

# Polynomially orderable classes of structures

Jan Van den Bussche\*      Dirk Van Gucht  
University of Antwerp      Indiana University

## Abstract

It is well known in descriptive computational complexity theory that fixpoint logic captures polynomial time on the class of ordered finite structures. The same is true on any class of structures on which a polynomial number of orders are definable in fixpoint logic. We call a class having this property *polynomially orderable*. We investigate this property, and give examples of polynomially orderable classes of graphs and groups.

## 1 Introduction and summary

In the field of descriptive computational complexity theory [11, 6, 1, 5], the complexity of computational problems is investigated in terms of the logics that can express them. Instances of a problem are represented as finite logical structures of some fixed similarity type. A standard result of the theory [10, 17] is that a property of ordered structures is recognizable in polynomial time if and only if it is definable in fixpoint logic. The restriction to ordered structures is important; many very simple polynomial-time properties of unordered finite sets, such as even cardinality, are not expressible in fixpoint logic [3]. Also on other classes of unordered structures, such as the complete binary trees [12], fixpoint logic does not equal polynomial time.

The question thus arises how much the restriction to ordered structures can be relaxed without losing the equivalence between fixpoint logic and

---

\*Contact address: UIA - Informatica, Universiteitsplein 1, B-2610 Antwerp, Belgium.  
E-mail: vdbuss@uia.ac.be. Tel.: +32-3-8202417. Fax: +32-3-8202421.

polynomial time. One approach to this question is to investigate classes of structures that do not explicitly include an order, but on which an order is definable in fixpoint logic (e.g., [4, 16, 9]), or, more generally, to investigate classes of rigid structures (e.g., [15]). An example of such a class is the binary trees with children labeled left or right (as opposed to unordered binary trees).

In the present paper, we propose another approach and look for classes of structures on which not a single order, but a polynomial number of orders are definable in fixpoint logic. We call such classes *polynomially orderable*. A polynomially orderable class may contain non-rigid structures; however, we observe that the number of automorphisms of any structure in the class is polynomially bounded. We also show that isomorphism among structures in a polynomially orderable class is decidable in polynomial time.

Polynomially orderable classes of structures are mainly interesting because fixpoint logic is still equivalent to polynomial time on these classes, as we will show. It is therefore interesting to look for general examples of polynomially orderable classes. We give two such examples: one dealing with graph structures (the so-called “ $k$ -sparse graphs”), the other dealing with group structures (the so-called “ $k$ -generated groups”).

Apart from the problem of looking for other interesting examples of polynomially orderable classes, some other interesting open problems remain which we discuss at the end of the paper.

## 2 Preliminaries

For our purposes, a *similarity type*  $\sigma$  is a finite set of relation names, where each relation name  $R$  is given with a natural number  $\alpha(R)$  called the *arity* of  $R$ . A (*finite*) *structure*  $\mathcal{A}$  over  $\sigma$  consists of

- a finite set  $A$ , called the *domain*, together with,
- for each  $R \in \sigma$ , a relation  $R^{\mathcal{A}}$  on  $A$  of width  $\alpha(R)$ .

The cardinality of  $A$  is called the *size* of  $\mathcal{A}$ .

*Fixpoint logic* [14, 2, 3] is usually defined as an extension of first-order logic with an operator  $\mu \varphi$  expressing the least fixpoint of the mapping from relations to relations defined by the formula  $\varphi$ . For our purposes we prefer

to alternatively view fixpoint logic as a programming language built from first-order logic using the programming constructs of relational assignment, composition, and inflationary while-change loop. It is well known [8, 1] that our alternative view on fixpoint logic is equivalent to the usual one.

We omit formal definitions of syntax and semantics of fixpoint logic programs; instead, we give an example.

**Example 2.1** Let  $\sigma$  consists of a single binary relation name  $E$ . Structures over  $\sigma$  can be viewed as directed graphs. The following program tests whether the input graph is connected or not by computing the transitive closure of the edge relation  $E$  in an auxiliary binary relation  $C$ :

$C := E$ ;

**while change do**

$C := C \cup \{(x, z) \mid (\exists y)(C(x, y) \wedge C(y, z))\}$ ;

$Connected := (\forall x \neq y)(C(x, y) \vee C(y, x))$ . ■

All assignments in the body of while-loops used in fixpoint logic programs must be “inflationary”. An assignment statement is called *inflationary* if it is of the form  $X := X \cup expr$ , with  $X$  a relation name and  $expr$  a first-order expression. The while-loop in the above example is inflationary. The inflationary restriction on while-loops guarantees that fixpoint logic programs run to completion in polynomial time.

The outcome of a program  $P$  on an input structure  $\mathcal{A}$  can be conveniently thought of as a structure  $P(\mathcal{A})$  over  $\tau$ , where  $\tau$  consists of all relation names occurring on the left-hand side of an assignment statement in  $P$ . This structure  $P(\mathcal{A})$  has the same domain as  $\mathcal{A}$  and consists of the relations computed by executing  $P$  on  $\mathcal{A}$ .

Let  $\mathbf{C}$  be a class of structures over  $\sigma$ . A *query* on  $\mathbf{C}$  is a property of structures in  $\mathbf{C}$  (i.e., a subset of  $\mathbf{C}$ ) that is closed under isomorphism.<sup>1</sup> A fundamental characteristic of fixpoint logic is that every property of structures that is expressible in fixpoint logic (i.e., that can be tested for by a fixpoint logic program) is indeed closed under isomorphism.

---

<sup>1</sup>Two structures  $\mathcal{A}$  and  $\mathcal{B}$  are called *isomorphic* if there is a bijection from the domain of  $\mathcal{A}$  to the domain of  $\mathcal{B}$  such that  $f(\mathcal{A}) = \mathcal{B}$ .

### 3 Polynomially orderable classes of structures

As noted in the previous section, every query expressible in fixpoint logic is in polynomial time. By a well-known theorem due to Immerman [10] and Vardi [17], on classes of “ordered” structures, the converse implication holds as well. A structure  $\mathcal{A}$  over  $\sigma$  is called *ordered* if  $\sigma$  contains the binary relation symbol  $\leq$  and  $\leq^{\mathcal{A}}$  is a total order on the domain of  $\mathcal{A}$ .

**Theorem 3.1** ([10, 17]) *Every polynomial-time query on a class of ordered structures is expressible in fixpoint logic.*

We will assume the reader to be familiar with the standard proof of this theorem (a detailed exposition can be found in [1]). The general idea of the proof is, given a polynomial-time Turing machine  $M$  recognizing the query, to construct a fixpoint logic program working in two stages: (1) create an encoding of the input structure, in accordance with the way it is ordered; (2) simulate the computation of  $M$  on this encoding.

We will point out that Theorem 3.1 can be generalized. Thereto, we introduce the following generalization of the notion of ordered structure:

**Definition 3.1** Assume  $\sigma$  contains a  $k$ -ary relation name  $T$  and a  $k + 2$ -ary relation name  $O$ . A structure  $\mathcal{A}$  over  $\sigma$  is called *polynomially ordered* if

- $T^{\mathcal{A}} \neq \emptyset$ ; and
- for each tuple  $t = (t_1, \dots, t_k) \in T^{\mathcal{A}}$ , the relation

$$O_t^{\mathcal{A}} := \{(a, b) \mid (t_1, \dots, t_k, a, b) \in O^{\mathcal{A}}\}$$

is a total order on the domain of  $\mathcal{A}$ .

Intuitively, relation  $T$  is a set of “tags”, each tag denoting an order. The relation  $O$  then indicates the order belonging to each tag. If  $n$  is the size of  $\mathcal{A}$  then  $O^{\mathcal{A}}$  can hold at most  $n^k$  orders. Since  $k$  is fixed, this is polynomial in  $n$ ; hence the name “polynomially ordered”.

**Theorem 3.2** *Every polynomial-time query on a class of polynomially ordered structures is expressible in fixpoint logic.*

**Proof.** Let  $q$  be a polynomial-time query on a class  $\mathbf{C}$  of polynomially ordered structures over  $\sigma$ . We can make any structure  $\mathcal{A}$  over  $\sigma$  into an ordered structure over  $\sigma \cup \{\leq\}$  by extending  $\mathcal{A}$  with an arbitrary total order  $\leq_0$  over its domain. We call such an extension an “ordered version” of  $\mathcal{A}$ . Now consider the class  $\mathbf{C}_{\leq}$  of all ordered versions of all structures in  $\mathbf{C}$ . This is a class of ordered structures. Define the query  $q_{\leq}$  on  $\mathbf{C}_{\leq}$  as follows:

$$(\mathcal{A}, \leq_0) \in q_{\leq} \iff \mathcal{A} \in q.$$

This query is in polynomial time and thus, by Theorem 3.1, expressible in fixpoint logic. The standard proof of that theorem gives us a program  $Q_{\leq}$  that, on input  $(\mathcal{A}, \leq_0)$ , creates an encoding of  $\mathcal{A}$  in accordance with  $\leq_0$ , and then simulates, on this encoding, the computation of a polynomial-time Turing machine  $M_{\leq}$  recognizing  $q_{\leq}$ .

Now recall that each  $\mathcal{A} \in \mathbf{C}$  is polynomially ordered, and thus effectively represents, for each tag  $t \in T^{\mathcal{A}}$ , an ordered version of itself, namely  $(\mathcal{A}, O_t^{\mathcal{A}})$  (cf. Definition 3.1). Either all of these ordered versions will satisfy  $q_{\leq}$ , or none of them, depending on whether  $\mathcal{A}$  satisfies  $q$  or not. Hence, in order to express  $q$ , it suffices to “parallelize” the program  $Q_{\leq}$  so that, on input  $\mathcal{A}$ , it will test whether  $(\mathcal{A}, O_t^{\mathcal{A}})$  satisfies  $q_{\leq}$ , in parallel for each  $t \in T^{\mathcal{A}}$ . For each of the orders given in the structure, the parallelized program will create a separate encoding, and it will then perform, in parallel, the simulations of the different computations of  $M_{\leq}$  on these different encodings.

This parallelization can be obtained as follows. Let  $k$  be the arity of  $T$ . Increase, by  $k$ , the arity of the auxiliary relations used for representing the Turing machine tape and for doing the necessary bookkeeping during the simulation. Replace each assignment statement

$$X := \{(x_1, \dots, x_m) \mid \varphi\}$$

by

$$X := \{(t_1, \dots, t_k, x_1, \dots, x_m) \mid T(t_1, \dots, t_k) \wedge \varphi'\},$$

where  $\varphi'$  is obtained from  $\varphi$  by replacing each atomic formula  $Y(y_1, \dots, y_\ell)$  by

$$Y(t_1, \dots, t_k, y_1, \dots, y_\ell),$$

where  $Y$  is an auxiliary relation name, and by replacing each atomic formula  $x \leq y$  by

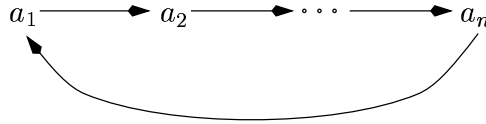
$$O(t_1, \dots, t_k, x, y).$$

This concludes the proof. ■

It is, of course, not very natural to assume that a structure given as input to a query is polynomially ordered. We are therefore interested in classes of structures that are “polynomially orderable” in fixpoint logic:

**Definition 3.2** Assume  $\sigma$  does *not* contain the relation names  $T$  and  $O$ . A class  $\mathbf{C}$  of structures over  $\sigma$  is called *polynomially orderable* if there exists a fixpoint logic program  $P$  computing a  $k$ -ary relation  $T$  and a  $k + 2$ -ary relation  $O$ , such that for each structure  $\mathcal{A}$  in  $\mathbf{C}$ , the extended structure  $(\mathcal{A}, T^{P(\mathcal{A})}, O^{P(\mathcal{A})})$  over  $\sigma \cup \{T, O\}$  is polynomially ordered.

**Example 3.1** Assume  $\sigma$  contains a binary relation name  $C$ , and let  $\mathcal{A}$  be a structure over  $\sigma$  such that  $C^{\mathcal{A}}$  is of the form



where  $\{a_1, \dots, a_n\}$  is the domain. We call  $\mathcal{A}$  a *cycle* (with respect to  $C$ ).

To each domain element  $a$  of  $\mathcal{A}$  we can associate an order  $O_a$  as follows. Let  $C_a$  be the binary relation obtained by cutting the cycle just before  $a$ , or formally:

$$C_a := \{(x, y) \mid C(x, y) \wedge y \neq a\}.$$

Then define  $O_a$  as the reflexive and transitive closure of  $C_a$ . So, we can polynomially order  $\mathcal{A}$  by defining  $T$  to be the domain and defining

$$O := \bigcup_{a \in T} (\{a\} \times O_a).$$

This can be computed in fixpoint logic.

Hence, any class of cycles is polynomially orderable. ■

As a corollary to Theorem 3.2 we obtain:

**Corollary 3.1** *Every polynomial-time query on a polynomially orderable class of structures is expressible in fixpoint logic.*

**Proof.** Let  $q$  be a polynomial-time query on a class  $\mathbf{C}$  of structures over  $\sigma$ , and assume  $\mathbf{C}$  is polynomially orderable by the fixpoint logic program  $P$ . Consider the class  $\mathbf{C}_{T,O} := \{(\mathcal{A}, T^{P(\mathcal{A})}, O^{P(\mathcal{A})}) \mid \mathcal{A} \in \mathbf{C}\}$ . This is a class of polynomially ordered structures over  $\sigma \cup \{T, O\}$ . Define the query  $q_{T,O}$  on  $\mathbf{C}_{T,O}$  as follows:

$$(\mathcal{A}, T^{P(\mathcal{A})}, O^{P(\mathcal{A})}) \in q_{T,O} \iff \mathcal{A} \in q.$$

This query is in polynomial time and thus, by Theorem 3.2, expressible by a fixpoint logic program  $Q_{T,O}$ . Hence, the composed program  $P; Q_{T,O}$  expresses  $q$ . ■

## 4 Examples

In this section, we present two rather general examples of polynomially orderable classes of structures. One deals with graphs, the other with groups.

### 4.1 Graphs

By a *graph structure* (or simply *graph*) we actually mean any structure  $G$  containing some binary relation  $E$ . This relation describes a directed graph over the domain of the structure.

**Definition 4.1** A graph  $G$  is called a *chain* if there is a natural number  $n > 0$  such that  $G$  is isomorphic to the graph with nodes  $1, \dots, n$  and set of edges  $\{(i, i + 1) \mid 1 \leq i < n\}$ . The nodes in  $G$  corresponding to 1 and  $n$  are called the *startpoint* and the *endpoint* of the chain, respectively.

**Definition 4.2** Let  $k$  be a natural number. A graph  $G$  is called *k-sparse* if  $G$  can be obtained as follows:

1. take the disjoint union of  $k$  chains;
2. add zero or more additional edges, such that each of these edges starts in an endpoint of some chain.

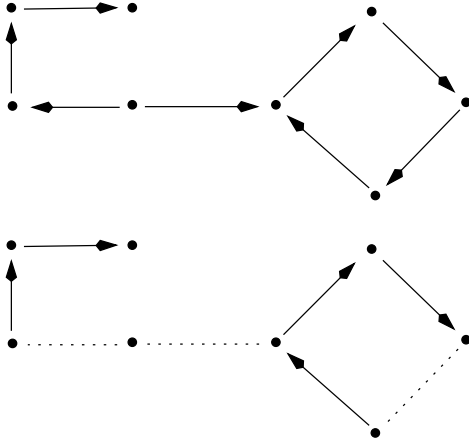


Figure 1: The graph shown on top is 3-sparse. This is illustrated at the bottom: the three chains (one of which consists of one single point) are shown in full lines and the additional edges starting in endpoints are shown in dotted lines.

---

**Example 4.1** Taking the degenerate case  $n = 1$  in Definition 4.1 we see that one single node can be viewed as a chain. Hence, a graph on  $m$  nodes is always  $m$ -sparse. However, the same graph may also be  $k$ -sparse for some  $k < m$ . For example, Figure 1 shows a 3-sparse graph which is not 2-sparse. The same figure also illustrates that every cycle graph (cf. Example 3.1) is 1-sparse. ■

**Proposition 4.1** *Let  $k$  be an arbitrary fixed natural number. Every class of  $k$ -sparse graphs is polynomially orderable.*

**Proof.** We describe a fixpoint logic program by which any  $k$ -sparse graph can be polynomially ordered. The tag relation  $T$  can be computed as follows:

$$E' := \{(x, y) \mid x \neq y \wedge E(x, y) \wedge \neg(\exists z)(E(x, z) \wedge z \neq y)\};$$

$$P := \{(x, x, x) \mid \mathbf{true}\};$$

**while change do**

$$P := P \cup \{(x, y, z) \mid (\exists u)P(x, u, y) \wedge E'(y, z) \wedge \neg(\exists v)P(x, z, v)\}$$

**od;**

$$T := \{(t_1, \dots, t_k) \mid \bigwedge_{1 \leq i < j \leq k} t_i \neq t_j \\ \wedge (\forall v) \bigvee_{i=1}^k (\exists x) P(t_i, x, v) \\ \wedge \neg(\exists v) \bigvee_{1 \leq i < j \leq k} ((\exists x) P(t_i, x, v) \wedge (\exists x) P(t_j, x, v))\}.$$

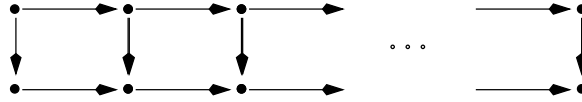
For each tag  $(t_1, \dots, t_k) \in T$  and each  $i = 1, \dots, k$ , the relation

$$P_{t_i} := \{(x, y) \mid P(t_i, x, y) \wedge x \neq y\}$$

spells out a simple path starting in  $t_i$ , such that every node on the path except possibly the last one has out-degree one. Moreover, every node in the graph lies in precisely one of the  $P_{t_i}$ . Hence, the concatenation  $P_{t_1} \dots P_{t_k}$  yields a total order of the domain. It is straightforward to write a fixpoint logic program that formally associates a total order to each tag in this manner. We thus obtain the desired  $k + 2$ -ary relation  $O$ .

Because the input graph is known to be  $k$ -sparse,  $T$  is not empty. Indeed, assume  $\gamma_1, \dots, \gamma_k$  are  $k$  disjoint chains from which the graph is built up. Then  $(t_1, \dots, t_k)$ , with  $t_i$  the startnode of  $\gamma_i$ , will be a tag in  $T$ . ■

Proposition 4.1 gives only a sufficient condition for a class of graphs to be polynomially orderable. For example, the class of graphs consisting of all those of the form



is polynomially orderable, but these graphs are not all  $k$ -sparse for some fixed  $k$ . (An order can be derived by first taking all nodes on the top and then taking all those at the bottom, each time in the order from left to right. The top nodes are characterized by their out-degree being two, the bottom ones by their out-degree being one.)

## 4.2 Groups

By a *group structure* (or simply *group*) we actually mean any structure  $G$  containing some ternary relation  $\star$  that, when interpreted as a binary operation on the domain of  $G$ , satisfies the well-known group axioms. In this paragraph we assume some familiarity with basic group theory.

**Definition 4.3** Let  $k$  be a natural number. A group is called *k-generated* if it can be generated by  $k$  of its elements.

A group of  $n$  elements is always  $n$ -generated; however, the same group can also be  $k$ -generated for  $k < n$ . Three natural examples of  $k$ -generated groups are:

- any group of order  $p^k$ , where  $p$  is prime (this follows from the First Sylow Theorem);
- any Abelian group that is a sum of at most  $k$  cyclic groups (we know that every Abelian group is a sum of cyclic groups).
- any dihedral group (symmetry group of a regular polygon) is 2-generated as it is generated by a rotation and a reflection.

**Proposition 4.2** Let  $k$  be an arbitrary fixed natural number. Any class of  $k$ -generated groups is polynomially orderable.

**Proof.** For each  $k$ -tuple  $(x_1, \dots, x_k)$  of elements, we generate all products of  $x_i$ 's and keep them in order lexicographically. If  $(x_1, \dots, x_k)$  is a generating set for the group, all elements of the group will be generated in this way. Because the group is  $k$ -generated we know such generating set exists and we obtain a tagged total order as desired. The formal fixpoint logic program computing the wanted relations  $T$  and  $O$  is shown in Figure 2. ■

## 5 Some implications

Every ordered structure is rigid; its only automorphism is the identity.<sup>2</sup> Likewise, every class of polynomially ordered structures is “polynomially” rigid, in the following sense:

**Definition 5.1** A class  $\mathbf{C}$  of structures is called *polynomially rigid* if there exists a polynomial  $p$  such that for each  $\mathcal{A}$  in  $\mathbf{C}$ , the number of automorphisms of  $\mathcal{A}$  is at most  $p(n)$ , where  $n$  is the size of  $\mathcal{A}$ .

---

<sup>2</sup>An automorphism is an isomorphism from a structure to itself.

$$\begin{aligned}
S &:= \{(\bar{x}, e) \mid (\forall y)(\star(y, e, y) \wedge \star(e, y, y))\}; \\
O &:= \{(\bar{x}, e, e) \mid S(\bar{x}, e)\}; \\
\mathbf{while\ change\ do} \\
S_1 \cup &= \{(\bar{x}, y, z) \mid S(\bar{x}, y) \wedge \star(y, x_1, z)\}; \\
S_2 \cup &= \{(\bar{x}, y, z) \mid S(\bar{x}, y) \wedge \star(y, x_2, z) \wedge \neg(\exists y') \star(y', x_1, z)\}; \\
&\vdots \\
S_k \cup &= \{(\bar{x}, y, z) \mid S(\bar{x}, y) \wedge \star(y, x_k, z) \wedge \neg(\exists y') \bigvee_{i=1}^{k-1} \star(y', x_i, z)\};
\end{aligned}$$

Let, in what follows,  $S_{new}(\bar{x}, z)$  be a shorthand for  $\neg S(\bar{x}, z) \wedge (\exists y) \bigvee_{i=1}^k S_i(\bar{x}, y, z)$ .

$$\begin{aligned}
O \cup &= \{(\bar{x}, z, z') \mid (S(\bar{x}, z) \wedge S_{new}(\bar{x}, z')) \\
&\quad \vee (S_{new}(\bar{x}, z) \wedge z' = z) \\
&\quad \vee [S_{new}(\bar{x}, z) \wedge S_{new}(\bar{x}, z') \wedge \\
&\quad \quad \bigvee_{i=1}^k ((\exists y) S_i(\bar{x}, y, z) \wedge (\exists y') S_i(\bar{x}, y', z') \\
&\quad \quad \quad \wedge (\exists y)(S_i(\bar{x}, y, z) \wedge (\forall y')(S_i(\bar{x}, y', z') \rightarrow O(\bar{x}, y, y')))] \\
&\quad \vee [S_{new}(\bar{x}, z) \wedge S_{new}(\bar{x}, z') \wedge \\
&\quad \quad \bigvee_{1 \leq i < j \leq k} ((\exists y) S_i(\bar{x}, y, z) \wedge (\exists y') S_j(\bar{x}, y', z'))]\}; \\
S \cup &= \{(\bar{x}, z) \mid (\exists y)(S_1(\bar{x}, y, z) \vee \dots \vee S_k(\bar{x}, y, z))\} \\
\mathbf{od;} \\
T &:= \{(\bar{x}) \mid (\forall y) S(\bar{x}, y)\}.
\end{aligned}$$

Figure 2: A program that polynomially orders any  $k$ -generated group. The notation  $\bar{x}$  is a shorthand for  $x_1, \dots, x_k$ , and the notation  $X \cup = \dots$  is a shorthand for  $X := X \cup \dots$ .

---

**Proposition 5.1** *Every polynomially orderable class  $\mathbf{C}$  of structures is polynomially rigid.*

**Proof.** Assume  $\mathbf{C}$  is polynomially ordered by the program  $P$ . Assume the tag relation  $T$  has arity  $k$ . Let  $\mathcal{A}$  be a structure in  $\mathbf{C}$  of size  $n$ . Let us denote the polynomially ordered structure  $(\mathcal{A}, T^{P(\mathcal{A})}, O^{P(\mathcal{A})})$  simply by  $(\mathcal{A}, T, O)$ . Recall from Definition 3.1 that  $O_t$ , for  $t$  a tag in  $T$ , denotes the order belonging to  $t$ .

Because fixpoint logic programs preserve isomorphisms, any automorphism of  $\mathcal{A}$  is also an automorphism of  $(\mathcal{A}, T, O)$ . Let  $t \in T$  be a tag, and let  $f$  and  $g$  be two automorphisms of  $\mathcal{A}$  such that  $f(t) = g(t)$ . Since  $f(O_t) = O_{f(t)}$ , and similarly for  $g$ , we have  $f(O_t) = g(O_t)$  and thus  $g^{-1}f(O_t) = O_t$ . But any permutation of a finite domain that leaves a total order on this domain invariant must be the identity. Hence  $g^{-1}f$  is the identity and thus  $g = f$ .

We have shown that the mapping  $f \mapsto f(t)$  is an injection from the set of automorphisms of  $\mathcal{A}$  into  $T$ . But  $T$  contains at most  $n^k$  tuples; hence,  $\mathcal{A}$  has at most  $n^k$  automorphisms. ■

The converse to Proposition 5.1 does not hold even if we replace “polynomially rigid” by “rigid”. Gurevich and Shelah [7] defined a class of rigid structures called “odd multipedes” and proved the theorem that no single linear order is definable on this class in fixpoint logic. However, an obvious adaptation of their proof actually shows that the class is not polynomially orderable either.

Another implication of polynomial orderability is the following:

**Proposition 5.2** *Isomorphism among structures in a polynomially orderable class  $\mathbf{C}$  is decidable in polynomial time.*

**Proof.** Assume  $\mathbf{C}$  is polynomially ordered by the program  $P$ . Let  $\mathcal{A}$  and  $\mathcal{B}$  be structures in  $\mathbf{C}$ . Denote the polynomially ordered structure  $(\mathcal{A}, T^{P(\mathcal{A})}, O^{P(\mathcal{A})})$  by  $\mathcal{A}'$  and similarly for  $\mathcal{B}'$ .

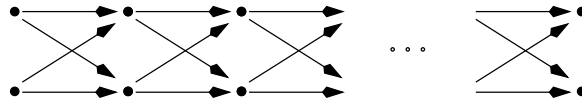
Because fixpoint logic programs preserve isomorphisms, any isomorphism  $f$  from  $\mathcal{A}$  to  $\mathcal{B}$  is also an isomorphism from  $\mathcal{A}'$  to  $\mathcal{B}'$ . Hence, if  $t \in T^{\mathcal{A}'}$  then  $f(t) \in T^{\mathcal{B}'}$ , and in this case,  $f$  is entirely determined by the pair of total orders  $(O_t^{\mathcal{A}'}, O_{f(t)}^{\mathcal{B}'})$ . Indeed, there can be only one isomorphism between two total orders.

As a result, to verify whether  $\mathcal{A}$  and  $\mathcal{B}$  are isomorphic we can use the following procedure:

1. if the sizes of  $\mathcal{A}$  and  $\mathcal{B}$  are different, they are not isomorphic.
2. otherwise, perform for each pair of tags  $(t, u) \in T^{\mathcal{A}'} \times T^{\mathcal{B}'}$ , the following:
  - (a) determine the unique isomorphism  $f$  from  $O_t^{\mathcal{A}'}$  to  $O_u^{\mathcal{B}'}$ , namely the one that maps the first element on the left to the first element on the right, the second element on the left to the second element on the right, and so on.
  - (b) verify whether  $f$  is an isomorphism from  $\mathcal{A}$  to  $\mathcal{B}$ .
3. if none of the above tests succeeds,  $\mathcal{A}$  and  $\mathcal{B}$  are not isomorphic.

This procedure clearly runs in polynomial time. ■

The converse to Proposition 5.2 does not hold. Indeed, isomorphism of graphs of bounded degree is in polynomial time [13]. However, the class of graphs of bounded degree of the following form:



is not polynomially rigid (a graph of length  $n$  has  $2^n$  automorphisms) and thus not polynomially orderable, by Proposition 5.1.

## 6 Open problems

We have seen that if a class is polynomially orderable, then fixpoint logic equals PTIME on this class. Is the converse true?

If *one single* order is definable on each structure of some class, then certainly that class is polynomially orderable. The converse is not true in general, since we have seen examples of polynomially orderable classes of non-rigid structures, while a single order can only be defined on a rigid structure. But the problem remains whether for classes of structures that are rigid, polynomial orderability implies definability of one single order. (This problem was suggested by Y. Gurevich.)

## Acknowledgment

We thank Steven Lindell for a stimulating discussion, and Anuj Dawar and Yuri Gurevich for most helpful communications.

## References

- [1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1994.
- [2] A.V. Aho and J.D. Ullman. Universality of data retrieval languages. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, pages 110–120, 1979.
- [3] A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.
- [4] A. Dawar. *Feasible Computation Through Model Theory*. PhD thesis, University of Pennsylvania, 1993.
- [5] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.
- [6] Y. Gurevich. Logic and the challenge of computer science. In E. Börger, editor, *Current Trends in Theoretical Computer Science*, pages 1–57. Computer Science Press, 1988.
- [7] Y. Gurevich and S. Shelah. On finite rigid structures. *Journal of Symbolic Logic*. To appear.
- [8] Y. Gurevich and S. Shelah. Fixed-point extensions of first-order logic. *Annals of Pure and Applied Logic*, 32:265–280, 1986.
- [9] L. Hella, P. Kolaitis, and K. Luosto. How to define a linear order on finite models. In *Proceedings 9th IEEE Symposium on Logic in Computer Science*, pages 40–49. IEEE Computer Society Press, 1994.
- [10] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

- [11] N. Immerman. Descriptive and computational complexity. In J. Hartmanis, editor, *Computational Complexity Theory*, volume 38 of *Proc. Symp. Applied Math.*, pages 75–91. American Mathematical Society, 1989.
- [12] S. Lindell. An analysis of fixed-point queries on binary trees. *Theoretical Computer Science*, 85(1):75–95, 1991.
- [13] E.M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25:42–65, 1982.
- [14] Y.N. Moschovakis. *Elementary induction on abstract structures*. North-Holland, 1974.
- [15] A. Seth. When do fixed point logics capture complexity classes? In *Proceedings 10th IEEE Symposium on Logic in Computer Science*, pages 353–363. IEEE Computer Society Press, 1995.
- [16] A. Stolboushkin. Axiomatizable classes of finite models and definability of linear order. In *Proceedings 7th IEEE Symposium on Logic in Computer Science*, pages 64–70. IEEE Computer Society Press, 1992.
- [17] M. Vardi. The complexity of relational query languages. In *14th ACM Symposium on Theory of Computing*, pages 137–146, 1982.