HAL open science

Hardness of monadic second-order formulae over succinct graphs

Guilhem Gamard, Pierre Guillon, Kévin Perrot, Guillaume Theyssier

▶ To cite this version:

Guilhem Gamard, Pierre Guillon, Kévin Perrot, Guillaume Theyssier. Hardness of monadic second-order formulae over succinct graphs. 2023. hal-03978957v1

HAL Id: hal-03978957

https://hal.science/hal-03978957v1

Preprint submitted on 8 Feb 2023 (v1), last revised 13 Jun 2024 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Hardness of monadic second-order formulae over succinct graphs

G. Gamard P. Guillon K. Perrot G. Theyssier

Version of February 8, 2023

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.

We actually prove this result in the terminology of automata network, which is a generalization of finite cellular automata over arbitrary graphs. This model arose from the biological modelization of neural networks and gene regulation networks. Our result states that every arborescent MSO property on the transition graph of automata networks is either NP-hard or coNP-hard.

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. 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 transition matrix). Now what if the graph is not typical, but presents some structure that allows a shorter encoding? Assume that the graph is described succintly, i.e, by a Boolean circuit which computes the adjacency relation between nodes, which are represented by binary numbers. One natural question is whether it is possible to exploit the circuits in better ways than just querying for possible edges, in order to more directly deduce structural information about the transition graph. Our main result essentially tells that it is impossible as soon as the property is non-trivial for bounded treewidth graphs.

Theorem 1.1. *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*. Many of the properties considered in the literature so far.

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. One of the initial intents behind this definition was to model the dynamics of gene regulation [4, 6, 13, 14, 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 [2, 3, 5, 9, 11, 17, 19, 20].

Let us give a slightly more formal description of automata networks. An AN is given by a finite digraph, called *communication graph*, where each node is endowed with a local update function. The graph is directed, so one-way transfer of information is possible. Different automata may have different transition tables and state sets. To update a node 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 nodes are updated synchronously (though an extensive literature has explored other *update modes* [25]), so that the state of v at time $t+1$ only depends on the states of its neighbors at time t.

This description suggests an alternative way to think about automata networks: they can be seen as a generalization of finite cellular automata. The generalization resides in that the "grid" of an AN can be an arbitrary (finite) digraph, rather than just a regular grid. Besides, each cell gets its own update function, so no uniformity is enforced. That analogy between automata networks and cellular automata is relevant, as we often meet the same sort of research problems in both cases: information needs to flow in the

network to enable nodes to perform useful tasks, so we need to understand what can or cannot be transmitted efficiently, and where.

In general, the automata in a network may be nondeterministic, so that the network as a whole may be nondeterministic. Hence we need to consider network updates as a relation, rather than a function. We now formalize this idea. Let F denote an automata network. A *configuration* of F is a possible state of the whole system, i.e., an assignment of a state to each node. If configuration x_1 evolves in one round into configuration x_2, then we say that x_1 *transitions* to x_2. We sometimes say that x_1 is a *predecessor* of x_2, or that x_2 is a *successor* of x_1. In general, automata networks are nondeterministic, so that a given configuration might have several successors.

The concepts of *predecessor* and *sucessor* allow to state virtually all interesting questions about F. For instance:

- Is F deterministic, i.e., do all configurations have exactly one successor? If not, how many configurations have more than one successor? How many have none? What is the maximal number of successors for a configuration?
- Is F injective, i.e., do all configurations have exactly one predecessor? How many have more than one? Or none? What is the maximal number of predecessors?
- Does F have a fixed point, i.e., a configuration that is its own successor? How many are there?
- Does F have a cycle of configurations? (A sequence of configurations $x_0, \ldots, x_{k-1}$ such that x_i transitions to x_{i+1}, with indices taken modulo k.) How many of them does it have? What are their lengths?

The *transition graph* of F, denoted by $\mathcal{G}_F$, is the digraph of the *transition* relation. In other terms, its vertices are the configurations of F and there is an edge from configuration x_1 to configuration x_2 if x_1 transitions to x_2. The transition graph is exponentially bigger than the communication graph. That graph describes the dynamical behavior of F: all the questions above (and many more) are actually questions about $\mathcal{G}_F$ rather than F.

We are interested in testing properties of $\mathcal{G}_F$, given F as input. When given as input to an algorithm, F is encoded through a Boolean circuit that, given two integer sequences $z_1, \ldots, z_n$ and $z'_1, \ldots, z'_n$ in binary, returns whether or not the first configuration can transition into the second one. This corresponds to a succinct representation for $\mathcal{G}_F$.

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. Similarly in biology, experimental observations show how each pair of nodes (in this case, genes) influence each other; for a given node, the influences should be summed or otherwise accumulated. This data does not give a complete transition table for each node, but can easily be encoded as Boolean circuits. If a transition table happens to be available, it can be re-encoded as a circuit quickly and efficiently without sacrificing much space.

In a sense, designing algorithms (respectively, hardness results) for automata network is like designing algorithms (respectively, hardness results) for succinct graphs. In

particular, every graph is the transition graph of an automata network, i.e., for every graph $G = (V, E)$ there is an automata network F such that $\mathcal{G}_F = G$ (we can choose F with just one nondeterministic automaton having $|V|$ states). The results from [26] already hint that the encoding cannot be smartly used to efficiently solved some questions. For instance,

Theorem 1.2 (see [26])**.** *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 F as input, whether $\mathcal{G}_F$ satisfies ϕ.*

In particular, first-order logic cannot express any non-trivial polynomial-time solvable question about the dynamics of 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. The intuition behind this hardness result is that automata networks are morally a model of computation, so they are subjected to some analogue of Rice's theorem:

Theorem 1.3 (see [1])**.** *Any nontrivial property of the function computed by a Turing machine is undecidable.*

The "function computed by a Turing machine" in Theorem 1.3 compares to "property of $\mathcal{G}_F$" in Theorem 1.2, while "undecidable" compares to "NP-hard or coNP-hard" and "trivial" compares to "$O(1)$". There is no hope to obtain undecidability results on automata networks because $\mathcal{G}_F$ is a finite object: we can always enumerate all configurations of F and reconstruct $\mathcal{G}_F$, then test whatever we want on it. That would take exponential space, though, so it makes sense to try and find more direct algorithms or hardness results for questions about $\mathcal{G}_F$. What we show is that such algorithms, under standard complexity assumptions ($\mathsf{P} \neq \mathsf{NP}$), cannot run in polynomial time.

There are other "Rice theorems" for models of computation, e.g., cellular automata: see [12].

Contributions. The present work started as an attempt to generalize Theorem 1.2 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 F deterministic?" is expressible in FO. If we restrict ourselves to deterministic ANs, that question is $O(1)$; in general, it is not. Thence the generalization requires to prove, in particular, that determinism is either NP- or coNP-hard.

Questions in MSO are stronger than questions in FO. For instance, MSO can express "$\mathcal{G}_F$ is connected", while FO cannot. Thence we need to prove, in particular, that connectedness of the transition graph is either NP- or coNP-hard.

Hardness of connectedness and determinism will both be consequences of our main result, Theorem 1.4. 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 not very exciting from the perspective of graph theory. 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 1.4. *If ϕ is an* arborescent *MSO sentence, then testing ϕ on the transition graph of automata networks is either* NP- *or* coNP-*hard.*

This theorem rephrases Theorem 1.1. 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, so "how many fixpoints are there?" 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 1.5. *There is a (nonarborescent) first-order sentence ψ such that, under plausible complexity assumptions, testing ψ in the transition graph of a given AN 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

Automata networks. An *automata network* is a finite digraph $F = (V, E)$ where each node v is equipped with a finite *state set* Q_v and a *local function* (also called *transition*

function):

$$f_v : \left(\prod_{(u,v)\in E} Q_u \right) \to Q_v.$$

A *configuration* is an assignment of a state to each node; formally, it is an element of $\prod_{u\in V} Q_u$. If x is a configuration and v a node of F, we write x_v for the state of v in x; formally, it is the projection $\pi_v(x)$. The *updated configuration* $F(x)$ is the configuration given by:

$$F(x)_v = f_v \left(\prod_{(u,v)\in E} x_u \right),$$

for every v in V. As the notation $F(x)$ suggests, we often view an automata network as a *global function*:

$$F : \prod_{u\in V} Q_u \to \prod_{u\in V} Q_u$$

from the set of configurations to itself.

We will consider *nondeterministic* automata networks throughout this paper, unless specified otherwise. The global function F becomes a *global relation* $F \subseteq \prod_{u\in V} Q_u \times \prod_{u\in V} Q_u$ (or equivalently a "multi-valued functions"), which tells whether a configuration can transition to another one. The terminology and notation of the deterministic case carry over to nondeterministic networks. This global nondeterminism can come from the Cartesian product of local nondeterministic $f_v \subseteq \prod_{u\in V} Q_u \times Q_v$ (a natural definition for nondeterministic automata networks), but it is generally not enough. For instance, a particle moving nondeterministically on a graph cannot be modeled be a Cartesian product of local nondeterministic choices made at each node, correlation between choices of neighboring nodes is required. To emphasize the locality of transitions in the general case, a nondeterministic automata network can be defined by $F = \{(x, y) : \forall U \in \mathcal{U}, (x_U, y_U) \in f_U\}$, where $\mathcal{U}$ is a set of subsets of V, and for each of its elements $U \in \mathcal{U}$, $f_U \subseteq \prod_{u\in U} Q_u \times \prod_{u\in U} Q_u$ is a relation "located" over U.

Remark 2.1. The automata networks constructed in our proof cannot generally be obtained through the Cartesian product of local determinism, but they are non-deterministic in a very weak sense: each configuration has only a bounded number of images (the transition graph has bounded degree). The bound depends only on the formula. This makes the result stronger that it is actually stated.

Another point of view is that of *succinct graph representations*, where the communication graph and the locality of map F is discarded while describing the graph $\mathcal{G}_F$. 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.

It is easy to transform an automata network F encoded as above into a succinct graph representation of $\mathcal{G}_F$. Conversely, one can see a succinct representation (N, C) of a graph G as a (degenerate) automata network with just one node and N states whose transition graph is G.

We always assume that a Boolean circuit is not bigger than its transition table: if it were, we could encode the table into a circuit of the same size (up to a polynomial factor).

MSO Logic. Given an automata network F, we want to test Monadic Second-Order logic formulae inside its transition graph $\mathcal{G}_F$. MSO formulae have two kinds of variables: vertices $(x_1, x_2, \dots)$ and sets of vertices $(X_1, X_2, \dots)$—in our case, vertices are configurations. Accordingly, there are two kinds of existential quantifiers: existence of a configuration, and existence of a set of configurations. 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 transition $(F(x_1, x_2)$, meaning $F(x_1) = x_2$ or more correctly $x_2 \in F(x_1))$. Note that the transition symbol F 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:

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

With this MSO signature, it is possible to express as a macro the relation $F^*(x, y)$: "there exists a chain $x = z_1, z_2, \dots, z_{n-1}, z_n = y$ such that $F(z_k, z_{k+1})$ 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 [23] 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, there is at least one bag in T containing 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 alternances). Unless stated otherwise: **increasing** means strictly increasing; **integer** means positive or null 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 problem ϕ-Dynamics as follows:

> ϕ-Dynamics
> *Input:* an automata network F (given as Boolean circuits for the local relation of each node).
> *Output:* does $\mathcal{G}_F \models \phi$?

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.

Analogously, we can define the model checking problem of ϕ on graphs given by a succinct representation.

> Succinct-ϕ
> *Input:* a succinct representation (C, N) of some graph G.
> *Output:* does $G \models \phi$?

Following the remarks on automata networks encoding and succinct representations, it is straightforward to show that for any MSO formula ϕ the problems ϕ-Dynamics and Succinct-ϕ are LOGSPACE-equivalent. Our main result is the following.

Theorem 3.1. *If ϕ has infinitely many models with the same treewidth k_1 and infinitely many countermodels with the same treewidth k_2, then ϕ-Dynamics is either NP-hard or coNP-hard. The result still holds if we restrict inputs of ϕ-Dynamics to automata networks with bounded non-determinism (transition graph with outdegree at most k, for large enough k depending on ϕ).*

It can also be phrased as a counterpoint to Courcelle's theorem, as follows:

Corollary 3.2. *If ϕ is an MSO property with infinitely many models and countermodels among graphs of bounded treewidth, then the problem Succinct-ϕ is either NP-hard or coNP-hard. The result still holds if we restrict to graphs of bounded degree.*

An MSO sentence is *arborescent* if it satisfies the condition of Theorem 3.1. All examples of sentences given in the previous section are arborescent. We will deal with 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 an automata network F such that $\mathcal{G}_F \models \phi$ if and only if S has at least one positive assignment. The network F has s automata with two states (0 and 1) and one "big" automaton with $|\Omega| = |Y|$ states. What F does is interpret the states of the s Boolean automata as an assignment for the variables of S, and evaluate S on that assignemnt. If it finds "true", then the big automaton transitions so as to realize a copy of Ω in $\mathcal{G}_F$. If it finds "false", then the big automaton realizes a copy of Y instead. (The Boolean automata never change their states.) Consequently, $\mathcal{G}_F$ 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 dynamics satisfies ϕ. Otherwise, the dynamics 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 F that depends on ϕ is the evaluation of S, but automata network are encoded as Boolean circuits. It is easy, given an instance of SAT, to produce in polynomial time circuits that evaluate a given assignment.

Of course, things are not that simple. First problem: we will not be able to control whether Ω (called a *saturating graph*) turns any graph into a model or into a countermodel. If it so happens that Ω turns any graph into a countermodel, then 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 designated vertices of G with designated 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, the reduction from SAT described above does not work anymore. Indeed we have to make some space for X and Z in the dynamics of F, and we have to account for the merged vertices (per definition

of $\oplus$). This requires care, because we cannot allow any extraneous configuration in the dynamics: every single configuration 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 3.1.

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 3.1.

4 A graph saturating all sentences of fixed quantifier rank

Proposition 4.1. *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 4.1.

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 4.2. *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 ith 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 4.3 (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 4.1. 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-Fraïsé 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 4.2, 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 $P = (p_0, \ldots, p_{k-1})$ 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 ith port of G is merged with the ith port of G'.

A *k-biboundaried graph*, or k-graph for short, is a digraph G endowed with two sequences of k ports: the *primary ports*, denoted by $P_1(G)$, and the *secondary ports*, denoted by $P_2(G)$. (So we have $|P_1(G)| = |P_2(G)| = k$.) 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 k-graphs; if w is a nonempty word over alphabet I, then define $\Delta^\Gamma(w)$ by induction over the length of w as follows:

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

See Figure 5.1—although the possibility that P_1 and P_2 intersect is not shown on that figure.

Proposition 5.1. *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 k-graphs $\Gamma = \{G_1, G_2, G_3\}$ 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 5.1. Fix a sentence ϕ and an integer k.

Definition 5.2. Let G, G' be k-graphs. We write $G \sim G'$ if and only if for every k-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. Call Σ the set of equivalence classes of $\sim$.

Lemma 5.3 (Theorem 13.1.1 from [23])**.** *The set Σ of equivalence classes of $\sim$ is finite.*

Definition 5.4. Let G denote a k-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 $\sim$.

Remark 5.5. 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 $\mathcal{C}_T(v)$.

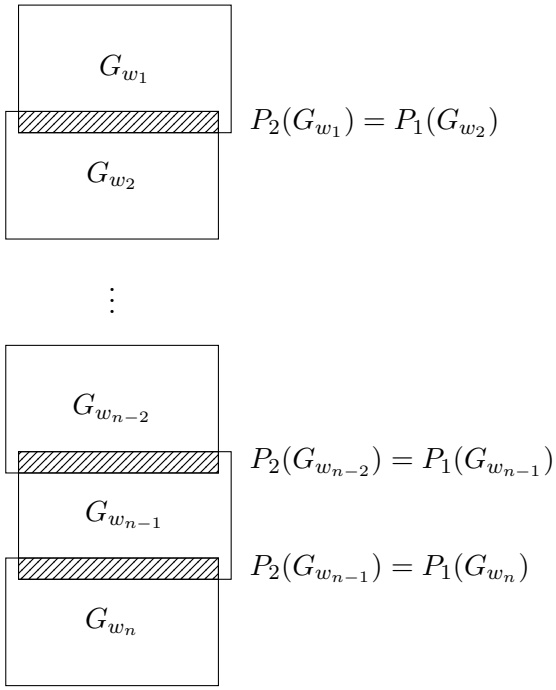

Figure 5.1: Illustration of Δ

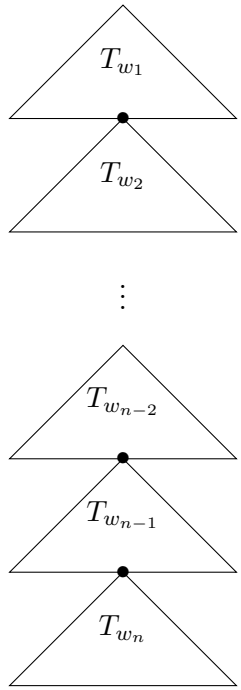

Figure 5.2: Illustration of Λ

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 5.5 ensures that they are either all models, or all countermodels. The next definition applies both to tree decompositions and to Σ-labeled trees.

Definition 5.6. 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 5.2.

Lemma 5.7. *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 $T \setminus (\mathcal{S}_T(v) - \{v\})$ 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. □

Remark 5.8. Let $\Gamma = \{G_i\}_{i \in I}$ a collection of k-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 5.2 are the nodes in hatched areas in Figure 5.1.

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 5.9. *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 5.1. The sentence ϕ has models with k-tree decompositions having arbitrarily high numbers of nodes. By Lemma 5.9, 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, such that D_i has depth at least i. View the D_i's as Σ-labeled trees. By Lemma 5.3 the set Σ of values for $\mathcal{C}(\cdot)$ is finite; let $n = |\Sigma| + 1$, so that any Σ-labeled path of length n contains two nodes with the same label. In particular, the model M_n has a tree decomposition D_n containing a path of length n, thus 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 5.3):

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

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 $P_2(G_3)$. Let $\Gamma = (G_1, G_2, G_3)$ and $\mathcal{T} = (T_1, T_2, T_3)$, so that by Remark 5.8, 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 5.7 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 5.10. Note that in Proposition 5.1, 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. Therefore it holds that an MSO formula has infinitely many models models among bounded treewidth graphs if and only if it does among bounded pathwidth ones.

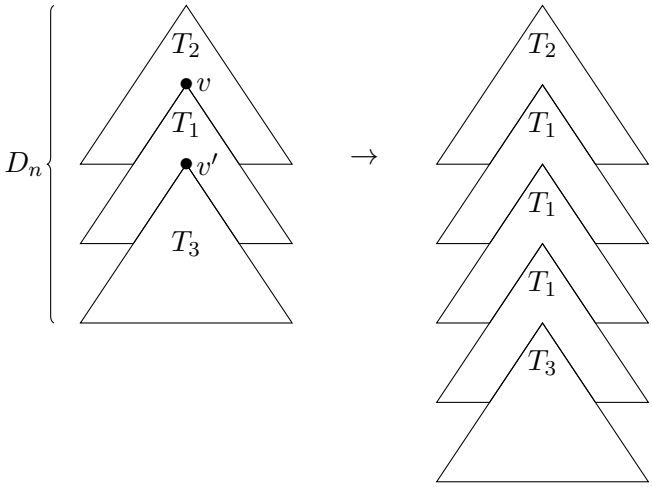

Figure 5.3: Proof of Proposition 5.1.

6 A reduction from ϕ-Dynamics 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 $\{0, \ldots, |G| - 1\}$.

Proposition 6.1. *Fix an MSO sentence ϕ. Let k be an integer and $\Gamma = \{G_0, G_1, G_2, G_3\}$ four k-graphs such that:*

(i) $|G_0| = |G_1|$*;*
(ii) $P_1(G_0) \cap P_2(G_0) = P_1(G_1) \cap P_2(G_1)$*; 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 ϕ-dynamics is NP*-hard, even if we restrict to inputs with bounded non-determinism.*

Definition 6.2. 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).

Lemma 6.3. *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) of Proposition 6.1. Then, there is an integer c and an automata network F such that*

$$\mathcal{G}_F = \Delta^\Gamma(2 \cdot \overline{S} \cdot 1^c \cdot 3),$$

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

Proof. The proof is in two steps. The first step only gives the configuration space and the global transition relation of F. It does not explain how these transitions are realized by an automata network, nor how Boolean circuits for that network may be produced in polynomial time: that is deferred to the second step.

First step. For now, let c be an arbitrary positive or null integer. (We will need to perform some form of padding in the second step and c will be that amount of padding; it remains unspecified in the first step.) 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 6.1. 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:

$$\begin{aligned} &P_1'(G_i) = (0, \ldots, k' - 1), \quad P_2'(G_i) = (|G_i| - k, \ldots, |G_i| - k'' - 1), \quad P_3'(G_i) = (|G_i| - k'', \ldots, |G_i| - 1), \\ &P_1'(G_3) = (0, \ldots, k' - 1), \quad P_2'(G_3) = (|G_3| - k', \ldots, |G_3| - 1), \qquad P_3'(G_3) = (k', \ldots, k - 1). \end{aligned}$$

Define the quantities:

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

The configuration space of F is $\{0, \ldots, n_2 + (2^s + c) \cdot n_1 + n_3 - 1\}$: see Figure 6.1 for a picture of its organization. The initial segment is the (unique) copy of G_2, followed by $2^s + c$ 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 $\ell = 2^s + c - 1$ be the index of the last copy of G_0 or G_1. For q in $\{-1, 0, \ldots, 2^s + c\}$, define (see the following paragraph for intuition):

$$\delta_0^q(r) = \delta_1^q(r) = \begin{cases} n_2 + q \cdot n_1 + r & \text{if } r \in \{0, \ldots, |G_1| - k'' - 1\}, \\ n_2 + \ell \cdot n_1 + r & \text{if } r \in \{|G_1| - k'', \ldots, |G_1| - 1\}; \end{cases} \tag{6.1}$$

$$\delta_2^q(r) = \begin{cases} r & \text{if } r \in \{0, \ldots, |G_2| - k'' - 1\}, \\ \delta_1^\ell(r) + |G_1| - |G_2| & \text{if } r \in \{|G_2| - k'', \ldots, |G_2| - 1\}; \end{cases} \tag{6.2}$$

$$\delta_3^q(r) = n_2 + (2^s + c) \cdot n_1 + r. \tag{6.3}$$

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 rth node of a copy of G_j in Figure 6.1, i.e., in the configuration space of F. 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 $2^s + c$) 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 (6.1) and (6.2) 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 transition function F, from the configuration space to itself. Recall that F is nondeterministic, so the value of $F(x)$ is a set.

- If $x < n_2$, then set:

$$F(x) = (\delta_2^0 \circ G_2)(x). \tag{6.4}$$

- If $x - n_2 < (2^s + c) \cdot n_1$, then by Euclidean division 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; if $q - 1$ (respectively q) is 2^s or higher, then set i (respectively j) to 1 instead. Set:

$$F(x) = \begin{cases} (\delta_j^q \circ G_j)(r) \sqcup (\delta_i^{q-1} \circ G_i)(r + n_i) & \text{if } r \in \{0, \dots, k' - 1\}, \\ (\delta_j^q \circ G_j)(r) & \text{otherwise.} \end{cases} \tag{6.5}$$

- Otherwise, $n_2 + (2^s + c) \cdot n_1 \leq x$, so let $q = 2^s + c$ and $r = x - (n_2 + (2^s + c) \cdot n_1)$. Let $i = \overline{S}(2^s - 1)$ if $c = 0$ and $i = 1$ otherwise, and set:

$$F(x) = \begin{cases} (\delta_3^0 \circ G_3)(r) \sqcup (\delta_i^{q-1} \circ G_i)(r + n_1) & \text{if } 0 \leq r < k', \\ (\delta_3^0 \circ G_3)(r) \sqcup (\delta_2^0 \circ G_2)(r + n_2) \bigsqcup_{t=0}^{\ell} (\delta_1^t \circ G_1)(r + n_1) & \text{if } k' \leq r < k, \\ (\delta_3^0 \circ G_3)(r) & \text{otherwise.} \end{cases} \tag{6.6}$$

Observe that $\mathcal{G}_F$ is, by construction, $\Delta^\Gamma(2 \cdot \overline{S} \cdot 1^c \cdot 3)$. Assume, for the sake of the discussion, that $S(0)$ 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 $\{0, \dots, k' - 1\}$ 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 $n_2 + n_1 + \{0, \dots, k' - 1\}$ in the global dynamics—see the overlaps in Figure 6.1). The edges going out of those shared nodes into G_0 are realized by the first case in Equation (6.5). 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 in the dynamics. 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 (6.6). That case assumes that all copies of G_0 and G_1 in the dynamics are in fact copies of G_1; this yields the desired results by Condition (iii).

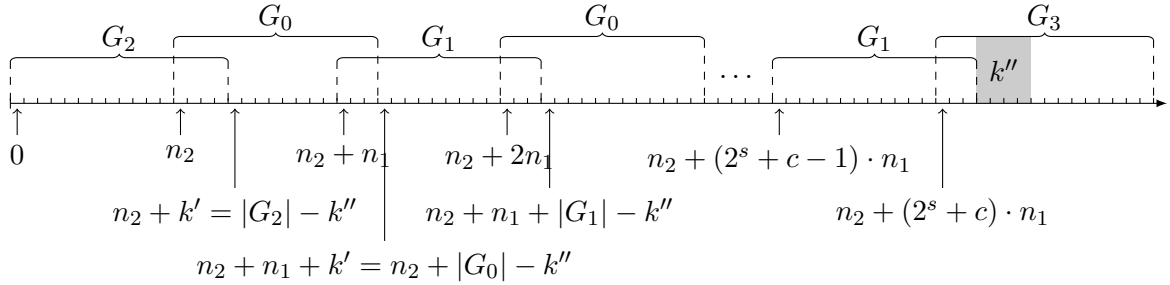

Figure 6.1: Proof of Lemma 6.3, a representation of $\{0, \ldots, n_2 + (2^s + c) \cdot n_1 + n_3 - 1\}$ as the configuration space of F. The gray area (k'') is actually common to all the graphs.

Second step. We have a configuration space and a transition relation; we want an automata network that realizes them. First, we determine (in polynomial time) the number of automata of F and the alphabet of each automaton. This process fixes the value of c, which does no harm, because the previous step worked for arbitrary values of c.

Let $p = n_2 + n_3$ and $d = \gcd(p, n_1)$; write $p = p' \cdot d$ and $n_1 = n' \cdot d$ for coprime integers p', n'. Find an integer t such that $p \leq 2^t$ and $\gcd(s + t, \varphi(n')) = 1$, where φ denotes Euler's totient and s is the number of Boolean variables of S. To do so, let $t' = s/\gcd(s, \varphi(n'))$, so that t' and $\varphi(n')$ have no common prime factors. Then let t'' denote a power of t' that exceeds $s + \lceil \log_2 p \rceil$ (compute it by successive squarings). Finally, take $t = t'' - s$.

Since $\gcd(p', n') = 1$, we can use Algorithm 17.1 from [22] to find an integer $e \geq 1$ such that $e^{s+t} \equiv p' \mod n'$. For the rest of the proof, assume that $e \geq 2$: indeed, if $e = 1$, then $p' \equiv 1 \mod n'$, so by Euler's formula we can choose $e = p'^{\varphi(n')}$ instead. Since Algorithm 17.1 from [22] runs in polynomial time and p', n' are constants, we can find e and t in polynomial time. (Recall that s is the number of variables of S, our input, so we can consider that s is given in unary.)

The automata network F has one node over alphabet $\{0, \ldots, d - 1\}$ and $s + t$ nodes over alphabet $\{0, \ldots, e - 1\}$. Its configuration space has $d \cdot e^{s+t}$ elements. Since $2 \leq e$, we have $2^s \leq e^s$ and $p \leq 2^t \leq e^t$; moreover we have $e^{s+t} \equiv p' \mod n'$, so there is an integer $c \geq 0$ such that $d \cdot e^{s+t} = p + (2^s + c) \cdot n$.

Now that we have the right number of configurations, it is easy to produce in polynomial time the Boolean circuits for F: they read the current configuration as an integer and apply the rules from the first step of this proof. The G_i are considered constant, and the only part that depends on S merely evaluates S. Moreover, each configuration requires at most two evaluations of S to compute the set of successors (in Equation 6.5, for the values of i and j). This concludes the proof. □

Proof of Proposition 6.1. Immediate consequence of Lemma 6.3: given an instance S of SAT, compute in polynomial time the corresponding automata network F, and ask ϕ-dynamics on that F. Moreover, the transition graphs of such automata networks are

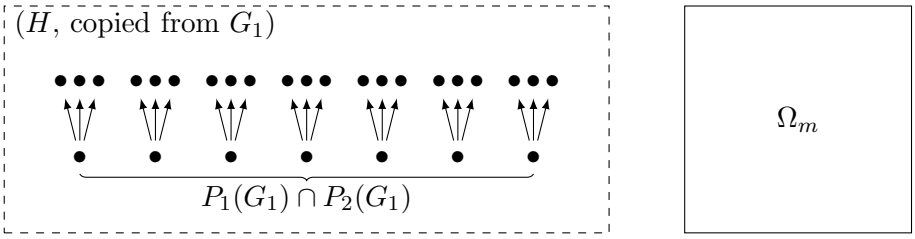

Figure 6.2: Representation of G_0 in the proof of Theorem 3.1.

all of the form $\Delta^\Gamma(w)$ for a fixed set of finite graphs Γ so they have a bounded degree, meaning that the constructed automata networks have bounded non-determinism. □

Proof of Theorem 3.1. Let ϕ denote an arborescent MSO sentence with m quantifiers. By Proposition 4.1, the graph Ω_m is either forbidden or sufficient for ϕ. Assume that Ω_m is forbidden for ϕ and prove that ϕ-dynamics is NP-hard. (If it was sufficient, then consider $\neg\phi$ instead—this would yield that ϕ-dynamics is coNP-hard.) By Proposition 5.1, 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 ϕ for every integer n. Let H denote the subgraph of G_1 spanned by:

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

and $G_0 = H \sqcup \Omega_m$ (see Figure 6.2). 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 6.1. This concludes the proof of the theorem. □

Remark 6.4. Note that the constructed automata networks are such that all nodes except one have the same alphabet. The proof of Lemma 6.3 is as close as we could get to a proof working for automata networks with constant uniform alphabet, which is the most usual setting in the literature. We have no idea how to close the gap to fit this setting.

7 Nonarborescent formulae

In this section, we explore what happens when the *arborescence* condition in Theorem 3.1 is negated.

First, if ϕ is an MSO sentence with finitely many models or with finitely many countermodels, then it is not arborescent. But in this case, we can test ϕ in constant time. Indeed, if ϕ has finitely many models, then there are finitely many automata networks whose dynamics satisfy ϕ (because we assume that Boolean circuits are never bigger than their transition tables). So testing ϕ amounts to testing whether the input belongs to a fixed list, which can be done in constant time: any input larger than the

largest element of the list can immediately be rejected without further computation. We call such sentences **trivial**.

More interestingly, the question "is $\mathcal{G}_F$ 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, make a network with s Boolean automata that views its own configuration as an assignement for S and evaluates it. If it finds "false", then each automaton can choose nondeterministically either to change state or to remain in the same state. If it finds "true", then all automata are required to stay in the same state. The transition graph of that network 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 ϕ-Dynamics is either NP- or coNP-hard. The next theorem shows that this conjecture is unlikely to hold.

Definition 7.1 (Definition 1 from [24])**.** 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 7.2. *There is a nontrivial first-order sentence ψ such that, if either SAT or UNSAT reduces to ψ-*Dynamics*, then there is a polynomial-time algorithm solving SAT for a robust set of sizes of instances.*

According to Proposition 8 and Theorem 12 of [24], our Theorem 7.2 implies:

Corollary 7.3. *Let ψ be given by Theorem 7.2. If either SAT or UNSAT reduces to ψ-*Dynamics*, 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 7.2. For now, let ψ be an arbitrary first-order sentence, and f a polynomial reduction from either SAT or UNSAT to ψ-Dynamics.

Definition 7.4. Let P be a polynomial and n a positive integer. The reduction f is **P-meager in** n if and only if for every instance S of SAT with size n, we have:

$$|f(S)| \leq \log P(n).$$

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

Let P denote a polynomial. If f is P-meager in some integer n, then all instances of SAT with size n are mapped to networks whose transition graphs have $O(P(n))$ vertices. Indeed, the transition graph is exponentially large in the network description. Besides, we can test ψ on a graph G in time polynomial in $|G|$, the degree of the polynomial being the quantifier rank of ψ (recall that ψ is a first-order sentence). Therefore:

Lemma 7.5. *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 whatever answer: f is not P-meager in n anyway. 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 7.6. *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:

$$t(n+1) > \max\{2^{P'(t(n))}, t(n)^k\}. \tag{7.1}$$

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 (7.1), 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 instances of SAT such that:

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

in a primitive recursive fashion (simply by enumerating 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 (7.1), 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 7.7. The **spectrum** of ψ is the set of sizes of models of ψ. In symbols:

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

See [21] 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 exatly $h(n)$ steps on input n written in binary.

Lemma 7.8. *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 [7, Theorem 4.5] and [8, 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 7.9. *There is a first-order sentence ψ and an integer d such that, for every increasing primitive-recursive function R, we have:*

$$\mathrm{Im}(R)^{(d)} \not\subseteq \mathrm{Spec}(\psi) \subseteq \mathbb{N}^{(d)}. \tag{7.2}$$

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 $(R'_n)_{n\in\mathbb{N}}$ 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:

$$E(n) = \{R_i(j) : 0 \leq i, j \leq n\}, \tag{7.3}$$

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

$$h(n) > \max E(n), \tag{7.4}$$

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 (7.3)–(7.4); since h is increasing, $R_n(i)$ is not in $\mathrm{Im}(h)$. The existence of desired formula ψ follows from Lemma 7.8. □

Proof of Theorem 7.2. Let ψ and d be given by Lemma 7.9 and assume that f is a polynomial reduction from either SAT or UNSAT to ψ-Dynamics. 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 7.5 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 7.6 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 7.9, 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 ϕ-Dynamics. Let us first point out that parametrization by the size or quantification rank of the formula fails. The formula $\exists x : F(x, x)$ expresses the property: "F has at least one fixed point." That question is NP-hard: given an instance S of SAT, produce an AN with $|S|$ Boolean automata that evaluates its own configuration on S; if it finds "true", nothing changes; if it finds "false", it transitions to the next configuration (in lexicographic order). 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 transition graph produced is either a huge cycle or a sequence of paths ending in loops. This implies that they all have treewidth 1, so parameterization by the treewidths of transition graphs also fails: we have NP-difficulty even when the treewidth is guaranteed to be at most 1.

Another relevant parameter, when taking the automata networks point of view, is the size of the alphabets Q_v used at each nodes. For instance, as it is usually the case in the automata networks literature, one could consider ϕ-dynamics restricted to automata networks where all nodes share the same alphabet of size q. Our reduction in Lemma 6.3 fails in this settings as it produces automata networks with possibly two different alphabet sizes among their nodes. 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 7.2 does not say whether the formula ϕ has a polynomial-time solvable ϕ-Dynamics problem. We do not know whether it could be the case for some

ϕ under reasonable complexity assumptions.

It would be interesting to have average-case difficulty rather than worst-case difficulty, e.g., to show that "draw a configuration at random and check whether it is a fixed point" is not a good algorithm to find fixed points in general. In the same vein, general hardness results on *probabilistic* automata network would also be welcome.

Finally, let us recall that the main result of this paper do 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 of bounded degree can be naturally described as labeled graphs. In such cases, arborescent formulae can have a polynomial-time ϕ-DYNAMICS 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 4.1.

References

[1] H. G. Rice. "Classes of Recursively Enumerable Sets and Their Decision Problems". In: *Transactions of the American Mathematical Society* 74 (1953), pp. 358–366.

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

[3] S. W. Golomb. *Shift Register Sequences.* Holden-Day Inc., 1967.

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

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

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

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

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

[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. Kari. "Rice's theorem for the limit sets of cellular automata". In: *Theoretical Computer Science* 127 (1994), pp. 229–254.

[13] D. Thieffry and R. Thomas. "Dynamical behaviour of biological regulatory networks – II. Immunity control in bacteriophage lambda". In: *Bull. Math. Biol.* 57 (1995), pp. 277–297.

[14] 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.

[15] C. Espinosa-Soto, P. Padilla-Longoria, and E. R. Alvarez-Buylla. "A gene regulatory network model for cell-fate determination during *Arabidopsis thaliana* flower development that is robust and recovers experimental gene expression profiles". In: *The Plant Cell* 16 (2004), pp. 2923–2939.

[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. Richard. "Local negative circuits and fixed points in non-expansive Boolean networks". In: *Discr. Appl. Math.* 159 (2011), pp. 1085–1093.

[20] J. Demongeot, M. Noual, and S. Sené. "Combinatorics of Boolean automata circuits dynamics". In: *Discr. Appl. Math.* 160 (2012), pp. 398–415.

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

[22] J. H. Silverman. *A friendly introduction to number theory*. 4th edition. Pearson Education, 2012. ISBN: 978-0-321-81619-1.

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

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

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

[26] 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.