Hardness of monadic second-order formulae over succinct graphs

Guilhem Gamard, Pierre Guillon, Kévin Perrot, Guillaume Theyssier

Hardness of monadic second-order formulae over succinct graphs

G. Gamard[1], P. Guillon[2], K. Perrot[3], and G. Theyssier[2]

[1]Lorraine University, CNRS, INRIA, LORIA, Nancy, France
[2]Aix-Marseille University, CNRS, I2M, Marseille, France
[3]Aix-Marseille University, CNRS, LIS, Marseille, France

June 13, 2024

Abstract

Our main result is a succinct *counterpoint* to Courcelle's meta-theorem as follows: every arborescent monadic second-order (MSO) property is either NP-hard or coNP-hard over graphs given by succinct representations. Succint representations are Boolean circuits computing the adjacency relation. Arborescent properties are those which have infinitely many models and countermodels with bounded treewidth.

Moreover, we explore what happens when the arborescence condition is dropped and show that, under a reasonable complexity assumption, the previous dichotomy fails, even for questions expressible in first-order logic.

1 Introduction

In this paper, we are interested in deciding properties of graphs defined in monadic second-order logic (MSO). A series of results by Courcelle deals with this question; in particular [10] proves that every MSO property is decidable in linear time, given a graph with bounded treewidth (encoded by its adjacency matrix). Now what if the graph is not arbitrary, but presents some structure that allows a shorter encoding? Assume that the graph is described succinctly, i.e, by a Boolean circuit which computes the adjacency relation between nodes, which are represented by binary numbers. In this case the adjacency matrix might be exponentially larger than the circuit representation, so Courcelle's theorem does not give a polynomial-time algorithm. One natural question is whether it is possible to exploit the circuits in better ways than just querying for all possible edges, in order to more directly deduce structural information about the graph. Our main result essentially tells that it is impossible as soon as the property is non-trivial for bounded treewidth graphs.

Theorem 1. *If ϕ is an arborescent MSO sentence, then testing ϕ on graphs represented succinctly is either* NP- *or* coNP*-hard.*

Arborescent means that ϕ has infinitely many models with some fixed treewidth, and infinitely many countermodels with some fixed treewidth. Formal definitions appear in Section 2, including a definition of *treewidth*.

Succinct representations of graphs have already been considered in the literature. In [7, Table 1], the authors establish a list of natural graph properties that are polynomial-time solvable when using the usual input representations (matrix or adjacency list) but

become (at least) NP-hard when using succinct representations. Interestingly, all of these properties fall into our meta-theorem 1. Moreover, their [7, Theorem 3.1] gives a sufficient condition for NP-hardness, which can be compared to a weak form of our gluing lemmas, though our final statement, involving logic, is incomparable. Finally, [7, Open problem 1] vaguely conjectures that all nontrivial properties are NP-hard, provided that a good definition of nontriviality be given. Theorem 1 proves this conjecture for MSO sentences and some natural notion of nontriviality (and adapting it by including the symmetric co-NP-hardness case), while Theorem 22 disproves it when this is relaxed, under reasonable complexity assumptions.

Later, [8] establishes that NP-hard properties with the usual representation become NEXPTIME-hard with succinct representations, and [12] exposes such complexity lower bound conversions for weaker classes, proving for example that connectivity and planarity testing are PSPACE-hard for succinctly represented graphs. Although a complexity blowup is expected when taking the succinct version of a problem, this is not a general fact. Actually, [21] shows that when taking CNF or DNF formula as succinct representations, there are examples of problems whose complexity does not increase when encoded in the new form, or increases to an intermediate complexity class less powerful than the exponential blow up. In the present paper, we focus on MSO property testing on graphs with succinct representations by circuits.

Our motivation comes from the world of automata networks. An *automata network* (AN) can be seen as a computer network where all machines hold a local state and update synchronously by reading neighboring states and applying a local transition. To update a machine (referred to as an *automaton*) v of the network, first collect the states of its inbound neighbors into a tuple, and then feed that tuple as an input symbol to the update function of v. Globally speaking, all automata are updated synchronously (though an extensive literature has explored other *update modes* [23]), so that the state of v at time $t+1$ only depends on the states of its neighbors at time t. One of the initial intents behind this definition was to model the dynamics of gene regulation [2, 4, 15, 18]. Nowadays, automata networks are also used as a setup for distributed algorithms and as a modelling tool in engineering. Those applications have motivated the study of automata network *per se* and many theoretical properties were found [1, 3, 9, 11, 17].

In general, the automata in a network may behave nondeterministically, so that the dynamics as a whole may be nondeterministic. The local behavior of each automaton is typically described as a formula or relation, which can be gathered into one Boolean circuit. Given two binary words $z_1, \dots, z_n$ and $z'_1, \dots, z'_n$ (referred to as *configurations*) assigning states to all automata of the network, the circuit returns whether or not the first configuration can transition into the second one. It succinctly encodes the directed graph (referred to as the *transition graph*) whose vertices are the configurations, and edges follow the transitions. While seemingly artificial, this encoding is relevant for applications. When automata networks are used to model actual computer networks, it is reasonable to assume access only to the source code of the programs run by the nodes. Boolean circuits represent this source code.

The results from [24] already hint that the encoding cannot be smartly used to solve some questions efficiently. For instance, with deterministic automata networks encoded as n-bit-input n-bit-output circuits (computing the unique successor of each configuration):

Theorem 2 (see [24])**.** *Let ϕ denote a question about graphs expressible in first-order logic. It is either $O(1)$, or NP-hard, or coNP-hard, given a deterministic automata network as input, whether its transition graph satisfies ϕ.*

In particular, first-order logic cannot express any non-trivial polynomial-time solvable question about the dynamics of deterministic automata networks, unless $\mathsf{P} = \mathsf{NP}$. This is a strong indication that it is indeed not tractable to analyze the Boolean circuits given to us in order to extract structural information about the dynamics: the best we can do is to evaluate the circuits to explore the transition graph.

Contributions. The present work started as an attempt to generalize Theorem 2 in two directions: from first-order logic (FO) to monadic second-order logic (MSO), and from deterministic automata networks to nondeterministic networks. Neither generalization is trivial.

Questions about general ANs are harder than questions about deterministic ANs. For instance, the question "is the AN deterministic?" is expressible in FO. If we restrict ourselves to deterministic ANs, that question is $O(1)$; in general, it is not. Thence our generalization will prove, in particular, that determinism is either NP- or coNP-hard.

Questions in MSO are stronger than questions in FO. In particular, all minor-hereditary properties are expressible in MSO. Thence our results prove, in particular, that testing whether the transition graph is series-parallel is either NP- or coNP-hard.

In addition to being technically harder, both generalizations are useful. Indeed, when restricting ourselves to deterministic networks, we restrict ourselves to transition graphs of out-degree one. This is a strong restriction from the perspective of succinct graphs taken in the present paper. The generalization to nondeterministic networks lifts that restriction, which enables future work to explore deeper connections between automata networks, Boolean circuits, and graph combinatorics. Moreover, MSO logic allows to express the relation "there is a chain of transitions from configuration x to configuration y", which FO cannot. That relation naturally arises in many practical questions.

Things turned out more complicated than expected, and we do not get a general result: Theorem 1 requires the arborescent hypothesis. Many of the properties considered in the literature so far, and in particular questions mentioned earlier in this introduction, are arborescent (up to turning counting questions into decision questions in the usual way, e.g., "how many fixpoints?" becomes "are there more than k fixpoints?" for a fixed k).

The *arborescent* condition is crucial in our proofs, since it gives the existence of regular families of models and countermodels on which to build a polynomial reduction. It is natural to ask whether it is necessary. In Section 7, we give the following partial answer:

Theorem 3. *There is a (nonarborescent) first-order sentence ψ such that, under plausible complexity assumptions, testing ψ on a given succinctly represented graph is neither constant time, nor NP-hard, nor coNP-hard.*

Contents. In Section 2, we give the definitions and notations. In Section 3, we restate the main result and give a proof outline. In Sections 4–6, we prove the main result; their respective roles are explained in the proof outline in Section 3. Readers willing to skip the technical details can safely skip these sections. In Section 7, we discuss nonarborescent sentences. We conclude with a discussion and a few suggestions for further research.

2 Definitions

For the formal exposure of our results, we restrict to the graph terminology, but occasionally also rephrase them in terms of automata networks.

Succinct graph representations (SGR). A graph G is said to be *succinctly represented* as a pair (N, C), where C is a Boolean circuit on $2n$ inputs and one output and N is an integer (encoded in binary) with $N \leq 2^n$, whenever there is a one-to-one labeling of the vertices of G onto $\{0, \cdots, N-1\}$ such that $C(x, y) = 1$ if and only if there is an edge from the vertex labeled x to the vertex labeled y. We denote $G = \mathcal{G}_{N,C}$, and we always assume that a Boolean circuit is not bigger than the adjacency matrix of the encoded graph (up to a polynomial factor), because such an encoding always exists.

MSO Logic. Given a SGR (N, C), we want to test Monadic Second-Order logic formulae over the graph $\mathcal{G}_{N,C}$. MSO formulae have two kinds of variables: vertices $(x_1, x_2, \dots)$ and sets of vertices $(X_1, X_2, \dots)$—Accordingly, there are two kinds of existential quantifiers: existence of a vertex, and existence of a set of vertices. The Boolean connectives $\neg$ and $\wedge$ are as usual. The universal quantifiers, bounded quantifiers and other Boolean connectives are derived from them. The atoms are: equality $(x_1 = x_2)$, membership $(x_1 \in X_1)$, and adjacency relation $E(x_1, x_2)$, meaning that $\mathcal{G}_{N,C}$ has an edge from x_1 to x_2. Note that the adjacency symbol E is a relational symbol in our signature, **not** a functional symbol.

A *sentence* is a closed monadic second-order formula, i.e., one where all variables are bound to a quantifier. Here are some examples, where $\mathcal{G}_{N,C}$ is seen as the graph of a non-deterministic dynamics (vertices are interpreted as configurations):

- *(Existence of a loop)* $\exists x : E(x, x)$.
- *(Unicity of the loop)* $\forall x, \forall x' : E(x, x) \wedge E(x', x') \implies x = x'$.
- *(Determinism)* $\forall x, \forall y, \forall y' : E(x, y) \wedge E(x, y') \implies y = y'$.
- *(Nontrivial cycle)* $\exists X, [\exists x \in X] \wedge [\forall x \in X, \exists y \in X : x \neq y \wedge E(x, y)]$.

With this MSO signature, it is possible to express as a macro the relation $E^*(x, y)$: "there exists a chain $x = z_1, z_2, \dots, z_{n-1}, z_n = y$ such that $E(z_k, z_{k+1})$ holds for every $1 \leq k \leq n-1$." On the other hand, it is not possible to express something like: "all configurations have the same out-degree." See [16] or [20] for more information about MSO logic in graphs.

Tree decompositions. A *tree decomposition* of a graph $G = (V, E)$ is a tree T whose nodes are labeled with subsets of V—called *bags*— satisfying the three conditions below. If p is a node of T, we write $B(p)$ for its label, i.e., the corresponding bag.

(i) Every node of G belongs to at least one bag.
(ii) For every edge (v_1, v_2) of G, at least one bag contains both v_1 and v_2.
(iii) For all nodes p, q, r of T, if q is on the (unique) path from p to r, then $B(p) \cap B(r) \subseteq B(q)$.

That definition is usually stated for undirected graphs, but it works without change for directed graphs. In other terms, we look at tree decompositions of the symmetric closures of the considered graphs. A graph has, in general, many different tree decompositions. The *width* of a decomposition is the size of its largest bag minus one. The *treewidth* of a graph is the minimal width among all of its tree decompositions. For every integer k, a *k-tree decomposition* means a tree decomposition of width k. In this paper, any tree (including tree decompositions) is regarded as rooted and oriented downwards: edges point away from root.

Additional conventions. If G is a graph, let $|G|$ denote the number of its nodes, dubbed its *size*. If S is an instance of SAT, let $|S|$ denote the number of its variables, also dubbed its *size*. If ϕ is an MSO sentence, its **quantifier rank** is its number of quantifiers (not the number of alternations). Unless stated otherwise: **increasing** means strictly increasing; **integer** means positive or zero integer; **polynomial** means nonconstant polynomial with integer coefficients.

3 Statement of the main result and proof outline

The problem. Given an MSO sentence ϕ, define the model checking problem of ϕ on graphs given by a succinct representation.

> SUCCINCT-ϕ
> *Input:* a succinct graph representation (N, C).
> *Output:* does $\mathcal{G}_{N,C} \models \phi$?

Observe that ϕ is *not* part of the input: it is considered constant. In other words, we have a family of problems parameterized by MSO sentences.

Our main result is a counterpoint to Courcelle's theorem:

Theorem 4. *If ϕ has infinitely many models with the same treewidth k_1 and infinitely many countermodels with the same treewidth k_2, then* SUCCINCT-ϕ *is either* NP*-hard or* coNP*-hard.*

An MSO sentence is *arborescent* if it satisfies the condition of Theorem 4. All examples of sentences given in the previous section are arborescent. We will discuss non-arborescent sentences in Section 7.

Proof outline. First, we show that there exists a "good" graph, call it Ω, such that $\Omega \sqcup G$ is always a model of ϕ (where $\sqcup$ denotes disjoint union), no matter what G is. Then, we show that there exists a "bad" graph, say Y, such that $Y \sqcup \cdots \sqcup Y$ is always a countermodel of ϕ, no matter how many disjoint copies of Y we put. We can arrange things so that Ω and Y have the same number of vertices.

Now we perform a reduction: we are given an instance S of SAT with s Boolean variables, and we produce a succinct graph representation (N, C) such that $\mathcal{G}_{N,C} \models \phi$ if and only if S has at least one positive assignment. We take $N = 2^s \cdot |\Omega| = 2^s \cdot |Y|$ so that we have 2^s groups of $|\Omega| = |Y|$ vertex labels. For each vertex label (n-bit string), the circuit C interprets the first s bits as an assignment of the variables of S, and evaluates S on that assignment. If it finds "true", then the corresponding $|\Omega|$ vertex labels realize a copy of Ω in $\mathcal{G}_{N,C}$. If it finds "false", then the corresponding $|Y|$ vertex labels realize a copy of Y instead. Consequently, $\mathcal{G}_{N,C}$ contains as many copies of Ω as positive assignments for S, and as many copies of Y as negative assignments for S. This completes the reduction: if there is at least one positive assignment, the defining property of Ω guarantees that the graph satisfies ϕ. Otherwise, the graph is only a pack of disjoint copies of Y, which does not satisfy ϕ.

This whole construction can be performed in polynomial time because Ω and Y do not depend on S: they only depend on ϕ, hence they are constants (recall that ϕ is *not* part of the input of the problem). The only part of C that depends on ϕ is the evaluation of S, but both are encoded as Boolean circuits, which is easy to implement.

Of course, things are not that simple. First problem: we are actually unable to control whether Ω (called a *saturating graph*) turns every graph into a model or every graph into

a countermodel. In the latter case, we will have to symmetrize all the remainder of the proof (in particular, $Y \sqcup \cdots \sqcup Y$ will have to be a model), and in that case we will get coNP-hardness instead of NP-hardness. On the other hand, it turns out that Ω does not depend on ϕ, but only on the quantifier rank of ϕ, which is pretty amusing. The details are explained in Section 4.

Second problem: we have to relax the requirements on Y. What we will actually get is a triple of graphs (X, Y, Z) such that $X \oplus Y \oplus \cdots \oplus Y \oplus Z$ is a countermodel of ϕ (or, if needed, a model of ϕ), no matter how many copies of Y are in there. The gluing operator $\oplus$ is more general than disjoint union; $G \oplus G'$ basically means: "take the disjoint union of G and G', but also merge some marked vertices of G with some marked vertices of G'". The details are explained in Section 5 (for notational convenience, X, Y, Z are called G_1, G_2, G_3 in that section—the subscripts come in handy).

Third problem: because of the concessions just made on Y, in the reduction from SAT described above, we have to account for the merged vertices (per definition of $\oplus$). This requires care, because we cannot allow any extraneous configuration in the graph: every single vertex has to belong to the (unique) copy of X, the (unique) copy of Z, some copy of Y, or some copy of Ω. The details are explained in Section 6, which also includes the final proof of Theorem 4.

Sections 4, 5 and 6 each start with a proposition, and the remainder of the section is the proof of the proposition. These three propositions together quickly yield a proof for Theorem 4.

4 A graph saturating all sentences of fixed quantifier rank

Proposition 5. *Fix $m \in \mathbb{N}$. There exists a graph Ω_m such that, for every MSO sentence ϕ of rank m, either:*

(i) for every graph G, we have $G \sqcup \Omega_m \models \phi$; or
(ii) for every graph G, we have $G \sqcup \Omega_m \not\models \phi$.

In the first case, we say that Ω_m is a *sufficient subgraph* for ϕ; in the second case, Ω_m is a *forbidden subgraph* for ϕ. A graph that is either sufficient or forbidden for a given sentence is called *saturating* for that sentence. The rest of this section is a proof of Proposition 5.

If G and G' are graphs, write $G \equiv_m G'$ if and only if G and G' satisfy exactly the same MSO sentences of quantifier rank m. Write $G \sqcup G'$ for the disjoint union of a copy of G and a copy of G'. If k is an integer, write $\bigsqcup^k G$ the disjoint union of k copies of G.

Lemma 6. *For every nonempty graph G, there exists an integer $q(G, m)$ such that $\bigsqcup^{q(G,m)} G \equiv_m \bigsqcup^{q(G,m)+1} G$.*

Proof. We show that there exists an integer $q(G, m_1, m_2)$ such that, in the MSO-Ehrenfeucht-Fraïssé game over the graphs

$$\gamma = \bigsqcup^{q(G,m_1,m_2)} G \qquad \text{and} \qquad \gamma' = \bigsqcup^{q(G,m_1,m_2)+1} G,$$

if Spoiler plays at most m_1 point moves and m_2 set moves, then Duplicator wins. (For more details about MSO-Ehrenfeucht-Fraïssé games, see Section 7.2 of [16].) To conclude the proof, it will suffice to set $q(G, m) = \max\{q(G, m_1, m_2) : m_1 + m_2 = m\}$. Reason by induction over m_2.

If $m_2 = 0$, then Spoiler only plays point moves. Set $q(G, m_1, 0) = m_1$. Since the game lasts m_1 turns, Spoiler touches at most m_1 copies of G in γ', thus they cannot point any difference with γ.

If $m_2 > 0$, then set:

$$\log_2 (q(G, m_1, m_2)) = |G| \cdot (q(G, m_1, m_2 - 1) + m_1 + m_2) .$$

Call γ_i (respectively γ'_i) the i^{th} copy of G in γ (respectively γ'), for $1 \leq i \leq q(G, m_1, m_2)$ (respectively $1 \leq i \leq q(G, m_1, m_2) + 1$). If Spoiler starts by playing a point move in γ_i, then Duplicator chooses the same vertex in γ'_i. The case of Spoiler playing in γ'_i is symmetric: up to reordering the γ'_i, we can assume that Spoiler never plays in $\gamma'_{q(G,m_1,m_2)+1}$. Duplicator continues this strategy as long as Spoiler plays point moves.

Now consider the first set move of Spoiler and suppose that it was in γ. Recall that we can assume no point move took place in $\gamma'_{q(G,m_1,m_2)+1}$. Spoiler just pointed a subset of vertices of γ, or equivalently pointed a subset of vertices of each γ_i separately. Call $V_1, \ldots, V_{2^{|G|}}$ all possible subsets of vertices of G and $f : \{1, \ldots, q(G, m_1, m_2)\} \to \{1, \ldots, 2^{|G|}\}$ the function such that for each γ_i, Spoiler pointed $V_{f(i)}$. For $1 \leq i \leq q(G, m_1, m_2)$, Duplicator plays in γ'_i the set $V_{f(i)}$, i.e., the same set of vertices as Spoiler played in γ_i. It remains to play a set of vertices for $\gamma'_{q(G,m_1,m_2)+1}$: choose n such that $|f^{-1}(n)| > q(G, m_1, m_2 - 1)$ and play V_n.

Call f' the function such that, for every i, Duplicator played $V_{f'(i)}$ in γ'_i. To finish the game, Duplicator actually plays $2^{|G|}$ games in parallel, one in each couple of graphs $(f^{-1}(n), f'^{-1}(n))$ for $1 \leq n \leq 2^{|G|}$. All those couples of graphs either have the same number of copies of G inside, or both have more than $q(G, m_1, m_2 - 1)$ copies of G, and Duplicator wins by induction.

Symmetrically, suppose that the first set move of Spoiler was in γ'. Up to reordering the γ'_i, we can suppose both that no point move took place in $\gamma'_{q(G,m_1,m_2)+1}$ and that $|f'(q(G, m_1, m_2) + 1)| > q(G, m_1, m_2 - 1)$. In γ_i, Duplicator plays the set $V_{f(i)}$. To finish the game, Duplicator actually plays $2^{|G|}$ games in parallel, one in each couple of graphs $(f^{-1}(n), f'^{-1}(n))$ for $1 \leq n \leq 2^{|G|}$. Once again all those couples of graphs either have the same number of copies of G inside, or both have more than $q(G, m_1, m_2 - 1)$ copies of G, so Duplicator wins by induction. □

Lemma 7 (Proposition 7.5 from [16])**.** *For every m, the relation $\equiv_m$ has finitely many equivalence classes.*

Call $a(m)$ the number of classes of $\equiv_m$ and let $A_1, \ldots, A_{a(m)}$ denote representatives of each class.

Proof of Proposition 5. Define Ω_m as follows: take $q(A_i, m)$ disjoint copies of A_i, for i ranging in $\{1, \ldots, a(m)\}$. We show that Ω_m is either a forbidden or a sufficient subgraph for ϕ. for every graph G, we have $G \equiv_m A_i$ for one i in $\{1, \ldots, a(m)\}$. Hence $\Omega_m \sqcup G \equiv_m \Omega_m \sqcup A_i$: in the MSO-Ehrenfeucht-Fraïssé game between those graphs, Duplicator can apply its winning strategies separately in Ω_m and in G/A_i. Consequently, adding G to Ω_m is equivalent to adding a $[q(A_i, m) + 1]^{\text{th}}$ copy of A_i to Ω_m. By definition of q in Lemma 6, the resulting graph is equivalent to Ω_m. As a conclusion, $\Omega_m \equiv_m \Omega_m \sqcup G$, and whether it is a model of ϕ or not does not depend on G. □

5 Pumping models of arborescent sentences

A *boundaried graph* is a digraph G endowed with an additional specific sequence $P = (p_0, \ldots, p_{\ell-1})$ of distinct nodes of G, called *ports*. The set of vertices, of edges and of ports of a boundaried graph G are denoted by $V(G)$, $E(G)$ and $P(G)$ respectively. If G and G' denote boundaried graphs with the same number of ports, then $G \oplus G'$ ("G glued to G'") is defined as the graph $G \sqcup G'$ where the i^{th} port of G is merged with the i^{th} port of G'.

A *ℓ-biboundaried graph*, or ℓ-graph for short, is a digraph G endowed with two sequences of ℓ ports: the *primary ports*, denoted by $P_1(G)$, and the *secondary ports*, denoted by $P_2(G)$. (So we have $|P_1(G)| = |P_2(G)| = \ell$.) For two such graphs G and G', write $G \oplus G'$ for $G \sqcup G'$ where $P_2(G)$ is identified with $P_1(G')$. Moreover, set $P_1(G \oplus G') = P_1(G)$ and $P_2(G \oplus G') = P_2(G')$. Note that $P_1(G)$ and $P_2(G)$ may intersect. Now let $\Gamma = \{G_i\}_{i \in I}$ be a finite family of ℓ-graphs; if w is a nonempty word over alphabet I, then define $\Delta^\Gamma(w)$ by induction over the length of w as follows:

$$\Delta^\Gamma(w_1) = G_{w_1}, \qquad \Delta^\Gamma(w_1 \ldots w_n) = \Delta^\Gamma(w_1 \ldots w_{n-1}) \oplus G_{w_n}.$$

See Figure 1—although the possibility that P_1 and P_2 intersect is not shown on that figure. We suppose that all nodes of all graphs constructed below are taken from a linearly ordered infinite set, so that we can specify ports of ℓ-graphs be just specifying sets of nodes, that will always be ordered according to that global order. Thus, to simplify notations, we often treat $P_1(G)$ and $P_2(G)$ as sets.

Proposition 8. *Let ϕ denote an MSO sentence and k an integer. If ϕ has infinitely many models of treewidth k, then there exists a triple of ℓ-graphs $\Gamma = \{G_1, G_2, G_3\}$ for some $\ell \leq k+1$ with $P_1(G_1) \cap P_2(G_1) \neq V(G_1)$ and such that $\Delta^\Gamma(2 \cdot 1^n \cdot 3)$ is a model of ϕ for every integer n.*

For the remainder of this section, we prove Proposition 8. Fix a sentence ϕ and an integer k.

Definition 9. Let G, G' be ℓ-graphs for some $\ell \leq k+1$. We write $G \sim G'$ if and only if for every ℓ-graph H, we have:

$$G \oplus H \models \phi \iff G' \oplus H \models \phi.$$

Of course this depends on ϕ and k, but they have been fixed for this section. Relation $\sim$ only holds among ℓ-graphs for the same number of ports ℓ (which will be bags of a k-tree decomposition, hence of size $\ell \leq k+1$). Call Σ the set of equivalence classes of $\sim$ for all $\ell \leq k+1$.

Lemma 10 (Theorem 13.1.1 from [20])**.** *The set Σ of equivalence classes of $\sim$ is finite.*

Definition 11. Let G denote a graph with a k-tree decomposition T. For a node v of T, let $\mathcal{S}(v)$ denote the largest subtree of T rooted in v. Call $\mathcal{N}(v)$ the boundaried graph consisting of the subgraph of G spanned by $\bigcup_{u \in \mathcal{S}(v)} B(u)$ and whose set of ports is $B(v)$. Finally, call $\mathcal{C}(v)$ the equivalence class of $\mathcal{N}(v)$ for the relation $\sim$. When the tree to which v belongs is unclear, we write $\mathcal{S}_T(v)$, $\mathcal{N}_T(v)$ and $\mathcal{C}_T(v)$ to specify it.

Observe that any tree decomposition of any graph can be viewed as a Σ-labeled tree: label each node v with $\mathcal{C}(v)$ instead of a bag. We say that the Σ-labeled tree *corresponds to* the tree decomposition. The following remark is an immediate consequence of the definition of $\sim$.

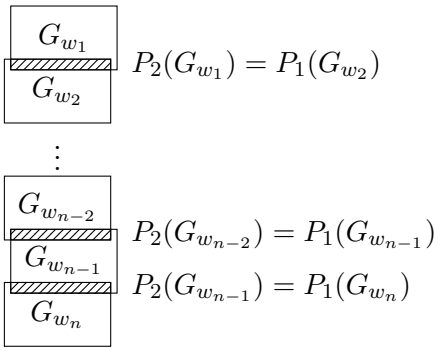

Figure 1: Illustration of Δ

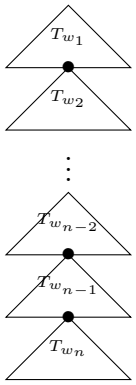

Figure 2: Illustration of Λ

Remark 12. Let G denote a graph with a k-tree decomposition T, and v the root of T. Whether $G \models \phi$ or not depends only on $\mathcal{C}_T(v)$.

A Σ-labeled tree is *positive* if it corresponds to the decomposition of a model of ϕ. A given Σ-labeled tree might correspond to decompositions of several different graphs, but Remark 12 ensures that they are either all models, or all countermodels. The next definition applies both to tree decompositions and to Σ-labeled trees.

Definition 13. Let T denote a tree, v a node of T, and T' another tree. Write $T[\mathcal{S}(v) \leftarrow T']$ for the tree obtained by replacing the largest subtree of T rooted in v with T'. If v is a leaf of T, then write $T \oplus_v T'$ for the tree $T[\mathcal{S}(v) \leftarrow T']$. When no confusion arises, we may drop the subscript and write $T \oplus T'$.

Now let $\mathcal{T} = \{T_i\}_{i \in I}$ denote a finite family of trees, each with a pointed leaf, and w a finite, nonempty word over alphabet I. Define $\Lambda^{\mathcal{T}}$ by induction over the length of w:

$$\Lambda^{\mathcal{T}}(w_1) = T_{w_1}, \qquad \Lambda^{\mathcal{T}}(w_1 \dots w_n) = \Lambda^{\mathcal{T}}(w_1 \dots w_{n-1}) \oplus T_{w_n},$$

where the pointed leaf of the result is inherited from T_{w_1} or T_{w_n}, respectively. See Figure 2.

Lemma 14. *Let G denote a model of ϕ with a k-tree decomposition T, and H a graph with a k-tree decomposition U whose root is called u. If v is a node of T such that $\mathcal{C}_T(v) = \mathcal{C}_U(u)$ then $T[\mathcal{S}(v) \leftarrow U]$, viewed as a Σ-labeled tree, is positive.*

Proof. Let T_1 denote $T \backslash (\mathcal{S}_T(v) - \{v\})$ and T_2 denote $\mathcal{S}_T(v)$. Call G_1 and G_2 the subgraphs of G spanned by the nodes in all the bags of T_1 and T_2, respectively, and set $P(G_1) = P(G_2) = B(v)$. Observe that $G = G_1 \oplus G_2$. Make H a boundaried graph by setting $P(H) = B(u)$. The relation $\mathcal{C}_U(u) = \mathcal{C}_T(v)$ implies that $\mathcal{N}_U(u) \sim \mathcal{N}_T(v)$, in other terms $H \sim G_2$, so by definition of $\sim$ we have $G = G_1 \oplus G_2 \models \phi \iff G_1 \oplus H \models \phi$ (recall that $\oplus$ is symmetric on boundaried graphs). Hence $G_1 \oplus H$ is a model of ϕ. Observe that $T[\mathcal{S}(v) \leftarrow U]$ corresponds to a tree decomposition of $G_1 \oplus H$ and the lemma is proved. $\square$

Remark 15. Let $\Gamma = \{G_i\}_{i \in I}$ a collection of ℓ-graphs and $\mathcal{T} = \{T_i\}_{i \in I}$ a collection of Σ-labeled trees, both indexed by the same finite set I. Suppose that for every i, the tree T_i corresponds to a decomposition of G_i such that the root's bag is $P_1(G_i)$ and there is a leaf whose bag is $P_2(G_i)$. Make that leaf the pointed leaf of T_i. Then, for every nonempty word w over alphabet I, the tree $\Lambda^{\mathcal{T}}(w)$ corresponds to a decomposition of $\Delta^{\Gamma}(w)$. As an illustration of this remark, we could say that the bags of the bold nodes in Figure 2 are the nodes in hatched areas in Figure 1.

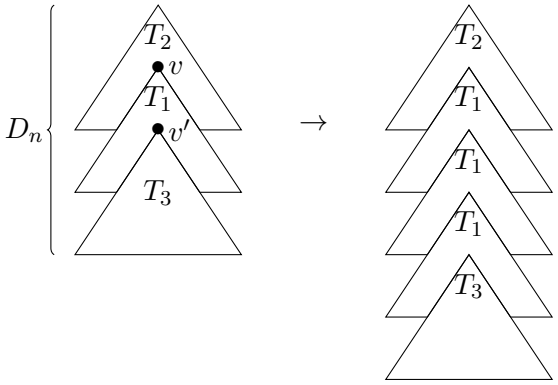

Figure 3: Proof of Proposition 8.

Recall that in a digraph (and in particular in a tree), the degree of a node is the sum of its in-degree and its out-degree.

Lemma 16. *If a graph G has a k-tree decomposition with n nodes, then it has a k-tree decomposition of degree 3 with at least n nodes.*

Proof. Split any node with more than 3 neighbors in the tree into a chain of nodes. □

Proof of Proposition 8. The sentence ϕ has models with k-tree decompositions having arbitrarily high numbers of nodes. By Lemma 16, there is a sequence $(M_i)_{i\in\mathbb{N}}$ of models and a sequence $(D_i)_{i\in\mathbb{N}}$ of respective k-tree decompositions of degree 3. Choose each D_i to be minimal (in the number of nodes among k-tree decompositions of degree 3 of M_i) and view the D_i's as Σ-labeled trees. The D_i's are unbounded since M_i's are unbounded. By Lemma 10 the set Σ of values for $\mathcal{C}(\cdot)$ is finite; so any large enough Σ-labeled path contains two nodes with the same label. In particular, for large enough n, D_n contains a path with two nodes v and v' such that $\mathcal{C}(v) = \mathcal{C}(v')$. Suppose without loss of generality that v has lesser depth than v' and let (see Figure 3):

$$T_1 = \mathcal{S}(v) \setminus (\mathcal{S}(v') - \{v'\}), \qquad T_2 = D_n \setminus (\mathcal{S}(v) - \{v\}), \qquad T_3 = \mathcal{S}(v').$$

Define the graphs G_1, G_2 and G_3 as the induced subgraphs of M_n spanned by all the bags of T_1, T_2 and T_3, respectively. Set $P_2(G_2) = P_1(G_1) = B(v)$ and $P_2(G_1) = P_1(G_3) = B(v')$; choose arbitrarily $P_1(G_2)$ and $P_2(G_3)$. Note that $P_1(G_1) \cap P_2(G_1) \neq V(G_1)$ otherwise it would mean that all the bags of T_1 are identical to $B(v) = B(v')$, but then we could obtain a strictly smaller tree decomposition of M_i by replacing T_1 with a single node with a single node of bag $B(v)$ thus contradicting the minimality of D_i. Let $\Gamma = (G_1, G_2, G_3)$ and $\mathcal{T} = (T_1, T_2, T_3)$, so that by Remark 15, the tree $\Lambda^{\mathcal{T}}(2 \cdot 1^m \cdot 3)$ corresponds to a decomposition of $\Delta^{\Gamma}(2 \cdot 1^m \cdot 3)$ for every m. Since $\mathcal{C}(v) = \mathcal{C}(v')$, Lemma 14 implies that for every integer m, the tree $\Lambda^{\mathcal{T}}(2 \cdot 1^m \cdot 3)$ is positive. Therefore $\Delta^{\Gamma}(2 \cdot 1^m \cdot 3)$ is a model of ϕ. □

Remark 17. Note that in Proposition 8, graphs G_1, G_2 and G_3 are constant so the family of graphs of the form $\Delta^{\Gamma}(2 \cdot 1^n \cdot 3)$ for $n \geq 1$ is of bounded pathwidth (see [14] for definition and background). Therefore it holds that an MSO formula has infinitely many models among bounded treewidth graphs if and only if it does among bounded pathwidth ones.

6 A reduction from Succinct-ϕ to SAT

In all this section, for any biboundaried graph G, assume that $V(G) = \{0, \ldots, |G| - 1\}$ and for a node u of G, write $G(u)$ the set $\{v : (u, v) \in E(G)\}$. Thus, $P_1(G)$, $P_2(G)$ and $G(u)$ are all subsets of $\{0, \ldots, |G| - 1\}$. We present our construction is a separate statement.

Definition 18. If S is an instance of SAT with s variables, then $\overline{S}$ is the word of length 2^s such that $\overline{S}_i$ is 1 if $S(i)$ is false, and 0 if $S(i)$ is true (viewing the binary expansion of i as a Boolean assignment for S).

Proposition 19. *Fix an MSO sentence ϕ. Let k be an integer and $\Gamma = \{G_0, G_1, G_2, G_3\}$ four k-graphs satisfying the following conditions:*

(i) $|G_0| = |G_1|$;
(ii) $P_1(G_0) \cap P_2(G_0) = P_1(G_1) \cap P_2(G_1) \neq V(G_1)$; *and*
(iii) for every p in $P_1(G_0) \cap P_2(G_0)$, we have $G_0(p) = G_1(p)$.

Suppose that, for every word w over alphabet $\{0, 1\}$, we have $\Delta^\Gamma(2 \cdot w \cdot 3) \models \phi$ if and only if w contains letter 0. Then Succinct-ϕ *is* NP*-hard.*

The main tool is the following lemma.

Lemma 20. *Let S be an instance of SAT with s variables, k an integer and $\Gamma = \{G_0, G_1, G_2, G_3\}$ four k-graphs satisfying conditions (i)–(iii) from Proposition 19. Then, there is a succinct graph representation (N, C) such that*

$$\mathcal{G}_{N,C} = \Delta^\Gamma(2 \cdot \overline{S} \cdot 3),$$

that can be computed in polynomial time given S (the G_i are considered constant).

Proof. If G is a biboundaried graph, define:

$$P'_1(G) = P_1(G) \setminus P_2(G), \qquad P'_2(G) = P_2(G) \setminus P_1(G), \qquad P'_3(G) = P_1(G) \cap P_2(G).$$

The rationale is that we will need to handle ports that are in $P_1(G) \cap P_2(G)$ as a special case later. By Conditions (i) and (ii), we have $|P'_1(G_1)| = |P'_1(G_0)| = |P'_2(G_1)| = |P'_2(G_0)|$; call k' that quantity. We also have $|P'_3(G_1)| = |P'_3(G_0)|$; call k'' that quantity. Observe that $k = k' + k''$.

Recall that for any k-graph G, we assume that $V(G) = \{0, \ldots, |G|-1\}$. Intuitively, we want to arrange things so that $P_1(G)$ is at the beginning of the interval (i.e., $\{0, \ldots, k-1\}$) and $P_2(G)$ at the end of the interval (i.e., $\{|G| - k, \ldots, |G| - 1\}$), so that it becomes easy to merge vertices when computing a gluing ($\oplus$) operation—see Figure 4. Things are unfortunately not that simple: we need to account for the possibility that $P_1(G) \cap P_2(G) \neq \emptyset$. Thus we will put P'_1 at the beginning, P'_3 at the end, and P'_2 just before P'_3. The only exception is in G_3: we put P'_3 just after P'_1, for reasons that we will explain later on. Formally speaking, for $i = 0, 1, 2$, assume without loss of generality that:

$$\begin{aligned}
P'_1(G_i) &= (0, \ldots, k' - 1), & P'_1(G_3) &= (0, \ldots, k' - 1),\\
P'_2(G_i) &= (|G_i| - k, \ldots, |G_i| - k'' - 1), & P'_2(G_3) &= (|G_3| - k', \ldots, |G_3| - 1),\\
P'_3(G_i) &= (|G_i| - k'', \ldots, |G_i| - 1), & P'_3(G_3) &= (k', \ldots, k - 1).
\end{aligned}$$

Define the quantities:

$$n_1 = |G_1| - k = |G_0| - k, \qquad n_2 = |G_2| - k, \qquad n_3 = |G_3|.$$

Note that $n_1 > 0$ by Condition (ii). We set $N = n_2 + 2^s \cdot n_1 + n_3$ so that the vertex labels are $\{0, \dots, n_2 + 2^s \cdot n_1 + n_3 - 1\}$: see Figure 4 for a picture of its organization. The initial segment is the (unique) copy of G_2, followed by 2^s copies of G_0 or G_1, followed by the (unique) copy of G_3. Those copies overlap to account for the merged vertices between all those graphs. Let $\ell = 2^s - 1$ be the index of the last copy of G_0 or G_1. For q in $\{-1, 0, \dots, 2^s\}$, define (see the following paragraph for intuition):

$$\delta_0^q(r) = \delta_1^q(r) = \begin{cases} n_2 + q \cdot n_1 + r & \text{if } r \in \{0, \dots, |G_1| - k'' - 1\}, \\ n_2 + \ell \cdot n_1 + r & \text{if } r \in \{|G_1| - k'', \dots, |G_1| - 1\}; \end{cases} \tag{1}$$

$$\delta_2^q(r) = \begin{cases} r & \text{if } r \in \{0, \dots, |G_2| - k'' - 1\}, \\ \delta_1^\ell(r) + |G_1| - |G_2| & \text{if } r \in \{|G_2| - k'', \dots, |G_2| - 1\}; \end{cases} \tag{2}$$

$$\delta_3^q(r) = n_2 + 2^s \cdot n_1 + r. \tag{3}$$

The functions δ map integers to integers, but we implicitly extend them to sets of integers, elementwise. Intuitively, $\delta_j^q(r)$ refers to the r^{th} node of a copy of G_j in Figure 4, i.e., as a vertex label of $\mathcal{G}_{N,C}$. There is only one copy of G_2 and one of G_3, so if $j = 2, 3$, this is unambiguous. But there are many (precisely 2^s) copies of G_0 and G_1, and whether a given graph is G_0 or G_1 will change according to S; therefore, the superscript q is used to specify which copy of G_0 (or G_1) we are targeting. It will be convenient to write δ_2^q and δ_3^q even though they do not depend on q.

Now the only difficulty resides in the k'' ports common to all occurrences of G_0 and G_1—which are also common to the copy of G_2 and the copy of G_3, by construction. They are "physically" located after the last copy of G_0 or G_1, which has index ℓ. There is a special case in Equations (1) and (2) to redirect any edges going into these nodes to the proper location. The graph G_3 is organized so that elements of $P_3'(G_3)$ are already at the right place. The edges going out of those nodes are handled later on.

We are now ready to give the global description of $\mathcal{G}_{N,C}$ in terms of the adjacency relation, i.e., the set of vertex labels corresponding to the out-neighbors of a given vertex label. We define C' a function mapping vertex labels to sets of vertex labels.

- If $x < n_2$, then set:

$$C'(x) = (\delta_2^0 \circ G_2)(x). \tag{4}$$

- If $x - n_2 < 2^s \cdot n_1$, then by Euclidean division by n_1 (recall $n_1 \neq 0$) let q, r be such that $x - n_2 = q \cdot n_1 + r$, with $r \in \{0, \dots, n_1 - 1\}$. Let i and j denote $\overline{S}(q-1)$ and $\overline{S}(q)$ respectively. However if $q = 0$, then set $i = 2$ instead. Set:

$$C'(x) = \begin{cases} (\delta_j^q \circ G_j)(r) \cup (\delta_i^{q-1} \circ G_i)(r + n_i) & \text{if } r \in \{0, \dots, k' - 1\}, \\ (\delta_j^q \circ G_j)(r) & \text{otherwise.} \end{cases} \tag{5}$$

- Otherwise, $n_2 + 2^s \cdot n_1 \leq x$, so let $q = 2^s$ and $r = x - (n_2 + 2^s \cdot n_1)$. Let $i = \overline{S}(2^s - 1)$, and set:

$$C'(x) = \begin{cases} (\delta_3^0 \circ G_3)(r) \cup (\delta_i^{q-1} \circ G_i)(r + n_1) & \text{if } 0 \leq r < k', \\ (\delta_3^0 \circ G_3)(r) \cup (\delta_2^0 \circ G_2)(r + n_2) \cup \bigcup_{t=0}^{\ell} (\delta_1^t \circ G_1)(r + n_1) & \text{if } k' \leq r < k, \\ (\delta_3^0 \circ G_3)(r) & \text{otherwise.} \end{cases} \tag{6}$$

Let C be a Boolean circuit such that $C(x, y) = 1$ if and only if $y \in C'(x)$. Observe that $\mathcal{G}_{N,C}$ is, by construction, $\Delta^\Gamma(2 \cdot \overline{S} \cdot 1^c \cdot 3)$. Assume, for the sake of the discussion, that $S(0)$

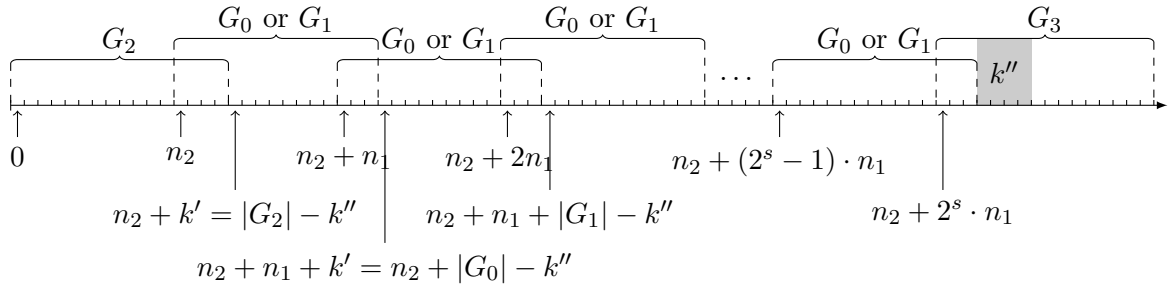

Figure 4: Proof of Lemma 20, a representation of $\{0, \ldots, n_2 + 2^s \cdot n_1 + n_3 - 1\}$ as the set of vertex labels of $\mathcal{G}_{N,C}$. The gray area (k'') is actually common to all the graphs.

evaluates to true and $S(1)$ to false, so the first two graphs after G_2 are G_0, then G_1. Those two graphs share k' vertices that they do not share with any other G_j in the construction. Those vertices are "physically" located at nodes $\{0, \ldots, k' - 1\}$ of the copy of G_1. The edges from G_0 going into those shared nodes naturally point correctly, because the last k' nodes of G_0 are identified with the first k' nodes of G_1 (both are, by definition, encoded by vertex labels $n_2 + n_1 + \{0, \ldots, k' - 1\}$—see the overlaps in Figure 4). The edges going out of those shared nodes into G_0 are realized by the first case in Equation (5). The same reasoning goes for $S(1)$ and $S(2)$, and so on.

As explained previously, there are k'' nodes that are shared by all the copies of G_0, G_1, G_2, G_3. The edges going into those nodes were handled in the definition of δ_j^q; the edges going out of those nodes are handled by the second case of Equation (6). That case assumes that all copies of G_0 and G_1 in the graph are in fact copies of G_1; this yields the desired results by Condition (iii).

Given S, it is straightforward to construct such a Boolean circuit C in polynomial time from the description above, because the G_i are considered constant, and the only part that depends on S merely evaluates S. Moreover, each vertex label requires at most two evaluations of S to compute its set of successors through C' (in Equation 5, for the values of i and j). This concludes the proof. □

We are now equipped to prove Proposition 19, leading to Theorem 4.

Proof of Proposition 19. Immediate consequence of Lemma 20: given an instance S of SAT, compute in polynomial time the corresponding succinct graph representation (N, C), and ask Succinct-ϕ on $\mathcal{G}_{N,C}$. □

Proof of Theorem 4. Let ϕ denote an arborescent MSO sentence with m quantifiers. By Proposition 5, the graph Ω_m is either forbidden or sufficient for ϕ. Assume that Ω_m is sufficient for ϕ and prove that Succinct-ϕ is NP-hard. (If it was forbidden, then consider $\neg\phi$ instead—this would yield that Succinct-ϕ is coNP-hard.) By Proposition 8 applied to $\neg\phi$, there exist a triple of graphs $\Gamma = (G_1, G_2, G_3)$ such that $\Delta^\Gamma(2 \cdot 1^n \cdot 3)$ is a model of $\neg\phi$ for every integer n. Let H denote the subgraph of G_1 spanned by:

$$[P_1(G_1) \cap P_2(G_1)] \cup \{j : \exists i \in P_1(G_1) \cap P_2(G_1), (i, j) \in E(G_1)\}$$

and $G_0 = H \sqcup \Omega_m$ (see Figure 5). Let $G_1' = \bigoplus_{i=1}^n G_1$, where n is the smallest integer such that $|G_1'| \geq |G_0|$. Finally, let G_0' denote G_0 with enough isolated nodes added so that $|G_0'| = |G_1'|$. It is possible to reorder the nodes of G_0', G_1' and add ports so that the quadruple $\Gamma' = (G_0', G_1', G_2, G_3)$ satisfies all the conditions of Proposition 19 (same as Lemma 20). This concludes the proof of the theorem. □

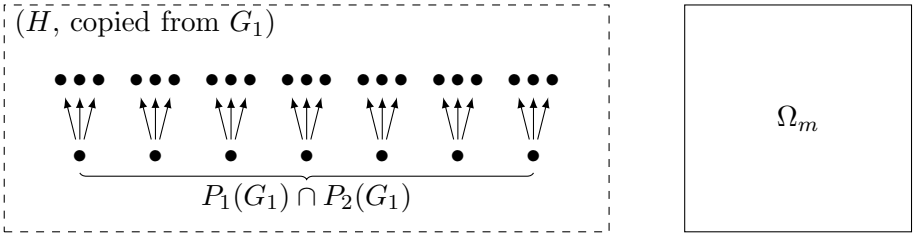

Figure 5: Representation of G_0 in the proof of Theorem 4.

7 Nonarborescent formulae

First, if ϕ is an MSO sentence with finitely many models or with finitely many countermodels, we can test ϕ in constant time (recall our convention that Boolean circuits are never bigger than the encoded graph). We call such sentences **trivial**. Note that no trivial MSO sentence is also arborescent.

In this section, we explore what focus on *nonarborescent*, yet *nontrivial* sentences, when neither Theorem 4, nor the above remark can be applied.

For instance, the question "is $\mathcal{G}_{N,C}$ a clique?" is expressible in first order, hence in MSO, but is both nonarborescent and nontrivial. It is easy to show that this question is coNP-hard: given an instance S of SAT with s variables, construct a circuit for a graph on $N = 2^s$ vertices that views n-bit vertex labels as an assignment for S and evaluates it. If it finds "false", then each bit can nondeterministically become either 0 or 1. If it finds "true", then all bits are required to stay in the same state. The succinctly represented graph is a clique if and only if all assignments for S evaluate to false. Producing Boolean circuits which evaluate a given instance of SAT is easily done in polynomial time, which concludes the reduction.

It is tempting to conjecture that for every nontrivial MSO formula ϕ, the problem Succinct-ϕ is either NP- or coNP-hard. The next theorem shows that this conjecture is unlikely to hold.

Definition 21 ([22, Definition 1])**.** A set M of integers is **robust** if and only if:

$$\forall k, \exists \ell \geq 2 : \{\ell, \ell + 1, \ldots, \ell^k\} \subset M.$$

This implies that M is infinite.

We denote by UNSAT the complement of SAT (*i.e.* CNF formulae that are not satisfiable).

Theorem 22. *There is a nontrivial first-order sentence ψ such that, if either SAT or UNSAT reduces to* Succinct-ϕ*, then there is a polynomial-time algorithm solving SAT for a robust set of sizes of instances.*

According to [22, Proposition 8, Theorem 12], our Theorem 22 implies:

Corollary 23. *Let ψ be given by Theorem 22. If either SAT or UNSAT reduces to* Succinct-ϕ*, then any problem in the polynomial hierarchy can be solved in polynomial time on a robust set of sizes of instances.*

The rest of this section is a proof of Theorem 22. For now, let ψ be an arbitrary first-order sentence, and f a polynomial reduction from either SAT or UNSAT to Succinct-ϕ. If S is a SAT (or UNSAT) instance, denote by $G_{f(S)}$ the graph succinctly represented by $f(S)$.

Definition 24. Let P be a polynomial and n a positive integer. The reduction f is **P-meager in n** if and only if for every *positive* instance S of SAT with size n, we have:

$$|G_{f(S)}| \leq P(n).$$

The set of values in which f is P-meager is called the **P-meagerness set** of f.

If f is P-meager in some integer n, then any SAT instance S of size n such that $G_{f(S)}$ has more than $P(n)$ vertices must be a negative instance. On the other hand, if $G_{f(S)}$ is smaller than $P(n)$, then we can test ψ on $G_{f(S)}$ in time polynomial in $|G_{f(S)}|$ (the degree of the polynomial being the quantifier rank of ψ, recall that ψ is a first-order sentence) and therefore in time polynomial in n. Thus we have the following lemma.

Lemma 25. *Assume that f is a reduction as before and let P be a polynomial. There is a polynomial-time algorithm that solves SAT on all instance sizes that are in the P-meagerness set of f.*

Proof. Given an instance S of SAT with size n, compute the network $f(S)$ in polynomial time; call G its transition graph. We can compute the size of G from $f(S)$ in polynomial time, because we have the list of nodes and the state set of each node. If $|G| > P(n)$, then return 'false': if f is not P-meager in n then the answer doesn't matter, and if f is P-meager in n then it means that S is a negative instance of SAT. Otherwise compute G itself, which can be done in polynomial time since $|G| \leq P(n)$, and evaluate ψ on G, which can also be done in polynomial time (the final answer is negated if reduction f is from UNSAT instead of SAT). □

Lemma 26. *Let f be a reduction as before and P be a polynomial. If f has a nonrobust P-meagerness set then, for every integer $d \geq 1$, either:*

(i) f produces a model of ψ whose size is not in $\mathbb{N}^{(d)} = \{n^d : n \in \mathbb{N}\}$; or
(ii) there exists an increasing primitive-recursive sequence μ such that $\mu(n)^d$ is the size of a model of ψ for each n.

Proof. By hypothesis there is an integer k such that, for every $\ell \geq 2$, there is at least one value among $\ell, \ell+1, \ldots, \ell^k$ in which f is not P-meager. Observe that if P' is the polynomial giving the execution time of f, then f produces networks whose transition graphs have size at most $2^{P'(n)}$ with n the size of the SAT or UNSAT instance. Let $t : \mathbb{N} \to \mathbb{N}$ be a primitive recursive function such that, for every n:

$$t(n+1) > \max\{2^{P'(t(n))}, t(n)^k\}. \tag{7}$$

Moreover since the meagerness set of f is nonrobust, we can additionally choose t such that f is not P-meager in $t(n)$, for every n. (Starting from a function t that satisfies Equation (7), given n, it is possible to enumerate all values $t(n), t(n)+1, \ldots, t(n)^k$ and to find the one for which f is not P-meager. That computation can be done in a primitive recursive fashion.)

For every n, let us pass to f a positive instance of SAT with size $t(n)$ (or a negative instance if reduction f is from UNSAT instead of SAT); the result is a sequence of automata networks, $(\alpha(n))_{n\in\mathbb{N}}$. This sequence can be made primitive recursive because t and f are. Call $\beta(n)$ the transition graph of $\alpha(n)$; since f is a reduction and we passed positive instances of SAT to it, the graph $\beta(n)$ is a model of ψ. By non-P-meagerness of f in $t(n)$, we can furthermore compute the positive instances of SAT such that:

$$P(t(n)) \leq |\beta(n)|,$$

in a primitive recursive fashion (simply by enumerating the satisfiable propositional formulae of size $t(n)$). Also, considering that P' is the running time of f, that a transition graph is at most exponential in the size of the network, and Equation (7), we have:

$$|\beta(n)| \leq 2^{P'(t(n))} < t(n+1).$$

Since $P(t(n)) \leq |\beta(n)| < P(t(n+1))$, the sequence $|\beta(n)|$ is increasing in n.

Moreover for every $d \geq 1$, there is an n_0 such that for every $n \geq n_0$, we have $|\beta(n+2)|^{1/d} > |\beta(n)|^{1/d}$ (as $|\beta(n+2)| > 2^{P'(|\beta(n)|)}$). If $|\beta(n)|^{1/d}$ is not an integer for some n then the reduction f produces a model of ψ whose size is not in $\mathbb{N}^{(d)}$ and the lemma follows. Otherwise, we define the map $\mu : \mathbb{N} \to \mathbb{N}$ by $\mu(n) = |\beta(2(n+n_0))|^{1/d}$; by the previous inequality μ is an increasing map. Finally, since $\alpha(n)$ is primitive recursive in n, so is $\beta(n)$ and so is $\mu(n)$. The lemma follows because by construction $\mu(n)^d$ is always the size of a model of ϕ. □

Definition 27. The **spectrum** of ψ is the set of sizes of models of ψ. In symbols:

$$\mathrm{Spec}(\psi) = \{|G| : G \models \psi\}.$$

See [19] for a survey about this notion.

A function $h : \mathbb{N} \to \mathbb{N}$ is **time-constructible** if and only if there is a Turing machine that, for every n, halts in exactly $h(n)$ steps on input n written in binary.

Lemma 28. *For every time-constructible function h, there exist a first-order sentence ψ_h and an integer d such that* $\mathrm{Spec}(\psi_h) = \mathrm{Im}(h)^{(d)}$, *where* $\mathrm{Im}(h)^{(d)} = \{n^d : n \in \mathrm{Im}(h)\}$.

Proof. By [5, Theorem 4.5] and [6, Theorem 3], we just have to prove that $\mathrm{Im}(h)$ is a NEXPTIME language. Given n, the algorithm guesses a word u of length at most n, runs h on input u and checks that it halts in n steps exactly. If $n \in \mathrm{Im}(h)$ then there is u such that $h(u) = n$ and $|u| \leq n$ because the machine cannot read more than n input symbols within n steps. □

Lemma 29. *There is a first-order sentence ψ and an integer d such that, for every increasing primitive-recursive function R, we have:*

$$\mathrm{Im}(R)^{(d)} \not\subseteq \mathrm{Spec}(\psi) \subseteq \mathbb{N}^{(d)}. \tag{8}$$

Proof. Let $(R_n)_{n\in\mathbb{N}}$ be a computable enumeration of increasing primitive-recursive functions. (To construct one, start from a computable enumeration of primitive-recursive functions $(R'_n)_{n\in\mathbb{N}}$ [13, Exercice I.7.4] and change R'_n into R_n as follows: $R_n : i \mapsto \max\{R_n(i-1)+1, R'_n(i)\}$. This transformation is computable and leaves increasing functions unchanged, so it hits all of them.)

For every integer n, define the set:

$$E(n) = \{R_i(j) : 0 \leq i, j \leq n\}, \tag{9}$$

and let h denote an increasing time-constructible function such that:

$$h(n) > \max E(n), \tag{10}$$

for instance, h may explicitly compute $\max E(n)$ and spend that many steps idling by decreasing a counter.

Let us show that for every n, there exists an element in $\mathrm{Im}(R_n) \setminus \mathrm{Im}(h)$. The set $\{h(0), \ldots, h(n-1)\}$ cannot contain $\{R_n(0), \ldots, R_n(n)\}$ because R_n is injective (since it is increasing). So there is an element i of $\{0, \ldots, n\}$ such that $R_n(i)$ does not belong to $\{h(0), \ldots, h(n-1)\}$. We have $R_n(i) < h(n)$ by Equations (9)–(10); since h is increasing, $R_n(i)$ is not in $\mathrm{Im}(h)$. The existence of desired formula ψ follows from Lemma 28. □

Proof of Theorem 22. Let ψ and d be given by Lemma 29 and assume that f is a polynomial reduction from either SAT or UNSAT to Succinct-ψ. Both the spectrum of ψ and its complement are infinite, so ψ is nontrivial. If there exists some polynomial P such that f has a robust meagerness set, then by Lemma 25 there exists a polynomial-time algorithm solving SAT on a robust set of instance sizes. Otherwise, since $\mathrm{Spec}(\psi) \subseteq \mathbb{N}^{(d)}$, Lemma 26 implies that there is an increasing primitive-recursive map μ such that, for every n, the quantity $\mu(n)^d$ is the size of a model of ψ. But then, by Lemma 29, one of those sizes will not be contained in the spectrum of ψ: a contradiction. □

Discussion and Future work

One could consider different parametrizations of Succinct-ϕ. Let us first point out that parametrization by the size or quantification rank of the formula fails. The formula $\exists x : E(x, x)$ expresses the property: "the graph has at least one loop" (or the transition graph has at least one fixed point, in terms of automata networks). That question is NP-hard: given an instance S of SAT, produce a succinct graph representation $(2^{|S|}, C)$ with labels on $|S|$ bits that evaluates it on S; if it finds "true", then the only edge is to itself; if it finds "false", then the only edge is to the lexicographic next label (cyclically). The formula above is virtually the smallest possible by any reasonable parameterization of logic formulae, so there is no hope to get fixed-parameter tractability if the parameter concerns the formula. Moreover, in this reduction, every (succinctly represented) graph produced is either a huge cycle or a sequence of paths ending in loops. This implies that they all have treewidth 2, so parameterization by the treewidth of graphs also fails: we have NP-difficulty even when the treewidth is guaranteed to be at most 2.

Another relevant parameter, when taking the automata networks point of view, is the size of the alphabets Q_v used at each automaton. For instance, as it is usually the case in the automata networks literature, one could consider MSO model checking on transition graphs of automata networks where all nodes share the same alphabet of size q. These must have q^n vertices, for some integer n. Our reduction in Lemma 20 fails in this settings, as it produces transition graphs $(\mathcal{G}_{N,C})$ whose number of vertices may not be a power of q. We do not know whether our main theorem holds for a fixed alphabet, but it is interesting to note that some formulae become trivial when fixing the alphabet: for instance, a formula asking that each configuration belong to a cycle of length two has no model with a ternary alphabet.

On the other hand, it remains to fully characterize which nonarborescent MSO sentences yield an NP- or coNP-hard problem, and what happens with those that do not. In particular, Theorem 22 does not say whether the formula ϕ has a polynomial-time solvable Succinct-ϕ problem. We do not know whether it could be the case for some ϕ under reasonable complexity assumptions. From our construction, we can nevertheless deduce that an arborescent MSO sentence ϕ is either proven to be NP-hard when $\Omega_m \models \phi$, or to be coNP-hard when $\Omega_m \not\models \phi$.

It would be interesting to consider probabilistic algorithms, e.g. showing that "draw a configuration at random and check whether it is a fixed point" cannot be a good algorithm to find fixed points in general. On another point of view, it could be interesting to obtain general hardness results on *probabilistic* automata networks, for example by considering circuits succintly computing a probability on graphs.

Finally, let us recall that the main result of this paper does not hold when restricting to some natural families of automata networks and using other input representation than circuits: for instance threshold automata networks or automata networks with bounded

degree can be naturally described as labeled graphs. In such cases, arborescent formulae can have a polynomial-time MSO model-checking problem, like "being a constant function". It would be interesting to understand what properties become tractable in these natural restrictions (which are often considered in the literature) and which fragment of MSO remains intractable.

Acknowledgments

The authors are very grateful to Édouard Bonnet (LIP, France) for the proof of Proposition 5, and to Aliénor Goubault–Larrecq for her suggestions on port sizes.

References

[1] B. Elspas. "The theory of autonomous linear sequential networks". In: *IRE Trans. Circ. Theory* 6 (1959), pp. 45–60.

[2] S. A. Kauffman. "Metabolic stability and epigenesis in randomly constructed genetic nets". In: *Journal of Theoretical Biology* 22 (1969), pp. 437–467.

[3] P. Cull. "Linear analysis of switching nets". In: *Biol. Cybernet.* 8 (1971), pp. 31–39.

[4] R. Thomas. "Boolean formalization of genetic control circuits". In: *Journal of Theoretical Biology* 42 (1973), pp. 563–585.

[5] N. Jones and A. Selman. "Turing Machines and the Spectra of First-Order Formulas". In: *J. Symb. Log.* 39.1 (1974), pp. 139–150.

[6] R. Fagin. "A spectrum hierarchy". In: *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 21 (1975), pp. 123–134.

[7] H. Galperin and A. Wigderson. "Succinct representations of graphs". In: *Information and Control* 56.3 (1983), pp. 183–198. DOI: 10.1016/s0019-9958(83)80004-7.

[8] C. H. Papadimitriou and M. Yannakakis. "A note on succinct representations of graphs". In: *Information and Control* 71.3 (1986), pp. 181–185. DOI: 10.1016/s0019-9958(86)80009-

[9] F. Robert. *Discrete Iterations: A Metric Study*. Springer Verlag, 1986.

[10] B. Courcelle. "The monadic second-order logic of graphs. I. Recognizable sets of finite graphs". In: *Information and Computation* 85.1 (1990), pp. 12–75.

[11] E. Goles and S. Martinez. *Neural and Automata Networks: Dynamical Behavior and Applications*. Kluwer Academic Publishers, 1990.

[12] J. L. Balcázar, A. Lozano, and J. Torán. "The complexity of algorithmic problems on succinct instances". In: *Computer Science* (1992), pp. 351–377.

[13] P. Odifreddi. *Classical Recursion Theory*. Classical Recursion Theory. Elsevier, 1992. ISBN: 9780444894830.

[14] H. L. Bodlaender. "A partial k-arboretum of graphs with bounded treewidth". In: *Theoretical Computer Science* 209.1-2 (1998), pp. 1–45. DOI: 10.1016/s0304-3975(97)00228-4.

[15] L. Mendoza and E. R. Alvarez-Buylla. "Dynamics of the genetic regulatory network for *Arabidopsis thaliana* flower morphogenesis". In: *J. Theoret. Biol.* 193 (1998), pp. 307–319.

[16] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004. ISBN: 978-3-540-21202-7.

[17] J. Aracena. “Maximum number of fixed points in regulatory Boolean networks”. In: *Bull. Math. Biol.* 70 (2008), pp. 1398–1409.

[18] G. Karlebach and R. Shamir. “Modelling and analysis of gene regulatory networks”. In: *Nature Rev. Mol. Cell Biol.* 9 (2008), pp. 770–780.

[19] A. Durand et al. “Fifty years of the spectrum problem: survey and new results”. In: *Bull. Symb. Log.* 18.4 (2012), pp. 505–553.

[20] R. Downey and M. Fellows. *Fundamentals of Parameterized Complexity*. Springer-Verlag London, 2013. ISBN: 978-1-4471-5558-4.

[21] Bireswar Das, Patrick Scharpfenecker, and Jacobo Torán. “CNF and DNF succinct graph encodings”. In: *Information and Computation* 253 (2017). LATA 2014, pp. 436–447. ISSN: 0890-5401. DOI: 10.1016/j.ic.2016.06.009.

[22] L. Fortnow and R. Santhanam. “Robust simulations and significant separations”. In: *Inform. and Comput.* 256 (2017), pp. 149–159.

[23] J. Demongeot and S. Sené. “About block-parallel Boolean networks: a position paper”. In: *Nat. Comput.* 19.1 (2020), pp. 5–13.

[24] G. Gamard et al. “Rice-like theorems for automata networks”. In: *Proceedings of the 38th Symposium on Theoretical Aspects of Computer Science*. Ed. by M. Bläser and B. Monmege. Vol. 187. LIPICs. Saarbrücken, Germany, 2021, pp. 8:1–8:15.