Available online at www.sciencedirect.com

SCIENCE DIRECT®

Annals of Pure and Applied Logic 139 (2006) 74–109

ANNALS OF PURE AND APPLIED LOGIC

www.elsevier.com/locate/apal

Succinct definitions in the first order theory of graphs

Oleg Pikhurko[a], Joel Spencer[b], Oleg Verbitsky[c]

[a] *Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA 15213, United States*
[b] *Courant Institute, New York University, New York, NY 10012, United States*
[c] *Department of Mechanics & Mathematics, Kyiv University, Ukraine*

Received 1 April 2004; accepted 1 April 2005
Available online 31 May 2005

Communicated by I. Moerdijk

Abstract

We say that a first order sentence A defines a graph G if A is true on G but false on any graph non-isomorphic to G. Let $L(G)$ (resp. $D(G)$) denote the minimum length (resp. quantifier rank) of such a sentence. We define the succinctness function $s(n)$ (resp. its variant $q(n)$) to be the minimum $L(G)$ (resp. $D(G)$) over all graphs on n vertices.

We prove that $s(n)$ and $q(n)$ may be so small that for no general recursive function f we can have $f(s(n)) \geq n$ for all n. However, for the function $q^*(n) = \max_{i \leq n} q(i)$, which is the least nondecreasing function bounding $q(n)$ from above, we have $q^*(n) = (1+o(1)) \log^* n$, where $\log^* n$ equals the minimum number of iterations of the binary logarithm sufficient to lower n to 1 or below.

We show an upper bound $q(n) < \log^* n + 5$ even under the restriction of the class of graphs to trees. Under this restriction, for $q(n)$ we also have a matching lower bound.

We show a relationship $D(G) \geq (1-o(1)) \log^* L(G)$ and prove, using the upper bound for $q(n)$, that this relationship is tight.

For a non-negative integer a, let $D_a(G)$ and $q_a(n)$ denote the analogs of $D(G)$ and $q(n)$ for defining formulas in the negation normal form with at most a quantifier alternations in any sequence of nested quantifiers. We show a superrecursive gap between $D_0(G)$ and $D_3(G)$ and hence between $D_0(G)$ and $D(G)$. Despite this, for $q_0(n)$ we still have a kind of log-star upper bound: $q_0(n) \leq 2 \log^* n + O(1)$ for infinitely many n.

E-mail addresses: spencer@cs.nyu.edu (J. Spencer), oleg@ov.litech.net (O. Verbitsky).
URL: http://www.math.cmu.edu/~pikhurko/ (O. Pikhurko).

doi:10.1016/j.apal.2005.04.003

Keywords: Definability; Finite graphs; First order logic; Turing machine simulation

1. Introduction

We study sentences about graphs expressible in the laconic first order language with two relation symbols $\sim$ and $=$ for, respectively, the adjacency and the equality relations. *First order* means that we are allowed to quantify only over vertices, as opposed to the second order logic case where we can quantify over sets of vertices. The difference between the first order and the second order worlds is essential. In the first order language we cannot express many basic properties of graphs, such as connectedness and the property of being bipartite (see, e.g., [28, Theorems 2.4.1 and 2.4.2]). On the other hand, the crucial fact for us is that the first order language is powerful enough to define any individual finite graph up to isomorphism. Indeed, a graph G with vertex set $V(G) = \{1, \ldots, n\}$ and edge set $E(G)$ is defined by the formula

$$\exists x_1 \ldots \exists x_n \forall x_{n+1} \Bigg(\bigwedge_{1 \le i < j \le n} \neg(x_i = x_j) \ \wedge \ \bigvee_{i \le n} x_{n+1} = x_i \\ \wedge \bigwedge_{\{i,j\} \in E(G)} x_i \sim x_j \ \wedge \bigwedge_{\{i,j\} \notin E(G)} \neg(x_i \sim x_j) \Bigg). \tag{1}$$

This fact, though very simple, highlights a fundamental difference between the finite and the infinite: there are non-isomorphic countable graphs satisfying precisely the same first order sentences (see, e.g., [28, Theorem 3.3.2]).

The question we address is how succinctly a graph G on n vertices can be defined by first order means. We consider two natural measures of succinctness — the length of a first order formula and its quantifier rank. The latter is the maximum number of nested quantifiers in the formula. Let $D(G)$ be the minimum quantifier rank of a closed first order formula defining G, that is, being true on G and false on any other graph non-isomorphic to G. The sentence (1) ensures that $D(G) \le n + 1$. This bound generally cannot be improved as $D(G) = n + 1$ for G being the complete or the empty graph on n vertices. However, for all other graphs we have $D(G) \le n$. Thus, it is reasonable to try to lower the trivial upper bound of $n + 1$ to some $u(n) \le n$ and explicitly describe all exceptional graphs with $D(G) > u(n)$. This is done in [21] with $u(n) = n/2 + O(1)$ (see also [23] for a generalization to arbitrary structures). More precisely, let us call two vertices of a graph *similar* if they are simultaneously adjacent or not to any other vertex. This is an equivalence relation and each equivalence class spans a complete or an empty subgraph. Let $\sigma(G)$ denote the maximum number of pairwise similar vertices in G. Then, as shown in [21],

$$\sigma(G) + 1 \le D(G) \le \max\left\{\frac{n+5}{2}, \sigma(G) + 2\right\}.$$

It seems doubtful that results of this sort can be obtained with upper bound $u(n) = cn + O(1)$ for each constant $c < 1/2$. The known Cai–Fürer–Immerman construction

[2] gives graphs with linear $D(G)$ which may serve as counterexamples to most natural conjectures in this direction.

While the paper [21] addresses the definability of n-vertex graphs in the worst case, in [14] we treat the average case. Let G be a random graph distributed uniformly among the graphs with vertex set $\{1, \ldots, n\}$. Then, as shown in [14],

$$|D(G) - \log_2 n| = O(\log_2 \log_2 n)$$

with probability $1 - o(1)$.

We now consider another extremal case of the graph definability problem. How succinct can a first order definition of a graph on n vertices be in the best case? That is, we study the succinctness function $q(n)$ defined as the minimum $D(G)$ over n-vertex G. We also define $L(G)$ to be the minimum length of a sentence defining G and $s(n)$ to be the minimum $L(G)$ over n-vertex G. Trivially, $q(n) < s(n)$. Our first result is that $s(n)$ and $q(n)$ may be so small that for no general recursive function f can we have $f(s(n)) \geq n$ for all n.

The proof is based on simulation of a Turing machine M by a first order formula A_M in which a computation of M determines a graph satisfying A_M and vice versa. Such techniques were developed in the classic research on Hilbert's *Entscheidungsproblem* by Turing, Trakhtenbrot, Büchi and other researchers (see [1] for survey and references). An important feature of our simulation is that it works if we restrict the class of structures to graphs. The key ingredient of our proof is a gadget allowing us to impose an order relation on the vertex set of a graph.

As a by-product, we obtain another proof of Lavrov's result [16] that the first order theory of finite graphs is undecidable. Our proof actually shows the undecidability of the $\forall^*\exists^p\forall^s\exists^t$-fragment of this theory for some p, s, and t.

From the fact that $q(n)$ and n are not recursively linked, it easily follows that, if a general recursive function $l(n)$ is monotone nondecreasing and tends to the infinity, then

$$q(n) < l(n) \text{ for infinitely many } n. \tag{2}$$

Our next result establishes a general upper bound

$$q(n) < \log^* n + 5 \qquad \text{for all } n. \tag{3}$$

Here $\log^* n$ equals the minimum number of iterations of the binary logarithm sufficient to lower n below 1. It turns out that this is the best possible monotonic upper bound for $q(n)$. Let $q^*(n) = \max_{i \leq n} q(i)$, which is the least monotone nondecreasing function bounding $q(n)$ from above. We prove that

$$q^*(n) \geq \log^* n - \log^* \log^* n - O(1). \tag{4}$$

As the upper bound (3) is monotonic, we obtain

$$q^*(n) = (1 + o(1)) \log^* n. \tag{5}$$

Comparing (5) to (2) with $l(n) = \log^* n$, we conclude that $q(n)$ infinitely often deviates from its "smoothed" version $q^*(n)$ and, in particular, is essentially nonmonotonic.

Proving (3) and (4), we use a robust technical tool given by the Ehrenfeucht game [5] (these techniques were also developed by Fraïssé [7] in a different setting).

As a matter of fact, we prove the upper bound (3) under the restriction of the class of graphs to trees only, that is, we have $q(n) \le q(n; trees) < \log^* n + 5$. Recall that, by (2), $q(n)$ is infinitely often so small that we cannot bound it from below by any "regular" function. The proof of this fact cannot be carried through for $q(n; trees)$ because, as a well-known corollary of the Rabin theorem [25], the first order theories of both all and finite trees are decidable and hence a Turing machine computation cannot be simulated by a first order sentence about trees. In fact, for $q(n; trees)$ we establish a matching lower bound, thereby determining this function asymptotically, namely,

$$q(n; trees) = (1 + o(1)) \log^* n.$$

We pay special attention to defining sentences having a restricted structure. For a non-negative integer a, let $D_a(G)$ and $q_a(n)$ denote the analogs of $D(G)$ and $q(n)$ for defining formulas in the negation normal form with at most a quantifier alternations in any sequence of nested quantifiers. The superrecursive gap between $s(n)$ and n is actually shown even under the restriction of the alternation number to 3. Note also that, as follows from a result in [14], $q_3(n) \le \log^* n + O(1)$ and hence (5) holds with alternation number 3.

On the other hand, we show a superrecursive gap between $D_0(G)$ and $D_3(G)$ and hence between $D_0(G)$ and $D(G)$. Despite this, for $q_0(n)$ we also have a kind of log-star upper bound: $q_0(n) \le 2 \log^* n + O(1)$ for infinitely many n. It is worth noting that this is not the first case where we have close results for the alternation number 0 and for the unbounded alternation number. In [14] we prove that for a random graph $D(G)$ and $D_0(G)$ are not so far apart from each other — that is, $D_0(G) \le (2+o(1)) \log_2 n$ with probability $1-o(1)$. Yet another result showing the same phenomenon is obtained in [21]. Given non-isomorphic graphs G and G', let $D(G, G')$ (resp. $D_0(G, G')$) denote the minimum quantifier rank of a sentence (resp. in the negation normal form with no quantifier alternation) which is true on exactly one of the graphs. As shown in [21], if both G and G' have n vertices, then $D(G, G') \le D_0(G, G') \le (n+5)/2$ and there are simple examples of such G and G' with $D(G, G') \ge (n + 1)/2$. Note that logically distinguishing non-isomorphic graphs with equal numbers of vertices has close connections to graph canonization algorithms (see, e.g., [2,8,21] and a monograph [12]).

Relating $D(G)$ and $L(G)$ to one another, we show that

$$D(G) \ge (1 - o(1)) \log^* L(G).$$

Using the bound (3), we show that this relationship is tight.

Focusing on defining formulas of restricted structure, we also consider prenex formulas. A superrecursive gap between $s(n)$ and n can actually be shown under the restriction to this class. Nevertheless, prenex formulas generally are not competitive against defining formulas with no restriction on structure. We observe that graphs showing a huge gap between $D(G)$ and $L(G)$ at the same time show a huge gap between $D(G)$ and its version for prenex defining formulas.

In conclusion, note that all of our results carry over to general structures over any relational vocabulary with at least one non-unary relation symbol. For the upper bounds this claim is straightforward because graphs can be viewed as a subclass of such structures which is distinguishable by a single first order sentence. The lower bounds hold true with minor changes in the proofs.

2. Background

2.1. Arithmetics

We define the *tower function* $T(i)$ by $T(0) = 1$ and $T(i) = 2^{T(i-1)}$ for each subsequent i. Sometimes this function will be denoted by *Tower* (i). Given a function f, we will denote by $f^{(i)}$ the i-fold composition of f. In particular, $f^{(0)}(x) = x$. By $\log n$ we always mean the logarithm base 2. The inverse of the tower function, the *log-star* function $\log^* n$, is defined by $\log^* n = \min\{i : T(i) \geq n\}$. For a real x, the notation $\lceil x \rceil$ (resp. $\lfloor x \rfloor$) stands for the integer nearest to x from above (resp. from below).

2.2. Graphs

Given a graph G, we denote its vertex set by $V(G)$ and its edge set by $E(G)$. The *order* of G, the number of vertices of G, will sometimes be denoted by $|G|$, that is, $|G| = |V(G)|$. The *neighborhood* of a vertex v consists of all vertices adjacent to v. A set $S \subseteq V(G)$ is called *independent* if it contains no pair of adjacent vertices. If $X \subseteq V(G)$, then $G[X]$ denotes the subgraph *induced* by G on X (or *spanned* by X in G). If $u \in V(G)$, then $G - u = G[V(G) \setminus \{u\}]$ is the result of removing from G the vertex u along with all incident edges.

The *distance* between vertices u and v, the minimum length of a path connecting the two vertices, is denoted by $d(u, v)$. If u and v are in different connected components of a graph, then $d(u, v) = \infty$. The *eccentricity* of a vertex v is defined by $e(v) = \max_{u \in V(G)} d(v, u)$. The *diameter* and the *radius* of a graph G are defined by $d(G) = \max_{v \in V(G)} e(v)$ and $r(G) = \min_{v \in V(G)} e(v)$ respectively. A path in a graph is *diametral* if its length is equal to the diameter of the graph. A vertex v is *central* if $e(v) = r(G)$.

Proposition 2.1 (*[19, Theorem 4.2.2]*). *Let T be a tree. If $d(T)$ is even, then T has a unique central vertex c and all diametral paths go through c. If $d(T)$ is odd, then T has exactly two central vertices c_1 and c_2 and all diametral paths go through the edge* $\{c_1, c_2\}$.

2.3. Logic

2.3.1. Formulas

First order formulas are assumed to be over the set of connectives $\{\neg, \wedge, \vee\}$. A *sequence of quantifiers* is a finite word over the alphabet $\{\exists, \forall\}$. If S is a set of such sequences, then $\exists S$ (resp. $\forall S$) means the set of concatenations $\exists s$ (resp. $\forall s$) for all $s \in S$. If s is a sequence of quantifiers, then $\bar{s}$ denotes the result of the replacement of all occurrences of $\exists$ by $\forall$ and vice versa in s. The set $\bar{S}$ consists of all $\bar{s}$ for $s \in S$.

Given a first order formula A, its set of *sequences of nested quantifiers* is denoted by *Nest*(A) and defined by induction as follows:

(1) $\mathit{Nest}(A) = \{\epsilon\}$ if A is atomic; here ϵ denotes the empty word.

(2) $\mathit{Nest}(\neg A) = \overline{\mathit{Nest}(A)}$.

(3) $\mathit{Nest}(A \wedge B) = \mathit{Nest}(A \vee B) = \mathit{Nest}(A) \cup \mathit{Nest}(B)$.

(4) $\mathit{Nest}(\exists x\, A) = \exists\, \mathit{Nest}(A)$ and $\mathit{Nest}(\forall x\, A) = \forall\, \mathit{Nest}(A)$.

The *quantifier rank* of a formula A, denoted by $qr(A)$, is the maximum length of a string in $Nest(A)$.

We adopt the notion of the *alternation number* of a formula (cf. [20, Definition 2.8]). Given a sequence of quantifiers s, let $alt(s)$ denote the number of occurrences of $\exists\forall$ and $\forall\exists$ in s. The *alternation number* of a first order formula A, denoted by $alt(A)$, is the maximum $alt(s)$ over $s \in Nest(A)$. The alternation number has an absolutely clear meaning for formulas in the *negation normal form*, where the connective $\neg$ occurs only in front of atomic subformulas. This number is defined for any formula A so that, if A is reduced to an equivalent formula A' in the negation normal form, then $alt(A) = alt(A')$.

Viewing a formula A as a string of symbols over the countable first order alphabet (where each variable and each relation is denoted by a single symbol), we denote the *length* of A by $|A|$. Note that if one prefers, in a natural way, to encode variable and relation symbols in a finite alphabet, then the length will increase but stay within $|A| \log |A|$.

We call A an $\exists$-*formula* (resp. $\forall$-*formula*) if any sequence in $Nest(A)$ with maximum number of quantifier alternations starts with $\exists$ (resp. $\forall$). We denote the set of formulas in the negation normal form with alternation number at most m by Λ_m. By $\Lambda_m^\exists$ (resp. $\Lambda_m^\forall$) we denote the subset of Λ_m consisting of formulas in Λ_{m-1} and $\exists$-formulas (resp. $\forall$-formulas) in $\Lambda_m \setminus \Lambda_{m-1}$. We will call formulas in $\Lambda_0^\exists$ and $\Lambda_0^\forall$ *existential* and *universal* respectively.

A *prenex formula* is a formula with all its quantifiers up front. In this case there is a single sequence of nested quantifiers and the quantifier rank is just the number of quantifiers occurring in a formula. Let Σ_1 and Π_1 denote, respectively, the sets of existential and universal prenex formulas. Furthermore, let Σ_m (resp. Π_m) be the extension of $\Sigma_{m-1} \cup \Pi_{m-1}$ with prenex formulas in $\Lambda_{m-1}^\exists$ (resp. $\Lambda_{m-1}^\forall$). Note that the classes of formulas Λ_m, $\Lambda_m^\exists$, $\Lambda_m^\forall$, Σ_m, and Π_m are defined so that they are closed with respect to subformulas.

The following lemma is an immediate consequence of the standard reduction of a formula to the prenex form.

Lemma 2.2. *The conjunction of Σ_m-formulas (resp. Π_m-formulas) is effectively reducible to an equivalent Σ_m-formula (resp. Π_m-formula). The same holds for the disjunction.* □

We write $A \equiv B$ if A and B are logically equivalent formulas and $A \doteq B$ if A and B are literally the same.

Lemma 2.3.

(1) *Any formula in $\Lambda_m^\exists$ is effectively reducible to an equivalent formula in Σ_{m+1}.*
(2) *Any formula in $\Lambda_m^\forall$ is effectively reducible to an equivalent formula in Π_{m+1}.*
(3) *Any formula in Λ_m is effectively reducible to an equivalent formula in Σ_{m+2} or, as well, to an equivalent formula in Π_{m+2}.*

Proof. Item 3 follows from Items 1 and 2 as Λ_m is included both in $\Lambda_{m+1}^\exists$ and $\Lambda_{m+1}^\forall$. To prove Items 1 and 2, we proceed by induction on m.

Consider the base case of $m = 0$. Assume that $A \in \Lambda_0^\exists$ and let $t = t(A)$ denote the total number of quantifiers and connectives $\wedge$, $\vee$ in A. We prove that A has an equivalent formula $A' \in \Sigma_1$ using induction on t. If $t = 0$, then A is quantifier free and hence in Σ_0. Let $t \geq 1$. Assume that $A \doteq \exists x B$. Since $t(B) = t(A) - 1$, the assumption of induction on t

applies to B. Therefore B reduces to an equivalent formula $B' \in \Sigma_1$ and we set $A' = \exists x B'$. Assume that $A \doteq B \wedge C$ (the case where $A \doteq B \vee C$ is similar). Neither of $t(B)$ and $t(C)$ exceeds $t(A)-1$ and, by the assumption of induction on t, for B and C we have equivalents B' and C' in Σ_1. Then $A \equiv B' \wedge C'$ reduces to an equivalent in Σ_1 by Lemma 2.2.

The reducibility of $\Lambda_0^{\forall}$ to Π_1 is proved similarly.

Let $m \geq 1$ and assume that Items 1 and 2 of the lemma are true for the preceding value of m. Given $A \in \Lambda_m^{\exists}$, we show how to find an equivalent formula $A' \in \Sigma_{m+1}$ (the reduction of $\Lambda_m^{\forall}$ to Π_{m+1} is similar). We again use induction on $t = t(A)$. If $t = 0$, then A is in Σ_0. Let $t \geq 1$. If $A \doteq \forall x B$, then $A \in \Lambda_{m-1}^{\forall}$ and, by the assumption of induction on m, A has an equivalent $A' \in \Pi_m \subset \Sigma_{m+1}$. If $A \doteq \exists x B$, $A \doteq B \wedge C$, or $A \doteq B \vee C$, then $B, C \in \Lambda_m^{\exists}$ and both $t(B)$ and $t(C)$ are smaller than $t(A)$. We are done by the assumption of induction on t and Lemma 2.2. □

A formula with all variables bound is called a *closed formula* or a *sentence*.

Lemma 2.4. *If A is a closed prenex formula of quantifier rank q with occurrences of h binary relation symbols, then it can be rewritten in an equivalent form A' with the same quantifier prefix so that $|A'| = O(hq^2 2^{hq^2})$.*

Proof. Let $B(x_1, \ldots, x_q)$ be the quantifier-free part of A. The B is a Boolean combination of $m = h\binom{q}{2}$ atomic subformulas and hence is representable as a DNF of length $O(m2^m)$. □

2.3.2. Structures

A *relational vocabulary* σ is a finite set of *relation symbols* augmented with their *arities*. We always assume the presence of the binary relation symbol $=$ standing for the equality relation and do not include it in σ. The only exception will be Section 5.4 where the presence or the absence of equality will be stated explicitly.

A *structure over vocabulary* σ (or an σ-structure) is a set along with relations that are named by symbols in σ and have the corresponding arities. We mostly deal with the vocabulary of a single binary relation symbol. A structure over this vocabulary can be viewed as a *directed graph* (or *digraph*). We treat *graphs* as structures with a single binary relation which is symmetric and anti-reflexive. This relation will be called the *adjacency* relation and denoted by $\sim$.

If all relation symbols of a sentence A are from the vocabulary σ and G is an σ-structure, then A is either true or false on G. In the former case G is called a *model* of A. We also say that G *satisfies* A. We call A *valid* if all σ-structures satisfy A. We call A *(finitely) satisfiable* if it has a (finite) model. Clearly, A is valid iff $\neg A$ is unsatisfiable.

2.3.3. Computability

Whenever we say that something can be done *effectively*, we mean that this can be implemented by an *algorithm*. No restrictions on running time or space are assumed. Professing *Church's thesis*, we here do not specify any definition of the algorithm. Nevertheless, we will refer to *Turing machines* (see Section 4.2.1) and *recursive functions* in Sections 4 and 5. As a basic fact, these two computational models are equally powerful, under an effective bijection between binary words and non-negative integer numbers.

Let X be a set of words over a finite alphabet. The *decision problem* for X is the problem of recognizing whether or not a given word belongs to X. If there is an algorithm that does it, the decision problem is *solvable* (or X is *decidable*).

The *halting problem* is the problem of deciding, for given Turing machine M and input word w, whether M eventually halts on w or runs forever. This is a basic unsolvable problem. It is well known that, if we fix w to be the empty word, the restricted problem remains unsolvable.

The *(finite) satisfiability problem* is the problem of recognizing whether or not a given sentence is (finitely) satisfiable (we here assume any natural encoding of formulas in a finite alphabet). Settling Hilbert's *Entscheidungsproblem*, Church and Turing proved that the satisfiability problem is unsolvable. The unsolvability of the finite satisfiability problem was shown by Trakhtenbrot [29].

A *general recursive function* is an everywhere defined recursive function.

2.3.4. The Bernays–Schönfinkel class of formulas and the Ramsey theorem

A class of formulas has the *finite model property* if every satisfiable formula in the class has a finite model. By the completeness of the predicate calculus with equality, the set of valid sentences is recursively enumerable. From here it is not hard to conclude that, if a class of formulas has the finite model property, the satisfiability and the finite satisfiability problems for this class are solvable.

The *Bernays–Schönfinkel class* consists of prenex formulas in which the existential quantifiers all precede the universal quantifiers, that is, this is another name for Σ_2.

Proposition 2.5 (*The Ramsey Theorem [26]*).[1] *For each vocabulary σ there is a general recursive function $f : \mathbf{N} \to \mathbf{N}$ such that the following is true: assume that a σ-sentence A with equality is in the Bernays–Schönfinkel class. If A has a model of some cardinality at least $f(qr(A))$ (possibly infinite), then it has a model in every cardinality at least $f(qr(A))$. As a consequence, the Bernays–Schönfinkel class of formulas with equality has the finite model property and hence both the satisfiability and the finite satisfiability problems restricted to this class are solvable.*

2.3.5. Definability

Let G and G' be non-isomorphic graphs and A be a first order sentence with equality over vocabulary $\{\sim\}$. We say that *A distinguishes G from G'* if A is true on G but false on G'. By $D(G, G')$ (resp. $D_k(G, G')$) we denote the minimum quantifier rank of a sentence (resp. with alternation number at most k) distinguishing G from G'.

We say that a sentence A *defines* a graph G (up to isomorphism) if A distinguishes G from any non-isomorphic graph G'. To ensure that A has no other models except graphs, we will tacitly assume that A has form $A \doteq \forall_x (x \not\sim x \wedge \forall_y (x \sim y \to y \sim x)) \wedge B$. By $D(G)$ (resp. $D_a(G)$) we denote the minimum quantifier rank of a sentence defining G (resp. with alternation number at most a). By $L(G)$ (resp. $L_a(G)$) we denote the minimum length of a sentence defining G (resp. with alternation number at most a).

[1] The *combinatorial Ramsey theorem*, a cornerstone of *Ramsey theory*, appeared in this paper as a technical tool.

A sentence is called *defining* if it defines a graph. Note that any defining sentence must contain the equality symbol. Let us stress that graphs G' in the above definition may have any cardinality.

Lemma 2.6. *All finite graphs and only finite graphs possess defining sentences.*

Proof. Any finite graph is indeed definable as it has at least the wasteful definition (1). By the upward Löwenheim–Skolem theorem (see [18, Corollary 2.35]), if a sentence with equality has an infinite model, it has a model of any infinite cardinality. For this reason, no infinite graph has a defining sentence in the sense of our definition. □

Lemma 2.7. *The class of defining $\Lambda_1^{\exists}$-sentences is decidable.*

Proof. Suppose that we are given a sentence $A \in \Lambda_1^{\exists}$. By Lemma 2.3(1), we can reduce it to an equivalent formula in the Bernays–Schönfinkel class and apply the Ramsey theorem. We are able to recognize whether A is defining in four steps.

(1) Check whether A is finitely satisfiable.
(2) If so, trying graphs one by one, we eventually find a graph of the smallest order n satisfying A (this is actually done in the first step, if it is based directly on the Ramsey theorem).
(3) Check whether there is any other graph of order n satisfying A.
(4) If not, check if a $\Lambda_1^{\exists}$-sentence $A \wedge \exists_{x_1,\dots,x_{n+1}} (\bigwedge_{1 \le i < j \le n+1} x_i \ne x_j)$ is satisfiable.

If not, and only in this case, A is defining. □

3. The Ehrenfeucht game

In this section we borrow a lot of material from [28, Section 2]. To make our exposition self-contained, we sketch some proofs that can be found in [28] in more detail.

The *Ehrenfeucht game* is played on a pair of structures of the same vocabulary. We give the definition conforming to the case of graphs.

Let G and H be graphs with disjoint vertex sets. The k-round Ehrenfeucht game on G and H, denoted by $\textsc{Ehr}_k(G, H)$, is played by two players, Spoiler and Duplicator (he and she for brevity), with k pairwise distinct pebbles $p_1, \dots, p_k$, each given in duplicate. Spoiler starts the game. A *round* consists of a move of Spoiler followed by a move of Duplicator. At the i-th move Spoiler takes pebble p_i, selects one of the graphs G or H, and places p_i on a vertex of this graph. In response Duplicator should place the other copy of p_i on a vertex of the other graph. It is permissible to place more than one pebble on the same vertex.

Let u_i (resp. v_i) denote the vertex of G (resp. H) occupied by p_i, irrespectively of which of the players placed the pebble on this vertex. If

$$u_i = u_j \text{ iff } v_i = v_j \qquad \text{for all } 1 \le i < j \le k,$$

and the component-wise correspondence $(u_1, \dots, u_k)$ to $(v_1, \dots, v_k)$ is a partial isomorphism from G to H, this is a win for Duplicator; Otherwise the winner is Spoiler.

The *a-alternation* Ehrenfeucht game on G and H is a variant of the game in which Spoiler is allowed to switch from one graph to another at most a times during the game, i.e., in at most a rounds he can choose the graph other than that in the preceding round.

Let $0 \le s \le k, r = k-s$, and assume that at the start of the game the pebbles $p_1, \dots, p_s$ are already on the board at vertices $\bar{u} = u_1, \dots, u_s$ of G and $\bar{v} = v_1, \dots, v_s$ of H. The r-round game with this initial configuration is denoted by $\textsc{Ehr}_r(G, \bar{u}, H, \bar{v})$. We write $G, \bar{u} \equiv_k H, \bar{v}$ if Duplicator has a winning strategy in this game.

It is not hard to check that $\equiv_k$ is an equivalence relation. The k-Ehrenfeucht value of a graph G with vertices $u_1, \dots, u_s$ marked by pebbles is the equivalence class it belongs to under $\equiv_k$. We let *Ehrv* (k, s) denote the set of all possible k-Ehrenfeucht values for graphs with s marked vertices. Let *Ehrv* (k) = *Ehrv* $(k, 0)$ denote the set of k-Ehrenfeucht values for graphs (with no marked vertex).

Lemma 3.1. *Assume that $s < k$. Let $\bar{u} = u_1, \dots, u_s$ and $S(G, \bar{u})$ denote the set of $\equiv_k$-equivalence classes of G with $s+1$ marked vertices $\bar{u}, u$ for all $u \in G \setminus \{u_1, \dots, u_s\}$. Then $G, \bar{u} \equiv_k H, \bar{v}$ iff $S(G, \bar{u}) = S(H, \bar{v})$.*

Proof. Consider the game $\textsc{Ehr}_{k-s}(G, \bar{u}, H, \bar{v})$. Suppose that $S(G, \bar{u}) \ne S(H, \bar{v})$; for example, there is $u \in V(G)$ such that $G, \bar{u}, u \not\equiv_k H, \bar{v}, v$ for any $v \in V(H)$. Let Spoiler select this u and let v denote Duplicator's response. From now on the players actually play $\textsc{Ehr}_{k-s-1}(G, \bar{u}, u, H, \bar{v}, v)$, where Spoiler has a winning strategy.

Suppose that $S(G, \bar{u}) = S(H, \bar{v})$. If Spoiler selects, for example, a vertex $u \in V(G)$, then Duplicator responds with $v \in V(H)$ such that $G, \bar{u}, u \equiv_k H, \bar{v}, v$ and hence has a winning strategy in the remaining part of the game. □

Lemma 3.2 (*[28, Theorem 2.2.1]*). *For any s and k, Ehrv (k, s) is a finite set. Furthermore, let $f(k, s) = |Ehrv\,(k, s)|$. Then*

$$f(k, k) \le 4^{\binom{k}{2}}, \tag{6}$$

$$f(k, s) \le 2^{f(k,s+1)} \tag{7}$$

for $s < k$.

Proof. The bound (6) holds because the $\equiv_k$-equivalence class of G with marked $u_1, \dots, u_k$ is determined by the equality relation on the sequence $u_1, \dots, u_k$ and the induced subgraph $G[\{u_1, \dots, u_k\}]$. The bound (7) holds because the $\equiv_k$-equivalence class of an arbitrary G with marked $\bar{u} = u_1, \dots, u_s$ is, according to Lemma 3.1, determined by $S(G, \bar{u})$, a subset of *Ehrv* $(k, s+1)$. □

As a consequence, we obtain the following bound.

Lemma 3.3 (*[28, Theorem 2.2.2]*). $|Ehrv\,(k)| \le T(k + 2 + \log^* k) + O(1)$. □

We say that a formula $A(x_1, \dots, x_s)$ with s free variables *defines* an Ehrenfeucht value $\alpha \in Ehrv\,(k, s)$ if A is true on a graph G with variables $x_1, \dots, x_s$ assigned vertices $u_1, \dots, u_s$ for exactly those $G, u_1, \dots, u_s$ which are in α.

Lemma 3.4 (*[28, Theorem 2.3.2]*). *For any $\alpha \in Ehrv\,(k, s)$ there is a formula A_α with $qr(A_\alpha) = k - s$ that defines α. Moreover,*

$$|A_\alpha| \le 18\binom{k}{2} \text{ if } s = k \text{ and} \tag{8}$$

$$|A_\alpha| \le f(k, s+1)\left(\max\left\{|A_\beta| : \beta \in Ehrv(k, s+1)\right\} + 10\right) \text{ if } s < k. \tag{9}$$

Proof. The bound (8) holds because every $\alpha \in Ehrv(k, k)$ is defined by a formula of the type

$$\bigwedge_{1\le i<j\le k} (*(x_i = x_j) \wedge \star(x_i \sim x_j)),$$

where $*$ and $\star$ are $\neg$ for some (i, j) and nothing for the others, depending on adjacencies among the marked vertices of a $G, u_1, \ldots, u_k$ in α.

Let $s < k$ and assume that every $\beta \in Ehrv(k, s+1)$ has a defining formula $A_\beta(x_1, \ldots, x_s, x)$ of quantifier rank $k - s - 1$. Consider an $\alpha \in Ehrv(k, s)$ and choose a representative $G, \bar{u}$ of α. Define $S(\alpha) = S(G, \bar{u})$, where the right hand side is as in Lemma 3.1. By this lemma, the definition does not depend on a particular choice of $G, \bar{u}$. We set

$$A_\alpha(x_1, \ldots, x_s) \doteq \bigwedge_{\beta\in S(\alpha)} \exists_x A_\beta(x_1, \ldots, x_s, x) \wedge \bigwedge_{\beta\notin S(\alpha)} \neg\exists_x A_\beta(x_1, \ldots, x_s, x).$$

It is clear that G with designated $\bar{u} = u_1, \ldots, u_s$ satisfies A_α iff the set of Ehrenfeucht values with additional designated u is equal to $S(\alpha)$. By Lemma 3.1, the latter condition is true iff $G, \bar{u}$ has Ehrenfeucht value α. □

Proposition 3.5. *Suppose that G and H are non-isomorphic graphs.*

(1) *Let $R(G, H)$ denote the minimum k such that G and H have different k-Ehrenfeucht values. Then $D(G, H) = R(G, H)$. In other words, $D(G, H)$ equals the minimum k such that Spoiler has a winning strategy in* $\text{EHR}_k(G, H)$.
(2) *$D_a(G, H)$ equals the minimum k such that Spoiler has a winning strategy in the a-alternation* $\text{EHR}_k(G, H)$.

We refer the reader to [28, Theorem 2.3.1] for the proof of the first claim and to [20] for the second claim.

Proposition 3.6.

$$D(G) = \max\{D(G, H) : H \text{ and } G \text{ are non-isomorphic}\},$$
$$D_a(G) = \max\{D_a(G, H) : H \text{ and } G \text{ are non-isomorphic}\}.$$

The first equality can be restated as follows: $D(G)$ equals the minimum k such that the k-Ehrenfeucht value of G contains only graphs isomorphic to G.

Proof. We give a proof of the first equality that can be easily adopted for the second equality. Denote the maximum in the right hand side by k. We have $k \le D(G)$ as a matter of definition. Conversely, let $\alpha \in Ehrv(k)$ be the class containing G. By Proposition 3.5, G is, up to isomorphism, the only member of α. For each $\beta \ne \alpha$ in $Ehrv(k)$, fix a representative H_β. Let C_β be a sentence of quantifier rank at most k distinguishing G from H_β. We use Lemma 3.2 saying that $Ehrv(k)$ is finite. The conjunction of all C_β defines

G and has quantifier rank k. Thus, $D(G) \le k$. (Alternatively, we could use the known fact that, over a finite vocabulary, there are only finitely many inequivalent sentences of bounded quantifier rank; cf. Lemma 5.6.) □

4. A superrecursive gap: simulating a Turing machine

Definition 4.1. We define the *succinctness function* $s(n)$ (for formula length) by

$$s(n) = \min_{|G|=n} L(G).$$

The variants with bounded alternation number are defined by

$$s_a(n) = \min_{|G|=n} L_a(G)$$

for each $a \ge 0$.

It turns out that $s(n)$ can be so small with respect to n that the gap between the two numbers cannot be bounded by any recursive function.

Theorem 4.2. *There is no general recursive function f such that*

$$f(s_3(n)) \ge n \qquad \text{for all } n. \tag{10}$$

Lemma 4.3 (*Simulation Lemma*). *Given a Turing machine M with k states, one can effectively construct a sentence A_M with single binary relation symbol $\sim$ and equality so that the following conditions are met.*

(1) $qr(A_M) = k + 16$.
(2) $|A_M| = O(k^2)$.
(3) $alt(A_M) = 3$.
(4) *A_M is effectively reducible to an equivalent prenex formula P_M whose quantifier prefix has length $k + O(1)$, begins with k existential quantifiers, and has three quantifier alternations.*
(5) *Any model of A_M is a graph. If M halts on the empty input word, then A_M has a unique model G_M and the order of G_M is bigger than the running time of M.*
(6) *M halts on the empty input word iff A_M has a finite model.*

Proof of Theorem 4.2. Let $g(k)$ denote the longest running time on the empty input word ϵ of a k-state Turing machine (non-halting machines are excluded from consideration). Recognizing whether or not a given Turing machine with k states halts on ϵ easily reduces to computation of $g(k)$. As this variant of the halting problem is well known to be undecidable, the function $g(k)$ cannot be bounded from above by any general recursive function. For each k, fix a machine M_k with k states whose running time attains $g(k)$. Let A_{M_k} be as in the Simulation Lemma, G_k be the model of A_{M_k}, and n_k be the order of G_k. Let $l(k) = ck^2$ be the upper bound for $|A_{M_k}|$ ensured by the lemma. Note that A_{M_k} defines G_k.

Suppose on the contrary that (10) is true for some general recursive f. Since $s_3(n_k) \le |A_{M_k}| \le l(k)$, for every k we have

$$g(k) < n_k \le f(s_3(n_k)) \le \max_{i \le l(k)} f(i),$$

a contradiction. □

The proof of the Simulation Lemma takes the rest of this section.

4.1. Gadgets

We enrich our language with connectives $\to$ and $\leftrightarrow$ for the implication and the equivalence. Since the alternation number was defined for formulas with connectives $\neg, \wedge, \vee$, we should stress that $\to$ and $\leftrightarrow$ are used as shorthand for their standard definitions through $\neg, \wedge, \vee$. We introduce the new uniqueness quantifier $\exists!$ via

$$\exists!_x F(x) \doteq \exists_x F(x) \wedge \forall_x \forall_y (F(x) \wedge F(y) \to x = y)$$

for any formula F with a free variable x and with no free occurrences of y. Note that one occurrence of the uniqueness quantifier contributes 2 in the quantifier rank and 1 in the alternation number. We use relativized versions of the existential and the universal quantifiers in the standard way:

$$\exists_{C(x)} F(x) \doteq \exists_x (C(x) \wedge F(x)),$$
$$\forall_{C(x)} F(x) \doteq \forall_x (C(x) \to F(x)).$$

To ensure that any model of A_M is a graph, we put in A_M the two graph axioms (the irreflexivity and the symmetry of the relation $\sim$).

4.1.1. Ordering

We give a formula $P(x, x')$ with two free variables x and x' that, in any model, shall determine an order on the neighborhood of x. Let $X = \{y : y \sim x\}$ and $X' = \{z : z \sim x'\}$. Then $P(x, x')$ is the conjunction of the following:

(P1) $\{x, x'\}$, X, X' are all disjoint and each of them is independent.
(P2) $\forall_{y \in X} \exists_{z \in X'}\ y \sim z$.
(P3) $\exists_{y \in X} \exists!_{z \in X'}\ y \sim z$.
(P4) $\exists_{y \in X} \forall_{z \in X'}\ y \sim z$.
(P5) $\forall_{y_1 \in X} \forall_{y_2 \in X} [\forall_{z \in X'} (y_1 \sim z \to y_2 \sim z) \vee \forall_{z \in X'} (y_2 \sim z \to y_1 \sim z)]$.
(P6) $\forall_{y_1 \in X} \forall_{y_2 \in X} [y_1 \ne y_2 \to \exists_{z \in X'} (y_1 \sim z \leftrightarrow y_2 \not\sim z)]$.
(P7) $\forall_{y \in X} [\exists_{z \in X'}\ y \not\sim z \to \exists_{y^+ \in X} \exists!_{z \in X'} (y^+ \sim z \wedge y \not\sim z)]$.
(P8) $\forall_{y \in X} [\exists!_{z \in X'}\ y \sim z \vee \exists_{y^- \in X} \exists!_{z \in X'} (y \sim z \wedge y^- \not\sim z)]$.

Note that $qr(P) = 4$, $alt(P) = 2$ (contributed by (P7) and (P8)), and $|P| = O(1)$.

Consider finite models of $P(x, x')$. For $y \in X$ let $N^*(y)$ be those $z \in X'$ adjacent to y. The $N^*(y)$ are distinct (P6), linearly ordered under inclusion (P5), are nonempty (P2), include a singleton (P3) and all of X' (P4), and the set of all cardinalities $|N^*(y)|$ has no gaps (either (P7) or (P8)). So we must have $|X| = |X'|$ and the elements can be ordered,

$x_1, \dots, x_s, x'_1, \dots, x'_s$, so that x_i, x'_j are adjacent precisely when $j \le i$. We induce on X a binary relation $\le$ defined by

$$y_1 \le y_2 \doteq \forall_{z \in X'} (y_1 \sim z \to y_2 \sim z).$$

In any model (even infinite) the properties (P1)–(P8) assure that $\le$ is a linear order with a least and greatest element. Furthermore, every y has a successor y^+ and a predecessor y^- except when y is the last or first element of X respectively.

4.1.2. Coordinatization

We now give a formula $COOR(x, x', t, t', z)$ that shall coordinatize the neighborhood of z. Let X, X', T, T', Z denote the neighborhoods of x, x', t, t', z respectively. Then $COOR$ is the conjunction of the following:

(C1) $x, x', t, t', z, X, X', T, T', Z$ are all disjoint. Z is an independent set. All neighbors of Z are in $\{z\} \cup X \cup T$. There is no edge between $X \cup X'$ and $T \cup T'$.
(C2) $P(x, x') \wedge P(t, t')$.
(C3) $\forall_{z \in Z} (\exists!_{x \in X} z \sim x \wedge \exists!_{t \in T} z \sim t)$.
(C4) $\forall_{x \in X} \forall_{t \in T} \exists!_{z \in Z} (z \sim x \wedge z \sim t)$.

Thus, each $z \in Z$ has a unique pair of coordinates (x, t) and each (x, t) corresponds to a unique z. Note that $qr(COOR) = qr(P) = 4$ and $alt(COOR) = alt(P) = 2$.

4.1.3. New functional and constant symbols

To facilitate further description of A_M, we will use new functional symbols. In particular, this will allow us to have new constant symbols as symbols of nullary functions.

Writing $\bar{v}$, we will mean a finite sequence of variables $v_1, v_2, \dots$. As soon as a statement $\forall \bar{y} \exists! x F(x, \bar{y})$ is put in A_M or is derivable from what is already put in A_M, we may want to denote this unique x by $\phi(\bar{y})$ and use ϕ as a new functional symbol in the standard way. That is, if $Q(u, \bar{z})$ is a formula with free variables $u, \bar{z}$, then

$$Q(\phi(\bar{y}), \bar{z}) \doteq \exists x (F(x, \bar{y}) \wedge Q(x, \bar{z})) \text{ or}$$
$$Q(\phi(\bar{y}), \bar{z}) \doteq \forall x (F(x, \bar{y}) \to Q(x, \bar{z})).$$

Both variants are admissible and an appropriate choice of one of them may reduce the alternation number of a formula. Furthermore, in this way we can express compositions of several functions (e.g. [18, Section 2.9]).

In particular, in any model of $COOR(x, x', t, t', z)$ we let $1, 2$ denote the first two elements of X (under $\le$) and 0 (it will represent time zero) the first element of T. The same character ω will be used for the last element of X or T, dependent on context. For v in X or T, v^- and v^+ are respectively its predecessor and successor (when defined). The notation (x, t) will be used as a binary function symbol with meaning as explained in the preceding subsection.

4.2. Capturing a computation by a formula

4.2.1. Definition of a Turing machine

For technical reasons, we prefer to use the model of a Turing machine where the *tape* is infinite in one direction. It is known (e.g. [15, Section 41]) that it is equivalent to the model with the tape infinite in both directions. At the start the tape consists of the special “Left End of Tape” symbol L, followed by an input word written down in the binary alphabet $\{a, b\}$, and followed onward by all “blank” symbols B. A symbol occupies one cell. Let $s_1, \dots, s_k$ be *states* of a Turing machine M, with s_1 the *initial* state and s_k the *final* state. At the start M is in state s_1 and its head is at the first B. A machine is defined by a set of instructions of the following type, where $\alpha, \beta \in \{L, a, b, B\}$.

$s_i \alpha \beta s_j$: If in state s_i reading a symbol α, overwrite β and go to state s_j.
$s_i \alpha$ *Right* s_j: If in state s_i reading a symbol α, move the head one cell to the right and go to state s_j.
$s_i \alpha$ *Left* s_j: If in state s_i reading a symbol α, move the head one cell to the left and go to state s_j.

If $\alpha = L$ in an instruction of the first type, then $\beta = L$. This is the only case when $\beta = L$. There is no instruction of the third type (“move to the left”) for $\alpha = L$. With this exception, for every $i < k$ and α there is a unique instruction for what to do in state s_i reading α. The machine halts immediately after coming to state s_k. If M halts, its *running time* is the number of instructions executed before termination.

4.2.2. Formula A_M

For notational simplicity, we use the same name for variables and corresponding semantical objects (ingredients of M and vertices of a graph G_M). The vertex H below shall be used to keep track of the tape header. A_M is the conjunction of the two graph axioms and a long formula of the form

$$\exists_{x,x',t,t',z,s_1,\dots,s_k,a,b,B,L,H} B_M(x, x', t, t', z, s_1, \dots, s_k, a, b, B, L, H).$$

The formula B_M whose all free variables are listed above is the conjunction of the following subformulas, where X, X', T, T', Z denote, as before, the neighborhoods of x, x', t, t', z respectively.

(A1) $x, x', t, t', z, s_1, \dots, s_k, a, b, B, L, H, X, X', T, T', Z$ are disjoint and consist of *all* the vertices of the graph.
(A2) $COOR(x, x', t, t', z)$.
(A3) All of the neighbors of a, b, B, L, H are in Z.
(A4) For all $x \in X$ and $t \in T$ the vertex (x, t) is adjacent to precisely one of a, b, B, L. We will write $VAL(x, t)$ for this value, which represents the symbol on the Turing machine at position (cell of the tape) x and time t. Note that, as $VAL(x, t)$ ranges over four possible values L, a, b, B, using this functional symbol requires no extra quantification. For example, the formula $VAL(x, t) = \alpha$ reads just $(x, t) \sim \alpha$.
(A5) All neighbors of H are in Z. For all $t \in T$ there is a unique $x \in X$ for which (x, t) is adjacent to H. We write $HP(t)$ for this x, which represents the header position. Thus, $HP(t) = x$ reads $(x, t) \sim H$. We shall write $VAL(t) = VAL(HP(t), t)$, the symbol that the header is looking at time t. If HP is used within VAL, it takes one extra quantifier. Note that a subformula $VAL(t) = \alpha$ has quantifier rank 2 and alternation number 0. Furthermore, $VAL(t^+) = \alpha$ has quantifier rank 4 and can be written with alternation number 0.
(A6) The neighbors of $s_1, \dots, s_k$ are all in T. For all $t \in T$ precisely one of $s_1, \dots, s_k$

is adjacent to t. We write $ST(t)$ for this s_i, which represents the state at time t. Note that $ST(t) = s \doteq t \sim s$.

We want the Turing machine to start in the standard position:
(A7) $VAL(1, 0) = L \wedge \forall_{x \neq 1} VAL(x, 0) = B \wedge HP(0) = 2 \wedge ST(0) = s_1$.
We want the Turing machine to end in the final state and not be there before that:
(A8) $\forall_{t \in T} (ST(t) = s_k \leftrightarrow t = \omega)$.
We want values on the tape not to change except (possibly) at the header position:
(A9) $\forall_{t \in T, t \neq \omega} \forall_{x \in X} (x \neq HP(t) \rightarrow VAL(x, t^+) = VAL(x, t))$.
We want the rightmost spot on the tape to be used (we need this for uniqueness of the model; we do not want to allow superfluous blanks):
(A10) $\exists_{t \in T} VAL(\omega, t) \neq B$.
We need that the instructions would not push the Turing machine to the right of $x = \omega$. For every s_i, α such that when at state s_i and value α the instruction would push the header to the right we have
(A11) $\neg \exists_{t \in T} (VAL(t) = \alpha \wedge ST(t) = s_i \wedge HP(t) = \omega)$.

We are down to the core workings of the Turing machine. For each instruction of the first type we have
(A12) $\forall_{t \in T} \forall_{x \in X} (ST(t) = s_i \wedge HP(t) = x \wedge VAL(t) = \alpha \rightarrow ST(t^+) = s_j \wedge VAL(t^+) = \beta \wedge HP(t^+) = x)$.
For each instruction of the second type we have
(A13) $\forall_{t \in T} \forall_{x \in X} (ST(t) = s_i \wedge HP(t) = x \wedge VAL(t) = \alpha \rightarrow ST(t^+) = s_j \wedge HP(t^+) = x^+ \wedge VAL(x, t^+) = \alpha)$.
For each instruction of the third type we have
(A14) $\forall_{t \in T} \forall_{x \in X} (ST(t) = s_i \wedge HP(t) = x \wedge VAL(t) = \alpha \rightarrow ST(t^+) = s_j \wedge HP(t^+) = x^- \wedge VAL(x, t^+) = \alpha)$.

4.2.3. Proof of the Simulation Lemma

Straightforward inspection shows that $qr(B_M) = 6$, contributed, for example, by (A9). This gives Item 1 of the lemma. Since we treat a variable as a single symbol, (A1) and (A6) have length $O(k^2)$, (A11)–(A14) have length $O(k)$, and all the others have constant length. This gives Item 2. Straightforward inspection shows that $alt(B_M) = 2$, contributed by (A2). This gives Item 3.

Item 4 requires a bit of extra work. As $A_M \in \Lambda_3^{\exists}$, Lemma 2.3 implies that A_M is reducible to an equivalent prenex formula with quantifier prefix $\exists^*\forall^*\exists^*\forall^*$. We make a stronger claim that one can achieve the prefix $\exists^*\forall^{O(1)}\exists^{O(1)}\forall^{O(1)}$. Note that B_M has a constant number of conjunctive members with constant length and hence they contribute a constant number of quantifiers. (A1) and (A6), though they have length dependent on k, contain a constant number of quantifiers. The remainder, (A11)–(A14), should be tackled with more care as every one of these components, though it has a constant number of quantifiers, occurs in B_M in $O(k)$ variants for various pairs s_i, α. Fortunately, all these occurrences can be replaced by a single formula with a constant number of quantifiers. For example, introducing two new variables s and c, we can replace the conjunction of all variants of (A11) by

$$\neg\exists_{t\in T}\exists_s\exists_c\left[\bigvee_{s_i,\alpha}(s=s_i\wedge c=\alpha)\wedge VAL(t)=c\wedge ST(t)=s\wedge HP(t)=\omega\right],$$

where the disjunction is over the specified pairs s_i, α.

Let us turn to Items 5 and 6. It should be clear that, if M halts, its computation is converted to a graph satisfying A_M, whose order exceeds the running time. Such a graph is unique up to isomorphism because the adjacencies of any finite model of A_M must mirror the actions of the Turing machine. For the same reason, any finite model of A_M is converted into a halting computation of M and hence, if A_M has a finite model, then M halts on the empty input. It remains to note that, if M halts, then A_M has no infinite model. Let m be the running time of M. In any model of A_M, the first m values of t must simulate m steps of M's computation. By (A8), the set T is therefore finite. By (A10), the cardinality of X cannot exceed the cardinality of T and hence X is finite too. It immediately follows that the other components of the model, X', T', and Z, are finite as well. The proof is complete.

5. Other consequences of the Simulation Lemma

5.1. There are succinct definitions by prenex formulas

Due to (1), any graph of order n is definable by a prenex formula of quantifier rank $n+1$ with alternation number 1. Though the class of prenex formulas may appear rather restrictive, it turns out that, if one is allowed to increase the alternation number to 3, then there are graphs definable by prenex formulas with very small quantifier rank.

Definition 5.1. Let $L_a^{prenex}(G)$ denote the minimum length of a closed prenex formula with alternation number at most a that defines a graph G. Furthermore,

$$s_a^{prenex}(n) = \min_{|G|=n} L_a^{prenex}(G).$$

Theorem 5.2. *There is no general recursive function f such that $f(s_3^{prenex}(n)) \geq n$ for all n.*

Proof. We proceed precisely as in the proof of Theorem 4.2 but using, instead of A_M, the prenex formula P_M given by the Simulation Lemma. We will need a recursive bound $|P_{M_k}| \leq l(k)$. We can take $l(k) = ck^2 4^{k^2}$ owing to Lemma 2.4. □

5.2. The set of defining sentences is undecidable

Theorem 5.3. *The class of defining sentences is undecidable.*

Proof. Given a Turing machine M, consider a sentence A_M as in the Simulation Lemma. If M halts on the empty input, A_M is defining. Suppose that M never halts. Then either A_M has no model or it has an infinite model. By Lemma 2.6, A_M is not defining in both cases. We have thereby reduced the halting problem (for the empty input) to the decision problem for the set of defining sentences. □

Note a partial positive result given by Lemma 2.7.

5.3. $D_0(G)$ and $D(G)$ are not recursively related

Obviously, $D(G) \le D_0(G)$ for all graphs G. How far apart from each other can these two values be? Is there a converse relation $D_0(G) \le f(D(G))$, for some general recursive function f? The answer is "no". We will actually prove a stronger fact. Let $D_{1/2}(G)$ denote the minimum quantifier rank of a $\Lambda_1^{\exists}$-sentence that defines G. Notice the hierarchy

$$D(G) \le D_3(G) \le D_2(G) \le D_1(G) \le D_{1/2}(G) \le D_0(G).$$

We are able to show a superrecursive gap even between $D_3(G)$ and $D_{1/2}(G)$.

Theorem 5.4. *There is no general recursive function f such that*

$$D_{1/2}(G) \le f(D_3(G))$$

for all graphs G.

Lemma 5.5. *The finite satisfiability of a $\Lambda_1^{\exists}$-sentence is decidable.*

Proof. By Lemma 2.3, a $\Lambda_1^{\exists}$-sentence effectively reduces to an equivalent formula in the Bernays–Schönfinkel class. The finite satisfiability of the latter is decidable by the Ramsey theorem. □

The next lemma is related to the well-known fact that, over a finite vocabulary, there are only finitely many pairwise inequivalent sentences of bounded quantifier rank (cf. [2, Lemma 4.4]).

Lemma 5.6. *Given $m \ge 0$, one can effectively construct a finite set U_m consisting of $\Lambda_1^{\exists}$-sentences of quantifier rank m so that every $\Lambda_1^{\exists}$-sentence of quantifier rank m has an equivalent in U_m.*

Proof. Any sentence A of quantifier rank m can be rewritten in an equivalent form A' so that A' uses at most m variables, where different occurrences of the same variable are not counted (see e.g. [21, Proposition 2.3]). Referring to this fact, we will put in U_m only sentences over the variable set $\{x_1, \dots, x_m\}$. We now prove the lemma in a stronger form saying that, for each m and k such that $0 \le k \le m$, one can construct a finite set $U_{m,k}$ which is universal for the class of $\Lambda_1^{\exists}$-formulas of quantifier rank k over the variable set $\{x_1, \dots, x_m\}$ with precisely k variables bound.

We proceed by induction on k. Consider the base case of $k = 0$. There are $a = 2\binom{m}{2}$ atomic formulas $x_i \sim x_j$ and $x_i = x_j$. Any quantifier-free formula is a Boolean combination of these and can be represented by a perfect DNF (except the totally false formula for which we fix representation $x_1 = x_1 \wedge x_1 \ne x_1$). The set $U_{m,0}$ consists of all 2^{2^a} such expressions.

$U_{m,k}$ will consist of two parts, $U_{m,k}^{\exists}$ and $U_{m,k}^{\forall}$, the former for formulas with at least one existential quantifier and the latter for formulas with no existential quantifier. If $k = 0$, we have $U_{m,0}^{\exists} = \emptyset$ and $U_{m,0}^{\forall} = U_{m,0}$. Assume that $k \ge 1$ and $U_{m,k-1}$ has already been constructed. We construct $U_{m,k}$ in four steps.

(1) Put in $U_{m,k}^{\exists}$ the formulas $\exists x_i A$ for all $A \in U_{m,k-1}$ and $i \le m$ such that no occurrence of x_i in A is bound.

(2) Put in $U^{\forall}_{m,k}$ the formulas $\forall x_i A$ for all $A \in U^{\forall}_{m,k-1}$ and $i \le m$ such that no occurrence of x_i in A is bound.
(3) Put in $U^{\exists}_{m,k}$ all monotone Boolean combinations of formulas from $U^{\exists}_{m,k}$ and $U^{\forall}_{m,k}$ as constructed in Steps 1 and 2 with at least one formula from $U^{\exists}_{m,k}$ involved.
(4) Put in $U^{\forall}_{m,k}$ all monotone Boolean combinations of formulas from $U^{\forall}_{m,k}$ as constructed in Step 2.

Finally, to obtain U_m exactly as claimed in the lemma, we set $U_m = U_{m,m}$. □

Proof of Theorem 5.4. Suppose on the contrary that such an f exists. Using the f, we will design an algorithm for the halting problem, contradicting the unsolvability of the latter.

Given a Turing machine M, we construct the sentence A_M as in the Simulation Lemma. Recall that

- $alt(A_M) = 3$;
- if M halts on the empty input, then A_M defines a finite graph G_M;
- if M does not halt, then A_M has no finite model.

Write $k = qr(A_M)$ and $m = \max_{i \le k} f(i)$. Thus, if G_M exists, then $D_3(G_M) \le k$ and, by the assumption, $D_{1/2}(G_M) \le m$.

Construct U_m as in Lemma 5.6 and add to every sentence in U_m the two graph axioms. We know that U_m contains a sentence defining G_M and this will help us to construct this graph (if it exists). Remove from U_m all finitely unsatisfiable formulas. This task is tractable by Lemma 5.5. For every remaining sentence, by brute-force search we eventually find a finite graph satisfying it (we need one model for every sentence and do not care that some sentences may have other models). Let $G_1, \ldots, G_l$ be the list of these graphs.

If M halts, one of the G_i's coincides with G_M and satisfies A_M. If M does not, none of the G_i's satisfies A_M. Thus, the verification of whether A_M is true on one of the G_i's allows us to recognize whether M halts on the empty input. □

Corollary 5.7.

(1) *There is no general recursive function f such that $D_0(G) \le f(D(G))$ for all graphs G.*
(2) *There is no general recursive function f such that $D_0(G, G') \le f(D(G, G'))$ for all non-isomorphic G and G'.*

Proof. (1) Suppose on the contrary that such an f exists. Then we would have $D_{1/2}(G) \le D_0(G) \le f(D(G)) \le \max_{i \le D_3(G)} f(i)$, contradictory to Theorem 5.4.

(2) Again, suppose that such an f exists. By Proposition 3.6, $D_0(G) = D_0(G, G')$ for some G'. It follows that $D_0(G) \le f(D(G, G')) \le \max_{i \le D(G)} f(i)$, contradictory to Item 1. □

It is also worth noting the following fact.

Theorem 5.8. *$D_0(G)$ and $D_{1/2}(G)$ are computable functions of graphs.*

Proof. We prove the theorem for $D_{1/2}(G)$; for $D_0(G)$ the proof is similar. Starting from $m = 2$, we trace through the universal set U_m given by Lemma 5.6 and, for each sentence

$A \in U_m$, check whether G satisfies A and, if so, whether A is defining. The latter can be done on account of Lemma 2.7. If no such A is found, we conclude that $D_{1/2}(G) > m$ and increase m by 1. □

Remark 5.9. A variant of Theorem 5.4 for the formula length is also true, even with a simpler proof (no reference to Lemma 5.6 is needed).

5.4. *An undecidable fragment of the theory of finite graphs*

Given a class of σ-structures $\mathcal{C}$, let $Sat(\mathcal{C})$ (resp. $Sat^{=}(\mathcal{C})$) be the set of formulas over σ without equality (resp. with equality) that have a model in $\mathcal{C}$. Furthermore, let $Sat_{fin}(\mathcal{C})$ (resp. $Sat^{=}_{fin}(\mathcal{C})$) be the set of formulas over σ without equality (resp. with equality) that have a finite model in $\mathcal{C}$. If X is one of the aforementioned sets and F is a class of formulas over σ, we call the intersection $F \cap X$ the *F-fragment of X*. We will be interested in the case where F is a *prefix class*, that is, consists of prenex formulas whose quantifier prefix agrees with a given pattern. Describing such a pattern, we use $\forall^*$ or $\exists^*$ to denote a string of all $\forall$ or all $\exists$ of any length.

Let $\mathcal{D}$ (resp. $\mathcal{S}$) denote the class of structures consisting of a single binary relation (resp. symmetric binary relation). In other words, $\mathcal{D}$ is the class of directed graphs. By $\mathcal{G}$ we denote the class of graphs, i.e., structures consisting of a single irreflexive symmetric relation.

On the basis of Church and Turing's solution of Hilbert's *Entscheidungsproblem*, Kalmár [13] proved that $Sat(\mathcal{D})$ is undecidable. Following the Kalmár result and the Trakhtenbrot theorem [29], Vaught [30] proved that the set $Sat_{fin}(\mathcal{D})$ and the set of formulas not in $Sat(\mathcal{D})$ are recursively inseparable, that is, no decidable set contains the former and is disjoint with the latter. In particular, both $Sat_{fin}(\mathcal{D})$ and $Sat(\mathcal{D})$ are undecidable. Currently a complete classification of prefix fragments of $Sat(\mathcal{D})$, $Sat_{fin}(\mathcal{D})$, $Sat^{=}(\mathcal{D})$, and $Sat^{=}_{fin}(\mathcal{D})$ is known (see [1], a reference book on the subject).

Church and Quine [3] established the undecidability of $Sat(\mathcal{S})$. Note that this result is easily extended to $Sat^{=}(\mathcal{G})$ (see also [24] whose method works also for $Sat^{=}_{fin}(\mathcal{G})$). The undecidability of $Sat(\mathcal{G})$ was proved by Rogers [27]. Lavrov [16] (see also [6, Theorem 3.3.3]) improved this by showing the recursive inseparability of $Sat_{fin}(\mathcal{G})$ and the set of formulas not in $Sat(\mathcal{G})$.

Lavrov's proof provides us with a reduction of the decision problem for $\mathcal{D}$ to the decision problem for $\mathcal{G}$. If combined with the known results on undecidable fragments of $Sat_{fin}(\mathcal{D})$, this gives us some undecidable fragments of $Sat_{fin}(\mathcal{G})$, for example, $\forall^9\exists^*\forall^*\exists^*$. However, this method apparently cannot give undecidable fragments with less than two star symbols. Gurevich [9,10] proves that the $\forall^5\exists^*$-fragments of $Sat_{fin}(\mathcal{G})$ and $Sat(\mathcal{G})$ are undecidable. Our Simulation Lemma has relevance to this circle of questions.

Theorem 5.10. *For some l, m, and n, the $\exists^*\forall^l\exists^m\forall^n$-fragment of $Sat^{=}_{fin}(\mathcal{G})$ is undecidable.*

Proof. By the Simulation Lemma, a Turing machine M halts on the empty input iff the formula A_M has a finite graph as a model. Thus, the conversion of A_M to a prenex formula according to Item 4 of the Simulation Lemma reduces this variant of the halting problem to the satisfiability problem for $\exists^*\forall^l\exists^m\forall^n$-formulas over finite graphs. □

The theorem should be contrasted with the decidability of the $\exists^*\forall^*$-fragment, which follows from the Ramsey theorem and the fact that the class of graphs is definable by a $\forall^2$-formula. We do not try to specify numbers l, m, n since the values derivable from our proof are, though not so big, surely improvable by extra technical efforts. Note that a variant of the theorem for $Sat_{fin}(\mathcal{D})$ is known to be true with best possible $l = m = n = 1$ (see [1, Theorem 3.3.2], which is Surányi's theorem extended to the finite satisfiability by Gurevich).

Note another equivalent form of Theorem 5.10. Let $Th^{=}_{fin}(\mathcal{G})$ denote the *first order theory of finite graphs with equality*, i.e., the set of first order sentences with relation symbols $\sim$ and $=$ that are true on all finite graphs. Observe that a sentence A is in $Th^{=}_{fin}(\mathcal{G})$ iff $\neg A$ is not in $Sat^{=}_{fin}(\mathcal{G})$. It follows that the $\forall^*\exists^l\forall^m\exists^n$-fragment of $Th^{=}_{fin}(\mathcal{G})$ is undecidable.

6. The succinctness function over trees: upper bound

We define a variant of the succinctness function for a class of graphs $\mathcal{C}$ (with respect to the quantifier rank) by

$$q(n; \mathcal{C}) = \min \{D(G) : G \in \mathcal{C}, |G| = n\} .$$

We here prove a log-star upper bound for the class of trees.

Theorem 6.1. $q(n; trees) < \log^* n + 5$.

The proof takes the rest of this section.

6.1. Rooted trees

A *rooted tree* is a tree with one distinguished vertex, which is called the *root*. If T is a tree and $v \in V(T)$, then T_v denotes the tree T rooted at v. An isomorphism of rooted trees should not only preserve the adjacency relation but also map one root to the other. Thus, for distinct $u, v \in V(T)$, rooted trees T_u and T_v, though having the same underlying tree T, may be non-isomorphic.

An *automorphism* of a rooted tree is an isomorphism from the tree onto itself. Obviously, any automorphism leaves the root fixed. We call a rooted tree *asymmetric* if it has no non-trivial automorphisms, that is, no automorphisms except the identity.

The *depth* of a rooted tree T_v, which is denoted by $depth\, T_v$, is the eccentricity of its root. If $(v, \ldots, u, w)$ is a path in T_v, then w is called a *child* of u. We define the relation of being a *descendant* to be the transitive and reflexive closure of the relation of being a child.

If $w \in V(T_v)$, then $T_v(w)$ denotes the subtree of T_v spanned by the set of all descendants of w and rooted at w. If w is a child of $u \in V(T_v)$, then $T_v(w)$ is called a *u-branch* of T_v.

6.2. Diverging trees

We call T_v *diverging* if, for every vertex $u \in V(T_v)$, all u-branches of T_v are pairwise non-isomorphic.

Lemma 6.2. *A rooted tree T_v is diverging iff its v-branches are pairwise non-isomorphic and each of them is diverging.*

Proof. Assume that T_v is diverging. Its v-branches are pairwise non-isomorphic by the definition. Furthermore, let $T_v(w)$ be a v-branch of T_v and $u \in V(T_v(w))$. Note that any u-branch of $T_v(w)$ is also a u-branch of T_v. Therefore, all of them are pairwise non-isomorphic and $T_v(w)$ is diverging.

For the other direction, consider a non-root vertex u of T_v and let $T_v(w)$ be the v-branch of T_v containing u ($w = u$ is possible). Note that any u-branch of T_v is also a u-branch of $T_v(w)$. Therefore, all of them are pairwise non-isomorphic and we conclude that T_v is diverging. □

Lemma 6.3. *A rooted tree T_v is diverging iff it is asymmetric.*

Proof. We proceed by induction on $d = \mathit{depth}\, T_v$. The base case of $d = 0$ is trivial. Let $d \geq 1$.

Assume that T_v is diverging. By Lemma 6.2, no automorphism of T_v can map one v-branch onto another v-branch. By the same lemma and the induction assumption, no non-trivial automorphism can map a v-branch onto itself. Thus, T_v has no non-trivial automorphism.

Assume now that T_v is asymmetric. Hence all v-branches are pairwise non-isomorphic and each of them is asymmetric. By the induction assumption, each v-branch is diverging. By Lemma 6.2 we conclude that T_v is diverging. □

We now carry over the notion of a diverging tree to (unrooted) trees. Clearly, any automorphism of a tree T either leaves central vertices c_1 and c_2 fixed or transposes them ($c_1 = c_2$ if the diameter $d(T)$ is even). If $d(T)$ is odd, Lemma 6.3 implies that T_{c_1} and T_{c_2} are simultaneously diverging or not. This makes the following definition correct: a tree T is *diverging* if the rooted tree T_c for a central vertex c is diverging. It is not hard to see that T is diverging iff one of the following conditions is met:

(1) T has no non-trivial automorphism.
(2) T has exactly one non-trivial automorphism and this automorphism transposes two central vertices of T.

6.3. Spoiler's strategy

In this section we exploit the characterization of the quantifier rank of a distinguishing formula as the length of the Ehrenfeucht game (see Proposition 3.5).

Lemma 6.4. *Suppose that in the Ehrenfeucht game on (G, G') some two vertices $x, y \in V(G)$ at distance k were selected so that their counterparts $x', y' \in V(G')$ are at a strictly larger distance (possibly infinity).*

Then Spoiler can win in at most $\lceil \log k \rceil$ extra moves, playing all the time inside G.

Proof. Spoiler sets $u_1 = x$, $u_2 = y$, $v_1 = x'$, $v_2 = y'$, and places a pebble on the middle vertex u in a shortest path from u_1 to u_2 (or either of the two middle vertices if $d(u_1, u_2)$ is odd). Let $v \in V(G')$ be selected by Duplicator in response to u. By the triangle inequality, we have $d(u, u_m) < d(v, v_m)$ for $m = 1$ or $m = 2$. For such m Spoiler resets $u_1 = u$, $u_2 = u_m$, $v_1 = v$, $v_2 = v_m$ and applies the same strategy once again. Therewith Spoiler ensures that, in each round, $d(u_1, u_2) < d(v_1, v_2)$. Eventually, unless Duplicator loses earlier, $d(u_1, u_2) = 1$ while $d(v_1, v_2) > 1$, that is, Duplicator fails to preserve adjacency.

To estimate the number of moves made, notice that initially $d(u_1, u_2) = k$ and for each subsequent u_1, u_2 this distance becomes at most $f(d(u_1, u_2))$, where $f(\alpha) = (\alpha + 1)/2$. Therefore the number of moves does not exceed the minimum i such that $f^{(i)}(k) < 2$. As $(f^{(i)})^{-1}(\beta) = 2^i\beta - 2^i + 1$, the latter inequality is equivalent to $2^i \geq k$, which proves the bound. □

Note that the bound of Lemma 6.4 is tight; more precisely, it cannot be improved to $\lceil \log k \rceil - 1$. For example, let C_n denote a cycle of length n and $2C_n$ the disjoint union of two such cycles. It is known (e.g. [28, Proof of Theorem 2.4.2] or [4, Example 2.3.8]) that Duplicator can survive in the Ehrenfeucht game on C_{2k+1} and C_{2k+2} in more than $\log k + 1$ rounds for any strategy of Spoiler, in particular, when Spoiler begins with selecting two antipodal vertices in C_{2k+2}. Furthermore, if $d(x', y') = \infty$, Duplicator can be persistent as well. For example, she can survive in the game on C_{2k} and $2C_{2k}$ during $\lfloor \log(2k - 1) \rfloor$ rounds for any strategy of Spoiler, in particular, when Spoiler's first move is in one component of $2C_{2k}$ and his second move is in the other component of $2C_{2k}$ (e.g. [4, Example 2.3.8]).

Lemma 6.5. *If graphs G and G' have different diameters (including the case where G is connected and G' is disconnected), then $D_1(G, G') \leq \lceil \log d(G) \rceil + 2$.*

Proof. Assume that $d(G) < d(G')$. Spoiler begins by selecting two vertices at distance $d(G) + 1$ in G', then jumps to G, and uses the strategy of Lemma 6.4. □

Lemma 6.6. *If G is a tree, G' is a connected non-tree, and $d(G) = d(G')$, then $D_0(G, G') < \lceil \log d(G) \rceil + 4$.*

Proof. Denote $k = d(G) = d(G')$. Let C be a shortest cycle in G'. Notice that C has length at most $2k + 1$. Spoiler begins by selecting in C a vertex z' along with its neighbors x' and y'. Let z, x, and y be the corresponding responses of Duplicator in G. The vertex z cannot be a leaf of G, or else Duplicator has lost. From now on Spoiler plays all the time in $H' = G' - z'$ and Duplicator is forced to play in $H = G - z$. In these graphs $d(x', y') \leq 2k - 1$ and $d(x, y) = \infty$. Therefore the strategy of Lemma 6.4 applies and Spoiler wins in at most $\lceil \log(2k - 1) \rceil$ extra moves. □

Lemma 6.7. *Let T and T' be two non-isomorphic diverging trees with $d(T) = d(T')$ (and hence $r(T) = r(T')$). Then $D(T, T') \leq r(T) + 1$.*

Proof. In the first move Spoiler selects x, a central vertex of T. Duplicator's response, x', should be a central vertex of T' because otherwise Spoiler selects a vertex y' in T' with $d(x', y') > r(T)$ and applies the strategy of Lemma 6.4. We will denote the vertices selected by the players in T and T' during the i-th round by x_i and x'_i; in particular, $x_1 = x$ and $x'_1 = x'$. Spoiler will play so that $(x_1, \ldots, x_i)$ and $(x'_1, \ldots, x'_i)$ are always paths. Another condition that will be obeyed by Spoiler is that $T_x(x_i)$ and $T'_{x'}(x'_i)$ are non-isomorphic.

Assume that the i-th round has been played. If exactly one of the vertices x_i and x'_i is a leaf (we will call such a situation terminal), then Spoiler prolongs that path for which this is possible and wins. Assume that neither of x_i and x'_i is a leaf and that $T_x(x_i)$ and $T'_{x'}(x'_i)$ are non-isomorphic (in particular, this is so for $i = 1$). By the definition of a diverging rooted tree, all $T_x(u)$ with u a child of x_i are pairwise non-isomorphic. The same concerns all $T'_{x'}(u')$ with u' a child of x'_i. It follows that there is a $T_x(u)$ not isomorphic to any of the $T'_{x'}(u')$'s or there is a $T'_{x'}(u')$ not isomorphic to any of the $T_x(u)$'s. Spoiler selects such u for x_{i+1} or u' for x'_{i+1}. Clearly, Spoiler has an appropriate move until a terminal situation occurs. The latter occurs in the $r(T)$-th round at latest. □

Lemma 6.8. *Let T and T' be two trees with $d(T) = d(T')$ (and hence $r(T) = r(T')$). Suppose that T is diverging but T' is not. Then $D(T, T') \le r(T) + 2$.*

Proof. In the first move Spoiler selects x', a central vertex of T'. Similarly to the preceding proof, we may suppose that Duplicator's response x is a central vertex of T. Let y' be a vertex of T' such that $T'_{x'}(y')$ is not diverging but, for any child z' of y', $T'_{x'}(z')$ is. Note that y' must have two children z'_1 and z'_2 such that $T'_{x'}(z'_1)$ and $T'_{x'}(z'_2)$ are isomorphic.

In subsequent moves Spoiler selects the path $P' = (x', \ldots, y', z'_1)$. Let $P = (x, \ldots, y, z)$ be Duplicator's response in T. If $T_x(z)$ and $T_{x'}(z'_1)$ have different depths d and d', say $d > d'$, then Spoiler prolongs P with $d' + 1$ new vertices and wins. It is clear that the prolonged path has at most $r(T) + 1$ vertices.

Suppose now that $d = d'$. If $T_x(z)$ and $T_{x'}(z'_1)$ are non-isomorphic, then Spoiler adopts the strategy of Lemma 6.7 and wins having made in total at most $r(T) + 1$ moves. If $T_x(z)$ and $T'_{x'}(z'_1)$ are isomorphic, then Spoiler selects z'_2. In response Duplicator must select a child of y different from z. Denote it by z^*. The subtree $T_x(z^*)$ is non-isomorphic to $T_x(z)$ and hence to $T'_{x'}(z'_2)$. Now Spoiler is able to proceed with $T_x(z^*)$ and $T'_{x'}(z'_2)$ as was described and wins having made in total at most $r(T) + 2$ moves (one extra move was made to switch from z'_1 to z'_2). □

Lemma 6.9. *Let T be a diverging tree of radius at least 6. Then $D(T) \le r(T) + 2$.*

Proof. Let T' be a graph non-isomorphic to T. The pair T, T' satisfies the condition of one of Lemmas 6.5–6.8. These lemmas provide us with bound $D(T, T') \le r(T) + 2$. By Proposition 3.6, we thereby have the bound for $D(T)$. □

We have shown that diverging trees are definable with quantifier rank not much larger than the radius. It remains to show that, given the radius, there are diverging trees with large order and, moreover, the orders of these large trees fill long segments of integers.

Lemma 6.10. *Given $i \ge 0$, let M_i denote the total number of (pairwise non-isomorphic) diverging rooted trees of depth at most i. Then $M_i = T(i)$.*

Proof. Let m_i denote the number of diverging rooted trees of depth precisely i. Thus, $m_0 = 1$ and $M_i = m_0 + \cdots + m_i$. By Lemma 6.2, a depth-$(i + 1)$ tree T_v is uniquely determined by the set of its v-branches, which are diverging rooted trees of depth at most i. Vice versa, any set of diverging rooted trees of depth at most i with at least one tree of depth precisely i determines a depth-$(i + 1)$ tree. It follows that $m_{i+1} = (2^{m_i} - 1)2^{M_{i-1}}$, where we put $M_{-1} = 0$. By induction, we obtain $m_i = T(i) - T(i - 1)$ and $M_i = T(i)$. □

Note that a diverging rooted tree of depth i can have the minimum possible number of vertices $i + 1$ (a path).

Lemma 6.11. *Let N_i denote the maximum order of a diverging rooted tree of depth i. Then $N_i > T(i - 1)$.*

Proof. The largest diverging rooted tree T_v of depth i has every one of M_{i-1} diverging rooted trees of depth at most $i - 1$ as a v-branch. Thus, $N_i > M_{i-1} = T(i - 1)$. □

Lemma 6.12. *For every n such that $i + 1 \le n \le N_i$ there is a diverging rooted tree of depth i and order n.*

Proof. We proceed by induction on i. The base case of $i = 0$ is trivial. Let $i \ge 1$. For $n = i + 1$ we are done with a path. We will prove that any diverging rooted tree T_v of depth i except the path can be modified so that it remains a diverging rooted tree of the same depth but the order becomes 1 smaller.

Let l be the smallest depth of a v-branch of T_v and fix a branch $T_v(w)$ of this depth with minimal order. If $T_v(w)$ is a path, we delete its leaf. If not, we reduce it by the induction assumption. □

Lemma 6.13. *Let $i \ge 2$. For every n such that $2i + 2 \le n \le 2N_i$, there is a diverging tree of order n and radius $i + 1$.*

Proof. If $n = 2m$ is even, consider the diverging rooted tree T_c with two c-branches, one of order m, the other of order $m - 1$, and both of depth i (excepting the case where $n = 2i + 2$ when the smaller branch has depth $i - 1$). Such branches do exist by Lemma 6.12. If $n = 2m + 1$ is odd, we add the third single-vertex c-branch. Since the root c is a central vertex of the underlying tree, the latter is diverging. □

Proof of Theorem 6.1. Let $n > 32 = 2T(3)$ and let $i \ge 3$ be such that $2T(i) < n \le 2T(i + 1)$. By Lemma 6.11, we have $2i + 6 < n < 2N_{i+2}$. Owing to Lemma 6.13, there exists a diverging tree T of order n and radius $i + 3$. Lemma 6.9 gives $D(T) \le i + 5 < \log^* n + 5$.

For every $n \le 32$ the required bound is provided by P_n, the path on n vertices. It is not hard to derive from Lemma 6.5 that $D_1(P_n) < \log n + 3$ for all n, which satisfies our needs for n in the range. □

7. The succinctness function over trees: zero alternations

Theorem 6.1 assumes no restriction on the alternation number. We now prove an analog of this theorem for $q_0(n; trees) = \min_{|T|=n} D_0(T)$, the succinctness function over trees with the strongest restriction on the alternation number. This is somewhat surprising in view of Corollary 5.7(1) asserting that $D_0(G)$ and $D(G)$ may be very far apart from one another.

Theorem 7.1. *For infinitely many n we have $q_0(n; trees) \le 2 \log^* n + O(1)$.*

The proof takes the rest of the section.

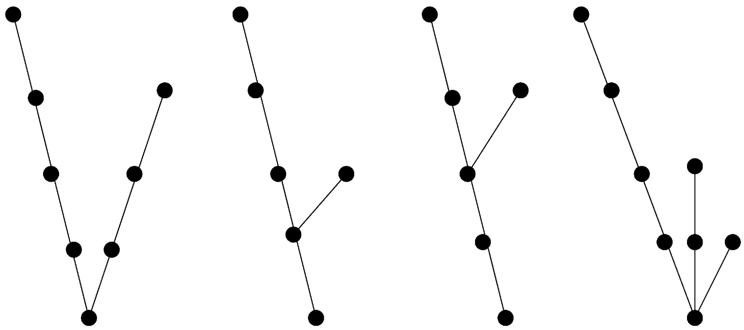

Fig. 1. R_0^*.

7.1. Ranked trees

We will modify the approach worked out in the preceding section. The proof of Theorem 6.1 was based on Lemmas 6.5–6.8. Note that the alternation number in Lemma 6.6 is 0. In Lemma 6.5 it is 1, but the bound of this lemma is actually stronger than we need and, at the cost of some relaxation, we will be able to improve the alternation number to 0 (see Lemmas 7.6 and 7.8 below). The real source of non-constant alternation number is Lemma 6.7 (Lemma 6.8 reduces to Lemma 6.7 and itself makes no new complication). To tackle the problem, we restrict the class of diverging trees so that we will still have relation $D_0(T) = O(r(T))$ and there will still exist trees with *Tower* $(r(T) - O(1))$ vertices.

We begin by introducing some notions and notation concerning rooted trees. Given a rooted tree T_v, let $B(T_v)$ denote the set of all v-branches of T_v. Given rooted trees $T_1, \dots, T_m$, we define $T = T_1 \odot \cdots \odot T_m$ to be the rooted tree with $B(T) = \{T_1, \dots, T_m\}$. By Lemma 6.2, if all T_i are pairwise non-isomorphic and diverging, then T is diverging as well. Obviously, $\mathit{depth}\, T = 1 + \max_i \mathit{depth}\, T_i$.

Let $T'_{v'}$ and T_v be rooted trees. We call $T'_{v'}$ a *rooted subtree* of T_v if $v' = v$ and $V(T') \subseteq V(T)$.

For each $i \geq 0$, we now define the class of rooted trees R_i^* as follows. Let $R_0^* = \{T_1^*, T_2^*, T_3^*, T_4^*\}$, the set of four rooted trees depicted in Fig. 1. Observe the following properties of this set.

(Z1) $|T_i^*| \leq 8$ for all i.
(Z2) $\mathit{depth}\, T_i^* = 4$ for all i.
(Z3) All T_i^* are diverging.
(Z4) No T_i^* is isomorphic to a rooted subtree of any other T_j^*.

Assume that R_{i-1}^* is already specified. We will need a large enough $F_i \subset 2^{R_{i-1}^*}$, a family of subsets of R_{i-1}^* which is an antichain with respect to the inclusion (i.e. no member of F_i is included in any other member of F_i). As one of suitable possibilities (which actually maximizes $|F_i|$ by Sperner's theorem), we fix

$$F_i = \binom{R^*_{i-1}}{\lfloor |R^*_{i-1}|/2 \rfloor},$$

the family of all $\lfloor |R^*_{i-1}|/2 \rfloor$-element subsets of R^*_{i-1}. Now

$$R^*_i = \left\{ \bigodot_{T \in S} T \,:\, S \in F_i \right\}.$$

Note that $|R^*_i| = |F_i|$.

It is clear that, if $T \in R^*_i$, then $B(T)$ consists of pairwise non-isomorphic rooted trees in R^*_{i-1}. By easy induction, we have the following properties of the class R^*_i for $i \geq 1$.

(R1) If $T \in R^*_i$, then $r(T) = \mathit{depth}\, T = i + 4$.
(R2) If $T \in R^*_i$, then $d(T) = 2i + 8$.
(R3) If $T \in R^*_i$, then the central vertex of T is equal to the root.
(R4) All $T \in R^*_i$ are diverging.
(R5) If T and T' are different members of R^*_i, then we have neither $B(T) \subset B(T')$ nor $B(T') \subset B(T)$.

We define R_i to be the set of underlying trees of rooted trees in R^*_i. Note that for different $T, T' \in R^*_i$ their underlying trees are non-isomorphic. If $i = 0$, this is evident. If $i \geq 1$, we use the fact that, as any isomorphism between the unrooted trees takes one central vertex to the other, it is also an isomorphism between the rooted trees. Note also that trees in R_i are diverging.

We will call trees in $R = \bigcup_{i=1}^{\infty} R_i$ *ranked*. If $T \in R_i$, we will say that T has *rank* i and write $\mathit{rk}\, T = i$.

Lemma 7.2. *Let N_i denote the minimum order of a tree of rank i. Then $N_i \geq T(i - O(1))$.*

Proof. Denote $M_i = |R_i|$. By the construction, we have

$$M_0 = 4, \qquad M_{i+1} = \binom{M_i}{\lfloor M_i/2 \rfloor} = \sqrt{\frac{2 + o(1)}{\pi M_i}}\, 2^{M_i},$$

and

$$N_{i+1} \geq 1 + \lfloor M_i/2 \rfloor N_i > M_i.$$

The lemma follows by simple estimation. □

7.2. Spoiler's strategy

Consider the Ehrenfeucht game on rooted trees $(T_v, T'_{v'})$. Let x_i denote the vertex of T_v selected in the i-th round. We call a strategy for Spoiler *continuous* if he plays all the time in T_v and, for each i, the induced subgraph $T[\{v, x_1, \ldots, x_i\}]$ is connected.

Lemma 7.3. *Let T_v and $T'_{v'}$ be non-isomorphic rooted trees in R^*_i. Then Spoiler has a continuous winning strategy in* $\textrm{EHR}_{i+7}(T_v, T'_{v'})$ *and hence $D_0(T_v, T'_{v'}) \leq i + 7$.*

Proof. We proceed by induction on i. In the base case of $i = 0$, Spoiler selects all non-root vertices of T_v in a continuous manner and wins by Property (Z4). Let $i \geq 1$. In the first move Spoiler selects w, a child of v such that, for any w', a child of v', branches $T_v(w)$ and $T'_{v'}(w')$ are not isomorphic. This is possible owing to Property (R5). Let w' denote Duplicator's response. Both $T_v(w)$ and $T'_{v'}(w')$ have rank $i - 1$. Spoiler now invokes a continuous strategy winning $\text{EHR}_{i+6}(T_v(w), T'_{v'}(w'))$, which exists by the induction assumption. □

Lemma 7.4. *Let T, T' be trees of the same even diameter and v, v' be their central vertices. Assume that Spoiler selects v but Duplicator responds with a vertex different from v'. Then Spoiler is able to win in the next $d(T)$ moves, playing all the time in T.*

Proof. In a continuous manner, Spoiler selects the vertices of a diametral path in T. Let $u \neq v'$ be the vertex selected by Duplicator in response to v. Duplicator should now exhibit a path of length $d(T') = d(T)$ with u at the middle, which is impossible by Proposition 2.1. □

Lemma 7.5. *Let T and T' be non-isomorphic ranked trees of the same rank. Then $D_0(T, T') \leq 2\,rk\,T + 9$.*

Proof. Let v and v' be central vertices of T and T' respectively. Spoiler starts by selecting v. If Duplicator does not respond with v', Spoiler applies the strategy of Lemma 7.4 and wins in the next $d(T)$ moves. If Duplicator responds with v', Spoiler applies the strategy of Lemma 7.3 and wins in the next $rk\,T + 7$ moves. In any case Spoiler wins in $1 + \max\{d(T), rk\,T + 7\} = 2\,rk\,T + 9$ moves. □

Lemma 7.6. *Let T be a ranked tree and G be either a tree of different diameter or a connected non-tree. Then $D_0(T, G) \leq 2\,rk\,T + 10$.*

Proof. If G is a tree, then $d(T) + 2$ moves are enough for Spoiler to win. In this case, he selects a path of length $\min\{d(T), d(G)\} + 1$ in the graph of larger diameter.

Suppose that G is a connected non-tree. If G has a cycle on at most $d(T) + 2$ vertices, Spoiler selects it and wins. Otherwise G must have a cycle on at least $d(T) + 3$ vertices. Spoiler wins by selecting a path on $d(T) + 2$ vertices of this cycle. □

Lemma 7.7. *Let T be a ranked tree and G be a non-ranked tree. If $d(T) = d(G)$, then $D_0(T, G) \leq 2\,rk\,T + 9$.*

Proof. Let v and c denote the central vertices of T and G respectively. The tree in which Spoiler plays will be specified below. In the first move Spoiler selects the central vertex of this tree. If Duplicator does not respond with the central vertex of the other tree, he loses in the next $d(T)$ moves by Lemma 7.4. Assume that she responds with the central vertex. Further play depends on which of three categories G belongs to. Let $k = rk\,T$. For any $w \in V(G)$ at distance k from c, we will call $G_c(w)$ an *apex* of G_c.

Case 1: G_c has an apex $G_c(w)$ which is not a rooted subtree of any of the four rooted trees in R_0^*. Spoiler plays in G. In the next k moves he selects the path from c to w. Duplicator is forced to select the path from v to a vertex u such that $T_v(u) \in R_0^*$. Spoiler is now able to win by selecting at most eight vertices of $G_c(w)$. The total number of moves does not exceed $1 + k + 8 = k + 9$.

Case 2: G has a vertex w such that $B(G_c(w))$ properly contains $B(H_w)$ for some $H_w \in R_i^*$, where $i = k - d(c, w)$. Spoiler plays in G. In the next $d(c, w)$ moves he selects the path from c to w. Let u denote the vertex selected by Duplicator in response to w and $F_u = T_v(u)$. Clearly, Duplicator must ensure the equality $d(v, u) = d(c, w)$ and hence $F_u \in R_i^*$.

If F_u and H_w are not isomorphic, then Spoiler restricts further play to H_w following a continuous strategy. Of course, Duplicator is forced to play in F_u. Spoiler is able to win in the next $i + 7$ moves according to Lemma 7.3.

Suppose now that F_u and H_w are isomorphic. In the next move Spoiler selects a child of w which is not in H_w. Duplicator must respond with a child of u in F_u. Denote it by x and let y be the vertex of H_w corresponding to x under the isomorphism from F_u to H_w. Recall that, by Lemma 6.3, diverging trees are asymmetric and therefore such an isomorphism is unique. In the next move Spoiler selects y. Duplicator must respond with z, another child of u in F_u. Note that $F_u(z)$ and $H_w(y)$ are not isomorphic since the latter is isomorphic to $F_u(x)$ but the former is not. From now on Spoiler restricts play to $F_u(z)$ and $H_w(y)$ using the strategy of Lemma 7.3, and wins in the next $i + 6$ moves. The total number of moves is at most $1 + d(c, w) + i + 8 = k + 9$.

Case 3: Neither 1 nor 2. Spoiler plays all the time in T. We will denote the vertices selected by him in the next k moves by $x_1, \ldots, x_k$ subsequently. Let $y_1, \ldots, y_k$ denote the corresponding vertices selected in G by Duplicator. Put also $x_0 = v$ and $y_0 = c$. Spoiler will play so that $x_0, x_1, \ldots, x_k$ will be a path. Let $1 \le i \le k$. Suppose that the preceding $x_0, \ldots, x_{i-1}$ are already selected. Assume that $T_v(x_{i-1})$ and $G_c(y_{i-1})$ are non-isomorphic (note that this is so for $i = 1$). As we are not in Case 2, x_{i-1} has a child x such that $T_v(x) \notin B(G_c(y_{i-1}))$. Spoiler takes this x for x_i thereby ensuring that $T_v(x_i)$ and $G_c(y_i)$ are non-isomorphic again, whatever y_i is selected by Duplicator. The final stage of the game goes on non-isomorphic $T_v(x_k)$ and $G_c(y_k)$. Spoiler selects all vertices of $T_v(x_k)$.

Note that $T_v(x_k) \in R_0^*$ and $G_c(y_k)$ is an apex of G. As we are not in Case 1, $G_c(y_k)$ is a rooted subtree of some $T_j^* \in R_0^*$. If $T_j^* = T_v(x_k)$, $G_c(y_k)$ must be a proper subtree of $T_v(x_k)$ and hence Spoiler has won. Otherwise, note that $T_v(x_k)$ cannot be a rooted subtree of $G_c(y_k)$ by Property (Z4). Again, this is Spoiler's win. The total number of moves equals $1 + k + 7 = k + 8$.

In any of the three cases Spoiler wins in $\max\{1 + d(T), k + 9\} = 2k + 9$ moves. □

Note that, if T is a ranked tree of rank k, then Lemmas 7.5–7.7 provide Spoiler with a winning strategy in the 0-alternation $\mathrm{E}\textsc{hr}_{2k+10}(T, G)$ whenever G is a connected graph non-isomorphic to T.

Lemma 7.8. *Let T be a ranked tree and H be a disconnected graph. Then $D_0(T, H) \le 2\,\mathrm{rk}\,T + 10$.*

Proof. We distinguish two cases.

Case 1: No component of H is isomorphic to T.

Subcase 1.1: H has a component G such that Spoiler is able to win $\mathrm{E}\textsc{hr}_{2k+10}(T, G)$ *playing all the time in G.* Spoiler plays exactly this game.

Subcase 1.2: H has no such component. In the first move Spoiler selects the central vertex of T. Suppose that Duplicator's response is in a component G of H. By Lemmas 7.5–7.7, we are either in the situation of Lemma 7.6 (with G a tree of diameter $d(G) < d(T)$) or in the situation of Lemma 7.7 (namely, in Case 3). In both situations Spoiler has a continuous winning strategy for $\mathrm{EHR}_{2k+10}(T, G)$ allowing him to play all the time in T starting from the central vertex. Spoiler applies it and wins as Duplicator is forced to stay in G.

Case 2: H has a component T' isomorphic to T. Spoiler plays in H. His first move is outside T'. Let $x \in V(T)$ be Duplicator's response. Let x' be the counterpart of x in T' (recall that ranked trees are asymmetric and hence x' is determined uniquely). Denote the central vertices of T and T' by v and v' respectively. In the second move Spoiler selects v'. If Duplicator does not respond with v, Spoiler applies the strategy of Lemma 7.4 and wins in the next $d(T)$ moves. Assume that Duplicator responds with v. Starting from the third move, Spoiler selects the vertices on the path between v' and x', one by one, starting from a child of v'. If Duplicator follows the path from v to x, she loses as x is already selected. Assume that Duplicator deviates at some point, selecting a vertex y not on the path, and let y' be the vertex on the path between v' and x' selected in this round by Spoiler. Note that the rooted subtrees $T_v(y)$ and $T'_{v'}(y')$ are non-isomorphic. Spoiler can therefore apply the continuous strategy of Lemma 7.3 and win in the next $i + 7$ moves, where $i = k - d(v, y)$. The total number of moves is at most $1+\max\{1+d(T), 1+d(x, y)+(i+7)\} = 2k+10$. □

Lemma 7.8 completes our analysis: if T is a ranked tree of rank k and G is an arbitrary graph non-isomorphic to T, then we have a winning strategy for Spoiler in the 0-alternation $\mathrm{EHR}_{2k+10}(T, G)$. By Proposition 3.6, we conclude that $D_0(T) \le 2\, rk\, T + 10$.

To complete the proof of Theorem 7.1, let T_i be a tree of rank i and order N_i as in Lemma 7.2. We have $q_0(N_i; \textit{trees}) \le D_0(T_i) \le 2i + 10 \le 2\log^* N_i + O(1)$, the latter inequality due to Lemma 7.2.

8. The succinctness function over trees: lower bound

Complementing the upper bound given by Theorem 6.1 we now prove a nearly tight lower bound on $q(n; \textit{trees})$.

Theorem 8.1. $q(n; \textit{trees}) \ge \log^* n - \log^* \log^* n - O(1)$.

It will be helpful to work with rooted trees. The first order language for this class of structures has a constant R for the root and the parent–child relation $P(x, y)$. Let T_v and T'_u be rooted trees and suppose that $T_v \equiv_k T'_u$. By Proposition 3.5, T_v and T'_u satisfy the same sentences of quantifier rank k. Then $T \equiv_k T'$ for the underlying trees. Indeed, take any sentence in the language for trees and replace the adjacency $x \sim y$ with $P(x, y) \vee P(y, x)$. We get a sentence with the same truth value in the language of rooted trees.

Let $g(k)$ be the number of $\equiv_k$-equivalence classes of rooted trees. Similarly to Lemma 3.3, we have $g(k) \le T(k + 2 + \log^* k) + O(1)$. Set

$$U(k) = \sum_{i=0}^{g(k)-1} (kg(k))^i.$$

Lemma 8.2. *Let T_v be a finite rooted tree. Then, for any $k \geq 1$, there exists a finite rooted tree T'_u with at most $U(k)$ vertices such that $T_v \equiv_k T'_u$.*

Proof of Theorem 8.1. Consider an arbitrary tree T of order n and let $k = D(T)$. Rooting it at an arbitrary vertex v, consider a rooted tree T_v. Let T'_u be as in Lemma 8.2. Thus, we have $T \equiv_k T'$ and $|T'| \leq U(k)$. By the choice of k, T and T' must be isomorphic. We therefore have

$$n \leq U(k) < (kg(k))^{g(k)} \leq T(k + \log^* k + 4) + O(1),$$

which implies $k \geq \log^* n - \log^* \log^* n - O(1)$. □

Lemma 8.2 follows from a series of lemmas.

Lemma 8.3. *Let T_v be a rooted tree and w a non-root vertex of T_v. Suppose that $T'_w \equiv_k T_v(w)$. Let T'_v be the result of replacing $T_v(w)$ by T'_w. Then $T_v \equiv_k T'_v$.*

Proof. Duplicator wins the Ehrenfeucht game on T_v, T'_v by playing it on $T_v(w)$, T'_w (since the root is a constant symbol she automatically plays root for root) and the identical vertices elsewhere. □

Lemma 8.4. *Let T_v be a rooted tree with $w_1, \ldots, w_s$ the children of the root v, and $\alpha_1, \ldots, \alpha_s$ the k-Ehrenfeucht values of the trees $T_v(w_i)$. Then the k-Ehrenfeucht value of T is determined by the α_i's.*

Proof. If T_v and T'_u have the same $\alpha_1, \ldots, \alpha_s$ we reach T'_u from T_v in s applications of Lemma 8.3. □

Lemma 8.5. *Suppose, in the notation of Lemma 8.4, that some value α appears as α_i more than k times. Let T_v^- be T_v but with only k of those subtrees. Then $T_v \equiv_k T_v^-$.*

Proof. The game has only k moves so Spoiler cannot go in more than k of these subtrees. □

Lemma 8.6. *If T_v is a representative of a given $\equiv_k$-equivalence class with minimum possible order, then each vertex of T_v has at most $kg(k)$ children.*

Proof. This easily follows from Lemmas 8.5 and 8.4 by induction on the depth. □

Lemma 8.7. *If T_v is a representative of a given $\equiv_k$-equivalence class with minimum possible order, then it has depth at most $g(k) - 1$.*

Proof. Take a longest path from the root to a leaf. If it has more than $g(k)$ vertices, it contains two vertices w and u such that u is a descendant of w and $T_v(u) \equiv_k T_v(w)$. Replacing $T_v(w)$ by $T_v(u)$, we obtain a smaller tree in the same $\equiv_k$-class. □

Lemma 8.2 immediately follows from Lemmas 8.6 and 8.7.

9. The smoothed succinctness function

Let $q(n) = q(n; all)$ denote the succinctness function for the class of all graphs. Since there are only finitely many pairwise inequivalent sentences of bounded quantifier rank,

$q(n) \to \infty$ as $n \to \infty$. We will show that $q(n)$ grows very slowly and, in a sense, irregularly. We first summarize information given by Theorems 4.2 and 6.1.

Corollary 9.1.

(1) *There is no general recursive function f such that $f(q(n)) \geq n$ for all n.*
(2) *There is no general recursive function $l(n)$ such that $l(n)$ is monotone nondecreasing, $l(n) \to \infty$ as $n \to \infty$, and $l(n) \leq q(n)$ for all n.*
(3) $q(n) < \log^* n + 5$.

Proof. (1) Note that $q(n) \leq s(n) \leq s_3(n)$. Now, if there were a general recursive function f such that $f(q(n)) \geq n$, then we would have $\max_{i \leq s_3(n)} f(i) \geq n$ contradictory to Theorem 4.2.

(2) Assume that such an $l(n)$ exists. Let $f(m)$ be the first value of i such that $l(i) > m$. Then $f(q(n)) > n$ contradictory to Item 1.

(3) As any upper bound on $q(n; \mathcal{C})$ is stronger if it is proved for a smaller class of graphs, this item is an immediate consequence of Theorem 6.1. □

Definition 9.2. We define the *smoothed succinctness function* $q^*(n)$ (for quantifier rank) to be the least monotone nondecreasing integer function bounding $q(n)$ from above, that is, $q^*(n) = \max_{m \leq n} q(m)$.

Theorem 9.3. $\log^* n - \log^* \log^* n - O(1) < q^*(n) < \log^* n + 5$.

Proof. Since the upper bound on $q(n)$ given by Corollary 9.1(3) is monotone, this is a bound on $q^*(n)$ as well. The lower bound is derivable from Lemma 3.3. This lemma states that $|Ehrv(k)| \leq T(k + 2 + \log^* k) + c$ for a constant c. Given $n > c + T(3)$, let k be such that $T(k + 2 + \log^* k) + c < n \leq T(k + 3 + \log^*(k + 1)) + c$. Assuming that n is sufficiently large, we have $k > \log^* n - \log^* \log^* n - 4$. According to Proposition 3.6, at most $|Ehrv(k)|$ graphs are definable with quantifier rank at most k. By the pigeonhole principle, there will be some $m \leq |Ehrv(k)| + 1 \leq n$ for which no graph of order precisely m is defined with quantifier rank at most k. We conclude that $q^*(n) \geq q(m) > k$ and hence $q^*(n) \geq \log^* n - \log^* \log^* n - 2$. □

We defined $q^*(n)$ to be the monotone function "closest" to $q(n)$. Notice that $q(n)$ itself lacks the monotonicity.

Corollary 9.4. *$q(i + 1) < q(i)$ for infinitely many i.*

Proof. Set $l(n) = \log^* n - \log^* \log^* n - 2$. We have just shown that $q^*(n) \geq l(n)$ for all n large enough. By Corollary 9.1(2), we have $q(n) < l(n)$ for infinitely many n. For each such n, let $m_n < n$ be such that $q(m_n) \geq l(n)$. Thus, $q(m_n) > q(n)$ and a desired i must exist between m_n and n. □

For each non-negative integer a and for $a = 1/2$, define $q_a(n) = \min_{|G|=n} D_a(G)$ and $q_a^*(n) = \max_{m \leq n} q_a(m)$. As is easily seen, Corollary 9.1(1) holds true for $q_3(n)$ as well. Note a strengthening of Corollary 9.1(3) that follows from a result in another of our papers. Let $G(n, p)$ denote a random graph on n vertices distributed so that each edge appears with probability p and all edges appear independently from each other.

Theorem 9.5 ([14]). *With probability approaching* 1 *as n goes to the infinity,*

$$D_3(G(n, n^{-1/4})) = \log^* n + O(1).$$

Corollary 9.6. $q_3(n) \le \log^* n + O(1)$ *and hence* $\log^* n - \log^* \log^* n - O(1) \le q_3^*(n) \le \log^* n + O(1)$.

10. Depth vs. length

Theorem 10.1. $L(G) \le T(D(G) + \log^* D(G) + O(1))$.

Proof. Given an Ehrenfeucht value α, let $l(\alpha)$ denote the shortest length of a formula defining α in the sense of Section 3. Define $l(k)$ to be the maximum $l(\alpha)$ over $\alpha \in Ehrv(k)$ and $l(k, s)$ the maximum $l(\alpha)$ over $\alpha \in Ehrv(k, s)$. Of course, $l(k) = l(k, 0)$. As in Section 3, $f(k, s) = |Ehrv(k, s)|$.

It is not hard to see that $L(G) \le l(D(G))$ and therefore it suffices to prove the bound $l(k) \le T(k + \log^* k + O(1))$ for all $k \ge 2$.

On account of Lemma 3.4, we have

$$l(k, k) < 18\binom{k}{2}$$

and

$$l(k, s) \le f(k, s + 1)(l(k, s + 1) + 10)$$

if $s < k$. We will use these relations along with the bounds of Lemma 3.2 for $f(k, s)$. Set $g(x) = x2^{x+1}$. A simple inductive argument shows that

$$f(k, s) \le 2^{g^{(k-s)}(9k^2)} \qquad \text{and} \qquad l(k, s) \le g^{(k-s)}(9k^2).$$

Since $g(x) \le 4^x$, we have $l(k, 0) \le T_4(k + 2 + \log^* k) \le T(k + \log^* k + O(1))$, where T_4 stands for the variant of the tower function built from 4's instead of 2's. □

Remark 10.2. Theorem 10.1 generalizes to structures over an arbitrary vocabulary. The proof requires only slight modifications.

We now observe that the relationship between the optimum quantifier rank and length of defining formulas is nearly tight.

Theorem 10.3. *There are infinitely many pairwise non-isomorphic graphs G with* $L(G) \ge T(D(G) - 6) - O(1)$.

Proof. The proof is given by a simple counting argument which can be naturally presented in the framework of Kolmogorov complexity (applications of Kolmogorov complexity for proving complexity-theoretic lower bounds can be found in [17]).

Denote the Kolmogorov complexity of a binary word w by $K(w)$. Let $\langle G\rangle$ denote the lexicographically first adjacency matrix of a graph G. Define the Kolmogorov complexity of G by $K(G) = K(\langle G\rangle)$. Notice that

$$K(G) \le L(G) + O(1).$$

By Theorem 6.1, there is a graph G_n on n vertices with

$$D(G_n) < \log^* n + 5. \tag{11}$$

The bound $K(w) < k$ can hold for less than 2^k words. It follows that for some $n \le 2^k$ we have $K(G) \ge k$ for all graphs G on n vertices. For this particular n we have

$$L(G_n) \ge \log n - O(1). \tag{12}$$

Combining (11) and (12), we see that G_n is as required. □

Of course, we could run the same argument directly with $L(G)$ in place of $K(G)$. An advantage of using the Kolmogorov complexity is in avoiding estimation of the number of formulas of length at most k.

In Section 5.1 we showed that prenex formulas are sometimes unexpectedly efficient in defining a graph. We are now able to show that, nevertheless, they generally cannot be competitive against defining formulas with no restriction on structure. Let $D^{prenex}(G)$ (resp. $L^{prenex}(G)$) denote the minimum quantifier rank (resp. length) of a closed prenex formula defining a graph G.

Theorem 10.4. *There are infinitely many pairwise non-isomorphic graphs G with $D^{prenex}(G) \ge T(D(G) - 8)$.*

Proof. Let G be as in Theorem 10.3. We have

$$L^{prenex}(G) \ge L(G) \ge T(D(G) - 6) - O(1).$$

On the other hand, by Lemma 2.4 we have

$$L^{prenex}(G) \le f(D^{prenex}(G)), \text{ where } f(x) = O(x^2 4^{x^2}).$$

It follows that

$$D^{prenex}(G) \ge \left(\frac{1}{\sqrt{2}} - o(1)\right)\sqrt{T(D(G) - 7)} \ge T(D(G) - 8),$$

provided $D(G)$ (or the order of G) is sufficiently large. □

11. Open questions

1. Let $D'(G)$ be the minimum quantifier rank of a first order sentence distinguishing a graph G from any non-isomorphic finite graph G'. Clearly, $D'(G) \le D(G)$. Can the inequality be sometimes strict?

2. Improve on the alternation number in Theorem 4.2. Note that this cannot be done with alternation number 0. By the Ramsey theorem, Turing machines cannot be simulated by 0-alternation formulas as this would contradict the unsolvability of the halting problem. In fact, we were recently able to show [22] that $q_0(n) \ge \log^* n - \log^* \log^* n - O(1)$.

3. Classify the prefix classes with respect to solvability of the finite satisfiability problem over graphs. Such a classification does exist by the Gurevich classifiability theorem

[1, Section 2.3]. In particular, can the prefix $\exists^*\forall^{O(1)}\exists^{O(1)}\forall^{O(1)}$ in Theorem 5.10 be shortened to $\exists^*\forall^{O(1)}\exists^{O(1)}$? Shortening to $\exists^*\forall^*$ is impossible due to the Ramsey theorem.

Note that for digraphs the complete classification is known (see [1] and references there). In the notation of Section 5.4, the minimal undecidable classes for $Sat^{=}_{fin}(\mathcal{D})$ are $\forall^*\exists$, $\forall\exists\forall^*$, $\forall\exists\forall\exists^*$, $\forall\exists^*\forall$, $\exists^*\forall\exists\forall$, $\exists^*\forall^{c+1}\exists$, $\forall^{c+1}\exists^*$, while the maximal decidable classes are $\exists^*\forall^*$ and $\exists^*\forall^c\exists^*$, where $c = 1$. For $Sat_{fin}(\mathcal{D})$ the classification is the same but with $c = 2$. If we consider $Sat^{=}(\mathcal{D})$ instead of $Sat^{=}_{fin}(\mathcal{D})$ and $Sat(\mathcal{D})$ instead of $Sat_{fin}(\mathcal{D})$, nothing in the classification changes. The reasons are that the maximal decidable classes have the finite model property and that the undecidability of the minimal undecidable classes is proved by reductions which preserve the finiteness of models.

4. How close to one another are $D_1(G)$ and $D_0(G)$? At least, are they recursively linked? The same question for $D(G)$ and $D_a(G)$ (for any $a = o(n)$) is also of interest. How far apart from one another can $D(G)$ and $D_1(G)$ be?

5. Estimate the succinctness function $q(n; \mathcal{C})$ for other classes of graphs (in particular, graphs of bounded degree, planar graphs). Note that Herre [11] proves the unsolvability of the first order theory of finite planar graphs with maximum degree 4. Thus, the possibility that our Theorem 4.2 has an analog for this class of graphs is not excluded.

6. Is $q(n)$ a non-recursive function? Is $D(G)$ an uncomputable function of graphs (T. Łuczak)? Of course, the former implies the latter. The same can be asked for $q_a(n)$ and $D_a(G)$ excepting $a \in \{0, 1/2\}$ (see Theorem 5.8).

7. We know that $q_3^*(n) = (1 + o(1)) \log^* n$. The cases of alternation numbers 0, 1, and 2 are open.

8. $|q(n+1) - q(n)| = O(1)$? Note that $q(n+1) - q(n) \le 1$ but this difference is negative infinitely often by Corollary 9.4.

Acknowledgment

We thank Nikolai Vereshchagin for a useful discussion.

References

[1] E. Börger, E. Grädel, Y. Gurevich, The Classical Decision Problem, Springer, 1997.
[2] J.-Y. Cai, M. Fürer, N. Immerman, An optimal lower bound on the number of variables for graph identification, Combinatorica 12 (1992) 389–410.
[3] A. Church, W. Quine, Some theorems on definability and decidability, J. Symbolic Logic 17 (1952) 179–187.
[4] H.-D. Ebbinghaus, J. Flum, Finite Model Theory, 2nd rev. edition, Springer Verlag, 1999.
[5] A. Ehrenfeucht, An application of games to the completeness problem for formalized theories, Fund. Math. 49 (1961) 129–141.
[6] Y. Ershov, I. Lavrov, A. Taimanov, M. Taitslin, Elementary theories, Uspekhi Mat. Nauk 20 (1965) 37–108 (in Russian); Russian Math. Surveys 20 (1965) 35–105 (English translation).
[7] R. Fraïssé, Sur quelques classifications des systems de relations, Publ. Sci. Univ. Alger 1 (1954) 35–182.
[8] M. Grohe, Isomorphism testing for embeddable graphs through definability, in: Proc. of the 32nd ACM Ann. Symp. on Theory of Computing, STOC, 2000, pp. 63–72.
[9] Y. Gurevich, Existential interpretation, Algebra Logic 4 (1965) 71–85.
[10] Y. Gurevich, Existential interpretation II, Arch. Math. Log. Grundl. forsch. 22 (1982) 103–120.

[11] H. Herre, Unentscheidbarkeit in der Graphentheorie, Bull. Acad. Pol. Sci., Ser. Sci. Math. Astron. Phys. 21 (1973) 201–208.
[12] N. Immerman, Descriptive Complexity, Springer-Verlag, 1999.
[13] L. Kálmar, Zurückführung des Entscheidungsproblems auf den Fall von Formeln mit einer einzigen, binären Funktionsvariablen, Compositio Math. 4 (1936) 137–144.
[14] J.H. Kim, O. Pikhurko, J. Spencer, O. Verbitsky, How complex are random graphs in first order logic?, Random Structures Algorithms 26 (2005) 119–145.
[15] S.C. Kleene, Mathematical Logic, John Wiley & Sons, 1967.
[16] I. Lavrov, Effective inseparability of the sets of identically true and finitely refutable formulae for certain elementary theories, Algebra Logika 2 (1963) 5–18 (in Russian).
[17] M. Li, P.M.B. Vitányi, An Introduction to Kolmogorov Complexity and its Applications, 2nd edition, Addison-Wesley, 1997.
[18] E. Mendelson, Introduction to Mathematical Logic, D. Van Nostrand Company, 1963.
[19] O. Ore, Theory of Graphs, American Mathematical Society, 1962.
[20] E. Pezzoli, Computational complexity of Ehrenfeucht–Fraïssé games on finite structures, in: G. Gottlob, K. Seyr (Eds.), Proc. of the CSL'98 Conf., in: Lecture Notes in Computer Science, vol. 1584, Springer-Verlag, 1999, pp. 159–170.
[21] O. Pikhurko, H. Veith, O. Verbitsky, The first order definability of graphs: Upper bounds for quantifier ranks, 2003 (submitted for publication). Available at http://arxiv.org/abs/math.CO/0311041.
[22] O. Pikhurko, J. Spencer, O. Verbitsky, Succinct definitions in first order graph theory II: No quantifier alternation, Manuscript, 2004. Available at http://arxiv.org/abs/math.LO/0405326.
[23] O. Pikhurko, O. Verbitsky, Descriptive complexity of finite structures: saving the quantifier rank, J. Symbolic Logic (in press).
[24] M.O. Rabin, A simple method for undecidability proofs and some applications, in: Proc. of the Int. Congr. on Logic, Methodology and Philosophy of Science, North-Holland, Amsterdam, 1964, pp. 58–68.
[25] M.O. Rabin, Decidability of second order theories and automata on infinite trees, Trans. Amer. Math. Soc. 141 (1965) 1–35.
[26] F. Ramsey, On a problem of formal logic, Proc. London Math. Soc. (2) 30 (1930) 264–286.
[27] H. Rogers, Certain logical reductions and decision problems, Ann. of Math. 64 (1956) 264–284.
[28] J. Spencer, The Strange Logic of Random Graphs, Springer Verlag, 2001.
[29] B. Trakhtenbrot, The impossibility of an algorithm for the decision problem for finite models, Dokl. Akad. Nauk SSSR 70 (1950) 569–572; Amer. Math. Soc. Transl. Ser. 2 23 (1963) 1–6 (English translation).
[30] R. Vaught, Sentences true in all constructive models, J. Symbolic Logic 25 (1960) 39–58.