Relational Queries over Interpreted Structures

Michael Benedikt
Bell Laboratories
263 Shuman Blvd
Naperville, IL 60566-7050, USA
Email: benedikt@bell-labs.com

Leonid Libkin*
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974, USA
Email: libkin@bell-labs.com

Abstract

We rework parts of the classical relational theory when the underlying domain is a structure with some interpreted operations that can be used in queries. We identify parts of the classical theory that go through 'as before' when interpreted structure is present, parts that go through only for classes of nicely-behaved structures, and parts that only arise in the interpreted case. The first category includes a number of results on language equivalence and expressive power characterizations for the active-domain semantics for a variety of logics. Under this semantics, quantifiers range over elements of a relational database. The main kind of results we prove here are *generic collapse* results: for generic queries, adding operations beyond order, does not give us extra power.

The second category includes results on the natural semantics, under which quantifiers range over the entire interpreted structure. We prove, for a variety of structures, *natural-active collapse* results, showing that using unrestricted quantification does not give us any extra power. Moreover, for a variety of structures, including the real field, we give a set of algorithms for eliminating unbounded quantifications in favor of bounded ones. Furthermore, we extend these collapse results to a new class of higher-order logics that mix unbounded and bounded quantification. We give a set of normal forms for these logics, under special conditions on the interpreted structures. As a by-product, we obtain an elementary proof of the fact that parity test is not definable in the relational calculus with polynomial inequality constraints. We also give examples of structures with nice model-theoretic properties over which the natural-active collapse fails.

1 Introduction

We would like to start with an example that can be found in most database textbooks. When relational algebra is introduced, the conditions in selection operators are defined to be boolean combinations of $x = y$ and $x < y$, where x and y are variables or constants. Typically, a few examples of programming in relational algebra are given before a relational language, such as SQL or QUEL, is introduced. Soon after that we will probably see an example of query like "Select employees who make at least 90% of their manager's salary". Such an example is likely to be followed by a remark that this query, strictly speaking, is not definable in relational algebra because it involves arithmetic operations, but it is definable in SQL or QUEL which allow arithmetic and comparisons of the form $x > 0.9 \cdot y$.

* Contact author. Address: Bell Laboratories, Room 2C-407, 600 Mountain Avenue, Murray Hill, NJ 07974, USA. Phone: (908)582-7647. Fax: (908)582-5857. Email: libkin@research.bell-labs.com.

Does it mean that the relational algebra is in some sense inadequate as a basic relational query language? It is certainly so if one has to deal with arithmetic, but database textbooks counter this criticism by saying that an extension to arithmetic is straightforward. It is indeed true that extending notations is straightforward, but it is not completely obvious how to extend the fundamental results and techniques of relational theory to include arithmetic. For example, if arithmetic constraints are allowed in selection predicates, how would one prove the classical result that parity test and transitive closure are not definable in relational algebra? It appears that the standard techniques are not immediately applicable. Some techniques (such as 0-1 laws) are not applicable at all, others are extremely awkward to apply. This situation extends beyond arithmetic to other interpreted operations over the items that could be stored in a database.

As noted in [26], the full scope of interaction between finite databases and fixed, possibly infinite, interpreted structure, has not been fully explored in the database community, although the question was already raised in the seminal paper of Chandra and Harel [17]. There are some exceptions: in addition to [26], one exception has been the attention to order relation. This includes well-known connections between query languages and complexity classes, cf. [1, 21]; also, order constraints have been studied in the context of DATALOG queries (see, e.g., [37, 51]). In this paper, we explore some of the nontrivial interaction of interpreted structure with relational structure, hopefully filling in a bit of the gap mentioned above. We focus on looking at analogs of classical equivalence and expressivity results, domain of quantification, and on complexity bounds.

Much recent interest in databases over interpreted structures stems from a non-classical model, the *constraint database model* introduced by Kanellakis, Kuper and Revesz [36]. They were motivated by new applications involving spatial and temporal data, which require storing and querying infinite collections. The constraint model generalizes Codd's relational model by means of "generalized relations". These are possibly infinite sets defined by quantifier-free first-order formulae in the language of some underlying infinite structure $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$. Here $\mathcal{U}$ is a set (assumed to be infinite), and Ω is a signature that consists of a number of interpreted functions and predicates over $\mathcal{U}$. For example, in spatial applications, $\mathcal{M}$ is usually the real field $\langle \mathbb{R}, +, *, 0, 1, < \rangle$, and generalized relations describe sets in $\mathbb{R}^n$.

One of the main contributions of [36] is an extension of relational calculus and DATALOG to generalized relations. The development of constraint query languages made clear some of the ways in which the interpreted structure adds new issues to the analysis of query languages. For example, although these extensions do add expressive power, in [10] it was shown that this extension, with the real field as the underlying structure, does not add *relational* expressive power: one cannot define parity or transitive closure, and, more generally, queries that are *generic* in the sense of [1, 18], other than those already definable with order. Since it is known that such expressive bounds cannot be obtained for arbitrary structures [28], these expressiveness results give indication that relational calculi over some interpreted structures are much more similar to pure relational calculus than others are.

The constraint model also shows how issues concerning quantification that do not arise in the classical theory come to the fore when interpreted structure is present. The 'classical' interpretation of quantifiers such as $\exists x.\varphi(x)$ occurring in relational calculus expressions is that $\varphi(x)$ is satisfied by some element of the *database*. This is called the *active-semantics* interpretation of queries (or just active interpretation). In contrast, the the constraint model interprets $\exists x.\varphi(x)$ as meaning that $\varphi(x)$ is satisfied by some element of the underlying universe $\mathcal{U}$. This is called the *natural interpretation*. The issue of two semantics arises in the context of pure relational calculus, but Hull and Su [33] proved that the expressive power of relational calculus is the same under both interpretations. This is what we call the *natural-active collapse*. (In fact, an earlier result [3] showed that infinite domains are unnecessary for pure relational calculus.) This collapse is known not to hold over arbitrary structures, hence questions concerning the two semantics become much richer in the interpreted setting.

In the first part of the paper, we study *active-semantics queries*. For those queries, we will review and extend a technique that has arisen recently [10, 43] in the context of first-order logic: *generic collapse* results. These state that generic queries definable over interpreted structures can be defined without additional interpreted operations. We simplify the proofs so that the generic collapse results become applicable to a variety of logics, such as fixpoint, second-order, and logics with counting. The collapse results are then used to show that for active-semantics queries many (but not all!) traditional relational theory results generalize smoothly to the interpreted setting: in particular, we show that many of the expressive power results and query equivalence results do extend to the interpreted setting.

To study the constraint model, then, we turn in the second half of this paper to the study of the natural semantics. It might appear that we have to always develop a completely separate theory for this new interpretation of quantifiers, unless we can somehow show some version of the natural-active collapse results in the constraint setting.

Significant progress on the issue of natural semantics expressivity was made in [45], which proved the analog of the Hull-Su theorem for *linear* constraints. These results were extended to polynomial constraints in [10] (only for generic queries) and [11] (for arbitrary queries). Although the above results show how to reduce questions about natural semantics to ones concerning active semantics for first-order queries over many structures, there are a number of issues left open. In particular, the proofs in [10] and [11] are quite involved: they use techniques of nonstandard universes [19] and o-minimal structures [46, 50]. In addition, the proofs in those papers are nonconstructive. In [33] (that deals with the pure case) and [45] (that deals with linear constraints), an *algorithm* is given that converts a natural query into an active one. However, in [10, 11] (which deal with polynomial constraints and other o-minimal structures), a mere existence of such a query is proved.

The main result of the second half of this paper is an *algorithm* for doing the conversion from natural quantification to active quantification: an effective version of the natural-active collapse. It is hoped that this algorithm will lay the groundwork for transforming a number of optimization algorithms that occur in the pure relational calculi to the interpreted setting.

Our investigation of natural semantics expressivity then moves from first-order logic to more expressive logics (fixpoint, second-order). We observe a new interesting phenomenon here: Even when we identify structures over which first-order queries have tame behavior under the natural interpretation (for example, $\langle \mathbb{R}, +, *, 0, 1, < \rangle$), fixpoint logics and second-order logic may still be too expressive. Any recursive query can be expressed in these languages, and under the naive interpretation of second-order quantifiers many of the standard languages lack closure. There are numerous subclasses, however, that one can restrict to that do not arise naturally in the pure case, and which have the same closure properties as active queries, while also admitting interesting expressive bounds. In particular, we consider a new class of logics that we call *hybrid*. The idea is that first-order quantification can still be interpreted naturally (that is, $\exists x.\varphi(x)$ means $\varphi(a)$ for some $a \in \mathcal{U}$), but the higher-order features are interpreted actively. For instance, for the hybrid second-order logic, second-order quantifiers range over subsets of the active domain, while first-order quantifiers range over $\mathcal{U}$. Similarly, fixpoints are only taken within the active domain. Such logics are of interest in the constraint database context, since the natural versions of the higher-order constructs are not available.

An interesting aspect of hybrid logics is that standard normal forms that hold straightforwardly for both natural and active interpretation logics become highly nontrivial in the hybrid case. We show in the second half of the paper that the behavior of hybrid logics is close to the behavior of first-order logic. In particular, we show that for reasonably-behaved structures we can obtain standard normal forms for both fixpoint and second-order logic in the hybrid case. For arbitrary interpreted structure we conjecture that these normal forms fail.

At this point, we have a set of results on normal forms and quantifier-bounding for the natural semantics; we go on to give some applications of this to the study of constraint queries. We start by giving a set of corollaries showing expressive bounds on the relational expressive power of constraint queries. These corollaries give, among other things, an elementary proof that parity can not be defined using relational calculus with polynomial inequality constraints. This new proof completely avoids techniques of nonstandard universes, and gives a constructive 'parameterized quantifier-elimination' procedure that can be seen as an extension of the work in [45, 6, 7, 14, 48, 47].

Finally, we give corollaries on the complexity of constraint query evaluation. Since the relational algebra and calculus are equivalent to pure first-order logic, they have AC^0 data complexity, cf. [1]. Adding constraints increases this complexity. For instance, if multiplication is in the signature, the AC^0 complexity bound is lost, cf. [15]. As an upper bound for complexity, it is known that if $\mathcal{M} = \langle \mathbb{R}, +, *, 0, 1, < \rangle$, then data complexity of first-order queries is NC, see [36, 14]. Since $AC^0 \subset NC$, one could hope for more precise information about the complexity of constraint queries over the real field. In the last part of the paper, we use results on bounding-quantification to get tighter bounds on query evaluation for the real field (the principal result being a TC^0 bound); we also get complexity bounds for a variety of other languages.

Organization Notations are introduced in Section 2. In Section 3, we deal with the active-semantics interpretation. We present the Ramsey technique (which is abstracted from the proofs in [10] and [45]), and show that it can be uniformly applied to a variety of logics (defined in Section 2), to obtain generic collapse results. This leads to a number of expressivity results for these logics over interpreted structures. We then aim to extend these results to study the expresssive power of algebras. We first show that natural extensions of relational algebra, DATALOG$^\neg$ and WHILE, are still equivalent with first-order, least- and partial-fixpoint logic over arbitrary structures. We also look at infinitary logic with finitely many variables (which subsumes all fixpoint logics), and show how to extend expressivity bounds to it.

In Section 4, we turn to the natural interpretation. We start by analyzing the pure case, that is, when no interpreted operations are present. We simplify the proof of the natural-active collapse from [33] and extend it to variable-bounded fragments of first-order and infinitary logic.

Our main goal is to give a *constructive* proof of the natural-active collapse for databases over well-behaved structures. We prove that the natural and active interpretations are equally expressive over any structure that is o-minimal [46, 50] and has quantifier elimination. Furthermore, if the quantifier-elimination procedure is effective, an active-semantics formula equivalent to a natural-semantics formula can be effectively computed. We then move to higher-order logics. We introduce hybrid second-order logic (in which second-order quantifiers range over the subsets of active domain), and hybrid fixpoint logics (in which fixpoint is taken within the active domain) and prove the natural-active collapse for them.

In section 5 we give applications of the results of the previous two sections. We derive expressive bounds on constraints queries in any nicely-behaved structure. We also apply the results of section

4 to prove a TC^0 complexity bound for first-order logic with polynomial constraints, thus improving the NC bound of [36].

This paper is based on two extended abstracts that appeared in the Proceedings of the 12th Symposium on Logic in Computer Science [11] and the Proceedings of the 16th Symposium on Principles of Database Systems [12].

2 Notations

Databases over infinite structures

In this paper, we study databases over infinite structures. Let $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$ be an infinite structure, where $\mathcal{U}$ is an infinite set, called the universe of the structure (in the database literature it is often called domain), and Ω is a set of interpreted functions, constants, and predicates. For example, the real field $\langle \mathbb{R}, +, *, 0, 1, < \rangle$ has $\mathbb{R}$ (the set of real numbers) as the universe, and the signature consists of functions $+$ and $*$, constants 0 and 1, and predicate $<$.

A (relational) *database schema* SC is a nonempty set of relation names $\{S_1, \ldots, S_l\}$ with associated arities $p_1, \ldots, p_l > 0$. Given a structure $\mathcal{M}$ and $X \subseteq \mathcal{U}$, an *instance* of SC over X is a family of finite sets, $\{R_1, \ldots, R_l\}$, where $R_i \subseteq X^{p_i}$. That is, each schema symbol S_i of arity p_i is interpreted as a finite p_i-ary relation over X. We use $\mathit{Inst}(SC, X)$ or $\mathit{Inst}(SC, \mathcal{M})$ to denote the set of all instances of SC over X or $\mathcal{M}$.

Given an instance D, $\mathit{adom}(D)$ denotes its *active domain*, that is, the set of all elements of $\mathcal{U}$ that occur in the relations in D. If S is a new n-ary symbol not in SC and R is a finite subset of $\mathcal{U}^n$, then D_R denotes the instance of $SC \cup \{S\}$ where S is interpreted as R.

In Section 4, we will use *o-minimal* structures [46]. A structure $\mathcal{M}$ is ordered if one of the symbols in Ω is $<$, interpreted as a linear order on the universe. Such a structure $\mathcal{M}$ is o-minimal, if every definable set is a finite union of points and open intervals $\{x \mid a < x < b\}$, $\{x \mid x < a\}$ and $\{x \mid x > a\}$. Definable sets are those of the form $\{x \mid \mathcal{M} \models \varphi(x)\}$, where φ is a first-order formula in the language of $\mathcal{M}$, possibly supplemented with symbols for constants from $\mathcal{M}$.

A structure admits quantifier elimination if, for every formula $\varphi(\vec{x})$, there is an equivalent quantifier-free formula $\psi(\vec{x})$ such that $\mathcal{M} \models \forall \vec{x}. \varphi(\vec{x}) \leftrightarrow \psi(\vec{x})$. A structure is *recursive* if its language is recursive and validity of atomic sentences is decidable. An example of o-minimal, recursive structure having quantifier elimination is the real field $\langle \mathbb{R}, +, *, 0, 1, < \rangle$: since it has quantifier elimination [48], it is decidable, and every formula $\varphi(x)$ is equivalent to a Boolean combination of constraints of the form $p(x) = 0$ or $p(x) < 0$, where p is a polynomial, which implies o-minimality. Some extensions of the real field are known to be o-minimal, for example, $\langle \mathbb{R}, +, *, e^x \rangle$ [54].

Logics

Since our goal is to develop a theory that can be used beyond the first-order case, we consider a variety of logics here. Fix a structure $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$. By $L(SC, \Omega)$ we denote the language that consists of the schema predicates and the symbols in Ω. By $\mathcal{FO}(SC, \mathcal{M})$ (or just $\mathcal{FO}(\mathcal{M})$ if the schema is understood) we denote first-order logic over the language $L(SC, \Omega)$; we use $\mathcal{FO}$ for first-order logic in the language of the schema. We also define the semantic notion $D \models \varphi(\vec{a})$, where $\varphi(\vec{x})$ is a formula and $\vec{a}$ a vector over $\mathcal{U}$. Note that $D \models \exists x.\varphi(x, \vec{a})$ means that for some $a_0 \in \mathcal{U}$, $D \models \varphi(a_0, \vec{a})$. This corresponds to the *natural* interpretation of queries, cf. [1, 33].

Under the *active* interpretation of first-order logic [1, 33], $\exists x$ means that x can be found within the active domain. To deal with this in the same framework as the natural interpretation (that is, to avoid introduction of a different notion of satisfaction), we introduce the *active quantification* of the form $\exists x \in adom.\varphi(x)$ and $\forall x \in adom.\varphi(x)$. The semantics is as follows: $D \models \exists x \in adom.\varphi(x, \vec{a})$ if for some $a_0 \in adom(D)$, $D \models \varphi(a_0, \vec{a})$, and likewise for $\forall x \in adom$. Note that this restricted quantification can be expressed in first-order logic. We shall call a formula *active* if all quantifiers in it are active. For every language $\mathbb{L}$, its subset consisting of the active formulae is denoted by $\mathbb{L}_{\text{act}}$.

Next, we consider fixpoint extensions of first-order logic. The presentation here differs slightly from the standard definitions for the finite [18, 21, 31, 2] or infinite [42] case because we have a mix of these cases: finite structures over an infinite universe. (This corresponds to what we called *hybrid* logics in the introduction.) We say that a predicate symbol S occurs negatively in a formula φ if it occurs in S under the scope of an odd number of negations. A formula φ is *positive in S* if S does not occur negatively in φ.

The *least-fixpoint* logic $\mathcal{FO}+\mathbf{lfp}$ adds the following construction rule: if S is an n-ary relation symbol not in the schema, and S occurs positively in a formula $\varphi(x_1, \ldots, x_n, \vec{y}, S)$, and $\vec{t}$ is an n-vector of variables or terms, then

$$[\textsc{Lfp}_{\vec{x},S}\varphi(\vec{x}, \vec{y}, S)](\vec{t})$$

is a formula, whose free variables are $\vec{y}$ and free variables from $\vec{t}$. The semantics is as follows. Given $D \in \mathit{Inst}(SC, \mathcal{U})$ and $\vec{a}$ of the same length as $\vec{y}$, define the sequence $R_0^{\vec{a}} = \emptyset$,

$$R_{i+1}^{\vec{a}} = \{(b_1, \ldots, b_n) \in adom(D)^n \mid D \models \varphi(\vec{b}, \vec{a}, R_i^{\vec{a}})\}.$$

It is known that this sequence is monotone: ($R_i^{\vec{a}} \subseteq R_{i+1}^{\vec{a}}$), and thus reaches a fixpoint, denoted by $R_\infty^{\vec{a}}$. Now $D \models [\textsc{Lfp}_{\vec{x},S}\varphi(\vec{x}, \vec{a}, S)](\vec{c})$ iff $\vec{c} \in R_\infty^{\vec{a}}$. The least-fixpoint logic over the logical language $L(SC, \Omega)$ will be denoted by $\mathcal{FO}+\mathbf{lfp}(\mathcal{M})$.

Note that we may have instances of first-order quantification in least-fixpoint formulae, which are interpreted naturally, but the fixpoint itself is always taken within the active domain, which makes the logic hybrid: it combines both active and natural interpretations. In the pure finite case, it is unnecessary to add parameters $\vec{y}$ to the iterated formula, because it does not add power [21]. The same is true in the pure infinite case, that is, when $R_{i+1}^{\vec{a}}$ is constructed as the set of *all* tuples $\vec{b}$ over $\mathcal{U}$ that satisfy $\varphi(\vec{b}, \vec{a})$. But it is not clear whether the extra parameters $\vec{y}$ can be dropped in hybrid fixpoint logics without losing expressiveness. We shall say that a fixpoint formula is in *normal form* if no application of $\textsc{Lfp}$ uses these extra parameters $\vec{y}$.

The *inflationary-fixpoint* logic $\mathcal{FO}+\mathbf{ifp}$ and the *partial-fixpoint* logic $\mathcal{FO}+\mathbf{pfp}$ are defined similarly to the least-fixpoint logic. They have constructors

$$[\textsc{Ifp}_{\vec{x},S}\varphi(\vec{x}, \vec{y}, S)](\vec{t}) \quad \text{and} \quad [\textsc{Pfp}_{\vec{x},S}\varphi(\vec{x}, \vec{y}, S)](\vec{t})$$

respectively, with no restriction on occurrences of S. The semantics is as above, where $R_\infty^{\vec{a}}$ is constructed as follows. For $\mathcal{FO}+\mathbf{ifp}$, $R_{i+1}^{\vec{a}} = R_i^{\vec{a}} \cup \{(b_1, \ldots, b_n) \in adom(D)^n \mid D_{R_i^{\vec{a}}} \models \varphi(\vec{b}, \vec{a})\}$ (this sequence is increasing, and hence reaches a fixpoint). For $\mathcal{FO}+\mathbf{pfp}$, the sequence R_i is constructed as for the least-fixpoint, and R_∞ is taken to be its fixpoint, if it exists, and $\emptyset$ otherwise.

The concept of a normal form is defined for inflationary- and partial-fixpoint logic in the same way it was defined for $\mathcal{FO}+\mathbf{lfp}$. When formulae of a fixpoint logic are restricted to normal form formulae only, we call such a fixpoint logic *closed.* In the case of finite domain, there is no loss of expressiveness due to restriction to the closed version of a fixpoint logic, cf. [1, 21].

We also consider logic with counting, $\mathcal{FO}+\mathbf{count}(\mathcal{M})$. The presentation we use follows [23]. This logic is two-sorted. A finite structure over $\mathcal{M}$ is of the form $\langle\{a_1,\ldots,a_n\},\{1,\ldots,n\},R_1,\ldots,R_l,1,n,<,+,*\rangle$. Here $\{a_1,\ldots,a_n\}\subset\mathcal{U}$, and the SC-relations R_is are interpreted over $\{a_1,\ldots,a_n\}$, while $<,+,*$ refer to the sort of natural numbers. Here we use $+$ and $*$ as ternary predicates. $\mathcal{FO}+\mathbf{count}(\mathcal{M})$ extends $\mathcal{FO}(\mathcal{M})$ by counting quantifiers $\exists ix$ where i refers to the sort of natural numbers, and x refers to the first sort. This quantifier, that binds x but not i, says that there are at least i satisfiers of φ. For example, $\exists! ix\varphi(x)\equiv(\exists ix\varphi(x))\wedge\forall j(\exists jx\varphi(x)\rightarrow j\le i)$ says that there are exactly i elements satisfying $\varphi(\cdot)$, and $\exists i,j(j+j=i)\wedge(\exists! ix\varphi(x))$ says that the number of satisfiers of φ is even – this is known not to be definable in $\mathcal{FO}$.

The *hybrid second-order logic* $\mathcal{HSO}$ permits second-order quantifiers $\exists S$ and $\forall S$ which are interpreted as follows: $D\models\exists S.\varphi(S)$, where S is k-ary, iff there exists a set $R\subseteq adom(D)^k$ such that $D_R\models\varphi$. Note that this notion is even weaker than weak second-order, where second-order quantifiers range over finite sets. We use $\mathcal{SO}$ for full second-order logic (k-ary second-order quantifiers range over subsets of $\mathcal{U}^k$). Formulae of $\mathcal{SO}$ or $\mathcal{SO}_{\text{act}}$ can be converted into normal form $Q_1S^1\ldots Q_kS^k.\psi$ where Q_iS^i are second-order quantifiers, and ψ is first-order. In the case of $\mathcal{HSO}$, it is not immediately clear if the same is true (see Corollary 11).

While all these logics are relevant to database query languages (as we shall see shortly), we shall also consider *infinitary* logic, which is of interest in finite-model theory, as logic which subsumes fixpoint logics and possesses nice properties, such as 0-1 law [39]. It is defined exactly as first-order logic, except that arbitrary disjunctions and conjunctions are allowed. That is, if $\{\varphi_i(\vec{x})\}$ is an arbitrary collection of formulae, then $\bigvee_i\varphi_i(\vec{x})$ and $\bigwedge_i\varphi_i(\vec{x})$ are formulae. We use $\mathcal{L}_{\infty\omega}$ to denote infinitary logic.

Suppose $\mathbb{L}$ is one of the logics introduced above, but the formation rules are modified so that only finitely many variables, $x_1,\ldots,x_k$, can be used in formulae. The restriction thus obtained is denoted by $\mathbb{L}^k$. For example, $\mathcal{L}^k_{\infty\omega}$ is infinitary logic with k variables. We use $\mathcal{L}^\omega_{\infty\omega}$ for $\bigcup_{k\in\mathbb{N}}\mathcal{L}^k_{\infty\omega}$.

In the absence of interpreted symbols in Ω, we speak of a *pure* logic (over a schema SC).

Queries definable by logics

A *query* is a mapping from $Inst(SC_1)$ to $Inst(SC_2)$, where SC_1 and SC_2 are two schemas. For simplicity of exposition, assume that SC_2 consists of a single p-ary relation. Given a logic $\mathbb{L}$ and a structure $\mathcal{M}$, we say that a query Q is $\mathbb{L}$-definable over $\mathcal{M}$ (or $\mathbb{L}(\mathcal{M})$-definable) if there exists an $\mathbb{L}$ formula $\varphi(x_1,\ldots,x_p)$ in the language $L(SC_1,\Omega)$ such that $Q(D)=\{\vec{a}\mid D\models\varphi(\vec{a})\}$. We denote this query by Q_φ.

For the special case of $\mathcal{M}$ being $\langle\mathcal{U},<\rangle$, we write $\mathbb{L}(<)$ instead of $\mathbb{L}(\langle\mathcal{U},<\rangle)$.

As in the case of relational calculus and algebra, we often consider queries that do not extend the active domain. Thus, we define the query Q^{act}_φ by $Q^{\text{act}}_\varphi(D)=\{\vec{a}\in adom(D)^p\mid D\models\varphi(\vec{a})\}$. Note that any query Q obtained in such a way is *domain-preserving*: $adom(Q(D))\subseteq adom(D)$.

Query languages

Relational calculus is just first-order logic over the database schema: its expressions are of the form $e = \{\vec{x} \mid \varphi(\vec{x})\}$ where $\varphi(\vec{x})$ is a $\mathcal{FO}$ formula in the language of the schema relations. By the Hull-Su theorem, we can use $\mathcal{FO}_{\text{act}}$ expressions. We use CALC to denote the family of all calculus queries under the active interpretation (that is, Q_{φ}^{act}), and ALG to denote relational algebra. It is a classical result of relational theory that CALC = ALG.

We consider DATALOG$^{\neg}$, which is datalog with negation allowed in bodies of rules. That is, a rule is of the form $H :- B_1, \ldots, B_n$, $n \geq 0$, where each B_i is an atom or a negated atom, and H is an atom. Following [1, 2], we give it a simple inflationary semantics. That is, each iteration infers new facts and adds them to the facts already inferred; thus, a fixpoint is always reached.

We also consider the WHILE language. It extends ALG by allowing the statement (*while change begin e end*) where e is an expression [2, 1, 18]. It iterates e as long as it changes at least one relation. A WHILE statement is either an assignment of the form $V{:=}E$, where V is a variable and E is an expression, or a *while* expression above. A WHILE program is a sequence of *while* statements. See [1, 2] for more details.

Equivalences for the finite domain

Query languages introduced here have been studied in depth in the classical relational theory. Many equivalence results are known in the pure finite case. By the pure finite case we mean this: the only free nonlogical symbols are the schema relations, queries are interpreted actively and their outputs are restricted to the active domain. That is, a query does not extend the active domain, and all quantification is active. Below we list some of the most important equivalences.

First, CALC = ALG = $\mathcal{FO}$ [1]. Similar equivalences have been obtained in the case of interpreted operations (with some restrictions) given by abstract datatypes, see [8]. For fixpoint logics, $\mathcal{FO}+\mathbf{lfp}$ = $\mathcal{FO}+\mathbf{ifp}$ = DATALOG$^{\neg}$ [2, 31] and WHILE = $\mathcal{FO}+\mathbf{pfp}$ [2].

In the presence of an order relation, these equivalences continue to hold and, in addition, $\mathcal{FO}+\mathbf{lfp}$ captures all generic PTIME queries, and $\mathcal{FO}+\mathbf{pfp}$ captures all generic PSPACE queries [34, 52]. Also, in the presence of order, $\mathcal{FO}+\mathbf{count}$ captures uniform TC^0 [5, 23].

$\mathcal{FO}+\mathbf{count}$ is closely connected to to relational languages with aggregates; this connection, hinted at in [40], was explicitly shown in [41]. Second-order logic was shown to be relevant to the study of languages for complex objects, see [32]. $\mathcal{L}^{\omega}_{\infty\omega}$ is of interest in finite-model theory because it subsumes fixpoint logics. Variable-bounded logics are studied primarily in (finite) model theory, but [53] demonstrated nice connections with expression and combined complexity.

Genericity

A query Q (that is, a mapping from $\mathit{Inst}(SC_1, X)$ to $\mathit{Inst}(SC_2, X)$) is *totally generic* [10] if, for any $D \in \mathit{Inst}(SC_1)$ and any injective map $\pi : \mathit{adom}(D) \cup \mathit{adom}(Q(D)) \to X$, it is the case that $Q(\pi(D)) = \pi(Q(D))$. A query is *locally generic* [10] if X is ordered, and the above holds for any injective monotone π. Clearly, total genericity implies local genericity. Examples of generic (locally or totally) queries are any ALG, DATALOG$^{\neg}$ or WHILE query, when no interpreted operations are present. Example of a locally but not totally generic query is $Q(S_1, S_2) \equiv \forall x \forall y. S_1(x) \wedge S_2(y) \to x < y$.

Also note that totally and locally generic queries that return finite results are domain-preserving: $adom(Q(D)) \subseteq adom(D)$, see [10]. For more of genericity, see [1, 9, 10].

3 Active interpretation

The active interpretation does not allow a query to ask any question about what is outside of the finite database: while an infinite structure is there, and we can even see a small part of it, we cannot ask much about it. So, how much more does the mere presence of this structure add to the expressiveness of query languages? The answer is – practically nothing. We start by proving this result for a variety of logics. Then we show that many of the equivalences among languages continue to hold, when languages are appropriately modified. We derive some complexity corollaries, and also consider infinitary logic as a separate case.

Note: in this section we only deal with active quantification, so we write $\mathbb{L}$ instead of $\mathbb{L}_{\text{act}}$. Similarly, we write Q_φ instead of Q_φ^{act}, since the natural semantics case is not considered in this section.

3.1 Ramsey property and expressivity bounds

The main goal of this section is to prove generic collapse results for a number of logics. We say that a logic $\mathbb{L}$ has a *locally generic collapse* over an ordered structure $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$ if, for every schema SC, every $\mathbb{L}(\mathcal{M})$-definable locally generic query on SC-databases, is already $\mathbb{L}(<)$-definable. That is, $\mathcal{M}$ is as expressive as just the order relation, with respect to locally generic queries. A logic $\mathbb{L}$ has a *generic collapse* over a structure $\mathcal{M}$ if every $\mathbb{L}(\mathcal{M})$-definable totally generic query on SC-databases is definable in pure $\mathbb{L}$.

This problem of collapsing signatures for the active quantification was considered for first-order logic in [10] and independently in [43]. However, the techniques in [10] relied heavily on translation into prenex form, and the extension to second-order logic in [11] was ad hoc. In [43], an elementary extension is used that possesses a set of indiscernibles, and it is unclear whether this technique works beyond the first-order case.

However, we show here that the technique of [10, 11] can be modified so that it can be applied to a variety of other logics. In particular, we show that a proof based on Ramsey's theorem [27], can proceed inductively on the structure of a formula, thus making it unnecessary to impose syntactic restrictions. Consequently, we get a series of results that give us expressivity bounds for logics under the active interpretation.

Definition 1 *Let $\mathbb{L}$ be a logic. We say that it has a* Ramsey property *over an ordered structure $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$ if, for any SC, the following is true:*

> *Let $\varphi(\vec{x})$ be an $\mathbb{L}$-formula in the language $L(SC, \Omega)$, and X an infinite subset of $\mathcal{U}$. Then there exists an infinite set $Y \subseteq X$ and a $L(SC, <)$ formula $\psi(\vec{x})$ such that for any $D \in Inst(SC, Y)$ and any $\vec{a}$ over Y, it is the case that $D \models \varphi(\vec{a}) \leftrightarrow \psi(\vec{a})$.*

We also speak of a formula φ having the Ramsey property if the above is true. We speak of the total *Ramsey property if ψ is a $L(SC)$ formula.*

As was shown previously [10, 11], the Ramsey property implies the following collapse for generic queries:

Lemma 1 (Generic Collapse Lemma)

1. *If $\mathbb{L}$ has the Ramsey property over $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$, and every $\mathbb{L}(<)$-query is locally generic, then $\mathbb{L}$ has the locally generic collapse over $\mathcal{M}$.*

2. *If $\mathbb{L}$ has the total Ramsey property over $\mathcal{M}$, and every $\mathbb{L}$-query is totally generic, then $\mathbb{L}$ has the generic collapse over $\mathcal{M}$.*

Proof. Let Q be a locally generic query definable in $\mathbb{L}(\mathcal{M})$. By the Ramsey property, we find an infinite $X \subseteq \mathcal{U}$ and a $\mathbb{L}(\langle \mathcal{U}, < \rangle)$-definable Q' that coincide on X. We claim they coincide everywhere. Let $D \in \mathit{Inst}(\mathcal{M})$. Since X is infinite, there exists partial monotone injective map π from $\mathit{adom}(D)$ into X. Since Q' is locally generic and thus Q and Q' do not extend active domains, we have $\pi(Q(D)) = Q(\pi D) = Q'(\pi D) = \pi(Q'(D))$ from which $Q(D) = Q'(D)$ follows. □

The condition that every $\mathbb{L}(<)$-query is locally generic, and every $\mathbb{L}$-query is totally generic, holds for all the logics we introduced. Thus, to limit their expressiveness over infinite structures, we have to prove the Ramsey property. First, we state a simple lemma that is often used as a first step in such proofs.

Lemma 2 *Let $\varphi(\vec{x})$ be an $\mathbb{L}$ formula in the language $L(SC, \Omega)$, where $\mathbb{L}$ is one of the logics introduced in the previous section. Then there exists an equivalent formula $\psi(\vec{x})$ such that every atomic subformula of ψ is either an $L(SC)$ formula, or a $L(\Omega)$ formula. Furthermore, for any set $\vec{z} \subseteq \vec{x}$ of free variables of φ, there is an equivalent formula $\psi(\vec{x})$ such that none of variables $\vec{z}$ occurs in an $L(SC)$-atomic formula.*

Proof. Introduce m fresh variables $z_1, \ldots, z_m$, where m is the maximal arity of a relation in SC, and replace any atomic formula of the form $R(t_1(\vec{y}), \ldots, t_l(\vec{y}))$, where $l \leq m$ and the t_is are Ω-terms, by $\exists z_1 \in \mathit{adom} \ldots \exists z_l \in \mathit{adom}. \bigwedge_i (z_i = t_i(\vec{y})) \wedge R(z_1, \ldots, z_l)$. Similarly use existential quantifiers to eliminate $\vec{z}$-variables from $L(SC)$-atomic formulae. □

The key in the inductive proofs of the Ramsey property is the case of Ω-atomic subformulae. This was the key idea of the proof for the first-order case in [10], though the lemma below was not stated explicitly.

Lemma 3 *Let $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$ be an infinite ordered structure, and $\varphi(\vec{x})$ an atomic formula. Then φ has the Ramsey property.*

Proof. We show that any infinite $X \subseteq \mathcal{U}$ contains an infinite sequence indiscernible for φ and every subformula of φ obtained by collapsing two or more variables. Let $\vec{x} = (x_1, \ldots, x_l)$, and let $\mathit{Part}_\prec(\vec{x})$ be the set of ordered partitions on $\{x_1, \ldots, x_l\}$; that is, partitions with an order relation $\prec$ on the blocks. For each $P \in \mathit{Part}_\prec(\vec{x})$, let $\zeta(P)$ be a $<$-formula specifying the partition, and let φ/P be φ in which a representative for a block in P replaces every occurrence of every variable from that block. Then $\mathcal{M} \models \varphi(\vec{x}) \leftrightarrow \bigvee_{P \in \mathit{Part}_\prec(\vec{x})} (\zeta(P) \wedge \varphi/P)$.

Let $Z \subseteq \mathcal{U}$ be an infinite set, and P an ordered partition of $\{x_1, \ldots, x_l\}$. Let $y_1, \ldots, y_s$ be representatives of blocks of P such that $\models \zeta(P) \rightarrow \bigwedge_{i<j}(y_i < y_j)$. By Ramsey's theorem [27], there exists an infinite subset $Z_P \subseteq Z$ such that either for every ordered tuple $(c_1, \ldots, c_s), c_1 < \ldots < c_s$, of elements of Z_P it is the case that $\models \varphi/P(\vec{c}) \leftrightarrow \mathbf{T}$, or for every such ordered tuple it is the case that $\models \varphi/P(\vec{c}) \leftrightarrow \mathbf{F}$. Consequently, either $\models (\zeta(P) \wedge \varphi/P) \leftrightarrow \zeta(P)$ for every $\vec{x}$ over Z_P, or $\models (\zeta(P) \wedge \varphi/P) \leftrightarrow \mathbf{F}$ for every $\vec{x}$ over Z_P.

Now let $P_1, \ldots, P_k$ be some enumeration of the (finite) set $Part_{\prec}(\vec{x})$. Let $Z^0 = \mathcal{U}$, $Z^1 = Z^0_{P_1}, \ldots, Z^i = Z^{i-1}_{P_i}, \ldots, Z^k = Z^{k-1}_{P_k}$. Then it follows that over Z^k, φ is equivalent to a Boolean combination of atomic $L(<)$-formulae, thus proving the lemma. □

Now an inductive argument proves:

Proposition 1 *The following have the Ramsey property:*

- $\mathcal{FO}$;
- $\mathcal{FO}+$**lfp**;
- $\mathcal{FO}+$**ifp**;
- $\mathcal{FO}+$**pfp**;
- $\mathcal{FO}+$**count**;
- $\mathcal{SO}$.

Proof. Fix a structure $\mathcal{M}$ and a schema SC. By Lemma 2 we assume without loss of generality that every atomic subformula is a $L(SC)$ formula or a $L(\Omega)$ formula. We prove the theorem by induction on the formula. We start with the $\mathcal{FO}$ case.

The base cases are obvious: for a $L(SC)$ formula, there is no need to change the formula or find a subset, and for a $L(\Omega)$ atomic formula it is given by Lemma 3.

Let $\varphi(\vec{x}) = \varphi_1(\vec{x}) \wedge \varphi_2(\vec{x})$, and $X \subseteq \mathcal{U}$ infinite. First, find ψ_1, $Y_1 \subseteq X$ such that for any D and $\vec{a}$ over Y_1, $D \models \varphi_1(\vec{a}) \leftrightarrow \psi_1(\vec{a})$. Next, by using the hypothesis for φ_2 and Y_1, find an infinite $Y_2 \subseteq Y_1$ such that for any D and $\vec{a}$ over Y_2, $D \models \varphi_2(\vec{a}) \leftrightarrow \psi_2(\vec{a})$. Then take $\psi = \psi_1 \wedge \psi_2$ and $Y = Y_2$. The case of $\varphi = \neg\varphi'$ is trivial.

For the existential case, let $\varphi(\vec{x}) = \exists y \in adom.\varphi_1(y, \vec{x})$. By the hypothesis, find $Y \subseteq X$ and $\psi_1(y, \vec{x})$ such that for any D and $\vec{a}$ over Y and any $b \in Y$ we have $D \models \varphi_1(b, \vec{a}) \leftrightarrow \psi_1(b, \vec{a})$. Let $\psi(\vec{x}) = \exists y \in adom.\psi_1(y, \vec{x})$. Then, for any D and $\vec{a}$ over Y, $D \models \psi(\vec{a})$ iff $D \models \psi_1(b, \vec{a})$ for some $b \in adom(D)$ iff $D \models \varphi_1(b, \vec{a})$ for some $b \in adom(D)$ iff $D \models \varphi_1(\vec{a})$, thus finishing the proof for the $\mathcal{FO}$ case.

Before we give the proof for the fixpoint logics, note that if a schema predicate occurs only positively in φ, then it occurs only positively in the formula ψ constructed by the above inductive argument. Also note that the separation Lemma 2 does not change positive occurrences to negative and vice versa.

Proof for $\mathcal{FO}+$**lfp**. The proof is by induction on syntactic complexity, that is, the total number of occurrences of the Boolean connectives, quantifiers, and the LFP operator. Since we are dealing with the finite case, we assume that that we only have closed fixpoint formulae. The basis cases and the cases of Boolean connectives and quantifiers are proved as before.

Let S be a new n-ary relation symbol that occurs positively in a $L(SC \cup \{S\}, \Omega)$ formula $\alpha(\vec{x})$, and let $\varphi(\vec{y}) = \mathrm{LFP}_{\vec{x},S}[\alpha(\vec{x}, S)](t_1(\vec{y}), \ldots, t_n(\vec{y}))$. We can assume without loss of generality that the fixpoint formula is applied to a vector of variables, since $\varphi(\vec{y})$ is equivalent to $\exists z_1 \in adom \ldots \exists z_n \in adom.\mathrm{LFP}_{\vec{x},S}[\alpha(\vec{x}, S)](\vec{z}) \wedge \bigwedge_i (z_i = t_i(\vec{y}))$.

Given infinite $X \subseteq \mathcal{U}$, we use the hypothesis to find an infinite $Y \subseteq X$ and a $L(SC \cup \{S\}, <)$ formula $\beta(\vec{x})$ such that for any $SC \cup \{S\}$-database $D_R \in \mathit{Inst}(SC \cup \{S\}, Y)$ it is the case that $D_R \models \alpha(\vec{a}) \leftrightarrow \beta(\vec{a})$ for all $\vec{a} \in Y^n$; also, we see that S occurs positively in β.

Now, fix $D \in \mathit{Inst}(SC, Y)$ and $\vec{a} \in Y^n$, and define $R_0(\alpha) = \emptyset$, and

$$R_i(\alpha) = \{\vec{b} \mid \vec{b} \subseteq adom(D), D_{R_{i-1}(\alpha)} \models \alpha(\vec{b})\},$$

and similarly define $R_i(\beta)$.

To show that $D \models \mathrm{LFP}_{\vec{x},S}[\alpha(\vec{x}, S)](\vec{a}) \leftrightarrow \mathrm{LFP}_{\vec{x},S}[\beta(\vec{x}, S)](\vec{a})$, it is enough to show that $R_i(\alpha) = R_i(\beta)$ for all i. We do this by induction on i: first, $R_0(\alpha) = R_0(\beta)$, and if $R_i(\alpha) = R_i(\beta) = R_i$, then $D_{R_i} \models \alpha(\vec{b}) \leftrightarrow \beta(\vec{b})$ for any $\vec{b}$ over $adom(D)$ by the hypothesis, so we get $R_{i+1}(\alpha) = R_{i+1}(\beta)$. This completes the proof for $\mathcal{FO}+\mathbf{lfp}$.

The proof for $\mathcal{FO}+\mathbf{ifp}$ and $\mathcal{FO}+\mathbf{pfp}$ follows along the same lines (it is, in fact, simpler, because one does not need to worry about positiveness).

For $\mathcal{FO}+\mathbf{count}$, one has to add a new case to the basis (atomic formulae of the second sort), and two new cases for the inductive proof. One is the second-sort quantification, which poses no problem, and the other is $\exists ix.\varphi(x, \vec{z})$. For this case one applies the hypothesis to φ to get an equivalent formula ψ over some infinite set Y, and then $\exists ix.\psi(x, \vec{y})$ is the required formula, because i satisfiers of φ over the active domain also satisfy ψ.

To prove the statement for $\mathcal{SO}$, one has to add a new case of the second-order quantifiers. We omit the easy proof. $\square$

Corollary 1 *Each of the logics in Proposition 1 has locally generic collapse.*

The main technique of the proof can easily be extended to other logics, (e.g., transitive closure logics [21]).

Total Ramsey property and generic collapse

It is clear from the proof of Proposition 1 that only the case of atomic $L(\Omega)$ formulae requires the introduction of the order relation. Thus, if atomic $L(\Omega)$ formulae had the total Ramsey property over $\mathcal{M}$, so would all of the logics in the statement of Proposition 1. We call a signature Ω *analytic* on $\mathbb{R}$ if it consists of real-analytic functions. For example, $(+, *)$ is an analytic signature. For those signatures, we can prove the generic collapse result. First, we need

Lemma 4 *Let $\mathcal{F} = \{f_i(\vec{x})\}_{i \in I}$ be a countable family of real-analytic functions, where $\vec{x} = (x_1, \ldots, x_l)$. Assume that none of the functions in $\mathcal{F}$ is identically zero. Let $X \subseteq \mathbb{R}$ be a set of cardinality of the* *continuum. Then it is possible to find a set $Y \subseteq X$ of cardinality of the continuum such that for any tuple $\vec{c}$ of l distinct elements of Y, none of $f_i(\vec{c}), i \in I$, equals zero.*

Proof. We need some notation first. Let f be a function, with $x_{i_1}, \ldots, x_{i_k}$ being some of its parameters. Let $c_{i_1}, \ldots, c_{i_k} \in \mathbb{R}$. Then $f[c_{i_1}/x_{i_1}, \ldots, c_{i_k}/x_{i_k}]$ denotes the function obtained from f by substituting c_{i_j} for x_{i_j}; this function has k fewer parameters than f. Since composition of real-analytic functions is real-analytic again, this function is real-analytic if so is f.

Let $A \subseteq X$. We say that A is $(\mathcal{F}, X)$-nice if A is infinite and for any tuple $\vec{c}$ of l distinct elements of A, none of $f_i(\vec{c})$ equals zero. We first prove that there exists a $(\mathcal{F}, X)$-nice set. Note that if $C \subset \mathbb{R}$ is countable and $\mathcal{H}$ is a countable collection of real-analytic functions with at most l parameters $x_1, \ldots, x_l$, and none of the functions in $\mathcal{H}$ is identically zero, then there exists $c \in \mathbb{R} - C$ such that none of the functions of form $h[c/x_i]$, $h \in \mathcal{H}$, is identically zero. Indeed, for any function $h(x_{i_1}, \ldots, x_{i_{k+1}})$, one can find a vector $(c_1, \ldots, c_k)$ such that $h'(x_{i_{k+1}}) = h[c_1/x_{i_1}, \ldots, c_k/x_{i_k}]$ is not identically zero. Since h' is still analytic, its set of zeros $Z(h, k+1)$ is at most countable. Similarly define sets $Z(h, i)$ for every $1 \leq i \leq l$, assuming such set to be empty if x_i is not a parameter of h. Now the set $Z = \bigcup_{h \in \mathcal{H}, 1 \leq i \leq l} Z(h, i)$ is countable, and thus any $c \in \mathbb{R} - (Z \cup C)$ proves our observation. This observation gives us a simple inductive construction that leads to a countable $(\mathcal{F}, X)$-nice set.

It is easy to see that the union of a chain of $(\mathcal{F}, X)$-nice sets is $\mathcal{F}$-nice again; thus there exists a maximal $\mathcal{F}$-nice set by Zorn's lemma. Let Y be a maximal $\mathcal{F}$-nice set. To conclude the proof of the lemma, we will show that Y has the cardinality of the continuum.

Let κ be the cardinality of Y; assume that κ is less than the cardinality of the continuum. Consider the set $\mathcal{G}$ of all functions of the form $g(x_i) = f[c_1/x_1, \ldots, c_l/x_l]$ where $f \in \mathcal{F}$ and substitution occurs in all positions except the ith one, and all c_js are distinct elements of Y. Then the cardinality of $\mathcal{G}$ is still κ.

Furthermore, each function in $\mathcal{G}$ is real-analytic, and not identically zero, since no f from $\mathcal{F}$ is zero on any vector of distinct elements from Y. Let Z_g be the the set of zeros of $g \in \mathcal{G}$. Since each $g \in \mathcal{G}$ is real-analytic, Z_g is at most countable. Furthermore, the cardinality of $Z = \bigcup_{g \in \mathcal{G}} Z_g$ is at most κ. Hence, there exists $a \in X - (Y \cup Z)$. But now it follows from our construction that $Y \cup \{a\}$ is $(\mathcal{F}, X)$-nice, thus contradicting the maximality of Y. This shows that Y has the cardinality of the continuum and completes the proof of the lemma.

Immediately from here we derive

Lemma 5 *Let $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$, where Ω is analytic. Let $\varphi(\vec{x})$ be an atomic $L(\Omega)$ formula. Then $\varphi(\vec{x})$ has the total Ramsey property.*

Proof. If φ is always true or always false, we are done. Otherwise, it is of the form $f_1(\vec{x}) = f_2(\vec{x})$, where $\vec{x} = (x_1, \ldots, x_l), l > 0$. Let χ_P be a formula specifying a partition of indexes $\{1, \ldots, l\}$, e.g., $(x_1 = x_2) \wedge (x_3 = x_4) \wedge \neg(x_1 = x_3)$ specifies the partition $\{\{1,2\},\{3,4\}\}$. Rewrite f_1 and f_2 to f_1^P and f_2^P by replacing each variable with an index belonging to a block of P by one representative of this block; thus, f_1^P and f_2^P are function in m_P variables, where m_P is the number of blocks of P. For example, if $f_1(x_1, \ldots, x_4) = x_1 x_2 + x_3 x_4$, then for the partition P above we have $f_1^P(x_1, x_3) = x_1^2 + x_3^2$. Now, by Lemma 4, if $f_1^P - f_2^P$ is not identically zero, then for any uncountable $X \subseteq \mathbb{R}$, there is an uncountable $Y \subseteq X$ such that for every $\vec{c}$ of m_P distinct elements over Y, $f_1^P(\vec{c}) \neq f_2^P(\vec{c})$. Thus, using an argument similar to that in the proof of Lemma 3, we get that over some uncountable set $X \subseteq \mathbb{R}$, φ is equivalent to $\bigvee \chi_P$ where P ranges over some set of partitions of $\{1, \ldots, l\}$. □

We now combine this lemma with the proof of Proposition 1 to obtain

Corollary 2 *If $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$, where Ω is analytic, and $\mathbb{L}$ is $\mathcal{FO}$, or $\mathcal{FO}+$**lfp**, or $\mathcal{FO}+$**pfp**, or $\mathcal{FO}+$**ifp**, or $\mathcal{FO}+$**count**, or $\mathcal{SO}$, then $\mathbb{L}$ has the total Ramsey property over $\mathcal{M}$.* $\square$

From Proposition 1 and corollary 2 we obtain the main result of this section.

Theorem 1 *Let $\mathbb{L}$ be $\mathcal{FO}$, or $\mathcal{FO}+$**lfp**, or $\mathcal{FO}+$**pfp**, or $\mathcal{FO}+$**ifp**, or $\mathcal{FO}+$**count**, or $\mathcal{SO}$. Let $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$ be an arbitrary ordered structure. Then $\mathbb{L}$ has locally generic collapse over $\mathcal{M}$. If $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$, where Ω is analytic, then $\mathbb{L}$ has generic collapse over $\mathcal{M}$.* $\square$

This can be used to derive expressivity results for higher-order logic under the active-domain semantics. For example, from the 0-1 law for $\mathcal{FO}+$**pfp** [38], one can get

Corollary 3 *Parity test is not definable in $\mathcal{FO}+$**pfp**$(\langle \mathbb{R}, +, * \rangle)$.* $\square$

3.2 Infinitary logic

Here we extend our results to infinitary logic. We are not interested in the full infinitary logic $\mathcal{L}_{\infty\omega}$, nor $\mathcal{L}^{\omega}_{\infty\omega}$ over ordered structures, because they express every property of finite structures [20]. Thus, we concentrate on $\mathcal{L}^{\omega}_{\infty\omega}$ over unordered structures.

We cannot use the inductive argument of Proposition 1 anymore, because it does not work for infinitary formulae. Indeed, for infinitary disjunction $\bigvee \varphi_i$, one would construct a decreasing family of infinite sets $X_1 \supseteq X_2 \supseteq \ldots$, but its intersection $\bigcap_i X_i$ is not guaranteed to be infinite. Thus, we use the approach that is closer to the proof of the collapse of generic queries for $\mathcal{FO}$ in [43].

Proposition 2 *Let $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ where Ω is analytic, and has countably many symbols. Then $\mathcal{L}^{\omega}_{\infty\omega}$ has generic collapse over $\mathcal{M}$.*

Proof. Let $\varphi(\vec{x})$ be a $\mathcal{L}^{\omega}_{\infty\omega}$ formula in $L(SC, \Omega)$. Assume it uses k variables. Since the number of atomic $L(SC, \Omega)$ formulae in finitely many variables is at most countable, we can assume that all the infinitary disjunctions and conjunctions are countable, and that the collection of atomic subformulae of φ is countable. As before, we assume without loss of generality that every atomic subformula of φ is either a $L(SC)$ formula or a $L(\Omega)$ formula. (We need at most m extra variable for existential quantifiers, where m is the maximum arity of a schema relation.)

Let $\gamma_i(\vec{z})$, $i \in I$ be the collection of all $L(\Omega)$ atomic subformulae of φ, where $\vec{z} = (z_1, \ldots, z_k)$. Each is of the form $f_{1i}(\vec{z}) = f_{2i}(\vec{z})$, where f_{1i} and f_{1i} are Ω-terms, that is, real-analytic functions. Let $g_i(\vec{z}) = f_{1i}(\vec{z}) - f_{2i}(\vec{z})$.

Now, as in the proof of Lemma 5, let $\chi_P(\vec{z})$ be a formula specifying a partition P on $\{z_1, \ldots, z_k\}$, and let $g^P(\vec{z})$ be a function obtained from g by identifying variables that belong to the block of a partition; the number of parameters of this function equals the number of blocks of P. For each function $g_i(\vec{z})$, let $\mathcal{P}_i = \{P \mid g_i^P \text{ is identically zero}\}$. Let

$$\beta_i(\vec{z}) \;\; \equiv \;\; \bigvee_{P \in \mathcal{P}_i} \chi_P(\vec{z}) \;\wedge\; \neg \bigvee_{P \notin \mathcal{P}_i} \chi_P(\vec{z}).$$

Now define a family of functions $\mathcal{G}$ as

$$\bigcup_{i \in I} \{g_i^P \mid P \notin \mathcal{P}_i\}.$$

Note that all functions in $\mathcal{G}$ are real-analytic, and none of them is identically zero. Applying Lemma 4 to $\mathcal{G}$, we find an infinite (in fact, uncountable) set $X \subseteq \mathbb{R}$ such that for any $g \in \mathcal{G}$ and for any $\vec{c}$ consisiting of distinct elements of X, $g(\vec{c}) \neq 0$. Let $\varphi'(\vec{z})$ be obtained from φ by replacing each $\gamma_i(\vec{z})$ with $\beta_i(\vec{z})$. Now a straightforward induction on the structure of φ shows that for any $D \in \mathit{Inst}(SC, X)$ and for any $\vec{c}$ over X, $D \models \varphi(\vec{c}) \leftrightarrow \varphi'(\vec{c})$. This shows that $\mathcal{L}^{\omega}_{\infty\omega}$ has the total Ramsey property over $\mathcal{M}$, and thus it has generic collapse over $\mathcal{M}$. □

Using the 0-1 law for infinitary logic [39], we obtain:

Corollary 4 *The parity test is not definable as a $\mathcal{L}^{\omega}_{\infty\omega}(\langle \mathbb{R}, +, * \rangle)$ query.* □

3.3 Equivalence results and extensions to algebra

We now want to extend the results above to the setting of relational algebras. In this section we show that a number of well-known results on equivalence between logics and relational query languages generalize straightforwardly in the presence of interpreted structures. We then use this to extend our results on expressivity of languages to relational algebra.

As explained before, for every logic $\mathbb{L}$, a $\mathbb{L}(\mathcal{M})$ formula $\varphi(\vec{x})$ with n free variables in the language $L(SC, \Omega)$, defines a query Q_φ on $\mathit{Inst}(SC, \mathcal{U})$ such that $Q(D) = \{\vec{a} \in \mathit{adom}(D)^n \mid D \models \varphi(\vec{a})\}$. That is, we consider only active domain quantification, and queries that do not extend the active domain.

We define the calculus over $\mathcal{M}$, denoted by $\textsc{Calc}(\mathcal{M})$ simply as $\mathcal{FO}(\mathcal{M})$. More precisely, its expressions are of the form $e = \{\vec{x} \mid \varphi(\vec{x})\}$, where φ is a $\mathcal{FO}_{\text{act}}(\mathcal{M})$ formula. An algebra over $\mathcal{M}$, denoted by $\textsc{Alg}(\mathcal{M})$, contains all the same operations as relational algebra $\textsc{Alg}$; the only difference is the selection predicates. Define *selection terms* by the grammar $st := \#i \mid f(st, \ldots, st)$ where f ranges over the function symbols in Ω. Then, *selection conditions* are given by $sc := C(st, \ldots, st) \mid st = st \mid \neg sc \mid sc \vee sc$, where C ranges over the predicates in Ω. For example, $\sigma_{\#1 * \#2 > \#1 + \#3}(R)$ is an algebra expression that selects triples (x, y, z) from R such that $x * y > x + z$. Similar extensions exist in the literature, see, for example, [22]. A simple extension of the classical equivalence $\textsc{Calc} = \textsc{Alg}$ yields the following.

Proposition 3 *For any $\mathcal{M}$, $\textsc{Calc}(\mathcal{M}) = \textsc{Alg}(\mathcal{M})$.*

Proof sketch. The proof follows the standard proof of the equivalence of calculus and algebra, cf. [1]. The extension of the translation from $\textsc{Alg}(\mathcal{M})$ to $\textsc{Calc}(\mathcal{M})$ is simple: if e is translated into $\varphi(\vec{x})$, then $\sigma_{sc}(e)$ is translated into $\varphi(\vec{x}) \wedge sc(\vec{x})$, where $sc(\vec{x})$ is obtained from sc by replacing $\#i$ with x_i. For the reverse translation, we assume without loss of generality that in formula φ atomic subformulae are separated by Lemma 2. Then we only have to deal with an additional case of $L(\Omega)$ atomic formulae. Given such an atomic formula $\psi(\vec{x})$ where $\vec{x}$ is of length n, we have to find a $\textsc{Alg}(\mathcal{M})$ expression e such that for every D, $e(D) = \{\vec{a} \in \mathit{adom}(D)^n \mid D \models \psi(\vec{a})\}$. Let e' be an $\textsc{Alg}$ expression that constructs the active domain. Then $e(D) = \sigma_{sc}(e'(D) \times \ldots \times e'(D))$, where the product is taken n times, and sc is obtained from ψ by replacing x_i with $\#i$. □

Next, we consider $\text{DATALOG}^\neg(\mathcal{M})$, which extends $\text{DATALOG}^\neg$ by allowing $L(\Omega)$-atomic formulae in the bodies of rules. For example, if $\mathcal{U} = \mathbb{R}$ and Ω contains addition, then the following $\text{DATALOG}^\neg(\mathcal{M})$ program

$$\begin{array}{lcl} R(x,y) & :- & E(x,y), x > y+y \\ R(x,y) & :- & E(x,z), x > z+z, R(z,y) \end{array}$$

defines the transitive closure of a subgraph that consists of the edges (x, y) with $x > 2y$. We assume that $\text{DATALOG}^\neg$ programs are evaluated inflationarily.

Proposition 4 *For any* $\mathcal{M}$*,* $\text{DATALOG}^\neg(\mathcal{M}) = \mathcal{FO}+\mathbf{ifp}(\mathcal{M}) = \mathcal{FO}+\mathbf{lfp}(\mathcal{M})$*.*

Proof. Given a $\mathcal{FO}+\mathbf{ifp}(\mathcal{M})$ formula $\varphi(\vec{x})$, we first transform it into an equivalent formula $\psi(\vec{x})$ such that every atomic subformula is either a $L(SC)$ formula, or a $L(\Omega)$ formula, and every fixpoint formula is applied to a vector of variables, cf. Lemma 2 and the proof of Proposition 1. Let $A = \{\alpha_1(\vec{x}_1), \ldots, \alpha_k(\vec{x}_k)\}$ be the collection of all $L(\Omega)$ atomic subformulae of ψ. Let $T_1, \ldots, T_k$ be new relation symbols of the same arities as $\vec{x}_1, \ldots, \vec{x}_k$. Let $\psi_T(\vec{x})$ be a $L(SC \cup \vec{T})$ formula obtained from ψ by replacing each $\alpha_i(\vec{x}_i)$ with $T_i(\vec{x}_i)$. Finally, for any database D, let $D_{\vec{T}}$ be a $SC \cup \vec{T}$ instance where T_i is interpreted by $\{\vec{a} \in adom(D)^{arity(\vec{x}_i)} \mid D \models \alpha(\vec{x}_i)\}$. Then $D \models \psi(\vec{a})$ iff $D_T \models \psi_T(\vec{a})$ for every $\vec{a} \in adom(D)^n$.

Now, according to [2], there exists a $\text{DATALOG}^\neg$ program P that uses the schema relations and T_i as edb relations such that for any $SC \cup \vec{T}$ instance D, P computes $Q_{\psi_T}(D)$. Thus, if P is preceded by the rules that first define the unary predicate Adom that computes the active domain of the SC relations, and then

$$T_i(x_1, \ldots, x_l) :- \text{Adom}(x_1), \ldots \text{Adom}(x_l), \alpha_i(x_1, \ldots, x_l)$$

for each T_i, then the resulting program P' (in which T_is are now idb relations) computes $Q_\psi(D)$ for any $D \in \mathit{Inst}(SC, \mathcal{M})$. Since ψ and φ are equivalent, we obtain that Q_φ is computed by a $\text{DATALOG}^\neg(\mathcal{M})$ program.

For the reverse inclusion $\text{DATALOG}^\neg(\mathcal{M}) \subseteq \mathcal{FO}+\mathbf{ifp}(\mathcal{M})$, we start with a $\text{DATALOG}^\neg(\mathcal{M})$ program P. Assume without loss of generality that it computes one relation. Construct a new program P' by replacing each $L(\Omega)$-atom $\alpha_i(\vec{x}_i)$ in a rule of P by $T_i(\vec{x}_i)$ where T_i is a new relational symbol. If we define D_T as before, then $P'(D_T) = P(D)$ for every D. By [2], there exists a $\mathcal{FO}+\mathbf{ifp}$ expression φ in the language $L(SC \cup \vec{T})$ such that $Q_\varphi(D) = P'(D)$ for every $D \in \mathit{Inst}(SC \cup \vec{T})$. Let ψ be obtained from φ by replacing each occurrence of $T_i(\vec{z})$ with $\alpha_i(\vec{z})$. A simple induction on the structure of the formula shows that $D \models \psi(\vec{a})$ iff $D_T \models \varphi(\vec{a})$ for each $\vec{a}$ of elements of $adom(D)$. Then, for every $D \in \mathit{Inst}(SC, \mathcal{M})$ we have: $Q_\psi(D) = Q_\varphi(D_T) = P'(D_T) = P(D)$. Thus, P is equivalent to ψ.

Finally, we have to show $\mathcal{FO}+\mathbf{ifp}(\mathcal{M}) \subseteq \mathcal{FO}+\mathbf{lfp}(\mathcal{M})$. The proof follows along the same line: replace each $L(\Omega)$-atomic formulae by a new predicate, and then use the result of [31] to find an equivalent least-fixpoint formula in the extended relational signature. Then replace the new predicates by the $L(\Omega)$ atomic formulae that define them. $\square$

Finally, consider $\text{WHILE}(\mathcal{M})$ which extends WHILE by allowing $\text{ALG}(\mathcal{M})$ expressions in place of ALG expressions. Extending the $\text{WHILE} = \mathcal{FO}+\mathbf{pfp}$ equivalence, we get

Proposition 5 *For any* $\mathcal{M}$*,* $\text{WHILE}(\mathcal{M}) = \mathcal{FO}+\mathbf{pfp}(\mathcal{M})$*.*

Proof sketch. We follow the main idea of the proof of Proposition 4. To translate from $\text{WHILE}(\mathcal{M})$ into $\mathcal{FO}+\textbf{pfp}(\mathcal{M})$, consider a WHILE program P. Get a new WHILE program P' by replacing each occurrence of $\sigma_{sc}(e)$ in P, where e is n-ary, with $e \cap T_{sc}$, where T_{sc} is a new relational symbol. Extend any SC-database D to D_T over a new schema that contains all T_{sc}s by interpreting T_{sc} as a set of all n-ary tuples over the active domain that satisfy sc. Then $P(D) = P'(D_T)$. By [2], there exists a $\mathcal{FO}+\textbf{pfp}$ expression φ such that $P'(D_T) = Q_\varphi(D_T)$. Now, as before, replace T_{sc} in φ by a first-order formula that defines it, to get a $\mathcal{FO}+\textbf{pfp}(\mathcal{M})$ formula ψ such that $Q_\psi(D) = Q_\varphi(D_T) = P'(D_T) = P(D)$. The reverse inclusion is similar. □

We now combine these results with the Ramsey technique to get expressivity bounds on the active-semantics queries over interpreted structures.

Corollary 5

1. *Let $\mathcal{M}$ be an arbitrary ordered infinite structure. Then, for locally generic queries,* $\text{ALG}(\mathcal{M}) = \text{ALG}(<)$*,* $\text{DATALOG}^\neg(\mathcal{M}) = \text{DATALOG}^\neg(<)$*, and* $\text{WHILE}(\mathcal{M}) = \text{WHILE}(<)$*.*

2. *If $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ where Ω consists of real-analytic functions, then every totally generic query in* $\text{ALG}(\mathcal{M})$*,* $\text{DATALOG}^\neg(\mathcal{M})$ *and* $\text{WHILE}(\mathcal{M})$ *is definable in* ALG*,* $\text{DATALOG}^\neg$ *and* WHILE*, respectively. Furthermore, the parity test is not definable in any of these languages.* □

The above results indicate that, with some effort, many of the results for pure calculi extend to active semantics in the presence of arbitrary interpreted structure. However, there are cases where extensions are dependent on specific properties of the interpreted structure, even in the case of active-semantics queries. One area where this happens in query safety over interpreted structures, explored in detail in [13]. We call an active formula $\varphi(\vec{x})$ in $L(SC, \Omega)$ is *safe* for an instance D if $Q_\varphi(D)$ is finite. In [13] it is shown that if $\mathcal{M} = \langle \mathbb{R}, +, *, 0, 1, < \rangle$, then there is a recursive function that takes such a formula $\varphi(\vec{x})$, and outputs another active-semantics formula $\varphi_{\text{safe}}(\vec{z})$ such that $Q_{\varphi_{\text{safe}}}(D)$ equals $Q_\varphi(D)$ if φ is safe for D, and $\emptyset$ otherwise. However, there are structures of $\mathbb{N}$ with all computable predicates in the signature for which no such translation – recursive or not – can exist.

4 Natural semantics

So far we have produced a set of techniques that can be applied to analyze expressiveness and complexity of 'classical' active-semantics queries in a variety of languages. We now want to consider how to approach natural semantics queries, which are essential for constraint databases. It appears that this case is "infinitely" harder than the active case, because now we can ask questions about *any* element in the universe. Thus, the proviso of the previous section that only active formulae are considered and that $\mathbb{L}$ is used in place of $\mathbb{L}_{\text{act}}$ is *not* in force in this section.

The main kind of result we prove here is that for certain structures $\mathcal{M}$, there is no gain in expressiveness by going from the active interpretation to the natural interpretation. That is, we say that a logic $\mathbb{L}$ admits the *natural-active collapse* over $\mathcal{M}$ if $\mathbb{L}(\mathcal{M}) = \mathbb{L}_{\text{act}}(\mathcal{M})$; in other words, for every schema and every $\mathbb{L}$-formula $\varphi(\vec{x})$ in the language $L(SC, \Omega)$, there exists an equivalent active formula $\psi(\vec{x})$ in the same language.

The Hull-Su theorem mentioned in the introduction states that pure $\mathcal{FO}$ admits the natural-active collapse. But this result is not robust: if $\mathcal{N} = \langle \mathbb{N}, +, *, 0, 1 \rangle$, then *every* recursive query is definable in $\mathcal{FO}(\mathcal{N})$ [28], but $\mathcal{FO}_{\text{act}}(\mathcal{N})$ cannot express parity (see Theorem 1); thus $\mathcal{FO}_{\text{act}}(\mathcal{N}) \neq \mathcal{FO}(\mathcal{N})$.

It was further shown that $\mathcal{FO}$ admits the natural-active collapse over $\langle \mathbb{R}, +, -, 0, 1, < \rangle$ [45]. One might ask whether a similar result holds for other structures, and for other logics. Our main goal here is to give positive answers to both questions. We extend the natural-active collapse for structures that have same model-theoretic properties as the real field $\langle \mathbb{R}, +, *, 0, 1, < \rangle$, and to some higher-order logics. In addition to proving such results, we are also interested in finding *algorithms* that convert natural queries into active queries.

We start by giving a new simple proof of the Hull-Su theorem, that can then be extended to show that $\mathcal{FO}^k$ collapses to $\mathcal{FO}^k_{\text{act}}$. We extend this result to infinitary logic. We then present our main result, which is a *constructive* proof that $\mathcal{FO}(\mathcal{M}) = \mathcal{FO}_{\text{act}}(\mathcal{M})$ when $\mathcal{M}$ is o-minimal and admits quantifier elimination; the transformation from a natural-semantics formula into an equivalent active-semantics one is effective if so is the quantifier elimination for the underlying structure. We then extend this result to some higher-order logics. This must be done with care, as for unrestricted fixpoint logics and second-order logics, the set of natural numbers is definable in nice structures such as the real field. However, by restricting them to their hybrid versions, we recover the natural-active collapse. We conclude by giving an example of a structure with a decidable first-order theory over which the natural-active collapse fails. This answers an open question from [28].

4.1 Natural-active collapse in the pure case

Our goal is to have a set of general algorithms for collapsing natural queries to active over interpreted structures. We start with the pure case, and give new algorithms for several logics. We also give a simple constructive proof of the Hull-Su theorem. Theorem 3 refines it to work with variable-bounded logics. These ideas will be expanded upon to deal with interpreted structures. The original proof in [33] is algorithmic but quite complex. In a recent paper [16], a simpler proof is given that uses many-sorted logic. Below we sketch a simple direct proof.

Theorem 2 (Hull-Su) $\mathcal{FO} = \mathcal{FO}_{\text{act}}$.

Proof sketch. The proof is by induction on the structure of the formula. The cases of atomic formulae and Boolean connectives are obvious. For the existential case, we define a transformation $[\gamma]^x$ that eliminates all free occurrences of variable x:

- If γ is $(x = x)$, then $[\gamma]^x = \mathbf{T}$;
- If γ is $(x = y)$ or $R(\ldots, x, \ldots)$, then $[\gamma]^x = \mathbf{F}$;
- If γ is any other atomic formula, then $[\gamma]^x = \gamma$;
- If $\gamma = \gamma_1 \vee \gamma_2$, then $[\gamma]^x = [\gamma_1]^x \vee [\gamma_2]^x$;
- If $\gamma = \neg\gamma'$, then $[\gamma]^x = \neg[\gamma']^x$;
- If $\gamma = \exists y \in adom.\gamma'$, then $[\gamma]^x = \exists y \in adom.[\gamma']^x$.

Let $\varphi(\vec{z}) = \exists x.\alpha(x, \vec{z})$ where $z = (z_1, \ldots, z_n)$. By the hypothesis, α is equivalent to an active formula $\alpha'(x, \vec{z})$. Assume without loss of generality that α' is in prenex form, and x is not a bound variable in any subformula of α'.

Define $\varphi_0(\vec{z}) = \exists x \in adom.\alpha'(x, \vec{z})$, $\varphi_i(\vec{z}) = \alpha'(z_i, \vec{z})$ and $\varphi_\infty(\vec{z}) = [\alpha'(x, \vec{z})]^x$. Let $\varphi'(\vec{z}) = \varphi_0 \vee (\bigvee_{i=1}^n \varphi_i) \vee \varphi_\infty$. Then a straightforward proof shows that $D \models \varphi(\vec{a}) \leftrightarrow \varphi'(\vec{a})$ for every instance D and $\vec{a} \in \mathcal{U}^n$. $\square$

The idea behind this proof can be implemented a bit more carefully to yield the following stronger result:

Theorem 3 *$\mathcal{FO}^k$ and $\mathcal{L}^k_{\infty\omega}$ admit the natural-active collapse: $\mathcal{FO}^k = \mathcal{FO}^k_{\text{act}}$ and $\mathcal{L}^k_{\infty\omega} = (\mathcal{L}^k_{\infty\omega})_{\text{act}}$.*

Proof is again by induction on formulae. We do the proof for infinitary formulae, and then the result for $\mathcal{FO}^k$ follows as a special case (when all disjunctions are finite). The base case of atomic formulae $(y = z)$ or $R(x_1, \ldots, x_p)$ is obvious, and so are the cases of the Boolean connectives as well as infinite disjunction $\bigvee_i \varphi_i$. We extend the definition of $[\gamma]^x$ from the previous proof to infinitary formulae by $[\bigvee_i \gamma_i]^x = \bigvee_i [\gamma_i]^x$.

Now let $\varphi(\vec{z}) = \exists x.\alpha(x, \vec{z})$ where α is, by induction, an active $\mathcal{L}^k_{\infty\omega}$ formula. First, do the following with α: introduce new variables x' and z'_i for each z_i in $\vec{z}$. For every quantifier Qx or Qz_i in α, replace it with Qx' or Qz'_i, and replace all occurrences of x or z_i bound by this quantifier by x' and z'_i. It is easy to see that we get an equivalent formula $\beta(x, \vec{z})$ that uses at most $2k$ variables. Note that in β, x and $\vec{z}$ are not used as bound variables; also, x and $\vec{z}$ do not occur under the scope of quantifiers that bind x' and z'_i.

Below is the key claim, that is proved by straightforward induction on the complexity of the formula.

Claim 1 *Let $\chi(x, \vec{z}, \vec{w})$ be an active $\mathcal{L}^\omega_{\infty\omega}$ formula such that x and $\vec{z}$ are not used as bound variables. Given a D with $A = adom(D)$, a vector $\vec{a}$ over $\mathcal{U}$ and a vector $\vec{c}$ over A, then for any b not in A and not in $\vec{a}$ we have:*

$$D \models \chi(b, \vec{a}, \vec{c}) \quad \textit{iff} \quad D \models [\chi]^x(\vec{a}, \vec{c})$$

Proof of claim: For atomic formulae, it follows from the definition of $[\gamma]^x$ and the fact that b is not in A and not in $\vec{a}$. The proof for the connectives $\neg$ and $\bigvee$ is obvious. Let $\chi(x, \vec{z}, \vec{w}) = \exists v \in adom.\chi_0(x, \vec{z}, \vec{w}, v)$; then $[\chi]^x = \exists v \in adom.[\chi_0]^x$. Fix D with $A = adom(D)$ and $\vec{a}$ and $\vec{c}$ as in the claim. Let b be an arbitrary element not in A and not in $\vec{a}$. Assume that $D \models \chi(b, \vec{a}, \vec{c})$; then $D \models \chi_0(b, \vec{a}, \vec{c}, a_0)$ for some $a_0 \in A$ and by the hypothesis, $D \models [\chi_0]^x(\vec{a}, \vec{c}, a_0)$. Thus, $D \models [\chi]^x(\vec{a}, \vec{c})$. The converse is proved similarly. Thus, the claim is proved.

Going back to the proof for the existential case, we now have a formula $\exists x.\beta(x, \vec{z})$ which is equivalent to $\varphi(\vec{z}) = \exists x.\alpha(x, \vec{z})$. Define $\varphi_0(\vec{z})$ as $\exists x \in adom.\alpha(x, \vec{z})$ and $\varphi_i(\vec{z}) = \alpha(z_i, \vec{z})$. Let

$$\varphi_{\text{act}}(\vec{z}) = \varphi_0(\vec{z}) \vee (\bigvee_{i=1}^{n} \varphi_i(\vec{z})) \vee [\beta]^x(\vec{z}),$$

where $\vec{z} = (z_1, \ldots, z_n)$. Then it follows immediately from the claim above that $D \models \exists x.\beta(x, \vec{z})$ iff $D \models \varphi_{\text{act}}(\vec{z})$; also, φ_{act} is an active formula.

The only problem is that φ_{act} may use extra variables x' and z_i's; in fact, this is so because $[\beta]^x$ uses them. (Note that φ_0 and φ_is do not use these new variables, because we used α instead of β in the definition of these formulae.) But now we look at any quantifier Qx' in $[\beta]^x$ and note (which is immediate from the property of β and the definition of $[\cdot]^x$) that x may not occur in the scope of this quantifier; thus, we replace it back with Qx. Similarly, we replace each Qz_i' with Qz_i because z_i may not occur in the scope of Qz_i' – this again follows from the properties of β and the translation $[\cdot]^x$. Hence, we get an active formula $\beta'(\vec{z})$ that is equivalent to $[\beta]^x(\vec{z})$ and does not use the variables x' and z_i'. That is, β' is a $\mathcal{L}^k_{\infty\omega}$ formula. Now $\varphi'_{\text{act}} = \varphi_0 \vee (\bigvee_i \varphi_i) \vee \beta'$ is an active k-variable formula equivalent to φ. This completes the proof. □

Corollary 6 *Pure $\mathcal{L}^\omega_{\infty\omega}$ admits the natural-active collapse.* □

4.2 Natural-active collapse over interpreted structures: The algorithm

The algorithm for converting natural-semantics queries into active-semantics queries is based on some properties of o-minimal structures, which are summarized next. After that, we present the main technical lemma, which lends itself to an algorithm for converting queries.

Precisely, the statement of the main result of this section is as follows.

Theorem 4 (Natural-active collapse) *Let $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$ be an o-minimal structure that admits quantifier elimination. Then $\mathcal{FO}(\mathcal{M})$ admits the natural-active collapse over $\mathcal{M}$. That is, for every schema SC, and for every $\mathcal{FO}$-formula $\varphi(\vec{x})$ in the language $L(SC, \Omega)$, there exists an equivalent active $\mathcal{FO}$-formula $\varphi_{\text{act}}(\vec{x})$ in the same language. Moreover, if $\mathcal{M}$ is recursive and the quantifier elimination procedure is effective, then the transformation from φ to φ_{act} is effective.* □

O-minimality Recall that an ordered structure $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$ is *o-minimal* if every definable set is a finite union of points and open intervals. Definable sets are those of the form $\{x \in \mathcal{U} \mid \mathcal{M} \models \varphi(x)\}$ where φ is a first-order formula in the language of Ω and constants for elements of $\mathcal{U}$.

Let us make a comment about terminology. In the literature on o-minimality, an interval is given by its endpoints, a and b, and it is either an open interval $(a, b) = \{c \mid a < c < b\}$, or closed $[a, b] = \{c \mid a \leq c \leq b\}$, or one of half-open half-closed versions $[a, b)$ or $(a, b]$. (We also include cases of intervals $(-\infty, b), (-\infty, b], [a, \infty), (a, \infty)$.) Thus, we can always speak of endpoints of intervals. Also, an equivalent definition of o-minimality is that every definable set is a finite union of intervals.

The following classical result establishes uniform bounds on the number of intervals in definable sets.

Fact 1 (see [46]) *If $\mathcal{M}$ is o-minimal, and $\gamma(\vec{y}, x)$ is a first-order formula in the language of $\mathcal{M}$, possibly supplemented with symbols for constants from $\mathcal{M}$, then there is an integer K_γ such that, for each vector $\vec{a}$ from $\mathcal{M}$, the set $\{x \mid \mathcal{M} \models \gamma(\vec{a}, c)\}$ is composed of fewer than K_γ intervals.* □

For every $\gamma(\vec{y}, c)$ in the language of $\mathcal{M}$ and constants, and every $\vec{a}$ over $\mathcal{M}$, $\gamma(\vec{a}, \mathcal{M}) = \{c \in \mathcal{U} \mid \mathcal{M} \models \gamma(\vec{a}, c)\}$ is a finite union of intervals. By the ith interval of $\gamma(\vec{a}, \cdot)$ we shall mean the ith interval of $\gamma(\vec{a}, \mathcal{M})$, in the usual ordering on $\mathcal{U}$.

We shall also need the following easy observations.

Observation 1: For every formula $\gamma(\vec{y}, x)$, and every i, there exists a first-order formula denoted by $\hat{\gamma}_i(\vec{y}, x)$ such that $\mathcal{M} \models \hat{\gamma}_i(\vec{a}, c)$ iff c is in the ith interval of $\gamma(\vec{a}, \cdot)$. In what follows, we always assume that the distinguished variable x is the last one.

Observation 2: If $\mathcal{M}$ is in addition recursive, and quantifier elimination is effective, then K_γ is computable for each γ. Indeed, for each i, write a sentence $\Gamma_i \equiv \exists x \exists \vec{y}\ \hat{\gamma}_i(\vec{y}, x)$ and check if it is true in $\mathcal{M}$, using quantifier-elimination and recursiveness of $\mathcal{M}$. Eventually, we find i such that Γ_i is false; this follows from the Fact above. Thus, K_γ can be taken to be this i.

Observation 3: Since intervals are first-order definable, we can use them in formulae. For example, given a formula $\gamma(\vec{y}, x)$, a number i, and another formula $\beta(\vec{z}, x)$, we can write a first-order formula $\alpha(\vec{y}, \vec{z}, x)$ saying that every x from the ith interval of $\gamma(\vec{y}, \cdot)$ satisfies $\beta(\vec{z}, x)$. This of course is just $\forall x\ (\hat{\gamma}_i(\vec{y}, x) \rightarrow \beta(\vec{z}, x))$, but we shall ocasionally use the interval notation in formulae, to simplify the presentation.

The main lemma Consider a first-order active-semantics formula in the language of $L(SC, \mathcal{M})$:

$$\alpha(\vec{x}, z) \quad \equiv \quad Q_1 y_1 \in adom \ldots Q_m y_m \in adom\ \beta(\vec{x}, \vec{y}, z)$$

where $\beta(\vec{x}, \vec{y}, \vec{z})$ is quantifier-free, and has the following properties:

- every atomic subformula of β is either a $L(SC)$ formula, or a $L(\mathcal{M})$ formula;
- there exists at least one $L(\mathcal{M})$ atomic subformula of β, and $m > 0$, and
- z does not occur in $L(SC)$ subformulae.

Let Ψ be the collection of all $L(\Omega)$ atomic subformulae of β, and their negations.

For formulae $\sigma(\vec{x}, \vec{y}, z)$, $\rho(\vec{x}, \vec{y}, z)$ and $\rho'(\vec{x}, \vec{y}, z)$ from Ψ, $i \leq K_\rho$, and $j \leq K_{\rho'}$, we let $\sigma_{ij}^{\rho\rho'}(\vec{x}, \vec{y}, \vec{s}, \vec{t})$, where $|\vec{s}| = |\vec{t}| = |\vec{y}|$, be the formula defined as follows:

$$\sigma_{ij}^{\rho\rho'}(\vec{x}, \vec{y}, \vec{s}, \vec{t}) \quad \equiv \quad \forall u\ \Big((\hat{\rho}_i(\vec{x}, \vec{s}, u) \wedge \hat{\rho'_j}(\vec{x}, \vec{t}, u)) \rightarrow \sigma(\vec{x}, \vec{y}, u) \Big)\,.$$

Let $\varphi(\vec{x})$ be $\exists z\ \alpha(\vec{x}, z)$.

Lemma 6 *Let D be a nonempty SC-database over $\mathcal{M}$. Let φ, α, β and Ψ be as above. Let $\vec{a}$ be a vector over $\mathcal{U}$. Then $D \models \varphi(\vec{a})$ if and only if there exist $\vec{b}, \vec{c} \in adom(D)^m$, two formulae $\rho(\vec{x}, \vec{y}, z)$ and $\rho'(\vec{x}, \vec{y}, z)$ in Ψ and $i \leq K_\rho, j \leq K_{\rho'}$ such that for the ith interval of $\rho(\vec{a}, \vec{b}, \cdot)$ and the jth interval of $\rho'(\vec{a}, \vec{c}, \cdot)$, denoted by I_0 and I_1 respectively, the following three conditions hold:*

1. *$I_0 \cap I_1 \neq \emptyset$.*
2. *For all $\vec{e} \in adom(D)^m$, and all $c, c' \in I_0 \cap I_1$, we have $\mathcal{M} \models \sigma(\vec{a}, \vec{e}, c) \leftrightarrow \sigma(\vec{a}, \vec{e}, c')$ for every $\sigma \in \Psi$.*

3. *$D \models \alpha'(\vec{b}, \vec{c}, \vec{a})$, where $\alpha'(\vec{s}, \vec{t}, \vec{x})$ is obtained from $\alpha(\vec{x}, z)$ by replacing each subformula $\sigma(\vec{x}, \vec{y}, z)$ from Ψ by $\sigma_{ij}^{\rho\rho'}(\vec{x}, \vec{y}, \vec{s}, \vec{t})$.*

Proof. For the *only if* part, assume that $D \models \varphi(\vec{a})$. That is, $D \models \exists z \alpha(\vec{a}, z)$. Let d witness this; that is, $D \models \alpha(\vec{a}, d)$. For every $\vec{e}$ over $adom(D)$, of the same length as $\vec{y}$, and every atomic $L(\Omega)$ subformula $\rho(\vec{x}, \vec{y}, z)$ of β, we define $I_d(\vec{e}, \rho)$ to be the maximal interval of $\rho(\vec{a}, \vec{e}, \mathcal{M}) = \{c \mid \mathcal{M} \models \rho(\vec{a}, \vec{e}, c)\}$ containing d, in the case when $\mathcal{M} \models \rho(\vec{a}, \vec{e}, d)$, or the the maximal interval of $\neg\rho(\vec{a}, \vec{e}, \mathcal{M})$ containing d, in the case when $\mathcal{M} \models \neg\rho(\vec{a}, \vec{e}, d)$. Let $\mathbf{I}_d$ be the (finite) collection $\{I_d(\vec{e}, \rho) \mid \vec{e} \in adom(D)^{|\vec{y}|}, \rho \in \Psi\}$. Since for each $\vec{e}$ and ρ we have $d \in I_d(\vec{e}, \rho)$, we obtain that $\bigcap \mathbf{I}_d \neq \emptyset$.

Now note that for any finite collection of intervals $I_1, \ldots, I_p$, there are two indices i and j such that $\bigcap_{l=1}^{p} I_l = I_i \cap I_j$. This is because each interval I_l is the intersection $A_l \cap B_l$ of an ideal and coideal in $\langle \mathcal{U}, < \rangle$. Since ideals (and coideals) of a chain form a chain, we get that $\bigcap_{l=1}^{p} A_l = A_i$ and $\bigcap_{l=1}^{p} B_l = B_j$ for some i and j, and from this the equation above follows.

¿From this it follows that there are two intervals, I_0 and I_1 in $\mathbf{I}_d$ such that $I_0 \cap I_1 = \bigcap \mathbf{I}_d$. Let $\vec{b}$ be such that I_0 is the ith interval of $\rho(\vec{a}, \vec{b}, \mathcal{M})$, and $\vec{c}$ be such that I_1 is the jth interval of $\rho'(\vec{a}, \vec{c}, \mathcal{M})$, where $\rho, \rho' \in \Psi$ (that is, ρ, ρ' are either atomic $L(\Omega)$ subformulae of φ, or negations of such atomic subformulae).

Let $\vec{e} \in adom(D)^{|\vec{y}|}$. Pick any $\sigma \in \Psi$ and any $c, c' \in I_0 \cap I_1$. Since $I_0 \cap I_1 = \bigcap \mathbf{I}_d$, we obtain that $c, c' \in I_0 \cap I_1 \subseteq I_d(\vec{e}, \sigma)$, which implies $\mathcal{M} \models \sigma(\vec{a}, \vec{e}, c) \leftrightarrow \sigma(\vec{a}, \vec{e}, c')$. This proves conditions 1 and 2 in the Lemma. (Note that the assumptions $D \models \varphi(\vec{a})$ was not used to prove these two conditions.)

To prove condition 3, notice that for every $L(\Omega)$ atomic subformula $\sigma(\vec{x}, \vec{y}, z)$ of φ, and for every $\vec{e} \in adom(D)^{|\vec{y}|}$, we have

$$\sigma(\vec{a}, \vec{e}, d) \quad \leftrightarrow \quad \forall u \in I_0 \cap I_1.\ \sigma(\vec{a}, \vec{e}, u)$$

since $I_0 \cap I_1 = \bigcap \mathbf{I}_d$.

Now, for any subformula $\gamma(\vec{x}, \vec{y}, z)$ of $\alpha(\vec{x}, z)$, let $\gamma'(\vec{s}, \vec{t}, \vec{x}, \vec{y})$ be the result of replacing each $\sigma(\vec{x}, \vec{y}, z)$ from Ψ by $\sigma_{ij}^{\rho\rho'}(\vec{x}, \vec{y}, \vec{s}, \vec{t})$.

We can now restate the above equivalence as:

$$(*) \qquad D \models \sigma(\vec{a}, \vec{e}, d) \leftrightarrow \sigma'(\vec{a}, \vec{e}, \vec{b}, \vec{c})$$

for every $\vec{e} \in adom(D)^{|\vec{y}|}$ (where $\vec{b}$ and $\vec{c}$ are the tuples necessary to define $I_0 \cap I_1$ above), where $\sigma(\vec{x}, \vec{y}, z)$ is atomic or negated atomic (i.e. $\sigma \in \Psi$).

But the above equivalence is preserved under Boolean combinations and active quantification over variables from $\vec{y}$ in σ, hence we obtain $(*)$ for every σ that is a subformula of α. Finally, this gives us

$$D \models \alpha(\vec{a}, d) \leftrightarrow \alpha'(\vec{a}, \vec{b}, \vec{c}).$$

Since $D \models \alpha(\vec{a}, d)$, we conclude $D \models \alpha'(\vec{a}, \vec{b}, \vec{c})$, proving 3).

To prove the *if* part, assume that there exist $\vec{b}, \vec{c} \in adom(D)^m$, $\rho, \rho' \in \Psi$, and $i \leq K_\rho, j \leq K_{\rho'}$ such that for I_0, I_1 defined as in the statement of the Lemma, conditions 1, 2, and 3 hold. Let d be an arbitrary element of $I_0 \cap I_1$. We claim that $D \models \alpha(\vec{a}, d)$, thus proving $D \models \varphi(\vec{a})$.

Indeed, for every $L(\Omega)$ atomic subformula $\sigma(\vec{x}, \vec{y}, z)$ of α, we have

$$\sigma(\vec{a}, \vec{e}, d) \quad \leftrightarrow \quad \forall u \in I_0 \cap I_1.\ \sigma(\vec{a}, \vec{e}, u)$$

for every $\vec{e}$ over $adom(D)$ – this follows from 2. That is, $\sigma(\vec{a}, \vec{e}, d) \ \leftrightarrow\ \sigma_{ij}^{\rho\rho'}(\vec{a}, \vec{e}, \vec{b}, \vec{c})$. As before, since this equivalence is preserved under Boolean combinations with $L(SC)$ atomic formula, and under active-domain quantification over variables from $\vec{y}$, we obtain

$$D \models \alpha(\vec{a}, d) \leftrightarrow \alpha'(\vec{b}, \vec{c}, a)$$

thus proving $D \models \alpha(\vec{a}, d)$. The lemma is proved. □

Remark Note that it is not required that α be in prenex normal form. The proof will work for any formula α that satisfies the following properties: every atomic subformula is either an $L(\Omega)$ formula, or $L(SC)$ formula, z only occurs in $L(\Omega)$ atomic formulae, and variables used in active-domain quantification in α are all distinct. It is clear that any active-semantics formula can be converted into an equivalent formula that satisfies these requirements.

The algorithm The algorithm that converts natural-semantics formulae into active-semantics formulae works by induction on the structure of the formula. To eliminate an existential quantifier $\exists z \in \mathcal{U}$, we use the lemma above, by translating its statement into first-order logic. We then notice that in the resulting translation, every quantifier not bounded by the active domain occurs only in subformulae which do not mention the SC predicates, that is, in $L(\Omega)$ subformulae. Since $\mathcal{M}$ admits quantifier elimination, this implies that every such occurrence of unbounded quantification can be eliminated, thus giving us an active-semantics formula.

The algorithm NATURAL–ACTIVE is shown on the next page. We can state the following.

Proposition 6 *Let $\mathcal{M}$ be o-minimal and admit quantifier-elimination. Let $\varphi(\vec{x})$ be any $L(SC, \Omega)$ first-order formula, and let $\varphi_{\rm act}$ be the output of* NATURAL–ACTIVE *on φ. Then, for every nonempty $D \in Inst(SC, \mathcal{M})$, $D \models \forall\vec{x}.\ \varphi(\vec{x}) \leftrightarrow \varphi_{\rm act}(\vec{x})$. Furthermore, if $\mathcal{M}$ is recursive and the quantifier-elimination procedure is effective, then all steps 4.1 – 4.10 are effective.*

Proof. This follows straightforwardly from Lemma 6, as Step 4 just encodes the conditions of the lemma in first-order logic, the quantifier-free formulae produced in steps 4.4, 4.5, and 4.8 exist due to quantifier-elimination in $\mathcal{M}$, and the existence of K follows from Fact 1. For Step 4.1, any active-semantics formula can be transformed into the required form by taking a conjunction with $\forall x \in adom.x = x$ to guarantee $m > 0$ and the existence of at least one atomic $L(\Omega)$ formula (notice that this does not change the truth value on *nonempty* databases).

The effectiveness follows from the effectiveness of quantifier-elimination, and computability of K_γ for each γ over a recursive $\mathcal{M}$ admitting effective quantifier-elimination. □

To conclude the proof of Theorem 4, we have to deal with the case of $adom(D)$ being empty. Let $\varphi(\vec{x})$ be a first-order $L(SC, \Omega)$ formula. Let $\varphi'_\emptyset(\vec{x})$ be obtained from φ by replacing each occurrence of $R(\cdots)$, where $R \in SC$, by *false*. Note that $\varphi'_\emptyset$ is a $L(\Omega)$ formula. Let $\varphi_\emptyset$ be a quantifier-free formula equivalent to $\varphi'_\emptyset$. A simple induction on formulae shows that for the empty SC-instance, $\emptyset_{SC}$, it is the case that $\emptyset_{SC} \models \varphi(\vec{a})$ iff $\mathcal{M} \models \varphi_\emptyset(\vec{a})$, for every $\vec{a}$. Thus, an active-semantics formula

$$\varphi'(\vec{x}) \quad \equiv \quad [(\exists x \in adom.\ x = x) \ \wedge\ \varphi_{\rm act}(\vec{x})] \ \vee\ [(\neg\exists x \in adom.\ x = x) \ \wedge\ \varphi_\emptyset(\vec{x})],$$

Algorithm NATURAL–ACTIVE

INPUT: $L(SC,\Omega)$ formula $\varphi(\vec{x})$
OUTPUT: $L(SC,\Omega)$ active-semantics formula $\varphi_{\text{act}}(\vec{x})$

1. If φ is an atomic formula, then $\varphi_{\text{act}} = \varphi$.
2. If $\varphi = \psi * \chi$, then $\varphi_{\text{act}} = \psi_{\text{act}} * \chi_{\text{act}}$ where $* \in \{\vee, \wedge\}$; if $\varphi = \neg\psi$, then $\varphi_{\text{act}} = \neg\psi_{\text{act}}$.
3. If $\varphi = \exists x \in adom\ \psi$, then $\varphi_{\text{act}} = \exists x \in adom\ \psi_{\text{act}}$.
4. Let $\varphi(\vec{x}) = \exists z\ \alpha^0(\vec{x}, z)$.

4.1 Let $\alpha(\vec{x}, z)$ be a formula equivalent to α^0_{act} which is of the form

$$Q_1 y_1 \in adom \ldots Q_m y_m \in adom\ \beta(\vec{x}, \vec{y}, z)$$

where $\beta(\vec{x}, \vec{y}, \vec{z})$ is quantifier-free, and has the following properties: every atomic subformula of β is either a $L(SC)$ formula, or a $L(\mathcal{M})$ formula; there exists at least one $L(\mathcal{M})$ atomic subformula of β, $m > 0$, and z does not occur in $L(SC)$ subformulae.

4.2 Let Ψ be the collection of all atomic $L(\Omega)$ subformulae of α, and their negations.

4.3 Let $K = \max_{\gamma \in \Psi} K_\gamma$.

4.4 For every pair of formulae $\rho, \sigma \in \Psi$, and every $i, j < K$, define $\chi^{\rho\sigma}_{ij}(\vec{x}, \vec{s}, \vec{t})$ to be the quantifier-free $L(\Omega)$ formula equivalent to

$$\exists u.\ \hat{\rho}_i(\vec{x}, \vec{s}, u) \wedge \hat{\sigma}_j(\vec{x}, \vec{t}, u)$$

Note that $|\ \vec{s}\ | = |\ \vec{t}\ | = m$.

4.5 For each $\rho, \sigma \in \Psi$, each $i, j < K$, and each $\tau \in \Psi$, define $\tau^{\rho\sigma}_{ij}(\vec{x}, \vec{y}, \vec{s}, \vec{t})$ as a quantifier-free formula equivalent to

$$\forall u.\ \hat{\rho}_i(\vec{x}, \vec{s}, u) \wedge \hat{\sigma}_j(\vec{x}, \vec{t}, u) \rightarrow \tau(\vec{x}, \vec{y}, u)$$

4.6 For each $\rho, \sigma \in \Psi$, each $i, j < K$, define $\alpha^{\rho\sigma}_{ij}(\vec{x}, \vec{s}, \vec{t})$ as α in which every $L(\Omega)$ atomic subformula $\tau(\vec{x}, \vec{y}, z) \in \Psi$ is replaced by $\tau^{\rho\sigma}_{ij}(\vec{x}, \vec{y}, \vec{s}, \vec{t})$.

4.7 Let $\text{same}_\beta(\vec{x}, \vec{r}, u, v)$ be

$$\bigwedge (\rho(\vec{x}, \vec{r}, u) \leftrightarrow \rho(\vec{x}, \vec{r}, v))$$

where the conjunction is taken over all the $L(\Omega)$ atomic subformulae ρ of β.

4.8 For each $\rho, \sigma \in \Psi$, each $i, j < K$, define $\eta^{\rho\sigma}_{ij}(\vec{x}, \vec{s}, \vec{t}, \vec{r})$ as a quantifier-free formula equivalent to

$$\forall u, v.\ (\hat{\rho}_i(\vec{x}, \vec{s}, u) \wedge \hat{\sigma}_j(\vec{x}, \vec{t}, u) \wedge \hat{\rho}_i(\vec{x}, \vec{s}, v) \wedge \hat{\sigma}_j(\vec{x}, \vec{t}, v)) \rightarrow \text{same}_\beta(\vec{x}, \vec{r}, u, v)$$

4.9 For each $\rho, \sigma \in \Psi$, each $i, j < K$, define $\pi^{\rho\sigma}_{ij}(\vec{x}, \vec{s}, \vec{t})$ as $\forall \vec{r} \in adom.\ \eta^{\rho\sigma}_{ij}(\vec{x}, \vec{s}, \vec{t}, \vec{r})$.

4.10 Output, as $\varphi_{\text{act}}(\vec{x})$, the formula

$$\exists \vec{s} \in adom\ \exists \vec{t} \in adom.\ \bigvee_{\rho,\sigma \in \Psi} \bigvee_{i,j < K} (\chi^{\rho\sigma}_{ij}(\vec{x}, \vec{s}, \vec{t}) \wedge \pi^{\rho\sigma}_{ij}(\vec{x}, \vec{s}, \vec{t}) \wedge \alpha^{\rho\sigma}_{ij}(\vec{x}, \vec{s}, \vec{t}))$$

where $\varphi_{\rm act}$ is produced by the algorithm NATURAL–ACTIVE, has the property that $D \models \forall \vec{x}.\ \varphi(\vec{x}) \leftrightarrow \varphi'(\vec{x})$, for every $D \in \mathit{Inst}(SC, \mathcal{M})$. This concludes the proof of Theorem 4. $\square$

From Tarski's quantifier-elimination, Theorem 4 and locally generic collapse, we get:

Corollary 7 *a) There is an algorithm that converts any $\mathcal{FO}(\langle \mathbb{R}, +, *, 0, 1, < \rangle)$-query into an equivalent $\mathcal{FO}_{\rm act}(\langle \mathbb{R}, +, *, 0, 1, < \rangle)$-query.*

b) Parity test cannot be defined as a relational calculus query with polynomial inequality constraints over the reals. $\square$

It follows from Lemma 6 that one only needs quantifier-elimination in Steps 4.4, 4.5 and 4.8 to obtain quantifier-free formulae equivalent to certain $L(\Omega)$ formulae. If we leave those formulae intact, we obtain, instead of a formula $\varphi_{\rm act}$, a different formula φ' with the properties that it is equivalent to φ, and that in every subformula $\exists x \gamma$ or $\forall x \gamma$ of φ', γ is a $L(\Omega)$ formula. From this, we conclude:

Corollary 8 (Natural Generic Collapse) *Let $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$ be an o-minimal structure. Then for every locally generic query $\varphi(\vec{x})$ in $\mathcal{FO}(\mathcal{M}, SC)$, there exists an equivalent active-semantics first-order query $\psi(\vec{x})$ in the language $L(<, SC)$.*

Proof. Indeed, consider φ' as above, and for every $L(\Omega)$ subformula γ of φ', define a predicate P_γ by letting $P_\gamma(\vec{x})$ iff $\models \gamma(\vec{x})$. Let Ω' expand Ω by all such predicates P_γ, and let $\mathcal{M}'$ be the expansion of $\mathcal{M}$ to Ω'. Thus, over $\mathcal{M}'$, φ' is equivalent to an active-semantics formula φ'' in the extended language, and hence, by the locally generic collapse, to an active $L(SC, <)$-formula ψ. It then follows, since $\mathcal{M}'$ is a definable extension of $\mathcal{M}$, that φ is equivalent to ψ over $\mathcal{M}$. $\square$

4.2.1 Counterexamples to collapse results for first-order logic

We have shown that first-order logic behaves particularly nicely with respect to natural quantification when we make some restrictions on the structure. O-minimality is one such restriction, and we have shown above that this leads to natural-active collapse, and hence to expressive bounds on generic queries. Are restrictions on the interpreted structure, such as o-minimality, necessary in order to get such results? We have mentioned earlier that over the structure $\mathcal{N} = \langle \mathbb{N}, +, *, < \rangle$, parity (and in fact every computable query) is definable by a natural semantics sentence [28]. Thus, the natural-active collapse and generic collapse both fail for this structure.

However, the structure $\mathcal{N}$ is highly undecidable, and one can ask whether there are more tractable structures for which collapse results and expressivity bounds fail. In [28] it was conjectured that the parity query cannot be defined by a natural semantics sentence over any structure $\mathcal{M}$ with a *decidable theory*. This conjecture was further studied in [6, 7].

The conjecture seems likely to be false, as an example of a structure with decidable theory for which the natural-active collapse fails was given in [43]. Namely, [43] considered the random graph, and showed that with natural quantification one can simulate monadic second-order logic. However, to express parity in monadic second-order, one seems to needs a linear order on the universe, and it is known that no infinite linear order is definable in the random graph. We now show how to circumvent this problem, and produce a structure with a decidable first-order theory for which parity is expressible as natural-semantics query.

More specifically, let $\mathcal{RT} = \langle \mathcal{U}, R\rangle$ be the random ternary relation on a countably infinite set $\mathcal{U}$: that is, any model that satisfies every sentence that is true in almost all finite 3-hypergraphs. Here 'almost all' is with respect to the uniform probability distribution: $R(a,b,c)$ holds of nodes a,b,c with probability one half, independently for each triple a,b,c. It is known [24] that the set of all such sentences forms a complete theory with infinite models, and that this theory is decidable.

Proposition 7 *There is a first-order natural semantics sentence that defines parity over $\mathcal{RT}$.*

Proof: A model of the random ternary relation has the property [24] that for every quantifier-free formula $\varphi(\vec{x},\vec{y})$ that is consistent in predicate logic, we have

$$\mathcal{RT} \models \forall\vec{x}\ \exists\vec{y}\ \varphi(\vec{x},\vec{y}).$$

We next consider, for every $1 \le k \le n < \omega$, a formula $\varphi_{n,k}(x_1,\ldots,x_n,y_1,\ldots,y_n,x,y,w)$ defined as

$$[(x,y)=(x_1,y_1)\vee\ldots\vee(x,y)=(x_n,y_n)] \to ([(x,y)=(x_1,y_1)\vee\ldots\vee(x,y)=(x_k,y_k)] \leftrightarrow R(w,x,y))$$

where $(x,y)=(x',y')$ is an abbreviation for $(x=x')\wedge(y=y')$. Applying the above, we see that

$$\mathcal{RT}\ \models\ \forall x_1,\ldots,x_n\forall y_1,\ldots,y_n\forall x\forall y\exists w\ \varphi_{n,k}(\vec{x},\vec{y},x,y,w).$$

Thus, for every finite collection S of ordered pairs from $\mathcal{U}$, and any $T\subset S$ there is a $c\in\mathcal{U}$ such that $\forall(a,b)\in S.\ ((a,b)\in T\leftrightarrow R(c,a,b))$.

For any $c\in\mathcal{U}$ and set $A\subset\mathcal{U}$, let $R_{c,A}$ be $\{(a,b)\in A^2 : R(c,a,b)\}$. The property above says exactly that every finite collection of pairs from A is of the form $R_{a,A}$ for some $c\in\mathcal{U}$.

Now consider a database schema SC that has one unary relation R_0, and the query q that returns true of database $D\in \mathit{Inst}(SC,\mathcal{RT})$ (that is, a finite subset of $\mathcal{U}$) exactly when there exists $c\in\mathcal{U}$ such that the binary relation $R_{c,adom(D)}$ defines a partial function on $adom(D)$, and this partial function maps its domain bijectively onto the complement of its domain within $adom(D)$. It is easy to show that this is expressible as the first-order query:

$$\exists w\forall y\in adom\forall z\in adom\forall y'\in adom\ \left(\begin{array}{cc} & \exists u\in adom\ (R(w,y,u)\vee R(w,u,y))\\ \wedge & (y'\neq y\to\neg(R(w,y,z)\wedge R(w,y',z))\\ \wedge & (y'\neq y\to\neg(R(w,z,y)\wedge R(w,z,y'))\end{array}\right)$$

It is clear that if $q(D)$ is true, then the parity of $adom(D)$ is even, since $R_{c,adom(D)}$ witnesses this. From the property mentioned in the first paragraph we see that any bijection on $adom(D)$ is of the form $R_{c,adom(D)}$ for some c in $\mathcal{U}$, and hence q is implied by the parity query. Hence parity is expressible as a first-order formula over $\mathcal{RT}$. $\square$

Corollary 9 *(see also [43]) There exists a structure $\mathcal{M}$ with a decidable first-order theory such that the natural-active collapse fails for first-order queries over $\mathcal{M}$.* $\square$

4.3 Natural-active collapse for higher-order logics

Note that even for very well-behaved structures, one cannot guarantee the natural-active collapse for the full second-order logic:

Proposition 8 *Let $\mathcal{M}$ be $\langle \mathbb{R}, +, *, 0, 1, < \rangle$. Then second-order logic does not have the natural-active collapse over $\mathcal{M}$. This continues to hold for any fragment of second-order that allows existential quantifiers over unary predicates (e.g., existential second-order, monadic Σ^1_1).*

Proof. The set of natural numbers can be defined by a second-order formula with one unary existential second order quantifier in the language of $+, 0, <$:

$$\varphi(n) \quad = \quad \exists P.[P(0) \wedge (\forall x.P(x) \to (x = 0 \vee P(x-1))) \wedge (\forall x \forall y.P(x) \wedge x < y < x+1 \to \neg P(y))] \ \wedge \ P(n).$$

Then it follows from [28] that any total recursive query on databases whose active domain consists only of natural numbers is in $\mathcal{SO}(\mathbb{R}, +, *, 0, 1, <)$. On the other hand, every locally generic active-semantics query in $\mathcal{SO}(\mathbb{R}, +, *, 0, 1, <)$ is expressible in $\mathcal{SO}$ over $L(SC, <)$ and thus has *PSPACE* data complexity, which proves the proposition. □ □

An analagous result could be given for fixpoint logics. However, we show below that for *hybrid* logics, the natural-active collapse can easily be recovered.

Theorem 5 *Let $\mathcal{M}$ be an o-minimal structure that admits quantifier elimination. Then $\mathcal{FO}+$**ifp**, $\mathcal{FO}+$**lfp**, $\mathcal{FO}+$**pfp** and $\mathcal{HSO}$ admit the natural-active collapse over $\mathcal{M}$. Moreover, the transformation from a natural formula to an active formula is effective if $\mathcal{M}$ is recursive and the quantifier elimination procedure is effective.*

Proof. We use an inductive argument on complexity, as in the proof of Theorem 4. We will give a single algorithm that witnesses the natural-active transformation for all of the logics above. In order to have this algorithm work for fixed point logic, we will maintain the following invariant in this algorithm: if a relation symbol (fixed or bound) occurs positively in φ, it occurs positively in φ_{act}.

The algorithms for atomic formulae and the boolean connectives are identical to the first-order case. Since the second-order and fixed point operators are done under the active interpretation, the induction for these operators themselves is trivial:

If $\varphi(\vec{y}, \vec{t}) = [\text{LFP}_{\vec{x},S}\alpha(\vec{x}, \vec{y}, S)](\vec{t})$ then $\varphi_{\text{act}} = [\text{LFP}_{\vec{x},S}\alpha_{\text{act}}(\vec{x}, \vec{y}, S)](\vec{t})$.

The resulting formula is well-formed given that, by induction, positivity is preserved by the transformations given here.

If $\varphi(\vec{y}, \vec{t}) = [\text{IFP}_{\vec{x},S}\alpha(\vec{x}, \vec{y}, S)](\vec{t})$ then $\varphi_{\text{act}} = [\text{IFP}_{\vec{x},S}\alpha_{\text{act}}(\vec{x}, \vec{y}, S)](\vec{t})$, and

If $\varphi(\vec{y}, \vec{t}) = [\text{PFP}_{\vec{x},S}\alpha(\vec{x}, \vec{y}, S)](\vec{t})$ then $\varphi_{\text{act}} = [\text{PFP}_{\vec{x},S}\alpha_{\text{act}}(\vec{x}, \vec{y}, S)](\vec{t})$

Finally, for hybrid second order we have the analogous rule:

If $\varphi(\vec{x}) = \exists S \in adom\ \alpha(\vec{x})$, then $\varphi_{\text{act}} = \exists S \in adom\ \alpha_{\text{act}}(\vec{x})$.

Note that the inductive rules for boolean connectives, fixed-point, and second-order quantification all preserve the invariant. It suffices, then, to specify how the algorithm deals with converting a formula of the form $\varphi(\vec{x}) = \exists z \alpha(\vec{x}, z)$ to an active formula φ_{act} , given that we know how to convert $\alpha(\vec{x}, z)$ to an equivalent active-semantics $\alpha_{\text{act}}(\vec{x}, z)$.

We have that $\alpha(\vec{x}, z)$ is an active formula which possibly has free predicate symbols T from outside of Ω in it. We normalize it so that: every atomic subformula of α is either a $L(SC')$ formula — where $SC' = SC \cup T$ — or an $L(\Omega)$ formula, so that there exists at least one $L(\Omega)$ atomic subformula of β, and so that z does not occur in $L(SC')$ subformulae. It is easy to see that we can do this without effecting the positivity of any SC' relation in any subformula of α. We now perform the exact same computation on α as we did in the algorithm of Theorem 4. Namely, we let Ψ be the collection of all atomic $L(\Omega)$ subformulae of α, and their negations, and let $K = \max_{\gamma \in \Psi} K_\gamma$, $\chi_{ij}^{\rho\sigma}(\vec{x}, \vec{s}, \vec{t})$ and $\tau_{ij}^{\rho\sigma}(\vec{x}, \vec{y}, \vec{s}, \vec{t})$ be defined exactly as in 4.4 of the first-order algorithm. Note that these last are all $L(\Omega)$ formulae, and they are defined without reference to the relational operators in α.

We now define $\alpha_{ij}^{\rho\sigma}(\vec{x}, \vec{s}, \vec{t})$ as the result of replacing in α every $L(\Omega)$ atomic subformula $\tau(\vec{x}, \vec{y}, z) \in \Psi$ by $\tau_{ij}^{\rho\sigma}(\vec{x}, \vec{y}, \vec{s}, \vec{t})$, just as in 4.6. Note that since we are only substituting $L(\Omega)$ subformulae by other $L(\Omega)$ formulae, positivity of SC' predicates is again not affected.

The formulae $\text{same}_\beta(\vec{x}, \vec{r}, u, v)$, $\eta_{ij}^{\rho\sigma}(\vec{x}, \vec{s}, \vec{t}, \vec{r})$, $\pi_{ij}^{\rho\sigma}(\vec{x}, \vec{s}, \vec{t})$ are syntactically identical to the formulae defined in steps 4.7-4.9 of the first-order algorithm.

Finally, we output as φ_{act}, as in 4.10, the formula

$$\exists \vec{s} \in adom\ \exists \vec{t} \in adom. \exists \vec{s} \bigvee_{\rho,\sigma \in \Psi} \bigvee_{i,j<K} (\chi_{ij}^{\rho\sigma}(\vec{x}, \vec{s}, \vec{t}) \wedge \pi_{ij}^{\rho\sigma}(\vec{x}, \vec{s}, \vec{t}) \wedge \alpha_{ij}^{\rho\sigma}(\vec{x}, \vec{s}, \vec{t})).$$

The proof that this algorithm is correct results from showing that Lemma 6 still holds when φ is allowed to contain active second-order operators (fixed-point, second-order existential, etc.). This in turn follows simply by noting that the proof of Lemma 6 goes through verbatim in the presence of the second-order operators: the only thing necessary is to confirm (immediately from the definition of the σ to σ' transformation) that the equivalence $(*)$ in Lemma 6 is preserved under active second-order quantification, and that $(*)$ is also preserved under the fixpoint operations. $\square$

Recall that by *closedness* of a fixpoint formula we mean that no application of a fixpoint operator involves extra free variables, that is, it is of the form $[\text{LFP}_{\vec{x},S}\varphi(\vec{x}, S)](\vec{t})$, or similarly for IFP and PFP. As mentioned in Section 2, it is known that the closed normal form holds when all quantifiers are restricted to be active-semantics, or when all quantifiers are restricted to be natural. ¿From the normal form for active queries, plus the previous collapse result, we see that hybrid fixpoint queries can also be converted to closed normal form, provided that the underlying structure is nice.

Corollary 10 (Normal form for fixpoint) *If $\mathcal{M}$ is o-minimal and has quantifier elimination, then*

$$\begin{array}{rcl} \mathcal{FO}+\mathbf{lfp}(\mathcal{M}) & = & closed\ \mathcal{FO}+\mathbf{lfp}(\mathcal{M}) \\ \mathcal{FO}+\mathbf{ifp}(\mathcal{M}) & = & closed\ \mathcal{FO}+\mathbf{ifp}(\mathcal{M}) \\ \mathcal{FO}+\mathbf{pfp}(\mathcal{M}) & = & closed\ \mathcal{FO}+\mathbf{pfp}(\mathcal{M}). \end{array}$$

As a second application, we see that hybrid second-order formulae over o-minimal structures (note the absence of the quantifier-elimination requirement) can be converted into the normal form defined in Section 2.

Corollary 11 (Normal form for $\mathcal{HSO}$) *If $\mathcal{M}$ is o-minimal, then every $\mathcal{HSO}(\mathcal{M}, SC)$ is equivalent to a $\mathcal{HSO}(\mathcal{M}, SC)$ formula in the normal form.*

Proof. Let $\mathcal{M}'$ be a definitional expansion of $\mathcal{M}$ that admits quantifier-elimination. Let φ be a $\mathcal{HSO}(\mathcal{M}, SC)$ formula. Then it is equivalent to an active $\mathcal{HSO}(\mathcal{M}', SC)$ formula φ'. Convert φ' into the normal form, and replace each new symbol in $\mathcal{M}'$ by its $\mathcal{M}$-definition. Since those occur only in the first-order part, we can convert the result into a normal form $\mathcal{HSO}(\mathcal{M}, SC)$ formula. $\Box$

Analogous results can be proved showing the tame behavior of other fragments of second-order logics via the same method, provided that the second-order constructs are again given the active interpretation.

5 Complexity applications and summary

In this section, we use the results on both natural and active-semantics queries to derive their data complexity. Recall that in order to define data complexity of a query Q, one first defines a language $L_Q = \{enc(D)\#enc(t) \mid t \in Q(D)\}$ where *enc* is an encoding of tuples and databases in some finite alphabet Σ, and $\#$ is a special marker. Then, for a complexity class $\mathcal{C}$, we say that the data complexity of Q is $\mathcal{C}$ if $L_Q \in \mathcal{C}$; similarly, the data complexity of a language is $\mathcal{C}$ if so is the data complexity of every query in the language.

As we need to encode databases over interpreted structures, in particular, the reals, we restrict our attention to the case when database elements are integers, as in [29]. Equivalently, we could also assume that databases store rational numbers represented as pairs of integers. In such a setting, two results are known. First, the data complexity of $\mathcal{FO}(\langle \mathbb{R}, +, -, 0, 1, < \rangle)$ is AC^0 [29]. (Recall that AC^0 is the class of problems definable with unbounded fan-in constant depth circuits that use *and*, *or* and *not* gates, and the number of gates is polynomial in the size of input, cf. [15].) This is the same as the bound on data complexity of $\mathcal{FO}_{\mathrm{act}}(<)$, cf. [1]. Second, the data complexity of $\mathcal{FO}(\langle \mathbb{R}, +, *, 0, 1, < \rangle)$ is NC [36].

Note that the AC^0 bound of [29] implies some expressivity bounds, such as inexpressibility of parity in $\mathcal{FO}(\langle \mathbb{R}, +, -, 0, 1, < \rangle)$, as parity is not in AC^0 [25]. We also know that the data complexity of $\mathcal{FO}(\langle \mathbb{R}, +, *, 0, 1, < \rangle)$ cannot be AC^0, since multiplication is not in AC^0 [25]. Still, we are able to lower the bound of [36]. In fairness to [36], we remark here that their bound applies to finitely-representable databases as well, and so does the bound of [29]. We only deal with finite databases here.

The class TC^0 extends AC^0 by allowing threshold gates, or equivalently majority gates [5, 44]. It is known that $\mathrm{AC}^0 \subset \mathrm{TC}^0 \subseteq NC^1 \subseteq L \subseteq NL \subseteq NC$ and all $\subseteq$ inclusions are conjectured to be strict [5].

Proposition 9 *Every query in $\mathcal{FO}(\langle \mathbb{R}, +, *, 0, 1, < \rangle)$ has TC^0 data complexity.*

Proof: Let $\mathcal{M} = \langle \mathbb{R}, +, *, 0, 1, < \rangle$. Fix a database schema SC, and let $\varphi'(\vec{x})$ be a query in $\mathcal{FO}(\langle \mathbb{R}, +, *, 0, 1, < \rangle)$. By the natural-active collapse, there exists an equivalent active-semantics query $\varphi(\vec{x})$. Let S be a relational symbol, not in SC, of arity $|\vec{x}|$, and let Φ be the sentence $\forall \vec{x}\ (S(\vec{x}) \to \varphi(\vec{x}))$. Let D be a SC database, and t a tuple of arity $|\vec{x}|$. Define a $SC \cup \{S\}$ database D' by interpreting S as $\{t\}$. Then $D' \models \Phi$ iff $D \models \varphi(t)$. Thus, it suffices to show that $enc(D')$ for $D' \models \Phi$ can be recognized in

TC^0. If we can show this, then to recognize $enc(D)\#enc(t)$ one transforms it into $enc(D')$ in constant time (by replacing the marker # by the marker used for separating relations) and checks if $D' \models \Phi$.

We next show that there is an expression e of $\text{ALG}(\langle \mathbb{R}, +, *, 0, 1, < \rangle)$ such that $e(D') = \{()\}$ if $D' \models \Phi$ and $e(D') = \{\}$ if $D' \models \neg\Phi$, where $()$ is the empty tuple. To do this, transform Φ to a prenex active-semantics sentence $Q_1 y_1 \ldots Q_n y_n\ \alpha(\vec{y})$ where α is quantifier-free; we also assume, as before, that every atomic subformula of α is either a $L(SC \cup \{S\})$ formula or a $L(\mathcal{M})$ formula.

Let ADOM be a relational algebra expression such that for every D, $\text{ADOM}(D)$ evaluates to $adom(D)$. We now translate subformulae of Φ into algebra expressions as follows. Each subformula $R(y_{i_1}, \ldots, y_{i_k})$ is translated as follows. Let e_R be $e_R^1 \times \ldots \times e_R^n$, where e_R^l is ADOM if $l \notin \{i_1, \ldots, i_k\}$ and e_R^l is $\Pi_{\#s}(R)$ if $l = i_s$. Let c be the conjunction of $\#i_s = \#(s+n)$ for all $s = 1, \ldots, k$. Then the translation of $R(y_{i_1}, \ldots, y_{i_k})$ is

$$\Pi_{\#1,\ldots,\#n}(\sigma_c(e_R \times R)).$$

Each atomic $L(\mathcal{M})$ formula $\gamma(\vec{y})$ is translated into $\sigma_{c(\gamma)}(\text{ADOM} \times \ldots \times \text{ADOM})$, where the product is taken n times, and $c(\gamma)$ is the selection condition obtained from γ by replacing each y_i by $\#i$. If ψ_1 and ψ_2 are translated into e_1 and e_2, then $\psi_1 \vee \psi_2$ is translated into $e_1 \cup e_2$, $\psi_1 \wedge \psi_2$ is translated into $e_1 \cap e_2$, and $\neg\psi_1$ is translated into $(\text{ADOM} \times \ldots \times \text{ADOM}) - e_1$. Finally, if we have a subformula ψ that translates into an expression e returning an m-ary relation, $\exists y \psi$ translates into $\Pi_{\#1,\ldots,\#m-1}(e)$, and $\forall y \psi$ translates into $\text{ADOM}^{m-1} - (\Pi_{\#1,\ldots,\#m-1}(\text{ADOM}^m - e))$. It is now routine to verify that when applied to Φ in the prenex form as above, this translation yields the required $\text{ALG}(\mathcal{M})$ expression.

Now it remains to be proved that any $\text{ALG}(\mathcal{M})$ query has TC^0 data complexity. This proof proceeds exactly as the proof of AC^0 data complexity for relational algebra (see [1]) with one exception: every time the σ_c operator is encountered, we have to compute the condition c. Each such c is of form $p_1(\vec{y})\{=, <, \neq, \not<\}p_2(\vec{z})$ where p_1, p_2 are polynomials. Since integer arithmetic (addition, multiplication, comparison) can be done in TC^0 [5, 44], we construct circuits that compute p_1 and p_2 first and then make the comparison. This shows that we can construct a threshold circuit that computes σ_c. The proof that all other operations can be computed by TC^0 circuits (in fact, by AC^0 circuits) is exactly the same as in [1]. The theorem is proved. □

In the following proposition we give complexity bounds for queries against databases over $\langle \mathbb{R}, +, *, 0, 1, < \rangle$ expressible in higher-order logics. Again, we assume that database elements are integers (so that they can be encoded). We restrict our attention to generic queries that do not extend the active domain of their input.

Proposition 10 *Let $\mathcal{M} = \langle \mathbb{R}, +, *, 0, 1, < \rangle$. Let Q be a query expressible in hybrid $\mathcal{FO}+\mathbf{ifp}(\mathcal{M})$ (or $\mathcal{FO}+\mathbf{lfp}(\mathcal{M})$, or $\mathcal{HSO}(\mathcal{M})$, or $\mathcal{FO}+\mathbf{pfp}(\mathcal{M})$). Assume that Q is locally generic and does not extend the active domain of its input. Then its data complexity is PTIME (respectively, PTIME, PH, PSPACE).*

Proof sketch: Suppose Q is given by a formula $\varphi(\vec{x})$ in hybrid $\mathcal{FO}+\mathbf{ifp}(\mathcal{M})$. By Theorem 5, there exists an equivalent formula $\varphi_{\text{act}}(\vec{x})$ in $\mathcal{FO}+\mathbf{ifp}_{\text{act}}(\mathcal{M})$. From generic collapse and local genericity of Q, we get that there is a formula $\psi(\vec{x})$ in $\mathcal{FO}+\mathbf{ifp}_{\text{act}}(<)$ equivalent to φ.

Given a database D and a tuple t, let $\{n_1, \ldots, n_k\}$ be $adom(D)$, where $n_1 < \ldots < n_k$. Let D' and t' be obtained from D and t by changing n_i to i, for each $i = 1, \ldots, k$. By genericitcy, $D \models \varphi(t)$ iff $D' \models \psi(t')$. Since the encodings $enc(D')$ and $enc(t')$ can be obtained from $enc(D)$ and $enc(t)$ in

		$\mathcal{FO}(\mathcal{M})$	$=$	$\mathcal{FO}_{\mathrm{act}}(\mathcal{M})$	$_{\mathrm{dp}}=$	$\textsc{Alg}(\mathcal{M})$
$\mathcal{FO}+\mathbf{lfp}(\mathcal{M})$	$=$	closed $\mathcal{FO}+\mathbf{lfp}(\mathcal{M})$	$=$	$\mathcal{FO}+\mathbf{lfp}_{\mathrm{act}}(\mathcal{M})$	$_{\mathrm{dp}}=$	$\textsc{Datalog}^{\neg}(\mathcal{M})$
$\mathcal{FO}+\mathbf{ifp}(\mathcal{M})$	$=$	closed $\mathcal{FO}+\mathbf{ifp}(\mathcal{M})$	$=$	$\mathcal{FO}+\mathbf{ifp}_{\mathrm{act}}(\mathcal{M})$	$_{\mathrm{dp}}=$	$\textsc{Datalog}^{\neg}(\mathcal{M})$
$\mathcal{HSO}(\mathcal{M})$	$=$	normal form $\mathcal{HSO}(\mathcal{M})$	$=$	$\mathcal{HSO}_{\mathrm{act}}(\mathcal{M})$		
$\mathcal{FO}+\mathbf{pfp}(\mathcal{M})$	$=$	closed $\mathcal{FO}+\mathbf{pfp}(\mathcal{M})$	$=$	$\mathcal{FO}+\mathbf{pfp}_{\mathrm{act}}(\mathcal{M})$	$_{\mathrm{dp}}=$	$\textsc{While}(\mathcal{M})$

Note: $\mathcal{C}\ _{\mathrm{dp}}=\ \mathcal{C}'$ means that the class of domain preserving ($adom(Q(D)) \subseteq adom(D)$) queries in $\mathcal{C}$ is $\mathcal{C}'$.

Figure 1: Summary of the expressiveness results for $\mathcal{M} = \langle \mathbb{R}, +, *, 0, 1, < \rangle$ or any other o-minimal structure that admits quantifier elimination

Logic	Complexity
$\mathcal{FO}(\mathcal{M})$	AC^0
$\mathcal{FO}+\mathbf{lfp}(\mathcal{M})$	PTIME
$\mathcal{FO}+\mathbf{ifp}(\mathcal{M})$	PTIME
$\mathcal{HSO}(\mathcal{M})$	PH
$\mathcal{FO}+\mathbf{pfp}(\mathcal{M})$	PSPACE

Note: in this table, a line $(\mathcal{L}, \mathcal{C})$ means that the data complexity of domain-preserving locally generic queries in $\mathcal{L}(\mathcal{M})$ is in $\mathcal{C}$.

Figure 2: Data complexity for queries over $\mathcal{M} = \langle \mathbb{R}, +, *, 0, 1, < \rangle$

polynomial time, and $D' \models \psi(t')$ can be tested in polynomial time (due to classical results on data complexity, cf. [1, 21]), we conclude that $D \models \varphi(t)$ can be tested in polynomial time. The proof for other logics is similar. $\Box$

We can now summarize the main results of the paper. Figure 1 puts together results about both active and natural semantics. Roughly, each line in Figure 1 shows the equivalence of a logic $\mathbb{L}$, its active-semantics version $\mathbb{L}_{\mathrm{act}}$, and a procedural language (if one exists).

Figure 2 summarizes the results on data complexity for databases over the reals. Note that the first line follows from the results of [29] (since the classes of locally generic domain-preserving queries in $\mathcal{FO}(\langle, \mathbb{R}, +, *, 0, 1, < \rangle)$ and $\mathcal{FO}(\langle, \mathbb{R}, +, -, 0, 1, < \rangle)$ are the same); others follow from Proposition 10.

Figure 3 sumamrizes the collapse results in the paper, for the three main kinds of collase: active-generic, natural-active, and natural-generic. Note that for the natural-generic collapse, some extensions have recently appeared in the literature [4, 9].

As was mentioned before, techniques developed here make it easy to prove similar results for other logics. For example, one can show that for the hybrid version of the transitive closure logic (see [35, 21] for the definition of transitive closure logics), the natural and the active interpretations coincide over the real field, and by the Ramsey property, the class of generic queries definable in it under the active interpretation does not depend on the set of operations Ω, and thus the data complexity of generic queries is NLOGSPACE.

o-minimal, QE

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\mathcal{FO}_{\mathrm{act}}(SC,\mathcal{M}) \arrow[r, equal, "\text{natural-active collapse}"] \arrow[d, equal, "\text{active generic collapse}"'] & \mathcal{FO}(SC,\mathcal{M}) \arrow[d, equal, "\text{natural generic collapse}"] \\
\mathcal{FO}_{\mathrm{act}}(SC,<) \arrow[r, equal] & \mathcal{FO}(SC,<)
\end{tikzcd}$$

all structures

o-minimal structures

Figure 3: Summary of collapse results

6 Conclusion

Our main goal was to delineate the extent to which standard results from the pure relational case 'go through as before' when interpreted structures are present. We have distinguished several aspects of the standard theory that extend routinely, as well as several whose extension requires special techniques and/or special assumptions on the interpreted structure. That is, we have three categories of results.

First, extensions of results from pure relational theory to any interpreted structure. We show that most of the standard language equivalences, expressive bounds and complexity results continue to hold with interpreted structures, assuming the active-domain semantics. Second, extensions of results from the pure case that hold with interpreted structure under special assumptions on the structure. We show that most statements about the natural semantics continue to hold for reasonably-behaved interpreted structures. Third, results that arise in the interpreted case that either do not arise or are not of interest in the pure case. We introduced a class of hybrid logics, that arise naturally in the interpreted case: those with 'mixed' quantification. These logics admit the same normal forms as their unmixed counterparts, if the structures are reasonably behaved.

This is only the tip of the iceberg for each of these classes. The understanding of appropriate algebras for queries with interpreted structures is still incomplete. We are working on algebras and range-restricted calculi for safe queries, other normal forms for nonboolean queries and operations that preserve safety. Although we have shown that for nicely-behaved structures we get very satisfying extensions of results about the natural semantics, we do not *characterize* the class of structures for which this holds. For some recent progress in that direction, see [4, 9]. It is also open whether the normal forms results for hybrid logics hold in general.

The algorithm for converting natural quantification to active is of interest in its own right. It extends work done in the constraint community (e.g., in [45] for linear constraints), and can also be seen as extending quantifier elimination algorithms, that originate with Tarski [48], to handle large parameter sets. We would like to have a unified picture of the relation between the algorithm presented here and those in, for example, [14, 45], as well as with uniform quantifier-elimination for the real field [6, 7], where a detailed complexity analysis was given. We plan to study the integration of such algorithms into optimization systems for constraint queries.

We are interested in understanding the connection between our results and metafinite model theory [26]. Some of the motivations for [26] are very close to those for our work, but it does not appear that we can use any of the results in [26] to derive any of our results.

Acknowledgements We thank Rick Hull, Gabriel Kuper, Martin Otto, Dan Suciu, Jan Van den Bussche, Victor Vianu, Scott Weinstein, and an anonymous referee for their comments on earlier drafts of this paper.

References

[1] S. Abiteboul, R. Hull and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[2] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *JCSS* 43 (1991), 62–124.

[3] A.K. Ailamazyan, M.M. Gilula, A.P. Stolboushkin and G.F. Shvarts. Reduction of a relational model with infinite domains to the finite-domain case. *Soviet Physics – Doklady*, 31 (1986), 11–13.

[4] J. Baldwin and M. Benedikt. Embedded finite models, stability theory and the impact of order. In *LICS'98*, pages 490–500.

[5] D.A. Barrington, N. Immerman, H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306,1990.

[6] S. Basu. An improved algorithm for quantifier elimination over real closed fields. In *FOCS'97*, pages 56–65. Full version to appear in *J. ACM.*

[7] S. Basu. Uniform quantifier elimination and constraint query processing. *ISSAC'97*, pages 21–27.

[8] C. Beeri and T. Milo. Comparison of functional and predicative query paradigms. *Journal of Computer and System Sciences*, 54 (1997), 3–33.

[9] O. Belegradek, A. Stolboushkin and M. Taitslin. Extended order-generic queries. *Annals of Pure and Applied Logic*, to appear.

[10] M. Benedikt, G. Dong, L. Libkin and L. Wong. Relational expressive power of constraint query languages. *J. ACM* 45 (1998), 1–34.

[11] M. Benedikt and L. Libkin. On the structure of queries in constraint query languages. *LICS'96*, pages 25–34.

[12] M. Benedikt and L. Libkin. Languages for relational databases over interpreted structures. In *PODS'97*, pages 87–98.

[13] M. Benedikt and L. Libkin. Safe constraint queries. In *PODS'98*, pages 99–108.

[14] M. Ben-Or, D. Kozen, J. Reif. The complexity of elementary algebra and geometry. *Journal of Computer and System Sciences* 32:251–264, 1986.

[15] R.B. Boppana and M. Sipser. The Complexity of Finite Functions. In *Handbook of Theoretical Computer Science*, Vol. A, chapter 14, (J. van Leeuwen editor), North-Holland, 1990.

[16] L. Cabibbo and J. Van den Bussche. Converting untyped formulas to typed ones. *Acta Informatica*, 35(8) (1998), 637–643.

[17] A. Chandra and D. Harel. Computable queries for relational databases. *Journal of Computer and System Sciences* 21(2):156–178, 1980.

[18] A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences* 25 (1982), 99–128.

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

[20] A. Dawar, S. Lindell, S. Weinstein. First order logic, fixed point logic, and linear order. In *Computer Science Logic '95*, LNCS vol. 1092, 1996, pages 161–177.

[21] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer Verlag, 1995.

[22] M. Escobar-Molano, R. Hull and D. Jacobs. Safety and translation of calculus queries with scalar functions. In *PODS'93*, pages 253–264.

[23] K. Etessami. Counting quantifiers, successor relations, and logarithmic space. *Journal of Computer and System Sciences*, 54 (1997), 400–411.

[24] R. Fagin. Probabilities on finite models. *Journal of Symbolic Logic* 41 (1976), 50–58.

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

[26] E. Grädel and Y. Gurevich. Metafinite model theory. *Information and Computation*, 140 (1998), 26–81.

[27] R.L. Graham, B.L. Rothschild and J.H. Spencer. *Ramsey Theory.* John Wiley & Sons, 1990.

[28] S. Grumbach and J. Su. Queries with arithmetical constraints. *Theoretical Computer Science* 173 (1997), 151–181. Extended abstract in *PCCP'95.*

[29] S. Grumbach, J. Su, and C. Tollu. Linear constraint databases. In *Proceedings of Logic and Comput. Complexity, 1994*, pages 426–446.

[30] S. Grumbach and J. Su. Finitely representable databases, *Journal of Computer and System Sciences* 55 (1997), 273–298.

[31] Y. Gurevich and S. Shelah. Fixed-point extensions of first-order logic. *Annals of Pure and Applied Logic* 32 (1986), 265–280.

[32] R. Hull, J. Su. On the expressive power of databases with intermediate types. *Journal of Computer and System Sciences* 43 (1991), 219–267.

[33] R. Hull and J. Su. Domain independence and the relational calculus. *Acta Informatica* 31:513–524, 1994.

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

[35] N. Immerman. Descriptive complexity: A logician's approach to computation. *Notices of the AMS* 42 (1995), 1127–1133.

[36] P. Kanellakis, G. Kuper, and P. Revesz. Constraint query languages. *Journal of Computer and System Sciences* 51 (1995), 26–52. Extended abstract in *PODS'90*.

[37] A. Klug. On conjunctive queries containing inequalities. *Journal of the ACM* 35(1):146–160, 1988.

[38] Ph. Kolaitis and M. Vardi. 0-1 laws and decision problems for fragments of second-order logic. *Information and Computation*, 87 (1990), 302–338.

[39] Ph. Kolaitis, M. Vardi. Infinitary logic and 0-1 laws. *Information and Computation*, 98 (1992), 258–294.

[40] L. Libkin and L. Wong. Query languages for bags and aggregate functions. *Journal of Computer and System Sciences*, 55(1997), 241–272.

[41] L. Libkin and L. Wong. On the power of aggregation in relational query languages. *DBPL'97*, Springer LNCS 1369, pages 260-280.

[42] Y. Moschovakis. *Elementary Induction on Abstract Structures.* North Holland, 1974.

[43] M. Otto and J. Van den Bussche. First-order queries on databases embedded in an infinite structure. *Information Processing Letters*, 60(1996), 37–41.

[44] I. Parberry and G. Schnitger. Parallel computation and threshold functions. *Journal of Computer and System Sciences* 36 (1988), 278–302.

[45] J. Paredaens, J. Van den Bussche, and D. Van Gucht. First-order queries on finite structures over the reals. *SIAM J. Comput.* 27(6) (1998), 1747–1763.

[46] A. Pillay, C. Steinhorn. Definable sets in ordered structures. III. *Trans. of the AMS* 309 (1988), 469–476.

[47] A.P. Stolboushkin and M.A. Taitslin. Linear vs. order constraint queries over rational databases. In *PODS'96*, pages 17–27.

[48] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* 2nd ed., Univ. California Press, 1951.

[49] J. D. Ullman. *Principles of Database and Knowledgebase Systems*, Volume I, Computer Science Press, 1989.

[50] L. van den Dries. *Tame Topology and O-minimal Structures.* Cambridge, 1998.

[51] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *Journal of Computer and System Sciences* 54(1):113–135, 1997.

[52] M.Y. Vardi. The complexity of relational query languages. In *STOC'82*, pages 137–146.

[53] M.Y. Vardi. On the complexity of bounded-variable queries. In *PODS'95*, pages 266–276.

[54] A.J. Wilkie. Model completeness results for expansions of the ordered field of real numbers by restricted Pfaffian functions and the exponential function. *J. Amer. Math. Soc.* 9 (1996), 1051–1094.