Finitely Representable Databases.

Stéphane Grumbach, Jianwen Su

JOURNAL OF COMPUTER AND SYSTEM SCIENCES 55, 273–298 (1997)
ARTICLE NO. SS971524

Finitely Representable Databases*

Stéphane Grumbach†

INRIA, Rocquencourt, PB 105, 78153 Le Chesnay, France

and

Jianwen Su‡

Computer Science Department, University of California, Santa Barbara, California 93106

Received December 20, 1994; revised October 14, 1996

We study infinite but finitely representable databases based on constraints, motivated by new database applications such as those involving spatio-temporal information. We introduce a general definition of finite representation and define the concept of a query as a generalization of a query over relational databases. We investigate the theory of finitely representable models and prove that it differs from both classical model theory and finite model theory. In particular, we show the failure of most of the well-known theorems of logic (compactness, completeness, etc.). An important consequence is that properties such as query satisfiability and containment are undecidable. We illustrate the use of Ehrenfeucht–Fraïssé games on the expressive power of query languages over finitely representable databases. As a case study, we focus on queries over dense order constraint databases. We consider in particular "order-generic" queries which are mappings closed under order-preserving bijections and topological queries, mappings closed under homeomorphisms. We prove that many interesting queries such as topological connectivity are not first-order definable with dense order constraints. We then consider an inflationary fixpoint query language, and prove that it captures exactly all PTIME order-generic queries. Finally, we give a rapid survey of recent results for more general contexts, such as polynomial constraints.

1. INTRODUCTION

Until recently, most data models and query languages have been designed for the modeling and the manipulation of finite collections of *simple* and *uninterpreted* data items. The classical network, hierarchical, and relational models [Ull82] are suitable for conventional database applications such as business applications. Extensions to *structured* data types, such as nested relations, complex objects, and object-oriented models, allow the modeling of more complex semantic relationships between data items [AHV95]. On the other hand, the ever increasing computational power enables the management and manipulation of wider collections of data and information from the real world. The information fundamental to new applications (e.g., scientific, geographical, digital map/image libraries, etc.) involve among other things numerical quantities, time, spatial shapes, etc. Such data are always interpreted over some arithmetic domain and are difficult to handle using the present database technology. Indeed, the representation and manipulation of interpreted data constitute a dramatic departure from a traditional database setting. Rather than having finite collections of data items (such as person names), temporal and spatial information is unbounded in nature. For instance, the map of a country consists of all infinite set of points on the earth surface, the duration of an eclipse can also be viewed as an infinite set of time instants. The manipulation of two-dimensional objects such as images or maps is among the fundamental difficulties raised by new applications. In current practice the new types of data are typically handled by coupling a database system with other systems in an ad hoc manner. There is a need for a sound database formalism to study fundamental issues related to conceptual modeling, physical data organization, query language design, and query optimization.

A new generation of data models for infinite collections of data items based on constraints is emerging in the literature since the seminal paper by Kanellakis, Kuper, and Revesz [KKR95]. We believe that two *fundamental principles* should govern the choice of data models and query languages for infinite collections of data: *infinite databases should admit finite representations*, and *a reasonable set of queries should be tractable*. The first principle states that the databases can still be stored in memory within finite space, while the second ensures that the data can be manipulated efficiently. In this paper, we consider classes of databases that satisfy these two principles and constitute a satisfactory answer to the trade-off between *classes of databases* and

* An extended abstract of this paper appeared as [GS94].

† Work supported in part by Esprit Project BRA AMUSING. E-mail: stephane.grumbach@inria.fr.

‡ Work supported in part by NSF Grants IRI-9109520 and IRI-9117094 and NASA Grant NAGW-3888. A part of work was done while visiting INRIA. E-mail: su@cs.ucsb.edu.

0022-0000/97 $25.00

complexity of queries. Our goal is to investigate the theoretical foundation of databases for infinite data and to develop technical tools for studying the data models and query languages.

There has been little investigation on infinite databases in the past. General results on the completeness of query languages for infinite recursive databases were reported by Hirst and Harel in [HH93], where two classes of recursive databases are considered, and it is shown that: (1) Quantifier-free first-order logic is complete on the class of all *recursive* databases, and (2) a version of Chandra and Harel's *QL* [CH80] is complete on *highly symmetric* databases. The first result of [HH93] indicates that the class of recursive structures is much too general for database purposes, since quantifier-free-first-order logic is already complete for this class. Therefore, many basic queries are simply not computable (projection, graph connectivity, etc.). These results reveal the importance of the trade-off between the class of structures taken as semantics and the class of admissible queries, which poses the challenging problem of exhibiting interesting classes that lie somehow between the recursive and the highly symmetric ones.

We concentrate on infinite databases that are *finitely representable* by first-order formulas in some logic language under the *context* of a first-order structure (such as rational numbers with the natural order or the real closed field). Our framework follows and generalizes the pioneer work by Kanellakis, Kuper, and Revesz [KKR95] who introduced *constraint query languages*. The novel idea there is to generalize the relations of the relational model [Cod70] by defining generalized tuples as conjunctions of constraints. For instance, the formula $x^2 + y^2 = 1 \wedge x \leqslant 0$ defines a binary generalized tuple. A generalized, or finitely representable, relation is then a finite set (disjunction) of such tuples. In the real plane for instance, it results in an infinite set of points, or tuples, over $\mathbb{R}^2$. Note that the finite representability does not imply the recursiveness of the databases. Indeed, we consider databases over uncountable domains such as the real numbers, and in the case of countable domains (such as the integers or the rationals), it depends upon the decidability of the corresponding theory.

The relational calculus over finitely representable relations constitutes a constraint query language which admits an efficient bottom-up and declarative semantics. It is suitable for expressing geometrical and topological queries, as shown by the examples. Like for the relational model, equivalent algebras were proposed [KG94, PVV94, GST94]. More powerful languages such as Datalog have also been studied [KKR95, GS95]. Practical implementation issues, such as indexing of finitely representable databases [KRVV93], have been considered, but are still among the most fundamental research issues in this field.

Different types of constraints have been considered in the literature including over dense or discrete ordered domains and involving various arithmetic operations, such as dense order equations and inequalities, linear equations and inequalities, real polynomial equations and inequalities, etc. The first fundamental observation of [KKR95] is that the relational calculus can be used as a query language as soon as the constraints are based on a decidable theory which admits the elimination of quantifiers. The evaluation of a (first-order) query is then done by quantifier elimination. The second fundamental observation of [KKR95] is that, although the decision problem of the theories underlying the constraints used (e.g., dense order without endpoints) may be high, the data complexity of the query language remains tractable. The data complexity is measured with respect to the size of the input database defined as the length of its finite representation. Low upper bounds of the data complexity for both the relational calculus and inflationary Datalog with negation over constraint databases have been obtained in [KKR95]. This shows the feasibility of this approach, which has been pursued in [Rev90, Kup93a, KG94, PVV94, KPV95, VGV95, CGK96, GS96b].

The goal of this paper is to investigate the expressive power of query languages over finitely representable databases. For that purpose, we study the underlying logic of constraint query languages and the corresponding model theory. Specifically, we study first-order logic when the semantics is restricted to *finitely representable models*. We prove that the model theory of finitely representable structures differs strongly from the classical model theory of all structures. In particular, like in the case of finite model theory (see the survey by Fagin [Fag93]), most of the classical theorems of logic including the compactness and the completeness theorems fail for finitely representable structures. This phenomenon holds for various strong restrictions on the class of structures. In particular, the model theory of recursive structures, studied in [HH94], also differs strongly from classical model theory. Finitely representable structures present also similarities with meta-finite structures [GG94]. In this later case though the structures combine an infinite mathematical context with a finite structure.

The failure of the main theorems of classical model theory suggests the need to develop new techniques for studying finitely representable structures. Powerful tools have been developed for finite model theory, such as the locality property of Gaifman [Gai81], or the 0/1 laws by Fagin [Fag76]; however, the model theory of finitely representable structures differs from finite model theory. In particular, none of the previous tools are known to be applicable in the presence of some regular mathematical structure such as an order relation for instance. Indeed, the presence of a total order implies the collapse of Gaifman's distance. Ehrenfeucht–Fraïssé games [Fra54, Ehr61] are among the few tools which apply to finitely representable structures. They are quite intuitive when the language

does not contain arithmetic operations, and is restricted to equality and an order relation. Once arithmetic is allowed, the combinatorics becomes rather tedious, although in some cases, proofs can still be carried out [ACGK94].

Complexity theoretic arguments are efficient to prove nondefinability results. It has been shown that some rather low upper bounds, in terms of Boolean circuits, of the relational algebra over finite structures carry over in the case of finitely representable structures. For example, it was first shown that the data complexity of first-order logic over dense-order constraint databases in some normal form is in AC^0 [KG94]. The result was recently extended to classes of linear constraint databases, called *k-bounded*, where the number of occurrences of the addition symbol in each constraint is bounded [GST94]. Since numerous examples of interesting queries, such as graph connectivity or region connectivity, are not in AC^0, they are not expressible in first-order logic with dense-order, or linear constraints.

We also investigate other criteria such as the invariance of the properties. There are interesting classes of queries that commute with groups of operations, such as permutations, rotations, translations, etc. This subject was explored in detail in [PVV94]. We prove that the class of first-order queries definable by dense-order constraints commute with the automorphisms of $\langle \mathbb{Q}, \leqslant \rangle$.

We illustrate these proof techniques on queries from computational geometry [Yao90], graph theory, and geographical databases. We study their definability in first-order and fixpoint logics over finitely representable databases. We show in particular the nonfirst-order definability of the connectivity of an area in the context of a dense order. The proof can be made either with an Ehrenfeucht–Fraïssé game argument, or by a complexity argument. We illustrate both techniques.

Specifically, this paper makes the following contributions. *First*, the notion of a "finitely representable" database is precisely defined in any context. *Second*, we define the notions of Boolean and non-Boolean queries and show that they generalize the classical notion of query introduced by Chandra and Harel [CH80, CH82]. It is further established that the quantifier elimination property of the theory of a first-order structure is a necessary and sufficient condition for a first-order formula to define a (non-)Boolean query. *Third*, we examine the theory of finitely representable models and prove that the compactness and the completeness theorems of classical first-order logic fail for finitely representable models under o-minimal context structures [DMM94]. We also show that satisfiability is not recursive and that if the context is decidable, validity over finitely representable models is co-r.e. Consequently, the basic properties including query satisfiability, containment, and equivalence are undecidable. *Fourth*, we study in detail order-generic queries, which are queries insensitive to order-preserving bijections, over databases defined using the rational numbers $\mathbb{Q}$ and its natural order $\leqslant$. We focus on two query languages: first-order calculus and inflationary Datalog with negation in two different context structures: the rational numbers $\mathbb{Q}$ and, the rational plane $\mathbb{Q}^2$, respectively. We establish the equivalence of both languages in the value and point based contexts. We also prove that both languages express only order-generic queries. We then concentrate on specific queries. We show that graph queries such as parity and graph connectivity, topological queries such as region connectivity, existence of a hole, Eulerian traversal, and homeomorphism are not definable in first order. However, all queries except the homeomorphism, are expressible in inflationary Datalog with negation. *Finally*, we prove that inflationary Datalog with negation expresses exactly the set of all dense-order queries computable in PTIME. This gives a complete characterization of the complexity class, extending the classical result for relational queries [Var82, Imm86].

More recently, Benedikt, Dong, Libkin, and Wong [BDLW96] used nonstandard techniques to obtain some significant generalization of nondefinability results for constraint query languages. They proved that the parity query is not expressible by first-order constraint languages with o-minimal arithmetic. These negative results were used to further establish the nondefinability of interesting topological queries by means of first-order reductions in [GS96a]. These queries include the ones considered in this paper and also the "minimal spanning tree" query that was raised as an open problem in [KKR95]. We give a brief survey of these recent results on complexity and expressive power in Section 7.

The paper is organized as follows. In Section 2, we briefly review first-order logic and formally define finitely representable databases. In Section 3, we study the model theory of finitely representable structures and investigate the differences between finitely representable model theory and both finite model theory and classical model theory. In Section 4, we formalize the notions of queries and query languages for finitely representable databases, define data complexity of queries, and give some general results on query languages. In Section 5, we examine in detail first-order query languages for dense-order queries and study the definability of interesting topological and graph queries, while in Section 6, we give a similar analysis on inflationary Datalog with negation and, in particular, present a complete characterization of PTIME queries. In Section 7, we present a brief overview of recent complexity and expressive power results on constraint query languages for more general constraints. Finally, open problems are discussed in Section 8.

2. FINITELY REPRESENTABLE DATABASES

In this section, we review the basic notions of first-order languages, structures, and theories (for more details see

[End72, EFT84]) and introduce *finitely representable models*. Throughout the whole paper, we consider only countable *first-order languages* with *equality*.

2.1. *First-Order Languages, Structures, and Theories*

A *first-order language* is a set consisting of *predicate*, *function*, and *constant* symbols. A *term* is a constant, a variable, or an expression $f(t_1, ..., t_n)$, where f is an n-ary function symbol and $t_1, ..., t_n$ are terms. *Formulas* are defined in the standard way: $P(t_1, ..., t_n)$ is an *atomic formula* if P is an n-ary predicate symbol and $t_1, ..., t_n$ are terms; $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\exists x\, \varphi$, and $\forall x\, \varphi$ are *formulas* if φ, ψ are formulas and x is a variable. (We use standard notations like "$\rightarrow$" or "$\leftrightarrow$" as shorthand.) A *sentence* is a formula with no free occurrences of variables. A formula is *quantifier free* if no quantifiers ($\forall$, $\exists$) occur in it.

A *structure* for a first-order language $\mathscr{L}$, or an $\mathscr{L}$*-structure*, is a pair $\mathscr{A} = \langle A, \rho \rangle$, where A is a nonempty set, called the *universe*, and ρ is an (interpretation) function mapping predicate, function, and constant symbols in $\mathscr{L}$ to appropriate relations, functions, and elements in A, respectively. The notion of *satisfaction* is defined in the standard way. We say equivalently that a structure $\mathscr{A}$ *satisfies* a sentence φ, that φ is *true* in $\mathscr{A}$, or that $\mathscr{A}$ is a *model* of φ, and we denote this by $\mathscr{A} \models \varphi$.

A *theory* is a set of sentences closed under *logical implication*. A theory $\mathscr{T}$ is *complete* if for every sentence φ, either $\varphi \in \mathscr{T}$ or $\neg\varphi \in \mathscr{T}$. The theory of a structure $\mathscr{A}$ is the set of sentences true in $\mathscr{A}$. It is of course complete. A theory $\mathscr{T}$ admits *quantifier elimination* if for every formula φ, there exists a quantifier-free formula ψ such that $\mathscr{T}$ logically implies the equivalence of φ and ψ. We say that a structure admits quantifier elimination when its theory does. This property is fundamental for constraint query languages.

Finally, a structure $\mathscr{A} = \langle A, \rho \rangle$ for a first-order language $\mathscr{L}$ is *recursive* if for each predicate symbol R in $\mathscr{L}$, the relation $R^{\mathscr{A}} = \rho(R)$ is recursive and for each n-ary function symbol f in $\mathscr{L}$, the relation $\{(e_1, ..., e_n, f^{\mathscr{A}}(e_1, ..., e_n)) \mid e_1, ..., e_n \in A\}$ is recursive, where $f^{\mathscr{A}} = \rho(f)$.

We consider the following two structures, which are of particular interest in the context of finitely representable databases:

- $\mathscr{Q} = \langle \mathbb{Q}, =, \leqslant, (q)_{q \in \mathbb{Q}} \rangle$, the ordered rational numbers, over the language $\mathscr{L}_{\leqslant} = \{=, \leqslant\} \cup \mathbb{Q}$.
- $\mathscr{R} = \langle \mathbb{R}, =, \leqslant, +, \times, (q)_{q \in \mathbb{Q}} \rangle$, the ordered real field, over the language $\mathscr{L}_{\times} = \{=, \leqslant, +, \times\} \cup \mathbb{Q}$.

The structure $\mathscr{Q}$ satisfies the theory of dense order without endpoints. It is complete and admits the elimination of quantifiers [CK73]. More formally,

THEOREM 2.1 [CK73]. *For each formula φ in $\mathscr{L}_{\leqslant}$, there exists a quantifier-free formula ψ in $\mathscr{L}_{\leqslant}$ such that $\mathscr{Q}$ logically implies the equivalence of φ and ψ.*

The structure $\mathscr{R}$ satisfies the theory of ordered real closed fields which is also complete and admits the elimination of quantifiers [Tar51]. More formally, Tarski proved that:

THEOREM 2.2 [Tar51]. *For each formula φ in the language $\{=, \leqslant, +, \times, 0, 1\}$, there exists a quantifier-free formula ψ in the same language such that the equivalence of φ and ψ holds in the real field.*

We assume that the associated languages contain a constant symbol for each rational number. These constants are necessary in the case of rational order constraints. They are redundant in the real case as it follows from Tarski's theorem. Nevertheless, for simplicity reasons, we assume that they are always part of the language. For comparison purposes and examples, we also consider the following structure:

- $\mathscr{N} = \langle \mathbb{N}, =, (n)_{n \in \mathbb{N}} \rangle$, the natural numbers with equality, over the language $\mathscr{L}_{=} = \{=\} \cup \mathbb{N}$.

2.2. *Finite Representability*

We assume some first-order language $\mathscr{L}$. A (*database*) *schema* σ is a finite set of relation symbols such that $\sigma \cap \mathscr{L} = \varnothing$. We always assume that the schema is disjoint from the first-order language, and we distinguish between logical predicates (such as $=$, $\leqslant$) in $\mathscr{L}$, and relations in σ.

Kanellakis, Kuper, and Revesz [KKR95] introduced the concept of a "k-ary generalized tuple," which is a conjunction of atomic formulas in $\mathscr{L}$ with k variables. For instance, in the context of real numbers the expression $(x \times x + y \times y = 1) \wedge (x \leqslant 0)$ is a binary generalized tuple in $\mathscr{L}_{\times}$ representing a half circle in the real plane. A k-ary finitely representable relation (or "generalized relation" [KKR95]) is a finite set of k-ary generalized tuples. In this framework, a tuple $[a, b]$ of the classical relational model [Cod70] is an abbreviation of the formula $(x = a \wedge y = b)$ involving only the equality symbol and constants. A finitely representable database I over σ is a collection of finitely representable relations, each associated to a relation name in the schema.

We now make these concepts precise in a general framework.

DEFINITION 2.3. Let $\mathscr{A}$ be an $\mathscr{L}$-structure with universe A, and $\varphi(x_1, ..., x_k)$ a quantifier-free formula in $\mathscr{L}$ with k distinct variables $x_1, ..., x_k$. A k-ary relation $S \subseteq A^k$ is *represented by φ over $\mathscr{A}$* if

$$\forall a_1, ..., a_k \in A, \quad \mathscr{A} \models \varphi(a_1, ..., a_k) \qquad \text{iff} \quad (a_1, ..., a_k) \in S.$$

If S is represented by φ over $\mathscr{A}$, we also call φ a *finite representation of S over $\mathscr{A}$*.

PROPOSITION 2.4. *If $\mathscr{A}$ is a recursive structure with universe A and $S \subseteq A^k$ is a k-ary relation represented by a*

quantifier-free formula φ with k distinct variables, then S is recursive.

Proof. Let $\bar{a}$ be a tuple in A^k. To determine whether $\bar{a} \in S$, it suffices to check if the quantifier-free formula $\varphi(\bar{a})$ is true in $\mathscr{A}$. If $\mathscr{A}$ is recursive. the truth of $\varphi(\bar{a})$ in $\mathscr{A}$ is decidable. ▮

EXAMPLE 2.5. Let a, b, c, d be rational numbers such that $a < c$ and $b < d$. In the rational plane, a filled rectangle with lower left and upper right corners (a, b) and (c, d), respectively, is represented by the following formula in $\mathscr{L}_{\leqslant}$ over $\mathscr{Q}$:

$$(a \leqslant x) \wedge (x \leqslant c) \wedge (b \leqslant y) \wedge (y \leqslant d).$$

In the three-dimensional real space, the top half of the sphere of distance d from the point (a, b, c) is defined by the formula $(x-a)^2 + (y-b)^2 + (z-c)^2 = d^2 \wedge 0 \leqslant x$ in $\mathscr{L}_{\times}$ (here u^2 stands for $u \times u$).

DEFINITION 2.6. Let $\mathscr{A}$ be an $\mathscr{L}$-structure with universe A. A k-ary relation $S \subseteq A^k$ is *finitely representable in* $\mathscr{L}' \subseteq \mathscr{L}$ (or $\mathscr{L}'$*-representable*) *over* $\mathscr{A}$ ($\mathscr{A}$ is omitted when it is clear from the context) if it can be represented by a quantifier-free formula in $\mathscr{L}'$ with k distinct variables. Let $\mathscr{B}$ be an expansion of $\mathscr{A}$ to the schema σ. The structure $\mathscr{B}$ is said to be $\mathscr{L}'$*-representable* (*over* $\mathscr{A}$) if for every relation symbol R in σ, $R^{\mathscr{B}}$ is $\mathscr{L}'$-representable (over $\mathscr{A}$).

Consider the languages $\mathscr{L}_{\leqslant}$ and $\mathscr{L}_{\times}$ and the structure $\mathscr{R}$. The half sphere in Example 2.5 is $\mathscr{L}_{\times}$-representable over $\mathscr{R}$. The following binary relations are $\mathscr{L}_{\leqslant}$-representable ($a, b, c, d \in \mathbb{Q}$):

- *Points* are represented by formulas of the form: "$(x = a) \wedge (y = b)$."
- *Segments* of vertical or horizontal lines, or of the diagonal line $x = y$.
- (*Filled*) *rectangles* formed by vertical or horizontal lines are represented by formulas of the form: "$(a \leqslant x) \wedge (x \leqslant b) \wedge (c \leqslant y) \wedge (y \leqslant d)$."
- (*Filled*) *triangles* formed by a vertical line, a horizontal line, and the diagonal line $(x = y)$ are represented by formulas of the form: "$(a \leqslant x) \wedge (x \leqslant y) \wedge (y \leqslant d)$."

Finitely representable relations are not arbitrary. In the case of $\mathscr{L}_{\leqslant}$, finitely representable binary relations are finite unions of points, line segments, rectangles, and triangles (see Sections 5 and 6).

DEFINITION 2.7. Let $\mathscr{A}$ be an $\mathscr{L}$-structure, $\mathscr{L}' \subseteq \mathscr{L}$, and σ a schema. An $\mathscr{L}'$*-representable* $(\mathscr{A}, \sigma)$-(*database*) *instance* is a mapping from relation symbols in σ to $\mathscr{L}'$-representable relations.

Remark. For technical convenience, we will sometimes view an instance as a mapping from relation symbols to formulas representing the intended finitely representable relation. It is clear that there are arbitrarily many formulas representing a given finitely representable relation. In practice, databases may just contain the quantifier-free formulas defining the relations. The rest is built-in. In the examples of the paper, we consider finite representability in $\mathscr{L}_{\leqslant}$ over $\mathscr{Q}$ and $\mathscr{R}$, and in $\mathscr{L}_{\times}$ over $\mathscr{R}$.

When everything is clear from the context, we will just say an $(\mathscr{A}, \sigma)$-instance or even an instance. In the following, we fix $\mathscr{A}$ and σ, and consider the class $K(\mathscr{A}, \sigma, \mathscr{L}')$ of $\mathscr{L}'$-representable $(\mathscr{A}, \sigma)$-instances. If the cardinality of the language $\mathscr{L}'$ is countable and the theory of $\mathscr{A}$ is recursive, then $K(\mathscr{A}, \sigma, \mathscr{L}')$ can be effectively enumerated. This is because to obtain an enumeration of the instances it suffices to enumerate quantifier-free formulas and check their equivalence with previously enumerated formulas. $K(\mathscr{A}, \sigma, \mathscr{L}')$ also has interesting closure properties; it is closed under finite union, intersection, and complement. This differs from finite model theory where the complement of a finite relation is generally infinite.

For each sentence φ in $\mathscr{L} \cup \sigma$, we denote by $K_{\varphi}(\mathscr{A}, \sigma, \mathscr{L}')$ the collection of $\mathscr{L}'$-representable $(\mathscr{A}, \sigma)$-instances satisfying φ.

In general, a finitely representable relation may be infinite. A finite relation, on the other hand, is representable by using only the equality predicate and constants. The converse does not hold. For example, the infinite set of all rational numbers except 0 is represented by the formula $\neg(x = 0)$. Nevertheless, the following can be verified.

PROPOSITION 2.8. *Let $\mathscr{A}$ be an $\mathscr{L}$-structure with universe A. A monadic relation R over A is finitely representable with equality and a constant symbol for each constant in R iff it is finite or co-finite.*

Proof. Let $S \subseteq A$ be a monadic relation. Clearly, if $S = \{a_1, ..., a_n\}$ is finite, S is represented by the formula $\psi = \bigvee_{1 \leqslant i \leqslant n} (x = a_i)$ and S is $\{=, a_1, ..., a_n\}$-representable. On the other hand, if S is co-finite, the complement S^c of S is finite and can be represented by a formula ϕ in some $\{=, a_1, ..., a_n\}$. Therefore S is represented by the formula $\neg\phi$ and also $\{=, a_1, ..., a_n\}$-representable.

For the other direction, without loss of generality we suppose that S is represented by a formula $\varphi = \bigvee_{1 \leqslant i \leqslant n} \psi_i$ in disjunctive normal form. Then, for each $1 \leqslant i \leqslant n$, if ψ_i is satisfiable in $\mathscr{A}$, then either (1) ψ_i is equivalent to $x = a$ for some constant $a \in A$, or (2) ψ_i is equivalent to a formula $\bigwedge_{1 \leqslant j \leqslant k} \neg(x = a_j)$ for some $a_1, ..., a_k$. Now, if (1) is true for every disjunct, S is finite. Otherwise, let S' be the set of all constants appearing in the disjuncts satisfying (2). The complement S^c of S is then contained in S'. Since S' is finite and $S^c \subseteq S'$, S is co-finite. ▮

If we impose more restrictions on the formulas representing instance relations, we can capture finiteness. Indeed, a monadic relation is finite iff it admits a representation by a formula in disjunctive normal form without negation and with equality and a constant symbol for each constant in R. This follows easily from the proof of the previous proposition.

Proposition 2.8 applies to any structure, and in particular to the structures $\mathscr{A}$ and $\mathscr{R}$. For richer representation languages with other predicates or functions than equality, finitely representable monadic relations are not necessarily finite or co-finite. However, for the languages that we consider, they satisfy some interesting properties.

The finitely representable sets are in general rather restricted. Note that discrete domains are not definable in the dense domains that we consider here. The set $\mathbb{Z}$ of integers for instance is not $\mathscr{L}_{\leqslant}$-representable over $\mathscr{Q}$ since each quantifier-free formula in $\mathscr{L}_{\leqslant}$ represents a finite union of intervals over $\mathbb{Q}$ (a consequence of Lemma 1.5.2 of [CK73]). In fact, the same holds for $\mathscr{R}$.

PROPOSITION 2.9. *Each $\mathscr{L}_{\times}$-representable monadic relation over $\mathbb{R}$ defines a finite set of intervals over $\mathbb{R}$.*

Proof. Consider an $\mathscr{L}_{\times}$-representable monadic relation represented by a quantifier-free formula in disjunctive normal form. Each disjunct is a conjunction of polynomial equations and inequalities. As a consequence of the fact that a polynomial function has a finite number of zeros (depending on its degree and coefficients), each inequality (or equation) defines a finite set of intervals. It is clear that the intersection and union of a finite number of sets of intervals only result in a finite set of intervals. The result holds since each monadic relation is a finite union of tuples, which in turn is a finite set of intervals. ∎

This property holds more generally for o-minimal signatures [DMM94], such as for instance reals with exponentiation. The results for monadic relations admit some generalizations to arbitrary arities. In particular, it can be verified for instance that the number of connected components of an $\mathscr{L}_{\times}$-representable k-ary relation is finite. The proof follows from the *cylindrical algebraic decomposition* of [Col75] (see also [Tar51, Ren92]).

3. FINITELY REPRESENTABLE MODEL THEORY

In this section, we analyze the differences between finite model theory, finitely representable model theory, and classical model theory. We shall see that the theory of finitely representable models differs from classical model theory, and we prove that, as for finite model theory [Fag93], the compactness theorem fails, and the validity over finitely representable models is in general co-r.e. (co-recursively enumerable) and not r.e. The finitely representable model theory also differs from the metafinite model theory [GG94], which is restricted to finite structures in an infinite mathematical context. Nevertheless, their model theory offers some similarities.

We prove the failure of important results of model theory in the case of finitely representable models under some context structures. We first show that the compactness theorem, and consequently the completeness theorem, fail over o-minimal context structures. Under the same assumption, we prove that the set of satisfiable sentences is not recursive. We then restrict further the assumption on the context structures, and prove that if the theory of the context structure is decidable, then the set of valid sentences is co-r.e.

We next define the *satisfiability over finitely representable models.*

DEFINITION 3.1. A sentence φ in $\mathscr{L} \cup \sigma$ is *satisfiable* (resp. *valid*) *over finitely representable models with respect to $\mathscr{A}$ and σ* if for some (resp. each) expansion $\mathscr{B}$ of $\mathscr{A}$ to σ which is $\mathscr{L}$-representable over $\mathscr{A}$, then $\mathscr{B} \models \varphi$.

We now prove the failure of the compactness theorem for o-minimal context structures. Assume that $\mathscr{L}$ is a first-order language with an order predicate. Let $\mathscr{A}$ be an $\mathscr{L}$-structure of an ordered domain A. The language $\mathscr{L}$ is o-*minimal* [DMM94] over $\mathscr{A}$ if every subset of A definable in $\mathscr{L}$ over $\mathscr{A}$ is a finite union of intervals. We say in the sequel that the context structure is o-minimal. Examples of o-minimal structures include the following $\mathscr{L}$-structures:

- $\mathscr{L} = \{=, \leqslant\} \cup \mathbb{N}$ and $\mathscr{A} = \langle \mathbb{N}, \leqslant, (n)_{n \in \mathbb{N}} \rangle$ (the structure of the ordered natural numbers).
- $\mathscr{L} = \mathscr{L}_{\leqslant}$ and $\mathscr{A} = \mathscr{Q}$ (the structure of the ordered rational numbers).
- $\mathscr{L} = \mathscr{L}_{\times}$ and $\mathscr{A} = \mathscr{R}$ (the structure of the ordered real numbers).

THEOREM 3.2. *The compactness theorem fails for finitely representable models in an o-minimal context.*

Proof. We assume a schema σ with one monadic relation R. Let τ_k be the sentence which states that (i) R contains a sequence of k nonconsecutive but increasing numbers $a_1, ..., a_k$, and (ii) for each $x \in R$, $a_1 \leqslant x$ and $x \leqslant a_k$ imply $x \in \{a_1, ..., a_k\}$. (Over a discrete domain, such as $\mathscr{N}$, the prime numbers, for instance, constitute a sequence of nonconsecutive numbers. Over a dense domain, any sequence of numbers is nonconsecutive, for instance, the natural numbers over the reals.) Define $\Sigma = \{\tau_k \mid k \geqslant 0\}$. Clearly each finite subset of Σ has a finitely representable model with respect to $\mathscr{A}$ and σ. But Σ does not admit any finitely representable model. Indeed, any model of Σ must contain an infinite sequence of disjoint closed segments. By o-minimality, it is not definable in $\mathscr{L}$, and therefore not finitely representable. ∎

The failure of the compactness theorem for restricted classes of models of interest in computer science was also investigated in [HH94, GST94]. Theorem 3.2 admits an immediate corollary, namely the failure of the completeness theorem.

COROLLARY 3.3. *The completeness theorem fails for finitely representable models in an o-minimal context.*

After the compactness theorem, another fundamental corollary of the completeness theorem fails. In classical model theory, the completeness theorem implies that the set of valid first-order sentences is r.e. It follows from Church's theorem that, as soon as the language contains some relation symbol that is not unary, the set of valid first-order sentences is not co-r.e. The contrary holds for finite model theory [Fag93]. The set of first-order sentences valid over finite structures is co-r.e. and Trakhtenbrot proved [Tra50] that it is not r.e. A similar phenomenon holds for finitely representable models. We first prove the following.

THEOREM 3.4. *Let $\mathscr{A}$ be an o-minimal $\mathscr{L}$-structure with an ordered domain A and σ be a schema containing at least one relation symbol with arity $\geqslant 2$. The set of sentences in $\mathscr{L} \cup \sigma$ satisfiable over finitely representable models with respect to $\mathscr{A}$ and σ is not recursive.*

Proof. We first consider the case where $\mathscr{A}$ has a dense order. The proof is done by a reduction from the satisfiability over finite structures. For simplicity, we assume $\sigma = \{R\}$, where R is an n-ary relation symbol for some $n \geqslant 2$. Let ϕ be a sentence in $\mathscr{L} \cup \sigma$. We construct a sentence ψ in $\mathscr{L} \cup \sigma$ such that ϕ has a finite model (($\{=\} \cup \sigma$)-structure) iff ψ has a finitely representable model (($\mathscr{L} \cup \sigma$)-structure). Let

$$\phi_i(x) = \exists x_1 \cdots x_{i-1} x_{i+1} \cdots x_n R(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$$

be the projection of R on the ith column. Then we define $\psi = \phi \wedge \bigwedge_{i=1}^{n} \alpha_i$, with

$$\alpha_i = \forall xy((\phi_i(x) \wedge \phi_i(y) \wedge x < y) \rightarrow \exists z(x < z < y \wedge \neg\phi_i(z))),$$

where $u < v$ denotes $u \leqslant v \wedge u \neq v$ and $u < w < v$ denotes $u < w \wedge w < v$.

Now suppose that ϕ has a finite model $\mathscr{M} = (M, R^{\mathscr{M}})$. Let h be a 1–1 mapping from M into A. Let $\mathscr{A}'$ be an expansion of $\mathscr{A}$ to σ such that $R^{\mathscr{A}'}$ is represented by the formula

$$\bigvee_{(a_1, ..., a_n) \in R^{\mathscr{M}}} \bigwedge_{i=1}^{n} x_i = h(a_i).$$

Clearly, $\mathscr{A}'$ is a finitely representable (in fact, only equality is used) expansion. Since the order over A is dense, $\mathscr{A}'$ satisfies α_i for each $1 \leqslant i \leqslant n$, and therefore is a model of ψ.

On the other hand, suppose that ψ has a finitely representable model $\mathscr{A}'$, which is an expansion of $\mathscr{A}$ to σ. We first prove that $R^{\mathscr{A}'}$ is actually finite. Since $R^{\mathscr{A}'}$ is a finitely representable relation, let ψ_R be a quantifier-free formula representing $R^{\mathscr{A}'}$. Let $\psi_{R_i}(x)$ be the monadic formula defining the projection $R_i = \{x \mid \phi_i|^R_{\psi_R}(x)\}$. Since $\mathscr{A}$ is o-minimal, $\psi_{R_i}(x)$ defines a finite set of intervals over A. On the other hand, $\mathscr{A}'$ satisfies α_i, and so for every pair of distinct elements $a, b \in R_i$ and $a < b$, there is some element c such that $a < c < b$ and $c \notin R_i$. It follows that $\psi_{R_i}(x)$ defines a finite set of elements. Hence, R_i is finite, and so is $R^{\mathscr{A}'}$. To complete the proof, we consider the $(\mathscr{L}_= \cup \sigma)$-structure $\mathscr{M} = (M, R^{\mathscr{M}})$, where $M = \bigcup_{1 \leqslant i \leqslant n} R_i$ and $R^{\mathscr{M}} = R^{\mathscr{A}'}$. It is clear that $\mathscr{M}$ is a finite model of ϕ.

The formula α_i over a dense order forces a finitely representable relation to be actually finite. In the case of a discrete order, the proof goes along the same line, with a new formula α_i. For instance, in the case of the discrete order over the natural numbers we could use

$$\alpha_i = \exists x(\neg\phi_i(x) \wedge \forall y(\phi_i(y) \rightarrow y < x)).$$

Then, $\mathscr{A}'$ as defined above satisfies α_i for each $1 \leqslant i \leqslant n$ and, therefore, is a model of ψ. Conversely, if ψ has a finitely representable model $\mathscr{A}'$, it is clear that $\psi_{R_i}(x)$ defines a finite set of elements. ∎

Note that in o-minimal contexts with a dense or discrete order, the finiteness of finitely representable structures is expressible. This differs from classical logic. Indeed, the finiteness of a structure in general is not first-order definable.

We next prove that if the context is decidable, then the set of valid sentences is co-r.e.

PROPOSITION 3.5. *Let $\mathscr{A}$ be an $\mathscr{L}$-structure such that the theory of $\mathscr{A}$ is recursive. Then the set of sentences in $\mathscr{L} \cup \sigma$ valid over $\mathscr{L}$-representable expansions of $\mathscr{A}$ to σ is co-r.e.*

Proof. We prove that the set of sentences in $\mathscr{L} \cup \sigma$ satisfiable over finitely representable models with respect to $\mathscr{A}$ and σ is r.e. Recall that $\mathscr{L}$ is assumed to be countable. The set of $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-instances can be effectively enumerated and each instance corresponds to an $\mathscr{L}$-representable expansion of $\mathscr{A}$ to σ. It suffices to enumerate their representations, i.e., quantifier-free formulas representing the relations in σ, and to check the equivalence of each instance with previously enumerated instances (the theory of $\mathscr{A}$ is recursive). Now, assume that $\sigma = \{R_1, ..., R_n\}$, and let $\mathscr{A}'$ be an expansion whose restriction to σ is an instance, and for each $1 \leqslant i \leqslant n$ let $\psi_i(\bar{x}_i)$ be some finite representation of R_i. We have that

$$\mathscr{A}' \models \varphi \quad \text{if and only if} \quad \mathscr{A} \models \varphi\,|^{R_1}_{\psi_1} \cdots |^{R_n}_{\psi_n},$$

where $\phi|^R_\psi$ is the formula obtained from ϕ by replacing each occurrence of R by the formula ψ. Then the result follows from the decidability of the theory of $\mathscr{A}$. ▮

The theory of dense order without endpoints and the theory of real closed fields mentioned above are recursive [CK73]. Therefore, validity over finitely representable databases over, for instance, $\mathscr{Q}$ and $\mathscr{R}$, is co-r.e. It follows from Theorem 3.4 that it is not r.e.

We next give an alternative proof that the set of first-order sentences valid over finitely representable models in the context of $\mathscr{N}$ is not r.e. based on Vaught's result [Vau60] on the validity of sentences over different classes of structures.

THEOREM 3.6. *Let σ be a schema containing at least one relation symbol with arity $\geqslant 2$. The set of sentences in $\mathscr{L}_= \cup \sigma$ valid over $\mathscr{L}_=$-representable models with respect to $\mathscr{N}$ and σ is not r.e.*

Proof. The proof is based on Vaught's result. Let $V_{\text{r.e.}}$ (respectively V_{fin}) be the set of sentences true in all r.e. (respectively finite) structures. Vaught [Vau60] showed that for each set of sentences V, if $V_{\text{r.e}} \subseteq V \subseteq V_{\text{fin}}$, then V is not r.e. Now, let $V^\sigma_{\mathscr{N}}$ be the set of sentences valid over finitely representable models with respect to $\mathscr{N}$ and σ. It is clear that each finite structure is also an $\mathscr{L}_=$-representable structure over $\mathscr{N}$, which in turn is an r.e. structure. Therefore, each sentence true in all r.e. structures is true in all $\mathscr{L}_=$-representable structures over $\mathscr{N}$; and each sentence true in all $\mathscr{L}_=$-representable structures over $\mathscr{N}$ is true in all finite structures, i.e., $V_{\text{r.e.}} \subseteq V^\sigma_{\mathscr{N}} \subseteq V_{\text{fin}}$. By Vaught's result, $V^\sigma_{\mathscr{N}}$ is not r.e. ▮

The notion of elementary equivalence is fundamental in model theory. Two structures $\mathscr{A}$ and $\mathscr{B}$ are elementary equivalent over a language $\mathscr{L}$ if they satisfy the same sentences, that is for each sentence φ in $\mathscr{L}$, $\mathscr{A} \models \varphi$ iff $\mathscr{B} \models \varphi$. This notion is uninteresting in finite model theory, since it coincides with isomorphism. The next theorem shows yet another similarity with finite model theory. The proof is in the same spirit as the one for finite models. No assumption is made on the context structure $\mathscr{A}$.

THEOREM 3.7. *Consider an $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-expansion $\mathscr{B}$. There exists a sentence $\sigma_{\mathscr{B}}$ in $\mathscr{L} \cup \sigma$ such that for each $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-expansion $\mathscr{B}'$, $\mathscr{B}'$ is isomorphic to $\mathscr{B}$ iff $\mathscr{B}' \models \sigma_{\mathscr{B}}$.*

Before proving Theorem 3.7, let us consider a simple example. Let I and J be two instances over $\mathscr{Q}$ and $\sigma = \{R\}$, where R is a monadic relation symbol. I and J are both finite sets of segments of rational numbers. The two expansions of $\mathscr{Q}$ to respectively I and J are isomorphic if there is an automorphism of $\mathscr{Q}$ which maps I to J, or equivalently, if I and J are constituted by two sequences of successive segments, respectively $s^I_1, ..., s^I_n$, and $s^J_1, ..., s^J_n$, with for all $i \leqslant n$, s^I_i and s^J_i of the same type (open interval, closed, half open-half closed, etc.). The sentence which characterizes the class of isomorphic instances states the existence of the bounds, and the definition of the intervals on these bounds.

Note that the notion becomes uninteresting over $\mathscr{R}$, since there is a unique automorphism of $\mathscr{R}$, namely the identity. We can now prove the theorem.

Proof of Theorem 3.7. Let $\mathscr{B}$ be an $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-expansion. Assume that σ has only one k-ary relation R and that $\psi_R(y_1, ..., y_k)$ is a representation of R in $\mathscr{B}$. We define the sentence $\sigma_{\mathscr{B}}$ in $\mathscr{L} \cup \sigma$ as follows. Let $a_1, ..., a_n$ be all the constants occurring in ψ_R. Let $\phi(a_1, ..., a_n)$ be the conjunction of all the closed atomic formulas in $\mathscr{L} \cup \{a_1, ..., a_n\}$ that are true in $\mathscr{A}$. Then, $\sigma_{\mathscr{B}}$ is defined by

$$\sigma_{\mathscr{B}} = \exists x_1 \cdots \exists x_n \phi(x_1, ..., x_n) \wedge \psi(x_1, ..., x_n),$$

where $\psi(x_1, ..., x_n) = \forall y_1 \cdots y_k (\psi_R|^{a_1}_{x_1} \cdots |^{a_n}_{x_n}(y_1, ..., y_k) \leftrightarrow R(y_1, ..., y_k))$ with $\varphi|^{a_i}_{x_i}$ denoting the formula φ in which each occurrence of a_i is replaced by x_i.

It follows from Proposition 4.4 in the following section, that if $\mathscr{A}$ is an $\mathscr{L}$-structure with universe A, μ an automorphism of $\mathscr{A}$, $S \subseteq A^n$ an n-ary relation over A, and φ a quantifier-free formula in $\mathscr{L}$ representing S, then, if $\mu(S)$ is finitely representable, $\mu(S)$ is represented by $\mu(\varphi)$, that is φ in which each constant a has been replaced by $\mu(a)$.

Therefore, for each $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-expansion $\mathscr{B}'$, such that $\mathscr{B}'$ is isomorphic to $\mathscr{B}$ by an isomorphism μ, then $\mathscr{B}' \models \mu(\psi_R)$, and so $\mathscr{B}' \models \sigma_{\mathscr{B}}$.

Conversely, if $\mathscr{B}'$ is an $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-expansion such that $\mathscr{B}' \models \sigma_{\mathscr{B}}$, then there exists $b_1, ..., b_n$ such that $\phi(b_1, ..., b_n)$ and $\psi(b_1, ..., b_n)$. Since $a_1, ..., a_n$ and $b_1, ..., b_n$ both satisfy ϕ, there is an automorphism μ of $\mathscr{A}$ mapping the a_i's to the b_i's, and $\mathscr{B}' = \mu(\mathscr{B})$. ▮

On the other hand, the theory of finitely representable models differs from finite model theory. Gaifman [Gai81] proved that first-order logic was local in a topological sense. This fails in presence of an order, since every neighborhood based on Gaifman's distance contains the whole domain. Techniques based on asymptotic probabilities of the truth of sentences also fail in presence of an order or arithmetic. Some results were obtained for very specific languages (see [Com88]). They do not carry over in the general case. Grädel and Gurevich presented some results in this direction for metafinite structures [GG94].

4. QUERIES AND QUERY LANGUAGES

We now define the fundamental concept of a query and introduce various subclasses of queries generalizing the classical generic relational queries, such as order-generic

or topological queries. We illustrate the definitions with examples from various fields, including geometry and graph theory. We then study when a logic, such as first-order logic, can be used as a query language, and define the data complexity of a query as a function of the input database size.

The notion of a (*relational*) *database query* was originally introduced by Chandra and Harel [CH80, CH82] as a mapping q from finite structures over a given signature σ to finite relations of a fixed arity, which is *partial recursive* and satisfies the following *consistency criterion*: If two structures over σ, I and J, are isomorphic by an isomorphism μ, then the answers $q(I)$ and $q(J)$ are isomorphic by the same isomorphism μ. This criterion was then called "genericity" in the database literature. We will speak in the sequel of *relational genericity* to distinguish it from alternative definitions.

Relational genericity was later generalized to "C-genericity" by Hull [Hu186] to allow the use of a set C of constants (which are left fixed by the isomorphisms) in a query expression. In this paper, we only consider the genericity notion in the original form, although the results can be generalized to include constants.

Genericity reflects "logical data independence" [Ull82] in database systems. Constraint databases give raise to new concepts of genericity [PVV94, BDLW96] that also reflect logical independence to various degrees. In the absence of a universal concept of genericity, we first adopt a general definition of a query which does not have any consistency criterion.

DEFINITION 4.1. Let $\mathscr{A}$ be an $\mathscr{L}$-structure, σ a schema, and R a k-ary relation symbol. A *k-ary query* is a partial recursive mapping from $(\mathscr{A}, \sigma)$-instances to $(\mathscr{A}, \{R\})$-instances.

Although genericity is not our main purpose, we consider more restricted classes of queries satisfying consistency criteria that are of interest in the present context. A natural way to generalize the concept of genericity for queries over constraint databases is to consider the set of morphisms which commute with the queries. This approach was pursued in [PVV94]. In the classical case, queries commute with automorphisms of the context structure, which is the domain with the equality predicate. So the set of morphisms which commute with the queries is the set of all permutations of the domain. We now introduce the following notion, based on the automorphisms of the context structure, which generalizes Chandra and Harel's genericity criterion to constraint databases.

DEFINITION 4.2. Let $\mathscr{A}$ be an $\mathscr{L}$-structure, σ a database schema, and R a k-ary relation symbol. A k-ary query q is *$\mathscr{L}$-generic* if for all $(\mathscr{A}, \sigma)$-instances I and J, whenever there is an automorphism μ of $\mathscr{A}$ satisfying $I = \mu(J)$, the $(\mathscr{A}, \{R\})$-instances $q(I)$ and $q(J)$ also satisfy $q(I) = \mu(q(J))$.

If the language $\mathscr{L}$ is restricted to equality, then $\{=\}$-*genericity* coincides with relational genericity on finite structures.

Remark. Note that an alternative definition of relational genericity can be found in the literature: a query q over $(\mathscr{A}, \sigma)$-instances is generic if q commutes with each permutation of A, i.e., for each $(\mathscr{A}, \sigma)$-instance I and each permutation ρ of A, $q(\rho(I)) = \rho(q(I))$. The previous definition is not equivalent to Definition 4.2 even with $\mathscr{L}$ restricted to equality. Indeed, it is easy to verify that if, for instance, both S and $\rho(S)$ are $\mathscr{L}_{\leqslant}$-representable sets over $\mathbb{Q}$ for each permutation ρ of $\mathbb{Q}$, then S has to be finite. Indeed, if S is not finite but $\mathscr{L}$-representable, it contains an interval. Now there are permutations ρ which break the interval into infinitely many "pieces," and so $\mu(S)$ is not finitely representable by Proposition 2.8.

Automorphisms of an $\mathscr{L}$-structure $\mathscr{A}$ do not have to preserve the finite representability of $(\mathscr{A}, \sigma)$-instances. We next consider the finite representation of $\mu(I)$, when μ is an automorphism which preserves the finite representability of I. We first define the effect of a morphism on a formula.

DEFINITION 4.3. Let $\mathscr{A}$ and $\mathscr{B}$ be two $\mathscr{L}$-structures with universes A and B (respectively) and μ a mapping from A to B. If φ is a formula in $\mathscr{L}$ (or any subpart of a formula, such as a term, an atom, etc.), the *image of φ under μ*, denoted by $\mu(\varphi)$, is obtained by replacing each constant c occurring in φ by $\mu(c)$.

PROPOSITION 4.4. *Let $\mathscr{A}$ be an $\mathscr{L}$-structure with universe A, μ an automorphism of $\mathscr{A}$, $S \subseteq A^n$ an n-ary relation over A, and φ a quantifier-free formula in $\mathscr{L}$ representing S. Then $\mu(S)$ is represented by $\mu(\varphi)$.*

Proof. Let $x_1, ..., x_n$ be the n distinct free variables in φ. It is sufficient to show

$$(a_1, ..., a_n) \in \mu(S) \Leftrightarrow \mathscr{A} \models \mu(\varphi)\,|_{a_1}^{x_1} \cdots |_{a_n}^{x_n}.$$

We prove the equivalence by an induction on the formula φ. For the basis case, we assume that $\varphi(x_1, ..., x_n) = p(t_1, ..., t_k)$ is an atomic formula, where p is a predicate in $\mathscr{L}$ and $t_1, ..., t_k$ are terms, we have

$$\begin{aligned}
(a_1, ..., a_n) \in \mu(S) &\Leftrightarrow (\mu^{-1}(a_1), ..., \mu^{-1}(a_n)) \in S \\
&\qquad \text{(definition of } \mu(S)\text{)} \\
&\Leftrightarrow \mathscr{A} \models p(t_1, ..., t_k)\,|_{\mu^{-1}(a_1)}^{x_1} \cdots |_{\mu^{-1}(a_n)}^{x_n} \\
&\qquad (p \text{ represents } S) \\
&\Leftrightarrow \mathscr{A} \models p(\mu(t_1), ..., \mu(t_k))\,|_{a_1}^{x_1} \cdots |_{a_n}^{x_n} \\
&\qquad \text{(since } \mu \text{ is an automorphism)} \\
&\Leftrightarrow \mathscr{A} \models \mu(p(t_1, ..., t_k))\,|_{a_1}^{x_1} \cdots |_{a_n}^{x_n} \\
&\Leftrightarrow \mathscr{A} \models \mu(\varphi)\,|_{a_1}^{x_1} \cdots |_{a_n}^{x_n}.
\end{aligned}$$

For the induction step, it is sufficient to consider the expressions $\varphi = \neg\phi$ and $\varphi = \phi \vee \psi$. Assume that S' is the relation represented by ϕ. Clearly,

$$\begin{aligned}(a_1, ..., a_n) \in \mu(S) &\Leftrightarrow (\mu^{-1}(a_1), ..., \mu^{-1}(a_n)) \in S \\ &\Leftrightarrow (\mu^{-1}(a_1), ..., \mu^{-1}(a_n)) \notin S' \\ &\Leftrightarrow \mathscr{A} \models \neg\mu(\phi)\,|_{a_1}^{x_1} \cdots |_{a_n}^{x_n} \\ &\qquad \text{(induction hypothesis)} \\ &\Leftrightarrow \mathscr{A} \models \mu(\neg\phi)\,|_{a_1}^{x_1} \cdots |_{a_n}^{x_n}.\end{aligned}$$

The proof for $\varphi = \phi \vee \psi$ is done analogously. ∎

By the above proposition, each automorphism of the structure $\mathscr{2}$ preserves the finite representability of $\mathscr{L}_{\leqslant}$-representable instances.

Suppose that $\mathscr{L}$ contains the order predicate $\leqslant$ and $\mathscr{A} = (A, \leqslant)$. A query q is *order-generic* (*$\leqslant$-generic*) if for all $(\mathscr{A}, \sigma)$-instances I and J, whenever there is an order-preserving bijection μ of $\mathscr{A}$ satisfying $I = \mu(J)$, then the $(\mathscr{A}, \{R\})$-instances $q(I)$ and $q(J)$ also satisfy $q(I) = \mu(q(J))$.

Under the above definition, each relational generic query is $\leqslant$-generic. But the converse is not true. The query that returns for instance all rational numbers between the smallest and largest numbers in the database is not relational generic. Order-generic queries over the rationals will be studied in detail in Sections 5 and 6. The next example shows that $\leqslant$-genericity is a rather restrictive notion.

EXAMPLE 4.5. There are very simple queries in the rational plane which are not $\leqslant$-generic:

- There is a line with empty intersection with the (binary) input, which separates the input, i.e., there are points in the input relation on both sides of the line.
- R defines a *grid* structure if it has a finite set of points and for each point (x, y) in R, there exist integers i_x, i_y such that $x = x_0 + i_x \Delta_x$ and $y = y_0 + i_y \Delta_y$ for some fixed x_0, y_0, Δ_x, Δ_y. The *grid query* checks if an input is a grid.
- Two disconnected areas have the same surface.
- The minimal spanning tree of a set of points.

To see that the first mapping above is not $\leqslant$-generic, consider the input binary relation R defined by

$$\begin{aligned}R(x, y) \equiv (y = 0 \wedge 0 \leqslant x \leqslant 100) \\ \bigvee (0 \leqslant y \leqslant 100 \wedge x = 0) \\ \bigvee (x = 5 \wedge y = 90).\end{aligned}$$

R contains an isolated point (5, 90) and two connected line segments (thick lines in Fig. 1a). Clearly, the isolated point cannot be separated by any straight line not intersecting the input (Fig. 1a). Consider the automorphism of $\mathscr{2}$,

$$\mu(x) = \begin{cases} x & \text{if } -\infty \leqslant x \leqslant 0 \\ 3x & \text{if } 0 \leqslant x \leqslant 10 \\ \dfrac{x + 80}{3} & \text{if } 10 \leqslant x \leqslant 40 \\ x & \text{if } 40 \leqslant x \leqslant \infty. \end{cases}$$

The image of R under μ is

$$\begin{aligned}\mu(R)(x, y) \equiv (y = 0 \wedge 0 \leqslant x \leqslant 100) \\ \bigvee (0 \leqslant y \leqslant 100 \wedge x = 0) \\ \bigvee (x = 15 \wedge y = 90)\end{aligned}$$

and the isolated point (15, 90) in the image $\mu(R)$ can now be separated from the other two segments of $\mu(R)$ by a straight line, namely $y = -x + 101$ (Fig. 1(b)).

Finally, we consider another interesting class of queries, "topological queries," which are queries invariant under topological transformations [PVV94, PSV96].

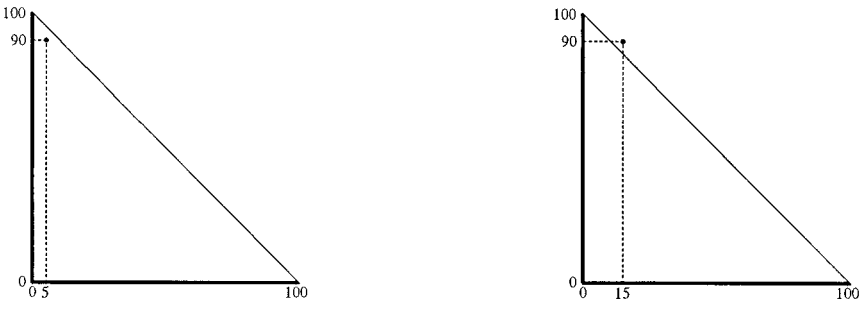

FIG. 1. *Line separation* is not $\leqslant$-generic.

DEFINITION 4.6. Let $\mathscr{A}$ be an $\mathscr{L}$-structure, σ a database schema, and R a k-ary relation symbol. A k-ary query q is *topological* if for all $(\mathscr{A}, \sigma)$-instances I and J, whenever there is a homeomorphism μ of $\mathscr{A}$ satisfying $I = \mu(J)$, then the $(\mathscr{A}, \{R\})$-instances $q(I)$ and $q(J)$ also satisfy $q(I) = \mu(q(J))$.

We next consider a point-based representation, relying on the structure $\mathscr{P}_{ab} = (\mathbb{Q}^2, \leqslant_1, \leqslant_2, a, b)$ of the rational plane $\mathbb{Q}^2$ with two ternary predicates $\leqslant_1$ and $\leqslant_2$, and two distinct constants a and b in $\mathbb{Q}^2$. The predicates $\leqslant_1$ and $\leqslant_2$ encode an order on respectively the first and the second projections of the points. More precisely, the following holds ($\overline{xy}$ denotes a point in $\mathbb{Q}^2$ with x the first projection, and y the other),

$$\begin{aligned}
&\leqslant_1(\overline{x_1y_1}, \overline{x_2y_2}, a) \quad \text{iff} \quad x_1 \leqslant x_2,\\
&\leqslant_2(\overline{x_1y_1}, \overline{x_2y_2}, a) \quad \text{iff} \quad y_1 \leqslant y_2,\\
&\leqslant_1(\overline{x_1y_1}, \overline{x_2y_2}, b) \quad \text{iff} \quad x_2 \leqslant x_1,\\
&\leqslant_2(\overline{x_1y_1}, \overline{x_2y_2}, b) \quad \text{iff} \quad y_2 \leqslant y_1,
\end{aligned}$$

where $\leqslant$ is the rational order.

Let $\mathscr{L}_{p_{ab}}$ be the first-order language of $\mathscr{P}_{ab}$. We consider queries over $(\mathscr{P}_{ab}, \sigma)$-instances (for some schema σ). We can now verify that in this point-based context each topological query is $\mathscr{L}_{p_{ab}}$-generic.

PROPOSITION 4.7. *Every topological query over $(\mathscr{P}_{ab}, \sigma)$-instances is $\mathscr{L}_{p_{ab}}$-generic.*

Proof. Let q be a topological query over $(\mathscr{P}_{ab}, \sigma)$-instances, and I, J be two $(\mathscr{P}_{ab}, \sigma)$-instances such that $I = \mu(J)$ for some automorphism μ of $\mathscr{P}_{ab}$. Since $q(I) = \rho(q(J))$ for each homeomorphism ρ of $\mathscr{P}_{ab}$ (such that $I = \rho(J)$) by the assumption that q is topological, it is sufficient to prove that μ is also a homeomorphism of $\mathscr{P}_{ab}$. To do that, we only need to verify that μ is bi-continuous, since it is clearly bijective. We prove that for each x_1, y_1 in $\mathbb{Q}$, μ is continuous at point $\overline{x_1y_1}$, and for each x_2, y_2 in $\mathbb{Q}$, μ^{-1} is continuous at point $\overline{x_2y_2}$. The proof uses standard real analysis techniques, and is thus omitted. ∎

The converse of Proposition 4.7 does not hold. For example, checking if two points are on the same vertical (or horizontal) line is $\mathscr{L}_{p_{ab}}$-generic but not topological. However, the example of $\mathscr{P}_{ab}$ raised an interesting question of whether an appropriate context structure can be found such that the class of topological queries coincides with the class of generic queries. This question is also relevant for other genericity notions considered in [PVV94].

4.1. *Query Languages*

Let $\mathscr{L}$ be a first-order language and σ a schema disjoint from $\mathscr{L}$. We shall consider $\mathscr{L}$ as a query language in the following sense: each formula φ in $\mathscr{L} \cup \sigma$ with free variables $x_1, ..., x_n$ ($n \geqslant 0$) defines a *query* (*expression*) *over* σ: $\{(x_1, ..., x_n) \mid \varphi\}$.

Suppose $q = \{(x_1, ..., x_n) \mid \varphi\}$ is a query over σ and I is an $(\mathscr{A}, \sigma)$-instance. Since for each $R \in \sigma$, $I(R)$ can be defined by a quantifier-free formula in $\mathscr{L}$, and φ is a formula in $\mathscr{L} \cup \sigma$, we can replace in φ each occurrence of the relation symbol $R \in \sigma$ by a formula defining $I(R)$. The resulting formula φ' is a formula in $\mathscr{L}$. The *answer of the query q on I*, denoted by $q(I)$, is defined by a quantifier-free formula ψ in $\mathscr{L}$ such that φ' and ψ are logically equivalent in $\mathscr{A}$. There are two fundamental issues that need to be resolved before considering $\mathscr{L}$ as a meaningful query language:

Q1. For an arbitrary formula in $\mathscr{L}$, does an equivalent quantifier-free formula always exist? If so, can it be effectively constructed?

Q2. Does every formula define an ($\mathscr{L}$-)generic query with respect to Definition 4.2?

Question Q1 is related to the quantifier elimination property of first-order theories. For the contexts $\mathscr{Q}$ and $\mathscr{R}$ considered here, their theories admit quantifier elimination and equivalent quantifier-free formulas can be effectively obtained [CK73, Tar51]. It is not always the case as shown in the following example.

EXAMPLE 4.8. The theory of $\langle \mathbb{N}, \leqslant, + \rangle$ (a.k.a. Presburger arithmetic) does not admit quantifier elimination. For example, the set of even numbers defined by the formula $\phi(x) \equiv \exists z(x = z + z)$ is not definable by a quantifier-free formula [End72].

It should be noted that the functions allowed in the language play a fundamental role in the quantifier elimination procedure. In particular, Van den Dries showed that the extension of the theory of real closed fields to a language with an exponential function, $\{=, \leqslant, +, \times, e^x, 0, 1\}$, does not admit a quantifier elimination procedure [Dri82].

For Question Q2, we first consider the case of Boolean query expressions, and show that they define generic queries.

PROPOSITION 4.9. *Let $\mathscr{A}$ be an $\mathscr{L}$-structure, and σ a database schema. For each constant-free first-order sentence φ in $\mathscr{L} \cup \sigma$, $q = \{( ) \mid \varphi\}$ defines a Boolean generic query over $\mathscr{A}$.*

Proof. We assume for simplicity that σ contains a single (k-ary) relation symbol R. The proof is accomplished by an induction on the structure of the formulas representing the $(\mathscr{A}, \sigma)$-instances. Let I and J be two $(\mathscr{A}, \sigma)$-instances, such that there is an automorphism μ of $\mathscr{A}$ satisfying $I = \mu(J)$. It is sufficient to prove that the answers $q(I)$ and $q(J)$ satisfy $q(I) = \mu(q(J))$, i.e., $q(I)$ and $q(J)$ have the same truth value. Let ψ be a representation of R in J. Then, by Proposition 4.4, a representation of R in I is given by $\mu(\psi)$. Since φ

is the formula for the query q, $q(I)$ is defined by $\varphi|^R_\psi$, while $q(J)$ is defined by $\varphi|^R_{\mu(\psi)}$. Since φ is constant free, $\varphi|^R_{\mu(\psi)} = \mu(\varphi|^R_\psi)$. ∎

Note that the above proposition holds independently of the fact that the underlying structure admits the quantifier elimination property. More generally, we can prove the following for non-Boolean queries.

PROPOSITION 4.10. *Let $\mathscr{A}$ be an $\mathscr{L}$-structure that admits quantifier elimination and φ be a constant-free formula in $\mathscr{L} \cup \sigma$ with free variables $x_1, ..., x_n$, then $\{(x_1, ..., x_n) \mid \varphi\}$ is a generic query.*

The proof is similar to that of Proposition 4.9 and thus is omitted. Although not explored in this paper, the above two propositions can be generalized to formulas (sentences) φ containing constants by allowing only automorphisms that are the identity on the constants in φ. Such an extension naturally generalizes the concept of C-genericity [Hu186].

4.2. *Data Complexity*

The "data complexity" of queries is defined based on computational devices and standard encodings of the input/output. Intuitively, the time (space) data complexity of a query is the time (space) needed in evaluating the query with respect to the size of the input database instance. Since the focus of the next two sections of the paper is on dense-order constraint (i.e., $\mathscr{L}_\leqslant$-representable) databases and queries, we formally introduce the notion of data complexity for dense order queries. For other kinds of constraints, it suffices to extend the tape alphabet to include the relation and function symbols in $\mathscr{L}$ and to define a standard binary encoding of the constants.

We first introduce the *standard encoding* of a database instance, which is obtained by encoding the quantifier-free formulas representing it. Formulas are encoded in the alphabet:

$$\{\#, [\ ,\ ], (\ ,\ ),\ \wedge,\ \vee,\ \neg,\ =,\ \leqslant, 0, 1,\ \div, \times\} \cup \sigma,$$

where σ is a signature. Natural numbers are encoded in binary notation, and rationals are encoded as pairs (fractions) of natural numbers ($n \div m$). Variables are encoded by $\times$ followed by a binary number. In the following we first illustrate the encoding of a relation defined in $\mathscr{L}_\leqslant$ with an example and then describe the encoding in the general case for relations and database instances.

EXAMPLE 4.11. Suppose that R is a binary relation defined by

$$R(x, y) \equiv ((2.75 \leqslant x) \wedge (x \leqslant 7) \wedge (x > y)) \vee (x \leqslant y).$$

The encoding of R is done as follows:

$$\mathrm{R}[\leqslant(\div(1011,100), \mathrm{x0}) \wedge \leqslant(\mathrm{x0}, 111)$$
$$\wedge\ \neg(\leqslant(\mathrm{x0}, \mathrm{x1})] \vee [\leqslant(\mathrm{x0}, \mathrm{x1})]\#$$

Let *enc* denote the encoding function. For atomic formulas of the form "$p(t_1, ..., t_k)$," the encoding is defined as:

$$\mathrm{enc}(p(t_1, ..., t_k)) = p(\mathrm{enc}(t_1), ..., \mathrm{enc}(t_k)).$$

The encoding of terms is made in the same way as for atomic formulas. If $\varphi \equiv \psi_1 \wedge \cdots \wedge \psi_\ell$ is a conjunction of atomic formulas, then

$$\mathrm{enc}(\varphi) = \mathrm{enc}(\psi_1) \wedge \cdots \wedge \mathrm{enc}(\psi_\ell).$$

Suppose I is a database and a k-ary relation $I(R)$ is represented by a formula in *disjunctive normal form*: $R(x_1, ..., x_k) \equiv \phi_1 \vee \cdots \vee \psi_\ell$, where each ϕ_i is a conjunction of atomic formulas. Then

$$\mathrm{enc}(I(R)) = R[\mathrm{enc}(\phi_1)] \vee \cdots \vee [\mathrm{enc}(\phi_\ell)]\#.$$

Finally, Let σ be a signature with n relation symbols and $R_1, ..., R_n$ be an enumeration of them. Then the encoding of an instance I is defined as

$$\mathrm{enc}(I) = \mathrm{enc}(I(R_1))\# \cdots \#\mathrm{enc}(I(R_n))\#\#.$$

The *size* of a database instance is defined as the size of the encoding of its *shortest* representation (assuming an enumeration of the relations of the database schema).

Let $\mathscr{C}$ be a complexity class, such as PTIME, PSPACE, etc. A query q is in $\mathscr{C}$ if there is a Turing machine[1] which on input a standard encoding of the input database instance produces a standard encoding of the output database instance in a bounded amount of resources allowed by $\mathscr{C}$ in the size of the input.

4.3. *Properties of Query Languages*

We consider some general decision problems concerning satisfiability, containment, and equivalence of queries on finitely representable databases.

Let $\mathscr{A}$ be an $\mathscr{L}$-structure and σ a signature. A query q is said to be *contained in* another query q' *with respect to $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-instances*, denoted by $q \sqsubseteq_{\mathscr{A},\sigma} q'$, if for each $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-instance I, $q(I) \subseteq q'(I)$. Similarly, q, q' are *equivalent with respect to $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-instances*, denoted $q \equiv_{\mathscr{A},\sigma} q'$, if $q \sqsubseteq_{\mathscr{A},\sigma} q'$ and $q' \sqsubseteq_{\mathscr{A},\sigma} q$; a Boolean query q is *satisfiable with respect to $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-instances*, if for some $\mathscr{L}$-representable $(\mathscr{A}, \sigma)$-instance I, $q(I)$ holds.

[1] Or any other device such as a family of Boolean circuits in the case of parallel complexity classes.

THEOREM 4.12. *Let σ be a schema containing at least one relation symbol with arity $\geqslant 2$. Query satisfiability, equivalence, and containment are undecidable in each of the following cases*:

1. $\mathscr{L}_{=} \cup \sigma$ *queries over* $(\mathscr{N}, \sigma)$*-instances*;
2. $\mathscr{L}_{\leqslant} \cup \sigma$ *queries over* $(\mathscr{Q}, \sigma)$*-instances*; *and*
3. $\mathscr{L}_{\times} \cup \sigma$ *queries over* $(\mathscr{R}, \sigma)$*-instances.*

In addition, it is also undecidable whether a query is $\mathscr{L}_{=}$-generic in the cases (2) *and* (3) *and whether an $\mathscr{L}_{\times} \cup \sigma$ query is $\leqslant$-generic.*

Proof. It is sufficient to prove the results for Boolean queries. Recall that each Boolean query is a sentence and defines a collection of structures. By Theorem 3.4, query satisfiability is undecidable in o-minimal contexts. This implies directly that equivalence and containment are also undecidable (consider the other query being the empty query). The undecidability of relational genericity and $\leqslant$-genericity can be easily established by reductions from the satisfiability problems, where the reductions are simply to pad an input query with a nonrelational generic or non-order generic query. ▮

In particular, Theorem 4.12 indicates that the properties listed above are undecidable in the first-order constraint query language of [KKR95] over $\mathscr{R}$.

In the next two sections, we restrict our focus to dense-order constraints databases and analyze the data complexity and the expressive power of the query languages, first-order in the following section, and Datalog in Section 6.

5. FIRST-ORDER ORDER-GENERIC QUERIES

We consider in this section both the "value" and the point-based modelizations. We prove in particular that the value and the point-based first-order languages are equivalent and that they both express order-generic queries. Many interesting topological queries such as "region connectivity," "existence of a hole," "Eulerian traversal," "k-dimensional homeomorphisms," and well-known graph queries (parity and transitive closure) are shown to be not expressible in first order. In terms of the proof techniques, the nondefinability results are mostly shown by a combination of data complexity results and AC^0-reductions. (Alternative first-order reductions were used in [GS96a].) We also illustrate the use of Ehrenfeucht–Fraïssé games in proving the language equivalence and nondefinability results.

Recall that $\mathscr{Q}$ is a model of the theory of dense orders without endpoints. The theory is decidable and admits quantifier elimination procedures [CK73]. Let $\mathscr{R}_{\leqslant} = (\mathbb{R}, \leqslant)$ be the structure of the real dense order. Since it is also a model of the theory of dense orders without endpoints, our results (apart from those depending explicitly from the cardinality) concerning query languages and databases using the rational dense order extend naturally to that using the real dense order.

The first-order query language for the value-based context, denoted as FO, is simply the first-order language $\mathscr{L}_{\leqslant} \cup \sigma$ (where σ is the underlying schema). Clearly each formula in $\mathscr{L}_{\leqslant} \cup \sigma$ defines an $\leqslant$-generic query over $\mathscr{L}_{\leqslant}$-representable instances since first-order mappings commute with automorphisms of $\mathscr{Q}$ (Proposition 4.10). In particular, each first-order Boolean query (sentence) defines a recursive collection of database instances over σ closed under automorphisms of $\mathscr{Q}$.

We consider a point based context structure slightly different from the structure $\mathscr{P}_{ab}$ (Proposition 4.7). Specifically, let $\mathscr{L}_p = \{\leqslant_1, \leqslant_2, \preccurlyeq\} \cup \mathbb{Q}^2$ be a first-order language with binary predicates and $\mathscr{P} = \langle \mathbb{Q}^2, \leqslant_1, \leqslant_2, \preccurlyeq, (p)_{p \in \mathbb{Q}^2} \rangle$ an $\mathscr{L}_p$-structure. Here $\leqslant_1$, $\leqslant_2$ are the orders on respectively the first and second projections of the points, and $\preccurlyeq$ is defined as $\overline{x_1 y_1} \preccurlyeq \overline{x_2 y_2}$ iff $x_1 \leqslant y_2$. The first-order query language in the point-based context, FO_p, is the language $\mathscr{L}_p \cup \sigma$ for some schema σ.

The *extension* of a mapping $\mu: \mathbb{Q} \to \mathbb{Q}$ to $\mathbb{Q}^2$ is defined as $\mu'(x, y) = (\mu(x), \mu(y))$ and the *restriction* of a mapping $\mu: \mathbb{Q}^2 \to \mathbb{Q}^2$ to $\mathbb{Q}$ is defined as $\mu'(x) = y$ if $\mu(x, 0) = (y, z)$ for some z. In the following we use μ to denote both a mapping and its extension/restriction.

LEMMA 5.1. *A mapping μ is an automorphism of $\mathscr{Q}$ iff it is an automorphism of $\mathscr{P}$.*

Proof. The "only if" direction is trivial. For the "if" part, let μ be an automorphism of $\mathscr{P}$. We show that μ', the restriction of μ, preserves the order. Suppose a, b are two arbitrary numbers in $\mathbb{Q}$. Then, $a \leqslant b$ iff $(a, 0) \leqslant_1 (b, 0)$ iff $\mu(a, 0) \leqslant_1 \mu(b, 0)$, the latter holds because μ is an automorphism of $\mathscr{P}$. Now $\mu(a, 0) \leqslant_1 \mu(b, 0)$ iff $\mu(a) \leqslant \mu(b)$ by definition of the restriction of μ. ▮

Let σ_p and σ be two schemas such that there is a bijection ρ from σ_p to σ satisfying the following: for each $R \in \sigma_p$, the arity of $\rho(R)$ is 2 times the arity of R. We can convert $(\mathscr{Q}, \sigma)$-instances and $(\mathscr{P}, \sigma_p)$-instances to each other by simulating the predicates in $\mathscr{L}_{\leqslant}$ and $\mathscr{L}_p$ in the opposite language. For example, a binary tuple $\overline{xy} \preccurlyeq \overline{uv}$ over σ_p can be converted to an equivalent 4-ary tuple $x \leqslant v$ over x, y, u, v. For this reason, we view each $(\mathscr{Q}, \sigma)$-instance also as a $(\mathscr{P}, \sigma_p)$-instance, assuming relations in σ are of even arity.

It is easy to see that under the previous assumptions on the schema, FO and FO_p are equivalent, i.e., they express exactly the same set of queries. Moreover, every constant-free query in FO (FO_p) is order-generic. The proof follows from Proposition 4.10 and Lemma 5.1. The equivalence of FO and FO_p follows from Theorems 5.9 and 5.8.

Alternatively, one can also provide a direct translation so that the predicates in one language are simulated in the other, and the quantified variables are also simulated (the details are omitted). Due to their equivalence, we only use FO in the formal statements and proofs in the remainder of the section.

The data complexity of FO queries has been studied in [KKR95, KG94, GST94]. In [KG94], Kanellakis and Goldin showed an AC^0 upper-bound for FO queries on normalized inputs. This result is further extended in [GST94] to an extension of FO with the addition operation.

THEOREM 5.2 [KG94, GST94]. *The data complexity of each query in FO is in* AC^0.

The proof of the AC^0 upper bound (see [KG94] for details) is done by an encoding of the dense-order constraint relations into finite relations. This encoding itself is not in AC^0. An algebra working on the representation in terms of finite relations was developed which simulates first-order operations on the dense-order constraint databases. This algebra is shown to be in AC^0. An alternative proof is given in [GST94], which does not require an encoding in terms of finite relations, and moreover, the AC^0 upper bound applies to a wider class.

We now study the definability of numerous queries in the first-order query language for dense-order constraint databases. The queries studied include topological queries such as "region connectivity," "existence of a hole," "Eulerian traversal," "k-dimensional homeomorphism," etc. and graph queries such as "parity" and "transitive closure" (graph connectivity). These queries are shown to be nondefinable in FO. We prove the results mainly using a complexity-theoretic approach but also illustrate the main ideas of the Ehrenfeucht–Fraïssé game-based approach. More specifically, we shall consider the following queries:

- Graph queries defined over finite input. The *parity* query over a schema with one unary relation returns *true* if the number of elements in the input is even; the *transitive closure* query over a schema with a binary relation simply returns the transitive closure of the relation.

- The *convexity* query defined over a schema with a single binary relation. The query answers *true* if the input defines a convex region (the straight line segment between each pair of points in the region is entirely in the region). Convexity is $\leqslant$-generic. Indeed, for each $\mathscr{L}_{\leqslant}$-representable instance I, the formula defining the input relation may contain only atomic formulas of the following forms (with free variables x, y),

$$x \leqslant c, \qquad x \leqslant y, \qquad y \leqslant c,$$

and their negations, where c is in $\mathbb{Q}$. Thus, only horizontal, vertical lines and the line $x = y$ are representable which are preserved under each automorphism of $\mathscr{Q}$ (applied to both dimensions simultaneously). Therefore, the collection of convex instances is closed under automorphisms of $\mathscr{Q}$.

- For each $k \geqslant 1$, the *k-convex covering* query. The query takes a binary relation as input and answers *true* if the input is equivalent to the union of at most k convex regions.

- For each $k \geqslant 1$, the *k-dimensional region connectivity* query. The query is defined over a schema with one k-ary relation symbol, R. The query returns *true* if R is connected. (A region is connected if every pair of points in the region can be linked by a continuous curve lying entirely in the region.)

- For each $k \geqslant 1$, the k-dimensional *at least one hole* and *exactly one hole* queries. Both queries have the same input schema as the k-dimensional region connectivity query. The queries answer *true* if the input region has at least or exactly one hole (respectively).

- For each $k \geqslant 1$, the *k-dimensional Eulerian traversal* query defined over the same schema as the k-dimensional region connectivity query. For each input which is a set of line segments in the k-dimensional rational space, the query returns *true* if there is a traversal which continuously goes through each line segment exactly once. In other words, if we view a line as a set of points, a traversal goes continuously through each point exactly once, except for a finite set of points (the crossings of lines).

- For each $k \geqslant 1$, the *k-dimensional homeomorphism* query. The input schema of the query consists of two k-ary relations. The query returns *true* if the two relations are k-homeomorphic and *false* otherwise. (Two point sets in the k-dimensional rational space $\mathbb{Q}^k$ are *k-homeomorphic* if there is a bi-continuous bijection on $\mathbb{Q}^k$ which maps one to the other.)

There are numerous other interesting queries such as the minimal spanning tree [KKR95] and various tests such as translation, (direct) isometry, similarity, or affinity of two point sets in $\mathbb{Q}^2$ [PVV94]. However, these queries do not commute with automorphisms of $\mathscr{Q}$ and are not order-generic.

The main definability results concerning first order can now be stated.

THEOREM 5.3. *The following queries are definable in FO*:

(i) *convexity*,

(ii) *k-convex covering for each k*,

(iii) *one-dimensional region connectivity*, *one-dimensional at least and exactly one hole*, *and one-dimensional Eulerian traversal.*

The following queries are not definable in FO:

(iv) *transitive closure and parity*,

(v) *for each $k \geqslant 2$, the k-dimensional region connectivity, at least and exactly one hole, Eulerian traversal, and*

(vi) *for each $k \geqslant 1$, k-homeomorphism.*

We now prove Theorem 5.3. The proof for (iii) is obvious. For example, the connectivity of one-dimensional regions holds if the input consists of at most one interval. The other cases are established through a series of lemmas (Lemma 5.4 to Lemma 5.7).

In [Kup90], finite databases and $\mathscr{L}_=$-generic queries over the structure $\mathscr{R}$ of real numbers were considered. It was claimed that each $\mathscr{L}_=$-generic query in $\mathscr{L}_\times$ is definable using only equality, i.e., in $\mathscr{L}_=$. Unfortunately, this result does not hold [Via93, Kup93b]. A simple counterexample due to Gurevich can be found in [AHV95, Ex. 17.27, p. 462], where a variant of the parity query (which is generic) was expressible with an order, but inexpressible without it. Nevertheless, a weaker result holds. More recently, it was shown that each $\mathscr{L}_=$-generic query in $\mathscr{L}_\times$ is definable with the equality and the order relation only (i.e., in $\mathscr{L}_\leqslant$) [BDLW96], which improves a similar result on the linear language ($\mathscr{L}_\times$ without the multiplication $\times$) proved earlier in [PVV95, ST96]. These results have important consequences on the expressive power of the languages with respect to relational queries. In particular, since the parity of the cardinality of a relation and the graph transitive closure are not expressible in $\mathscr{L}_\leqslant$ by Theorem 5.3(iv), it follows that they are not expressible in $\mathscr{L}_\times$. Moreover, in [GS96a], first-order reductions [Imm87] were used to show that many queries including the minimum spanning tree are not expressible in $\mathscr{L}_\times$.

LEMMA 5.4. *The queries convexity and k-convex covering for each $k \geqslant 0$ are definable in FO.*

Proof. We first show that the *convexity* query is expressible in FO. Let $\sigma = \{R\}$ be the database schema and I an instance of σ. Then, $I(R)$ is equivalent to a formula $\bigvee_i \bigwedge_j \psi_{i,j}$, where $\psi_{i,j}$ is either $s = t$, $s \leqslant t$, or their negations, and s, t are either constants in $\mathbb{Q}$ or variables. Hence each atomic formula represents either a line parallel to the horizontal or vertical axis, or the diagonal line $x = y$, or the complement, or a half plane defined by a line of the above three types. It is easy to see that for $I(R)$ to define a single convex, the region it defines must be of one of a fixed finite set of shapes. Several possible shapes in dimension 2 are shown in Fig. 2. Now the query in FO expressing the convexity query checks if the input corresponds to one of these known shapes and the construction is straightforward but tedious.

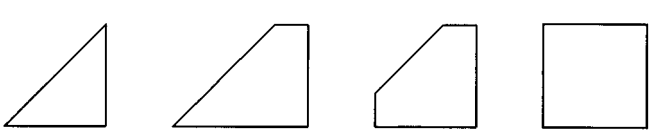

FIG. 2. Some possible convex regions.

For the *k-convex covering* query, Let $k \geqslant 0$ be fixed. We can simply "try" all possible combinations of covering by k largest possible shapes from the set of fixed shapes from the lower left corner to the upper right corner of the input. Due to the fact that there are only a fixed number of possible convex shapes representable in the databases, the query can pick a corner which violates convexity and then construct the largest possible convexes to cover it. The process need to be repeated at most k times. ∎

LEMMA 5.5. *For each integer $k \geqslant 2$, the k-dimensional region connectivity, at least and exactly one hole queries are not definable in FO.*

Proof. The proof is accomplished by reductions from the Boolean function MAJORITY to the queries of our concern. The function MAJORITY takes as input n Boolean variables $x_1, ..., x_n$ and outputs 1 if more than half of the variables have the value 1, and 0 otherwise. The reductions can be done in AC^0. Since MAJORITY is not in AC^0 [FSS84] while $FO \subseteq AC^0$ (Theorem 5.2), it follows that the *region connectivity*, and the *at least* and *exactly one hole* queries are not definable in FO.

Formally, we should consider a "relational" majority problem corresponding to the problem on Boolean variables. The correspondence is straightforward. Consider two relations R and S with $S \subseteq R$. Then MAJORITY(R, S) holds if $2|S| \geqslant |R|$. If $|R| = n$, the Boolean variables $(x_1, ..., x_n)$ correspond to the elements $(a_1, ..., a_n)$ of R and $x_i = 1$ iff $S(a_i)$ holds.

We now describe the AC^0-reduction to *region connectivity*. We fix k and consider a two-dimensional plane in the k-dimensional space. In the following, we use (i, j) to refer to a point in the plane. Let $(x_1, ..., x_n)$ be the input for MAJORITY. We construct the following line segments in the plane:

- From $(n, (n+1)/2)$ to (n, n), from $(0, n)$ to (n, n), and from $(0, 0)$ to $(n, 0)$;
- For each $1 \leqslant i \leqslant n$ and each $0 \leqslant j < n$, from $(i-1, j)$ to $(i, j + x_i)$ (corresponding to x_i).

An example of the resulting line segments is shown in Fig. 3a, where $n = 6$ and

$$x_1 x_2 x_3 x_4 x_5 x_6 = 1\ 0\ 1\ 0\ 1\ 1.$$

It is quite easy to see that the set of line segments is connected if and only if MAJORITY$(x_1, ..., x_n) = 1$, since the line starting from the point $(0, 0)$ reaches the vertical line at the right of the figure (line segment from $(n, (n+1)/2)$ to (n, n)). To complete the proof, it is noted that dense-order

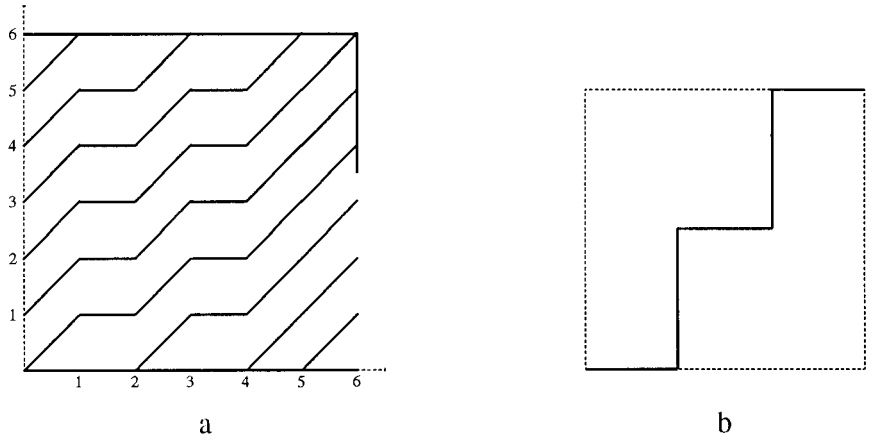

FIG. 3. Illustration of reductions.

constraints do not allow diagonal lines. Hence in constructing the dense order database, we have to replace each diagonal line $((i, j), (i+1, j+1))$ by three horizontal and two vertical line segments as shown in Fig. 3b.

The reduction from MAJORITY to both *at least* and *exactly one hole* queries is constructed as follows. Let $x_1, ..., x_n$ be the input variables for MAJORITY. Assume again we consider a two-dimensional subplane in the k-dimensional space and use (i, j) to refer to a point in the plane. We construct the following line segments in the plane for the relation R (Fig. 4):

- From $(0, 0)$ to $(n, 0)$ and from $(n, \lfloor n/2 \rfloor)$ to $(n, \lfloor n/2 \rfloor + 1)$, and
- For each $1 \leqslant i \leqslant n$ and each $0 \leqslant j < n$ from $(i-1, j)$ to $(i, j+x_i)$ (corresponding to x_i).

Now, MAJORITY$(x_1, ..., x_n) = 1$ if and only if the point $(0, 0)$ eventually reaches a point (n, i) for some $i > \lfloor n/2 \rfloor$ if and only if the line segment $(n, \lfloor n/2 \rfloor)$ to $(n, \lfloor n/2 \rfloor + 1)$ forms a single hole with the segment $(0, 0)$ to $(n, 0)$ (Fig. 4), i.e., both at least and exactly one hole queries will produce the answer *true*. The reduction is again obviously in AC^0. ∎

It is interesting to note here that for the case where $k \geqslant 3$, the proof can be done by an alternative AC^0 reduction from the Boolean function PARITY that is known to be outside of AC^0 [FSS84, Ajt83]. The PARITY function takes as input n Boolean variables and returns *true* if an even number of variables have the value 1, and *false* otherwise. Suppose $x_1, ..., x_n$ are input Boolean variables for PARITY. Again we fix k and consider a three-dimensional subspace. We also use (i, j, k) to refer to a point in the subspace. In the first step of the reduction, we select those variables having the value 1. Let $\{a_1, ..., a_m\}$ be the resulting set, i.e., satisfying the condition $x_j = 1$ iff $j = a_i$ for some $1 \leqslant i \leqslant m$. Without loss of generality, we assume that for each $2 \leqslant i \leqslant m$, $a_{i-1} < a_i$. For each $1 \leqslant i \leqslant m$, we represent a_i by the point $(a_i, 0, 0)$ in the subspace. The input relation is obtained as follows. For each $1 \leqslant i \leqslant (m-2)$, we construct (straight) line segments from $(a_i, 0, 0)$ via points $(a_i, 0, a_i)$, $(a_i, 1, a_i)$, $(a_{i+2}, 1, a_i)$, $(a_{i+2}, 0, a_i)$, and eventually to $(a_{i+2}, 0, 0)$. Finally, the input relation also includes a set of lines starting from $(a_m, 0, 0)$ via points $(a_m, 1, 0)$, $(a_1, 1, 0)$, and then to $(a_1, 0, 0)$. Figure 5 shows some of the lines in the database. It is easy to verify that the input relation is region connected

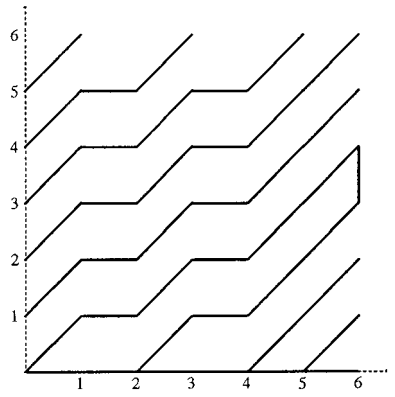

FIG. 4. Reduction to at least and exactly one hole queries.

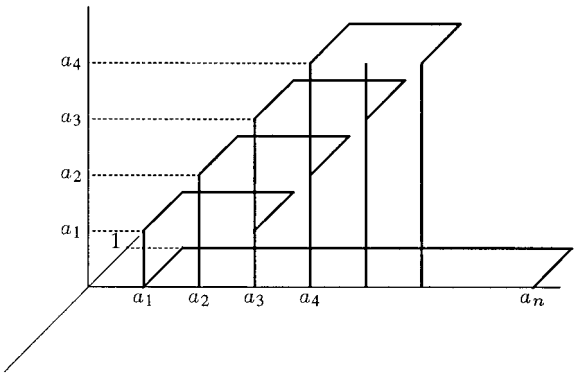

FIG. 5. Reduction from PARITY.

iff m is even, i.e., PARITY returns *true*. Note that this reduction does not naturally apply to the case when $k = 2$, since the lines linking odd nodes and even nodes would overlap. Furthermore, the reduction becomes non first order (nor in AC^0) if it lays the lines of even (or odd) nodes in the other side of the y axis. This alternative proof exhibits the difference between the Boolean functions MAJORITY and PARITY in proving non definability results. ▮

The reduction from PARITY to *region connectivity* shown in Fig. 5 can be easily modified into a reduction to *transitive closure*. For the *transitive closure* query, the reduction is done such that each endpoint and intersection of two lines is a node in the graph, and each straight line becomes an edge. Hence PARITY is *true* iff the graph is connected. It follows that *transitive closure* is not expressible. To show that relational *parity* is not in FO, we use a reduction which constructs the set $\{i \mid x_i = 1\}$. Then PARITY is *true* iff parity is *true*. It follows that:

LEMMA 5.6. *The transitive closure and parity queries are not definable in FO.*

Next we prove the nondefinability of the *Eulerian traversal* query.

LEMMA 5.7. *The k-dimensional Eulerian traversal query for each $k \geqslant 2$ and the ℓ-dimensional homeomorphism query for each $\ell \geqslant 1$ are not expressible in FO.*

Proof. The proof is based on reductions from the Boolean function HALF which outputs 1 if exactly half of its inputs have the value 1 and the other half 0. It can be verified that using HALF gates, one can compute MAJORITY by circuits of constant depth [FSS84], which means MAJORITY reduces to HALF via AC^0 reductions. Therefore HALF is not in AC^0.

The idea of the construction of the reduction from HALF to *k-dimensional Eulerian traversal* ($k \geqslant 2$) is to use pairs of parallel line segments in the reduction slightly modified from the one used in Lemma 5.5. An example of the reduction for $x_1x_2x_3x_4x_5x_6 = 1\,0\,1\,0\,1\,1$ is shown in Fig. 6. Now if HALF outputs 1, the parallel lines originating from (0, 0) go to just below the lowest teeth on the right. Hence a Eulerian traversal exists. Otherwise, if the lines from (0, 0) go too high (Fig. 6) or too low, the lines will be broken into at least two connected parts and Eulerian traversal is impossible.

The reduction from HALF to *k-dimensional homeomorphism* is as follows. Suppose $x_1, ..., x_n$ are the inputs for HALF. We consider a one-dimensional subspace in the k-dimensional space and simply use (i) to denote a point in this subspace. The reduction constructs two finite sets R_1, R_2 of points such that $R_1 = \{-i \mid 1 \leqslant i \leqslant n\}$ and $R_2 = \{i, n+i \mid x_i = 1\}$. Hence, HALF is *true* iff R_1 and R_2 have the same number of points and thus are homeomorphic. Clearly both reductions are in AC^0. ▮

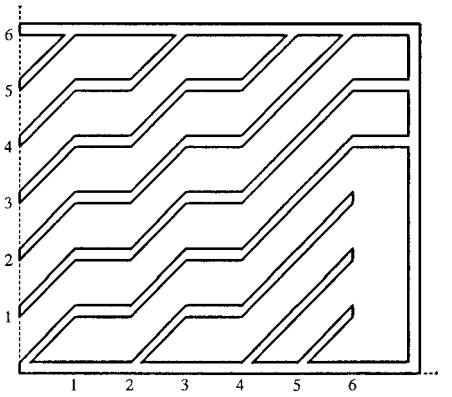

FIG. 6. Reduction to Euler traversal.

In Section 6 we will see that the *Eulerian traversal* query is nevertheless computable in PTIME.

Ehrenfeucht–Fraïssé Games and Applications

Despite the failure of many tools from classical model theory for constraint databases, Ehrenfeucht–Fraïssé games remain a powerful technique for constraint query languages. We first briefly review Ehrenfeucht–Fraïssé games and then present applications of the games.

Let $\mathscr{A}$ and $\mathscr{B}$ be two $\mathscr{L}$-structures with universes A and B (respectively). Suppose further that r is a positive integer. The *game of length r associated with $\mathscr{A}$ and $\mathscr{B}$* is played by two players, I (the spoiler), and II (the duplicator), making r moves each. Player I starts by picking an element in A or B and player II responds by picking an element in the opposite structure. This is repeated for r times. At each move, player I has the choice of the structure, and player II must play in the opposite structure. Let a_i (respectively b_i) be the ith element picked in A (respectively B). Player II wins the round $\{(a_1, b_1), ..., (a_r, b_r)\}$ iff the mapping $a_i \mapsto b_i$ (for $1 \leqslant i \leqslant r$) is an isomorphism of the substructures of $\mathscr{A}$ and $\mathscr{B}$ generated by $\{a_1, ..., a_r\}$ and $\{b_1, ..., b_r\}$ (respectively), $\mathscr{A}/\{a_1, ..., a_r\}$ and $\mathscr{B}/\{b_1, ..., b_r\}$.

Player II *wins the game of length r* associated with $\mathscr{A}$ and $\mathscr{B}$ if she has a winning strategy, i.e., player II can always win any game of length r on $\mathscr{A}$ and $\mathscr{B}$, no matter how player I plays. This is denoted by $\mathscr{A} \equiv_r \mathscr{B}$. Clearly $\equiv_r$ is an equivalence relation on structures.

Intuitively, the equivalence $\mathscr{A} \equiv_r \mathscr{B}$ says that $\mathscr{A}$ and $\mathscr{B}$ cannot be distinguished by looking at just r elements at a time in the two structures. The main result concerning Ehrenfeucht–Fraïssé games is that the ability to distinguish among structures using games of length r is equivalent to that using first-order sentences of quantifier depth r.

THEOREM 5.8 [Fra54, Ehr61]. *Let $\mathcal{L}$ be a first-order language, $\mathcal{A}$ and $\mathcal{B}$ two $\mathcal{L}$-structures, and r a positive number in $\mathbb{N}$. $\mathcal{A}$ and $\mathcal{B}$ satisfy the same set of sentences in $\mathcal{L}$ with quantifier depth r iff $\mathcal{A} \equiv_r \mathcal{B}$.*

If σ is a database schema, then $\mathcal{L}$-representable $(\mathcal{A}, \sigma)$-instances are in fact $(\mathcal{L} \cup \sigma)$-structures. Therefore, Ehrenfeucht–Fraïssé games are applicable in this context. We will present two specific applications of Ehrenfeucht–Fraïssé games in the study of constraint query languages: one merely technical to establish the intimate relationships between sentences (Boolean queries) in FO and in FO_p, and the other is an alternative proof for region connectivity in Lemma 5.5.

Let σ be a schema whose relations have even arity. We consider the corresponding schema σ_p for the point representation which consists of the same number of relation symbols, the arity of each is half of the corresponding relation in σ. Since $(\mathcal{L}_{\leqslant}, \sigma)$-instances and $(\mathcal{L}_p, \sigma_p)$-instances can be translated into each other, we view each instance of one also an instance of the other. We denote by $\mathcal{A} \equiv_r^p \mathcal{B}$ the equivalence associated to the point game of length r. To simplify terminologie, we call the games on $(\mathcal{L}_{\leqslant}, \sigma)$-instances the *value* games and that on $(\mathcal{L}_p, \sigma_p)$-instances the *point* games.

THEOREM 5.9. *Suppose σ is a schema in FO and σ_p its dual in FO_p. Let I, J be two $(\mathcal{L}_{\leqslant}, \sigma)$-instances and $r \in \mathbb{N}$ be positive*:

1. *$I \equiv_r J$ as $(\mathcal{L}_{\leqslant}, \sigma)$-instances implies $I \equiv_{r/2}^p J$ as $(\mathcal{L}_p, \sigma_p)$-instances.*

2. *$I \equiv_{r^2}^p J$ as $(\mathcal{L}_p, \sigma_p)$-instances implies $I \equiv_r J$ as $(\mathcal{L}_{\leqslant}, \sigma)$-instances.*

Proof. To see (1), we simply apply the winning strategy of the value game on I, J while playing the point game on I, J. Since in each round of the point game, we can at most reproduce two rounds of the value game, the winning strategy can be pursued during $r/2$ steps. To prove (2), we essentially employ the same strategy. However, in this case the winning strategy of the point game may not correspond exactly to the same elements played in the value game. We consider the full two-dimensional matrix formed by the r elements of the value game in each dimension. Assume that there is a winning strategy for the point game with r^2 steps. Suppose now that $a_1, ..., a_n$ are all the n constants picked in one structure in the value game and player I picks a distinct a_{n+1}. To respond, player II considers the winning strategy for the point game, given that all the $(n+1)^2$ points defined with $a_1, ..., a_{n+1}$ have been chosen on the corresponding structure. Player II considers a point with a_{n+1} in at least one dimension and consequently picks in the opposite structure the corresponding new constant. (It must be a single new element since otherwise the isomorphism will be broken.) ∎

It is unclear whether the quadratic number of steps in the above theorem is necessary or if a smaller number of steps is sufficient to reconstruct the winning strategy for the value game from the other.

We now turn to an application of the Ehrenfeucht–Fraïssé games. In this case, we use the following proposition implied by Ehrenfeucht–Fraïssé's general result (Theorem 5.8).

PROPOSITION 5.10. *Let ϕ be a sentence in $\mathcal{L}$ with quantifier depth r, and $\mathcal{A}$ and $\mathcal{B}$ two $\mathcal{L}$-structures. If $\mathcal{A} \models \phi$ and $\mathcal{A} \equiv_r \mathcal{B}$, then $\mathcal{B} \models \phi$. Similarly, If $\mathcal{A} \models \phi$ and $\mathcal{A} \equiv_{r^2}^p \mathcal{B}$, then $\mathcal{B} \models \phi$.*

Proposition 5.10 provides a technique for proving that a Boolean query K is not definable by any first-order sentence. Indeed, it is sufficient to exhibit, for each r, two $\mathcal{L}$-representable $(\mathcal{A}, \sigma)$-instances $\mathcal{A}_r$ and $\mathcal{B}_r$ such that $\mathcal{A}_r$ belongs to K, $\mathcal{B}_r$ does not, and $\mathcal{A}_r \equiv_r \mathcal{B}_r$.

We use the point game to illustrate a proof of Lemma 5.5 for the *region connectivity* query. The query input is a binary relation (in terms of $\mathcal{L}_{\leqslant}$). The players play on two instances. They choose points alternatively. We show that player II has a winning strategy for the point game. It should be noted that the point game offers an advantage over the value game in this context, namely that it corresponds better to the intuition of what is going on. By Theorem 5.9, it suffices to describe a winning strategy for the point game.

We first describe the two instances on which the game is played. Both instances consist of collections of line segments of finite length, one being a connected region and the other not. The two instances, $\mathcal{A}_r$ and $\mathcal{B}_r$, are almost identical except for one part. Each looks like two imbricated combs. But they share one tooth in $\mathcal{A}_r$ and none in $\mathcal{B}_r$ (Fig. 7), so that $\mathcal{A}_r$ is connected while $\mathcal{B}_r$ is not. The teeth are not straight lines but complex zigzags. The number of teeth and the number of segments in a tooth is exponential in r (the number of steps of the game). Moreover, the "distinguished points," such as the root of a teeth, its extremity, and the points where the line segments meet, have no coordinates in common. So, in particular, the root and the extremity of a tooth are not on a vertical line, and the teeth all have different lengths.

We now consider the winning strategy of player II. At each step of the game he has to maintain the isomorphism between the two substructures of $\mathcal{A}_r$ and $\mathcal{B}_r$ restricted to the domain of selected points. To maintain the isomorphism, it suffices to ensure that the point chosen at each step is in the relation iff the previous point chosen by player I is, and that the relationship (respective positions) with the other chosen points are similar in the two structures. Given the definition of the two structures, this is a rather easy task.

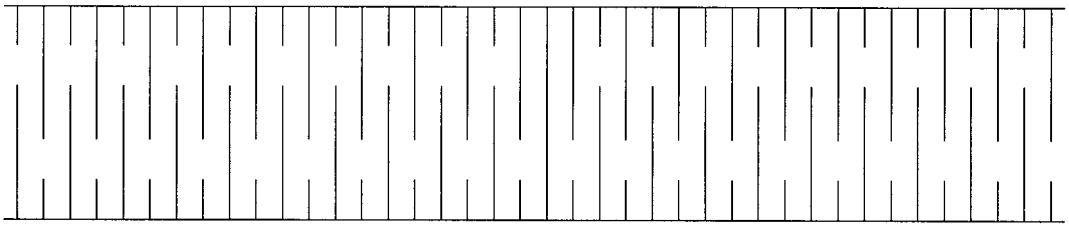

FIG. 7. A skeleton of the instance $\mathscr{A}_r$.

We assume that they each player has r^2 pebbles that can be placed on the points. The winning strategy of the duplicator is mostly to copy moves (by playing identical points). Problems arise only when the spoiler plays on the tooth on $\mathscr{A}_r$ linking the two combs. If the spoiler plays on its two roots, the duplicator, plays on the roots of two opposite teeth in $\mathscr{B}_r$. Now, if the spoiler plays on their extremities, the duplicator plays on the extremities of other teeth in the neighborhood in $\mathscr{A}_r$. Since the number of teeth and the number of angles in a tooth are exponential in r, this can go on for r^2 moves. The duplicator plays on points such that the coordinates of the marked points in the two instances satisfy identically $=$ and $\leqslant$. What makes this strategy possible is that the distances do not have to be respected. This is due to the fact that the language is restricted to the predicates $=$ and $\leqslant$.

The formal proof goes by induction on the number of moves. It is a bit tedious to write but there is no difficulty.

The previous example shows that Ehrenfeucht–Fraïssé games offer an intuitive proof technique for nondefinability results in the context of dense-order constraints. The point game was developed to offer a more intuitive setting than the value-based game, where the players play on the coordinates of points versus points themselves. The games can easily be used to prove nondefinability results of other queries in FO.

In particular, it can be seen that the parity of the number (if finite) of points, segments, connected components, etc. cannot be expressed in FO nor can the transitive closure of finite graphs. The winning strategy of player II in most of these examples follows the same lines as the classical proofs in the context of finite model theory with a discrete order. There are sometimes even easier to formulate since there is no successor function, and so the size of the structures on which the game is played can be chosen smaller.

The Ehrenfeucht–Fraïssé games, however, cannot be extended to languages including arithmetic operations as a proof technique for showing nondefinability results. Indeed arithmetic predicates or functions introduce combinatorics which imply extremely complicated winning strategies and, therefore, abolish the elegance of this proof technique. Even worse, in some cases the distinctive power of the game is much too strong, and player I wins.

Consider, for instance, addition and multiplication. Any two instances such as $\mathscr{A}_r$ and $\mathscr{B}_r$ in the previous proof can be distinguished using a property of the coordinates of the points based on $+$ and $\times$. Any algebraic number can indeed be defined by a first-order formula. So, in particular the following could be defined: there is a point (a, a') in $R^{\mathscr{A}_r}$ not in $R^{\mathscr{B}_r}$, where a and a' are some specific algebraic numbers.

6. FIXPOINT ORDER-GENERIC QUERIES

In this section, we study two fixpoint query languages for databases in valued-based and point-based representations, namely $DATALOG^{\neg}$ and $DATALOG_p^{\neg}$. The languages are extensions of inflationary Datalog with negation for constraint databases. We first show that the query languages express only order-generic queries and then establish their equivalence. We prove that many interesting topological queries studied in the previous section such as *region connectivity*, *existence of a hole*, *Eulerian traversal*, *parity*, and *transitive closure* can be expressed in $DATALOG^{\neg}$ and $DATALOG_p^{\neg}$. The main result of the section is that $DATALOG^{\neg}$ ($DATALOG_p^{\neg}$) captures exactly the class of order-generic queries computable in PTIME. The proof technique involves a novel use of a normal form for dense-order constraint databases. We first review the $DATALOG^{\neg}$ language, discuss a few example of queries, and then prove the PTIME characterization result.

The language $DATALOG^{\neg}$ was originally introduced in [KKR95]. A $DATALOG^{\neg}$ query has the same syntax as Datalog [AHV95] except that negation and constraints involving $\leqslant$ are allowed in the body of its rules. Informally, a *rule* is of the following form:

$$A(x_1, ..., x_n) \leftarrow B(y_1, ..., y_m), ..., \neg C(z_1, ..., z_k), ..., s_1 \leqslant t_1, ..., s_\ell \leqslant t_\ell,$$

where $x_1, ..., x_n, y_1, ..., y_m, z_1, ..., z_k, s_1, ..., s_\ell, t_1, ..., t_\ell$ are (not necessarily distinct) variables or rational numbers and $A, B, ..., C, ...$ are predicate symbols. (Note that the usual safety restrictions on variables in $x_1, ..., x_n$ are not necessary here.) A $DATALOG^{\neg}$ *query* is a set of rules such that each

predicate symbol in the schema σ appears only in the body (right-hand-side) of any rule and a special predicate symbol (*ans*) is the answer predicate.

The semantics of $DATALOG^{\neg}$ queries is defined in the usual manner. Each rule is viewed as an FO query. Initially all relations corresponding to predicates not in the database schema are empty. In an iteration, all rules (FO queries) are executed; the union of the result of a rule and the relation of its head predicate is stored in the same head relation. The iteration is repeated until the head relations reach a fixpoint. It was shown in [KKR95] that the query evaluation process stops after a finite number of iterations for each database instance and consequently the output of a query is always an $\mathscr{L}_{\leqslant}$-representable relation. In other words, $DATALOG^{\neg}$ can be evaluated bottom-up and in closed form.

Since each $DATALOG^{\neg}$ query simply iterates some FO queries for a finite number of times, it is easy to verify that each inflationary $DATALOG^{\neg}$ program defines an $\leqslant$-generic query. The proof is based on Proposition 4.10 and an induction on the number of iterations.

$DATALOG_p^{\neg}$ is the fixpoint extension of FO_p for the point based representation defined in a way similar to $DATALOG^{\neg}$. Since queries in $DATALOG_p^{\neg}$ are evaluated iteratively using first-order queries, it is easy to see that $DATALOG^{\neg}$ and $DATALOG_p^{\neg}$ express exactly the same set of queries. In the remainder of the section, we focus on $DATALOG^{\neg}$ in the sequel.

THEOREM 6.1. *Every constant-free query in* $DATALOG^{\neg}$ *is* $\leqslant$*-generic.*

If $\leqslant$-genericity is generalized to include constants, it can be easily shown that $DATALOG^{\neg}$ (possibly with constants) satisfies the generalized $\leqslant$-genericity property.

The data complexity of $DATALOG^{\neg}$ queries has been studied by Kanellakis, Kuper, and Revesz in [KKR95], where it is shown that the data complexity of $DATALOG^{\neg}$ is in PTIME.

THEOREM 6.2 [KKR95]. *The data complexity of each* $DATALOG^{\neg}$ *query is in PTIME.*

We now show that some of the queries not expressible in FO discussed earlier are definable in $DATALOG^{\neg}$.

EXAMPLE 6.3. We consider the *two-dimensional region connectivity* query. The intuitive idea is to perform alternatively horizontal, vertical, and diagonal "sweeps." We can start from some point (e.g., the lowest and leftmost) within the input region R and store it in a temporary relation S at the initial iteration step. That is at the initial step S contains a single point of R. For odd (respectively even) iterations, we extend S horizontally (respectively vertically) by adding points of R into S which are horizontally (respectively vertically) connected to S. The process ends when S stops growing. Finally, if S is the same as R, the algorithm stops and answers "the region is connected"; otherwise, it is "not connected."

To express the query in $DATALOG^{\neg}$ we first prepare the sweeps by constructing a relation *Sweep* of arity 4 such that Sweep(x, y, u, v) if the points (x, y) and (u, v) are both in R and on the same *vertical* line or on the same *horizontal* line or on the same *diagonal* line $(x = y)$, and the segment between them is entirely in R. The last two conditions can be stated as:

- $x = u$ and for each z between y and v, the point (x, z) is in R, or
- $y = v$ and for each z between x and u, the point (z, y) is in R, or
- $x = y$ and $u = v$ and for each z between x and u, the point (z, z) is in R.

It is easy to see that *Sweep* can be expressed in first order. We can then construct the connectivity relation of points in R using the following classical transitive closure rules:

$$\mathrm{Conn}(x, y, u, v) \leftarrow \mathrm{Sweep}(x, y, u, v)$$

$$\mathrm{Conn}(x, y, u, v) \leftarrow \mathrm{Conn}(x, y, w, z), \mathrm{Conn}(w, z, u, v).$$

Now R is connected if every pair of points in R is connected in *Conn*. Finally a subtle point under the inflationary semantics is to delay the connectivity check until the computation of *Conn* is completed. This can be dealt with using the timestamp technique described in [AHV95] (in the proof of Lemma 14.4.4).

Note that in the above example, the number of sweeps needed in the $DATALOG^{\neg}$ program is roughly the number of necessary "turns" the input relation contains, which depends on the number of constraints in the input. Also, although the example only shows the two-dimensional case, the extension to k-dimensional case (for an arbitrary $k \geqslant 2$) is not difficult.

Since *region connectivity* is expressible in $DATALOG^{\neg}$, it is not hard to see that the queries *at least* and *exactly one hole* are also expressible. Indeed, a closed region has at least one hole iff its complement is not connected, exactly one hole iff its complement has two connected regions. Open regions need a more thorough case analysis but they can be dealt with using *region connectivity*.

EXAMPLE 6.4. We consider the *Eulerian traversal* query for the two-dimensional case. Again, the method can be extended to the general case. The query can be expressed in $DATALOG^{\neg}$ in several steps. The first step is to check if the input consists of only lines. This is done by examining if the input contains a rectangle. Then, a connectivity check is

performed, using the *region connectivity* query. If the input passes both tests, we can extract all intersection points and end points. This can be done using the *Sweep* relation and checking if a point expands in more than two directions (intersection point) or only one direction (end point). Finally, we have to ensure that the input has at most two end points, and only intersection points with an even number of directions of expansion. Essentially, after obtaining intersection and end points, the problem reduces to a (finite) graph problem.

As a final example of query, we consider the *one-dimensional homeomorphism*. In the one dimensional case, the query consists in checking if two input relations have the same sequence of points and intervals. This is clearly easy to express in $DATALOG^{\neg}$. Finally, we can state the following result.

THEOREM 6.5. *The following queries are expressible in* $DATALOG^{\neg}$:

1. *For each* $k \geqslant 1$, *k-dimensional region connectivity*, *k-dimensional at least one-hole*, *and exactly one-hole*, *k-dimensional Eulerian traversal.*
2. *One-dimensional homeomorphism.*
3. *Parity and* (*graph*) *transitive closure.*

Note that it is open if the *k-dimensional homeomorphism* query (for $k \geqslant 2$) is expressible in $DATALOG^{\neg}$. However, there is an easy reduction from the graph isomorphism problem to this query when $k \geqslant 3$. Therefore, if *k-dimensional homeomorphism* ($k \geqslant 3$) is expressible in $DATALOG^{\neg}$, then graph isomorphism is in PTIME. The definability of the queries discussed above is summarized in Fig. 8.

Let $\text{PTIME}_{\mathcal{Q}}$ be the set of all $\leqslant$-generic queries computable in PTIME. We show that $DATALOG^{\neg}$ captures the set of all queries in $\text{PTIME}_{\mathcal{Q}}$. This extends the result that inflationary $DATALOG^{\neg}$ captures all relational queries in the context of finite databases [AV91].

Queries	*First-Order*	$DATALOG^{\neg}$
Convexity	Yes	Yes
Convex Covering	Yes	Yes
Region connectivity	No	Yes
At least one hole	No	Yes
Exactly one hole	No	Yes
Eulerian Traversal	No	Yes
Homeomorphism	No	?
Parity	No	Yes
Graph connectivity	No	Yes

FIG. 8. Definability of some queries.

THEOREM 6.6. $DATALOG^{\neg} = \text{PTIME}_{\mathcal{Q}}$.

Remark. A similar statement can be found in [KKR95] and also in [KG94], but its meaning is different. What was proved there was that "(inflationary) $DATALOG^{\neg}$ can express any relational database query computable in PTIME" (Theorem 3.15 of [KKR95]). A relational database query is a mapping from finite relational databases to finite relations. In other words, it was shown that

$$\begin{aligned}\text{PTIME}_{\text{relational input/output}} &\subseteq \text{inflationary } DATALOG^{\neg} \\ &\subseteq \text{PTIME}_{\text{dense-order input/input}} \\ &= \text{PTIME}_{\mathcal{Q}}.\end{aligned}$$

In our result, $\text{PTIME}_{\mathcal{Q}}$ denotes a set of queries over dense-order constraint databases, and not over relational databases as defined in [CH80].

The remainder of the section is devoted to the proof of Theorem 6.6.

Let Q be a PTIME $\leqslant$-generic query and let M be the Turing machine with a one way infinite tape that computes Q. We construct a $DATALOG^{\neg}$ program P_Q that simulates M. The simulation follows three standard steps: constructing an encoding of the input database, simulating the moves of M on the encoding, and then decoding the answer of M to generate the output in constraint form.

Suppose that the query Q has data complexity n^{ℓ}, where n is the size of the database. The program P_Q uses a relation $Comp(j_1, ..., j_\ell, i_1, ..., i_\ell, a, q)$ with arity $2\ell + 2$ to encode a computation of the machine M, where $(j_1, ..., j_\ell)$ indicates the step in the computation (since $DATALOG^{\neg}$ is inflationary, the use of step identifiers simplifies the construction of P_Q), $(i_1, ..., i_\ell)$ identifies a cell on the Turing tape, a is the symbol on the cell, q is either the current state and the head is on the cell or "$-$," and $(j_1, ..., j_\ell)$ indicates the step in the computation where this configuration occurs.

Clearly, simulating the moves of the Turing machine M in $DATALOG^{\neg}$ can be done in the standard way (e.g., [CH80, Var82, Imm86, HS91, HS93]; see also [AHV95]). The main difficulties of the proof are the encoding and decoding steps. We next show how this can be done in $DATALOG^{\neg}$.

Intuitively, the encoding relies on a syntactic normal form of the instances. For example, a binary relation R that defines a point set over $\mathbb{Q}^2$ can be represented by a finite set of regions, each of which has one among a fixed set of atomic shapes: (i) *isolated points*, (ii) *line segments* (open end), (iii) *triangles* (open boundary), and (iv) *rectangles* (open boundary) (see Fig. 9). Note that atomic shapes can also have a border at infinite (bi-infinite line, half-plan, etc.). For each dimension, there is a fixed number of distinct atomic shapes. This property is fundamental to obtain the encoding of an input database.

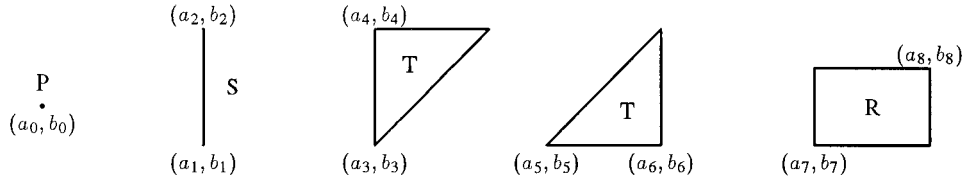

FIG. 9. Atomic shapes in dimension 2.

In the formal development, we introduce the notion of a "primitive" tuple. We define the predicate $<$ such that $x < y$ iff $x \leqslant y \wedge \neg x = y$.

DEFINITION 6.7. A tuple t is *primitive* if t is a conjunction of atomic formulas involving only $=$ and $<$.

A primitive tuple cannot have negation. The following are examples of atomic shapes:[2] $3 < x < 5 \wedge y = 5$ and $x = y \wedge 4 < x$; but the tuple $\neg x = y \wedge 0 \leqslant x$ is not primitive for two reasons: it includes (i) negation and (ii) the predicate $\leqslant$.

EXAMPLE 6.8. Consider the primitive tuple $t = 0 < x_1 < 5 \wedge 0 < x_2 < x_1 \wedge x_3 < 3$ over variables x_1, x_2, x_3. It can be represented in the tabular form [KG94]

	x_1	x_2	x_3
$\bar{l}$	0	0	$-\infty$
$\bar{u}$	5	$+\infty$	3
$\bar{\mu}$	$=$	$>$	?
	$<$	$=$	?
	?	?	$=$

where $\bar{l}(\bar{u})$ are the lower (respectively upper) bounds for the variables and the matrix $\bar{\mu} = \{u_{i,j}\}$ indicates the relationships among the variables. In the above tuple, $\mu_{2,1}$ = "$<$" represents $x_2 < x_1$ and $\mu_{1,3} =$? means that there is no ordering relationship between x_1 and x_3. (Note that the degenerated case $x_2 = 1$ is represented by having $l_2 = u_2 = 1$).

It can be shown (by an induction on the number of atomic formulas) that each primitive tuple corresponds to a tuple in tabular form. If ϕ, ψ are formulas in $\mathscr{L}_{\leqslant}$ with free variables in $x_1, ..., x_k$ (or k-ary tuples, relations), ϕ *implies* ψ iff $\mathscr{Q} \models \forall x_1 \cdots x_k(\phi \rightarrow \psi)$, and ϕ, ψ are *equivalent* if they imply each other. Given two equivalent primitive tuples t_1 and t_2, we say that t_1 is *tighter* than t_2 if (in their tabular form) each lower bound in t_1 is bigger than the corresponding lower bound in t_2, and each upper bound in t_1 is smaller than the corresponding upper bound in t_2, and each $\mu_{i,j}$ in t_1 is "tighter" than or equal to that in t_2 ("$<$" and "$>$" are tighter than "?"). A tuple t is *prime* if it is primitive and tighter than each primitive tuple equivalent to t. It was shown that for each primitive tuple, there always exists a unique prime tuple equivalent to it [KG94]. For example, the tuple t in Example 6.8 is not prime; the equivalent prime tuple is

$$0 < x_1 < 5 \wedge 0 < x_2 < 5 \wedge x_3 < 3 \wedge x_2 < x_1.$$

Prime primitive tuples are also called "canonical" tuples in [KG94].

DEFINITION 6.9. Let R be a k-ary relation. A prime tuple t is *maximal* in R if t implies R and there does not exist a prime tuple t' such that $t \neq t'$, t implies t', and t' implies R.

A set S of tuples is *nonredundant* if for each tuple t in S; S and $S - \{t\}$ are not equivalent. Now the encoding step of P_Q is to "extract," from each input relation R, a nonredundant set S of prime tuples maximal in R such that S and R are equivalent. We call such a set S a *cover* of R.

LEMMA 6.10. *For each k-ary relation R with n constraints* (*counting multiple occurrences of a constraint in distinct tuples*), *the number of tuples in each cover of R is bounded by a polynomial* $f(n)$.

Proof. The proof consists of two steps. First each tuple t in R with m constraints can be decomposed into at most $O(m^k)$ primitive tuples. To obtain the primitive tuples, we divide the m constraints into collections C_i and $C_{i,j}$ ($1 \leqslant i, j \leqslant k$ and $i < j$), where C_i contains all constraints involving only the variable x_i and $C_{i,j}$ all constraints involving x_i, x_j. If any of C_i or $C_{i,j}$ is not satisfiable, t is not satisfiable and can be eliminated. Now assume all collections are consistent. For each collection C_i, since C_i is a monadic relation, it defines a finite set of intervals. Let $D_i = \{\psi_1, ..., \psi_h\}$ be the set of constraints of form $l < x_i < u$ defining the intervals. Each collection $C_{i,j}$ is either empty or equivalent to $x_i = x_j$, $x_i < x_j$, or $x_j < x_i$. Let $\phi_{i,j}$ be the constraint equivalent to the nonempty $C_{i,j}$. It is now clear that t is equivalent to the following set of primitive tuples:

$$\bigvee_{t_i \in D_i,\ 1 \leqslant i \leqslant k} \left(t_1 \wedge \cdots t_k \wedge \bigwedge_{C_{i,j} \text{ is not empty}} \phi_{i,j} \right).$$

Now R is equivalent to a set S of at most polynomially many primitive tuples. Each element in every cover of R

[2] We wrote $a < x \wedge x < b$ as $a < x < b$ as usual.

must include exclusively a tuple in S. Hence the total number of tuples in a cover is at most polynomial in n. ▮

In simulating the Turing machine M, P_Q will encode each k-ary relation into a $2(k^2+2k)$-ary finite relation, where each tuple represents a prime tuple in some cover of R. We illustrate the detail of the encoding in the following example.

EXAMPLE 6.11. Consider the prime tuple "$0<x_1<5 \wedge 0<x_2<5 \wedge x_3<3 \wedge x_2<x_1$" that is equivalent to the tuple in Example 6.8. Since we need a few special symbols such as $\pm\infty$, $<$, $>$, ?, $=$, we use pairs of rational numbers (a, b) to encode both numbers and symbols in the following manner. When $a=0$, (a, b) encodes the rational number b, when $a=1$, (a, b) represents special symbols: (1, 0) for the symbol "$=$," (1, 1) for "$-\infty$," (1, 2) for "$+\infty$," (1, 3) for "$<$," (1, 4) for "$>$," (1, 5) for "?."

Now each primitive tuple is can be represented by six bounds and at most $3\times 3=9$ symbols for $\mu_{i,j}$, i.e., a 15-ary tuple: $(l_1, u_1, l_2, u_2, l_3, u_3, \mu_{1,1}, \mu_{1,2}, ..., \mu_{3,3})$. For the tuple shown above, the encoding is (overlines are added for readability):

$$(\overline{0,0},\overline{0,5},\ \overline{0,0},\overline{0,5},\ \overline{1,1},\overline{0,3},\ \overline{1,0},\overline{1,4},\ \overline{1,5},$$
$$\overline{1,3},\overline{1,0},\overline{1,5},\ \overline{1,5},\overline{1,5},\overline{1,0}).$$

LEMMA 6.12. *Let R be a k-ary relation in the schema. There is a $DATALOG^{\neg}$ program $P_{\mathrm{enc}(R)}$ which outputs an encoding of a cover of R.*

Proof (Sketch). We describe the algorithm and give evidence that it can be done in $DATALOG^{\neg}$. Consider the tabular representation of primitive tuples of arity k (see Example 6.8). A *tuple type* is a pair $(\bar{e}, \bar{\mu})$, where $\bar{e}\in\{=, <\}^k$ and $\bar{\mu}$ is the matrix used in the tabular representation. A primitive tuple is of type $(\bar{e}, \bar{\mu})$ if in its tabular representation, $\bar{\mu}$ is the matrix and for each pair of lower and upper bounds l_i, u_i $(1\leqslant i\leqslant k)$, $l_i e_i u_i$. Intuitively, each tuple type defines a particular shape in k-dimensional space. Clearly, there are only a fixed number of distinct tuple types. Let $\tau_1, ..., \tau_\ell$ be an enumeration of all tuple types. The $DATALOG^{\neg}$ program $P_{\mathrm{enc}(R)}$ will loop through each type and find an encoding of a cover of R by the following procedure. Let ϕ_R be the relation to hold the encoding of R:

1. Let $\phi_R=\varnothing$. For $1\leqslant i\leqslant\ell$, do steps (2) and (3).
2. For each (k^2+2k)-ary tuple t, if t encodes a primitive tuple that is prime, maximal in R, and not contained in ϕ_R, add t to ϕ_R.
3. Loop through all tuples t just added into ϕ_R (based on their lowest and leftmost position), if $\phi_R-\{t\}$ and ϕ_R are equivalent, delete t from ϕ_R.

Now to see that step (2) can be done in $DATALOG^{\neg}$, we explain the case when $k=2$ and τ_i defines open rectangles (other situations are similar). Clearly it is easy to check if an encoding t represents an open rectangle by ensuring that (i) the number and symbol encoded are correct, (ii) each lower bound is smaller than the corresponding upper bound, and (iii) the matrix type is the same as the one defined by τ_i. To complete the step, we also need to test if it is contained in R and there is no larger rectangle that is contained in R and contains the tuple t. Both conditions are expressible in first order. For step (3), recursion is necessary and since the tuples are totally ordered, expressing it in $DATALOG^{\neg}$ is straightforward. ▮

By Lemma 6.12, there is a $DATALOG^{\neg}$ program that encodes each relation in the database into a finite relation. Once this is completed, we need to generate the initial configuration for the Turing machine M. The difficulty here is that we cannot directly store the rational numbers occurring in the encoding of the relations directly on the Turing tape because we do not have direct representations (encodings) for them. However, since the query Q is $\leqslant$-generic, it commutes with any automorphism of Q. We apply an automorphism ρ of $\mathcal{Q}$ defined below to the database I and construct the initial configuration of M using the (isomorphic) database $\rho(I)$. The automorphism ρ is also remembered as a finite relation. When M completes, the output $Q(\rho(I))$ is mapped back by ρ^{-1}. The $\leqslant$-genericity of Q implies that $\rho^{-1}(Q(\rho(I)))=Q(\rho^{-1}(\rho(I)))=Q(I)$, which is the desired result.

Since there are only a finite set $\mathrm{adom}(I)$ of rationals occurring in the encoding of the database I the automorphism is defined by an order preserving mapping from $\mathrm{adom}(I)$ to the integers:

$$f(a)=\begin{cases}0, & \text{if } a=0\\ i, & \text{if } a>0 \text{ is the } i\text{th smallest positive number in } \mathrm{adom}(I)\\ -i, & \text{if } a<0 \text{ is the } i\text{th largest negative number in } \mathrm{adom}(I).\end{cases}$$

For each integer i, if the binary representation of $|i|$ is $1a_1, ..., a_l$ (when $i\neq 0$) or 0 (when $i=0$), we use $\mathrm{bin}(i)$ to denote the following binary relation:

$$\mathrm{bin}(i)=\begin{array}{|c|c|}\hline 0 & \mathit{sign}\\ 1 & 1\\ 2 & a_1\\ \cdots & \cdots\\ l & a_l\\ \hline\end{array}\qquad \text{if}\quad i\neq 0;\quad \mathrm{bin}(0)=\boxed{0}\boxed{0},$$

where *sign* is "1" if $i>0$ and "-1" if $i<0$.

LEMMA 6.13. *There is a program* P_{adom} *which produces the following* (*finite*) *ternary relation*

$$S_\rho = \bigcup_{a \in \text{adom}(I),\ \rho(a) = i} (\{a\} \times \text{bin}(i)),$$

where ρ *is some automorphism of* $\mathcal{Q}$.

Proof. By Lemma 6.12, we can easily build a unary relation S that holds adom(I). We can now loop through adom(I) and define $\rho(a)$ for each $a \in \text{adom}(I)$, and use another loop to generate bin($\rho(a)$). The details are tedious. ∎

Proof of Theorem 6.6. The inclusion of inflationary $DATALOG^\neg$ in $\text{PTIME}_{\mathcal{Q}}$ has been shown in [KKR95]; each query in $DATALOG^\neg$ expresses a query over dense order databases and can be computed in PTIME. We only prove the converse inclusion. Let Q be a query over dense order constraints computable in polynomial time and M be a Turing machine that computes Q. By definition, M operates on the standard encoding of each input database and produces a standard encoding of the answer to Q. Let I be an input database and ρ be the automorphism as defined above. The proof consists of the following steps:

1. There is a $DATALOG^\neg$ program, P_{init}, which constructs an initial configuration of the Turing machine M for the (encoded) database $\rho(I)$ isomorphic to I and the initial configuration is stored in the finite relation *Comp*.

2. There is a $DATALOG^\neg$ program, P_{comp}, which simulates the computation of M and computes an encoding of the output $Q(\rho(I))$ from an encoding of $\rho(I)$.

3. There is a $DATALOG^\neg$ program, P_{decode}, which computes from the encoded answer $Q(\rho(I))$ on the output tape of M the constraint relation $Q(I)$.

The program P_{init} uses the programs $P_{\text{enc}(R)}$ to obtain the relational representation of each relation R in I and then uses the program P_{adom} to construct the encoding of the initial configuration of M. The steps are expressible in $DATALOG^\neg$. The program P_{comp} simulating M's moves is done in the standard way (for example, see [HS91, HS93]). Finally, the program P_{decode} will first use the automorphism relation S_ρ to obtain a finite representation of the answer $\rho^{-1}(Q(\rho(I))) = Q(I)$ from the encoded output on the Turing tape, and then construct a constraint relation from its relational representation directly. The two steps are done similarly to the encoding process but in the reverse order. ∎

Except for technical differences in the encoding/decoding steps, the above proof follows the same ideas as for the relational case [Var82, Imm86]. We note here that in this paper the complexity classes of queries are defined based on Turing machines computing answers. An alternative definition is to define complexity (classes) of queries based on testing whether a tuple is in the answer. For the classical relational databases, various well-known complexity-based query classes including LOGSPACE and PTIME coincide under the two definitions. For constraint queries, the class $\text{PTIME}_{\mathcal{Q}}$ under the two definitions also coincide (proof omitted).

7. MORE RECENT RESULTS ON EXPRESSIVE POWER AND COMPLEXITY

There have been numerous results recently on the expressive power of constraint query languages. In this section we give a brief survey of the results related to complexity and expressive power.

Data Complexity

The data complexity of the first-order query language $\mathcal{L}_\times$, denoted here FO($\leqslant$, $+$, $\times$), for $\mathcal{R}$ was shown to be in NC [KKR95]. The NC result follows from the decidability of the theory of real closed fields [Tar51] and its tractability for the case of a fixed number of variables [BKR86, Ren92]. While this remains the best known upper bound, there are several improvements in various subcases, depending on whether the input database is finite or consists of only linear constraints (i.e., $\times$ does not occur), and whether the query language contains the multiplication. The results for the restricted cases are summarized in the following table:

	FO($\leqslant$, $+$)	FO($\leqslant$, $+$, $\times$)
Finite databases	AC^0 [GST94]	TC^0 [*BL*96]
Linear constraint databases	NC^1 [*GS*96*a*]	NC [KKR95]

The AC^0 result for FO($\leqslant$, $+$) for finite databases follows from a result in [GST94] that establishes the AC^0 bound for a subclass of linear constraint databases. This technique was further extended to show the NC^1 bound for the general case in [GS96a]. On the other hand, Benedikt and Libkin [BL96] studied finite databases and proved the TC^0 bound for queries in FO($\leqslant$, $+$, $\times$). The class TC^0 extends AC^0 with threshold gates. It is known that $\text{AC}^0 \subset \text{TC}^0 \subseteq \text{NC}^1$ and it is generally believed that the inclusion $\subseteq$ is strict.

Expressive Power

The most significant result on the expressive power of constraint languages is that parity and transitive closure over finite databases are not expressible in FO($\leqslant$, $+$, $\times$) [BDLW96, BL96], which extends Lemma 5.6. The proof uses a novel technique from nonstandard analysis. Prior to

this result, it was proved that the parity and transitive queries are not expressible in $FO(\leqslant, +)$ [GST94, PVV95, ST96] using different techniques.

It is worth mentioning here that an important step in nondefinability proofs is to establish the equivalence of expressive power of a first-order language under the natural semantics and under the *active domain* semantics. (In the active domain semantics the range of variables is restricted to the set of all elements occurring in the database and the query itself.) It is now known that (1) $FO \equiv FO^{adom}$ [HS94], (2) $FO(\leqslant, +) \equiv FO(\leqslant, +)^{adom}$ [PVV95], and (3) $FO(\leqslant, +, \times) \equiv FO(\leqslant, +, \times)^{adom}$ [BDLW96, BL96].

As far as queries over general finitely representable databases are concerned, [GS96a] uses first-order reductions to prove that all queries in the second part of Theorem 5.3 are not definable in $FO(\leqslant, +, \times)$. Furthermore, the minimum spanning tree query was also shown not expressible in $FO(\leqslant, +, \times)$, which resolved an open problem raised in [KKR95].

8. FINAL REMARKS AND OPEN PROBLEMS

We introduced the theory of finitely representable models and analyzed the differences with finite model theory and classical model theory. Most of the fundamental classical theorems (completeness and compactness) of model theory were shown to fail in the context of finitely representable structures. There is a lack of tools to investigate the properties of first-order logic over finitely representable models. Ehrenfeucht–Fraïssé games are among the few tools that still hold, but they result in complex and heavy combinatorics in the presence of arithmetic functions. We believe that various proof techniques could be adapted to the case of finitely representable structures. This includes, in particular, the locality in the context of dense orders and results based on the asymptotic probabilities of sentences.

In the case of dense-order constraint databases, we achieved a detailed study of the expressive power of first-order logic; in particular, we gave a characterization of PTIME in terms of $DATALOG^{\neg}$ over dense-order constraint databases. Nondefinability results have been generalized and extended to more general constraints. On the other hand, the generalization of logical characterization of complexity classes is difficult. It remains an interesting open problem whether similar characterization can be found for polynomial constraint databases.

Despite the seeming advantages, constraint databases are still in the theoretical investigation stage. Indeed many questions need to be answered before efficient implementation can become possible, including dealing with finite precision arithmetic [GS96b], aggregate functions [Kup93a, CGK96], indexing [KRVV93], etc.

ACKNOWLEDGMENTS

The authors thank Victor Vianu and the referees for helpful comments.

REFERENCES

[ACGK94] F. Afrati, S. Cosmadakis, S. Grumbach, and G. Kuper, Expressiveness of linear vs. polynomial constraints in database query languages, *in* "Proceedings, Workshop on the Principles and Practice of Constraint Programming, 1994."

[AHV95] S. Abiteboul, R. Hull, and V. Vianu, "Foundations of Databases," Addison–Wesley, Reading, MA, 1995.

[Ajt83] M. Ajtai, Σ_1^1 formulae on finite structures, *Ann. Pure Appl. Logic* **24** (1983), 1–48.

[AV91] S. Abiteboul and V. Vianu, Datalog extensions for database queries and updates, *J. Comput. System Sci.* **43**, No. 1 (1991), 62–124.

[BDLW96] M. Benedikt, G. Dong, L. Libkin, and L. Wong, Relational expressive power of constraint query languages, *in* "Proceedings, ACM Symposium on Principles of Database Systems, 1996."

[BKR86] M. Ben-Or, D. Kozen, and J. Reif, The complexity of elementary algebra and geometry, *J. Comput. System Sci.* **32**, No. 2 (1986), 251–264.

[BL96] M. Benedikt and L. Libkin, On the structure of queries in constraint query languages, *in* "Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, 1996."

[CGK96] J. Chomicki, D. Q. Goldin, and G. M. Kuper, Variable independence and aggregation closure, *in* "Proceedings, ACM Symposium on Principles of Database Systems, 1996."

[CH80] A. K. Chandra and D. Harel, Computable queries for relational data bases, *J. Comput. System Sci.* **2**, No. 2 (1980), 156–78.

[CH82] A. K. Chandra and D. Harel, Structure and complexity of relational queries, *J. Comput. System Sci.* **25**, No. 1 (1982), 99–128.

[CK73] C. C. Chang and H. J. Keisler, "Model Theory," Studies in Logic, Vol. 73, North-Holland, Amsterdam, 1973.

[Cod70] E. F. Codd, A relational model of data for large shared data banks, *Comm. ACM* **13**, No. 6 (1970), 377–387.

[Col75] G. E. Collins, Quantifier elimination for real closed fields by cylindric decompositions, *in* "Proceedings, 2nd GI Conference Automata Theory and Formal Languages," Lecture Notes in Computer Science, Vol. 35, pp. 134–183, Springer-Verlag, New York/Berlin, 1975.

[Com88] K. J. Compton, 0-1 laws in logic and combinatorics, *in* "NATO Adv. Study Inst. on Algorithms and Order" (I. Rival, Ed.), pp. 353–383, D. Reidel, 1988.

[DMM94] L. Van den Dries, A. Macintyre, and D. Marker, The elementary theory of restricted analytic fields with exponentiation, *Ann. of Math.* (1994).

[Dri82] L. Van den Dries, Remarks on Tarski's problem concerning $(r, +, \times, exp)$, *in* "Logic Colloquium" North-Holland/Elsevier, New York, 1982.

[EFT84] H. D. Ebbinghaus, J. Flum, and W. Thomas, "Mathematical Logic," Springer-Verlag, New York/Berlin, 1984.

[Ehr61] A. Ehrenfeucht, An application of games to the completeness problem for formalized theories, *Fund. Math.* **49** (1961).

[End72] H. B. Enderton, "A Mathematical Introduction to Logic," Academic Press, 1972.

[Fag76] R. Fagin, Probabilities on finite models, *J. Symb. Logic* **41**, No. 1 (1976), 50–58.

[Fag93] R. Fagin, Finite model theory—a personal perspective, *Theoretical Computer Science* **116**, No. 1 (1993), 3–31.

[Fra54] R. Fraïssé, Sur les classifications des systèmes de relations, *Publ. Sci. Univ Alger* **I** (1954), 1.

[FSS84] M. Furst, J. B. Saxe, and M. Sipser, Parity, circuits, and the polynomial-time hierarchy, *Math. Systems Theory* **17** (1984), 13–27.

[Gai81] H. Gaifman, On local and non local properties, *in* "Proceedings, Herbrand Symposium Logic Colloquium" (J. Stern, Ed.), pp. 105–135, North-Holland, 1981.

[GG94] E. Grädel and Y. Gurevich, Metafinite model theory, *in* "Logic and Computational Complexity" (D. Leivant, Ed.), Lecture Notes in Computer Science, Vol. 960, Springer-Verlag, 1994.

[GS94] S. Grumbach and J. Su, Finitely representable databases (extended abstract), *in* "Proceedings, ACM Symposium on Principles of Database Systems,1994."

[GS95] S. Grumbach and J. Su, Dense order constraint databases, *in* "Proceedings, 14th ACM Symposium on Principles of Database Systems, San Jose, May 1995."

[GS96a] S. Grumbach and J. Su, Queries with arithmetical constraints, *Theoretical Computer Science* (1996), to appear. An extended abstract appeared in "Proceedings, International Conference on Principles and Practice of Constraint Programming (CP95)" under the title "First-order Definability over Constraint Databases."

[GS96b] S. Grumbach and J. Su, Towards practical constraint databases, *in* "Proceedings, ACM Symposium on Principles of Database Systems, 1996."

[GST94] S. Grumbach, J. Su, and C. Tollu, Linear constraint query languages: Expressive power and complexity, *in* "Logic and Computational Complexity: International Workshop LCC '94," Springer-Verlag, New York/Berlin, 1994.

[HH93] T. Hirst and D. Harel, Completeness results of recursive data bases, *in* "Proceedings, 12th ACM Symposium on Principles of Database Systems, 1993," pp. 244–252.

[HH94] T. Hirst and D. Harel, Recursive model theory, draft, 1994.

[HS91] R. Hull and J. Su, On the expressive power of database queries with intermediate types, *J. Comput. System Sci.* **43**, No. 1 (1991), 219–267.

[HS93] R. Hull and J. Su, Algebraic and calculus query languages for recursively typed complex objects, *J. Comput. System Sci.* **47**, No. 1 (1993), 121–56.

[HS94] R. Hull and J. Su, Domain independence and the relational calculus, *Acta Informatica* **31**, No. 6 (1994), 513–524.

[Hul86] R. Hull, Relative information capacity of simple relational schemata, *SIAM J. Comput.* **15**, No. 3 (1986), 856–886.

[Imm86] N. Immerman, Relational queries computable in polynomial time, *Inform. Control* **68** (1986), 86–104.

[Imm87] N. Immerman, Languages that capture complexity classes, *SIAM J. Comput.* **16**, No. 4 (1987), 760–778.

[KG94] P. C. Kanellakis and D. Q. Goldin, Constraint programming and database query languages, *in* "Proceedings, 2nd Conference on Theoretical Aspects of Computer Software (TACS)," Vol. 789, Lecture Notes in Computer Science, Springer-Verlag, 1994.

[KKR95] P. Kanellakis, G. Kuper, and P. Revesz, Constraint query languages, *J. Comput. System Sci.* **51**, No. 1 (1995), 26–52. (An extended abstract appeared in Proceedings, PODS '90).

[KPV95] B. Kuijpers, J. Paredaens, and J. Van den Bussche, Lossless representation of topological spatial data, *in* "Advances in Spatial Databases, 4th Int. Symp., SSD '95" (M. J. Egenhofer and J. R. Herring, Eds.), pp. 1–13, Springer, 1995.

[KRVV93] P. Kanellakis, S. Ramaswamy, D. Vengroff, and J. Vitter, Indexing for data models with constraints and classes, *in* "Proceedings, ACM Symposium on Principles of Database Systems," pp. 233–243, 1993.

[Kup90] G. M. Kuper, On the expressive power of the relational calculus with arithmetic constraints, *in* "Proceedings, International Conference on Database Theory," pp. 202–211, 1990.

[Kup93a] G. M. Kuper, Aggregation in constraint databases, *in* "Proceedings, Workshop on the Principles and Practice of Constraint Programming, April 1993," pp. 176–183.

[Kup93b] G. M. Kuper, personal communications, 1993.

[PSV96] C. H. Papadimitriou, D. Suciu, and V. Vianu, Topological queries in spatial databases, *in* "Proceedings, ACM Symposium on Principles of Database Systems, 1996."

[PVV94] J. Paredaens, J. Van den Bussche, and D. Van Gucht, Towards a theory of spatial database queries, *in* "Proceedings, 13th ACM Symposium on Principles of Database Systems 1994," pp. 279–88.

[PVV95] J. Paredaens, J. Van den Bussche, and D. Van Gucht, First-order queries on finite structures over the reals, *in* "Proceedings, IEEE Symposium on Logic in Computer Science, 1995."

[Ren92] J. Renegar, On the computational complexity and geometry of the first-order theory of the reals, *Journal of Symbolic Computation* **13** (1992), 255–352.

[Rev90] P. Z. Revesz, A closed form for datalog queries with integer order, *in* "Proceedings, International Conference on Database Theory 1990," pp. 187–201.

[ST96] A. P. Stolboushkin and M. A. Taitslin, Linear vs. order constrained queries over rational databases, *in* "Proceedings, ACM Symposium on Principles of Database Systems, 1996."

[Tar51] A. Tarski, "A Decision Method for Elementary Algebra and Geometry," University of California Press, Berkeley, CA, 1951.

[Tra50] B. A. Trakhtenbrot, The impossibility of an algorithm for the decision problem for finite models, *Doklady Akademii Nauk SSR* **70** (1950), 569–572.

[Ull82] J. D. Ullman, "Principles of Database Systems (2nd edition)," Computer Science Press, Potomac, Maryland, 1982.

[Var82] M. Vardi, Relational queries computable in polynomial time, *in* "Proceedings, 14th ACM Symposium on Theory of Computing, 1982," pp. 137–146.

[Vau60] R. L. Vaught, Sentences true in all constructive models, *J. Symb. Logic* **25**, No. 1 (1960), 39–53.

[VGV95] L. Vandeurzen, M. Gyssens, and D. Van Gucht, On the desirability and limitations of linear spatial database models, *in* "Advances in Spatial Databases, 4th International Symposium, SSD '95" pp. 14–28, Springer, 1995.

[Via93] V. Vianu, personal communications, 1993.

[Yao90] F. F. Yao, Computational geometry, *in* "Handbook of Theoretical Computer Science" (J. van Leeuwen, Ed.), Vol. A, Chap. 7, pp. 343–389, North-Holland, 1990.