) successors of \( c \) in both \( G \) and \( G' \) are isomorphic. Hence, the claim also holds for such \( c \). For remaining cases \( c \) is at distance at most \( d \) to either \( b \) or \( a \), and \( a \leq c, c \leq b, \) or \( b \leq c \), the same argument as the one above shows that \( N_d^{(G, U_1, \ldots, U_m)}(c) \) is isomorphic to \( N_d^{(G', U_1, \ldots, U_m)}(c) \), and hence the claim holds.

By Theorem 4.2, it follows that \( (G, U_1, \ldots, U_m) \) and \( (G', U_1, \ldots, U_m) \) agree on \( \Psi \). In particular, this means that \( (G', U_1, \ldots, U_m) \models \psi \) and hence \( G' \models \exists Z_1, \ldots, Z_m \psi \), that is, \( G' \models \Psi \). However, \( G' \) is not acyclic, which contradicts our assumption that \( \Psi \) is an EMSO sentence defining Acyclicity.

\[ \square \]

Proposition 4.4. Feedback Vertex Set is inexpressible in EMSO logic.

Proof. Assume for a contradiction that there exists an EMSO formula \( \phi(X) \) expressing Feedback Vertex Set. Consider the EMSO formula \( \phi'(X) = \phi(X) \land (\forall x : \neg X(x)) \). Then \( G \models \phi'(\emptyset) \) if and only if \( G \) is acyclic. However, since \( \phi'(\emptyset) \) can be rewritten as a closed EMSO formula (by replacing, for each vertex variable \( z \), each occurrence of the atom \( X(z) \) by false), this contradicts Lemma 4.3.

\[ \square \]

ACM Transactions on Computation Theory, Vol. 12, No. 4, Article 22. Publication date: September 2020.
4.2 Reductions between $3$-CNF-Sat and Feedback Vertex Set

In this subsection, we examine the existence of polynomial-time serf-reductions between $3$-CNF-Sat and Feedback Vertex Set. Based on the framework developed by Dell and van Melkebeek [22], we can show that it is unlikely that $3$-CNF-Sat is polynomial-time serf-reducible to Feedback Vertex Set:

**Proposition 4.5.** Unless the polynomial-time hierarchy collapses to its third level, $3$-CNF-Sat is not polynomial-time serf-reducible to Feedback Vertex Set.

**Proof.** Dell and van Melkebeek [22] showed that unless coNP is a subset of NP/poly—which would imply that the polynomial-time hierarchy collapses to its third level, $3$-CNF-Sat does not have a communication protocol of cost $O(n^{3-\varepsilon})$, where $\varepsilon > 0$ and $n$ is the number of Boolean variables in a $3$-CNF-Sat instance. If $3$-CNF-Sat were polynomial-time serf-reducible to Feedback Vertex Set, then this would imply that there is a polynomial time reduction that, given an instance of $3$-CNF-Sat with $n$ variables, encodes it as an instance of Feedback Vertex Set with $O(n)$ vertices, and hence, with instance size $O(n^2 \lg n) = O(n^{3-\varepsilon})$, for any $0 < \varepsilon < 1$. This in turn, would imply the existence of a communication protocol of cost $O(n^{3-\varepsilon})$ for $3$-CNF-Sat.

While we are unable to rule out—under plausible assumptions—the existence of a polynomial-time serf-reduction from Feedback Vertex Set to $3$-CNF-Sat, we give a polynomial-time reduction from Feedback Vertex Set to $3$-CNF-Sat at the cost of a logarithmic factor increase in the number of variables in the $3$-CNF-Sat instances. We first need the following lemma:

**Lemma 4.6.** Let $G$ be a graph with $n$ vertices. Then $G$ has a feedback vertex set of cardinality $k$ if and only if there is a bijection $\varphi : V(G) \rightarrow [n]$ such that, for all $uv, uw \in E(G)$:

$$(\varphi(u) > k) \land (\varphi(v) > k) \land (\varphi(w) > k) \Rightarrow (\varphi(v) > \varphi(u)) \lor (\varphi(w) > \varphi(u))$$

**Proof.** Suppose that $G$ has a feedback vertex set $F$ of cardinality $k$. Let $\varphi_1 : F \rightarrow [k]$ be any bijection. We show next how to extend $\varphi_1$ to a bijection $\varphi : V(G) \rightarrow [n]$. Let $H = G[V(G) \setminus F]$, and note that $H$ is a forest. We associate with every vertex $v$ in $H$ a number in $\{1, \ldots, n\}$ as follows: Initialize a value, $\text{stamp}$, to $k + 1$. Perform an in-order traversal of $H$ during which a vertex is assigned the value stamp when it is visited, and increment the value of stamp (by 1) after visiting each vertex. Denote by $\text{stamp}(v)$ the number assigned to a vertex $v \in H$. Define the function $\varphi : V(G) \rightarrow [n]$ by:

$$\varphi(v) = \begin{cases} 
\varphi_1(v) & \text{if } v \in F \\
\text{stamp}(v) & \text{otherwise}.
\end{cases}$$

Clearly, $\varphi$ is a bijection assigning each vertex a number in $[n]$. To show that $\varphi$ satisfies the property stated in the lemma, let $uv, uw \in E(G)$ such that $\varphi(u), \varphi(v), \varphi(w) > k$. By the definition of $\varphi$, the vertices $u, v, w$ are all in $H$. Moreover, since $uv, uw \in E(G)$, $u, v, w$ must belong to the same tree, $T$, in $H$, and at least one of $v, w$ must be a child of $u$ in $T$. Since the vertices in $T$ were assigned numbers according to an in-order traversal of $H$, and hence of $T$, it follows that either $\varphi(v) = \text{stamp}(v) > \text{stamp}(u) = \varphi(u)$, or $\varphi(w) = \text{stamp}(w) > \text{stamp}(u) = \varphi(u)$, and the statement of the lemma follows.

To prove the converse, suppose that there exists a bijection $\varphi : V(G) \rightarrow [n]$ satisfying the property in the statement of the lemma. Let $F = \{v \in V(G) \mid \varphi(v) \in [k]\}$. We show that $F$ is a feedback vertex set of $G$. Suppose, to get a contradiction, that this is not the case, and let $C = (u_1, \ldots, u_r = u)$ be a cycle in $G[V(G) \setminus F]$. Since $u_1u_2, u_1u_{r-1} \in E(G)$, by the property in the statement of the lemma, we have $\varphi(u_2) > \varphi(u_1)$ or $\varphi(u_{r-1}) > \varphi(u_1)$. Without loss of generality, assume that $\varphi(u_2) > \varphi(u_1)$; the argument in the other case is the same. Applying the property in the statement of the...
lemma to the vertices \( u_{i-1}, u_i, u_{i+1} \), for \( i = 2, \ldots, r - 1 \), with \( u_i \) assuming the role of \( u \) in the statement of the lemma, we obtain the following: \( \phi(u_1) > \phi(u_2) \) or \( \phi(u_3) > \phi(u_2), \phi(u_2) > \phi(u_3) \) or \( \phi(u_4) > \phi(u_5), \ldots, \) and \( \phi(u_{r-2}) > \phi(u_{r-1}) \) or \( \phi(u_r) = \phi(u_1) > \phi(u_{r-1}). \) Since \( \phi(u_2) > \phi(u_1) \), the above implies that \( \phi(u_2) < \phi(u_3), \phi(u_3) < \phi(u_4), \ldots, \phi(u_{r-1}) < \phi(u_r) = \phi(u_1), \) and hence that \( \phi(u_2) < \phi(u_1). \) This contradicts that \( \phi(u_2) > \phi(u_1) \), and completes the proof. \( \Box \)

**Theorem 4.7.** There is a polynomial-time many-one reduction that takes an instance \((G, k)\) of Feedback Vertex Set and produces an equivalent instance \( F \) of CNF-SAT such that \( F \) has \( O(n \log n) \) variables, \( n^{O(1)} \) clauses, and width \( O(\log n) \), where \( n = n(G) \).

**Proof.** Let \((G, k)\) be an instance of Feedback Vertex Set, and let \( n = n(G) \); assume without loss of generality that \( V(G) = [n] \). By Lemma 4.6, \((G, k)\) is a yes-instance of Feedback Vertex Set if and only if there exists a bijection \( \phi : V(G) \rightarrow [n] \) satisfying the property in the statement of the lemma. We will construct an instance \( F \) of CNF-SAT such that \( F \) is satisfiable if and only if such a bijection \( \phi \) exists.

We construct \( F \) as follows: Let \( s = \lfloor \log n \rfloor \). \( F \) has \( s \cdot n \) Boolean variables, partitioned into \( n \) blocks, \( B_1, \ldots, B_n \), such that each block consists of a sequence of exactly \( s \) variables. The (truth) assignment to the sequence of variables in block \( B_i \), \( i \in [n] \), will serve as the binary representation of the image of vertex \( i, \phi(i) \), under the sought bijection \( \phi \). We add a set of CNF clauses to \( F \) to ensure that the truth assignment to (the sequence of variables of) each block is the binary representation of a number in \([n]\). Since each block consists of \( s \leq \log n + 1 \) variables, it is easy to see that in \( O(n) \) time we can compute a set of \( n^{O(1)} \) CNF clauses, each of width \( O(\log n) \), that we add to \( F \) to encode the aforementioned requirement for a block. (For instance, this can be done by constructing the truth table of a Boolean function that stipulates this requirement, and then expressing this function in CNF.) The total number of clauses needed to encode the above requirement for all \( n \) blocks is \( n^{O(1)} \).

To encode that the truth assignment corresponds to a bijection, we add to \( F \) a set of CNF clauses, each of width \( O(\log n) \), encoding that, for each pair of distinct numbers \( i, j \in [n] \), the truth assignments to blocks \( B_i \) and \( B_j \) correspond to different numbers. Since the total number of variables in two blocks is \( 2s \leq 2 \log n + 2 \), the number of CNF clauses needed to encode that two distinct blocks are assigned two distinct numbers is \( n^{O(1)} \), and hence the total number of CNF clauses needed over all pairs of blocks is \( n^{O(1)} \). Finally, we need to encode that the assignment corresponds to a function \( \phi \) that satisfies the property in Lemma 4.6. Since \( k \leq n \) is given, the binary representation of \( k \) is fixed and has length at most \( s \). For two edges \( uv, uw \in E(G) \), the property in the lemma can be encoded by adding \( n^{O(1)} \) CNF clauses, each of width \( O(\log n) \), to \( F \), and hence encoding that the truth assignment corresponds to a function that satisfies the property in the lemma can be done by adding \( n^{O(1)} \) CNF clauses to \( F \), each of width \( O(\log n) \).

It follows from the above that an instance \( F \) with \( O(n \log n) \) variables can be constructed in polynomial time, such that \( F \) is a yes-instance of CNF-Sat if and only if a bijection \( \phi \) satisfying the property in Lemma 4.6 exists, and consequently, if and only if \((G, k)\) is a yes-instance of Feedback Vertex Set. Moreover, \( F \) has \( n^{O(1)} \) clauses and width \( O(\log n) \). \( \Box \)

We now define a variant of Feedback Vertex Set, denoted Monochromatic 3-Feedback Vertex Set, that we show to be equivalent under self-reducibility to 3-CNF-SAT. An instance of Monochromatic 3-Feedback Vertex Set consists of a graph \( G \) and a nonnegative integer \( k \). Each edge \( e \in E(G) \) is associated with a set of colors, denoted \( \text{Colors}(e) \). The question is to decide if there exists a subset \( Q \subseteq V(G) \) of vertices of cardinality at most \( k \) such that, for every 3-cycle \( C \) in \( G \) satisfying \( \bigcap_{e \in E(C)} \text{Colors}(e) \neq \emptyset \), \( C \) contains at least one vertex of \( Q \). That is, \( Q \) breaks every 3-cycle in \( G \) whose edges can all be assigned the same color from their color lists. It can be easily shown that Monochromatic 3-Feedback Vertex Set is NP-complete by adapting the standard reduction
from Vertex Cover to Feedback Vertex Set (the color lists of all edges are singletons and consist of the same color). By $d$-Monochromatic 3-Feedback Vertex Set, where $d \geq 1$, we denote the restriction of Monochromatic 3-Feedback Vertex Set to instances in which the total number of colors assigned is at most $d$ (that is, $|\bigcup_{e \in E(G)} \text{Colors}(e)| \leq d$). We have the following theorem:

**Theorem 4.8.** For any integer $d \geq 10$, 3-CNF-Sat and $d$-Monochromatic 3-Feedback Vertex Set are equivalent under serf-reductions. Therefore, $d$-Monochromatic 3-Feedback Vertex Set is solvable in subexponential time if and only if ETH fails.

**Proof.** It is easy to see that $d$-Monochromatic 3-Feedback Vertex Set is serf-reducible to 3-CNF-Sat, for any integer constant $d \geq 1$. Given an instance $(G, k)$ of Monochromatic 3-Feedback Vertex Set, where $G$ has $n$ vertices, in polynomial-time, we can construct an equivalent instance $F$ of 3-SAT with $n$ variables, each corresponding to a vertex in $G$, as follows: We enumerate all 3-cycles in $G$, which can be done in polynomial time. For each 3-cycle $C$, if $C$ satisfies $\bigcap_{e \in E(G)} \text{Colors}(e) \neq \emptyset$, we create a clause in $F$ consisting of the disjunctions of three variables corresponding to the vertices in $C$. Finally, we add $O(n)$ 3-CNF clauses to $F$ to encode the cardinality $k$ of the solution sought for $(G, k)$ [28]; these clauses stipulate that any satisfying assignment to $F$ must assign $k$ variables in $F$ to 1. Clearly, this is a serf-reduction from $d$-Monochromatic 3-Feedback Vertex Set to 3-CNF-Sat.

To prove that 3-CNF-Sat is serf-reducible to $d$-Monochromatic 3-Feedback Vertex Set, for any integer constant $d \geq 10$, since by Lemma 2.2 3-CNF-Sat is serf-reducible to 3-3-Sat, and serf-reducibility is transitive [35], it suffices to show that for any integer-constant $d \geq 10$, 3-3-Sat is serf-reducible to $d$-Monochromatic 3-Feedback Vertex Set. We prove the aforementioned statement for the case when $d = 10$, and—obviously—the result will follow for any $d > 10$.

Let $F$ be an instance of 3-3-Sat with variables $x_1, \ldots, x_n$, and clauses $C_1, \ldots, C_m$. We will assume, without loss of generality, that each clause in $F$ contains exactly 3 literals, and that no clause in $F$ contains both a literal and its negation. We start by showing how $F$ can be reduced in polynomial time to an instance $(H, n)$ of $(n + m)$-Monochromatic 3-Feedback Vertex Set with $3n$ vertices, and we will then show how to modify the reduction to yield an equivalent instance $(G, n)$ of 10-Monochromatic 3-Feedback Vertex Set. We construct $H$ as follows: Create a 3-cycle $(x_i, \overline{x}_i, y_i)$, called a variable-cycle, for each $i \in [n]$; let $c_i$ be a color, and add $c_i$ to $\text{Colors}(e)$ for each edge $e$ on the cycle $(x_i, \overline{x}_i, y_i)$. Each variable-cycle ensures that exactly one of $(x_i, \overline{x}_i)$ is in any solution to the instance $(H, n)$. For each clause $C_i$, $i \in [m]$, create a 3-cycle $y_i$, called a clause-cycle, consisting of the three vertices in $H$ corresponding to the literals in $C_i$; let $c_{n+i}$ be a color, and for each edge $e$ in $y_i$, add $c_{n+i}$ to $\text{Colors}(e)$. This completes the construction of $H$. It is easy to see that $H$ is a yes-instance of $(n + m)$-Monochromatic 3-Feedback Vertex Set if and only if $F$ is satisfiable. This can be argued as follows: First observe that, by construction of $H$, all monochromatic cycles in $H$ are 3-cycles that are either variable-cycles or clause-cycles. If $F$ is satisfiable, the vertices corresponding to the literals of a satisfying assignment to $H$ break all monochromatic cycles in $H$: Each variable-cycle contains exactly one such vertex, and each clause-cycle contains at least one such vertex, namely, a vertex corresponding to a literal that satisfies the clause. Conversely, suppose that there exists a subset of vertices $Q \subseteq H$ of cardinality $n$ that breaks all monochromatic cycles in $H$. Since the variable-cycles are monochromatic and mutually vertex-disjoint, $Q$ contains exactly one vertex from each variable-cycle; without loss of generality, we may assume that $Q$ does not contain any vertex $y_i$, $i \in [n]$, since each such vertex can be safely substituted by either $x_i$ or $\overline{x}_i$ to obtain a subset of vertices of the same cardinality that breaks all monochromatic cycles in $H$. Now the truth assignment to $F$ that assigns true to the literals corresponding to vertices in $Q$ is a valid truth assignment that satisfies all clauses in $F$; this is because $Q$ breaks all clause-cycles in $H$, and hence, each clause contains a literal that corresponds to a vertex in $Q$. 

ACM Transactions on Computation Theory, Vol. 12, No. 4, Article 22. Publication date: September 2020.
We now describe how to modify the instance \((H, n)\) to obtain an equivalent instance \((G, n)\), where \(V(G) = V(H)\) and \(E(G) = E(H)\), albeit with the color lists of the edges in \(G\) assigned in total at most 10 colors. To do so, it suffices to assign at most 10 colors to the edges in \(G\) while ensuring that the monochromatic cycles in \(G\) are precisely the variable-cycles and clause-cycles in \(H\). To do so, it suffices to color the variable-cycles and clause-cycles in \(H\) using a total of 10 colors so no monochromatic cycle is formed using edges from more than one (clause) cycle. The latter property can be ensured by assigning colors to the edges of the variable-cycles and edge-cycles in \(H\) so the edges of any two monochromatic cycles that are not vertex-disjoint are colored with different colors.

To achieve the above, we construct an auxiliary graph \(A\) as follows: For each variable-cycle and each clause-cycle, create a vertex in \(A\); add an edge between two vertices in \(A\) if and only if their corresponding cycles share a vertex in \(H\). The auxiliary graph \(A\) has maximum degree at most 9: Each vertex in \(H\) appears in exactly one variable-cycle and in at most 3 clause-cycles (each variable \(x_i\) in \(F\) appears at most 3 times in total, positively and negatively, in \(F\)). By Brook’s theorem [8], \(A\) can be properly colored (i.e., adjacent vertices in \(A\) are colored with different colors) with at most 10 colors; let \(\pi\) be such a proper coloring of the vertices in \(A\). We use \(\pi\) to color the variable-cycles and clause-cycles of \(H\) to obtain the graph \(G\) as follows: For each variable/clause-cycle \(\gamma\) in \(H\), let \(c\) be the color that \(\pi\) assigns to the vertex corresponding to \(\gamma\) in \(A\). For each edge \(e\) of \(\gamma\), add \(c\) to Colors\((e)\) in \(G\). Let \((G, n)\) be the resulting instance. Clearly, \((G, n)\) is an instance of 10-MONOCROMATIC 3-FEEDBACK VERTEX SET that is computable in polynomial time. From the above discussion, it follows that \((G, n)\) is a yes-instance of 10-MONOCROMATIC 3-FEEDBACK VERTEX SET if and only if \(F\) is satisfiable. Noting that \(n(G) = n(H) = 3n\), it follows that the above reduction is a polynomial-time serf-reduction from 3-3-SAT to 10-MONOCROMATIC 3-FEEDBACK VERTEX SET. This completes the proof. 

Finally, we remark that for every fixed \(d\), the optimization formulation of \(d\)-MONOCROMATIC 3-FEEDBACK VERTEX SET can be expressed in EMSO when enhanced by atomic operators that can identify edge colors in a standard way. Notably, since there are at most \(d\) colors, we may assume that the \(E\) operator is replaced by the operators \(E_1, \ldots, E_d\) where \(E_i(x, y)\) is true if and only if \(x\) and \(y\) are connected by an edge equipped with color \(i\). This allows us to express \(d\)-MONOCROMATIC 3-FEEDBACK VERTEX SET using the following formula:

\[
\phi_{d,3FVS}(S) = \forall a, b, c \left( (\neg E_1(a, b)) \lor (\neg E_1(b, c)) \lor (\neg E_1(a, c)) \lor (a \in S) \lor (b \in S) \lor (c \in S) \right) \land \\
(\neg E_2(a, b)) \lor (\neg E_2(b, c)) \lor (\neg E_2(a, c)) \lor (a \in S) \lor (b \in S) \lor (c \in S) \right) \land \cdots \land (\neg E_d(a, b)) \lor (\neg E_d(b, c)) \lor (\neg E_d(a, c)) \lor (a \in S) \lor (b \in S) \lor (c \in S) \right).
\]

5 CONCLUSION

In this article, we proposed a complexity class, Linear Monadic NP, in an effort to capture problems that are serf-reducible to \(k\)-CNF-SAT (for any integer-constant \(k \geq 3\)). We showed that this class extends and refines the class SNP proposed by Impagliazzo et al. [35]. In particular, we showed that the logic fragment (EMSO)—based on which this class is defined—allows us to place many natural problems in this class whose serf-reducibility to \(k\)-CNF-SAT was not observed before.

An important question that ensues from our work is whether we can identify some “natural” graph problems that are not serf-reducible to \(k\)-CNF-SAT, assuming a plausible complexity hypothesis. Our work suggests that FEEDBACK VERTEX SET may be one such problem, but unfortunately, we are unable to provide a rigorous argument to that effect. We feel that this line of research is missing a systematic approach for showing that certain problems are not serf-reducible to \(k\)-CNF-SAT, and more generally, a systematic approach for ruling out the existence of a serf-reduction between two problems. Such a systematic approach will require a deeper study of the notion of serf-reducibility.
We finally would like to draw the following remark: A natural question to ask is whether serf-reducibility to \( k \)-CNF-SAT is equivalent to the notion of self-sparsification (as in the sparsification lemma for \( k \)-CNF-SAT): The existence of a subexponential-time self-reduction that reduces an instance of the problem to an equivalent instance whose (instance) size is linear in the designated parameter. These two notions, in general, seem to be orthogonal. On one hand, it can be shown that Feedback Vertex Set is self-sparsifiable [30, 41], but this does not seem to imply that Feedback Vertex Set is serf-reducible to \( k \)-CNF-SAT. On the other hand, CLIQUE is serf-reducible to \( k \)-CNF-SAT, but unless ETH fails, CLIQUE is not self-sparsifiable, since this would imply that CLIQUE is solvable in subexponential time.

APPENDIX

A LIST OF WELL-KNOWN PROBLEMS THAT ARE EQUIVALENT TO \( k \)-CNF-SAT MODULO SERF-REDUCIBILITY

**INDEPENDENT SET**

**Instance:** An undirected graph \( G \); \( k \in \mathbb{N} \).

**Question:** Does \( G \) contain an independent set of cardinality \( k \)?

**Complexity parameter:** Number of vertices in \( G \).

**Reference:** Impagliazzo et al. [35].

**Comment:** Remains equivalent to \( k \)-CNF-SAT modulo serf-reducibility even on graphs of maximum degree at most 3 (see Reference [37]).

**VERTEX COVER**

**Instance:** An undirected graph \( G \); \( k \in \mathbb{N} \).

**Question:** Does \( G \) contain a vertex cover of cardinality \( k \)?

**Complexity parameter:** Number of vertices in \( G \).

**Reference:** Impagliazzo et al. [35].

**Comment:** Remains equivalent to \( k \)-CNF-SAT modulo serf-reducibility even on graphs of maximum degree at most 3.

**CLIQUE**

**Instance:** An undirected graph \( G \); \( k \in \mathbb{N} \).

**Question:** Does \( G \) contain a clique of cardinality \( k \)?

**Complexity parameter:** Number of vertices in \( G \).

**Reference:** Impagliazzo et al. [35].

**Comment:** Is equivalent to \( k \)-CNF-SAT modulo serf-reducibility for any \( k \geq 3 \).

**\( k \)-COLORING**

**Instance:** An undirected graph \( G \); \( k \in \mathbb{N} \).

**Question:** Can the vertices of \( G \) be properly colored with \( k \) colors?

**Complexity parameter:** Number of vertices in \( G \).

**Reference:** Impagliazzo et al. [35].

**Comment:** Is equivalent to \( k \)-CNF-SAT modulo serf-reducibility for any \( k \geq 3 \).

**DOMINATING SET**

**Instance:** An undirected graph \( G \); \( k \in \mathbb{N} \).

**Question:** Does \( G \) contain a dominating set of cardinality \( k \)?

**Complexity parameter:** Number of vertices in \( G \).

**Reference:** This article shows the serf-reducibility to \( k \)-CNF-SAT. The other direction follows by the standard (serf)reduction from VERTEX COVER.

**Comment:** Remains equivalent to \( k \)-CNF-SAT modulo serf-reducibility even on graphs of maximum degree at most 6.
Non-Blocker

**Instance:** An undirected graph $G$; $k \in \mathbb{N}$.

**Question:** Does $G$ contain a set $S$ of $k$ vertices such that every vertex in $S$ has a neighbor not in $S$?

**Complexity parameter:** Number of vertices in $G$.

**Reference:** This problem is the dual of DOMINATING SET.

Perfect Code

**Instance:** An undirected graph $G$; $k \in \mathbb{N}$.

**Question:** Does $G$ contain a subset $S$ of $k$ vertices such that, for every vertex $v \in V$, we have $|N[v] \cap S| = 1$?

**Complexity parameter:** Number of vertices in $G$.

**Reference:** This article shows the serf-reducibility to $k$-CNF-Sat. The other direction follows by a (serf)reduction from 3-CNF-Sat (e.g., see Reference [17]).

$r$-Dominating Set

**Instance:** An undirected graph $G$; $k \in \mathbb{N}$. ($r \geq 1$ is a fixed integer-constant.)

**Question:** Does $G$ contain a set $S$ of $k$ vertices such that each vertex in $G$ is within distance $r$ from some vertex in $S$?

**Complexity parameter:** Number of vertices in $G$.

**Reference:** This article shows the serf-reducibility to $k$-CNF-Sat; the other direction follows by an easy reduction.

Independent Dominating Set

**Instance:** An undirected graph $G$; $k \in \mathbb{N}$.

**Question:** Does $G$ contain a independent dominating set of cardinality $k$?

**Complexity parameter:** Number of vertices in $G$.

**Reference:** This article shows the serf-reducibility to $k$-CNF-Sat. The other direction follows by the standard reduction from 3-CNF-Sat, starting from instances of 3-3-Sat.

Dominating Clique

**Instance:** An undirected graph $G$; $k \in \mathbb{N}$.

**Question:** Does $G$ contain a dominating clique of cardinality $k$?

**Complexity parameter:** Number of vertices in $G$.

**Reference:** This article shows the serf-reducibility to $k$-CNF-Sat. The other direction follows by the standard reduction from 3-CNF-Sat, starting from instances of 3-3-Sat.

Red-Blue Dominating Set

**Instance:** An undirected graph $G$; $k \in \mathbb{N}$.

**Question:** Does $G$ contain a subset of $k$ red vertices that dominate all blue vertices?

**Complexity parameter:** Number of vertices in $G$.

**Reference:** This article shows the serf-reducibility to $k$-CNF-Sat. The other direction follows by an easy serf-reduction from DOMINATING SET.

**Comment:** Remains equivalent to $k$-CNF-Sat modulo serf-reducibility even on graphs of maximum degree at most 7.

$r$-Threshold Dominating Set

**Instance:** An undirected graph $G$; $k \in \mathbb{N}$. $r \geq 1$ is a fixed integer-constant

**Question:** Does $G$ contain a set $S$ of $k$ vertices such that, for each $v \in V(G)$, it holds that $|N[v] \cap S| \geq r$?
Complexity parameter: Number of vertices in $G$.
Reference: This article shows the serf-reducibility to $k$-CNF-Sat. The other direction (for any $r \geq 1$) follows by an easy serf-reduction from Dominating Set.

**Linear Circuit-Sat**

**Instance:** A Boolean circuit $C$ with $n$ input variables and $O(n)$ gates.

**Question:** Is $C$ satisfiable?

**Complexity parameter:** $n$.

**Reference:** The paper [38] shows the equivalence between this problem and $k$-CNF-Sat modulo serf-reducibility.

**Comment:** Remains equivalent to $k$-CNF-Sat even when the degree (i.e., number of occurrences) of each input variable is at most 3.

**Linear Monotone/Antimonotone Circuit-Sat**

**Instance:** A Boolean circuit $C$ with $n$ input variables and $O(n)$ gates such that all occurrences of the input variables are positive/negative; $k \in \mathbb{N}$.

**Question:** Does $C$ have a satisfying assignment of weight $k$?

**Complexity parameter:** $n$.

**Reference:** Serf-reducibility to Linear Circuit-Sat can be achieved by encoding the weight $k$ using $O(n)$ clauses (e.g., see Reference [29]). The other direction follows by a standard reduction from Vertex Cover/Independent Set on bounded-degree graphs.

**Comment:** Remains equivalent to $k$-CNF-Sat even when the degree of each input variable is at most 3.

**Linear CNF-Sat**

**Instance:** A Boolean CNF-formula $F$ with $n$ input variables and $O(n)$ clauses.

**Question:** Is $F$ satisfiable?

**Complexity parameter:** $n$.

**Reference:** One direction follows from the trivial (serf)reduction to Linear Circuit-Sat. The other direction follows because 3-3-Sat is a special case.

**Comment:** Remains equivalent to $k$-CNF-Sat even when the degree of each variable is at most 3. The weighted monotone/antimonotone version is equivalent to $k$-CNF-Sat and remains so even when the degree of each variable is at most 3.

**Linear Hitting Set**

**Instance:** A set $U$ of $n$ elements, and a family $\mathcal{F}$ of $O(n)$ subsets of $U$; $k \in \mathbb{N}$.

**Question:** Is there a subset $S \subseteq U$ of cardinality $k$ that has a nonempty intersection with every $F \in \mathcal{F}$?

**Complexity parameter:** $n$.

**Reference:** Equivalence to $k$-CNF-Sat follows by standard (serf)reductions to/from Linear Monotone CNF-Sat.

**Comment:** Remains equivalent to $k$-CNF-Sat even when each element in $U$ appears in at most three sets in $\mathcal{F}$.

**Linear Set Cover**

**Instance:** A set $U$ of $n$ elements, and a family $\mathcal{F}$ of $O(n)$ subsets of $U$; $k \in \mathbb{N}$.

**Question:** Are there $k$ sets in $\mathcal{F}$ whose union is $U$?

**Complexity parameter:** $n$. 
Reference: Equivalence to \( k \)-CNF-Sat follows by standard (serf)reductions to/from LINEAR HITTING SET

Comment: Remains equivalent to \( k \)-CNF-Sat even when each element in \( U \) appears in at most three sets in \( \mathcal{F} \).

\( r \)-Domatic Partition

Instance: An undirected graph \( G; k \in \mathbb{N} \). \( r \geq 1 \) is a fixed integer-constant.

Question: Can \( V(G) \) be partitioned into \( r \) sets such that each set is a dominating set for \( G \)?

Complexity parameter: Number of vertices in \( G \).

Reference: This article shows the serf-reducibility to \( k \)-CNF-Sat. The other direction follows by a known (serf)reduction from SET COVER [46] (starting from instances of LINEAR SET COVER).

Linear Not-All-Equal CNF-Sat

Instance: A Boolean CNF-formula \( F \) with \( n \) input variables and \( O(n) \) clauses.

Question: Does \( F \) have a satisfying assignment in which each clause in \( F \) contains at least one false literal?

Complexity parameter: \( n \).

Reference: The framework in this article can be used to show membership in Linear Monadic NP. The Linear Monadic NP completeness follows by a standard (serf)reduction from 3-3-Sat.

Linear Set Splitting

Instance: A set \( S \) of \( n \) elements, and a family \( \mathcal{F} \) of \( O(n) \) subsets of \( S \).

Question: Can \( S \) be partitioned into \( S_1, S_2 \) such that each set in \( \mathcal{F} \) has nonempty intersection with both \( S_1 \) and \( S_2 \)?

Complexity parameter: \( n \).

Reference: The framework in this article can be used to show membership in Linear Monadic NP. The Linear Monadic NP completeness follows by a standard (serf)reduction from LINEAR NOT-ALL-EQUAL CNF-SAT.

REFERENCES


[35] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which problems have strongly exponential complexity?


Received September 2016; revised May 2020; accepted June 2020