Graph properties checkable in linear time in the number of vertices

Etienne Grandjean, Frédéric Olive

Graph properties checkable in linear time in the number of vertices

Etienne GRANDJEAN
GREYC CNRS UMR-6072, Universite de Caen, Campus II
Bd Marechal Juin, BP 5186, 14032 Caen cedex, France
tel: (+33) 2 31 56 73 34 ; fax: (+33) 2 31 56 73 30
E-mail: Grandjean@info.unicaen.fr

and

Frédéric OLIVE
LIF, Universite de Provence, CMI
39 rue Joliot Curie, 13453 Marseille Cedex 13, France
tel : (+33) 4 91 11 36 09 ; fax : (+33) 4 91 11 36 02
E-mail: olive@gyptis.univ-mrs.fr

This paper originates from the observation that many classical NP graph problems, including some NP-complete problems, are actually of very low nondeterministic time complexity. In order to formalize this observation, we define the complexity class vertexNLIN, which collects the graph problems computable on a nondeterministic RAM in time $O(n)$, where n is the number of vertices of the input graph $G = (V, E)$, rather than its usual size $|V| + |E|$. It appears that this class is *robust* (it is defined by a natural restrictive computational device; it is logically characterized by several simple fragments of existential second-order logic; it is closed under various combinatorial operators, including some restrictions of transitive closure) and *meaningful* (it contains many natural NP problems: connectivity, hamiltonicity, non-planarity,*etc*). Furthermore, the very restrictive definition of vertexNLIN seems to have beneficial effects on our ability to answer difficult questions about complexity lower bounds or separation between determinism and nondeterminism. For instance, we prove that vertexNLIN strictly contains its deterministic counterpart, vertexDLIN, and even that it does not coincide with its complementary class, co-vertexNLIN. Also, we prove that several famous graph problems (*e.g.* planarity, 2-colourability) do not belong to vertexNLIN, although they are computable in *deterministic* time $O(|V| + |E|)$.

Key Words: linear time, nondeterminism, complexity lower bounds, combinatorial problems, finite model theory, existential second-oder logic.

CONTENTS

INTRODUCTION

Except for some tradeoff space-time lower bounds results (see *e.g.* [2, 6, 14, 27]) we do not know of any proved complexity lower bounds for any natural NP problem on a general-purpose model of computation such as the Random Access Machine (RAM). We see at least two reasons. First, we know little about the relationships between nondeterminism and determinism, with the exception of some technical results such as the separation of deterministic and nondeterministic linear time for Turing machines (see [40]), but we have no idea of how to generalize this result to linear time on RAMs. Furthermore, most natural NP problems are of very low nondeterministic complexity. Grandjean [22, 23] gave evidence that most natural NP problems belong to the class NLIN, *i.e.*, are recognizable in linear time on nondeterministic RAMs. Further, the famous SAT problem can be recognized by a RAM that only uses a linear number $O(n)$ of deterministic steps and a sublinear $O(n/\log n)$ number of nondeterministic steps, where n is the size of the formula, *i.e.*, its number of variable occurrences (see [23]).

Most general-purpose lower bounds results follow from a careful analysis of the proof that the problem in concern is hard for a complexity class[1]. With regard to time bounds, it is symptomatic that, on the one hand we know some NLIN-complete problems via DTIME($O(n)$) reductions (see [41, 10, 20]), on the other hand we cannot imagine any candidate problem for completeness in the nondeterministic quadratic time class or in any similar polynomial superlinear time class. Notice that even in the PTIME class many classical problems are of very low deterministic complexity, more precisely, quasi-linear time $O(n \times (\log n)^{O(1)})$ (sorting, minimal spanning tree, single-source shortest path problems, *etc*) or in linear time (connectivity, planarity, Horn satisfiability,*etc*).

One goal of this paper is to show that some classical combinatorial problems are even easier on the nondeterministic model. It is folklore to notice that the natural certificates of many NP graph problems have size *linear in the number of vertices*. More accurately, we prove in this paper that a number of graph problems including connectivity, biconnectivity, Hamiltonicity and nonplanarity, belong to a very restricted complexity class denoted vertexNLIN. It means that they are recognizable on a nondeterministic RAM in time $O(n)$, where $n = |V|$ is the number of vertices of the input graph $G = (V, E)$ which may be much less than the size of the graph, usually defined as $n + e$, where $e = |E|$. Intuitively, any positive instance $G = (V, E)$ of such a problem, e.g., connectivity (resp. nonplanarity), has a proof, e.g. a spanning tree (resp. a subgraph homeomorphic to K_5 or $K_{3,3}$) S of size $O(n)$, $n = |V|$, which is checked in (deterministic) time $O(n)$. If the graph is not sparse, *i.e.*, $n = o(e)$, this is a nondeterministic time bound which is sublinear in the time of the graph. Notice that this can be obtained because in our computational model, the input is assumed to be separated from the workspace.

[1] For instance, from the fact that each NSPACE(n) problem is reducible to QBF (Quantified Boolean Formulas validity) in space $O(\log n)$ and time $O(n^2 \log n)$ (see [49]), it follows that QBF $\notin$ NSPACE($o(\sqrt{n/\log n})$).

More precisely, our RAM has specific read-only input registers, e.g., n^2 boolean registers $E(i,j), i,j < n$, that represent the adjacency matrix of the input graph and $O(n)$ read/write registers R_i, $i = O(n)$, each of which contains a number whose magnitude is $O(n)$. In this model, it makes sense to check nondeterministically a property, *e.g.*, connectedness, in time $O(n)$, which is much less than the input size, namely $\Theta(n^2)$; in particular, only a small part of the input, namely $O(n)$ registers, can be read in one computation.

A natural question arises –positively answered in this paper: is vertexNLIN a robust complexity class ? Since it is, in some sense, a linear time complexity class (in fact a generalization of this notion, since the reference parameter n is no longer the input size) it is useful to recall some points about such a delicate notion. In [16], Grädel argues that " *it is not clear at all what should be the* right *notion of linear time computability* " and doubts that there could be an adequate - *i.e.* intuitive and including linear time classical algorithms - and robust - *i.e.* machine-independent - definition of linear time. In [16] and [26], the authors circumvent that problem by considering some robust closures of linear time (see also [44] and [3] for other points of view). However, [45], [25], [23] and [24] introduce and study a notion of linear time, namely the deterministic (resp. nondeterministic) class DLIN (resp. NLIN) and argue that both classes are adequate and robust. In particular, DLIN (resp. NLIN) has algebraic (resp. logical) characterizations from which complete problems can be derived as shown in [45], [25] (resp.[24], [37]).

A second goal of this paper is to study a general notion of nondeterministic time complexity on RAMs and to establish its equivalent logical characterizations. More precisely, let σ be any first-order vocabulary which may include relation, function and constant symbols. We are interested in σ-problems, that means decision problems for sets of finite σ-structures. For instance, a graph problem is a σ-problem for $\sigma = \{E\}$, where E is a binary relation symbol. For any function $T : \mathbb{N} \to \mathbb{N}$, $T(n) \geq n$, let $\mathrm{NTIME}^\sigma(T(n))$ denote the class of σ-problems that are recognized by nondeterministic RAMs[2] in time $O(T(n))$, where n is the domain size of the input. Several logics appear in the paper. They are all fragments of existential second-order logic. We denote by ESO^σ this last logic. That is, ESO^σ is the class of formula of the form: $\Phi \equiv \exists\tau\phi$, where τ is a tuple of relation and function symbols of various arities, and ϕ is a first-order formula of signature $\sigma \cup \tau$. Notice that in this definition, the second order variables all stand in front of the formula (but we could give up this constraint), whereas the first-order part ϕ is not necessarily prenex. If ϕ is in prenex form and if furthermore, its variables $x_1, \ldots, x_d$ are all universally quantified, we denote $\Phi \in \mathrm{ESO}^\sigma(\forall d)$. In other terms, $\mathrm{ESO}^\sigma(\forall d)$ is the subset of ESO^σ whose formulas have the form: $\Phi \equiv \exists\tau\forall\boldsymbol{x}\psi$, where τ is a tuple of relation and function symbols of various arities, $\boldsymbol{x}$ is a d-tuple of first-order variables, and ψ is a quantifier-free formula of signature $\sigma \cup \tau$. We denote by $\mathrm{ESO}^\sigma(\text{arity } k, \forall d)$ the set of formulas in $\mathrm{ESO}^\sigma(\forall d)$ whose ESO relation and function symbols are all

[2] In this paper, the notation "NTIME" is used differently compared the previous papers : it refers to time complexity on nondeterministic RAMs, rather than nondeterministic Turing Machines.

of arity $\leq k$. Finally, we also denote by ESO^σ (resp. $\mathrm{ESO}^\sigma(\forall d)$, $\mathrm{ESO}^\sigma(\text{arity}\, k, \forall d)$) the class of σ-problems definable in these logics. In this paper we refine Fagin's characterization of NP:

$$\bigcup_d \mathrm{NTIME}^\sigma(n^d) = \mathrm{ESO}^\sigma \quad \text{(see [12])}$$

by proving the equalities

$$\mathrm{NTIME}^\sigma(n^d) = \mathrm{ESO}^\sigma(\text{arity } d, \forall d) = \mathrm{ESO}^\sigma(\forall d), \tag{1}$$

for any vocabulary σ and any integer $d > 0$. A similar result was proved in [18, 19, 24] but the new result is more general: the parameter d is now *independent* of the arities of the σ symbols. Note also that no built-in symbol is required in the ESO^σ formulas. Another aside contribution of this paper is a purely logical (machine-independent) proof of the second equality above.

Most important is the case $d = 1$ of the previous equalities: on the one hand, it shows the robustness of the class $\mathrm{vertexNLIN}^\sigma =_{\mathrm{def}} \mathrm{NTIME}^\sigma(n)$; on the other hand, it gives a logical method and complementary tools to prove that a specific problem belongs to this class. For example, [19] proves that the *ESO* "quantifier" $(\exists_{\text{lin order}} <)$ can be defined in $\mathrm{ESO}^\sigma(\forall 1)$. In this paper, we prove that other useful constructs are also definable in this restricted logic: the ancestor relation in trees and forests, any depth-first order in a graph, the transitive closure of any unary function, etc. Those logical tools are exactly what we need to prove that many combinatorial problems such as connectivity or non planarity belong to vertexNLIN.

Starting from the observation that most of our graph problems in vertexNLIN are monotone–a graph problem $\mathcal{P}$ is monotone if $G \in \mathcal{P}$ implies $G' \in \mathcal{P}$ for any extension $G' = G \cup \{\alpha\}$ where α is any new edge–we study the monotone restriction of the class vertexNLIN. Monotone NP problems were studied by Iain Stewart [46, 47, 48] who, in particular, proved some equalities that can be reformulated as follows: for any relational vocabulary σ,

$$\text{monotone-NP}^\sigma = \bigcup_d \mathrm{NTIME}^{\sigma+}(n^d) = \mathrm{ESO}^{\sigma+}. \tag{2}$$

Here $\text{monotone-NP}^\sigma$ denotes the class of monotone σ-problems in NP, $\mathrm{ESO}^{\sigma+}$ denotes the class of σ-problems definable by ESO formulas where any relation σ-symbol only occurs positively and $\mathrm{NTIME}^{\sigma+}(T(n))$ denotes the class of σ-problems computable in time $O(T(n))$ on an NRAM which rejects whenever it reads an input 0 (*i.e.* consults a tuple of an input σ-relation r which does not belong to r). In this paper, we prove the following equalities which are similar to (1) and refine (2): for any relational vocabulary σ and any integer $d > 0$,

$$\begin{aligned} \text{monotone-NTIME}^\sigma(n^d) &= \mathrm{NTIME}^{\sigma+}(n^d) \\ &= \mathrm{ESO}^{\sigma+}(\text{arity } d, \forall d) = \mathrm{ESO}^{\sigma+}(\forall d). \end{aligned} \tag{3}$$

In particular, this shows the robustness of the class monotone-vertexNLIN which includes many combinatorial problems: Hamiltonicity, connectivity, nonplanarity, etc.

Last, we study some structural properties of the class vertexNLIN. By giving a simple combinatorial method to prove that a number of graph properties are not in vertexNLIN, we demonstrate that this class is not closed under complementation and also that DLIN \ vertexNLIN is not empty, which yields the strict inclusion vertexNLIN $\subsetneq$ NLIN. The method to prove that a specific graph property $\mathcal{P}$ does not belong to vertexNLIN consists in exhibiting, for each $n \in \mathbb{N}$, a graph G_n of n vertices and a set A_n of $\Omega(n^2)$ edges that "flip-flop" property $\mathcal{P}$: this obliges every non deterministic algorithm for $\mathcal{P}$ on input G_n to read all the $\Omega(n^2)$ edges of A_n.

Let us now present a detailed plan of the paper:

- Section 1 gives some preliminaries about the computational model and the logics involved in the paper.
- In Section 2, we prove (Theorem 2.1) the logical characterization of NTIME(n^d) above mentioned: NTIME(n^d) = ESO(arity $d, \forall d$).
- In Section 3 we show the robustness of NTIME(n^d): we give alternative logical characterizations of this class (Theorem 3.1).
- Section 4 deals with monotone classes: Theorem 4.2 states the characterizations of monotone-NTIME(n^d) quoted above in (3).

In the rest of the paper, we restrict our attention to the class vertexNLIN = NTIME(n) and to the logic that characterizes it, ESO($\forall 1$).

- Section 5 provides a kind of *toolbox* of the "semantical constructions" that can be freely handled in this logic.
- Section 6 uses the previous results to prove, by purely logical means, that several well-known graph problems belong to vertexNLIN.
- In Section 7, we demonstrate that many other combinatorial problems do not belong to vertexNLIN. Then, we prove and discuss the structural properties of vertexNLIN.
- Finally, Section 8 states some conclusive remarks and open problems.

Three groups of sections can be read independently one from the others: the first one consists of Sections 2, 3 and 4, in which various logical charaterizations of NTIME(n^d) are proved; the second one consists of Sections 5 and 6, which provide logical tools used to prove that various combinatorial problems belong to vertexNLIN; the third one is Section 7, which is devoted to structural complexity.

1. PRELIMINARIES

We will often deal with *tuples* of objects. We denote them by *bold* letters. A d-tuple is said to be of *arity* d. When we want to insist on the arity of a tuple $\boldsymbol{x}$, we sometimes denote it by $\boldsymbol{x}_d$, where $d = \text{arity}(\boldsymbol{x})$.

1.1. Structures and problems

For all notations related to finite model theory, we refer to the usual conventions (see [11], for instance). Our inputs are finite first-order structures. A *signature* (or

vocabulary) σ is a finite set of relation and function symbols each of which has a fixed arity which can be zero (a 0-ary function symbol is a constant symbol). The *arity* of σ, denoted by arity(σ), is the maximal arity of its symbols. A vocabulary is *relational* if it does not contain any function symbol. When σ and τ are two disjoint signatures, we often denote by $\sigma\tau$ their union $\sigma \cup \tau$. A *structure* S of vocabulary σ, or *σ-structure*, consists of a finite domain D of cardinality $n > 1$, and, for any symbol $s \in \sigma$, an interpretation of s over D, often denoted by s for simplicity. The set of interpretations of the σ-symbols over D is called the *interpretation* of σ over D and, when no confusion results, it is also denoted σ. The *cardinality* of a structure is the cardinality of its domain. For instance, a graph or digraph (V, E) can be encoded in two natural ways:

- As a σ-structure $G = \langle D, \sigma \rangle$ where the domain is $D = V$ and where σ is reduced to a binary relation symbol interpreted on D as the edge relation E ;
- As a σ'-structure $G' = \langle D, \sigma' \rangle$ where the domain is $D = V \cup E$ and where σ' is a pair of unary function symbol $\{\text{head}, \text{tail}\}$ interpreted as follows on $V \cup E$: for each $x \in V$, $\text{head}(x) = \text{tail}(x) = x$ and for each $\alpha \in E$, $\text{head}(\alpha) = x$ and $\text{tail}(\alpha) = y$ if the edge α links the vertex x to the vertex y.

The first of those structures is called the *relational* representation of the graph (V, E) while the second is its *functional* representation.

For any signature σ, we denote by STRUC(σ) the class of (finite) σ-structures. We are interested in decision problems. A *σ-problem* is a set $\mathcal{P} \subseteq$ STRUC(σ) that is closed under isomorphism. A typical example is a set MODELS(ϕ) of the σ-structures which satisfy some fixed formula ϕ. Let σ be a relational vocabulary and $S, S' \in$ STRUC(σ). We say that S' is an *extension* of S, and we write $S \subseteq S'$, if $s^S \subseteq s^{S'}$ for each relation symbol $s \in \sigma$. A σ-problem $\mathcal{P}$ is *monotone* if it is closed under extension. The set of monotone σ-problems is denoted by Monotone$^\sigma$. Thus, a set $\mathcal{P} \subseteq$ STRUC(σ) is in Monotone$^\sigma$ if $\mathcal{P}$ is a σ-problem and if furthermore, for all $S, S' \in$ STRUC(σ) we have: $(S \in \mathcal{P}$ and $S \subseteq S') \Rightarrow S' \in \mathcal{P}$. For example, the set of connected graphs, for the above relational representation of graphs, is monotone.

1.2. Computational model and complexity classes

Our computational model is the Nondeterministic Random Access Machine with read-only input registers. A σ-NRAM (or NRAM, for short) M is designed to store an input σ-structure $S = \langle D, \sigma \rangle$, where $D = [n] =_{\text{def}} \{0, \ldots, n-1\}$ (recall that σ might contain function symbols). It consists of:

- *input registers*:
  - a register N supposed to contain the cardinality n of the input, and
  - for each σ-symbol s of arity q, and for each tuple $\boldsymbol{i} \in [n]^q$, one register $s[\boldsymbol{i}]$ supposed to store the value $s(\boldsymbol{i})$;
- $r + 1$ *special registers* (also called *accumulators*), $A, B_1, \ldots, B_r$, where $r =$ arity(σ);
- *the main memory* which consists of registers $R_0, R_1, \ldots$

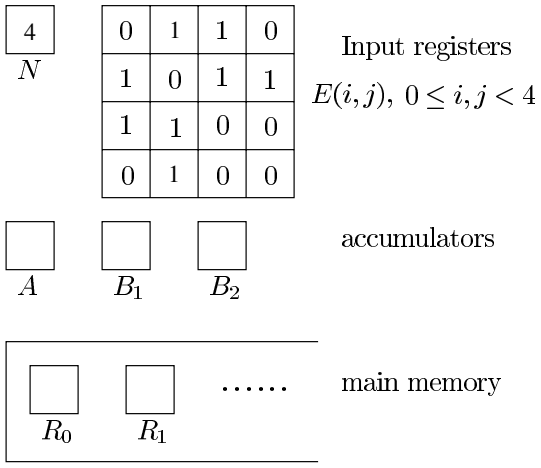

FIG. 1. an NRAM for a graph problem

Such a σ-NRAM for a graph problem ($\sigma = \{E\}$) is represented in Figure 1. Input registers are read-only. The other registers are read/write. The program of M is a sequence of labeled instructions of the following forms (1-11):

1. $A := N$
2. $A := s[B_1, \ldots, B_q]$ where $s \in \sigma$ and $q = \text{arity}(s)$
3. $A := 0$
4. $A := A + 1$
5. $A := R_A$
6. $B_i := A$, $1 \leq i \leq r$
7. $R_A := B_i$, $1 \leq i \leq r$
8. If $A = B_i$ then instr i_0 else instr i_1, $1 \leq i \leq r$
9. *guess*(A)
10. *accept*
11. *reject*

Semantics of the model:

- At the beginning of the computation, each read/write register contains 0.
- R_A denotes the register R_i whose address, i, is the content of accumulator A.
- Instruction (9), *guess*(A), is our nondeterministic instruction: it stores any integer in A.

M *accepts* a σ-structure S if some computation of M on input S reaches the statement (10): *accept*. A σ-problem $\mathcal{P}$ belongs to NTIME$^{\sigma}(T(n))$ if there is a σ-NRAM M such that:

(i) $\mathcal{P}$ is the set of σ-structures accepted by M;

(ii) each computation of M on every input σ-structure S, only uses integers in $O(T(n))$ and stops within time $O(T(n))$, where n is the cardinality of S.

We will focus on the class $\mathrm{NTIME}^{\sigma}(n)$. When σ is unary and contains at least one unary function symbol, the size and the cardinality (*i.e.* size of the domain) of σ-structures are linearly related, since a unary function $f : [n] \to [n]$ can be described by its list of values, $f(0), \ldots, f(n-1)$. Therefore, in this case, $\mathrm{NTIME}^{\sigma}(n)$ coincides with the class NLIN defined in [24] (see also [45, 25]). Otherwise (*i.e.* if $\mathrm{arity}(\sigma) \geq 2$), the class $\mathrm{NTIME}^{\sigma}(n)$ gathers problems recognizable nondeterministically in time linear in the cardinality of the input structure, but sublinear in its size. Notably, if E is a binary relation symbol, $\mathrm{NTIME}^{E}(n)$ is the class of graph and digraph problems recognizable nondeterministically in time linear in the number of vertices of an input graph given by its relational representation, that is, represented by its adjacency matrix. For this reason, we call vertexNLIN this class of graph problems. By analogy, we will often denote by $\mathrm{vertexNLIN}^{\sigma}$ the class $\mathrm{NTIME}^{\sigma}(n)$ when $\mathrm{arity}(\sigma) \geq 2$.

Let σ be a relational vocabulary. A positive σ-NRAM M is a σ-NRAM for which any access to the input σ-structure should be *positive*. More precisely, instruction (2) is replaced by

(2+) If $r(B_1, \ldots, B_q)$ then instr i else reject;

where $r \in \sigma$. Of course, any σ-problem accepted by a positive σ-NRAM is monotone. Such a machine is very similar to the so-called *Conjunctive Random Access Turing Machine* of Iain Stewart [46]. Complexity classes $\mathrm{NTIME}^{\sigma+}(T(n))$ and $\mathrm{vertexNLIN}^{\sigma+}$ are defined accordingly.

Remark.

- One may imagine that our instruction "*guess*(A)" which puts any integer into A is too powerful. It is not the case since, as it can be easily shown, such an $O(T(n))$-time bounded σ-NRAM can be simulated within the same time by another σ-NRAM which guesses integers in $O(T(n))$ and, more generally, only uses integers of magnitude $O(T(n))$ (as addresses and register contents).
- Our complexity classes $\mathrm{NTIME}^{\sigma}(T(n))$ (resp. $\mathrm{NTIME}^{\sigma+}(T(n))$) are computationally robust. More precisely, they are invariant under many changes of the allowed set of instructions. E.g., they do not change if we allow not only incrementation by 1 (instruction (4)) but also addition, substraction and/or multiplication of register contents, provided all the integers manipulated by an $O(T(n))$-time bounded NRAM are required to be of magnitude $O(T(n))$ (for details, see, *e.g.*, [21, 25]).

In this paper, we also occasionally use the deterministic RAM model and deterministic time complexity (see Section 7). A σ-RAM is similar to a σ-NRAM up to the following two changes:

- it is deterministic. That is, it does not use the nondeterministic instruction (9);
- it may perform additions. That is, the instruction $A := A + B_i$, $1 \leq i \leq r$, is allowed.

A σ-problem belongs to $\mathrm{DTIME}^{\sigma}(T(n))$ if it is recognized by a σ-RAM that uses only integers in $O(T(n))$ and stops within time $O(T(n))$.

1.3. Logic and definability classes

We use the usual definitions and notations in logic and finite model theory (see [11]). We are interested by definability in *Existential Second Order logic* (ESO). Given two disjoint signatures $\sigma = \{s_1, \ldots, s_p\}$ and $\tau = \{t_1, \ldots, t_q\}$ (where the s_i's and the t_i's are relation and function symbols of various arities), we write $\Phi \equiv \exists\tau\phi(\sigma, \tau)$ to mean that Φ has the form $\exists t_1 \ldots \exists t_q \phi(s_1, \ldots, s_p, t_1, \ldots, t_q)$. For a vocabulary σ, we denote by ESO^σ the class of σ-formulas Φ of the form

$$\Phi \equiv \exists\tau\phi(\sigma, \tau)$$

where τ is a signature disjoint from σ and ϕ is a first-order sentence of vocabulary $\sigma\tau$. For simplicity, we confuse in our notation a class of σ-formulas, *e.g.* ESO^σ, and the class of σ-problems $\mathcal{P}$ they define. Namely, $\mathcal{P} \in \mathrm{ESO}^\sigma$ means that there exists $\Phi \in \mathrm{ESO}^\sigma$ such that $\mathcal{P} = \mathrm{MODELS}(\Phi)$, *i.e.*, $S \in \mathcal{P}$ iff $S \models \Phi$. As in [18, 20, 24], we are interested in syntactic restrictions of ESO^σ:

- $\Phi \in \mathrm{ESO}^\sigma(\mathrm{var}\ d)$ means that the first-order part ϕ of Φ (which is not necessarily in prenex form) contains at most d (first-order) variables which may be quantified several times;
- $\Phi \in \mathrm{ESO}^\sigma(\forall d)$ means that Φ is in the Skolemized prenex form $\exists\tau\forall\boldsymbol{x}\phi(\sigma, \tau, \boldsymbol{x})$ where ϕ is quantifier-free, $\boldsymbol{x}$ is a tuple of first-order variables, and $\mathrm{arity}(\boldsymbol{x}) = d$;
- $\Phi \in \mathrm{ESO}^\sigma(\mathrm{arity}\, k, \forall d)$ if it fulfils the same conditions as above and if furthermore, $\mathrm{arity}(\tau) \leq k$.

Remark. Clearly, the class of problems defined in $\mathrm{ESO}^\sigma(\forall d)$ is not modified if formulas of the more liberal form

$$\Phi : \exists\tau\forall\boldsymbol{x}_d\exists\boldsymbol{y}\phi(\sigma, \tau, \boldsymbol{x}_d, \boldsymbol{y})$$

are allowed, since, via Skolemization of the existential variables $\boldsymbol{y}$, one obtains a formula of the required form. So, for convenience, we shall write formulas in the more liberal form above.

For a relational vocabulary σ, we denote by $\mathrm{ESO}^{\sigma+}$, $\mathrm{ESO}^{\sigma+}(\mathrm{var}\ d)$, $\mathrm{ESO}^{\sigma+}(\forall d)$ and $\mathrm{ESO}^{\sigma+}(\mathrm{arity}\, k, \forall d)$ the similar classes of σ-formulas where each (relation) symbol of σ only occurs positively, *i.e.*, in the scope of an even number of negations (assuming that the only connectives are $\neg$, $\wedge$ and $\vee$).

2. A LOGICAL CHARACTERIZATION OF NTIME(n^d)

The following equality refines Fagin's Theorem [12] and generalizes results of [18, 19, 24]:

THEOREM 2.1. *For any signature σ and any integer $d > 0$:*

$$\mathrm{NTIME}^\sigma(n^d) = \mathrm{ESO}^\sigma(\mathit{arity}\, d, \forall d).$$

Proof. The two corresponding inclusions are proved in Lemma 2.1 and Proposition 2.1.❑

LEMMA 2.1. $\mathrm{ESO}^{\sigma}(\mathit{arity}\, d, \forall d) \subseteq \mathrm{NTIME}^{\sigma}(n^d)$.

Proof. Let ρ be a signature of arity d and $\phi \equiv \exists\rho\forall \boldsymbol{x}_d \psi(\boldsymbol{x}_d)$ be a formula of $\mathrm{ESO}^{\sigma}(\mathrm{arity}\, d, \forall d)$. The following nondeterministic algorithm $\mathcal{A}$ clearly recognizes $\mathcal{P} = \mathrm{MODELS}(\phi)$.

Algorithm $\mathcal{A}$: On any input $([n], \sigma) \in \mathrm{STRUC}(\sigma)$,

(a) guess an interpretation over $[n]$ of each relation or function symbol in ρ ;

(b) if the expanded structure $([n], \sigma, \rho)$ satisfies $\forall \boldsymbol{x}_d \psi(\boldsymbol{x}_d)$ then *accept*, else *reject*.

Clearly, an $NRAM^{\sigma}$ M can implement (a) using $O(n^d)$ *guess* instructions (9) that guess the $O(n^d)$ values of the d-ary functions in ρ. Those values are easily stored in $O(n^d)$ registers R_i of the main memory. Then, M executes (b) within $O(n^d)$ deterministic steps: more precisely, for each $\boldsymbol{a} \in [n]^d$, $([n], \sigma, \rho) \models \psi(\boldsymbol{a})$ is checked in constant time since $\mathit{length}(\psi)$ is fixed and since M can evaluate each subterm or atomic subformula $s(\boldsymbol{b})$ ($s \in \sigma \cup \rho$, $\boldsymbol{b} \subseteq \boldsymbol{a}$) in one step with exactly one access to the input (if $s \in \sigma$) or to the main memory (if $s \in \rho$).❑

We now prove the converse inclusion: $\mathrm{NTIME}^{\sigma}(n^d) \subseteq \mathrm{ESO}^{\sigma}(\mathrm{arity}\, d, \forall d)$. As usual, in order to describe a computation, it is crucial to have a linear order $<$, which intuitively encodes the time order. It is easy to express the existence of such a linear order in $\mathrm{ESO}^{\sigma}(\mathrm{arity}\, d, \forall d)$, for $d \geq 2$:

- for $d \geq 3$, we simply have to write that there exists a binary relation which is antisymmetric, transitive and linear: this can be done using at most three universally quantified first-order variables.
- For $d = 2$, the above mentioned defining properties of linear orders are not directly expressible. However, we can quite easily overcome this difficulty by expressing that there exists, in addition to the binary relation $<$, a unary function succ that is forced to be a *successor* function while $<$ is compelled to coincide with the transitive closure of succ. It is possible to express the second constraint using only two variables, by hand of an inductive argument (related to succ).

The unary arity of succ and the possibility to define $<$ from succ by an inductive argument allow to write these constraints with only *two* first-order variables.

However, the case $d = 1$ is much more difficult. It was solved in [19] as follows: with only one first-order variable, there is no more hope to force explicitly a binary relation to be a linear order. On the other hand, there exists a first-order formula $\mathit{built}(\nu)$ over a unary signature ν whose models can be easily equipped with a linear order *implicitly* defined via an existential first-order formula $\mathit{order}(x, y)$ of arity 2. Consequently, an assertion of the form: "there exists a linear order $<$ on the domain D such that $D \models \Phi(<)$", where $\Phi(<)$ is a formula involving $<$, can be rephrazed: "there exists an interpretation of ν over the domain D such that $\langle D, \nu\rangle \models \mathit{built}(\nu) \wedge \Phi'(\nu)$", where Φ' is obtained from Φ by replacing each atomic formula $t_1 < t_2$ by $\mathit{order}(t_1, t_2)$.

In order to allow a better understanding of the way these two formulas help to express in ESO(arity $1, \forall 1$) the existence of a linear order, we recall the original result (in a slightly different formulation):

LEMMA 2.2 (Grandjean [19]). *There exist two first-order formulas built(ν) and order(x, y, ν) over a unary vocabulary ν such that:*

1. *"built" is a sentence of the form $\forall x \phi(x, \nu)$, where ϕ is quantifier-free;*
2. *"order" is a formula with two free variables x, y, of the form $\exists z \psi(x, y, z, \nu)$, where ψ is quantifier-free;*
3. *"built" has exactly* one *model (up to isomorphism) in each cardinality and*
4. *on each model $\langle D, \nu \rangle$ of built(ν), order(x, y) defines a linear order. That is:*

$$\begin{array}{c} \textit{if } \langle D, \nu \rangle \models \textit{built}(\nu), \textit{ then} \\ \{(a, b) \in D^2 \textit{ s.t. } \langle D, \nu \rangle \models \textit{order}(a, b)\} \textit{ is a linear order of } D. \end{array}$$

Proof. See [19], Proposition 4 for 3. and Lemma 5 for 4.❑

COROLLARY 2.1 (Grandjean [19]). *The definability class $ESO^{\sigma}(arity\, 1, \forall 1)$ is not enlarged by the addition of the second-order quantifier $(\exists_{\text{lin order}} <)$ (to be read: "there exists a linear order $<$ of the domain such that..."). More precisely, any formula Ψ of the form*

$$(\exists_{\text{lin order}} <) \Phi(\sigma, <),$$

where $\Phi \in \text{ESO}^{\sigma, <}(arity\, 1, \forall 1)$ is equivalent to a formula $\Psi' \in \text{ESO}^{\sigma}(arity\, 1, \forall 1)$.

Proof. Without loss of generality, assume that symbol $<$ only occurs positively in Φ. Define formula $\Psi' \equiv \exists \nu : \textit{built}(\nu) \wedge \Phi'(\sigma, \nu)$, where Φ' is obtained from Φ by replacing each inequality $t_1(x) < t_2(x)$ by the existential formula $\textit{order}(t_1(x), t_2(x), \nu)$. Clearly, by Conditions 3. and 4. of Lemma 2.2, Ψ' is logically equivalent to Ψ. Finally, the form of formulas *built* and *order* (see Conditions 1. and 2. of Lemma 2.2) guarantees that Ψ' can be written in $\text{ESO}^{\sigma}(\text{arity}\, 1, \forall 1)$, by Skolemization.❑

Using the quantifier $(\exists_{\text{lin order}} <)$, we shall express an $\text{NTIME}^{\sigma}(n^d)$ computation in $\text{ESO}^{\sigma}(\text{arity}\, d, \forall d)$ even for $d = 1$:

PROPOSITION 2.1. $\text{NTIME}^{\sigma}(n^d) \subseteq \text{ESO}^{\sigma}(arity\, d, \forall d)$.

Proof. In order to avoid heavy notations, let us give the proof for $d = 1$, the general case being similar. It is reminiscent of a similar proof in [17] and [24] (see also [45]). Let $\mathcal{P}$ be a σ-problem (*i.e.*, a set of σ-structures which is closed under isomorphism) recognized by a σ-NRAM M in time $O(n)$. Without loss of generality, assume that M only uses integers (addresses and registers) smaller than cn and always stops in time *at most* cn, for a fixed integer $c > 1$; in particular, the instants (resp. steps) of a computation are exactly numbered $0, 1, \cdots, cn - 1$ and a final instruction (*accept* or *reject*) is performed at step $cn - 1$ or before (if it is

performed before, it is repeated till the step $cn-1$). So, it is natural to encode such a computation over a linearly ordered structure with domain $[cn]$ as described below. At the end of the proof, we describe how to adapt the encoding for the smaller domain $[n]$.

Let $instr_0, instr_1, \cdots, instr_k$ denote the sequence of the instructions of the program of M. Without loss of generality, assume that $instr_k$ is the only *accept* instruction. In the encoding, we will use the successor function $Succ$, associated to the linear order $<$ and also denoted $Succ(x) = x+1$, and the associated constants $0, 1, 2, \ldots, k$ and $max = cn-1$ (assume $k < cn$), and also the constant n. Additionally to the input relation and function symbols $s \in \sigma$ defined in $[n]$ and arbitrarily extended to $[cn]$, *e.g.* with zero values, we encode a computation of M on input structure $([cn], (s)_{s\in\sigma})$ by the following new unary functions

$$I, A, (B_i)_{i\leq r}, R_A, R'_A : [cn] \to [cn]$$

which, with the exception of R'_A, are intended to describe the situation of M at instant t, that is the instant *before* step t is performed. More precisely:

- $I(t)$ holds the current instruction number (*e.g.* $I(0) = 0$ and $I(max) = k$);
- $A(t)$ and $B_i(t)$, $i = 1, 2, \ldots, r$, hold the current values of registers A and B_i, respectively (*e.g.* $A(0) = B_i(0) = 0$);
- $R_A(t)$ holds the current value of the register whose address is currently contained in register A (*e.g.* $R_A(0) = 0$) and
- $R'_A(t)$ holds the value of the same register *after* step t.

By case distinction according to the value of $I(t)$, most of the logical description of the computation of M is straightforward. *E.g.*, if M performs a statement $A := A+1$ at step t, then the formula will force $A(t+1) = A(t)+1$. The main complication arises for the instruction $A := R_A$ which loads into the accumulator A the content of the register whose address was contained in A before the execution of the instruction. Note that the functions defined in that way do *not* explicitly encode the values of all the memory registers of M at each instant but only the content of the register to which A points. So, how can we get the right value of $R_A(t)$? If the register pointed to by A at instant t has never been visited before t, then it contains its initial value : 0. Otherwise, let u be the last instant before t when A contained the value $A(t)$. Then the last modification of the register pointed to by A at instant t has been performed during step u, and $R_A(t) = R'_A(u)$. In other terms, either there is no $i < t$ such that $A(i) = A(t)$, and $R_A(t) = 0$ (case 1) or $u = \mathsf{max}\{i < t : A(i) = A(t)\}$ exists, and $R_A(t) = R'_A(u)$ (case 2). Now, it is essential to notice that the alternative between these two cases can be rephrazed as follows: assume we have lexicographically ordered the pairs $(A(t), t)$, $t \in [cn]$. Then, either $t = 0$, and therefore, $(A(t), t) = (0, 0)$ and $R_A(t) = 0$, or there exists $u < t$ such that $(A(u), u)$ is the predecessor of $(A(t), t)$ for the lexicographic order over the pairs $(A(i), i)$. In this case, *case 1 occurs if* $A(t) \neq A(u)$*; case 2 occurs otherwise.*

Let us number the cn ordered pairs $(A(t), t)$, $t \in [cn]$ according to their lexicographical order, in other words, $Lex(y) = (A(t), t)$, $y \in [cn]$, means the y^{th} ordered pair is $(A(t), t)$. It is now obvious that if $y = x+1$, and $Lex(x) = (A(u), u)$,

then u is the instant concerned in the above cases (1,2). The essential point is that t is the only universally quantified first-order variable in that description. The other variables involved x, y and u are existentially quantified, and, hence, can be Skolemized. Formally, the function $Lex : [cn] \to [cn]^2$ is represented by two unary functions $Lex_1, Lex_2 : [cn] \to [cn]$. Now we present the first-order formulas (with only one universally quantified variable) whose conjunction will form the first-order part of the $\mathrm{ESO}^\sigma(\mathrm{arity}\,1, \forall 1)$-formula. In order to simplify notation and obtain a natural encoding of an accepting computation of M, we introduce the formulas in an informal way that uses case distinction as in [45] and [25] (but these formulas are essentially equivalent to those of [18] and [24]). Recall that the function I only takes a fixed number of values $0, 1, \ldots, k$. For convenience, we freely use abbreviations for instruction case distinction, *e.g.*, "$I(t)$ is $A := 0$" is an abbreviation for the formula

$$I(x) = i_1 \vee I(x) = i_2 \vee \cdots \vee I(x) = i_p,$$

where the i_j are all the numbers of the instructions $A := 0$ in the program. In all the following "formulas" that define I, A, B_i, $i \leq r$, the quantification $(\forall t < max)$ is implicit and we use informal expressions that are easily encoded in logic such as "if", "and" and "otherwise". G denotes a new unary function symbol existentially quantified (for the *guess* instruction). Recall that instructions of the form (8.) in an NRAM are as follows: *"If $A = B_i$ then instr i_0 else instr i_1"* (see subsection 1.2). Then $I(0) = 0$, $A(0) = 0$, $B_i(0) = 0$ (for each i) and for all t:

$$I(t+1) = \begin{cases} i_0 & \text{if } I(t) \text{ is of the form (8) and } A = B_i \\ i_1 & \text{if } I(t) \text{ is of the form (8) and } A \neq B_i \\ I(t) & \text{if } I(t) \text{ is of the form (10) or (11) (\textit{accept} or \textit{reject})} \\ I(t)+1 & \text{otherwise} \end{cases}$$

$$A(t+1) = \begin{cases} n & \text{if } I(t) \text{ is } A := N \\ F(B_1(t), \ldots, B_q(t)) & \text{if } I(t) \text{ is } A := F(B_1, \ldots, B_q) \\ 0 & \text{if } I(t) \text{ is } A := 0 \\ A(t)+1 & \text{if } I(t) \text{ is } A := A+1 \\ R_A(t) & \text{if } I(t) \text{ is } A := R_A \\ G(t) & \text{if } I(t) \text{ is } \textit{guess}(A) \\ A(t) & \text{otherwise} \end{cases}$$

$$B_i(t+1) = \begin{cases} A(t) & \text{if } I(t) \text{ is } B_i := A \\ B_i(t) & \text{otherwise} \end{cases}$$

The following formula, now quantified by $\forall t$, defines the value $R'_A(t)$ contained in the register of address $A(t)$ *after* step t:

$$R'_A(t) = \begin{cases} B_i(t) & \text{if } I(t) \text{ is } R_A := B_i \\ R_A(t) & \text{otherwise} \end{cases}$$

There remains to define the functions Lex (in fact, Lex_1 and Lex_2) and R_A. Lex is obviously defined by the two formulas $\forall t \exists x : Lex(x) = (A(t), t)$ and $(\forall x <$

$max) : Lex(x) < Lex(x+1)$ and R_A is defined by

$$R_A(0) = 0 \ \wedge \ (\forall t > 0)\exists u \exists x \left(\begin{array}{l} \mathrm{Lex}(x) = (A(u), u) \wedge \\ \mathrm{Lex}(x+1) = (A(t), t) \wedge \\ (A(t) = A(u) \rightarrow R_A(t) = R'_A(u)) \wedge \\ (A(t) > A(u) \rightarrow R_A(t) = 0) \end{array} \right)$$

Finally, the formula $I(max) = k$ expresses that the last intruction performed is $instr_k = accept$. So, we have proved that our σ-problem $\mathcal{P}$ is defined on domain $[cn]$ by a formula ϕ of the form

$$(\exists_{\text{lin order}} <)\exists \tau \forall x : \psi(x).$$

Here ψ is a quantifier-free formula that uses, in addition to τ, the symbols $<$, $Succ$, 0, max and n, while τ is a unary vocabulary including I, A, $(B_i)_{i \leq r}$, R_A, R'_A, G and Lex. Of course, $Succ$, 0 and max are easily definable with $<$ and hence can be eliminated.

It remains to explain how to modify the relations, functions, constants, and the formula if the domain is $[n]$ instead of $[cn]$. For convenience, assume that each atomic subformula involving any unary function symbol F is of the form $F(u) = v$ where u, v are individual variables, and that the first-order part of our formula is of the form

$$\forall t \exists \boldsymbol{x} \psi(t, \boldsymbol{x}),$$

where ψ is quantifier-free and $\boldsymbol{x}$ is a tuple of variables. Each element $b \in [cn]$ is naturally represented by the ordered pair (i, a) such that $b = i \times n + a$, $i \in [c]$, $a \in [n]$. The rest of the encoding is a consequence of that representation:

- The universally quantified variable $\forall t$ is replaced by $\bigwedge_{i \in [c]} \forall t$ (which intuitively means $\forall (i, t) \in [c] \times [n]$), and the existential part of the prefix, $\exists x$ is modified similarly;
- The constant n, that is $1 \times n + 0$, is replaced by the ordered pair $(1, 0)$ and the linear order $<$ is encoded similarly;
- Each atomic subformula involving an input relation or function symbol $s \in \sigma$ is not modified (recall that in such a formula, the arguments are forced to belong to $[n]$);
- Each atomic subformula of the form $F(u) = v$ where F is an ESO unary function symbol symbol ($F = I, A, \ldots$) and u, v are any variables, is replaced by the conjunction

$$R^F_{i,j}(u) \wedge F_i(u) = v$$

which intuitively means $F(i, u) = (j, v)$ and in which $R^F_{i,j}$ and F_i are new unary relation and function symbols, respectively, with $i, j \in [c]$. To enforce the functional nature of $R^F_{i,j}$, we finally make the conjunction of the first-order sentence so modified and of some sentences which mean that for every $i \in [c]$ and $u \in [n]$, $R^F_{i,j}(u)$ holds for exactly one $j \in [c]$.

The details of the encoding are left to the reader. By Corollary 2.1, this finishes the proof that $\mathcal{P}$ belongs to $\mathrm{ESO}^{\sigma}(\mathrm{arity}\,1, \forall 1)$.❑

3. OTHER LOGICAL CHARACTERIZATIONS

It is natural to ask how robust is the computational/logical class $\mathrm{NTIME}^{\sigma}(n^d) = \mathrm{ESO}^{\sigma}(\mathrm{arity}\,d, \forall d)$ from a logical point of view, *i.e.* to look for other logical characterizations of this class. *E.g.*, is it equal to $\mathrm{ESO}^{\sigma}(\mathrm{arity}\,d)$, the similar class when the number of first-order (universal) variables is no longer bounded? We cannot answer this question, which is related to a conjecture by Fagin about the arity hierarchy [13]. However, we can prove the following result:

THEOREM 3.1. *For every vocabulary σ and every integer $d > 0$:*

$$\mathrm{ESO}^{\sigma}(\mathit{arity}\,d, \forall d) = \mathrm{ESO}^{\sigma}(\forall d) = \mathrm{ESO}^{\sigma}(\mathit{var}\ d).$$

The inclusions $\mathrm{ESO}^{\sigma}(\mathrm{arity}\,d, \forall d) \subseteq \mathrm{ESO}^{\sigma}(\forall d) \subseteq \mathrm{ESO}^{\sigma}(\mathrm{var}\ d)$ are trivial. The converse inclusions are proved in the following Lemma 3.1 and Proposition 3.1. Before proving Lemma 3.1, it seems useful to examine an example. Let us consider the formula

$$\Phi \equiv \exists S, T : \forall x S(x) \vee \forall x T(x),$$

where S, T are unary relation symbols. Clearly, Φ belongs to $\mathrm{ESO}^{\sigma}(\mathrm{var}\ 1)$. And putting Φ under prenex form in a natural way would provide the formula $\exists S, T \forall x, y : S(x) \vee T(y)$, which belongs to $\mathrm{ESO}^{\sigma}(\forall 2)$. Nevertheless, we can build an $\mathrm{ESO}^{\sigma}(\forall 1)$-formula equivalent to Φ. First, let us quote all the subformulas of the first-order part of Φ:

$$\alpha(x) \equiv S(x);\ \beta \equiv \forall x S(x);$$
$$\gamma(x) \equiv T(x);\ \delta \equiv \forall x T(x);$$
$$\phi \equiv \forall x S(x) \vee \forall x T(x).$$

Now, we associate to these formulas some new relation symbols $R_\alpha, \ldots, R_\phi$ and we define the formulas $\Delta_\alpha, \ldots, \Delta_\phi$ as follows:

$$\Delta_\alpha \equiv \forall x : R_\alpha(x) \to S(x);\ \Delta_\beta \equiv \forall x : R_\beta \to R_\alpha(x);$$
$$\Delta_\gamma \equiv \forall x : R_\gamma(x) \to T(x);\ \Delta_\delta \equiv \forall x : R_\delta \to R_\gamma(x);$$
$$\Delta_\phi \equiv R_\phi \to R_\beta \vee R_\delta.$$

It is easy to see that the formula $\Phi' \equiv \exists S, T \exists R_\alpha, \ldots, R_\phi : R_\phi \wedge \Delta_\alpha \wedge \Delta_\beta \wedge \Delta_\gamma \wedge \Delta_\delta \wedge \Delta_\phi$ is equivalent to Φ and has a prenex form in $\mathrm{ESO}^{\sigma}(\forall 1)$. (Notice furthermore that we could get rid of 0-ary predicates by replacing each such predicate P_0 by the atom $P_1(c)$, where P_1 is a new unary predicate and c any existentially quantified constant.) The next lemma generalizes this construction.

LEMMA 3.1. $\mathrm{ESO}^{\sigma}(\mathit{var}\ d) \subseteq \mathrm{ESO}^{\sigma}(\forall d)$.

Proof. The proof looks like the proof of a similar but less general result of [17] (Proposition 2.4). Let $\mathcal{P}$ be a σ-problem in $\mathrm{ESO}^{\sigma}(\mathrm{var}\ d)$, *i.e.* we have $\mathcal{P} = \mathrm{MODELS}(\Psi)$ for a formula of the form $\Psi \equiv \exists\tau\psi(\sigma,\tau)$, where τ is any signature and ψ is a first-order $\sigma \cup \tau$-sentence with exactly d individual variables $\boldsymbol{x} = x_1$, $x_2, \ldots, x_d$ which may be quantified several times. Without loss of generality, let us assume that ψ contains only the connectives $\wedge$, $\vee$ and $\neg$ such that no quantifier is in the scope of a negation. We also assume that ψ contains no existential quantifier (existential variables can be Skolemized). We have to transform ψ into prenex form with also d (universal) variables. The key observation is that any subformula $\theta(\boldsymbol{u})$ of ψ contains at most d free variables $\boldsymbol{u} \subseteq \boldsymbol{x}$, $\boldsymbol{u} = u_1, \ldots, u_k$, $k \leq d$. Let us associate to each subformula $\theta(\boldsymbol{u})$ a new relation symbol R_θ of same arity. Intuitively, $R_\theta(\boldsymbol{u})$ represents $\theta(\boldsymbol{u})$. Then, let us associate an implication Δ_θ to each subformula $\theta(\boldsymbol{u})$ as follows:

- if $\theta(\boldsymbol{u})$ is quantifier-free, take $\Delta_\theta \equiv \forall \boldsymbol{u} : R_\theta(\boldsymbol{u}) \rightarrow \theta(\boldsymbol{u})$;
- otherwise:

  – if $\theta(\boldsymbol{u}) \equiv \forall v \theta'(\boldsymbol{u}, v)$, take $\Delta_\theta \equiv \forall \boldsymbol{u} \forall v : R_\theta(\boldsymbol{u}) \rightarrow R_{\theta'}(\boldsymbol{u}, v)$;

  – if $\theta(\boldsymbol{u}) \equiv \theta'(\boldsymbol{v}) \wedge \theta''(\boldsymbol{v})$ where $\boldsymbol{u} = \boldsymbol{v} \cup \boldsymbol{w}$, take $\Delta_\theta \equiv \forall \boldsymbol{u} : R_\theta(\boldsymbol{u}) \rightarrow (R_{\theta'}(\boldsymbol{v}) \wedge R_{\theta''}(\boldsymbol{w}))$;

  – if $\theta(\boldsymbol{u}) \equiv \theta'(\boldsymbol{v}) \vee \theta''(\boldsymbol{w})$ where $\boldsymbol{u} = \boldsymbol{v} \cup \boldsymbol{w}$, take $\Delta_\theta \equiv \forall \boldsymbol{u} : R_\theta(\boldsymbol{u}) \rightarrow (R_{\theta'}(\boldsymbol{v}) \vee R_{\theta''}(\boldsymbol{w}))$.

It can be shown by an easy induction that ψ is logically equivalent to the formula:

$$\psi' \equiv \exists R_{\theta_1}, \ldots, R_{\theta_p} : R_\psi \wedge \Delta_{\theta_1} \wedge \ldots \wedge \Delta_{\theta_p},$$

where $\theta_1, \ldots, \theta_p$ enumerate the set of subformulas of ψ, including ψ itself. More precisely, the implication $\psi \rightarrow \psi'$ is straightforward if each R_θ is given its intuitive meaning. For the converse implication, notice that, by an easy induction on the structure of θ, $\Delta_{\theta_1} \wedge \ldots \wedge \Delta_{\theta_p}$ implies that for each subformula $\theta(\boldsymbol{u})$ of ψ we have $\forall \boldsymbol{u}(R_\theta(\boldsymbol{u}) \rightarrow \theta(\boldsymbol{u}))$. Hence, by taking $\theta = \psi$, ψ follows from ψ' by Modus Ponens applied to R_ψ and $R_\psi \rightarrow \psi$. Since each conjunct Δ_θ is of the form $\forall \boldsymbol{u} \theta'(\boldsymbol{u})$, $\boldsymbol{u} \subseteq \boldsymbol{x}$, for a quantifier-free formula θ', the formula ψ' can be put into the form $\exists R_{\theta_1}, \ldots, R_{\theta_p} \forall \boldsymbol{x} \psi''(\boldsymbol{x})$, where ψ'' is quantifier-free, as required. This concludes the proof of Lemma 3.1.❑

PROPOSITION 3.1. $\mathrm{ESO}^{\sigma}(\forall d) \subseteq \mathrm{ESO}^{\sigma}(\mathit{arity}\, d, \forall d)$.

Proof. As before, for simplicity, we give the proof only for $d = 1$ (the proof of the general case is similar). First of all, we need to establish a normalization of the logic $\mathrm{ESO}^{\sigma}(\forall 1)$. Recall that the formulas of this class are of the form (after Skolemization) : $\exists \boldsymbol{f} \forall x \phi$, where $\boldsymbol{f}$ is a sequence of second-order symbols of various arities, x is a first-order variable, and ϕ is a quantifier-free formula of signature $\sigma \cup \boldsymbol{f}$ (see Subsection 1.3). We can assume without loss of generality that each $f \in \boldsymbol{f}$ is a *function* symbol: if it not the case, transform each relation symbol R into a function symbol $\hat{R}$ of the same arity and replace each atomic formula

$R(\tau_1,\ldots,\tau_q)$ by $\hat{R}(\tau_1,\ldots,\tau_q)=c$, where c is any constant symbol, belonging to σ or existentially quantified (if c does not exist, create it by adding c to the ESO symbols of the formula). The point is, as we will prove it in the next lemma, that we can furthermore assume that all the terms and subterms occurring in ϕ are of the form: $\tau(x)=f(\tau_1(x),\ldots,\tau_k(x))$, where f is a k-ary function symbol of $\sigma\cup\boldsymbol{f}$ which doesn't occur in any of the subterms $\tau_i(x)$ $(i=1,\ldots,k)$. Before giving a formal proof of this fact, let us illustrate it by an example: consider the formula $\phi\equiv\forall x: u(u(x,0),v(x))=v(v(x))$, where u,v are, respectively, binary and unary function symbols and 0 is a constant symbol. The terms and subterms occurring in ϕ are the following :

$$\begin{array}{lll}\tau_1(x)=x & \tau_2(x)=0 & \tau_3(x)=u(x,0)\\ \tau_4(x)=v(x) & \tau_5(x)=u(u(x,0),v(x)) & \tau_6(x)=v(v(x))\end{array}$$

Some of these terms (namely, τ_5 and τ_6) don't fulfil the above requirement. Let us now introduce, for each of these terms $\tau_i(x)$, a unary function symbol $\tilde{\tau}_i$, and consider the following formulas, that relate these functions to the terms :

$$\begin{array}{lll}\tilde{\tau}_1(x)=x & \tilde{\tau}_2(x)=0 & \tilde{\tau}_3(x)=u(\tilde{\tau}_1(x),\tilde{\tau}_2(x))\\ \tilde{\tau}_4(x)=v(\tilde{\tau}_1(x)) & \tilde{\tau}_5(x)=u(\tilde{\tau}_3(x),\tilde{\tau}_4(x)) & \tilde{\tau}_6(x)=v(\tilde{\tau}_4(x))\end{array}$$

Now, let us denote by Δ the conjunction of these six formulas. Clearly, the formula $\phi'\equiv\exists\tilde{\tau}_1\ldots\exists\tilde{\tau}_6\forall x:\Delta(u,v,0,\tilde{\tau}_1,\ldots,\tilde{\tau}_6,x)\wedge\tilde{\tau}_5(x)=\tilde{\tau}_6(x)$ is equivalent to ϕ and has the required form. The following lemma generalizes this result.

LEMMA 3.2. *Each formula ϕ in $\mathrm{ESO}^\sigma(\forall 1)$ is equivalent to a formula ϕ' in $\mathrm{ESO}^\sigma(\forall 1)$, where each (sub)term of the form $f(\tau_1(x),\ldots,\tau_q(x))$ is such that no subterm $\tau_i(x)$ contains the function symbol f.*

Proof. As before, we assume without loss of generality that our formula $\phi\equiv\exists\boldsymbol{f}\forall x\psi(x)$ (where ψ is quantifier-free) contains no ESO relation symbol. Let $\mathrm{term}(\psi)$ denote the set of terms and subterms of ψ. To each $\tau\in\mathrm{term}(\psi)$, we associate a new unary function symbol $\tilde{\tau}$, which intends to represent τ, and a formula $\delta_\tau(f,\tilde{\tau},x)$ which inductively defines the function $\tilde{\tau}$ as follows :

- if τ is x or a constant symbol, then $\delta_\tau(f,\tilde{\tau},x)$ is the formula $\tilde{\tau}(x)=\tau$;
- otherwise, *i.e.* if τ is of the form $f(\tau_1(x),\ldots,\tau_q(x))$, $\delta_\tau(f,\tilde{\tau},x)$ is the formula $\tilde{\tau}(x)=f(\tilde{\tau}_1(x),\ldots,\tilde{\tau}_q(x))$.

Now, set $\phi'\equiv\exists\boldsymbol{f}(\exists\tilde{\tau})_{\tau\in\mathrm{term}(\psi)}\forall x[\bigwedge_{\tau\in\mathrm{term}(\psi)}\delta_\tau(f,\tilde{\tau},x)\wedge\psi^*(x)]$, where $\psi^*(x)$ is the formula $\psi(x)$ in which each $\tau\in\mathrm{term}(\psi)$ that is not a proper subterm is replaced by $\tilde{\tau}(x)$. Clearly, ϕ' is equivalent to ϕ and has the required form. This concludes the proof of Lemma 3.2.❏

Let us now prove Proposition 3.1 (for $d=1$). Let $\phi\equiv\exists\boldsymbol{f}\forall t\psi(t)\in\mathrm{ESO}^\sigma(\forall 1)$. Assume (without loss of generality) that it satisfies the condition of Lemma 3.2. We want to eliminate every function symbol f of arity $q>1$ in the ESO prefix $\boldsymbol{f}$. Let $f(\boldsymbol{\tau}_0(t)),\ldots,f(\boldsymbol{\tau}_{k-1}(t))$ be the list of all the occurrences of f in ψ (each

$\boldsymbol{\tau}_i(t)$ is a q-tuple of terms $(\tau_i^1(t), \ldots, \tau_i^q(t)))$. In order to eliminate f, we search to interpret each term $f(\boldsymbol{\tau}_i(t))$ as the image of t by a new unary function. So let us consider k new unary function symbol $F_0, \ldots, F_{k-1}$ and denote by $\tilde{\psi}$ and θ the following formulas:

- $\tilde{\psi}$ is the formula ψ where every term $f(\boldsymbol{\tau}_i(t))$ is replaced by $F_i(t)$ and
- $\theta \equiv \forall t, t' \bigwedge_{i,i'<k} [\boldsymbol{\tau}_i(t) = \boldsymbol{\tau}_{i'}(t') \rightarrow F_i(t) = F_{i'}(t')]$.

(Here, $\boldsymbol{\tau}_i(t) = \boldsymbol{\tau}_{i'}(t')$ is the natural abbreviation for $\bigwedge_{1\leq \ell \leq q} \tau_i^\ell(t) = \tau_{i'}^\ell(t')$.) Now consider the following formula $\tilde{\phi}$ where f no longer occurs:

$$\tilde{\phi} \equiv \exists F_0, \ldots, F_{k-1} : \theta \wedge \forall t \tilde{\psi}$$

We claim that the formula ϕ is equivalent to $\tilde{\phi}$. The argument of our claim is the well-known easy fact that follows.

FACT 3.2. *Let $G : X \rightarrow Y$ and $F : X \rightarrow Z$ be two functions on the same domain X. Then, the two following assertions are equivalent:*

1. *for all $x, y \in X$, $G(x) = G(y)$ implies $F(x) = F(y)$;*
2. *there exists $f : Y \rightarrow Z$ such that $F = f \circ G$.*

To prove our claim, apply Fact 3.2 with the sets $X = [k] \times D$, $Y = D^q$, $Z = D$ and the functions $F(i,t) = F_i(t)$, $G(i,t) = \boldsymbol{\tau}_i(t)$. Unfortunately, there are *two* (universally quantified) first-order variables t, t' in the subformula θ of $\tilde{\phi}$. In order to obtain an equivalent *one*-variable formula as required, we use the same idea and techniques as in the proof of Proposition 2.1. Once again, an ESO linear order $<$ is introduced: it is used to lexicographically order the set of $(q+2)$-tuples: $S = \{(\boldsymbol{\tau}_i(t), i, t), i < k, t \in D\}$. The crucial point is that, for each value $\boldsymbol{v} \in D^q$, the set $I_{\boldsymbol{v}} = \{(\boldsymbol{\tau}_i(t), i, t) : \boldsymbol{\tau}_i(t) = \boldsymbol{v}\}$ forms an interval of S for the lexicographical linear order. In other words, tuples with the same value $\boldsymbol{\tau}_i(t)$ are contiguous for this order. Since $|S| = k|D|$, there is a lexicographically *increasing bijection*, denoted Lex, of $[k] \times D$ onto S, which is defined via the following formulas ψ_{bij} and ψ_{inc}:

$$\psi_{\text{bij}} \equiv \forall (i,t) \exists (j,x) : \text{Lex}(j,x) = (\boldsymbol{\tau}_i(t), i, t)$$

$$\begin{aligned} \psi_{\text{inc}} \equiv \ & \forall (j,x) < (k-1, \max) \exists (j',x') : \\ & (j',x') = \text{Succ}(j,x) \wedge \text{Lex}(j,x) < \text{Lex}(j',x') \end{aligned}$$

Remark. For readability, we use the suggestive and concise notations $\forall(i,t)$, $\exists(j,x)$, etc, that are easy to translate; $\text{Succ}(j,x)$ denotes the successor of the ordered pair (j,x) in the lexicographical order of $[k] \times D$ and similarly for relations $=$ and $<$ between ordered pairs. Notice that our formal syntax represents the function $\text{Lex} : [k] \times D \rightarrow D^{q+2}$ by $k(q+2)$ functions $\text{Lex}_j^i : D \rightarrow D$, where $i < q+2$ and $j < k$.

It is now easy to check that the subformula θ of $\tilde{\phi}$ is equivalent to the following formula θ':

$$\theta' \equiv (\exists_{\text{lin order}} <)(\exists \text{Lex})[\psi_{\text{bij}} \wedge \psi_{\text{inc}} \wedge \psi_{\text{funct}}]$$

Here, the conjunct ψ_{funct} expresses the fact that on each interval $I_{\boldsymbol{v}}$ (see above), two successive elements $\text{Lex}(j,x) = (\boldsymbol{\tau}_i(t), i, t)$ and $\text{LexSucc}(j,x) = (\boldsymbol{\tau}_{i'}(t'), i', t')$ fulfil $F_i(t) = F_{i'}(t')$. That is, ψ_{funct} is the formula:

$$\begin{gathered}\forall (j,x) < (k-1, \max)\ \exists (j',x')\ \exists (i,t)\ \exists (i',t')\ : \\ (j',x') = \text{Succ}(j,x)\ \wedge\ \text{Lex}(j,x) = (\boldsymbol{\tau}_i(t), i, t)\ \wedge\ \text{Lex}(j',x') = (\boldsymbol{\tau}_{i'}(t'), i', t') \\ \wedge\ (\boldsymbol{\tau}_i(t) = \boldsymbol{\tau}_{i'}(t') \rightarrow F_i(t) = F_{i'}(t')).\end{gathered}$$

One easily transforms θ' and, finally, $\tilde{\phi}$ into the (Skolemized) $\text{ESO}^{\sigma}(\text{arity}\,1, \forall 1)$ required form. This concludes the proof of Proposition 3.1 and completes the proof of Theorem 3.1.❑

4. SIMILAR RESULTS FOR MONOTONE CLASSES

In this section, each input vocabulary σ is required to be relational. Iain Stewart has studied several logical descriptions of monotone σ-problems in NP. In [47] and [48], he showed that several monotone problems, including HAMILTON and CUBIC-SUBGRAPH, are complete for monotone-NP via monotone projection translations. In [46], he proved the following theorem, whose proof is used in the proof of Lemma 4.2:

THEOREM 4.1 (Stewart). $\quad \text{monotone-NP}^{\sigma} = \bigcup_d \text{NTIME}^{\sigma+}(n^d) = \text{ESO}^{\sigma+}$.

Proof. Clearly, we have the inclusions:

$$\text{ESO}^{\sigma+} \subseteq \bigcup_d \text{NTIME}^{\sigma+}(n^d) \subseteq \text{monotone-NP}^{\sigma} \subseteq \text{monotone-ESO}^{\sigma},$$

because of Fagin's characterization of NP (namely, $\bigcup_d \text{NTIME}^{\sigma}(n^d) = \text{ESO}^{\sigma}$). The theorem will be an immediate consequence of the inclusion:

$$\text{monotone-ESO}^{\sigma} \subseteq \text{ESO}^{\sigma+}. \tag{4}$$

Let us prove this inclusion: given a problem $\mathcal{P} \in \text{monotone-ESO}^{\sigma}$ over a relational signature $\sigma = \{R_1, \ldots, R_k\}$, there exists an ESO^{σ}-formula ϕ such that $\mathcal{P} = \text{MODELS}(\phi)$. The reader can easily check that, by monotonicity, $\phi(\boldsymbol{R})$ is equivalent to the following ESO^{σ}-formula:

$$\phi' \equiv \exists \boldsymbol{R}'\ :\ \phi(\boldsymbol{R}') \wedge \boldsymbol{R}' \subseteq \boldsymbol{R},$$

where $\boldsymbol{R}' = (R'_1, \ldots, R'_k)$ is a list of new relation symbols such that $\text{arity}(R'_i) = \text{arity}(R_i) = a_i$, where $\phi(\boldsymbol{R}')$ is obtained from $\phi(\boldsymbol{R})$ by replacing each R_i by R'_i and where $\boldsymbol{R}' \subseteq \boldsymbol{R}$ stands for the conjunction $\bigwedge_i \forall \boldsymbol{x}_{a_i} (R'_i(\boldsymbol{x}_{a_i}) \rightarrow R_i(\boldsymbol{x}_{a_i}))$. Then $\mathcal{P} = \text{MODELS}(\phi')$, and since the input relation symbols R_i occur only positively in ϕ', we conclude that $\mathcal{P} \in \text{ESO}^{\sigma+}$.❑

For any degree d of nondeterministic polynomial time, we can prove the following analogue of Theorems 2.1 and 3.1, thus refining Theorem 4.1:

THEOREM 4.2. *For any integer $d > 0$ and any relational vocabulary σ, we have:*

$$\text{monotone-NTIME}^{\sigma}(n^d) = \text{NTIME}^{\sigma+}(n^d)$$
$$= \text{ESO}^{\sigma+}(\mathit{arity}\, d, \forall d) = \text{ESO}^{\sigma+}(\forall d) = \text{ESO}^{\sigma+}(\mathit{var}\ d).$$

Proof. This theorem is the consequence of a series of class inclusions with, in particular, the two following lemmas:

LEMMA 4.1. $\text{ESO}^{\sigma+}(\mathit{arity}\, d, \forall d) \subseteq \text{NTIME}^{\sigma+}(n^d)$.

Proof. The proof is a variant of that of Lemma 2.1 to which the reader is invited to refer. Let $\mathcal{P} \in \text{ESO}^{\sigma+}(\text{arity}\, d, \forall d)$ for a relational signature $\sigma = \{R_1, \ldots, R_k\}$. Then $\mathcal{P} = \text{MODELS}(\phi)$ for a formula $\phi \equiv \exists\rho\forall\boldsymbol{x}\psi(\boldsymbol{x})$, where $\text{arity}\,(\rho) = \text{arity}\,(\boldsymbol{x}) = d$ and ψ is a disjunctive normal form $\gamma_0(\boldsymbol{x}) \vee \ldots \vee \gamma_{q-1}(\boldsymbol{x})$ in which the R_i's occur only positively. Then $\mathcal{P}$ is recognized by a nondeterministic algorithm similar to $\mathcal{A}$ (cf. proof of Lemma 2.1) where Part (b) is replaced by the following new part:

for each $\boldsymbol{a} \in [n]^d$, guess a number i. If $i < q$ check that $([n], \sigma, \rho) \models \gamma_i(\boldsymbol{a})$.
If not, reject. If no rejection occurs, accept.

It is essential to notice that since γ_i is a conjunction of literals where each $R_j \in \sigma$ occurs positively, then in each accepting computation, each access to the input can be realized by a positive σ-NRAM instruction of the form (2+), as required.❑

LEMMA 4.2. $\text{monotone-ESO}^{\sigma}(\mathit{arity}\, d, \forall d) \subseteq \text{ESO}^{\sigma+}(\mathit{arity}\, d, \forall d)$.

Proof. For the sake of simplicity, let us prove this result for $d = 1$ and $\sigma = \{R\}$, where R is a k-ary relation symbol. The general case is similar. As justified in the proof of Theorem 4.1, any sentence $\phi(R) \in \text{ESO}^{\sigma}$ expressing a monotone property of σ-structures is equivalent to the following sentence $\phi'(R) \in \text{ESO}^{\sigma+}$:

$$\exists R' : \phi(R') \wedge \forall\boldsymbol{x}(R'(\boldsymbol{x}) \rightarrow R(\boldsymbol{x})),$$

where R' is a new k-ary relation symbol and $\phi(R')$ is obtained from $\phi(R)$ by replacing R by R'.

Now, let us assume $\phi(R) \in \text{ESO}^{\sigma}(\text{arity}\, 1, \forall 1)$. The equivalent formula $\phi'(R)$ above belongs to $\text{ESO}^{\sigma+}$ but does not belong to $\text{ESO}^{\sigma}(\text{arity}\, 1, \forall 1)$ in case $k > 1$, as it is required. More precisely, we show: if $\phi(R)$ is of the form $\exists\nu\forall y\psi(R, \nu, y)$ with $\text{arity}\,(\nu) = 1$, and if ψ is quantifier-free, then $\phi(R)$ is equivalent to the following sentence $\phi'(R) \in \text{ESO}^{\sigma+}$, where $\text{arity}\,(R') = k$:

$$\exists\nu\exists R' : \forall y\psi(R', \nu, y) \wedge \forall\boldsymbol{x}(R'(\boldsymbol{x}) \rightarrow R(\boldsymbol{x})).$$

We now transform the formula $\phi'(R)$ into an equivalent formula of $\text{ESO}^{\sigma+}(\text{arity}\, 1, \forall 1)$ in two steps:

(1) first, we transform $\phi'(R)$ into $\phi_1(R) \in \mathrm{ESO}^{\sigma+}(\mathrm{arity}\,1)$;
(2) then, we transform $\phi_1(R)$ into $\phi_2(R) \in \mathrm{ESO}^{\sigma+}(\mathrm{arity}\,1, \forall 1)$.

Let $R'(\boldsymbol{t}_i(y))_{i\in I}$ denote the set of distinct atomic subformulas of $\psi(R', \nu, y)$ that involve R'. (Note: each $\boldsymbol{t}_i(y)$ is a tuple of terms of the same arity as R' and R.) Step (1) essentially consists in replacing each atom $R'(\boldsymbol{t}_i(y))$ by $R_i(y)$, where R_i is a new relation symbol. More precisely, we show the following:

Claim. $\phi'(R)$ is equivalent to the following formula $\phi_1(R) \in \mathrm{ESO}^{\sigma+}(\mathrm{arity}\,1)$:

$$\exists \nu (\exists R_i)_{i\in I} \forall y \left(\begin{array}{l} \bigwedge_{i\in I} (R_i(y) \to R(\boldsymbol{t}_i(y)) \ \wedge \ \psi'((R_i)_{i\in I}, \nu, y) \wedge \\ \bigwedge_{i\in I} \forall z \bigwedge_{j\in I} \boldsymbol{t}_i(y) = \boldsymbol{t}_j(z) \to (R_i(y) \leftrightarrow R_j(z)) \end{array} \right)$$

where ψ' denotes the quantifier-free formula ψ where each atom $R'(\boldsymbol{t}_i(y))$ has been replaced by $R_i(y)$.

Indeed, $\phi'(R)$ clearly implies $\phi_1(R)$: interpret $R_i(y)$ as $R'(\boldsymbol{t}_i(y))$ for each y. The converse implication is obtained by defining the Boolean values of R' as follows, on the universe of the input structure $\langle [n], R\rangle$:

(a) $R'(\boldsymbol{t}_i(y)) := R_i(y)$ for each $i \in I$ and $y \in [n]$;
(b) $R'(\boldsymbol{x}) := R(\boldsymbol{x})$ if the tuple $\boldsymbol{x} \in [n]^k$ is distinct from each tuple $\boldsymbol{t}_i(y)$, $i \in I$, $y \in [n]$.

The coherence of the first item of this definition follows from the third conjunct of $\phi_1(R)$. Conditions (a) and (b), in addition to the first conjunct of ϕ_1, imply together $R' \subseteq R$. Finally, from the formula $\forall y \psi'((R_i)_{i\in I}, \nu, y)$, one easily deduces $\forall y \psi(R', \nu, y)$ by Condition (a). This proves the claim.

The three conjuncts of $\phi_1(R)$ have the required form: they involve only unary ESO symbols (ν, $(R_i)_{i\in I}$) and only *one* first-order variable y, except for the third conjunct in which the new variable z appears. But this third conjunct can be written in $\mathrm{ESO}(\mathrm{arity}\,1, \forall 1)$:

Claim. The formula $\psi_1 \equiv \forall y \bigwedge_{i\in I} \forall z \bigwedge_{j\in I} \boldsymbol{t}_i(y) = \boldsymbol{t}_j(z) \to (R_i(y) \leftrightarrow R_j(z))$ is equivalent to some formula in $\mathrm{ESO}(\mathrm{arity}\,1, \forall 1)$.

A similar assertion has yet been proved in the proof of Proposition 3.1. The only difference is that we are now interested in unary relation symbols instead of unary function symbols. But the proof is nearly the same and the reader is invited to refer to it. This concludes the proof of Lemma 4.2.❑

Now, we can prove Theorem 4.2 in considering Figure 2. In this scheme, $A \to B$ stands for $A \subseteq B$ and $A \leftrightarrow B$, for $A = B$. Besides, the labels of the arrows refer to the following arguments:

(0) follows immediately from the definitions of the involved classes;
(1) follows from the equality $\mathrm{NTIME}^{\sigma}(n^d) = \mathrm{ESO}^{\sigma}(\mathrm{arity}\,d, \forall d)$ proved in Section 2 (Theorem 2.1) and from the definition monotone-$\mathcal{C} := \mathrm{Monotone}^{\sigma} \cap \mathcal{C}$ for any class of σ-problems $\mathcal{C}$;

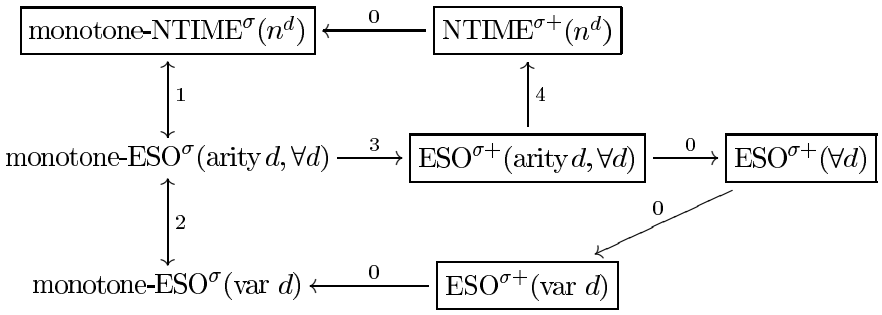

FIG. 2. Inclusions between the complexity classes

(2) follows from the equality ESO$^{\sigma}$(arity $d, \forall d$) = ESO$^{\sigma}$(var d) proved in Section 3 (Theorem 3.1) and from the definition of monotone-$\mathcal{C}$;

(3) is Lemma 4.2;

(4) is Lemma 4.1.

Clearly, this scheme implies the equality between all the involved classes and, in particular, the equality between the framed classes. This completes the proof of Theorem 4.2.❑

5. SEMANTICAL INVARIANCE PROPERTIES OF ESO($\forall$1)

In Section 6, we shall prove that some well-known graph problems belong to the class vertexNLIN. In order to establish these memberships by purely logical means (*i.e.* by proving the definability of these problems in ESO($\forall$1)), we first examine, in the present section, some syntactical extensions of ESO($\forall$1), which will simplify the formulation of graph properties under consideration. We prove that these syntactical extensions do not enlarge the semantical scope of ESO($\forall$1), so that the above mentionned properties appear to be in vertexNLIN. Such a result has already been proved in Section 2: Corollary 2.1 precisely states that existential quantifications over linear orders ($\exists_{\text{lin order}} <$) do not enlarge ESO($\forall$1) from a semantical point of view. This result is a key argument of many definability results in ESO($\forall$1) and it will be widely used in the present section. However, this extension will not be sufficient to give a correct and useful ESO($\forall$1)-formulation of some rather sophisticated graph properties. In particular, the logical descriptions of many such properties seem to require the use of two first-order variables, in order to fully describe the behaviour of the edge relation of the graphs. To perform these logical characterizations in the more restrictive logic ESO($\forall$1), we shall first have to "translate" the property under consideration into an equivalent property dealing with a spanning forest (resp. tree) of the graph which, in turn, can be logically characterized with a single first-order variable (because of the unary arity of the forest). For instance, we shall prove the ESO($\forall$1)-definability of connectivity using the fact that a graph is connected iff it is spanned by some tree, the latter being easily expressed in our logic.

For these reasons, the core of this section consists of the proof that ESO($\forall$1) is not enlarged by existential quantification over forests and over several functions and relations (including transitive closure) related to forests (Subsection 5.2). Then we shall concentrate on *functional graphs*, that is, graphs of unary functions. We shall state that in those graphs, both transitive closure (Subsection 5.3) and a certain notion of distance (Subsection 5.4) can be defined in ESO($\forall$1). These results notably attest the robustness of the logic ESO($\forall$1) and, in turn, the robustness of the complexity class vertexNLIN.

First of all, let us give a precise meaning to the assertion: "existential quantification over such and such a class of structures (linear orders, forests, etc) does not enlarge ESO($\forall$1)".

5.1. Existential quantification over sets of structures

In the rest of the paper, we deal with the logic ESO($\forall$1). From now on, we denote it by ESO_1 (or ESO_1^σ when we want to restrict it to a particular signature).

Let σ, τ be two disjoint signatures and $\mathcal{T}$ be a set of finite τ-structures. The logic $\mathrm{ESO}_1^\sigma[\mathcal{T}]$ is the set of formulas of the form: $(\exists\tau \in \mathcal{T})\Phi$ with $\Phi \in \mathrm{ESO}_1^{\sigma\tau}$. The semantic of such a formula is naturally defined: a σ-structure $\langle D, \sigma\rangle$ satisfies $(\exists\tau \in \mathcal{T})\Phi$ iff there exists an interpretation of τ on D such that $\langle D, \tau\rangle \in \mathcal{T}$ and $\langle D, \sigma, \tau\rangle \models \Phi$. The condition $\langle D, \tau\rangle \in \mathcal{T}$ will often be denoted by: $\tau \in \mathcal{T}(D)$. Therefore,

$$\langle D, \sigma\rangle \models (\exists\tau \in \mathcal{T})\Phi \text{ iff there exists } \tau \in \mathcal{T}(D) \text{ such that } \langle D, \sigma, \tau\rangle \models \Phi$$

Let σ, τ, τ' be three pairwise disjoint signatures, $\mathcal{T} \subseteq \textsc{struc}(\tau)$, $\mathcal{T}' \subseteq \textsc{struc}(\tau')$. We write $\mathrm{ESO}_1^\sigma[\mathcal{T}] \subseteq \mathrm{ESO}_1^\sigma[\mathcal{T}']$ when each formula of the first logic is equivalent (on σ-structures) to a formula of the second logic. In other terms:

$$\begin{gathered}\mathrm{ESO}_1^\sigma[\mathcal{T}] \subseteq \mathrm{ESO}_1^\sigma[\mathcal{T}'] \\ \text{iff} \\ (\forall\phi \in \mathrm{ESO}_1^\sigma[\mathcal{T}])\,(\exists\phi' \in \mathrm{ESO}_1^\sigma[\mathcal{T}'])\text{ s.t. } \textsc{models}(\phi') = \textsc{models}(\phi).\end{gathered}$$

When the converse inclusion also holds, we note $\mathrm{ESO}_1^\sigma[\mathcal{T}] = \mathrm{ESO}_1^\sigma[\mathcal{T}']$. When this equality holds for any signature σ, we write $\mathrm{ESO}_1[\mathcal{T}] = \mathrm{ESO}_1[\mathcal{T}']$. In the particular case where $\mathcal{T}' = \emptyset$ (*i.e.* $\mathrm{ESO}_1[\mathcal{T}] = \mathrm{ESO}_1$), we say that *existential quantification over $\mathcal{T}$ does not enlarge* ESO_1.

Definition and example. Let us denote by LINORD the set of finite structures $\langle D, <\rangle$, where $<$ is a linear order on the domain D. Corollary 2.1 precisely says that existential quantification over LINORD does not enlarge ESO_1. In our new formalism, we can write:

$$\mathrm{ESO}_1[\textsc{linord}] = \mathrm{ESO}_1, \tag{5}$$

and this equality must be understood as follows: for any signature σ and any formula ϕ of the form $(\exists < \in \textsc{linord})\psi$, whith $\psi \in \mathrm{ESO}_1^{\sigma,<}$, there exists a formula $\phi' \in \mathrm{ESO}_1^\sigma$ such that $\textsc{models}(\phi) = \textsc{models}(\phi')$, and conversely. (Notice that the

existential quantification over a linear order $<$, previously denoted by $(\exists_{\text{lin order}} <)$ is now written: $(\exists < \in \text{LINORD})$.)

The goal of the following subsections is to state such "invariance results" for several sets of structures $\mathcal{T}$. Let us mention some easy remarks about relative inclusions between logics $\text{ESO}_1[\mathcal{T}]$. Until the end of this subsection, τ, τ', τ'' denote any three signatures and $\mathcal{T}, \mathcal{T}', \mathcal{T}''$ three sets of structures such that $\mathcal{T} \subseteq \text{STRUC}(\tau)$, $\mathcal{T}' \subseteq \text{STRUC}(\tau')$ and $\mathcal{T}'' \subseteq \text{STRUC}(\tau'')$. First, observe that our semantical inclusion is transitive. That is:

$$\text{ESO}_1[\mathcal{T}] \subseteq \text{ESO}_1[\mathcal{T}'] \subseteq \text{ESO}_1[\mathcal{T}"] \;\Rightarrow\; \text{ESO}_1[\mathcal{T}] \subseteq \text{ESO}_1[\mathcal{T}"] \tag{6}$$

Of course, this transitivity result can be extended to semantical equalities. That is, (6) still holds when replacing "$\subseteq$" by "$=$".

Now, suppose $\mathcal{T}$ is definable in $\text{ESO}_1^{\tau}[\mathcal{T}']$. In other words, there exists a formula $\Phi \equiv (\exists \tau' \in \mathcal{T}')\phi(\tau, \tau')$ in $\text{ESO}_1^{\tau}[\mathcal{T}']$ such that $\mathcal{T} = \text{MODELS}(\Phi)$. Then, any formula $\Psi \equiv (\exists \tau \in \mathcal{T})\psi(\sigma, \tau)$ in $\text{ESO}_1^{\sigma}[\mathcal{T}]$ is clearly equivalent to $\exists\tau(\exists\tau' \in \mathcal{T}') : \phi(\tau, \tau') \wedge \psi(\sigma, \tau)$. This last formula can be written as $(\exists \tau' \in \mathcal{T}')\exists\tau\phi(\tau, \tau') \wedge \psi(\sigma, \tau)$, which can be proved to be in $\text{ESO}_1^{\sigma}[\mathcal{T}']$ by easy closure properties of this logic. Since these remarks hold for any signature σ, they can be summarized by:

$$\mathcal{T} \text{ is definable in } \text{ESO}_1^{\tau}[\mathcal{T}'] \;\Rightarrow\; \text{ESO}_1[\mathcal{T}] \subseteq \text{ESO}_1[\mathcal{T}'] \tag{7}$$

We shall often make use of the following result: we say that $\mathcal{T}$ is a *complete restriction of* $\mathcal{T}'$ if $\tau \subseteq \tau'$, if each structure $\langle D, \tau\rangle$ of $\mathcal{T}$ can be expanded into a structure $\langle D, \tau'\rangle$ belonging to $\mathcal{T}'$ and if furthermore each structure $\langle D, \tau'\rangle \in \mathcal{T}'$ is an expansion of a structure $\langle D, \tau\rangle \in \mathcal{T}$. With this definition, each formula $\Psi \equiv (\exists \tau \in \mathcal{T})\Phi(\sigma, \tau)$ of $\text{ESO}_1^{\sigma}[\mathcal{T}]$ is obviously equivalent to a formula Ψ' of $\text{ESO}_1^{\sigma}[\mathcal{T}']$. Namely, if we denote $\tau' = \tau \cup \rho$, then $\Psi' \equiv (\exists \tau\rho \in \mathcal{T}')\Phi(\sigma, \tau)$. Thus we have:

$$\mathcal{T} \text{ is a complete restriction of } \mathcal{T}' \;\Rightarrow\; \text{ESO}_1[\mathcal{T}] \subseteq \text{ESO}_1[\mathcal{T}'] \tag{8}$$

Definition and example.. Let us consider the signature $\{<, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}\}$ in which $<$ is a binary relation symbol, pred and succ are unary function symbols, and min and max are constant symbols. Let us furthermore denote by FULL-LINORD the set of finite structures $\langle D, <, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}\rangle$, where $<$ is a linear order, pred and succ are its associated predecessor and successor functions, min and max are its associated minimal and maximal elements. Hence, trivially, LINORD is a complete restriction of FULL-LINORD (consequently, $\text{ESO}_1[\text{LINORD}] \subseteq \text{ESO}_1[\text{FULL-LINORD}]$).

We now mention two easy but useful remarks. Let $\tau_1, \ldots, \tau_k$ be k signatures such that $\sigma, \tau_1, \ldots, \tau_k$ are pairwise disjoint. Let $\mathcal{T}_1 \subseteq \text{STRUC}(\tau_1)$, $\ldots$, $\mathcal{T}_k \subseteq \text{STRUC}(\tau_k)$ be k sets of structures. We denote by $\text{ESO}_1^{\sigma}[\mathcal{T}_1, \ldots, \mathcal{T}_k]$ the set of formulas of the form: $(\exists \tau_1 \in \mathcal{T}_1)\ldots(\exists \tau_k \in \mathcal{T}_k)\Phi$, where $\Phi \in \text{ESO}_1^{\sigma\tau_1\cdots\tau_k}$. The semantic associated to this logic is as expected. Recalling that an equality such as $\text{ESO}_1[\mathcal{T}] = \text{ESO}_1$

stands for $\forall\sigma\ :\ \mathrm{ESO}_1^\sigma[\mathcal{T}] = \mathrm{ESO}_1^\sigma$, one can easily prove:

$$(\forall i = 1, \ldots, k :\ \mathrm{ESO}_1[\mathcal{T}_i] = \mathrm{ESO}_1) \Rightarrow \mathrm{ESO}_1[\mathcal{T}_1, \ldots, T_k] = \mathrm{ESO}_1 \qquad (9)$$

As the above implication holds for any signatures τ_i, we obtain, when $\tau_1, \ldots, \tau_k$ are pairwise disjoints copies of τ: for any $\mathcal{T} \subseteq \mathrm{STRUC}(\tau)$,

$$\mathrm{ESO}_1[\mathcal{T}] = \mathrm{ESO}_1 \Rightarrow \mathrm{ESO}_1[\mathcal{T}, \ldots, \mathcal{T}] = \mathrm{ESO}_1 \qquad (10)$$

and we shall keep in mind the meaning of this implication in the following form:

if $\mathrm{ESO}_1[\mathcal{T}] = \mathrm{ESO}_1$, then for every signature σ, any formula of the form $(\exists\tau_1 \in \mathcal{T}) \ldots (\exists\tau_k \in \mathcal{T})\Phi$, with $\Phi \in \mathrm{ESO}_1^{\sigma\tau_1\cdots\tau_k}$, is equivalent to a formula of ESO_1^σ.

We conclude this subsection with two invariance results of ESO_1:

LEMMA 5.1. $\mathrm{ESO}_1[\text{FULL-LINORD}] = \mathrm{ESO}_1$.

Proof. Let $\Psi \equiv (\exists(<, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}) \in \text{FULL-LINORD})\Phi$ be a formula in $\mathrm{ESO}_1^\sigma[\text{FULL-LINORD}]$, where σ is any signature. Then Ψ has the same models $\langle D, \sigma\rangle$ as

$$\Psi' \equiv (\exists < \in \text{LINORD})(\exists \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max})\ (\Phi' \wedge \Phi),$$

where Φ' is the formula

$$\forall x\ :\ (\mathsf{min} \leq x \leq \mathsf{max})\ \wedge\ (x \neq \mathsf{min} \rightarrow (\mathsf{pred}(x) < x \wedge \mathsf{succ}\,\mathsf{pred}(x) = x)).$$

Indeed, Φ' forces $\mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}$, respectively, to be the predecessor, successor, minimum and maximum related to the existentially quantified linear order $<$: just consider a strictly increasing enumeration of D according to $<$, say $a_1 < a_2 < \cdots < a_n$, and prove, from Φ', that $a_1 = \mathsf{min}$, $a_n = \mathsf{max}$ and for each $1 < i < n$, $a_i = \mathsf{pred}(a_{i+1}) = \mathsf{succ}(a_{i-1})$ (by recurrence on i). As Ψ' clearly has a prenex form in $\mathrm{ESO}_1^\sigma[\text{LINORD}]$, it yields $\mathrm{ESO}_1[\text{FULL-LINORD}] \subseteq \mathrm{ESO}_1[\text{LINORD}]$. But the converse inclusion also holds (see the previous example). Therefore $\mathrm{ESO}_1[\text{FULL-LINORD}] = \mathrm{ESO}_1[\text{LINORD}]$ and the conclusion follows from the equations (5) and (6) stated above.❑

The last result has a different flavour, since it deals with *ordered* structures, that is, with structures over a signature that contains a built-in linear order. More precisely: let σ be any signature and $<$ be a binary relation symbol. We call *ordered* σ-structure any structure $S = \langle D, \sigma, <\rangle$ over the signature $\sigma \cup \{<\}$ in which $<$ is interpreted as a linear order. This allows to identify D to the initial segment of $\mathbb{N}$ of size $|D|$ and to view some functions over this initial segment as functions over D.

In particular, let us temporarily use the following notations: if $|D| = n$, and if k is an integer strictly smaller than n, we denote by $\overline{k}$ the k^{th} successor of the minimal element of $(D, <)$. *E.g.*, if min (resp. max) denotes the minimal (resp. maximal)

element of $(D,<)$, then $\overline{0} = \mathsf{min}$ and $\overline{n-1} = \mathsf{max}$. Then, we denote by $\text{ARITH}_<(D)$ the set of functions $+, -, \times, \mathsf{div}, \mathsf{mod} : D \times D \to D$ defined as follows: for all $k, \ell < n$,

$$\begin{array}{rcl} \overline{k} + \overline{\ell} & = & \overline{\mathit{Min}(k+\ell, n-1)} \\ \overline{k} \times \overline{\ell} & = & \overline{\mathit{Min}(k\ell, n-1)} \\ \overline{k} - \overline{\ell} & = & \overline{\mathit{Max}(k-\ell, 0)} \\ \overline{k} \ \mathsf{div} \ \overline{\ell} & = & \overline{q} \\ \overline{k} \ \mathsf{mod} \ \overline{\ell} & = & \overline{r}, \end{array}$$

where, assuming $\ell \neq 0$, q and r are the unique integers such that $k = q\ell + r$, $q < n$, $r < \ell$, and where *Min* (resp. *Max*) maps each $(i,j) \in \mathbb{N}^2$ to i (resp. j) if $i \leq j$, and to j (resp. i) otherwise. (We write $\overline{k}\,\overline{\ell}$ instead of $\overline{k} \times \overline{\ell}$.)

For any signature σ, we denote by $\text{STRUC}_<(\sigma)$ the set of $\sigma \cup \{<\}$-structures in which $<$ is interpreted by a linear order. A subset of $\text{STRUC}_<(\sigma)$ will be called a set of *ordered σ-structures* and we will denote it by $\mathcal{T}_<$ to recall that the interpretation of σ over a structure $S \in \mathcal{T}_<$ depends on a built-in linear order on S. Given a set of ordered τ-structures $\mathcal{T}_<$, we denote by $\text{ESO}_1^\sigma[\mathcal{T}_<]$ the set of formulas of the form:

$$(\exists \tau \in \mathcal{T}_<)\Phi(<, \sigma, \tau)$$

with $\Phi \in \text{ESO}_1^{<,\sigma,\tau}$. The semantic of this logic is as follows: the models of such a formula are *ordered* structures. An ordered σ-structure $\langle D, <, \sigma\rangle$ satisfies $(\exists \tau \in \mathcal{T}_<)\Phi$ iff there exists an interpretation of τ on D such that: $\langle D, <, \tau\rangle \in \mathcal{T}_<$ and $\langle D, <, \sigma, \tau\rangle \models \Phi$. Finally, we write $\text{ESO}_1^\sigma[\mathcal{T}_<] = \text{ESO}_1^\sigma[<]$ when for each formula $\Phi \in \text{ESO}_1^\sigma[\mathcal{T}_<]$ there exists a formula $\Phi' \in \text{ESO}_1^\sigma[<]$ such that Φ and Φ' have the same ordered σ-models. The last result of this subsection is given without proof. It attests the invariance of ESO_1 under arithmetical extensions.

LEMMA 5.2 (Olive). *For any signature σ,* $\text{ESO}_1^\sigma[\text{ARITH}_<] = \text{ESO}_1^\sigma[<]$.

Proof. We just sketch very roughly the proof. It can be found in [36], Theorem 2.31, p. 104. First, one can prove the definability in $\text{ESO}_1^\sigma[<]$ of the unary functions π_0, π_1, π_2 and of the constant b defined as follows over a domain D: $b = \lfloor\sqrt{|D|}\rfloor$ and $\forall x \in D$: $\pi_0(x), \pi_1(x), \pi_2(x) < b$ and $x = \pi_0(x) + \pi_1(x)b + \pi_2(x)b^2$. This is done recursively by forcing $\pi_0(\mathsf{succ}(x))$, $\pi_1(\mathsf{succ}(x))$ and $\pi_2(\mathsf{succ}(x))$ to fit their rigth values, with respect to the values of $\pi_0(x)$, $\pi_1(x)$ and $\pi_2(x)$. Then, one can define the restrictions of $+$ and $\times$ to $[b]$. More precisely, the unary functions A and M such that $\forall x \in D$, $A(x) = \pi_0(x) + \pi_1(x)$ and $M(x) = \pi_0(x)\pi_1(x)$ can be defined in $\text{ESO}_1^\sigma[<]$. Once again, this is done recursively, by stating the value of $A(x)$ (resp. $M(x)$) when $\pi_1(x) = 0$ and by relating $A(x+b)$ (resp. $M(x+b)$) to $A(x)$ (resp. $M(x)$) (notice that the function $x \mapsto x + b$ is itself trivially definable from succ). At last, the definability of the functions $+$ and $\times$ is easily deduced from those of A and M. The definability of $-$, div and mod is proved similarly.❑

5.2. Prefix order in a forest

The results of this subsection and of the next one ("Transitive closure of a function") are essentially due to a collaboration with C. Lautemann [32] and S. Ranaivoson [42]

Let D be a finite domain. We say that a function $F : D \to D$ is a forest over D if F has no cycle except the loops $F(x) = x$. We denote it by $F \in \text{FOREST}(D)$. If $F : D \to D$ is a forest, we denote by $\mathsf{desc}_F(x)$ the set of the descendants of the node $x \in D$ in the forest, including x. In other words, $\mathsf{desc}_F(x) = \{y \in D : \exists i \in \mathbb{N} \text{ s.t. } F^i(y) = x\}$.

A *prefix order* of the forest $\langle D, F\rangle$ is a linear order $<$ of D satisfiying : $\forall x \in D$, $F(x) \leq x$ and $\mathsf{desc}_F(x)$ is an interval with respect to $<$. We write $(F, <) \in \text{PREFORD-FOREST}(D)$ when $F \in \text{FOREST}(D)$ and $<$ is a prefix order on this forest.

One can easily prove that a linear order $<$ is a prefix order of a given forest $\langle D, F\rangle$ if, and only if, for each $x \in D$ there exists $l_x \in D$ such that $\mathsf{desc}_F(x) = [x, l_x]$. The node l_x is thus the *last descendant* of x in F according to $<$ and the function $x \mapsto l_x$ is called the *last function* associated to $(F, <)$.

Given a prefix ordered forest $(D, F, <)$, each node $x \in D$ which is not a root may have siblings smaller than it (according to $<$). If this is the case, we call *left sibling* of x the largest of those siblings; if it is not, we assume that the left sibling of x is x itself. We extend this notion to the roots in the following way: the left sibling of the smallest root r_1 is r_1 itself ; the left sibling of a root $r > r_1$ is the biggest root smaller than r. Finally, we call *left function* associated to $(D, F, <)$ the unary function over D which maps every node onto its left sibling. Figure 3 shows a forest F together with a left function (in dotted lines) associated to the prefix order underlying the chosen planar representation of F.

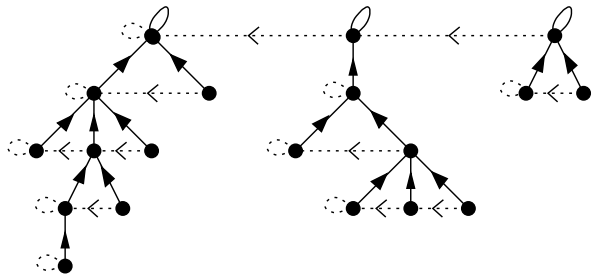

FIG. 3. a forest F (bold lines) and its left function (dotted lines)

The *root function* associated to F maps each $x \in D$ onto the root r of its component in the forest (*i.e.* $F(r) = r$ and $x \in \mathsf{desc}_F(r)$).

We say that a tuple $(F, \mathsf{root}, \mathsf{last}, \mathsf{left})$ is a *full forest* over D if:

- $F \in \text{FOREST}(D)$;
- root is the root function associated to F ;

• last and left are, respectively, the last function and the left function associated to F, relatively to the same prefix order over F.

We denote by FULL-FOREST(D) the set of such tuples.

PROPOSITION 5.1 ([32, 42]). *Let D be a finite domain and F, root, last, left be four unary functions over D. Then $(F, \mathsf{root}, \mathsf{last}, \mathsf{left}) \in$ FULL-FOREST(D) if, and only if, there exists $(<, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}) \in$ FULL-LINORD(D) such that, for every $x \in D$:*

(a) $F(x) \leq x \leq \mathsf{last}(x) \leq \mathsf{last}F(x)$;
(b) one of these three assertions is true:

(i) $\mathsf{pred}(x) = F(x) \wedge \mathsf{left}(x) = x$;

(ii) $x = F(x) \wedge \mathsf{left}(x) = F\mathsf{left}(x) \wedge \mathsf{last}\,\mathsf{left}(x) = \mathsf{pred}(x)$;

(iii) $x \neq F(x) \wedge \mathsf{left}(x) \neq F\mathsf{left}(x) \wedge F\mathsf{left}(x) = F(x) \wedge \mathsf{last}\,\mathsf{left}(x) = \mathsf{pred}(x)$;

(c) $(F(x) = x \rightarrow \mathsf{root}(x) = x) \wedge \mathsf{root}F(x) = \mathsf{root}(x)$.

Proof. A full forest trivially fulfils conditions (a) through (c), if we take for $<$ the prefix order over F according to which last and left are the last and left functions associated to F. So, we only have to prove the sufficiency of $(a) - (c)$. That is, let us assume that there exists $(<, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}) \in$ FULL-LINORD(D) such that $(F, \mathsf{root}, \mathsf{last}, \mathsf{left})$ satisfies conditions (a) to (c) for every $x \in D$ and let us prove that:

• F is a forest and root is its root function ;
• last and left are the last and left functions associated to F relatively to the same prefix order.

In the following, we shall often use the characteristic property of FULL-LINORD given in the proof of Lemma 5.1. We recall it there: let $<$ be a linear order over D, $\mathsf{pred}, \mathsf{succ} : D \rightarrow D$ and $\mathsf{min}, \mathsf{max} \in D$. Then $(<, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}) \in$ FULL-LINORD(D) iff for every $x \in D$,

(d) $(\mathsf{min} \leq x \leq \mathsf{max}) \ \wedge \ (x \neq \mathsf{min} \rightarrow (\mathsf{pred}(x) < x \wedge \mathsf{succ}\,\mathsf{pred}(x) = x))$.

F is a forest and root is its root function.

According to (a), any F-circuit $x_1 \xrightarrow{F} \cdots \xrightarrow{F} x_p \xrightarrow{F} x_1$ must satisfy $x_1 \geq \cdots \geq x_p \geq x_1$ and thus $x_1 = \cdots = x_p$. Consequently, all F-circuits are loops and F is a forest. Furthermore, Condition (c) allows to prove inductively that root is constant on each connected component of F and maps each root to itself. That is, root is the root function associated to F.

last and left are the last and left functions associated to F relatively to the same prefix order.

Actually, we shall prove that the above existentially quantified $<$ is necessarily a prefix order over D and that last and left are the last and left functions associated

to F according to *this* prefix order. That is, we will prove the statement: *for each* $x \in D$, $\mathsf{desc}_F(x) = [x, \mathsf{last}(x)]$ *and* $\mathsf{left}(x)$ *is the left sibling of* x *according to* $<$. Notice that Condition (a) affirms the inclusion $[x, \mathsf{last}(x)] \subseteq [F(x), \mathsf{last}F(x)]$ which in turn, allows to prove inductively the inclusion $\mathsf{desc}_F(x) \subseteq [x, \mathsf{last}(x)]$. This will help us to prove the above statement by recurrence on the level ℓ of x in the forest F. That is, we prove by induction that the following assertion holds for any $\ell < |D|$:

for each node x *of level* ℓ *in* F*:*
$\mathsf{desc}_F(x) = [x, \mathsf{last}(x)]$ *and* $\mathsf{left}(x)$ *is the left sibling of* x*.*

We denote by (H_ℓ) this recurrence hypothesis. In order to prove that H_0 and $(H_\ell \Rightarrow H_{\ell+1})$ hold, let us notice the following: if $x \in D$ is a root of F, it must satisfy (bi) or (bii) (since condition $(biii)$ demands $F(x) \neq x$). In the first case, $\mathsf{left}(x) = x$; in the second case, $\mathsf{left}(x)$ is a root (since $F\mathsf{left}(x) = \mathsf{left}(x)$) such that $\mathsf{last}\,\mathsf{left}(x) = \mathsf{pred}(x)$. But this last equality implies, by Conditions (d) and (a), that $\mathsf{left}(x)$ is smaller than x. Consequently, in both cases $\mathsf{left}(x)$ is a root smaller than x. Analogously, any "non root" node x must satisfy (bi) or $(biii)$. And we can prove as above that these conditions force $\mathsf{left}(x)$ to be a "non root" smaller than x. So, we will remind the following consequences of Condition (b):

if x is a root (resp. a non root node), then x satisfies (bi) or (bii)
(resp. (bi) or $(biii)$) and $\mathsf{left}(x)$ is a root (resp. a non root node)
smaller or equal to x.

We will call this assertion Condition (b').

The case of roots (H_0)

Let $r_1 < r_2 < \cdots < r_k$ be the strictly increasing list of the roots of F. We first prove the two following facts:

- $\mathsf{pred}(r_1) = r_1 = \mathsf{left}(r_1)$. Indeed, since r_1 is a root, $\mathsf{left}(r_1)$ is a root smaller or equal to r_1 (by (b')) and thus $\mathsf{left}(r_1) = r_1$ (since r_1 is the smallest root). Furthermore, r_1 satisfies (bi) or (bii). In the first case, $\mathsf{pred}(r_1) = F(r_1) = r_1$; in the second, the equality $\mathsf{last}\,\mathsf{left}(r_1) = \mathsf{pred}(r_1)$ leads to $\mathsf{last}(r_1) = \mathsf{pred}(r_1)$ and then, to $r_1 \leq \mathsf{pred}(r_1)$ (see Condition (a)). Therefore, we still have $r_1 = \mathsf{pred}(r_1)$. Note that, by Condition (d), this equality means $r_1 = \mathsf{min}$.
- For every $i < k$, $\mathsf{pred}(r_{i+1}) = \mathsf{last}(r_i)$ and $\mathsf{left}(r_{i+1}) = r_i$. Let us prove this by recursion on i.

This assertion is fulfilled by $i = 1$: r_2 being a root, it must satisfy (bi) or (bii) (by (b')). If it satisfies (bi), then $\mathsf{pred}(r_2) = F(r_2) = r_2$ and thus $r_2 = \mathsf{min}$ (by (d)). But this contradicts the fact that $r_2 > r_1$. Therefore, (bii) must hold for r_2. It implies $\mathsf{pred}(r_2) = \mathsf{last}\,\mathsf{left}(r_2)$ and consequently $\mathsf{left}(r_2) \leq \mathsf{last}\,\mathsf{left}(r_2) = \mathsf{pred}(r_2) < r_2$ (by (a) and (d)). Then $\mathsf{left}(r_2)$ is a root (by (b')) strictly smaller than r_2. That is: $\mathsf{left}(r_2) = r_1$ and the equality $\mathsf{pred}(r_2) = \mathsf{last}\,\mathsf{left}(r_2)$ becomes $\mathsf{pred}(r_2) = \mathsf{last}(r_1)$.

Now, consider $j \in \{2, \ldots, k-1\}$ and assume that the recurrence hypothesis is satisfied for each $i < j$. The node r_{j+1} is a non minimal root. For the same reasons than r_2, it fulfils (bii) and $\mathsf{left}(r_{j+1})$ is a root strictly smaller than r_{j+1}. Let $i \leq j$ be such that $\mathsf{left}(r_{j+1}) = r_i$. By the recurrence hypothesis, $\mathsf{last}(r_i) = \mathsf{pred}(r_{i+1})$. In

the same time, $\mathsf{last}\,\mathsf{left}(r_{j+1}) = \mathsf{pred}(r_{j+1})$ (by (bii)). Consequently, $\mathsf{pred}(r_{j+1}) = \mathsf{pred}(r_{i+1})$ and, since $r_{j+1} \neq \mathsf{min}$ and $r_{i+1} \neq \mathsf{min}$: $r_{j+1} = r_{i+1}$. Therefore, $i = j$ and $\mathsf{left}(r_{j+1}) = r_j$, $\mathsf{last}(r_j) = \mathsf{pred}(r_{j+1})$.

So, we have proved the following sequence of inequalities:

$$[r_1, \mathsf{last}(r_1)] < [r_2, \mathsf{last}(r_2)] < \cdots < [r_k, \mathsf{last}(r_k)], \tag{11}$$

with $r_1 = \mathsf{min} = \mathsf{left}(r_1)$, $\mathsf{pred}(r_{i+1}) = \mathsf{last}(r_i)$ and $\mathsf{left}(r_{i+1}) = r_i$. This shows that left fulfils its expected interpretation, as far as roots are concerned. Also, the schema (11) obviously shows that the intervals $[r_i, \mathsf{last}(r_i)]$ are disjoint. However we have seen before that for each i, $\mathsf{desc}_F(r_i) \subseteq [r_i, \mathsf{last}(r_i)]$. As the subsets $\mathsf{desc}_F(r_i)$, $i = 1, \ldots, k$, clearly form a partition of D, these last inclusions lead to the demanded equalities: $\mathsf{desc}_F(r_i) = [r_i, \mathsf{last}(r_i)]$.

Inductive step ($H_\ell \Rightarrow H_{\ell+1}$)

Let us now assume that H_ℓ holds for a given $\ell \geq 0$. In order to prove that $H_{\ell+1}$ holds, we only have to prove that for each node x of level ℓ and for each child y of x in the forest: $\mathsf{desc}_F(y) = [y, \mathsf{last}(y)]$ and $\mathsf{left}(y)$ is the left sibling of y. So, suppose that $x \in D$ is a node of level ℓ and denote by $x_1 < x_2 < \cdots < x_p$ the strictly increasing list of its children. We prove in "one move" that all the x_i's fulfil the expected conditions. The proof is almost the same as in the base case:

First, x_1, which is not a root, must fulfil (bi): otherwise, by (b'), it would fulfil $(biii)$, that is: $\mathsf{left}(x_1)$ is not a root, $F\mathsf{left}(x_1) = F(x_1)$ and $\mathsf{last}\,\mathsf{left}(x_1) = \mathsf{pred}(x_1) < x_1$. Thus, $\mathsf{left}(x_1)$ would be a sibling of x_1 strictly smaller than x_1: a contradiction. Thus, $\mathsf{pred}(x_1) = F(x_1) = x$ and $\mathsf{left}(x_1) = x_1$.

Now, we can inductively prove that for each $i < p$, $\mathsf{pred}(x_{i+1}) = \mathsf{last}(x_i)$ and $\mathsf{left}(x_{i+1}) = x_i$: by their definition, all the x_i's, $i > 1$, have to satisfy Condition $(biii)$ (if such an x_i satisfies (bi), then $\mathsf{pred}(x_i) = F(x_i) = x = \mathsf{pred}(x_1)$: a contradiction). For each $i > 1$, this implies that $\mathsf{left}(x_i)$ is a sibling of x_i strictly smaller than x_i and such that $\mathsf{last}\,\mathsf{left}(x_i) = \mathsf{pred}(x_i)$. For $i = 2$, this imposes $\mathsf{left}(x_2) = x_1$ and $\mathsf{last}(x_1) = \mathsf{pred}(x_2)$. Then, one shows recursively, similarly to the base case, that $\mathsf{pred}(x_{i+1}) = \mathsf{last}(x_i)$ and $\mathsf{left}(x_{i+1}) = x_i$ for each $i = 1, \ldots, p-1$. Thus we have proved the decomposition scheme:

$$x < [x_1, \mathsf{last}(x_1)] < [x_2, \mathsf{last}(x_2)] < \cdots < [x_p, \mathsf{last}(x_p)], \tag{12}$$

with $\mathsf{pred}(x_1) = x$, $\mathsf{pred}(x_{i+1}) = \mathsf{last}(x_i)$ and $\mathsf{left}(x_{i+1}) = x_i$. The scheme (12) obviously implies that the sets $\{x\}$, $[x_1, \mathsf{last}(x_1)]$, ..., $[x_p, \mathsf{last}(x_p)]$ are mutually disjoints. Moreover, by the induction hypothesis, we have $\mathsf{desc}_F(x) = [x, \mathsf{last}(x)]$. Since for each i, $\mathsf{desc}_F(x_i) \subseteq [x_i, \mathsf{last}(x_i)]$ and since, trivially, the sets $\{x\}$, $\mathsf{desc}_F(x_1)$, ..., $\mathsf{desc}_F(x_p)$ form a partition of $\mathsf{desc}_F(x)$, the same argument as above allows to conclude that $\mathsf{desc}_F(x_i) = [x_i, \mathsf{last}(x_i)]$ for each i. This finally assures that $<$, last and left fit their expected interpretations and concludes the proof of Proposition 5.1.❑

COROLLARY 5.1. $\mathrm{ESO}_1[\text{FULL-FOREST}] = \mathrm{ESO}_1$.

Proof. Proposition 5.1 precisely states that the set of structures FULL-FOREST is definable in ESO_1[FULL-LINORD], *via* the formula:

$$\exists(<, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}) \in \text{FULL-LINORD}\ \forall x :$$

$$\{\ F(x) \leq x \leq \mathsf{last}(x) \leq \mathsf{last}F(x)\ \} \wedge$$

$$\left\{ \begin{array}{c} \mathsf{pred}(x) = F(x) \wedge \mathsf{left}(x) = x \\ \vee \\ x = F(x) \wedge \mathsf{left}(x) = F\mathsf{left}(x) \wedge \mathsf{last}\,\mathsf{left}(x) = \mathsf{pred}(x) \\ \vee \\ x \neq F(x) \wedge \mathsf{left}(x) \neq F\mathsf{left}(x) \wedge F\mathsf{left}(x) = F(x) \wedge \mathsf{last}\,\mathsf{left}(x) = \mathsf{pred}(x) \end{array} \right\}$$

$$\wedge\ \{\ (F(x) = x \rightarrow \mathsf{root}(x) = x) \wedge \mathsf{root}F(x) = \mathsf{root}(x)\ \}$$

As ESO_1[FULL-LINORD] = ESO_1, by Lemma 5.1, FULL-FOREST is also definable in ESO_1, and the conclusion follows from Implication (7).❑

We shall now state a generalization of this result, using the formalism described in the previous subsection. First, let us introduce some new notations:

- FULL-FOREST+TC is the set of structures $\langle D, F, \mathsf{root}, \mathsf{last}, \mathsf{left}, F^* \rangle$ such that: $(F, \mathsf{root}, \mathsf{last}, \mathsf{left}) \in$ FULL-FOREST(D) and F^* is the transitive closure of F (that is: $F^*(x, y)$ iff $\exists i \in \mathbb{N} : F^i(x) = y$) and
- FULL-TREE+TC is the set of structures $\langle D, T, \mathsf{root}, \mathsf{last}, \mathsf{left}, T^* \rangle$ which are in FULL-FOREST+TC and such that T is a tree (*i.e.* T is connected).

COROLLARY 5.2 (in collaboration with C. Lautemann and S. Ranaivoson).
The following equalities hold:

(a) ESO_1[FULL-FOREST+TC] = ESO_1 *and*
(b) ESO_1[FULL-TREE+TC] = ESO_1

Proof.

(a) Suppose F is a forest of domain D, $<$ is a prefix order over F and last is the last function of F with respect to $<$. Then for any $x, y \in D$, $F^*(x, y)$ holds iff $x \in \mathsf{desc}_F(y)$. But $\mathsf{desc}_F(y) = [y, \mathsf{last}(y)]$. Therefore we have: $F^*(x, y)$ iff $y \leq x \leq \mathsf{last}(y)$.

Now, let us denote by Θ the quantifier-free matrix of the formula described in the proof of Corollary 5.1, so that FULL-FOREST is defined by the formula:

$$\exists(<, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}) \in \text{FULL-LINORD}\ \forall x :$$
$$\Theta(x, <, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}, F, \mathsf{root}, \mathsf{last}, \mathsf{left})$$

Since this formula forces $(F, <)$ to be a prefix ordered forest and last to be the associated last function, it is clear that each formula $\exists(F, \mathsf{root}, \mathsf{last}, \mathsf{left}, F^*)\Phi$ of

ESO_1[FULL-FOREST+TC] is equivalent to the formula:

$$\exists(<, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}) \in \text{FULL-LINORD}\ \exists F, \mathsf{root}, \mathsf{last}, \mathsf{left}, F^* \ : \\ (\forall x\, \Theta(x)) \wedge \Phi'$$

where Φ' is obtained from Φ by replacing each atomic formula $F^*(t_1(x), t_2(x))$ (where $t_1(x)$, $t_2(x)$ are terms over the only first-order variable x occurring in Φ) by the formula $t_1(x) \leq t_2(x) \leq \mathsf{last}(t_1(x))$. As this formula clearly belongs to ESO_1[FULL-LINORD], it can be written in ESO_1, by Lemma 5.1. Finally, each formula of ESO_1[FULL-FOREST+TC] is thus proved logically equivalent to a formula of ESO_1 and the result follows.

(b) A tree is a forest with only one root. Therefore, each formula

$$\exists(T, \mathsf{root}, \mathsf{last}, \mathsf{left}, T^*) \in \text{FULL-TREE+TC} \ : \ \Phi$$

in ESO_1[FULL-TREE+TC] is equivalent to the formula:

$$\exists(T, \mathsf{root}, \mathsf{last}, \mathsf{left}, T^*) \in \text{FULL-FOREST+TC}\ \exists r \ : \ (\forall x : \mathsf{root}(x) = r) \wedge \Phi,$$

which can be written in ESO_1, by (a).

This concludes the proof of Corollary 5.2.❑

We conclude this subsection by a remark which relates the statements of Corollary 5.2 to the way we will use them in the following. First, let us introduce the following definitions:

- ROOTED-FOREST is the set of structures $\langle D, F, \mathsf{root}\rangle$ such that $F \in \text{FOREST}(D)$ and root is its root function ;
- FOREST+TC is the set of structures $\langle D, F, F^*\rangle$ such that $F \in \text{FOREST}(D)$ and F^* is its transitive closure.
- ROOTED-FOREST+TC is the set of structures $\langle D, F, \mathsf{root}, F^*\rangle$ such that (F, root) is in ROOTED-FOREST(D) and (F, F^*) is in FOREST+TC.
- TREE is the set of structures $\langle D, T\rangle$ such that T is a tree.
- TREE+TC is the set of structures $\langle D, T, T^*\rangle$ such that $T \in$ TREE and T^* is its transitive closure.

Remark. The above sets of structures are all complete restrictions of either the set FULL-FOREST+TC or the set FULL-TREE+TC. Therefore, by Corollary 5.2 and Implication (8), existential quantifications over these sets does not enlarge ESO_1.

5.3. Transitive closure of a function

Let D be a finite domain of cardinality $n \geq 2$ and f a unary function over D. We also call f the directed graph of f, that is the binary structure (D, E), where $E(x, y)$ holds if $f(x) = y$. The shape of functional graphs are well-known: they look as forests, except that the root of any connected component can be replaced by a cycle. Such a graph is represented in Figure 4.

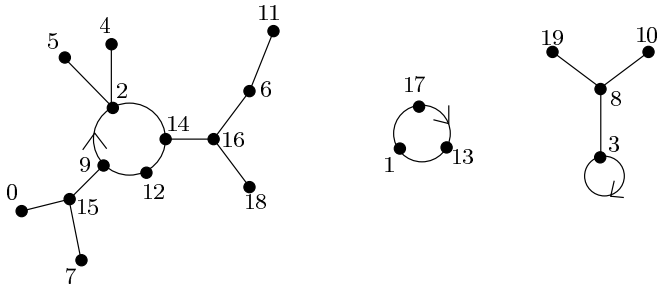

FIG. 4. a unary function f_0 over $D = \{0, \ldots, 19\}$

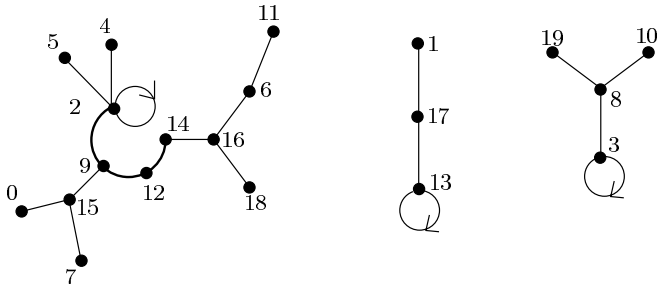

FIG. 5. a forest F_0 obtained from f_0

Let us now consider a forest F of domain D. We say that F is *obtained from* f if each root of F is on a circuit of f and if furthermore F and f coincide on every $x \in D$ which is not a root of F. A forest F_0 obtained from the function f_0 is given in Figure 5.

LEMMA 5.3. *Let D be a finite domain and $f : D \to D$. Let F be a forest of domain D, root_F its associated root function and F^* its transitive closure.*

(a) F is obtained from f if and only if, for every $x \in D$:

$\mathsf{root}_F f(x) = \mathsf{root}_F(x)$ *and* $F(x) \neq x \to F(x) = f(x)$.

(b) If F is obtained from f, then for every $x, y \in D$:

$f^*(x, y)$ *iff* $F^*(x, y) \vee (F^*(x, \mathsf{root}_F(x)) \wedge F^*(f\mathsf{root}_F(x), y))$.

Proof. Let us first notice that if $F \in \text{FOREST}(D)$ fulfils (a) and if $x_1, \ldots, x_p$ is a tuple of nodes distinct from any root of F, then we clearly have:

$$x_1 x_2 \ldots x_p \text{ is a path in } f \text{ iff } x_1 x_2 \ldots x_p \text{ is a path in } F. \qquad (*)$$

(a) The *"only if"* condition is trivial. Let us check the *"if"* one: the equality $\mathsf{root}_F f(x) = \mathsf{root}_F(x)$ guarantees, by induction on f, that the function root_F is constant on each connected component of f. Therefore, each connected component C of f contains exactly one root of F, and we only have to verify that this root lies on the circuit of C. Assume, for the sake of

a contradiction, that the f-circuit $x_1x_2\dots x_px_1$ of C does not contain any root of F. Then, by $(*)$, $x_1x_2\dots x_px_1$ is also a circuit of F. This implies $x_1 = x_2 = \dots = x_p$ (since F is a forest) and thus, $x_1 = F(x_1)$: a contradiction.

(b) *"Only if".* Let $x = x_1 \xrightarrow{f} x_2 \xrightarrow{f} \dots \xrightarrow{f} x_p \xrightarrow{f} y$ be a simple path from x to y in f. If none of the x_i's is a root for F, then this path is also a path in F and we have $F^*(x,y)$. Otherwise, let i be such that $F(x_i) = x_i$. This i is unique, since there is only one root in each connected component of f and since the path $xx_2\dots x_py$ is simple. Consequently, $xx_2\dots x_i$ and $x_{i+1}\dots y$ are paths in f that do not contain any F-root. Hence they are paths in F and we have $F^*(x,x_i)$ and $F^*(x_{i+1},x)$. The equalities $x_i = \mathsf{root}_F(x)$ and $x_{i+1} = f(x_i)$ yield the conclusion.

"If". If $F^*(x,y)$, then either $x = y$, and the conclusion is clear, or there exists a simple path $x = x_1 \xrightarrow{F} \dots \xrightarrow{F} x_p \xrightarrow{F} y$ in F such that $F(x_i) \neq x_i$ for every i (since the path is simple). By $(*)$, this F-path is therefore also a path in f and we have $f^*(x,y)$. If $F^*(x,\mathsf{root}_F(x)) \wedge F^*(f\mathsf{root}_F(x),y)$, the previous remark leads to $f^*(x,\mathsf{root}_F(x)) \wedge f^*(f\mathsf{root}_F(x),y)$ and therefore, to $f^*(x,y)$.

■

The proof of Assertion (b) leads to the following remark, that will be used in the next section:

Remark. Suppose that $f^*(x,y)$ holds and call P the simple path from x to y in f. Then:

either P is also the path from x to y in F,
or P has the form $xx_2\dots\mathsf{root}_F(x)f\mathsf{root}_F(x)\dots x_py$,
where $x\dots\mathsf{root}_F(x)$ and $f\mathsf{root}_F(x)\dots y$ are also paths in F.

The previous lemma has an immediate consequence, in terms of "robustness" of the class ESO_1. Let us first define the following class of structures:

FUNCTION+TC is the set of structures $\langle D, f, f^*\rangle$ where f is a unary function over D and f^* is the transitive closure of f.

COROLLARY 5.3. $\mathrm{ESO}_1[\text{FUNCTION+TC}] = \mathrm{ESO}_1$.

Proof. It immediately follows from Lemma 5.3 that each formula

$$\Psi \equiv \exists(f, f^*) \in \text{FUNCTION+TC} \ : \ \Phi(f, f^*)$$

of $\mathrm{ESO}_1[\text{FUNCTION+TC}]$ is equivalent to the formula Ψ':

$$\exists(F, \mathsf{root}_F, F^*) \in \text{ROOTED-FOREST+TC} :$$
$$(\forall x : \ \mathsf{root}_F f(x) = \mathsf{root}_F(x) \wedge F(x) \neq x \to F(x) = f(x))$$
$$\wedge\ \Phi'(f, F, \mathsf{root}_F),$$

where Φ' is obtained from Φ by replacing each atomic formula $f^*(t_1(x), t_2(x))$ by the formula:

$$F^*(t_1(x), t_2(x)) \vee (F^*(t_1(x), \mathsf{root}_F(t_1(x))) \wedge F^*(f\mathsf{root}_F(t_1(x)), t_2(x))).$$

The formula Ψ' so obtained is in ESO_1[ROOTED-FOREST+TC] and thus, by Corollary 5.2, it can be written in ESO_1. Hence the result.❑

5.4. Distance in an ordered functional graph

The goal of this subsection is to refine the main result of the previous one: we have stated in Corollary 5.3 that the transitive closure of a unary function can be expressed in ESO_1. That is, we can express, in this logic, that there exists a path between two nodes of a given functional graph. But *what about the length of this path ?* Can we also define it in ESO_1? In other words, can we define in ESO_1 the function dist_f, with respect to a given unary function f, that maps each pair $(x, y) \in f^*$ onto the length of the shortest f-path between x and y ? A problem arises in this formulation: such a function dist_f has to take its values in $\mathbb{N}$. Thus it cannot be described, *a priori*, as the interpretation of a function symbol over a domain D. But we have seen, in Subsection 5.1, how to overcome this difficulty: when a domain D is given with a built-in linear order, its elements can be identified to the nonnegative integers 0, 1, ..., $|D| - 1$. As the distance function maps each ordered pair (x, y) of $D \times D$ onto an integer *strictly smaller than* $|D|$, it can be represented by a binary function over D, provided D is equipped with a linear order. And this will be also the case for some other functions (height, length) considered in this subsection. Once our above question will be well reformulated, we shall answer it positively (Corollary 5.4). For this purpose, let us briefly introduce some new definitions:

Let $(D, <)$ be a linearly ordered domain, whose elements are denoted by 0, 1, ..., $|D| - 1$. Let f be a unary function over D.

When $f^*(x, y)$ holds, there exists a unique simple path from x to y in f. We call the *distance from x to y in f* the length of this path (*i.e.* the number of edges occurring in the path). It follows that, if $f^*(x, y)$ holds, the distance from x to y in f is the least $i < |D|$ such that $y = f^i(x)$. We call the *distance function associated to f according to $<$* the binary function over D that maps each pair $(x, y) \in f^*$ to the (representative of the) distance from x to y in f. For the sake of completeness, we assume that this function maps each $(x, y) \notin f^*$ to 0.

The *length of x in f* is the length of the circuit of the f-connected component on which x lies. We call *length function associated to f with respect to $<$* the unary function over D that maps each $x \in D$ onto the (representative of the) length of x in f.

For the function represented in Figure 4 for instance, the distance from 11 to 2 is 6 and the length of 11 is 4.

Besides, if F is in FOREST(D), the *height of x in F* is the distance from x to its F-root, that is, the smallest $i < |D|$ such that $F(F^i(x)) = F^i(x)$. We call *height function associated to F with respect to $<$* the unary function over D that maps each $x \in D$ onto the (representative of the) height of x in F.

Let f and dist be two function symbols of respective arities 1 and 2. We denote by FUN+DIST$_<$ the set of ordered structures $\langle D, <, f, \mathsf{dist}\rangle$ of signature (f, dist) in which dist is the distance function associated to the unary function f with respect to the linear order $<$.

LEMMA 5.4. *Let $\langle D, f, f^*\rangle \in$ FUNCTION+TC and $<$ be a linear order over D. Let $\mathsf{dist}_f : D \times D \to D$. Then dist_f is the distance function associated to f with respect to $<$ iff:*

there exists $(F, \mathsf{root}_F, F^*) \in$ ROOTED-FOREST+TC(D),

there exists $(+, \times, -,\ \mathsf{div}\ ,\ \mathsf{mod}\ ) \in$ ARITH$_<$(D),

there exist $\mathsf{height}_F, \mathsf{length}_f : D \to D$,

such that the following conditions $(a-d)$ hold:

$(a)\ \forall x :\ \mathsf{root}_F f(x) = \mathsf{root}_F(x)\ \wedge\ (F(x) \neq x \to F(x) = f(x))$;

$(b)\ \forall x :\ F(x) = x \to \mathsf{height}_F(x) = 0\ \wedge$
$\quad F(x) \neq x \to \mathsf{height}_F(x) = 1 + \mathsf{height}_F F(x)$;

$(c)\ \forall x :\ \mathsf{length}_f(x) = 1 + \mathsf{height}_F f\, \mathsf{root}_F(x)$;

(d) *for every $x, y, z \in D$, the assertion $z = \mathsf{dist}_f(x, y)$ is equivalent to:*

$$\begin{array}{ll} & \{\, f^*(x,y) \wedge F^*(x,y) \wedge z = \mathsf{height}_F(x) - \mathsf{height}_F(y)\,\} \\ \vee & \{\, f^*(x,y) \wedge \neg F^*(x,y) \wedge z = \mathsf{height}_F(x) + \mathsf{length}_f(x) - \mathsf{height}_F(y)\,\} \\ \vee & \{\, \neg f^*(x,y) \wedge z = 0\,\}. \end{array}$$

Proof. The forward implication is easy (see Figure 5). We prove the converse implication by looking at the following consequences of statements $(a-d)$:

(a) This is the condition for F to be *obtained* from f (see Lemma 5.3).

(b) By this condition, height_F takes the value 0 on each root, and each further step outside the loop increases its value by one. This inductively forces height_F to be the height function associated to F according to $<$.

(c) Let $x \in D$. Recall that the length of x in f is the length of the cycle of the f-connected component of x. As this cycle contains $\mathsf{root}_F(x)$, it can be written:

$$\mathsf{root}_F(x) \xrightarrow{f} f\mathsf{root}_F(x) \xrightarrow{f} \ldots \xrightarrow{f} \mathsf{root}_F(x).$$

Thus, the length of x in f is 1 plus the length of the f-path $f\mathsf{root}_F(x) \xrightarrow{f} \ldots \xrightarrow{f} \mathsf{root}_F(x)$. But this path is also a path in F and its length can be viewed as the distance from $f\mathsf{root}_F(x)$ to $\mathsf{root}_F(x)$ in F, that is, as the height of $f\mathsf{root}_F(x)$ in F. Therefore, the length of x in f is $1 + \mathsf{height}_F f\, \mathsf{root}_F(x)$ and Condition (c) implies that length_f is the length function associated to f according to $<$.

(d) Let us temporarily denote by Dist_F (resp. Dist_f) the distance function associated to F (resp. f) according to $<$. Since F is a forest, we clearly have, for any $x \in D$:

$$\mathsf{Dist}_F(x, \mathsf{root}_F(x)) = \mathsf{height}_F(x)$$

and consequently, for any $(x, y) \in F^*$:

$$\begin{aligned} \mathsf{Dist}_F(x, y) &= \mathsf{Dist}_F(x, \mathsf{root}_F(x)) - \mathsf{Dist}_F(y, \mathsf{root}_F(y)) \\ &= \mathsf{height}_F(x) - \mathsf{height}_F(y). \end{aligned}$$

Now, let us "compute" $\mathsf{Dist}_f(x, y)$ according to the situation of the ordered pair (x, y): if $f^*(x, y)$ does not hold, then $\mathsf{Dist}_f(x, y) = 0$; if $f^*(x, y)$ holds, let P be the simple f-path from x to y. Then $\mathsf{Dist}_f(x, y)$ is the length of this path and we have, by the remark of page 35: either P is also an F-path, or P has the form

$$P = x \xrightarrow{f} \ldots \xrightarrow{f} \mathsf{root}_F(x) \xrightarrow{f} f(\mathsf{root}_F(x)) \xrightarrow{f} \ldots \xrightarrow{f} y,$$

where $x \ldots \mathsf{root}_F(x)$ and $f\mathsf{root}_F(x) \ldots y$ are also paths in F. The first case yields

$$\mathsf{Dist}_f(x, y) = \mathsf{Dist}_F(x, y) = \mathsf{height}_F(x) - \mathsf{height}_F(y)$$

and the second one:

$$\begin{aligned} \mathsf{Dist}_f(x, y) &= \mathsf{Dist}_F(x, \mathsf{root}_F(x)) + 1 + \mathsf{Dist}_F(f\mathsf{root}_F(x), y) \\ &= \mathsf{height}_F(x) + 1 + \mathsf{height}_F(f\mathsf{root}_F(x)) - \mathsf{height}_F(y). \end{aligned}$$

That is, by Condition (c):

$$\mathsf{Dist}_f(x, y) = \mathsf{height}_F(x) + \mathsf{length}_f(x) - \mathsf{height}_F(y).$$

It is now easily seen that Condition (d) implies the equality $\mathsf{dist}_f = \mathsf{Dist}_F$.

This concludes the proof of Lemma 5.4.❑

COROLLARY 5.4. $\mathrm{ESO}_1[\text{FUN+DIST}_<] = \mathrm{ESO}_1[<]$.

Proof. By Lemma 5.4, every formula

$$\Psi \equiv \exists(f, \mathsf{dist}_f) \in \text{FUN+DIST}_< \ : \ \Phi(<, \sigma, f, \mathsf{dist}_f)$$

of $\mathrm{ESO}_1^{\sigma}[\text{FUN+DIST}_<]$ has the same ordered models as the formula:

$$\begin{gathered} \exists(f, f^*) \in \text{FUNCTION+TC} \\ \exists(F, \mathsf{root}_F, F^*) \in \text{ROOTED-FOREST+TC} \\ \exists(+, \times, -, \ \mathsf{div} \ , \ \mathsf{mod} \ ) \in \text{ARITH}_< \\ \exists \mathsf{height}_F \, \exists \mathsf{length}_f \end{gathered}$$

$$\forall x \left\{ \begin{array}{l} \mathsf{root}_F f(x) = \mathsf{root}_F(x) \ \wedge \ (F(x) \neq x \rightarrow F(x) = f(x)) \ \wedge \\ (F(x) = x \rightarrow \mathsf{height}_F(x) = 0) \ \wedge \\ (F(x) \neq x \rightarrow \mathsf{height}_F(x) = 1 + \mathsf{height}_F F(x)) \ \wedge \\ \mathsf{length}_f(x) = 1 + \mathsf{height}_F f \, \mathsf{root}_F(x) \end{array} \right\} \wedge \ \Phi',$$

where Φ' is obtained from Φ by replacing each atomic subformula

$$t_x = \mathsf{dist}_f(u_x, v_x)$$

(where u_x, v_x, t_x are terms built over the only first-order variable x occurring in Φ) by the formula

$$\begin{array}{ll} & \{\, f^*(u_x, v_x) \wedge F^*(u_x, v_x) \wedge t_x = \mathsf{height}_F(u_x) - \mathsf{height}_F(v_x) \,\} \\ \vee & \{\, f^*(u_x, v_x) \wedge \neg F^*(u_x, v_x) \wedge t_x = \mathsf{height}_F(u_x) - \mathsf{height}_F(v_x) + \mathsf{length}_f(u_x) \,\} \\ \vee & \{\, \neg f^*(u_x, v_x) \wedge t_x = 0 \,\}. \end{array}$$

The formula Ψ' so obtained can be written in $\mathrm{ESO}_1^{\sigma}[<]$ by Corollary 5.3, Corollary 5.2, Lemma 5.2 and Implication (9) of Subsection 5.1.❑

Remark. We have said that this last result holds in case the involved structures are equipped with a built-in linear order. Otherwise, the statement of Corollary 5.4 has no precise meaning since the notion of "distance function" must refer to a linear order. But what happens if we consider an *existentially quantified* linear order $<$? We are still able, in this case, to build the functions height and length and the set $\mathrm{ARITH}_<$ related to $<$, as in Lemma 5.4. And we can in turn define a notion of distance associated to a given unary function f according to *this* linear order. Of course, an assertion of the form $t_x = \mathsf{dist}_f(u_x, v_x)$ for such a function dist_f is of no intrinsic interest since it has different meanings according to the choice of $<$. On the other hand, an equality such as $\mathsf{dist}_f(u_x, v_x) = \mathsf{dist}_f(u'_x, v'_x)$ is order invariant. That is, if it holds for (the distance function related to) a given linear order, then it holds for any linear order. Thus, our ability to express the distance function of f according to a (existentially quantified) linear order $<$ allows to say, in ESO_1, that two ordered pairs of nodes (u_x, v_x) and (u'_x, v'_x) are linked by paths with the same number of edges in a given functional graph. In other words, the logic ESO_1 is not enlarged if we allow subformulas of the form $\mathsf{dist}_f(u_x, v_x) = \mathsf{dist}_f(u'_x, v'_x)$, where dist_f is a distance function associated to a unary function f involved in the formula. The details are left to the reader.

The last result of this section does not refer explicitly to graph properties, although it is a straightforward consequence of the ESO_1-definability of the distance function of a functional graph. By the definability of transitive closure, we can easily assert in ESO_1 that a unary function g is obtained by iterated compositions of a given unary function f (*i.e.* for each x, there exists $i \in \mathbb{N}$ such that $g(x) = f^i(x)$). Indeed, this assertion is equivalent to $g \subseteq f^*$, where f^* denotes the transitive closure of f and g is viewed as an edge relation, thus it can be translated in ESO_1 by: $\forall x f^*(x, g(x))$. Now, if we restrict our attention to ordered structures, the definability of distance functions allows us to refine the previous assertion by specifying, for each x, the number i of compositions of f needed to pass from x to $g(x) = f^i(x)$. This is the meaning of the next lemma. Let us formalize it:

Let $(D, <)$ be a linearly ordered domain and f, g, h be three unary functions over D. We denote by $g = f^h$ the fact that $g \subseteq f^*$ and that for each x, $h(x)$ is the

(representative of the) number of applications of f over x needed to pass from x to $g(x)$. In other words, $g = f^h$ means:

$$\forall x \in D:\ g(x) = f^{h(x)}(x) = \underbrace{ff\ldots f}_{h(x)\text{ times}}(x)$$

Now, we denote by ITER-COMPO$_<$ (for *iterated composition*) the set of ordered structures $\langle D, <, f, g, h\rangle$ such that $g = f^h$. Then:

LEMMA 5.5. $\mathrm{ESO}_1[\text{ITER-COMPO}_<] = \mathrm{ESO}_1[<]$.

Proof. Let f be a unary function over a linearly ordered domain $(D, <)$. Let length_f and dist_f be its corresponding length and distance functions according to $<$. Consider $x, y \in D$ and suppose that there exists $i \in \mathbb{N}$ such that $y = f^i(x)$ (*i.e.*, suppose $f^*(x, y)$). Then, by definition of the distance function, $\mathsf{dist}_f(x, y)$ is the least such i. Furthermore, if y is not on a circuit of f then there is a unique path in f from x to y and therefore $\mathsf{dist}_f(x, y)$ is the unique i such that $y = f^i(x)$. Otherwise, there may exist some $j > \mathsf{dist}_f(x, y)$ such that $y = f^j(x)$. More precisely, if y lies on a circuit of f, then:

- either y is a loop, and $\forall j > \mathsf{dist}_f(x, y)$: $y = f^j(x)$
- or y lies on a non-empty circuit of f (or equivalently: $\mathsf{dist}_f(f(y), y) > 0$), and therefore $f^{\mathsf{length}_f(x)}(y) = y$, which yields, for any $j > \mathsf{dist}_f(x, y)$:

$$y = f^j(x) \text{ iff } j - \mathsf{dist}_f(x, y) \equiv 0\,[\mathsf{length}_f(x)]\,.$$

Finally, under the assumption $f^*(x, y)$, the assertion $y = f^i(x)$ is equivalent to:

$$\begin{gathered}
\{\, i = \mathsf{dist}_f(x, y)\,\} \\
\text{or} \\
\{\, f(y) = y \ \text{ and } \ i > \mathsf{dist}_f(x, y)\,\} \\
\text{or} \\
\left\{ \begin{array}{c} \mathsf{dist}_f(f(y), y) > 0 \ \text{ and } \ i > \mathsf{dist}_f(x, y) \ \text{ and } \\ (i - \mathsf{dist}_f(x, y)) \bmod \mathsf{length}_f(x) = 0 \end{array} \right\}
\end{gathered}$$

Consequently, for any functions $g, h : D \to D$ and any $x \in D$ such that $f^*(x, g(x))$, the assertion $g(x) = f^{h(x)}(x)$ is equivalent to:

$$\begin{gathered}
\{\, h(x) = \mathsf{dist}_f(x, g(x))\,\} \\
\text{or} \\
\{\, fg(x) = g(x) \ \text{ and } \ h(x) > \mathsf{dist}_f(x, g(x))\,\} \\
\text{or} \\
\left\{ \begin{array}{c} \mathsf{dist}_f(fg(x), g(x)) > 0 \ \text{ and } \ h(x) > \mathsf{dist}_f(x, g(x)) \ \text{ and } \\ (h(x) - \mathsf{dist}_f(x, g(x))) \bmod \mathsf{length}_f(x) = 0 \end{array} \right\}
\end{gathered}$$

Before concluding, it remains to notice that:

- $g = f^h$ iff $(g \subseteq f^*$ and $\forall x \in D : g(x) = f^{h(x)}(x))$;

• $g \subseteq f^*$ iff $(\forall x \in D\colon\ x = g(x)$ or $\mathsf{dist}_f(x, g(x)) > 0)$;

• the function length_f is completely characterized by the following facts: it is invariant on each connected component of f ; it maps each loop onto 0 ; for each x lying on a non-empty f-circuit, *i.e.* for each x such that $x \xrightarrow{f} f(x) \xrightarrow{f} \ldots \xrightarrow{f} x$ is a non-empty f-path, length_f take the value $1 + \mathsf{dist}_f(f(x), x)$.

It is now easy to see that for every signature σ, any formula

$$\exists (f, g, h) \in \text{ITER-COMPO}_< : \Phi(\sigma)$$

of $\text{ESO}_1^\sigma[\text{ITER-COMPO}_<]$ is equivalent to the formula:

$$\begin{array}{c} \exists (+, \times, -,\ \mathsf{div}\ ,\ \mathsf{mod}\ ) \in \text{ARITH}_< \\ \exists (f, \mathsf{dist}_f) \in \text{FUN+DIST}_< \ \exists \mathsf{length}_f \ \exists g \ \exists h\ : \\ \Psi \ \wedge \ \Phi \end{array}$$

where Ψ is the conjunction of the following first-order formulas over the signature $\{+, \times, -,\ \mathsf{div}\ ,\ \mathsf{mod}\ , f, \mathsf{dist}_f, \mathsf{length}_f, g, h\}$:

$$\begin{array}{lll} \psi_1 \equiv \forall x : & \mathsf{length}_f(f(x)) = \mathsf{length}_f(x) \wedge & \\ & f(x) = x \rightarrow \mathsf{length}_f(x) = 0 \wedge & \\ & \mathsf{dist}_f(f(x), x) > 0 \rightarrow \mathsf{length}_f(x) = 1 + \mathsf{dist}_f(f(x), x) & \\ \psi_2 \equiv \forall x : & fg(x) = g(x) \vee \mathsf{dist}_f(fg(x), g(x)) > 0 & \\ \psi_3 \equiv \forall x : & h(x) = \mathsf{dist}_f(x, g(x)) \vee & \\ & \{\, fg(x) = g(x) \ \wedge \ h(x) > \mathsf{dist}_f(x, g(x)) \,\} \vee & \\ & \left\{ \begin{array}{l} \mathsf{dist}_f(fg(x), g(x)) > 0 \ \wedge \ h(x) > \mathsf{dist}_f(x, g(x)) \wedge \\ (h(x) - \mathsf{dist}_f(x, g(x))) \ \mathsf{mod}\ \mathsf{length}_f(x) = 0 \end{array} \right\} & \end{array}$$

Thus the above formula belongs to $\text{ESO}_1^\sigma[\text{ARITH}_<, \text{FUN+DIST}_<]$. Implication (9) of Subsection 5.1, Lemma 5.2 and Corollary 5.4 yield the conclusion.❑

6. SOME PROBLEMS IN vertexNLIN

Using the logical toolbox of the previous section, we are now in a position to prove in an elegant and concise way that a number of combinatorial problems are in vertexNLIN.

We denote by DIGRAPH the set of finite structures of signature $\{E\}$, where E is a binary relation symbol. We denote by GRAPH the set of finite structures $\langle D, E\rangle \in$ DIGRAPH for which E is symmetric. Here follow some digraph (resp. graph) problems:

HAMILTON $= \{G \in$ GRAPH such that G admits a Hamiltonian cycle$\}$;
CONNEX $= \{G \in$ GRAPH such that G is connected$\}$;
STRONG-CONNEX $= \{G \in$ DIGRAPH such that G is strongly connected$\}$;
BICONNEX $= \{G \in$ GRAPH such that G is biconnected$\}$;
CUBIC-SUBGRAPH $=$
$\{G \in$ GRAPH such that G admits a nonempty cubic partial subgraph$\}$.
(*i.e.* $G = (V, E) \in$ CUBIC-SUBGRAPH iff there exist $V' \subseteq V$ and $E' \subseteq V'^2 \cap E$ such that $V' \neq \emptyset$ and each vertex of the graph (V', E') is of degree 3.)

NON-PLANAR $= \{G \in$ GRAPH such that G is not planar$\}$;

Finally, we denote by f-CENTER the class of connected graphs $G = (V, E)$ that contain an *f-center*, that is a vertex c such that for any vertex $x \in V$, there is a path of length smaller than $f(n-1)$ (where $n = |V|$) that links c to x.

PROPOSITION 6.1. HAMILTON, CONNEX, STRONG-CONNEX, CUBIC-SUBGRAPH, f-CENTER *(for any unary function f definable in* ESO_1*) and* BICONNEX *all belong to* vertexNLIN.

Proof. A graph $G = (V, E)$ is **Hamiltonian** iff we can linearly order its vertices in such a way that two successive vertices are linked by an edge and the maximal vertex is linked to the minimal one. Therefore, HAMILTON is characterized by the following formula:

$$\exists(\leq, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}) \in \text{FULL-LINORD}$$
$$E(\mathsf{max}, \mathsf{min}) \wedge (\forall x \neq \mathsf{max}) E(x, \mathsf{succ}(x))$$

which can be written in ESO_1 by Lemma 5.1.

A graph $G = (V, E)$ is **connected** iff it can be spanned by a tree. This yields a characterization of CONNEX by the following formula:

$$\exists T \in \text{TREE}\, \forall x \,:\, T(x) \neq x \rightarrow E(T(x), x)$$

according to which T is a tree whose all edges that are not loop are in E. This formula can be written in ESO_1 by Corollary 5.2 and by the remark following it.

A directed graph $G = (V, E)$ is **strongly connected** iff there exists a spanning tree T for G and a spanning tree T^r for G^r (the reverse graph (V, E^r) of G, defined by: $E^r(x, y)$ iff $E(y, x)$) with the same root. Thus STRONG-CONNEX is characterized by the formula:

$$(\exists T \in \text{TREE})(\exists T^r \in \text{TREE})\, \forall x :$$
$$\{Tx = x \leftrightarrow T^r x = x\} \wedge \{Tx \neq x \rightarrow (E(Tx, x) \wedge E(x, T^r x))\}$$

which can be written in ESO_1 by Corollary 5.2, by the remark of page 33 and by Implication (9) of Subsection 5.1.

The following sentence characterizes the problem CUBIC-SUBGRAPH:

$$\exists f_1, f_2, f_3\, \exists c \,:$$

$$f_1(c) \neq c \;\wedge\; \forall x \left\{ \begin{array}{c} [f_1(x) = f_2(x) = f_3(x) = x] \;\vee \\ \left[\begin{array}{c} f_1(x), f_2(x), f_3(x) \text{ and } x \text{ are pairwise distinct} \\ \wedge \bigwedge_i E(x, f_i(x)) \;\wedge\; \bigwedge_i \bigvee_j f_j f_i(x) = x \end{array} \right] \end{array} \right\}$$

which is obviously in ESO_1. (Observe that the existence of c such that $f_1(c) \neq c$ guarantees that the cubic subgraph is nonempty.)

Let f be a unary function definable in ESO_1 (*e.g.*, $x^{1/2}$, $\log(x)$, *etc*). We know that dist_T can be defined in $\mathrm{ESO}_1[<]$ if T is a unary function, then, *a fortiori*, if T is a tree. Furthermore, we have seen (see the remark following the proof of Corollary 5.4) that equalities such as $\mathsf{dist}_T(u_x, v_x) = \mathsf{dist}_T(u'_x, v'_x)$ (where u_x, v_x, u'_x, v'_x are first-order terms) are order invariant and therefore, can be expressed in ESO_1 (*i.e.* without help of a built-in linear order). We let the reader verify that it is also the case for inequalities such that $\mathsf{dist}_T(u_x, v_x) \leq f(t_x)$, when f is definable in ESO_1. Now, a graph G has an f-center c iff it is spanned by a tree T of root c such that, for all $x \in V$, $\mathsf{dist}_T(c, x) \leq f(\mathsf{max})$. Therefore, f-CENTER is characterized by the following formula (where $\Delta(f)$ is the ESO_1-formula that defines f):

$$\exists(<, \mathsf{pred}, \mathsf{succ}, \mathsf{min}, \mathsf{max}) \in \text{FULL-LINORD}\ \exists T \in \text{TREE}\ \exists f\ \exists c\ \forall x\ :$$

$$(\, x \neq T(x) \rightarrow E(T(x), x)\,)\ \wedge\ T(c) = c\ \wedge\ \mathsf{dist}_T(c, x) \leq f(\mathsf{max})\ \wedge\ \Delta(f)$$

which can be written in ESO_1 by the above remarks.

The definability of BICONNEX in ESO_1 is justified by the following well known result (see [1], for example):

LEMMA 6.1. *A graph $G = \langle V, E\rangle$ is* **biconnected** *iff there exists a spanning tree T for G (e.g. its depth-first search spanning tree) such that the following conditions hold: (i) the root of T has at most one child ; (ii) for any vertex $x \in V$ which is neither the root of T nor a child of the root, there is some vertex y in the subtree $\mathsf{desc}_T(x)$ which is adjacent in G to a proper T-ancestor z of $T(x)$ (that is, to a T-ancestor z of $TT(x)$).*

Hence we get a characterization of our problem by the following formula:

$$\begin{gathered}\exists(T, T^*) \in \text{TREE}{+}\text{TC}\\ \forall x : T(x) \neq x \rightarrow E(x, T(x)) \wedge\\ \exists u \forall x : (TT(x) = T(x) \wedge T(x) \neq x) \rightarrow x = u \wedge\\ \forall x \exists y \exists z : TT(x) \neq T(x) \rightarrow (T^*(y, x) \wedge T^*(TT(x), z) \wedge E(y, z))\end{gathered}$$

whose prenex form is in ESO_1 by Corollary 5.2 and by the remark that follows it. This concludes the proof of Proposition 6.1.❑

We are now going to prove that NON-PLANAR belongs to the class vertexNLIN. In order to build a logical ESO_1-characterization of NON-PLANAR we could use Kuratowski's characterization: a graph is not planar iff it contains a subgraph homeomorphic to K_5 or $K_{3,3}$. It could be done by expressing that there exist several mutually disjoint paths between some specified pairs of vertices. Because of the technicality of such an assertion, we prefer to use another characterization of nonplanar graphs given by Harary (see [7], Theorem 4.11 p. 100, for instance). Let us first recall that a graph $\mathcal{H}$ is said *contractible* to some graph $\mathcal{H}'$ if $\mathcal{H}$ can be transformed into $\mathcal{H}'$ by successive identifications of pairs of adjacent vertices. Then we have:

PROPOSITION 6.2 (Harary, see [7]). *A graph G is nonplanar iff G contains a subgraph $\mathcal{H}$ which is contractible to K_5 or $K_{3,3}$.*

But we trivially get the following characterization of such contractions, due to Ranaivoson [43]:

LEMMA 6.2. *Let $\mathcal{H} = (V, E) \in$ GRAPH. Then:*

1. *$\mathcal{H}$ is contractible to K_5 iff V contains five mutually disjoint sets $V_1, \ldots, V_5$ such that both following conditions hold:*
   - (a) *$\mathcal{H}$ restricted to V_i, $i = 1, \ldots, 5$, is connected ;*
   - (b) *for each pair (V_i, V_j), $1 \leq i < j \leq 5$, there is an edge $\{x, y\}$ of $\mathcal{H}$ such that $x \in V_i$ and $y \in V_j$.*
2. *$\mathcal{H}$ is contractible to $K_{3,3}$ iff V contains six mutually disjoint sets $U_1, U_2, U_3, V_1, V_2, V_3$ such that both following conditions hold:*
   - (a) *each $\mathcal{H}_{U_i}$ (resp. $\mathcal{H}_{V_i}$), $i = 1, 2, 3$, is connected ;*
   - (b) *for each pair (U_i, V_j), $1 \leq i, j \leq 3$, there is an edge $\{x, y\}$ of $\mathcal{H}$ such that $x \in U_i$ and $y \in V_j$.*

Hence we get:

COROLLARY 6.1. NON-PLANAR *belongs to* vertexNLIN.

Proof. As the sets U_i, V_j involved in Lemma 6.2 are connected, they can be viewed as connected components of a spanning forest F of G. Now, assume that two of these sets, say U and V, are respectively rooted in u and v, according to F. Then, U and V are related by an edge of G iff there exist two vertices a and b in the graph such that: $F^*(a, u)$ and $F^*(b, v)$ and $E(a, b)$. This remark allows to interpret Proposition 6.2 and Lemma 6.2 by the following formula, which therefore characterizes NON-PLANAR:

$$\exists (F, F^*) \in \text{FOREST+TC}$$

$$\{\forall x : F(x) \neq x \rightarrow E(x, F(x))\}$$
$$\wedge$$
$$\left\{ \begin{array}{l} \left[\exists v_1, \ldots, v_5 : \bigwedge_{1 \leq i \leq 5} F(v_i) = v_i \ \wedge \bigwedge_{1 \leq i < j \leq 5} v_i \neq v_j \ \wedge \right. \\ \qquad \left. \bigwedge_{1 \leq i < j \leq 5} \exists a, b \ (F^*(a, v_i) \wedge E(a, b) \wedge F^*(b, v_j)) \right] \\ \qquad \vee \\ \left[\exists u_1, u_2, u_3, v_1, v_2, v_3 : \bigwedge_{1 \leq i \leq 3} (F(u_i) = u_i \wedge F(v_i) = v_i) \ \wedge \right. \\ \qquad \bigwedge_{1 \leq i < j \leq 3} (u_i \neq u_j \wedge v_i \neq v_j \wedge u_i \neq v_i) \ \wedge \\ \qquad \left. \bigwedge_{1 \leq i, j \leq 3} \exists a, b \ (F^*(a, u_i) \wedge E(a, b) \wedge F^*(b, v_j)) \right] \end{array} \right\}$$

and this formula can be written in ESO_1 by Corollary 5.2 and by the remark following it. This yields the expected conclusion.❑

7. STRUCTURAL COMPLEXITY OF vertexNLIN

In this section, we study for vertexNLIN the main questions that are of interest for any complexity class such as P, NP, NLOGSPACE, DLIN, NLIN, etc.

- *upper/lower bounds*: to prove that some natural problems do or do not belong to the concerned class;
- *structural complexity*: has the class some structural property ? For instance, is it closed under complementation ? is it strictly included in some other class ?

Surprinsingly, although most of those questions are open and seem very hard for most classical complexity classes, we are going to solve many of them for vertexNLIN. For instance, we shall prove for quite some combinatorial problems that they do not belong to vertexNLIN.

Notation. For a decision problem $\mathcal{P}$, let NON-$\mathcal{P}$ denote the complement (*i.e.* negation) problem of $\mathcal{P}$. For a complexity class $\mathcal{C}$, let co-$\mathcal{C}$ denote the class of problems whose complements belong to $\mathcal{C}$.

Here follow some new graph decision problems:

IS-TREE $= \{G \in$ GRAPH s.t. G is connex and acyclic$\}$;
IS-FOREST $= \{G \in$ GRAPH s.t. G is acyclic$\}$;
EULER $= \{G \in$ GRAPH s.t. G has some Eulerian cycle$\}$;

(Recall that a cycle in G is Eulerian if it uses each edge exactly once.)

PERF-MATCH $= \{G \in$ GRAPH s.t. G has some perfect matching$\}$;

for any $k \in \mathbb{N}^*$ define the k-colourability problem:

k-COLOUR $= \{G \in$ GRAPH s.t. G can be coloured with k colours$\}$;
COLOUR $= \{(G,k) \in$ GRAPH $\times \mathbb{N}^*$ s.t. G can be coloured with k colours$\}$;
CLIQUE $= \{(G,k) \in$ GRAPH $\times \mathbb{N}^*$ s.t. G contains a clique of size $k\}$;
PATH $= \{(G,s,t)$ s.t. G is a graph and s, t are two vertices related by a path$\}$.

First, we are going to prove that many of those problems and/or their complements do not belong to vertexNLIN. The proofs are quite easy and uniform: to prove that a property $\mathcal{P}$ does not belong to vertexNLIN, we essentially construct a family (G_n) of graphs in $\mathcal{P}$ with arbitrary large cardinality n and a set A_n of $\Theta(n^2)$ edges such that, for every $\alpha \in A_n$: $G_n \cup \{\alpha\} \notin \mathcal{P}$ (resp. $G_n \setminus \{\alpha\} \notin \mathcal{P}$). As a consequence, any σ-NRAM M that recognizes $\mathcal{P}$ has to read *all* the $\Theta(n^2)$ bits corresponding to A_n in the input adjacence matrix of G_n ; this is because if some bit $\alpha \in A_n$ was not read, then the same accepting computation of M would also accept $G_n \cup \{\alpha\} \notin \mathcal{P}$ (resp. $G_n \setminus \{\alpha\} \notin \mathcal{P}$), a contradiction.

PROPOSITION 7.1. *The following problems do not belong to* vertexNLIN*:*

IS-FOREST	IS-TREE	NON-IS-TREE	EULER
CLIQUE	NON-COLOUR	NON-CLIQUE	NON-PATH
NON-CONNEX	NON-HAMILTON	(k-COLOUR)$_{k\geq 2}$	COLOUR
PLANAR	NON-PERF-MATCH	NON-EULER	

Proof. For each problem $\mathcal{P}$ among those above quoted, we shall prove that there exists a constant $c > 0$ such that any σ-NRAM that recognizes $\mathcal{P}$ should read more than cn^2 bits of the input matrix. This will imply the result. This will be done by the construction of sets G_n and A_n as above mentioned. These constructions will be generally obtained by bringing together some of the following sets and graphs:

- $V_m = \{a_0, \ldots, a_{m-1}\}$, $V'_m = \{b_0, \ldots, b_{m-1}\}$, $V_m \cap V'_m = \emptyset$;
- the path graph $P_m = (V_m, E_m^P)$ where $E_m^P = \{\{a_{i-1}, a_i\} : \ 0 < i < m\}$ and its copy $P'_m = (V'_m, E_m^{P'})$ where $E_m^{P'} = \{\{b_{i-1}, b_i\} : \ 0 < i < m\}$;
- the clique graph $K_m = (V_m, E_m^K)$ where $E_m^K = \{\{a_i, a_j\} : \ i < j < m\}$ and its copy $K'_m = (V_m, E_m^{K'})$ where $E_m^{K'} = \{\{b_i, b_j\} : \ i < j < m\}$;
- the cycle graph $C_m = (V_m, E_m^C)$ where $E_m^C = E_m^P \cup \{a_{m-1}, a_0\}$;
- the stable graph $S_m = (V_m, \emptyset)$.

Now, let us describe the construction of families (G_n) and (A_n) for each problem involved in the statement:

Problem IS-TREE *(resp.* IS-FOREST*).*

Let G_n be the path graph P_n which is a tree (resp. a forest). Let $A_n = \{\{a_i, a_j\} : i + 1 < j < n\}$. Then, $|A_n| = \Theta(n^2)$ and for every $\alpha \in A_n$, $G_n \cup \{\alpha\}$ has a cycle and thus is not a tree (resp. a forest), as required.

Since the proofs for the other problems are very similar, we essentially give, for each problem $\mathcal{P}$, the graphs $G_n \in \mathcal{P}$ and the sets of edges A_n. We leave the details to the reader.

Problem NON-IS-TREE.

Let $n = 2m$ and G_n be the disjoint union of the path graphs P_m and P'_m. Clearly, G_n is not connected and therefore, is not a tree. Let $A_n = \{\{a_i, b_j\} : \ i, j < m\}$. Clearly, $G_n \cup \{\alpha\}$ is a tree for every $\alpha \in A_n$.

Problem EULER.

Let G_n be the cycle graph $C_n = (V_n, E_n^C)$ and $A_n = \{\{a_i, a_j\} : \ i < j < n \text{ and } \{a_i, a_j\} \notin E_n^C\}$. Clearly, $G_n \in$ EULER and $G_n \cup \{\alpha\} \notin$ EULER for every $\alpha \in A_n$.

Problem CLIQUE *(resp.* NON-COLOUR*).*

Let G_n be the n-clique graph K_n (notice that G_n is not $(n-1)$-colourable). Let $A_n = \{\{a_i, a_j\} : \ i < j < n\}$. Obviously, $G_n \setminus \{\alpha\}$ contains no n-clique (resp. is $(n-1)$-colourable) for every $\alpha \in A_n$.

Problem NON-CLIQUE.

Let G_n be the n-stable S_n which contains no 2-clique. Let $A_n = \{\{a_i, a_j\} : \ i < j < n\}$. Then $G_n \cup \{\alpha\}$ contains a 2-clique for every $\alpha \in A_n$.

Problem NON-PATH *(resp.* NON-CONNEX*).*

Let $n = 2m + 2$ and G_n be the disjoint union of the clique graphs K_m, K'_m and two new vertices s, t with the additional edges $\{s, a_i\}_{i<m}$ and $\{b_j, t\}_{j<m}$. Let

$A_n = \{\{a_i, b_j\} : i < j < m\}$. Clearly, there is no (s,t)-path in G_n (resp. G_n is not connected), but there is one in $G_n \cup \{\alpha\}$ (resp. $G_n \cup \{\alpha\}$ is connected) for every $\alpha \in A_n$.

Problem NON-HAMILTON.

Same proof as for NON-PATH, with the same graph G_n, but with the additional edge $\{s,t\}$.

Problem k-COLOUR, $k \geq 2$.

Let $n = m + k - 1$ and G_n be the disjoint union of the stable graph S_m and the clique graph K'_{k-1}, with the additional edges $\{a_i, b_j\}$, for $i < m$ and $j < k-1$. Let $A_n = \{\{a_i, a_j\} : i < j < m\}$. Clearly, G_n can be coloured with k colours, but $G_n \cup \{\alpha\}$ cannot, for every $\alpha \in A_n$.

Problem COLOUR.

This problem generalizes k-COLOUR.

Problem PLANAR.

Let $G_n = (V_n, E_n)$ be any triangulated planar graph of cardinality n. By Euler's formula (see [7] for example), we have $|E_n| = 3n - 6$, which is the maximal number of edges of planar graph of cardinality n. Let $A_n = \{\{a_i, a_j\} : i < j < n \text{ and } \{a_i, a_j\} \notin E_n\}$. Then, for every $\alpha \in A_n$, $G_n \cup \{\alpha\}$ has too many edges to be planar.

Problem NON-PERF-MATCH.

Let $n = 4m + 2$ and G_n be the disjoint union of the clique graphs K_{2m+1} and K'_{2m+1}. Let $A_n = \{\{a_i, b_j\} : i, j < m\}$. Then, G_n has no perfect-matching but for every $\alpha \in A_n$, $G_n \cup \{\alpha\}$ has one that includes α. Note that an easy variant of the above construction (left to the reader) can also prove that the restriction of the problem NON-PERF-MATCH to bipartite graphs does not belong to vertexNLIN.

Problem NON-EULER.

This proof is slightly more complicate than the previous ones. First, recall that a graph is Eulerian if and only if it is connected and each of its vertices is of even degree. Let $n = 4m+2$ and G_n be the disjoint union of K_{2m+1} and K'_{2m+1}, so that the degree of each vertex is even. Nevertheless, G_n is not connected and thus, is not Eulerian. The new idea consists in adding to G_n a fixed number of edges (not only one) so that the graph becomes connected and each vertex remains of even degree. For each ordered pair $(i,j) \in [m] \times [m]$, let

$$A_{ij} = \{\{a_{2i}, b_{2j}\}, \{a_{2i}, b_{2j+1}\}, \{a_{2i+1}, b_{2j}\}, \{a_{2i+1}, b_{2j+1}\}\}.$$

Notice that the $\Theta(n^2)$ sets of edges A_{ij} are pairwise disjoint. Let M be a σ-NRAM that reconizes the problem NON-EULER and, consequently, accepts G_n by a computation $\mathcal{C}_n$. Now, assume there exist $i, j < m$ such that $\mathcal{C}_n$ reads none of the four bits corresponding to A_{ij}. Then, the same computation $\mathcal{C}_n$ accepts the modified graph $G'_n = G_n \cup A_{ij}$. This is a contradiction since G'_n is Eulerian. So, we have proved that each computation of M that accepts G'_n must read at least one input

bit of A_{ij} for each ordered pair $(i,j) \in [m]^2$. Hence, it should read at least $m^2 = \Theta(n^2)$ distinct input bits, as claimed. This concludes the proof of Proposition 7.1.❑

By using those counterexamples, we are now in a position to answer several natural questions about the structural complexity of vertexNLIN. For example, we have proved in Subsection 5.3 that ESO_1 = vertexNLIN is not enlarged by existential quantification over transitive closure of unary functions. It easily implies that this logic is not enlarged if one allows ESO_1-formulas to refer to transitive closure of any unary function, whether it is existentially quantified or is a part of the input. In contrast, we have:

COROLLARY 7.1. *If one allows transitive closure of binary relations in* ESO_1*-formulas, then some properties that are not in* vertexNLIN *become definable.*

Proof. Problem NON-PATH, which does not belong to vertexNLIN (contrarily to PATH), is clearly defined by the formula $\neg E^*(s,t)$, where E^* denotes the transitive closure of the input binary relation E.❑

COROLLARY 7.2.

(a) vertexNLIN $\neq$ co-vertexNLIN.
(b) DLIN$\setminus$(vertexNLIN$\cup$co-vertexNLIN) $\neq \emptyset$. *Furthermore, there are problems* $\mathcal{P}$ *in* DLIN *such that every nondeterministic algorithm for* $\mathcal{P}$ *and every nondeterministic algorithm for* NON-$\mathcal{P}$ *both require* $\Omega(n^2)$ *steps.*
(c) vertexNLIN $\subsetneq$ NLIN.

Proof.

(a) For example, CONNEX and HAMILTON belong to vertexNLIN while their complements do not.
(b) Problems IS-TREE and EULER are such separating problems.
(c) Immediate consequence of (b) since DLIN $\subseteq$ NLIN.

■

Notice that Proposition 7.1 and Corollary 7.2 provide some precise informations about the comparative rôle of determinism and nondeterminism, in particular, for the resolution of specific problems such as the following:

Problem EULER (resp. IS-TREE) belongs to $\mathrm{DTIME}^{\sigma}(n^2)$. This should be compared to the fact that if EULER (resp. IS-TREE) belongs to $\mathrm{NTIME}^{\sigma}(T(n))$ or to co-$\mathrm{NTIME}^{\sigma}(T(n))$, then $T(n) = \Omega(n^2)$ should hold. That means that neither nondeterminism nor co-nondeterminism can significantly help in solving problems EULER and IS-TREE.

Problems HAMILTON, CONNEX and PERF-MATCH are very different from their complements: all three belong to $\mathrm{NTIME}^{\sigma}(n)$ but we have proved that if any of them belong to co-$\mathrm{NTIME}^{\sigma}(T(n))$, then $T(n) = \Omega(n^2)$ should hold.

The rôle of nondeterminism is essential in the fact that the graph problems we have studied belong to vertexNLIN. In the deterministic model, they all require at least $\Omega(n^2)$ steps. Natural questions arise about the deterministic restriction of vertexNLIN, that is, about vertexDLIN$^\sigma =_{\text{def}}$ DTIME$^\sigma(n)$:

- Is vertexDLIN a robust complexity class that contains significant problems ?
- Does the strict inclusion vertexDLIN $\subsetneq$ vertexNLIN $\cap$ co-vertexNLIN hold ?

Part of the answers will be provided by the analysis of two new digraph properties:

In a digraph, a *leader* is a vertex "liked" by everybody, *i.e.* such that the edge (x, s) exists for every vertex $x \neq s$. A *sink* is a leader s for which no edge (s, x) starting from s exists. (Note that a digraph may have several leaders whereas it cannot have more than one sink.) We denote:

LEADER $= \{G \in$ DIGRAPH s.t. G has a leader$\}$;
SINK $= \{G \in$ DIGRAPH s.t. G has a sink$\}$.

The next result can be found without proof in [9]:

PROPOSITION 7.2. SINK *belongs to* vertexDLIN.

Proof. (Communicated by [5].) Let us consider the following algorithm whose input is a digraph given by its 0/1 adjacency matrix $E[i, j]_{i,j<n}$.

ALGORITHM 1 (SINK).

integer sink-candidate, j ;
begin
  sink-candidate := 0; $j := 1$;
  while $j < n$ **do**/* *no vertex* $i < j$ *is a sink, except possibly sink-candidate* */
    if $E[j,$sink-candidate$] = 1$ **then** /* j *is not a sink* */
      $j := j + 1$
    else /* *sink-candidate is not a sink* */
      begin sink-candidate := j ; $j := j + 1$ **end**
  end while
    /* $j = n$ *: no vertex* $i < n$ *is a sink, except possibly sink-candidate* */
  Check whether sink-candidate is really a sink by consulting both the line and the column numbered sink-candidate ;
  if it is not **then reject** /* *the digraph has no sink* */
  else accept;
end.

Our inserted comments indicate how to prove the correctness of the algorithm, which obviously runs in time $O(n)$.❑

So, vertexDLIN contains a significant combinatorial problem. On the other hand, one can show that this complexity class, which generalizes the class DLIN in some

way, is similarly robust (see [25]). Let us now look for candidate problems to separate vertexDLIN from vertexNLIN. First, notice that graph problems expressed by first-order sentences of the form $\exists x \forall y \psi(E, x, y)$, where ψ is a quantifier-free formula with only two first-order variables x, y, trivially belong to vertexNLIN $\cap$ co-vertexNLIN. E.g., problems LEADER and SINK do, since they are expressed by the following respective sentences:

$\phi_{\text{leader}} \equiv \exists x \forall y \, : \, y \neq x \rightarrow Eyx$;

$\phi_{\text{sink}} \equiv \exists x \forall y \, : \, y \neq x \rightarrow (Eyx \wedge \neg Exy)$.

PROPOSITION 7.3.

LEADER *belongs to* (vertexNLIN $\cap$ co-vertexNLIN) $\setminus$ vertexDLIN. *More precisely, for every deterministic algorithm $\mathcal{A}$ that decides* LEADER *and for each integer n, there exists a graph $G_n(\mathcal{A})$ of cardinality n such that the computation of $\mathcal{A}$ on input $G_n(\mathcal{A})$ reads at least $n^2 - n$ input bits.*

Proof. (Essentially due to Lautemann [33].) There only remains to prove the complexity lower bound. First, note that the result of a deterministic algorithm $\mathcal{A}$ only depends on the sequence of input bits it reads during the computation. Let us fix the cardinality n. Without loss of generality, assume that the only queried bits are $E[i, j]$, for $i \neq j$, since all the diagonal bits $E[i, i]$ are zero. We can imagine those nondiagonal bits supplied by an adversary, who uses the following strategy: when queried "$E[i, j]$" ?

- if question "$E[i, j]$?" has been previously asked, she gives the same answer ;
- otherwise, *i.e.* if the question "$E[i, j]$?" is asked for the first time, she answers:

  – 1, if there is some $i' \neq i$ such that question "$E[i', j]$?" has not yet been asked ;

  – 0 otherwise, *i.e.* when all the bits of column j have been queried.

Let $\mathcal{C}_n(\mathcal{A})$ denote the computation so defined. Assume that $\mathcal{C}_n(\mathcal{A})$ stops after having queried less than the $n^2 - n$ nondiagonal bits. Also, assume that $\mathcal{C}_n(\mathcal{A})$ accepts. Then, set the (nondiagonal) non queried bits to 0. This gives an accepted input which is not in LEADER: a contradiction. So, $\mathcal{C}_n(\mathcal{A})$ rejects. Now, set the (nondiagonal) non queried bits to 1. This gives at least one column where the $n - 1$ nondiagonal bits are all 1's. Hence, the rejected input belongs to LEADER: a contradiction. This proves that $\mathcal{C}_n(\mathcal{A})$ reads all the $n^2 - n$ nondiagonal bits of the input $G_n(\mathcal{A})$ so obtained. This concludes the proof of Proposition 7.3.❑

Figure 6 summarizes the main results of this section. The problems quoted without brackets have been proved as belonging to the precise intersection on which they lie on the figure. Those between brackets are just *candidates* to belong to a given subset. Subsets that contain only candidate problems (*e.g.* co-vertexNLIN $\setminus$ NLIN) or no problem at all (*e.g.* (co-vertexNLIN $\cap$ vertexNLIN) $\setminus$ DLIN) are possibly empty. They are marked with a "?".

Remark.

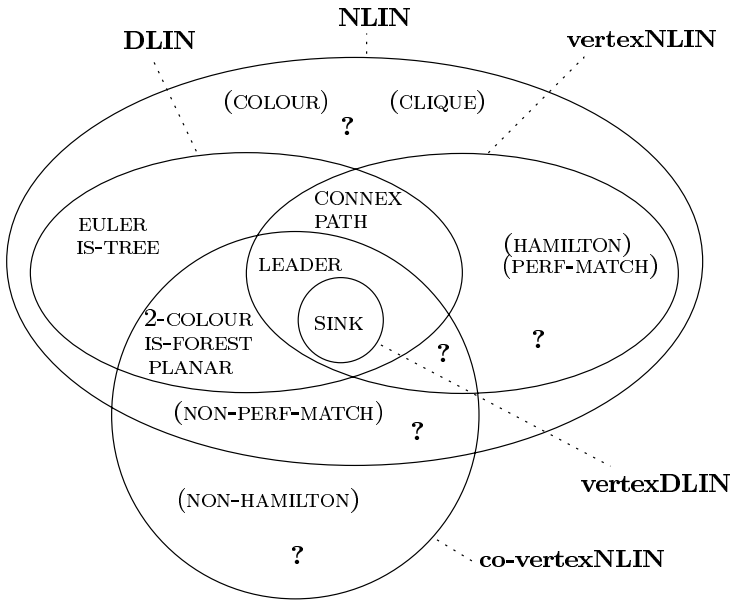

FIG. 6. Mutual inclusions between the complexity classes mentionned in this section

• NON-2-COLOUR belongs to vertexNLIN since a graph is not colourable with 2 colours iff it contains an odd length cycle.

• NON-PERF-MATCH belongs to NLIN because for a given matching M in a given graph G, one can check in deterministic linear time whether M is a maximum matching of G. For bipartite graphs, this is proved for example in [39]; for general graphs, this is proved by sophisticated technics in [35, 50] or in [4]. That yields the following NLIN algorithm:

- *guess* a non perfect matching M of G;
- *check* that M is a maximum matching of G.

8. CONCLUSIONS AND OPEN PROBLEMS

The main aim of computational complexity theory is to determine the intrinsic time (resp. space) complexity of "natural" problems. We think that logic, or more precisely, descriptive complexity, gives us tools and results to study and to better understand that complexity. This paper was initially motivated by the following two items:

An observation: most "natural" NP-complete problems belong to NLIN, *i.e.*, are recognized by NRAMs in nondeterministic linear time ; *e.g.* Grandjean [22] mentions that the 21 NP-complete problems exhibited by Karp [31] are in NLIN; for a graph problem $\mathcal{P}$, that means $\mathcal{P}$ is recognized in time $O(|V| + |E|)$, where $G = (V, E)$ is the input graph, or, equivalently (by our results [24]), $\mathcal{P}$ is defined by an ESO(arity $1, \forall 1$) formula on the domain $V \cup E$;

A question (asked by Grandjean and Lynch in *FMT open problems*, [Oberwolfach 94, problem 8] and [Luminy 95, problem 23]): investigate the class of graph properties that can be defined in $\mathrm{ESO}^{\sigma}(\mathrm{arity}\,1)$, where $\sigma = \{E\}$ and E is a binary relation symbol, *i.e.*, by existential second-order formulas with unary function and constant symbols only, interpreted in the domain of vertices.

In the present paper, we have studied in detail vertexNLIN, that is the class of σ-problems (*i.e.*, decision problems every input of which is a first-order structure of any fixed signature σ) that are recognized in nondeterministic time $O(n)$ where n is the cardinality of the domain of the input structure. So, the time $O(n)$ can be much less than the input size. For instance, the size of a graph presented by its adjacency matrix is exactly its number of bits, n^2. The conclusions of our study are the following:

$*$ vertexNLIN is a robust complexity class, as shown by its closure properties (*e.g.*, for some restricted transitive closure operators) and its various logical characterizations such as $\mathrm{ESO}^{\sigma}(\forall 1) = \mathrm{ESO}^{\sigma}(\mathrm{arity}\,1, \forall 1)$. (Note that this last characterization states that this class is (strictly?) included in $\mathrm{ESO}^{\sigma}(\mathrm{arity}\,1)$.)

$*$ vertexNLIN (and hence also $\mathrm{ESO}^{\sigma}(\mathrm{arity}\,1)$) contains many classical combinatorial problems, including CONNEX, NON-PLANAR and HAMILTON.

$*$ Although vertexNLIN appears to be a genuine complexity class, as attested by the two previous items, we have succeeded in proving, by simple arguments, several structural properties of this class; the key point is that for any specific graph problem we have studied till now, we have been able to prove that it belongs or does not belong to vertexNLIN; in particular, it is the case for all the graph problems known to belong to DLIN (deterministic linear time) we have studied; actually, we have exhibited inside DLIN (see Figure 6) a kind of microcosm of our complexity questions with a strict partition of DLIN in five nonempty subclasses:

1. the vertexDLIN problems (*e.g.* SINK);
2. some problems in (vertexNLIN $\cap$ co-vertexNLIN) $\setminus$ vertexDLIN (*e.g.* LEADER);
3. some other ones in vertexNLIN $\setminus$ co-vertexNLIN (*e.g.* CONNEX and PATH);
4. some other ones in co-vertexNLIN $\setminus$ vertexNLIN (*e.g.* 2-COLOUR and PLANAR) and finally,
5. the other ones out of vertexNLIN $\cup$ co-vertexNLIN (*e.g.* EULER and IS-TREE).

$*$ vertexNLIN appears to be the minimal nondeterministic time complexity class for graph problems, or, more generally, for decision problems of first-order structures. (Note that its deterministic counterpart, vertexDLIN, although less significant, is also of some interest.) We cannot imagine a nondeterministic process that recognizes any significant graph problem in time $o(n)$ where n is the number of vertices of the graph.

Our thesis, which may explain the difficulty to establish complexity lower bounds, is that "natural" combinatorial problems are generally of very low complexity in the following sense. While some of them, *e.g.* the contraction problem of acyclic digraphs (see [41]) are NLIN-complete, most of them either belong to vertexNLIN or to co-vertexNLIN, or, as EULER and COLOUR, can be solved in time $O(n)$ by "alternating" RAMs, namely RAMs which can perform nondeterministic (*i.e.* existential) "guess" instructions and (dual) co-nondeterministic (*i.e.* universal) "guess" instructions, with a fixed number k of alternations between existential and universal instructions[3]. Let ATIME-ALT$^\sigma(n,k)$ denote this complexity class for σ-problems and let us define the vertex-linear hierarchy[4] as the union:

$$\text{vertexLinH}^\sigma = \bigcup_k \text{ATIME-ALT}^\sigma(n,k).$$

Notice that this class can be easily characterized by second-order formulas (SO) with second-order relation or function symbols of arity ≤ 1 (resp. of arity ≤ 1 and only 1 first-order variable), namely:

$$\text{vertexLinH}^\sigma = \text{SO}^\sigma(\text{arity } 1) = \text{SO}^\sigma(\text{arity } 1, \text{var } 1).$$

(Compare those equalities with the similar characterization of LinH by J.F. Lynch [34] and N. Immerman [29, 30].)

Of course, the main significant (realistic!) time complexity is neither the nondeterministic time, nor the alternating time (here, with a fixed number of alternations), but is deterministic time. Unfortunately, we are still unable to prove any significant deterministic time lower bound on a general-purpose model of computation, for any natural problem in NP. However, we hope to have explained convincingly "where" the difficulty lies. Most of (or all ?) the "natural" problems are of nondeterministic "minimal complexity", that means they can be solved in time linear with respect to their size (or, better, with respect to their domain cardinality), provided nondeterminism (or its generalization: alternation, with a fixed number of alternations between nondeterministic phases and co-nondeterministic phases) is allowed.

Thus, the paradigm *determinism/nondeterminism* is still and ever the crucial and central point of computational complexity, but it merits to be studied in the linear context for at least as good reasons as in the traditional polynomial context. In other words, the DLIN $\stackrel{?}{=}$ NLIN question is quite significant and more precise than the P $\stackrel{?}{=}$ NP question. We hope that this paper can contribute to convince the readers of the interest of that new problematic. Specifically, our study of the class vertexNLIN, because it allows simple proofs, may help or give some indications of methods to manage the more significant DLIN $\stackrel{?}{=}$ NLIN question.

[3] for a detailed presentation of alternating machines, see for instance Papadimitriou's book [38]

[4] similar to the linear time hierarchy for words problems, denoted LinH or *rudimentary languages*. See Hajek and Pudlak's book [28].

Let us conclude this paper by giving a list of open problems:

1. Characterize the graph (resp. digraph) problems, such as SINK and LEADER, that belong to vertexNLIN $\cap$ co-vertexNLIN (a seemingly very strong condition). Are they all in PTIME ? in DLIN ?
2. Does $\mathrm{ESO}^{\sigma}(\forall 1) = \mathrm{ESO}^{\sigma}(\mathrm{arity}\,1)$ holds for $\mathrm{arity}\,(\sigma) = 1$? Notice that the equality fails for $\mathrm{arity}\,(\sigma) = 2$ since, *e.g.*, the set of complete graphs does not belong to $\mathrm{ESO}^{\sigma}(\forall 1)$. A positive answer would imply the equality, for each integer d, $\mathrm{ESO}^{\sigma}(\forall d) = \mathrm{ESO}^{\sigma}(\mathrm{arity}\,d)$ when $\mathrm{arity}\,(\sigma) \leq d$, and would yield (by the hierarchy theorem proved by Cook [8] for nondeterministic time complexity) the strictness of the arity hierarchy (an old and difficult open problem of [13]).
3. Exhibit a (nondirected) graph problem in vertexDLIN as natural as the digraph problem SINK.
4. Prove for all the classical NP-complete graph (digraph) problems, *e.g.*, KERNEL, 3-COLOUR, CUBIC-SUBGRAPH (see Garey & Johnson's book [15]) that they belong or do not belong to vertexNLIN and that each of them does not belong to co-vertexNLIN (such a systematic proof of nonbelonging would be a weakened form of the conjecture that each of them does not belong to co-NLIN).
5. Prove a conjunctive logical characterization of vertexNLIN, similar to the conjunctive characterization of NLIN given by [37]. This would provide some "natural" vertexNLIN-complete problems (via very strict reductions such as the "affine" reductions of [25]).
6. The classical graph properties we have studied, either belong to vertexNLIN, *i.e.*, can be recognized within nondeterministic time $O(n)$, or require $\Omega(n^2)$ nondeterministic time. Can we fill this gap, *i.e.*, exhibit "natural" graph problems that are recognized nondeterministically in time $o(n^2)$ and that are not in vertexNLIN ? (Of course, the "nonnatural" set of graphs that have at least $n^{3/2}$ edges fulfils those conditions.)

ACKNOWLEDGMENT

We would like to warmly thank C. Lautemann and S. Ranaivoson who have obtained in collaboration with us (references [32, 33, 42, 43]) some of the results of this paper, in particular the results mentioned in Subsections 5.2 and 5.3, in addition to Lemma 6.2, Corollary 6.1 and Proposition 7.3. This collaboration and those results were essential in the genesis of this paper. Thanks to G. Bonfante for his proof of Proposition 7.2. Thanks to Arnaud Durand, who suggested to us that a purely logical proof of Proposition 3.1 was possible. We also gratefully acknowledge the help of F. Madelaine in the correction of many English mistakes of the manuscript. Finally, we are pleased to thank the referees for their many remarks and suggestions that help us to notably improve the readability of the paper.

REFERENCES

1. A. V. Aho, J. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
2. P. Beame. A general sequential time-space tradeoff for finding unique elements. *SIAM Journal on Computing*, 20(2):270–277, 1991.
3. A. Ben-Amram and N. Jones. Computational complexity via programming languages: constant factors do matter. *Acta Informatica*, 37:83–120, 2000.
4. N. Blum. A new approach to maximum matching in general graphs. In *ICALP*, pages 586–597, 1990.

5. G. Bonfante, 2001. Personal communication.
6. A. Borodin and S. Cook. A time-space tradeoff for sorting on a general sequential model of computation. *SIAM Journal on Computing*, 11(2):287–297, 1982.
7. G. Chartrand and L. Lesniak. *Graphs and Digraphs*. Wadsworth and Brooks, California, 1986.
8. S.A. Cook. A hierarchy of nondeterministic time complexity. *Journal of Computer and System Sciences*, 7:343–353, 1973.
9. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. Mc Graw Hill, 1991.
10. N. Creignou. *Temps Linéaire et Problèmes NP-Complets*. PhD thesis, Université de Caen, France, 1993.
11. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1995.
12. R. Fagin. Generalized first–order spectra and polynomial–time recognizable sets. In R. M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings*, pages 43–73, 1974.
13. R. Fagin. A spectrum hierarchy. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:123–134, 1975.
14. L. Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60(2):337–353, 2000.
15. M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.
16. E. Grädel. On the notion of linear time computability. *International Journal of Foundations of Computer Sciences*, 1:295–307, 1990.
17. E. Grandjean. The spectra of first-order sentences and computational complexity. *SIAM Journal on Computing*, 13(2):356–373, May 1984.
18. E. Grandjean. Universal quantifiers and time complexity of random access machines. *Mathematical System Theory*, 13:171–187, 1985.
19. E. Grandjean. First-order spectra with one variable. *Journal of Computer and System Sciences*, 40:136–153, 1990.
20. E. Grandjean. A nontrivial lower bound for an NP problem on automata. *SIAM Journal on Computing*, 19:438–451, 1990.
21. E. Grandjean. Invariance properties of RAMs and linear time. *Computational Complexity*, 4:62–106, 1994.
22. E. Grandjean. Linear time algorithms and NP-complete problems. *SIAM Journal on Computing*, 23:573–597, 1994.
23. E. Grandjean. Sorting, linear time and the satisfiability problem. *Annals of Mathematics and Artificial Intelligence*, 16:183–236, 1996.
24. E. Grandjean and F. Olive. Monadic logical definability of nondeterministic linear time. *Computational Complexity*, 7:54–97, 1998.
25. E. Grandjean and T. Schwentick. Machine-independent characterizations and complete problems for deterministic linear time. *SIAM Journal on Computing*, 32(1):196–230, 2002.
26. Y. Gurevich and S. Shelah. Nearly linear time. *LNCS*, 363:108–118, 1989.
27. Y. Gurevich and S. Shelah. Nondeterministic linear-time tasks may require substantially nonlinear deterministic time in the case of sublinear work space. *Journal of the A.C.M.*, 37(3):674–687, 1990.
28. P. Hájek and P. Pudlak. *Metamathematics of First-Order Arithmetic*. Springer, Berlin, 1993.
29. N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16:760–778, August 1987.
30. N. Immerman. *Descriptive Complexity*. Graduate Texts in Computer Science. Springer, 1999.
31. R.M. Karp. Reducibility among combinatorial problems. In *Complexity of Computers Computations*, IBM Symp. 1972. Plenum Press, New York, 1972.
32. C. Lautemann, 1999. Personal communication.
33. C. Lautemann, 2001. Personal communication.
34. J.F. Lynch. Complexity classes and theory of finite models. *Mathematical System Theory*, 15:127–144, 1982.

35. S. Micali and V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *FOCS*, pages 17–27, 1980.

36. F. Olive. *Caractérisations logiques des problèmes NP : robustesse et normalisation.* PhD thesis, Université de Caen, France, 1996.

37. F. Olive. A conjunctive logical characterization of non-deterministic linear time. In LNCS, editor, *Proc. 11th Annual Conference of the EACSL (CSL'97)*, volume 1414, pages 360–372, 1998.

38. C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley Publishing Company, 1994.

39. C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, 1982.

40. W.J. Paul, N. Pippenger, E. Szemeredi, and W.T. Trotter. On determinism versus non determinism and related problems (preliminary version). In *24th Annual Symposium on Foundations of Computer Science*, pages 429–438. IEEE, 1983.

41. S. Ranaivoson. Nontrivial lower bounds for some NP-complete problems on directed graphs. In *LNCS*, volume 533, pages 318–339. CSL'90, 1991.

42. S. Ranaivoson, 1999. Personal communication.

43. S. Ranaivoson, 2000. Personal communication.

44. K. Regan. Machine models and linear time complexity. In *SIGACT News*, volume 24, Fall 1993.

45. T. Schwentick. Algebraic and logical characterizations of deterministic linear time classes. In *Proc. 12th Symposium on Theoretical Aspects of Computer Science (STACS'97)*, pages 463–474, 1997.

46. I. A. Stewart. Logical descriptions of monotone NP problems. *J. of Logic and Computation*, 4:337–357, 1994.

47. I. A. Stewart. On completeness for NP via projection translations. *Mathematical System Theory*, 27:125–157, 1994.

48. I. A. Stewart. Complete problems for monotone NP. *Theoretical Comp. Sci.*, 145:147–157, 1995.

49. L.J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Comp. Sci.*, 3:1–22, 1977.

50. V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{|V|}|E|)$ general graph maximum matching algorithm. *Combinatorica*, 14(1):71–109, 1994.