DIMACS Technical Report 96-01
January 1996

On Order-Generic Queries

by

Oleg V. Belegradek
Department of Mathematics
Kemerovo State University
Kemerovo, Russia 650043
`beleg@kaskad.kemerovo.su`

Alexei P. Stolboushkin[1]
Department of Mathematics
UCLA
Los Angeles, CA 90024-1555
`aps@math.ucla.edu`

Michael A. Taitslin[2]
Department of Computer Science
Tver State University
Tver, Russia 170000
`mat@mat.tvegu.tver.su`

[1]This work has been partially supported by NSF Grant CCR 9403809.
[2]A part of this research was carried out while the author was visiting UCLA and supported in part by NSF Grant CCR 9403809, and while the author was visiting DIMACS and Princeton and supported in part by a grant from Princeton University.

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

We consider relational databases organized over an ordered domain with some additional relations—a typical example is the ordered domain of rational numbers together with the ternary relation $+$ of addition. In the focus of our study are the first order queries that are invariant under order-preserving "permutations"—such queries are called order-generic. In fact, we consider two formalizations of this notion: "generic", and "locally generic", queries. For several domains order-generic queries fail to express more than pure order queries, for example, every order-generic query over rational numbers with $+$ can be rewritten without $+$. Our goal is to find general conditions on the domain that allow for such a simplification of order-generic queries.

An important difference of this paper from a recent series of related papers (see, for example, [14, 2]) is that we generalize all notions to the case of finitely representable database states—as opposed to finite states—and develop a general lifting technique that, essentially, allows us to extend any result of the kind we are interested in, from finite to finitely-representable states, and thus all the results in this paper are proved for the general case of *constraint databases.*

On the simplification of order-generic queries, we offer two types of results. The results of the first type address *equivalence* of order-generic queries that use the additional predicates (extended queries) to pure order queries. Here we establish the necessary and sufficient condition for an extended query to be equivalent to a pure order query, in terms of behavior of this extended query over the ultra-product of this domain by a non-principal ultra-filter over ω. This characterization relies upon the Continuum Hypothesis, and is used as instrumental in establishing equivalence of generic, as well as locally generic, extended queries to pure order queries for several classes of domains, including in particular the so-called o-minimal domains, integers with $+$, and such. Although, by Shoenfield's Absoluteness Theorem, the Continuum Hypothesis can be eliminated from these latter results, the fact remains, no translation algorithm can be extracted from arguments of this type.

That is why we also offer *effective algorithms* of translating extended to pure order queries for divisible ordered Abelian groups. Such groups—including real or rational numbers with $+$—are all o-minimal.

1 Introduction

1.1 Infinite domains

Although relational databases were invented for finite collections of data, it is often convenient to assume that there is an infinite *domain*—for example, integer or rational numbers or strings—such that the data elements are chosen from this domain. Functions and relations defined over the entire domain, like $<$ and $+$, may also be used in querying, for example, if the language of first-order logic FO is used as the query language, its formulas may use database relations as well as the domain relations, while variables range over the entire domain.

The choice of a specific domain is normally dictated by the applications. Most often, it is integer, or rational, or real numbers with $<$ and some arithmetical operations. Several proof techniques, however, do not rely on specific properties of these domains, and, as a matter of fact, can be applied to wide classes of domains. For instance, several results in Benedict et al. [2] and in this paper are formulated and proved for the so-called o-minimal structures—a diverse family of structures that includes many domains normally considered in the database theory, but, on the other hand, does include quite weird structures, whose relevance to the database theory—or, for that matter, whose very existence—may be legitimately put into question. Yet, our results are applicable to several recursive domains with decidable theories, upon which no disagreement should arise.

1.2 Finitely representable relations

In the traditional relational database theory, the database relations are finite. The trouble with this is, answers yielded by relational queries may or may not be finite. This makes the traditional relational model not closed, in the sense that the output data of queries are of a different nature than input. [1, 4] observed, however, that these output relations are particularly simple, for simple domains (say, for ordered natural numbers or such). Kanellakis et al. [10, 11] concentrated on ordered domains of real, and rational numbers, and observed that, since first order theories of these admit elimination of quantifiers, the answers to first order queries can be represented as quantifier-free first order formulas, and then, if we allow database relations to be arbitrary relations representable by quantifier-free first order formulas to begin with, the so modified relational model becomes closed in the above sense. Such relations are called *finitely representable* (f.r.)

These finitely representable databases are a logical choice, because finitely representable relations appear as results of queries dealing with finite relations anyway, and it is also a natural choice in many applications, say, in geographical databases (cf. [10, 11]).

1.3 Ordered domains and generic queries

For example, consider ordered rational numbers. A query, then, is a mapping that maps every finitely representable database state into a new finitely representable relation. Observe that, if we take any such pair—an f.r. database state and the answer to the query in this state, an f.r. relation—and uniformly change some of the constants used in the finite representations while preserving order between different constants, then the new pair agrees with the mapping. In this sense, all the queries that we can express in the first order language FO are *generic.*

There are some rather simple generic queries, however, that are not expressible. For example, a Boolean query that says that the cardinality of a set—a unary relation—is odd, is not expressible in FO. More examples can be found in [10, 11, 9]. The problem we are interested in is to try to increase the expressive power of FO, while preserving the genericity. There are several ways to increase the expressive power of queries. One of them is to add additional domain functions or relations.

For example, in Grumbach et al. [7, 6] extended queries over the domain $\mathbf{Q}$ of rational numbers together with the relation $<$ of the rational linear order and the usual operation $+$ were considered. Clearly, $x + x = x$ defines 0, which is not definable without $+$, on the other hand, queries like $x + x = x$ are not about order at all, as they are not preservable under order automorphisms. To level the playing field, they consider *locally generic* extended queries only, that is the queries preservable under partial $<$-isomorphisms of the universe. To put it another way, while Arithmetical computations can be carried out inside the query, the input-output relation defined by this query may rely only on the order properties of its input.

Over rational numbers, FO queries that only use $<$ were shown to have the uniform data complexity AC^0 ([9]). Attempts to distinguish the resulting extended queries from order queries in this domain using specific combinatorial or spatial queries not in AC^0—like parity, Eulerian traversal, or region connectivity—have been unsuccessful, and, finally, [7, 6] proved the AC^0 uniform data complexity for the extended queries over finitely representable inputs *with integer constants only.* However, the question of whether or not extended queries are more expressive than order queries, has remained open, and as Grumbach and Su [5] pointed out, "...there is a serious lack of proof techniques..." in this area.

This very problem can be considered in a general form. Consider a list Ω of relational names. Consider two signatures $L_0 = \{<\}$, and $L = \{<\} \bigcup \Omega$. For a database scheme $SC = \{R_1, \ldots, R_n\}$, denote $L_0^+ = L_0 \bigcup SC$ and $L^+ = L \bigcup SC$. Call FO in L_0^+ *restricted,* and FO in L^+ *extended.* The *general problem* is then, under which conditions on the domain are generic extended queries reducible to restricted queries?

Notice that this problem admits several interpretations:

- "generic" may be understood as "locally generic", or a different notion of genericity may be used, for instance, preservation under order automorphisms is simply referred to in this paper as "genericity". While for many domains (rational numbers with $+$, real number with $+, *$, and, generally, for an arbitrary doubly-transitive domain) it does not make a difference, for some other domains it may

- "reducible" may mean that the two languages are equivalent, or that a recursive translation exists

- the problem may be formulated for either all, finite or infinite, database states, or for f.r. states only, or for finite states only

In Section 3 we show that, if all possible states were considered, no translation would be possible even in such simple an example as the additive group of rational numbers.

But of course the really interesting cases are those of finite and of finitely representable database states. In Section 4 we show that these two cases can be treated uniformly. One of the main results of the present paper, Theorem 4.8, is that, over every ordered domain, finitely representable states can be uniformly represented as finite states of another database scheme, with the additional property that these finite database states are FO expressible (in the restricted language) in the old database scheme, and vice versa. This technique, in effect, allows us to lift any result on translatability of extended queries into restricted queries over *finite* database states, to the *finitely representable* states. Recursivity is preserved as well.

This technique can also be used to expand applicability of several other results for finite database states to the case of finitely representable states.

1.4 Recursive translation

Paradaens et al. [14] recently considered real numbers with $+$, and showed that, over finite database states, generic extended queries can be recursively[1] translated into restricted (pure order) queries. Due to our lifting result, the same is automatically true for all finitely representable states.

In this paper, we prove a more general result on recursive translation of generic extended into restricted FO queries over every ordered divisible Abelian group, thus answering, for example, the question from [7, 6] of the role of addition in databases over rational numbers.

Although recursive, our translation is highly inefficient computationally. The size of the pure order formula generated by our algorithm is multi-exponential in the size of the original generic formula with addition. We feel this is a natural phenomenon, in the sense that no efficient (say, polynomial-size) translation is possible. This offers yet another angle of looking at the relative expressive power of extended and restricted queries. Indeed, using an extended language may be beneficial in that it may offer a compact way of expressing generic queries.

Notice also, the set of generic extended queries is undecidable. Our translation algorithm *assumes* genericity, and, if applied to a non-generic query, gives a non-equivalent formula of the restricted language.

These issues are explored in Section 5. The technique in this part is similar to that in [14], and actually, all the results for ordered divisible Abelian groups that we prove can alternatively be proved by refining the methods from [14].

[1] Although the algorithm is not explicit in their paper.

1.5 Non-recursive translation

Benedikt et al. [2] recently showed that the generic extended, and restricted queries have the same expressive power over every *o-minimal* domain.[2] Examples of *o*-minimal structures include, for instance, real numbers with $+$, $*$, and $<$, as well as many other structures. Since every ordered divisible Abelian group is *o*-minimal, this, in one sense, covers the results discussed in Subsection 1.4.

Notice however, that in another sense, this result is of a different nature. First, the proof in [2] is not constructive and does not give an algorithm for translation. Further, this proof cannot be made constructive. This can be argued upon along the following lines. Take an *o*-minimal structure whose first order theory is undecidable, while the first order theory of $<$ alone is decidable—such structures do exist. If a recursive translation existed, this would lead to contradiction.

In this paper, we use model-theoretic techniques to obtain substantially stronger results of this sort. First, we give a necessary and sufficient condition, in terms of domains, for the generic extended queries to have the same expressive power as restricted queries. Second, this technique is developed to cover locally generic queries as well, and we formulate simple sufficient conditions for the locally generic extended queries to have the same expressive power as restricted queries. Specifically, we show that the conditions are satisfied for the following domains:

- *o*-minimal structures
- integer or natural numbers with $+, <$

Thus, our results cover all the known, and some new, cases. Yet the proofs are relatively compact and straightforward. These results are presented in Section 6.

2 Basic Notions

A structure of a relational signature L is a non-empty set with a mapping which assigns to every relation name from L a relation of the same arity over the set. Let U be a countable structure over the signature L. This structure is called the *universe*. In this paper, we always consider ordered universes. This means that L includes a binary relational symbol $<$ whose interpretation in U satisfies the axioms of linear order. Let us denote $L_0 = \{<\}$, and $L = \{<\} \bigcup \Omega$. In words, Ω denotes the additional relational symbols in L.

Databases operating over U use non-signature relational symbols as well. A *database scheme* is a finite collection of relation names of fixed arities. A *database state* (over U) is an assignment to these relation symbols of concrete relations of corresponding arities over U. These relations are called *database relations*. A database state is *finite* iff all the relations are finite. We fix a database scheme SC and denote $L_0^+ = L_0 \bigcup SC$, and $L^+ = L \bigcup SC$.

[2] Again, they considered finite states only, but this can be lifted to finitely representable states using our technique in this paper.

A *database query* can formally be defined as a mapping that takes in a database state (of a fixed d.b. scheme), and produces a new relation, of a fixed arity, over U. Thus, every query has an arity. Specifically, queries of arity 0 are called *Boolean.* A Boolean query defines a mapping from d.b. states to $\{0,1\}$, or, in other words, subsets of all possible d.b. states of a given d.b. scheme.

Queries are formulated using *query languages,* the simplest of which is the language of first-order logic FO. Formulas (=queries) of this language use $=$, as well as the relation names of the signature and of the database scheme. Thus, a d.b. state essentially defines a structure of a larger signature with U as the domain; then a formula with n free variables defines an n-ary relation over U.

Generally, an FO query may yield an infinite answer even in a finite d.b. state. [10] introduced the notion of *finitely representable* database state as a database state where every relation corresponding to a relation name from SC is defined (independently on the others) by a quantifier-free formula using $=, <$, and constants for the elements of U.

We consider two languages for querying. Queries of the first one are FO formulas of signature L_0^+. We call them restricted. Queries of the second language are FO formulas of the signature L^+. We call them extended.

It is easy to see that the restricted queries are *generic,* that is that they are preserved under order-preserving bijections of U. In other words, if $\phi : U \longleftrightarrow U$ is an automorphism of $\langle U, <, = \rangle$, and a restricted query Q transforms a database state s into a relation R, then Q transforms $\phi(s)$ into $\phi(R)$. The problem with extended queries is, they may not be generic.

We will also use a stronger notion of *locally generic* queries. In the simplest possible case of Boolean queries and finite states, locally generic queries are those which are preserved under partial $<$-isomorphisms of the active domain—as usual, the active domain AD of a finite state is the (finite) set of all the elements of the universe that occur in some tuple in some of the database relations. For non-Boolean queries, this notion can be generalized by requiring preservation under partial isomorphisms of the active domain *together with the elements of the tuple.*

Formally, for a state s and an $R \in SC$, let $AD(s)$ denote the active domain of s, and let $s(R)$ denote the relation assigned to R in s. If a is a tuple of elements of U and s is a state, let $\{AD(s), a\}$ denote the union of $AD(s)$ and the elements of a. If κ is a mapping from $\{AD(s), a\}$ into U and $a = (a_1, \ldots, a_k)$, let $\kappa(a) = (\kappa(a_1), \ldots, \kappa(a_k))$, and let $\kappa(s)$ be the state that assigns $\kappa(s(R))$ to $R \in SC$, where for an m-ary R,

$$\kappa(s(R)) = \{(\kappa(b_1), \ldots, \kappa(b_m)) : (b_1, \ldots, b_m) \in s(R)\}.$$

A query $\phi(\bar{x})$ is *locally generic* over finite states over U iff for every finite database state s, every tuple t, and for every partial $<$-isomorphism κ from $\{AD(s), t\}$ into U,

$$s \models \phi(t) \Longleftrightarrow \kappa(s) \models \phi(\kappa(t)).$$

Finally, the notion of local genericity can be generalized to finitely representable states by appropriately defining the notion of active domain of such a state. The definition will be given in Section 4. We will show that, under our definition, the active domain of a finitely representable d.b. state always is finite, and that there always exists a canonical finite representation in which all constants are taken from the active domain. Then the definition of local genericity for finite states is literally applied to f.r. states with the understanding that $\kappa(s)$ is this canonical finite representation of s with $\kappa(c)$ substituted for c, for every constant $c \in AD(s)$.

Of course, every locally generic query is generic. Conversely, for several domains, every generic query is locally generic. A sufficient condition for this is the so-called *double transitivity* of the domain: a domain is called doubly transitive iff for any $a_1 < b_1, a_2 < b_2$ in the domain, there exists an automorphism of the domain mapping a_1, b_1 to a_2, b_2, respectively. For instance, real and rational numbers are doubly transitive, while integer numbers are not.

3 Impossibility of translation over all states

The goal of this section is to compare restricted and generic extended queries from the viewpoint of their expressive power over *all possible states,* whether finitely representable or not. We show that, generally speaking, extended querying is more expressive than restricted, even in very simple situations. Let $\mathbf{Q}$ be the universe of rational numbers with the usual order relation, and consider a ternary relation $+$ of rational addition as part of the extended language.

Theorem 3.1 *Finiteness of database states is expressible by an extended query but is not expressible by any restricted query.*

PROOF: The fact that the query is generic is obvious. To express it, first recall that the active domain AD of the database state (that is, the elements present in the database state) is expressible by an FO query. Let $AD(x)$ be this query. The d.b. state is finite iff AD is finite. Observe now that a set of rational numbers is finite iff the following two conditions hold:

1. the set fully belongs to an interval, and
2. there is a rational quantity $r > 0$ such that any two distinct elements in the set are at least r apart from one another.

Therefore, we can express this query as:

$$(\exists ulr > 0)((\forall x)(AD(x) \longrightarrow (l < x \wedge x < u))\wedge$$

$$(\forall xy)((AD(x) \wedge AD(y) \wedge x < y) \longrightarrow x + r < y)).$$

Here, the "$x + r < y$" is an obvious abbreviation.

Of course the boundedness of AD is expressed in the first part of the formula without $+$; we claim, however that the constant-separation is not expressible without $+$.

Formally, we claim that if a Boolean restricted query holds in every finite database state, it also holds in an infinite state that belongs to an interval and consists of isolated points. A formal proof of this can be carried out for a database scheme consisting of a single unary relation by applying a Fraïssé-Ehrenfeucht game. *Q.E.D.*

Note that the technique in this theorem is **Q**-specific, and cannot be carried out for, say, real numbers.

Corollary 3.2 *Extended querying is more expressible than restricted w.r.t. Boolean generic queries over* $\langle \mathbf{Q}, <, + \rangle$.

Of course, the examples of this sort are many. To begin with, in the standard model **N** of Arithmetic generic extended queries—which use $+$ and $*$—can express more than order queries. In **N**, and in other such examples, however, this remains true even if we consider finite database states only.

In $\langle \mathbf{Q}, <, + \rangle$, the situation is otherwise. We will see that, over finite or finitely representable queries, every generic extended query is equivalent to a restricted query, and moreover, this equivalence is recursive. Thus, our example really draws a line between finite and finitely representable states on one side, and essentially infinite states, on the other.

4 Finite and finitely representable d.b. states

The goal of this section is to show that finite and finitely representable states can be treated uniformly. To achieve this goal, we show that finitely representable relations can be represented uniformly by finite relations, of a different signature, such that these f.r. states and their finite "codes" can be mapped to each other by restricted queries.

For example, take the f.r. binary relation represented by the quantifier-free formula:

$$x_1 < 3 \wedge x_2 > x_1.$$

This relation contains infinitely many tuples, including $(1, 2)$, $(2, 17)$, etc. To reconstruct this relation, it suffices to know:

- all the constants used in the representation—in our case, just 3—and
- at least one representative of each "equivalence" class of tuples w.r.t. the constants, present in the relation

For example, to reconstruct this relation, it suffices to know that 3 is the only constant, and that:

- $(1, 1)$ is not in the relation
- $(1, 2)$ is in the relation

- $(1, 3)$ is in the relation
- $(3, 3)$ is not in the relation
- $(3, 4)$ is not in the relation
- $(4, 4)$ is not in the relation
- $(4, 5)$ is not in the relation
- $(2, 1)$ is not in the relation
- $(3, 1)$ is not in the relation
- $(4, 3)$ is not in the relation
- $(5, 4)$ is not in the relation

For example, how do we go about deciding whether $(1.5, 2.7)$ is in the relation? Well, clearly, $(1.5, 2.7)$ is in iff $(1, 2)$ is in. This is because the two tuples are indistinguishable by $<$ applied to either their coordinates, or to a coordinate and a constant used in the definition (3). This phenomenon is formally captured by our definition of *invariance:* our relation is 3-invariant. Generally, we will see that any f.r. relation is invariant w.r.t. the set of its "essential" constants.

Hence, first, we would want to come up with a restricted query expressing the set of all constants used in the finite representation—in our case, used in $x_1 < 3 \wedge x_2 > x_1$. The trouble here is, our relation can alternatively be expressed as

$$x_1 < 3 \wedge x_2 > x_1 \vee x_1 = 1 \wedge x_2 = 2.$$

Clearly, we are not going to be able to recover 1 or 2, since they are shadowed by the first conjunct. That is why, in defining the notion of the active domain, we referred to those constants "present in *every* finite representation of the relation". 3 is such a constant, while 1 and 2 are not. This notion of the "essential" constant is captured below by the definition of a *boundary point* of a relation. The fact that 3 is going to be present in any definition of our relation is clear.

This done, we will proceed to expressing the (finite!) set of tuple-representatives. Again, this idea cannot be implemented literally. The reason for this is, there is no way to include $(1, 1)$ as a representative, and to not include $(2, 2)$ at the same time. This turns out to be a minor, though messy, obstacle. Briefly, we well define a bunch of relations on the active domain to describe which tuples belong to the original relation. Here we will use the invariance.

Of course, our finite representation is specific in that it has a fixed ordering of the coordinates—we call such relations *prime.* We show however that every finitely representable relation can be broken down to prime relations.

To carry out these ideas, we will need a few technical definitions, which are somewhat more complex than the intuition above seems to require, because, in fact, we are doing arbitrary ordered sets, whereas the above explanation is only good for dense orders.

An expression of one of the forms $x < c$, $x > c$, or $x = c$ where x is a variable and c is a name of an element of U, is called simple. A relation R is said to be *simple* if it can be finitely represented as a conjunction of disjunctions of simple formulas. R is said to be *prime* if it is representable in the form

$$\bigwedge_{i<j} x_i < x_j \wedge \gamma,$$

where γ represents a simple relation. A finitely-representable database state is said to be simple (prime) if every database relation in this state is simple (prime).

Given a prime relation P of arity k, we say that two elements r_1, r_2 of U are *P-separable* iff there exists a position j in the relation P such that:

$$(\exists x_1 x_2 \ldots x_{j-1} x_{j+1} \ldots x_k) \quad \left(\bigwedge_{i=1}^{j-2} x_i < x_{i+1} \wedge \bigwedge_{i=j+1}^{k-1} x_i < x_{i+1} \wedge \right.$$
$$x_{j-1} < r_1 < x_{j+1} \quad \wedge \quad x_{j-1} < r_2 < x_{j+1} \wedge$$
$$\left. (P(x_1 x_2 \ldots x_{j-1} r_1 x_{j+1} \ldots x_k) \;\not\equiv\; P(x_1 x_2 \ldots x_{j-1} r_2 x_{j+1} \ldots x_k)) \right).$$

Hence, this relation of separability is expressible as a restricted query.

An element x of U is said to be a *left* (respectively, *right*) *boundary point* of a prime relation P iff for each $y \in U$, $y > x$ (respectively, $y < x$) there exist P-separable $u, v \in U$ such that $x \leq u \leq y$, $x \leq v \leq y$ (respectively, $x \geq u \geq y$, $x \geq v \geq y$). The maximal (minimal) element of U is a *boundary point* of P iff it is a right (respectively, left) boundary point of P. An intermediate element of U is a *boundary point* of P iff it is a right or left boundary point of P. Clearly, the set of boundary points of any prime relation P can be expressed as a restricted query. Denote this query by $\Gamma_P(x)$.

An interval $[x, y]$ is a set $\{z : x \leq z \leq y\}$.

Lemma 4.1 *Fix a finite representation of P. If an interval $[x, y]$ does not contain a constant in the representation and $x < y$, then the interval contains no pair of P-separable elements.*

PROOF: Let $u, v \in [x, y]$. Replace u with v in γ. The replacement cannot change the truth of $u < c$, $u > c$, or $u = c$ for any constant c in the representation. So the replacement cannot change the truth of γ. Thus the interval contains no pair of P-separable points. *Q.E.D.*

Lemma 4.2 *Fix a finite representation of P. Any boundary point of P is a constant in the representation or is adjacent to such a constant.*

PROOF: Suppose x is a boundary point of P and is not an endpoint in U (the other case is similar). If x were neither a constant nor an element adjacent to a constant in the representation, there would exist y and z such that $y < x < z$ and the intervals $[y, x]$ and $[x, z]$ both would contain no constants in the representation. Thus the intervals would contain no pair of P-separable points (Lemma 4.1), and x would not be a boundary point of P. *Q.E.D.*

So, the set of the boundary points for each prime relation P is finite.

Every quantifier-free formula representing a relation P of an arity k can always be represented as a disjunction of the following form:

$$\bigvee_{\rho \in R_k} (\rho \wedge \delta_\rho),$$

where R_k is a collection of all the maximal conjunctions of equalities and comparisons between $x_1, x_2, \ldots, x_k$, and δ_ρ is a quantifier-free formula in which inter-variable comparisons or equalities are not used. Then for every $\rho \in R_k$, $\rho \wedge P$ can easily be encoded by a prime relation (of possibly a smaller arity). The union of boundary points of the prime relations $\rho \wedge P$ for all $\rho \in R_k$ is the set of boundary points of P.

For a finitely representable state, *its active domain* is the union of the sets of boundary points of all the database relations of the state.

Let R be a set of elements of U. A relation P is said to be *R-invariant* iff:

$$(\forall \vec{x}\vec{y})$$

$$\left((\forall z \in R) \left(\bigwedge_i (x_i > z \equiv y_i > z \wedge x_i < z \equiv y_i < z) \right) \longrightarrow (P(\vec{x}) \equiv P(\vec{y})) \right).$$

In other words, if two tuples behave the same way with respect to R, they belong to P simultaneously. The following lemma is obvious.

Lemma 4.3 *A relation P is simple iff there exists a finite set R of elements of U such that P is R-invariant.*

The following lemma is central in our consideration.

Lemma 4.4 *Let P be a prime relation. If r, q are P-separable and $r < q$, then the interval $[r, q]$ contains a boundary point of P.*

PROOF: Suppose r and q both are not endpoints in U (the other cases are similar). Fix a finite representation of P. Consider the sequence $c_1 < \ldots < c_k$ of all the constants in the representation that belong to the interval $[r, q]$. Suppose for a moment that none of the constants $c_i (i = 1, \ldots, k)$ is a boundary point. So for each constant c_i there is such an interval $[u_i, v_i]$ ($c_{i-1} \le u_i$ and $v_i \le c_{i+1}$) that $u_i < c_i < v_i$ and $[u_i, v_i]$ contains no P-separable points. Thus r is not P-separated from c_1, c_1 is not P-separated from $c_2, \ldots$, and c_k is not P-separated from q. Hence, r is not P-separated from q. *Q.E.D.*

Let γ be a formula, P be a relation symbol, and let δ be a formula of the same arity as P. By $\gamma\left[\frac{P}{\delta}\right]$ we denote the result of substitution of δ for all occurrences of P in γ. As usual, this may require renaming of variables.

The following lemma reduces arbitrary finitely representable states to simple, which will be dealt with later.

Lemma 4.5 *For any database scheme σ there exist another database scheme κ, $\sigma \bigcap \kappa = \emptyset$, and* FO *formulas $\{\delta_P\}_{P\in\sigma}$ of the scheme κ, and $\{\delta_P\}_{P\in\kappa}$ of the scheme σ, δ_P of the arity of P, such that for any finitely representable σ-state for any $P \in \sigma$:*

1. *δ_P yields a simple answer*
2. *for any $P \in \sigma$, $(\forall \vec{x}) \left(\delta_P \left[\frac{Q}{\delta_Q}\right]_{Q\in\kappa}(\vec{x}) \equiv P(\vec{x})\right)$ holds in every finitely representable state.*

PROOF: It suffices to prove the lemma for prime database states only. Indeed, every quantifier-free formula representing a relation P of arity k can always be represented as a disjunction of the following form:

$$\bigvee_{\rho\in R_k} (\rho \wedge \delta_\rho),$$

where R_k is a collection of all the maximal conjunctions of equalities and comparisons between $x_1, x_2, \ldots, x_k$, and δ_ρ is a quantifier-free formula in which inter-variable comparisons or equalities are not used. Then for every $\rho \in R_k$, $\rho \wedge P$ can easily be encoded by a prime relation (of possibly a smaller arity).

Assume henceforth that all the relations are prime. For every $P \in \sigma$, include the relation symbol P_* of the same arity k into κ. Define

$$\delta_{P_*}(x_1, x_2, \ldots, x_k)$$

as follows (recall that $\Gamma_P(z)$ is a query expressing the set of boundary points of P):

$$(\exists y_1 y_2 \ldots y_k)((\forall z)\left(\Gamma_P(z) \longrightarrow \bigwedge_{i=1}^{k}((x_i > z \equiv y_i > z) \wedge (x_i < z \equiv y_i < z))\right) \\ \wedge P(y_1, y_2, \ldots, y_k)).$$

Essentially, $\delta_{P_*}(x_1, x_2, \ldots, x_k)$ extends $P(x_1, x_2, \ldots, x_k)$ to a relation invariant w.r.t. the set of boundary points of P. Denote $x_1 < x_2 < \ldots < x_k$ by ρ_P. Obviously, answers to $\delta_{P_*}(x_1, x_2, \ldots, x_k)$ will not necessarily satisfy ρ_P.

Since the set of boundary points of a prime relation is finite, and

$$\delta_{P_*}(x_1, x_2, \ldots, x_k)$$

yields an invariant relation w.r.t. the set of boundary points of P, this answer is simple (Lemma 4.3).

Notice that $\rho_P \wedge \delta_{P_*}(x_1, x_2, \ldots, x_k)$ is equivalent to $P(x_1, x_2, \ldots, x_k)$. Clearly, P implies $\rho_P \wedge \delta_{P_*}$. Let $\rho_P \wedge \delta_{P_*}(a_1, a_2, \ldots, a_k)$ and $\neg P(a_1, a_2, \ldots, a_k)$. Then there exist $b_1, b_2, \ldots, b_k$, and $i \leq k$ such that:

1. $a_1, a_2, \ldots, a_k$ and $b_1, b_2, \ldots, b_k$ are positioned the same way w.r.t. the set of boundary points of P
2. for all $j = 1, 2, \ldots, k-1$, $a_j < a_{j+1}$ and $b_j < b_{j+1}$
3. $P(b_1, b_2, \ldots, b_k)$, but $\neg P(a_1, a_2, \ldots, a_k)$
4. $a_1 = b_1, \ldots, a_{i-1} = b_{i-1}$, and $a_i < b_i$ (the case $a_i > b_i$ is similar)

If a_i, b_i were P-separable then by Lemma 4.4 there would be a boundary point between a_i and b_i—a contradiction. Otherwise $a_1, a_2, \ldots, a_k$ and $a_1, a_2, \ldots, a_i,\ b_{i+1}, \ldots, b_k$ satisfy the same four conditions as above for $i+1$.

Hence, we can define δ_P as $\rho \wedge P_*$. This proves the lemma. *Q.E.D.*

We also proved the following lemma.

Lemma 4.6 *For any f.r.relation P, there exists a finite representation of P such that any constant in the representation is a boundary point of P.*

A finite representation of P is called *canonical* iff any constant in the representation is a boundary point of P.

A query $\phi(\bar{x})$ is said to be *locally generic* over finitely representable states over U iff for ever finitely representable database state s, for any tuple t, and for any partial $<$-isomorphism κ from $\{AD(s), t\}$ into U,

$$s \models \phi(t) \iff \kappa(s) \models \phi(\kappa(t)),$$

where $\kappa(s)$ is the canonical finite representation of s with $\kappa(c)$ substituted for c, for every constant $c \in AD(s)$.

For a simple relation P of an arity k, define the relation $x \not\approx_P y$ of P-separability as follows:

$$(\exists i < k)(\exists x_1 x_2 \ldots x_{i-1} x_{i+1} \ldots x_k)$$

$$(P(x_1, x_2, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k) \not\equiv P(x_1, x_2, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_k)).$$

The following lemma completes our reduction of f.r. states to finite states:

Lemma 4.7 *For any database scheme κ there exist another database scheme η, $\kappa \bigcap \eta = \emptyset$, and* FO *formulas $\{\delta_P\}_{P\in\kappa}$ of the scheme η, and $\{\delta_P\}_{P\in\eta}$ of the scheme κ, δ_P of the arity of P, such that for any simple κ-state for any $P \in \kappa$:*

1. *δ_P yields a finite answer*
2. *for any $P \in \kappa$, $(\forall \vec{x}) \left(\delta_P \left[\frac{Q}{\delta_Q}\right]_{Q\in\eta} (\vec{x}) \equiv P(\vec{x}) \right)$ holds for every simple state*

PROOF: Take any relation symbol $P \in \kappa$. By assumption, a finite representation for P is a quantifier-free first order formula whose atomic formulas are either equalities or comparisons between variables on one side and constants on the other. Consider the (finite!) set B of boundary points of P. The formula $\Theta(x)$ that would express this set is easy to write using $\not\simeq_P$. Clearly, P is B-invariant, and therefore, the behavior of P can be explained in terms of B. Formally, we will introduce 3^k new relation symbols $\{P_g\}$ of the same arity as P indexed with sequences of $\{<,>,=\}$ of length k. These new relations are going to be defined as subset of B^k, and will therefore be finite.

$<$ in ith position in the index indicates that the boundary point in this position the entire interval of inseparable elements immediately preceding the point. Similarly, $>$ indicates the interval following this point, and $=$ indicates the point itself. For example, for a ternary P, $\delta_{P_{=<=}}(x, y, z)$ can be written as follows:

$$\Theta(x) \wedge \Theta(y) \wedge \Theta(z) \wedge$$

$$(\forall u)\left((u < y \wedge (\forall v)(\Theta(v) \wedge v < y \longrightarrow v < u)\right) \longrightarrow P(x, u, z)) .$$

In the opposite direction, P can be reconstructed using all $\{P_g\}$ and their active domain. Technical details are left to the reader. *Q.E.D.*

The following main theorem follows from Lemmas 4.5, 4.7:

Theorem 4.8 *For any database scheme σ there exist another database scheme η, $\sigma \bigcap \eta = \emptyset$, and* FO *formulas $\{\delta_P\}_{P\in\sigma}$ of the scheme η, and $\{\delta_P\}_{P\in\eta}$ of the scheme σ, δ_P of the arity of P, such that for any finitely representable σ-state for any $P \in \eta$:*

1. *δ_P yields a finite answer*
2. *for any $P \in \sigma$, $(\forall \vec{x})\left(\delta_P\left[\frac{Q}{\delta_Q}\right]_{Q\in\eta}(\vec{x}) \equiv P(\vec{x})\right)$ holds for every finitely representable state*

5 Recursive translation

For the remainder of the section, we fix a divisible ordered Abelian group $\langle \mathbf{G}, +, < \rangle$. The reader, however, can continue to think in terms of $\langle \mathbf{Q}, +, < \rangle$. Within this section, we will use an alternative, more traditional, notation FO_+ and FO for extended and restricted queries, respectively.

Within this section, queries generic over all finite states will be referred to as generic.

We will need a few technical definitions. A *linear combination* is an expression of the form:

$$\sum_{i=1}^{n} \alpha_i \zeta_i,$$

where $\{\zeta_1, \zeta_2, \ldots, \zeta_n\}$ is a list of distinct variables, and $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is a list of rational constants. The *rank* of this linear combination is defined as:

$$D + D \sum_{i=1}^{n} |\alpha_i|,$$

where D is the smallest common denominator of $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. Intuitively, the notion of rank reflects the size of the shortest representation of the linear combination in question.

Let $S = \langle \beta_1, \beta_2, \ldots, \beta_m \rangle$ be a finite sequence of elements of $\mathbf{G}$, and let $0 < \beta_i < \beta_{i+1}$, for all i. For a natural number B, S is said to be *B-wide* iff for every i, $0 < i \leq m$, all linear combinations of $\{\beta_1, \beta_2, \ldots, \beta_{i-1}\}$ of rank at most B are less than β_i. In particular, it implies that no linear combination of rank $\leq B$ of elements of S with non-zero coefficients equals 0.

Now let $S = \langle \beta_1, \beta_2, \ldots, \beta_m \rangle$ be a B^4-wide set of elements of $\mathbf{G}$, and let γ be an element of $\mathbf{G}$. γ is said to be *B-linear in S*, iff γ is a linear combination of elements of S of rank at most B. Notice that such a linear combination, if exists, is unique, because the set is B^4-wide. In this case, this linear combination is called *the exact B-distance* from S to γ. If such a linear combination does not exist, then *the approximate B-distance* from S to γ is one of the following:

1. $-B\beta_m$, if $\gamma < -B\beta_m$
2. $B\beta_m$, if $\gamma > B\beta_m$
3. the pair s_1, s_2 of linear combinations (of rank $\leq B$) of elements of S such that s_1 is the maximal linear combination less than γ, and s_2 is the minimal linear combination greater than γ

The B-distance from S to γ is either the exact B-distance, or the approximate B-distance. Notice that the B-distance is always uniquely defined.

For a given finite database state, its *active domain AD* is the set of all the elements of the domain that occur in some tuple in some of the database relations. The active domain of a finite d.b. state s is always finite, and can be expressed by a query that only uses the database relation symbols (and does not use either $<$ or $+$).

Given an FO_+-query, we want to specialize its quantifiers to either the active domain, or to the rest of the domain. So for the rest of this subsection we assume that all quantifiers are specialized in this sense.

Theorem 5.1 *Let ϕ be a generic Boolean FO_+-query. There exists a Boolean FO-query ψ that is equivalent to ϕ over finite database states over $\langle \mathbf{G}, <, = \rangle$. Such a ψ can be effectively constructed by ϕ.*

PROOF: So we assume that query ϕ is in the prenex normal form, and all its quantifiers are specialized. For simplicity assume that the active domain is nonempty. Notice that for every relation symbol P in our database scheme it can be assumed that only active domain variables (ranging over the active domain) are used in atomic P-formulas in ϕ.

Let q be the number of non-active domain quantifiers in ϕ, and let $x_1, x_2, \ldots, x_q$ be the sequence of the non-active domain variables used in the non-active domain quantifiers in the order of their nesting. Because ϕ is generic, it can always be assumed that the active domain is $2^{2^{2(q+1)}}$-wide—this can always be achieved by moving the state to very big positive numbers and making the distances between successive elements big enough. Let $f : \mathbf{N} \longrightarrow \mathbf{N}$, $f(i) = 2^{2^{2i}}$. It is easy to see that $f(i+1) = (f(i))^4$.

We are going to describe a procedure of translation of the formula ϕ to a formula without $+$. The procedure consists of two parts. In the first part, we will go down the non-active domain quantifiers string, starting from the outermost quantifier. Suppose this quantifier is $(\exists x_i)\delta$ where δ is the part of the formula quantified by this quantifier.

We replace this quantifier for x_i with a disjunction where each disjunct assumes that x_i is at a certain distance from the active domain together with values of $x_1, x_2, \ldots, x_{i-1}$. For instance, the cases where x_i is assumed to be at short *exact* distances from the set consist of all linear combinations of rank at most $f(q-i+1)$ where some of the variables ζ_j are replaced with x_j, for some $j < i$, and the remaining are quantified over AD. For example, the following linear combination:

$$\frac{3}{2}\zeta_1 + \zeta_2$$

will generate the following four disjuncts:

$$\left[x_i \Longrightarrow \frac{3}{2}x_1 + x_2\right] \delta \vee$$

$$(\exists \zeta_1) \left[x_i \Longrightarrow \frac{3}{2}\zeta_1 + x_2\right] \delta \vee$$

$$(\exists \zeta_2) \left[x_i \Longrightarrow \frac{3}{2}x_1 + \zeta_2\right] \delta \vee$$

$$(\exists \zeta_1 \zeta_2) \left[x_i \Longrightarrow \frac{3}{2}\zeta_1 + \zeta_2\right] \delta.$$

For example, the disjunct $(\exists \zeta_1) \left[x_i \Longrightarrow \frac{3}{2}\zeta_1 + x_2\right] \delta$ asserts that x_i is the linear combination of x_2 and an element of the active domain which is selected by the quantifier $(\exists \zeta_1)$.

Clearly, there are finitely many such cases. The part of the formula enclosed in square brackets is called a *label,* and indicates that x_i is assumed to be this fixed linear combination. Labels will be used for the second part of our translation.

Similarly, when x_i is assumed to be exactly in between two consecutive linear combinations L_1 and L_2, we consider all possible pairs of linear combinations of rank at most $f(q-i+1)$ and introduce labels of the form $[x_i \Longrightarrow L_1..L_2]$. In this case, additionally, we need to write a formula saying that no linear combination of rank at most $f(q-i+1)$ lies in between L_1 and L_2:

$$(\forall L)(L > L_1 \longrightarrow L \geq L_2).$$

and the considered L is of rank at most $f(q-i+1)$. We can replace this with a big conjunction.

Writing such a disjunction is although messy, but routine. This way, δ is replaced under this transformation with a huge disjunction of conjunctions of formulas (with new quantifiers), each ending with δ. Each of the copies of the δ is then processed further.

The case of the universal quantifier is reducible to the existential case. The routine technical details of the first stage of translation are left to the reader.

The resulting formula is, literally, not equivalent to ϕ, because the quantifiers for $x_1, x_2, \ldots, x_q$ are removed. However, we now have a lot of information about our assumed choices for $x_1, x_2, \ldots, x_q$—via the distances—and during the second stage of our translation we are going to get rid of $x_1, x_2, \ldots, x_q$ altogether.

This second stage is carried out "from-inside-out"—first, we get rid of x_q, then of x_{q-1}, etc.

By induction, at the step i, $i = q, q-1, \ldots, 1$, we assume that the elementary formulas are $P(\vec{y})$ where $\vec{y}$ does not contain any of the $x_1, x_2, \ldots, x_q$, as well as equalities and comparisons between linear combinations of active domain variables and $x_1, x_2, \ldots, x_i$. Further, if j is the maximal number such that x_j is involved in an equality/inequality, then the rank of both the linear combinations is at most $f(q-j) = 2^{2^{2(q-j)}}$. This condition is satisfied initially.

We are going to replace each occurrence of x_i by its distance from $AD \bigcup \{x_1, x_2, \ldots, x_{i-1}\}$, as follows. If we consider an equality of two linear combinations of rank at most $f(q-i)$, transform it into the form:

$$x_i = E,$$

where E is a linear combination not involving x_i. Clearly, the rank of E is at most $f(q-i+1)$.

If the part of the formula where we are translating this equality is labeled with $[x_i \Longrightarrow L]$, replace the equality above with $E = L$. By construction, the rank of L is at most $f(q-i+1)$.

If, on the other hand, the part of the formula where we are translating this equality follows the assumption that x_i was **not** at an exact distance from $AD \bigcup \{x_1, x_2, \ldots, x_{i-1}\}$, the equality can clearly be replaced by **false**.

The case of comparison between two linear combinations can be considered similarly. When we are following an exact distance, the x_i can be got rid of trivially, by substitution.

In other cases, the inequality is either trivially true, or trivially false, or can be strengthened. For example, if we are following a label $[x_i \Longrightarrow L_1..L_2]$, and our comparison is:

$$x_i < E,$$

our comparison can be replaced by

$$L_2 < E \vee L_2 = E.$$

Clearly, the induction hypothesis is again satisfied.

Finally, we arrive at the point when all x_i disappear. Hence, all equalities and comparisons are between the linear combinations of rank at most $f(q)$ of variables quantified over the active domain. Every such linear combination can be replaced by a disjunction of such linear combinations where the variables are not equal to one another, and are placed in the decreasing order. For example, $x_1 + 3x_2 = x_3$ can be replaced with:

$$(x_1 = x_2 \wedge x_1 = x_3 \wedge 4x_1 = x_1) \vee$$

$$(x_1 < x_2 \wedge x_2 < x_3 \wedge 3x_2 + x_1 = x_3) \vee$$

etc...

Observe that all equalities then become false, except if are graphical, that is if all the coefficient are the same at the same elements—observe that we still need simple equalities between two variables to check whether they denote the same element of the domain. Similarly, since short linear combinations between the domain elements can be resolved by comparing the first distinct coefficients coefficients, every comparison can be replaced by a series of simple comparisons and equalities between variables.

This proves the theorem. *Q.E.D.*

Theorem 5.2 *Let $\phi(\vec{x})$ be a generic* FO_+*-query. There exists an* FO*-query $\psi(\vec{x})$ that is equivalent to $\phi(\vec{x})$ over finite database states over $\langle \mathbf{G}, <, = \rangle$. Such a $\psi(\vec{x})$ can be effectively constructed by $\phi(\vec{x})$.*

PROOF: Notice that if two tuples $\vec{r}, \vec{w}$ or elements of $\mathbf{G}$ are positioned the same way in the sense of $<, =$ w.r.t. the active domain, then $\phi(\vec{r})$ and $\phi(\vec{w})$ are equivalent, because the query is generic.

This observation allows us to replace $\phi(\vec{x})$ by a formula where free variables occur only to equalities and comparisons between themselves or variables specialized over the active domain. Observe that the procedure described in our proof of Theorem 5.1 does not change such atomic formulas.

For example, $\phi(u, v)$ is equivalent to the following disjunction:

$$(\phi(u,v) \wedge u < v) \vee (\phi(u,v) \wedge u = v) \vee (\phi(u,v) \wedge v < u).$$

The first disjunct is equivalent to the disjunction of all possible arrangements of u and v with respect to AD. One of the arrangements is:

$$(\exists x_1 x_2 x_3 x_4 \in AD)(x_1 < u < x_2 < x_3 < v < x_4 \wedge$$

$$(\forall x \in AD)((x_1 < x \longrightarrow x_2 \leq x) \wedge (x_3 < x \longrightarrow x_4 \leq x)).$$

We replace the disjunct with the following one:

$$(\exists x_1 x_2 x_3 x_4 \in AD)(\exists y z)(x_1 < u < x_2 < x_3 < v < x_4 \wedge$$

$$x_1 < y < x_2 \wedge x_3 < z < x_4 \wedge$$

$$(\forall x \in AD)((x_1 < x \longrightarrow x_2 \leq x) \wedge (x_3 < x \longrightarrow x_4 \leq x)) \wedge$$

$$\phi(y, z)).$$

Q.E.D.

Finally, using Theorem 4.8, we can expand our results to arbitrary finitely representable states (but not beyond this—see Corollary 3.2).

Theorem 5.3 *Let $\phi(\vec{x})$ be an* FO_{+}*-query generic over all finitely representable states over $\langle \mathbf{G}, <, = \rangle$. There exists an* FO*-query $\psi(\vec{x})$ that is equivalent to $\phi(\vec{x})$ over finitely representable database states over $\langle \mathbf{G}, <, = \rangle$. Such a $\psi(\vec{x})$ can be effectively constructed by $\phi(\vec{x})$.*

PROOF: Using Theorem 4.5, rewrite the query $\phi(\vec{x})$ to a query $\phi'(\vec{x})$ over a new signature η, such that:

- there are formulas $\{\delta_P\}_{P\in\eta}$ over the initial signature σ with the property $\phi(\vec{x}) \iff \phi'\left[\frac{P}{\delta_P}\right]_{P\in\eta}(\vec{x})$, and
- for any $P \in \eta$, δ_P yields a finite answer

By Theorem 5.2, ϕ' is equivalent to a $+$-free formula ψ' over all finite d.b. states. Then, $\psi'\left[\frac{P}{\delta_P}\right]_{P\in\eta}(\vec{x})$ is a $+$-free formula equivalent to $\phi(\vec{x})$. Denote this formula by ψ. Clearly, this ψ can be constructed effectively, which finishes the proof. *Q.E.D.*

6 Non-recursive equivalence

In this section, we use techniques from model theory. The reader is referred to [3] for standard definitions and results.

Fix a non-principal ultra-filter D over the set ω of all natural numbers. Let V be the ultra-power of U by D. An n-ary relation on V is called *pseudo-finite* iff it is a D-ultra-product of a sequence of n-ary finite relations on U. Recall the following:

Łoś Theorem *If elements $a_1, \ldots, a_m$ are ultra-products of*

$$a_1(i)\ (i \in \omega), \ldots, a_m(i)\ (i \in \omega),$$

and relations $P_1, \ldots, P_s$ are ultra-products of

$$P_1(i)\ (i \in \omega), \ldots, P_s(i)\ (i \in \omega)$$

then for any m-ary FO *formula Φ of the signature $\{=, P_1, \ldots, P_s\}$,*

$$(V, P_1, \ldots, P_s, a_1, \ldots, a_m) \models \Phi$$

if and only if

$$\{i \in \omega : (U, P_1(i), \ldots, P_s(i), a_1(i), \ldots, a_m(i)) \models \Phi\} \in D.$$

Consider a structure W of a signature O, $a \in W$ and $A \subseteq W$. A set p of FO formulas over the signature $O \bigcup \{c_b : b \in A\}$ with the only free variable x is said to be *consistent over A in W* iff for every finite sequence $\psi_1, \ldots, \psi_r$ of formulas in p,

$$(W, b : b \in A) \models (\exists x)(\psi_1 \wedge \ldots \wedge \psi_r).$$

A consistent p is said to be *maximal over A in W* iff for no set q of formulas consistent over A in W, $p \subset q$. A maximal consistent set of formulas over A in W is called a *type over A in W*. Let p be a type over A in W, and $q \subseteq p$. We say that *q implies p* iff p is the only type over A in W that contains q.

The *type of a over A in W*, in symbols $\mathrm{tp}(a/A)$, is the set of all the formulas ψ of the signature $O \bigcup \{c_b : b \in A\}$ with one free variable such that:

$$(W, b : b \in A, a) \models \psi.$$

A structure W is called *$\aleph_1$-saturated* iff for every countable subset A of W and every type p over A in W, there is an element a in W such that $p = \mathrm{tp}(a/A)$.

It can be shown by standard methods that V is an $\aleph_1$ -saturated structure of signature L. Moreover, if a state s is the ultra-products of finite states s_n $(n < \omega)$ and a k-tuple a is the ultra-products of k-tuples a_n $(n < \omega)$, the structure (V, s, a) is $\aleph_1$ -saturated as well.

Structures W_1 and W_2 of a signature O are called *elementary equivalent* iff for every formula ψ of the signature O without free variables, $W_1 \models \psi$ iff $W_2 \models \psi$. Any two elementary equivalent $\aleph_1$-saturated structures of the cardinality $\aleph_1$ are isomorphic.

Assuming the Continuum Hypothesis, the cardinality of V is $\aleph_1$.

Theorem 6.1 *An extended query Φ is equivalent to a restricted query over all finite database states over U, iff Φ is generic over all pseudo-finite database states over V.*

PROOF: If over all finite database states over U, Φ is equivalent to a restricted query Ψ, they remain equivalent for all pseudo-finite states over V. But Ψ and all other restricted queries are stable under order automorphisms of V.

Let k be the arity of Φ. Assume that over all finite database states over U, Φ is equivalent to no restricted query. Let ω be the set of all natural numbers, and let $(\phi_n : n < \omega)$ be an enumeration of all k-ary L_0^+-formulas, that is, of all $<$-queries. For every $n < \omega$ there exist tuples a_n and b_n of elements of U, and finite states s_n and r_n over U, which agree on $\phi_0, \ldots, \phi_n$ for a_n and b_n, and which do not agree on Φ for a_n and b_n. In other words, for $j = 0, \ldots, n$,

$$(U, s_n, a_n) \models \phi_j \text{ iff } (U, r_n, b_n) \models \phi_j,$$

but

$$(U, s_n, a_n) \models \Phi \text{ and } (U, r_n, b_n) \models \neg\Phi.$$

Let s, r, a, b be the ultra-products of s_n, r_n, a_n, b_n $(n < \omega)$, respectively. It is obvious that (V, s, a) is elementary equivalent to (V, r, b) over the signature $L_0^+ \bigcup \{c_1, \ldots, c_k\}$ (recall that k is the arity of Φ). Since V is $\aleph_1$-saturated, there is a $<$-automorphism of V transforming a to b and s to r. But Φ is true in (V, s, a) and false in (V, r, b). Hence, Φ is not generic over pseudo-finite states over V. *Q.E.D.*

Let $(D, Q_1, \ldots, Q_r)$ be a structure, and let A, B be subsets of D. A mapping h from A onto B is called *$\{Q_1, \ldots, Q_r\}$-isomorphism* iff for every k-ary FO formula ψ of the signature $\{Q_1, \ldots, Q_r\}$ and every k-tuple $a_1, \ldots, a_k$ of elements of A,

$$(D, Q_1, \ldots, Q_r, a_1, \ldots, a_k) \models \psi$$

is equivalent to

$$(D, Q_1, \ldots, Q_r, h(a_1), \ldots, h(a_k)) \models \psi.$$

For a state s, let $AD(s)$ denote its active domain. If a is a tuple of elements of V and s is a state, let $\{AD(s), a\}$ be the union of the elements of $AD(s)$ and the elements of a.

Theorem 6.2 *Let Φ be a k-ary extended query which is locally generic over all finite states over U, but not generic over pseudo-finite states over V. Then there exist pseudo-finite states s and r over V, k-tuples a and b of elements V, and an L-indiscernible sequence Σ of elements of V such that:*

1. *$\{AD(s), a\}$ and $\{AD(r), b\}$ are subsets of Σ and there is an L_0^+-isomorphism h from $\{AD(s), a\}$ onto $\{AD(r), b\}$*
2. *$(V, s, a) \models \Phi$ and $(V, r, b) \models \neg\Phi$*

PROOF: Since Φ is not generic over pseudo-finite states over V, there exist ultra-products s, r, a, b of, respectively, s_n, r_n, a_n, b_n $(n < \omega)$, and a $<$-automorphism h of V transforming a to b, and s to r, such that Φ is true in (V, s, a) and false in (V, r, b).

By Łoś Theorem, we may assume that for every natural n, Φ is true in (U, s_n, a_n) and false in (U, r_n, b_n). Let $(\theta_n : n < \omega)$ be an enumeration of all L-formulas. For each $n \in \omega$, there is an infinite $\{\theta_1, \ldots, \theta_n\}$-indiscernible sequence Σ_n of elements of U. For any natural number n, choose L_0^+-isomorphisms f_n and g_n from $\{AD(s_n), a_n\}$ and $\{AD(r_n), b_n\}$, respectively, into Σ_n. Let Σ be the ultra-product of Σ_n $(n < \omega)$. By Łoś Theorem, Σ is an L-indiscernible sequence of elements of V. Because of the local genericity of Φ, for every $n \in \omega$, Φ is true in $(U, f_n(s_n), f_n(a_n))$ and false in $(U, g_n(r_n), g_n(b_n))$. Let f be the ultra-product of f_n $(n < \omega)$ and g be the ultra-product of g_n $(n < \omega)$. Then by Łoś Theorem, Φ is true in $(V, f(s), f(a))$ and false in $(V, g(r), g(b))$. Moreover, $\{AD(f(s)), f(a)\}$ and $\{AD(g(r)), g(b)\}$ are subsets of Σ, and ghf^{-1} is an L_0^+-isomorphism from $\{AD(f(s)), f(a)\}$ onto $\{AD(g(r)), g(b)\}$. *Q.E.D.*

As a side remark, note that this technique implies the result from [2] and [13] about the so-called *active semantics*. Consider FO formulas over the signature L^+. Any quantifier-free formula is called active. Let $AD(x)$ be a formula which says that x is an element from the active domain. If ψ is an active formula, so are:

$$(\forall x)(AD(x) \longrightarrow \psi),$$

$$(\exists x)(AD(x) \wedge \psi).$$

An extended query is called active if it can be represented by an active formula.

Theorem 6.3 *Any active extended query is equivalent to a restricted query.*

PROOF: Let Φ be an active representation of the extended query. Consider the mapping h from Theorem 6.2. Since Σ is an L-indiscernible sequence, this h is, in fact, an L-isomorphism from $\{AD(s), a\}$ onto $\{AD(r), b\}$. Using induction in the number of quantifiers in Φ, we can prove that

$$(V, s, a) \models \Phi \text{ iff } (V, r, b) \models \Phi.$$

This contradicts (2) of Theorem 6.2. *Q.E.D.*

Theorem 6.4 *Suppose that for each $a \in V$ and each pseudo-finite subset A of V, there is a countable subset A_0 of A such that* $\mathrm{tp}(a/A_0)$ *implies* $\mathrm{tp}(a/A)$*. Then every extended query which is locally generic over all finite states over U is generic over all pseudo-finite states over V, and is equivalent over all finite database states over U to a restricted query.*

PROOF: Consider the mapping h from Theorem 6.2. Since Σ is an L-indiscernible sequence, this h is, in fact, an L-isomorphism from $\{AD(s), a\}$ onto $\{AD(r), b\}$. We are going to show that h is actually an L^+-isomorphism. This contradicts (2) of Theorem 6.2.

The idea is to show a winning strategy in the Fraïssé-Ehrenfeucht game for ω steps. The only non-trivial part is Lemma 6.5. *Q.E.D.*

Lemma 6.5 *Let A and B be pseudo-finite subsets of V, and let $g : A \to B$ be an L-isomorphism from A onto B. Then, for any $a \in V$, there is a $b \in V$ such that $g \bigcup \{(a, b)\}$ is an L-isomorphism from $A \bigcup \{a\}$ onto $B \bigcup \{b\}$.*

PROOF: Let A_0 be a countable subset of A such that $\mathrm{tp}(a/A_0)$ implies $\mathrm{tp}(a/A)$. For each formula ψ of the signature $L \bigcup \{c_b : b \in A\}$, denote by $g(\psi)$ the formula obtained from ψ by replacing, for all $b \in A$, c_b with $c_{g(b)}$. For each $B \subseteq A$, let $g(\mathrm{tp}(a/B))$ be

$$\{g(\psi) : \psi \in \mathrm{tp}(a/B)\}.$$

Since g is an L-isomorphism, $g(\mathrm{tp}(a/A_0))$ is a type over $g(A_0)$, $g(\mathrm{tp}(a/A))$ is a type over $g(A)$, and $g(\mathrm{tp}(a/A_0))$ implies $g(\mathrm{tp}(a/A))$. It follows from the $\aleph_1$-saturation of V that there is an element b in V such that

$$\mathrm{tp}(b/g(A_0)) = g(\mathrm{tp}(a/A_0)).$$

Hence,

$$\mathrm{tp}(b/g(A)) = g(\mathrm{tp}(a/A)).$$

Q.E.D.

The following two theorems exemplify applications of this technique.

Let $\chi(x, \bar{y})$ be an L-formula. The notation means that $x, \bar{y}$ is the list of the free variables of the formula χ of the signature L. For any fixed sequence $\bar{c}$ of elements of a structure W of the signature L, the set

$$\{u \in W : (W, u, \bar{c}) \models \chi\},$$

is called definable with the parameters $\bar{c}$ in W. For a subset A of W, a set B is *A-definable in W* iff there is a sequence $\bar{c}$ of elements of A such that B is definable with the parameters $\bar{c}$ in W. An element a is A-definable iff $\{a\}$ is A-definable.

A set of one of the forms:

$$\{x : x < a\}, \{x : x \leq a\}, \{x : a < x\}, \{x : a \leq x\},$$

or

$$\{x : b < x < a\}, \{x : b < x \leq a\}, \{x : b \leq x < a\}, \{x : b \leq x \leq a\},$$

where a and b are constants is called an *interval.* U is called *o-minimal* if every subset of U which is definable with parameters in U is a finite union of intervals.

Theorem 6.6 *If U is o-minimal, every locally generic over all finite database states over U extended query is generic over all pseudo-finite database states over V, and is equivalent to a restricted query over all finite database states over U.*

PROOF: Follows from Lemma 6.7. *Q.E.D.*

Lemma 6.7 *Let U be an o-minimal structure, A be a pseudo-finite subset of V, and let a be an element of V. There is a countable $A_0 \subseteq A$ such that* $\mathrm{tp}(a/A_0)$ *implies* $\mathrm{tp}(a/A)$.

PROOF: First, note that V also is o-minimal. Indeed, every structure which is elementary equivalent to an o-minimal structure is o-minimal too (see [15], [12], and [16]).

It is easy to see that, for each set P definable in an o-minimal structure with parameters $\bar{c}$, every boundary point of P is definable in the structure with the parameters $\bar{c}$.

Let F be a finite set in U, $d \in U$, and let $\chi(x, \bar{y})$ be an L-formula. Then among the boundary points of the sets of the form

$$\{u \in U : (U, u, \bar{c}) \models \chi \wedge u \leq d\},$$

where $\bar{c}$ are chosen in F, there is the greatest one. Similarly, among the boundary points of the sets of the form

$$\{u \in U : (U, u, \bar{c}) \models \chi \wedge d \leq u\},$$

where $\bar{c}$ are chosen in F, there is the least one. By Łoś Theorem, the same continues to be true for pseudo-finite sets F in V, and for $d \in V$, provided the sets have any boundary points. Denote the two elements, if they exist in V, by $m(F, d, \chi)$ and $M(F, d, \chi)$.

If a is A-definable, the result is obvious. Suppose it is not the case. Then, for every $\chi(x, \bar{c})$ in $\mathrm{tp}(a/A)$, a is an internal point of

$$\chi(V, \bar{c}) = \{b \in V : (V, b, \bar{c}) \models \chi\}.$$

Hence $\chi(V, \bar{c})$ contains the interval $\{u \in V : m(A, a, \chi) < u < M(A, a, \chi)\}$, if both the elements are defined, or otherwise $\{u \in V : u < M(A, a, \chi)\}$, if $m(A, a, \chi)$ is not defined, or $\{u \in V : m(A, a, \chi) < u\}$, if $M(A, a, \chi)$ is not defined—the case when they are both undefined is trivial. Since the set of all L-formulas is countable, there is a countable $A_0 \subseteq A$ such that all defined $m(A, a, \chi)$, $M(A, a, \chi)$ are A_0-definable, and we are done. *Q.E.D.*

Theorem 6.8 *If U is the additive group of integer numbers or the additive semigroup of natural numbers, every locally generic over all finite states over U extended query is generic over all pseudo-finite database states over V, and, therefore, is equivalent for all finite database states over U to a restricted query.*

PROOF: Follows from Lemma 6.9. *Q.E.D.*

Lemma 6.9 *Let U be the additive group of integer numbers or the additive semigroup of natural numbers. Let A be a pseudo-finite subset of V and $a \in V$. There is a countable $A_0 \subseteq A$ such that* $\mathrm{tp}(a/A_0)$ *implies* $\mathrm{tp}(a/A)$.

PROOF: If a is A-definable, the result is obvious. Suppose it is not the case. It is easy to see that there exists a subset q of $\mathrm{tp}(a/A)$ such that q implies $\mathrm{tp}(a/A)$ and any formula of q is equivalent to a formula of the form

$$\psi \wedge \theta(i_1, j_1) \wedge \ldots \wedge \theta(i_r, j_r),$$

for some natural numbers $i_1 > 0, j_1, \ldots, i_r > 0, j_r$ where ψ is $c_1 < x < c_2$, $c_1 < x$ or $x < c_2$ with A-definable constants c_1 and c_2, and $\theta(i, j)$ is $(\exists y) x = i * y + j$. Let us use the following abbreviations: $1 * y$ for y; $(i+1) * y$ for $i * y + y$.

Now, the proof of Lemma 6.7 can be carried out with no changes.
Q.E.D.

Our proofs of Theorems 6.6 and 6.8 use the Continuum Hypothesis. However, the theorems are Σ^1_2 Arithmetical statements. By Shoenfield's Absoluteness Theorem (see [8], Theorem 98), they are absolute. Thus, it is possible to construct proofs of our theorems which do not use the Continuum Hypothesis.

Finally, similarly to the case of recursive translation, all our results for finite states can be lifted to finitely representable states using Theorem 4.8.

7 Conclusion

Further work needs to be done for integer numbers. For instance, the authors are under impression that an effective translation algorithm for locally generic queries over $(\mathbf{Z}, +)$ can be extracted from a Fraïssé-Ehrenfeucht-style proof of decidability of Presburger Arithmetic. Then, how much higher than $+$ in $\mathbf{Z}$ can we go? Obviously, $+, *$ make locally generic extended queries impossible to express as pure order, by diagonalization. We **conjecture** that if, in an extension of $(\mathbf{Z}, +)$, locally generic extended queries express more than restricted, the first order theory of the domain is undecidable. Stolboushkin and Taitslin [17] discussed irrelevance of undecidable domains to databases.

Acknowledgments

We are grateful to late Paris Kanellakis, and also to Jianwen Su, and Jan Van den Bussche for helpful comments and suggestions.

References

[1] A.K. Ailamazian, M.M. Gilula, A.P. Stolboushkin, and G.F. Schwarz. Reduction of the relational model with infinite domains to the case of finite domains. *Doklady Akademii nauk SSSR*, 286(2):308–311, 1986. Russian.

[2] M. Benedikt, G. Dong, L. Libkin, and L. Wong. Relational expressive power of constraint query languages. Manuscript, 1995.

[3] C.C. Chang and H.J. Keisler. *Model Theory.* North Holland, 3rd edition, 1990.

[4] M.M. Gilula, A.P. Stolboushkin, G.F. Schwarz, and K.V. Shvachko. Some algorithmic problems in the theory of relational databases. In A.K. Ailamazian, editor, *Problem-Oriented Computational Systems*, volume 125 of *Problems in Cybernetics*, pages 81–91. USSR Academy of Sciences, Moscow, USSR, 1986. Russian.

[5] S. Grumbach and J. Su. Finitely representable databases. In *Proc. 13th ACM Symp. on Principles of Database Systems*, 1994.

[6] S. Grumbach and J. Su. Dense-order constraint databases. In *Proc. 14th ACM Symp. on Principles of Database Systems*, pages 66–77, 1995.

[7] S. Grumbach, J. Su, and C. Tollu. Linear constraint databases. In *Proc. Logic and Computational Complexity (LCC'94), Springer-Verlag Lecture Notes in Computer Science*, 1995. To appear.

[8] T. Jech. *Set Theory.* Academic Press, 1978.

[9] P.C. Kanellakis and D.Q. Goldin. Constraint programming and database query languages. In *Proc. International Symposium on Theoretical Aspects of Computer Software (TACS'94)*, pages 96–120, 1994.

[10] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. In *Proc. 9th ACM Symp. on Principles of Database Systems*, pages 299–313, 1990.

[11] P.C. Kanellakis, G.M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, August 1995.

[12] J.F. Knight, A. Pillay, and C. Steinhorn. Definable sets in ordered structures, II. *Transactions of American Mathematical Society*, 295(2):593–605, 1986.

[13] M. Otto and J. Van den Bussche. First-order queries on databases embedded in an infinite structure. Manuscript, 1995.

[14] J. Paradaens, J. Van den Bussche, and D. Van Gucht. First-order queries on finite structures over reals. In *Proc. 10th IEEE Symp. on Logic in Computer Science*, pages 79–87. IEEE Computer Society Press, 1995.

[15] A. Pillay and C. Steinhorn. Definable sets in ordered structures, I. *Transactions of American Mathematical Society*, 295(2):565–592, 1986.

[16] A. Pillay and C. Steinhorn. Definable sets in ordered structures, III. *Transactions of American Mathematical Society*, 309(2):469–476, 1988.

[17] A.P. Stolboushkin and M.A. Taitslin. Finite queries do not have effective syntax. In *Proc. 14th ACM Symp. on Principles of Database Systems*, pages 277–285, 1995.