Descriptive Complexity Theory for Constraint Databases

Erich Grädel and Stephan Kreutzer

Lehrgebiet Mathematische Grundlagen der Informatik,
RWTH Aachen, D-52056 Aachen,
{graedel, kreutzer}@informatik.rwth-aachen.de

Abstract. We consider the data complexity of various logics on two important classes of constraint databases: dense order and linear constraint databases. For dense order databases, we present a general result allowing us to lift results on logics capturing complexity classes from the class of finite ordered databases to dense order constraint databases. Considering linear constraints, we show that there is a significant gap between the data complexity of first-order queries on linear constraint databases over the real and the natural numbers. This is done by proving that for arbitrary high levels of the Presburger arithmetic there are complete first-order queries on databases over $(\mathbb{N}, <, +)$. The proof of the theorem demonstrates a simple argument for translating complexity results for prefix classes in logical theories to results on the complexity of query evaluation in constraint databases.

1 Introduction

Descriptive complexity theory studies the relationship between logical definability and computational complexity. In particular one looks for results saying that, on a certain class $\mathcal{K}$ of structures, a logic L (like first-order logic or least fixed point logic) *captures* a complexity class $\mathcal{C}$. This means that (1) for every fixed sentence $\psi \in L$, the complexity of evaluating ψ on structures from $\mathcal{K}$ is a problem in the complexity class $\mathcal{C}$, and (2) every property of structures in $\mathcal{K}$ that can be decided with complexity $\mathcal{C}$ is definable in the logic L. Two important examples of such results are Fagin's Theorem, saying that existential second-order logic captures NP on the class of all finite structures, and the Immerman-Vardi Theorem, saying that least fixed point logic captures PTIME on the class of all ordered finite structures. Indeed, on *ordered* finite structures, logical characterizations of this kind are known for all major complexity classes. On the other hand it is not known, and one of the major open problems in the area, whether PTIME can be captured by any logic, if no ordering is present. We refer to [1, 10] for background on descriptive complexity.

Up to now, descriptive complexity has been considered almost exclusively on finite structures. But the research program of descriptive complexity makes sense also for classes of infinite structures, provided that they admit a finite presentation. There have been a few studies of descriptive complexity theory on infinite structures concerning, for instance, metafinite structures and complexity theory over the reals [3, 4], recursive structures [8] and, as we do in the present paper, constraint databases (see e.g. [12, 6, 5] and the references there).

Constraint databases are a modern database model admitting infinite relations that are finitely presented by quantifier-free formulae (constraints) over some fixed background structure. For example, to store geometrical data, it is useful to have not just a finite set as the domain of the database, but to include all real numbers 'in the background'. Also the presence of interpreted functions, like addition and multiplication, is desirable. The constraint database framework introduced by Kanellakis, Kuper and Revesz [12] meets both requirements. Formally, a constraint database consists of a *context structure* $\mathfrak{A}$, like $(\mathbb{R}, <, +, \cdot)$, and a set $\{\varphi_1, \dots, \varphi_m\}$ of quantifier-free formulae defining the database relations. We give the precise definition in the next section.

When studying the data complexity of constraint query languages, it soon became clear that allowing recursion in query languages leads to non-closed or undecidable query languages even for rather simple context structures. On the other hand there are promising results for non-recursive languages in many interesting contexts. For the context structure $(\mathbb{R}, <)$ a LOGSPACE data complexity for first-order logic has been established by Kanellakis, Kuper, and Revesz which was later improved to AC^0 by Kanellakis and Goldin [11]. In [12] it has also been shown that first-order logic still has data complexity NC if the context structure is extended by addition and multiplication. Thus first-order logic is well-suited as a query language for spatial databases where the context structure is the field of reals.

In this paper we will consider the complexity of query evaluation in two important cases: (1) linear constraint databases, where the context structure is $(\mathbb{R}, <, +)$ or $(\mathbb{N}, <, +)$, and (2) constraint databases over dense linear orders.

It turns out that the data complexity of first-order query on linear constraint databases depends heavily on the universe. The data complexity of first-order queries on databases over $(\mathbb{R}, <, +)$ is known to be in NC. It has been conjectured by Grumbach and Su [6] that this is also true for the context structure $(\mathbb{N}, <, +)$. We refute this conjecture here by showing that we find complete first-order queries for each level of the polynomial time hierarchy.

As stated above, allowing recursion in query languages tends to result in undecidable languages. For instance, we will observe that this is the case for linear constraint queries over $(\mathbb{R}, <, +)$. An exception are *dense order constraint databases*, where the context structure is $(\mathbb{R}, <)$ (or any other dense linear order without endpoints). There we can incorporate recursion and still end up in decidable and closed languages. For instance, it has been shown in [12, 5] that inflationary DATALOG with negation has PTIME data complexity and, in fact, that it captures PTIME on dense order constraint databases. We continue this line of research and present a general technique that allows to lift capturing results from the class of ordered finite structures to constraint databases over $(\mathbb{R}, <)$. This is done by associating with every constraint database over $(\mathbb{R}, <)$ a finite ordered structure, called the *invariant* of the database which carries all the information stored in the infinite database. The finite database can be defined by first-order formulae and therefore with very low data complexity. A query on the constraint database can be evaluated in the invariant in such a way, that the result of the original query can be regained from the answer on the finite database with very low data complexity. Indeed the invariant is first-order interpretable in the original database, and this allows to translate any formula that represents a query on the invariant into an equivalent formula over the original database. In this way capturing results are lifted from ordered finite structures to dense order constraint databases.

2 Constraint Databases

The basic idea in the definition of constraint databases is to allow infinite relations which have a finite representation by a quantifier-free formula. Let $\mathfrak{A}$ be a τ-structure, called the *context structure*, and $\varphi(x_1, \dots, x_n)$ be a quantifier-free formula of vocabulary τ that may contain elements from A as parameters. Let $\sigma := \{R_1, \dots, R_k\}$ be a relational signature disjoint from τ.

We say that an n-ary relation $R \subseteq A^n$ is *represented by* $\varphi(x_1, \dots, x_n)$ *over* $\mathfrak{A}$, if $R = \{(a_1, \dots, a_n) : \mathfrak{A} \models \varphi(a_1, \dots, a_n)\}$. A σ*-constraint database* over the context structure $\mathfrak{A}$ is an expansion $\mathfrak{B} = (\mathfrak{A}, R_1, \dots, R_k)$ of $\mathfrak{A}$ where all σ-relations R_i are finitely represented by formulae φ_{R_i} over $\mathfrak{A}$. The set $\Phi := \{\varphi_{R_1}, \dots, \varphi_{R_k}\}$ is called a *finite representation of* $\mathfrak{B}$. The set of finitely representable relations over $\mathfrak{A}$ is denoted by $\mathrm{Rel}_{fr}(\mathfrak{A})$ and the set of all constraint databases over $\mathfrak{A}$ is denoted by $\mathrm{Exp}_{fr}(\mathfrak{A})$. The signature τ is called the *context signature* whereas σ is called the *database signature*.

By definition, constraint databases are expansions of a context structure by finitely representable database relations. Note that the same relation can be represented in different ways, e.g. $\varphi_1 := x < 10 \wedge x > 0$ and $\varphi_2 := (0 < x \wedge x < 6) \vee (6 < x \wedge x < 10) \vee x = 6$ are different formulae but define the same relation. Two representations Φ and Φ' are $\mathfrak{A}$*-equivalent*, if they represent the same database over $\mathfrak{A}$.

To measure the complexity of algorithms taking constraint databases as inputs we have to define the size of a constraint database. Unlike finite databases, the size of constraint databases cannot be given in terms of the number of elements stored in them but has to be based on a representation of the database. Note that equivalent representations of a database need not to be of the same size. Thus the size of a constraint database cannot be defined independent of a particular representation. In the following, whenever we speak of a constraint database $\mathfrak{B}$, we have a particular representation Φ of $\mathfrak{B}$ in mind. The size $|\mathfrak{B}|$ of $\mathfrak{B}$ then is defined as the sum of the length of the formulae in Φ. This corresponds to the standard encoding of constraint databases by the formulae of their representation.

Constraint queries. Let $\mathfrak{A}$ be a τ-structure and σ a relational signature. A *constraint query* $Q : Exp_{fr}(\mathfrak{A}) \to Rel_{fr}(\mathfrak{A})$ is a mapping from σ-constraint databases over $\mathfrak{A}$ to finitely representable relations over $\mathfrak{A}$. In the sequel we are interested only in queries defined by formulae of a given logic $\mathfrak{L}$. In order to define queries by $\mathfrak{L}$-formulae, we require the context structures to admit quantifier elimination for $\mathfrak{L}$. This means that every $\mathfrak{L}$-formula φ is equivalent in $\mathfrak{A}$ to a quantifier-free formula. If $\mathfrak{A}$ admits quantifier elimination for $\mathfrak{L}$, then every formula $\psi(x_1, \dots, x_k) \in \mathfrak{L}[\tau \cup \sigma]$ defines a query Q_φ taking a σ-constraint database $\mathfrak{B}$ over $\mathfrak{A}$ to the set $\{\bar{a} \in A^k \ : \ \mathfrak{B} \models \psi(\bar{a})\}$, and the result of the query is itself finitely representable.

Typical questions that arise when dealing with constraint query languages are the complexity of query evaluation for a certain constraint query language and the definability of a query in a given language. For a fixed query formula $\varphi \in \mathcal{L}$, the *data complexity* of the query Q_φ is defined as the amount of resources (e.g. time, space, or number of processors) needed to evaluate the function that takes a representation Φ of a database $\mathfrak{B}$ to a representation of the answer relation $Q_\varphi(\mathfrak{B})$.

3 Linear Constraints

In this section we consider linear constraint databases, that is, databases defined over the context structures $(\mathbb{R}, <, +)$, $(\mathbb{Q}, <, +)$ or $(\mathbb{N}, <, +)$. The data complexity of linear constraint queries in the context of $(\mathbb{R}, <, +)$ and $(\mathbb{Q}, <, +)$ has been studied by Grumbach, Su, and Tollu in [5, 7]. In [5] it is claimed that "first-order queries on linear constraint databases have a NC_1 data complexity."

First, we briefly discuss the possibility whether more powerful query languages than first-order logic can be effectively evaluated on linear constraint databases. However, a simple argument shows that adding a recursion mechanism to first-order logic leads to non-closed or undecidable languages. For example, the (FO+DTC)-formula $nat(x) := [\mathrm{DTC}_{x,y}(x + 1 = y)](0, x)$ defines the natural numbers, and multiplication of natural numbers can be defined by the (FO+DTC)-formula $mult(x, y, z) := [\mathrm{DTC}_{uv,u'v'}(u+1 = u' \wedge v+x = v')](00, yz)$.

It follows that Hilbert's 10th problem (or the existential theory of arithmetic) can be reduced to the evaluation of existential FO+DTC-queries on linear constraint databases.

Theorem 1. *Every query language over the context structure* $(\mathbb{R}, <, +)$ *which is at least as expressive as existential* FO+DTC *is undecidable.*

Thus the result by Grumbach and Su cannot be extended to query languages allowing recursion. We now show that the result does also not generalize to linear constraint queries over the natural numbers.

Presburger arithmetic (PrA), the theory of the structure $(\mathbb{N}, <, +)$, is well known to be decidable. Strictly speaking, we have to expand $(\mathbb{N}, <, +)$ by divisibility relations $a \mid x$ (for all parameters $a \in \mathbb{N}$), because otherwise the theory would not admit quantifier elimination and hence non-Boolean queries could not be evaluated in closed form. Note that $a \mid x$ is of course definable in $(\mathbb{N}, <, +)$ but not by a quantifier-free formula. However, we will show that even the evaluation of boolean first-order queries is much more complex in the context of the Presburger arithmetic than on $(\mathbb{R}, <, +)$. This result relies on complexity results for fragments of PrA with bounded quantifier prefixes. Let $\mathcal{Q} := Q_1 \cdots Q_k$ be a word in $\{\exists, \forall\}^*$. Then $[\mathcal{Q}] \cap \mathrm{PrA}$ is the set of sentences of the form $\psi := Q_1 x_1 \cdots Q_k x_k \varphi$ such that $(\mathbb{N}, <, +) \models \psi$ and φ is quantifier-free. It has been shown [2, 15] that the complexity of such fragments of Presburger arithmetic may reside on arbitrary high levels of the polynomial-time hierarchy. Essentially the evaluation of formulae with $m+1$ quantifier blocks of bounded length is in the m-th level of the hierarchy.

Theorem 2 (Grädel, Schöning). *Let $m \geq 1, r_1, \ldots, r_m \geq 1$ and $r_{m+1} \geq 3$. Then, for odd m, $[\exists^{r_1} \forall^{r_2} \cdots \exists^{r_m} \forall^{r_{m+1}}] \cap \mathrm{PrA}$ is Σ^p_m-complete, and $[\forall^{r_1} \exists^{r_2} \cdots \forall^{r_m} \exists^{r_{m+1}}] \cap \mathrm{PrA}$ is Π^p_m-complete. For even m, $[\exists^{r_1} \forall^{r_2} \cdots \forall^{r_m} \exists^{r_{m+1}}] \cap \mathrm{PrA}$ is Σ^p_m-complete and $[\forall^{r_1} \exists^{r_2} \cdots \exists^{r_m} \forall^{r_{m+1}}] \cap \mathrm{PrA}$ is Π^p_m-complete.*

The proof of the following theorem exhibits a simple argument for translating such complexity results for prefix classes in logical theories to results on the complexity of query evaluation in constraint databases.

Theorem 3. *Let ψ be a first-order boolean query on constraint databases over $(\mathbb{N}, <, +)$. Then the data complexity of ψ is in the polynomial-time hierarchy. Conversely, for each class Σ^p_k, resp. Π^p_k of the polynomial time hierarchy there is a fixed query ψ whose data complexity is Σ^p_k-complete, resp. Π^p_k-complete.*

Proof. We can assume that $\psi = Q_1 x_1 \cdots Q_k x_k \varphi$ with φ quantifier-free and with database relations $R_1, \ldots, R_m$. Given a database $\mathfrak{B} = (\mathbb{N}, <, +, R_1, \ldots, R_m)$ where the database relations are represented by $\beta_1, \ldots, \beta_m$ over $(\mathbb{N}, <, +)$, let $\psi' := \mathit{unfold}(\psi, \mathfrak{B})$ be the unfolded query, obtained by replacing in ψ all occurrences of $R_1, \ldots, R_m$ by the defining formulae $\beta_1, \ldots, \beta_m$. Since the β_i are quantifier-free ψ' has the same prefix as ψ and length bounded by $O(|\mathfrak{B}|)$ (given that ψ is considered fixed). Obviously $\mathfrak{B} \models \psi$ if and only if $\psi' \in [Q_1 \ldots Q_k] \cap \mathrm{PrA}$. Hence the data complexity of ψ is in the polynomial-time hierarchy (and actually, we can read off the level of the hierarchy directly from the prefix of ψ).

For the second assertion of the theorem, consider any quantifier prefix $\mathcal{Q} = Q_1 \ldots Q_m$. Let R be an m-ary relation symbol and let $\psi_{\mathcal{Q}}$ be the query $Q_1 x_1 \cdots Q_m x_m R x_1 \ldots x_m$. The decision problem for $[\mathcal{Q}] \cap \mathrm{PrA}$ reduces to the evaluation problem of $\psi_{\mathcal{Q}}$ on constraint databases over $(\mathbb{N}, +, <)$. Indeed, for every sentence $\varphi = Q_1 x_1 \cdots Q_m x_m \varphi'(x_1, \ldots, x_m)$ in $\mathrm{FO}(<, +)$, let $\mathfrak{B}_\varphi$ be the $\{R\}$-database over $(\mathbb{N}, <, +)$ such that $R^{\mathfrak{B}_\varphi}$ is represented by φ'. The size of $\mathfrak{B}_\varphi$ is bounded by the length of φ. Clearly, φ is true in $(\mathbb{N}, <, +)$ if and only if $\mathfrak{B}_\varphi \models \psi_{\mathcal{Q}}$.

Hence, by choosing $\mathcal{Q}$ as indicated by Theorem 2, the evaluation problem for $\psi_{\mathcal{Q}}$ is Σ^p_k-complete, resp. Π_k-complete. □

We have seen that first-order logic can express quite complex queries. We now consider sub-classes of first-order logic which can still be efficiently evaluated. It has been shown by Lenstra and Scarpellini (see references in [2]) that for all fixed dimensions $t \in \mathbb{N}$, $[\exists^t] \cap \mathrm{PrA}$ and $[\forall^t] \cap \mathrm{PrA}$ are in PTIME. Thus, by an argument similar to the one above, we can show the following theorem.

Theorem 4. *Existential and universal boolean queries on constraint databases over* $(\mathbb{N}, <, +)$ *have* PTIME *data complexity.*

However, as soon as we admit queries with alternation depth two, the evaluation problem is NP- or Co-NP-hard. This follows from a result by Schöning [15] who proved that $[\exists\forall] \cap PrA$ is NP-complete, strengthening a result in [2].

4 Dense Linear Orders

We now consider the complexity of query evaluation in the context of dense linear orders. We prove a general result which allows us to give precise complexity bounds for the data complexity of various logics such as transitive closure or fixed-point logic and to extend results on logics capturing complexity classes from the realm of finite ordered structures to constraint databases over dense linear orders. Given a fixed query, its evaluation in a database can be done by (1) transforming the database into a finite structure, called its invariant, (2) evaluating a slightly modified version of the query on the invariant, and (3) transforming the result of the evaluation to an answer of the original query.

We fix the context structure $\mathfrak{A} := (\mathbb{R}, <)$ and a query ψ of vocabulary $\{<\} \cup \sigma$ with database signature $\sigma = \{R_1, \ldots, R_k\}$. Let $P^\psi \subseteq \mathbb{R}$ be the (finite) set parameters that occur in ψ. The query has to be transformed so that it can be evaluated in the invariant. This transformation is independent of a particular database and can be seen as a compilation or preprocessing step. To set up the evaluation method outlined above, we define two mappings. The first, *inv*, maps databases to their corresponding invariants; the second, π, maps the answer of the query on the invariant to the answer of the original query.

4.1 The invariant of a constraint database

Definition 5. Let $\sigma := \{R_1, \ldots, R_k\}$ be a signature, $\mathfrak{B}$ be a σ-database over $(\mathbb{R}, <)$, $P \subset \mathbb{R}$ a set of elements, and $\bar{b}$ a tuple of real numbers.

- The *complete atomic type of* $\bar{b}$ *over* P *with respect to* $\mathfrak{B}$, written as $atp_P^{\mathfrak{B}}(\bar{b})$, is the set of all atomic and negated atomic formulae $\varphi(\bar{x})$ over the signature $\{<, R_1, \ldots, R_k\}$ using parameters from P such that $\mathfrak{B} \models \varphi(\bar{b})$. We omit the index P if P is empty and denote by $otp^{\mathfrak{B}}(\bar{b})$, resp. $otp_P^{\mathfrak{B}}(\bar{b})$, the complete atomic type of $\bar{b}$ (over P) with respect to $\mathfrak{B}$ over the signature $\{<\}$.
- A maximally consistent set of atomic and negated atomic $\sigma \cup \{<\}$-formulae $\varphi(\bar{x})$ is a *complete atomic type (over* P*) in the variables* $\bar{x}$, if it is a complete atomic type (over P) of a tuple $\bar{b}$ with respect to a σ-expansion of $\mathfrak{A}$. We write $atp^{\mathfrak{B}}(\bar{x})$, resp. $atp_P^{\mathfrak{B}}(\bar{x})$, for a complete atomic type (over P) in the variables $\bar{x}$ over the database signature σ of $\mathfrak{B}$.

A type is an n-type if it has n free variables. We omit $\mathfrak{B}$ if it is clear from the context. When speaking about types we always mean complete atomic types throughout this chapter.

We call complete atomic types over $\sigma \cup \{<\}$ also *complete database types.* Database types are of special interest here because the database type of a tuple $\bar{b}$ determines everything we can say about $\bar{b}$ in terms of the database, especially in which database relations $\bar{b}$ stands.

Suppose $\mathfrak{B}$ is a database and $P^{\mathfrak{B}}$ the set of parameters used in its definition. Recall from the introduction that there are different ways to represent the database $\mathfrak{B}$. The set of parameters used in these representations will generally differ from $P^{\mathfrak{B}}$. We define a set of parameters, called the canonical parameters, which can be extracted from $\mathfrak{B}$ independent of its representation.

Definition 6. Suppose $\mathfrak{B} = (\mathbb{R}, <, R_1^{\mathfrak{B}}, \dots, R_k^{\mathfrak{B}})$ is a database. The set $cp(\mathfrak{B}) \subset \mathbb{R}$ of *canonical parameters of* $\mathfrak{B}$ is the set of elements p satisfying the following condition.
For at least one n-ary relation $R \in \{R_1^{\mathfrak{B}}, \dots, R_k^{\mathfrak{B}}\}$ there are $a_1, \dots, a_n \in \mathbb{R}$, an $\epsilon \in \mathbb{R}, \epsilon > 0$, and an ϵ-neighbourhood $\delta = (p-\epsilon, p+\epsilon)$ of p such that one of the following holds.

- For all $q \in \delta, q < p$ and for no $q \in \delta, q > p$ we have $R\bar{a}[p/q]$. Here $R\bar{a}[p/q]$ means that all components $a_i = p$ are replaced by q.
- For all $q \in \delta, q > p$ and for no $q \in \delta, q < p$ we have $R\bar{a}[p/q]$.
- $R\bar{a}[p/q]$ holds for all $q \in \delta \backslash \{p\}$ but not for $q = p$.
- $R\bar{a}[p/q]$ holds for $q = p$ but not for any $q \in \delta \backslash \{p\}$.

Lemma 7. *All canonical parameters of $\mathfrak{B}$ occur explicitly in all representations of $\mathfrak{B}$.*

In particular this implies that the cardinality of $cp(\mathfrak{B})$ is bounded by the size of any representation of $\mathfrak{B}$.

We show in the next lemma that an atomic order type over $cp(\mathfrak{B})$ uniquely determines a complete database type. It follows that every two tuples realizing the same atomic order type over $cp(\mathfrak{B})$ occur in the same database relations. Thus the parameter set $cp(\mathfrak{B})$ is sufficient to define a representation of $\mathfrak{B}$.

Lemma 8. *Suppose $\mathfrak{B}$ is a database and $\bar{a}, \bar{b} \in \mathbb{R}^k$ are two k-tuples.*

(i) If $otp^{\mathfrak{B}}_{cp(\mathfrak{B})}(\bar{a}) = otp^{\mathfrak{B}}_{cp(\mathfrak{B})}(\bar{b})$, then $atp^{\mathfrak{B}}(\bar{a}) = atp^{\mathfrak{B}}(\bar{b})$.
(ii) If $otp^{\mathfrak{B}}_{cp(\mathfrak{B})}(a_i) = otp^{\mathfrak{B}}_{cp(\mathfrak{B})}(b_i)$ for all $1 \leq i \leq k$ and $otp^{\mathfrak{B}}(\bar{a}) = otp^{\mathfrak{B}}(\bar{b})$, then $otp^{\mathfrak{B}}_{cp(\mathfrak{B})}(\bar{b}) = otp^{\mathfrak{B}}_{cp(\mathfrak{B})}(\bar{a})$.
(iii) If $P \supseteq cp(\mathfrak{B})$ is a superset of $cp(\mathfrak{B})$, then $otp^{\mathfrak{B}}_P(\bar{b}) = otp^{\mathfrak{B}}_P(\bar{a})$ implies $otp^{\mathfrak{B}}_{cp(\mathfrak{B})}(\bar{a}) = otp^{\mathfrak{B}}_{cp(\mathfrak{B})}(\bar{b})$.

Proof. The proof of the second and third part are straightforward. To prove the first part suppose for the sake of contradiction that $atp^{\mathfrak{B}}(\bar{b})$ and $atp^{\mathfrak{B}}(\bar{a})$ differ. Then there is an atomic or negated atomic formula φ such that $\mathfrak{B} \models \varphi(\bar{a})$ but $\mathfrak{B} \not\models \varphi(\bar{b})$. If φ is of the form $x_i < x_j$, then $a_i < a_j$ but not $b_i < b_j$, which contradicts the assumption that $otp^{\mathfrak{B}}_{cp(\mathfrak{B})}(\bar{b}) = otp^{\mathfrak{B}}_{cp(\mathfrak{B})}(\bar{a})$.
Now suppose φ is of the form $Rx_1 \cdots x_r$, where $r := ar(R)$. Let $\mathcal{C} := (\bar{c}_0, \bar{c}_1, \dots, \bar{c}_k)$

be a sequence of points in $\mathbb{R}^k$, such that for $c_{ij} := b_j$ for all $j \le i$ and $c_{ij} := a_j$ for all $j > i$. Thus $\bar{c}_0 = \bar{a}$, $\bar{c}_k = \bar{b}$, $\bar{c}_1 = (b_1, a_2, \dots, a_k)$, $\bar{c}_2 = (b_1, b_2, a_3, \dots, a_k)$, and so on. Further, let $L := (l_1, \dots, l_k)$ be a sequence of lines such that the endpoints of l_i are c_{i-1} and c_i. As $\mathfrak{B} \models \varphi(\bar{a})$ but $\mathfrak{B} \not\models \varphi(\bar{b})$, there is an l_j that intersects both $R^{\mathfrak{B}}$ and $\mathbb{R}^k \backslash R^{\mathfrak{B}}$. Assume w.l.o.g. that $a_j < b_j$. Let $\bar{q} := \bar{c}_{j-1}$. Then there is a $p \in \mathbb{R}$ with $a_j < p \le b_j$ such that $R^{\mathfrak{B}}\bar{q}$ but not $R^{\mathfrak{B}} q_1, \dots, q_{j-1}, p, q_{j+1}, \dots, q_k$. We claim that there is at least one canonical parameter d with $a_j \le d \le p$. To prove this claim, let $A := \{a \in \mathbb{R} \ : \ a_j \le a \text{ and } R^{\mathfrak{B}} q_1, \dots, q_{j-1} a' q_{j+1}, \dots, q_k \text{ for all } a_j \le a' \le a\}$. Let d be the supremum of A. Then, by Definition 6, c is a canonical parameter and $a_j \le d \le p$. This proves the claim. Thus $\bar{a}$ and $\bar{b}$ do not satisfy the same complete order type over $cp(\mathfrak{B})$ which contradicts the assumption. □

One implication of the lemma is the following. Suppose we want to decide if $R\bar{a}$ holds for a tuple $\bar{a} := a_1, \dots, a_k$ and a k-ary database relation R. The question can be answered if we know whether $R\bar{b}$ holds for a tuple $\bar{b} := b_1, \dots, b_k$ such that $\bar{a}$ and $\bar{b}$ realize the same order type and each b_i realizes the same 1-order type over $cp(\mathfrak{B})$ as a_i. This will be the central idea in the definition of the invariant.

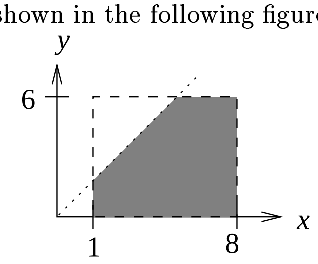
The relevant set of parameters that we need for the evaluation of ψ on a database $\mathfrak{B}$ is $P^{\mathfrak{B},\psi} := \{0, 1\} \cup cp(\mathfrak{B}) \cup P^{\psi}$. The constants 0 and 1 are included because they will be needed in the definition of the invariant.

Since P^{ψ} is finite and Definition 6 of the set of canonical parameters can obviously be formalized in first-order logic, it follows that for any fixed ψ, the set $P^{\mathfrak{B},\psi}$ is uniformly first-order definable over $(\mathfrak{R}, <, 0, 1, P^{\psi})$.

Lemma 9. *There exists a first-order formula $\delta(x)$ of vocabulary $\{<, 0, 1, P^{\psi}\} \cup \sigma$ such that for every σ-database $\mathfrak{B} = (\mathbb{R}, <, R_1^{\mathfrak{B}}, \dots, R_k^{\mathfrak{B}})$, $P^{\mathfrak{B},\psi} = \{a \in \mathbb{R} : \mathfrak{B} \models \delta(a)\}$.*

We are now ready to define the invariant. Given a database $\mathfrak{B}$, define an equivalence relation $\sim$ on $\mathbb{R}$ such that two elements a and b are $\sim$-equivalent if and only if they realize the same 1-order type over $P^{\mathfrak{B},\psi}$. As $P^{\mathfrak{B},\psi}$ is first-order definable the equivalence relation $\sim$ is first-order definable as well. The set of equivalence classes $\mathbb{R}_{\sim}$ serves as the universe of the invariant. To complete the definition we have to specify the database relations.

Before we give the detailed definition of the relations in the invariant, we illustrate the idea by an example. Consider a database $\mathfrak{B}$ with a single binary relation S represented by $\varphi_S(x, y) := x > 1 \wedge x < 8 \wedge y > 0 \wedge y < 6 \wedge y < x$. The relation is shown in the following figure.



As explained above, the invariant depends not only on the database $\mathfrak{B}$ but also on the parameters used in the query ψ. To simplify the example let ψ be an arbitrary query with no extra parameters. Thus the set $P^{\mathfrak{B},\psi}$ consists of the four elements $0, 1, 6, 8$ and there are nine different $\sim$-equivalence classes, namely the intervals $(-\infty, 0), \{0\}, (0, 1), \{1\}, (1, 6), \{6\}, (6, 8), \{8\}$, and $(8, \infty)$. Recall that these equivalence classes form the universe of the invariant. Thus the relation S has somehow to be defined in terms of these classes. Obviously it is not enough to factorize S by $\sim$, because as $5 \sim 5.1$, the equivalence classes $[5]$ and $[5.1]$ are equal, but $([5.1], [5]) \in S$ and $([5], [5]) \notin S$. Thus $S_{/\sim}$ would not be well-defined.

Instead of simply factorizing a m-ary relation R by $\sim$ we consider the set C_R of $(m+1)$-tuples $([a_1], \dots, [a_k], \rho)$, where $[a_i] \in \mathbb{R}_{/\sim}, 1 \leq i \leq m$ and ρ denotes an m-order type, such that $([a_1], \dots, [a_m], \rho) \in C_R$ if and only if there is a $\bar{b} \in \mathbb{R}^m$ realizing ρ such that $R\bar{b}$ holds and $a_i \sim b_i$ for all $1 \leq i \leq m$. In the example above, the set C_S consists of the set of all triples $([a_1], [a_2], \rho)$ such that $[a_1] \times [a_2]$ is in the rectangle marked by the dashed line in the figure and ρ is the order type $x < y$.

The idea behind the definition of the relation in the invariant is to use the set C_R as a finite relation carrying all the information necessary to restore the original database relation R.

Note that the set $ord(m)$ of different m-order types is finite for all m. Thus we can assign to each order type $p \in ord(m)$ a binary word $\xi_m(p) \in \{0, 1\}^{\ell(m)}$ where $\ell(m) := \min\{\ell : 2^\ell \geq |ord(m)|\}$. For $m = 2$ we define ξ_2 to be the encoding taking $x < y$ to 00, $x = y$ to 01, and $y < x$ to 10. Once such an encoding ξ_m is fixed, the set C_R can be represented by a set $C'_R := \{([a_1], \dots, [a_m], \bar{t}) : ([a_1], \dots, [a_m], \rho) \in C_R \text{ and } \xi(\rho) = \bar{t}\}$. This gives the definition of the relations in the invariant.

Definition 10. Let $\sigma := \{R_1, \dots, R_k\}$ and $\mathfrak{B}$ be a σ-database over $(\mathbb{R}, <)$. The *invariant* $\mathfrak{B}'$ of $\mathfrak{B}$ is a finite structure with universe U over the signature $\{<, R'_1, \dots, R'_k\}$, where

- $U := \mathbb{R}_{/\sim}$,
- $[x] < [y]$ if and only if $x < y$ and $x \not\sim y$, and
- If $R \in \sigma$ has arity m, then the corresponding relation R' has arity $m + \ell(m)$ and $R'[a_1] \dots [a_m]t_1 \dots t_{\ell(m)}$ holds in $\mathfrak{B}'$ iff there are $b_1, \dots, b_m \in \mathbb{R}$ with $\xi_m(otp(\bar{b})) = t_1 \cdots t_{\ell(m)}$ so that $R_i^{\mathfrak{B}} b_1 \dots b_m$ and $[a_i] = [b_i]$ for $1 \leq i \leq m$.

The mapping *inv* is defined as the function taking databases to their invariants.

We also need a function taking the finite encoding of relations back to their representation.

Definition 11. Let S be a $(m + \ell(m))$-ary relation of the form indicated by Definition 10. The function

$$\hat{\pi} : S \mapsto \varphi_S(x_1, \dots, x_m) := \bigvee_{\bar{a}\bar{t} \in S} (\sigma_m(\bar{x}, \bar{t}) \wedge \bigwedge_{j=1}^{m} (x_j \sim a_j)),$$

maps S to a formula φ_S representing the corresponding relation on the original database. Here $\sigma_m(\bar{x}, \bar{i})$ is a formula stating that $\bar{x}$ satisfies the order type specified by $\bar{i}$. The corresponding function mapping relations on the invariant to finitely representable relations over the database is $\pi : S \mapsto \{\bar{a} \ : \ \mathfrak{A} \models \hat{\pi}(S)[\bar{a}]\}$.

Lemma 12. *The invariant $inv(\mathfrak{B})$ is an ordered finite structure whose cardinality is linearly bounded in the size of any representation of $\mathfrak{B}$.*

Proof. For any set P, the number of 1-order types over P is $2|P| + 1$. The cardinality of $inv(\mathfrak{B})$ is the number of 1-order types over $P^{\mathfrak{B},\psi}$. Recall that $|P^{\mathfrak{B},\psi}| = |cp(\mathfrak{B})| + O(1)$ (since ψ is considered fixed) and that the size of $cp(\mathfrak{B})$ is bounded by the size of any representation of $\mathfrak{B}$. □

Corollary 13. *The functions inv and $\hat{\pi}$ can be computed in* LOGSPACE.

Proof. The LOGSPACE-computability of inv is a direct implication of the previous lemma and a result by Kanellakis, Kuper and, Revesz stating that first-order queries can be evaluated in LOGSPACE. For $\hat{\pi}$, let S be a $(m + \ell(m))$-ary answer of a query on an invariant. As an implication of the previous lemma, the size of S is polynomially bounded in the size of any representation of $\mathfrak{B}$. All the algorithm to calculate $\hat{\pi}(S)$ has to do is to output the disjunction of the formulae $(\sigma_m(\bar{x}, \bar{i}) \wedge \bigwedge_{j=1}^{k}(x_j \sim a_j))$ for every tuple $\bar{a}\bar{i} \in S$. Clearly, this can be done in LOGSPACE. □

4.2 The transformation of the query

Having defined the invariant of a database, we have to explain how the query has to be transformed for evaluation in the invariant. This translation of the formulae follows the same ideas described above, namely to increase the arity of the relations to store the order type. While translating a formula with free variables $\{x_1, \ldots, x_m\}$ we introduce new free variables $\bar{i}$ to hold the order type.

It will be necessary to compare order types over a different number of variables. Suppose that ρ_1, ρ_2 are order types in the variables $x_1, \ldots, x_m$ and $x_1, \ldots, x_n$, respectively, where $m \leq n$. We say that ρ_2 *extends* ρ_1, if $\rho_1 \subseteq \rho_2$. This means that the order type ρ_2 behaves on $x_1, \ldots, x_m$ in the same way as ρ_1. In the query transformation we need a formula $extends_{mn}(\bar{i}, \bar{j})$ stating that $\bar{i} := i_1, \ldots, i_{\ell(m)}$ codes some m-order type ρ_1, $\bar{j} := j_1, \ldots, j_{\ell(n)}$ codes a n-order type ρ_2, and ρ_2 extends ρ_1. The formula is defined as

$$extends_{mn}(\bar{i}, \bar{j}) := \bigvee_{\rho_2 \in ord(n)} (\xi_n(\rho_2) = \bar{j} \rightarrow \bigvee_{\substack{\rho_1 \in ord(m) \\ \rho_2 \text{ extends } \rho_1}} \xi_m(\rho_1) = \bar{i}).$$

Definition 14. Suppose σ is a database schema and τ the signature of the invariants corresponding to σ-databases. Further, let $\mathcal{L}$ be a logic from {FO, FO+DTC, FO+TC, FO+LFP, FO+PFP}. $f : \mathcal{L}[\sigma] \rightarrow \mathcal{L}[\tau]$ is defined inductively as follows.

- Let $\psi(x,y) := x < y$. Then $(f\psi)(x,y,i_1,i_2) := x \le y \wedge i_1 = 0 \wedge i_2 = 0$.
- Let $\psi(x) := x < c$. Then $(f\psi)(x,i) := x < [c] \wedge i = 0$.
- An equality $\psi(x,y) := x = y$ is translated to $(f\psi)(x,y,i_1,i_2) := x = y \wedge i_1 = 0 \wedge i_2 = 1$.
- An equality $\psi(x) := x = c$ corresponds to $(f\psi)(x,i) := x = [c] \wedge i = 0$.
- Let $\psi(x_1,\dots,x_j) := R_i u_1 \dots u_m$ where the u_i are either constants or variables from $\{x_1,\dots,x_j\}$ and all x_i occur in $\{u_1,\dots,u_m\}$. Then $(f\psi)(x_1,\dots,x_j,i_1,\dots,i_{\ell(j)}) := R'_i v_1 \dots v_m \bar{i}$, where $v_r := \begin{cases} x_s & \text{if } u_r = x_s, \\ [c] & \text{if } u_r = c. \end{cases}$
- Let $\psi(x_1,\dots,x_m) := \psi_1(y_1,\dots,y_{m_1}) \wedge \psi_2(z_1,\dots,z_{m_2})$, where all y_i and z_i occur in $\bar{x}$. Let $\bar{i} := i_1,\dots,i_{\ell(m)}$, $\bar{j} := j_1,\dots,j_{\ell(m_1)}$, and $\bar{j}' := j'_1,\dots,j'_{\ell(m_2)}$. Then $(f\psi)(\bar{x},\bar{i}) := \exists\bar{j}\exists\bar{j}'\, extends_{m_1 m}(\bar{j},\bar{i}) \wedge extends_{m_2 m}(\bar{j}',\bar{i}) \wedge (f\psi_1)(\bar{y},\bar{j}) \wedge (f\psi_2)(\bar{z},\bar{j}')$.
- For $\psi := \neg\varphi$, set $(f\psi) := \neg(f\varphi)$.
- Let $\psi(x_1,\dots,x_m) := \exists y\ \varphi(\bar{x},y)$. Then $(f\psi)(x_1,\dots,x_m,\bar{i}) := \exists y \exists j_1,\dots, \exists j_{\ell(m+1)} extend_{m(m+1)}(\bar{i},\bar{j}) \wedge (f\varphi)(\bar{x},y,\bar{j})$.
- Let $\psi(\bar{u},\bar{v}) := [\mathrm{DTC}_{\bar{x},\bar{y}}\varphi(\bar{x},\bar{y})](\bar{u},\bar{v})$.
  Then $(f\psi)(\bar{u},\bar{v},\bar{i}) := [DTC_{\bar{x},\bar{y},\bar{j}}(f\varphi)(\bar{x},\bar{y},\bar{j})](\bar{u},\bar{v},\bar{i})$.
- Let $\psi(\bar{u}) := [\mathrm{LFP}_{R,\bar{x}}\varphi(R,\bar{x})](\bar{u})$.
  Then $(f\psi)(\bar{u},\bar{i}) := [\mathrm{LFP}_{R',\bar{x},\bar{j}}(f\varphi)(R',\bar{x},\bar{j})](\bar{u},\bar{i})$.
- The rules for the TC, IFP- and PFP-operators are defined analogously.

All parts of the evaluation algorithm have now been defined. The next theorem proves its correctness.

Theorem 15. *Let $\psi \in \mathcal{L}$, where $\mathcal{L}$ is one of the logics in Definition 14, be a query, $\mathfrak{B}$ be a database over $(\mathbb{R},<)$ and $\mathfrak{B}' := inv(\mathfrak{B})$ be the invariant corresponding to $\mathfrak{B}$. Then $\psi^{\mathfrak{B}} = \pi((f\psi)^{\mathfrak{B}'})$.*

Proof. The proof is by induction on the structure of the query. The argument for the boolean operations is straightforward and therefore omitted. Also, we only give the argument for the LFP-operator and omit the cases of formulae built by DTC, TC, and PFP-operators which are treated in precisely the same way.

• For $\psi(x,y) := x < y$, the set $\psi^{\mathfrak{B}}$ contains the pairs $(a,b) \in \mathbb{R}^2$ such that $a < b$. By definition, $f(\psi)$ is $x \le y \wedge i_1 = 0 \wedge i_2 = 0$. Evaluating $(f\psi)$ on $\mathfrak{B}'$ results in the set $C := \{(a,b,i_1,i_2)\ :\ a \le b, i_1 = 0, i_2 = 0\}$. Transforming this set with the mapping $\hat{\pi}$ yields the formula $\varphi_C(x,y) := \bigvee_{(a,b,i_1,i_2)\in C}(\sigma_2(x,y,i_1,i_2) \wedge x \sim a \wedge y \sim b)$. As i_1 and i_2 are 0 for all tuples $(a,b,i_1,i_2) \in C$, $\sigma_2(x,y,i_1,i_2)$ reduces to $x < y$ and thus $\pi(C)$ equals $\{(a,b) \in \mathbb{R}^2\ :\ a < b\}$.

• Let $\psi(x) := x = c$. Then $(f\psi)(x,i) := x = [c] \wedge i = 0$ and $(f\psi)$ evaluates on $\mathfrak{B}'$ to the set $C := \{([c],0)\}$. Thus $\hat{\pi}(C)$ results in the formula $\varphi(x) := \sigma_1(x,0) \wedge x \sim c$. This formula is satisfied only by c because $c \in P$ and therefore the only member of $[c]$ is c itself. We get $\pi(C) := \{c\} = \psi^{\mathfrak{B}}$.

• Let $\psi(x_1,\dots,x_j) := R_s u_1 \dots u_m$ as in Definition 14. We assume w.l.o.g. that the first arguments of the relation are the variables and the parameters come thereafter, that is $u_1 = x_1,\dots,u_j = x_j$ and $u_{l+1} = c_1,\dots,u_m = c_{m-j}$. The transformed query is $(f\psi)(x_1,\dots,x_j,\bar{i}) := R'_s x_1 \dots x_j [c_1] \dots [c_{m-j}]\bar{i}$. Evaluating $f(\psi)$ on $\mathfrak{B}'$ yields the set $C := \{([a_1],\dots,[a_j],[c_1],\dots,[c_{m-j}],\bar{i}) \in R_s'^{\mathfrak{B}'}\}$. Now we have to show that $\pi(C) = \psi^{\mathfrak{B}}$. Suppose that $(a_1,\dots,a_m) \in \pi(C)$. Then there is a disjunct $\varphi := \sigma_m(x_1,\dots,x_m,\bar{i}) \wedge \bigwedge_r (x_r \sim b_r)$ in $\hat{\pi}(C)$ with $(\bar{b},\bar{i}) \in C$ and $\mathfrak{B} \models \varphi(\bar{a})$. As $(\bar{b},\bar{i}) \in R'^{\mathfrak{B}'}$ and therefore, by Definition 10, $(a_1,\dots,a_m) \in R^{\mathfrak{B}}$ we get $\bar{a} \in \psi^{\mathfrak{B}}$. Conversely, suppose that $(a_1,\dots,a_m) \in R^{\mathfrak{B}}$. Then $([a_1],\dots,[a_m],\bar{i})$ is in $R'^{\mathfrak{B}'}$, where $\xi_m(otp(\bar{a})) = \bar{i}$, and $\sigma_m(\bar{x},\bar{i}) \wedge \bigwedge_r a_r \sim x_r$ occurs as a disjunct in $\hat{\pi}(C)$. Obviously this formula is satisfied by $\bar{a}$ and therefore $\bar{a} \in \pi(C)$.

- Let $\psi(x_1,\dots,x_m) := \exists y\, \varphi(\bar{x},y)$. The transformed formula is $(f\psi)(\bar{x},\bar{i}) := \exists y \exists j_1,\dots,j_{\ell(m+1)} extend_{m(m+1)}(\bar{i},\bar{j}) \wedge (f\varphi)(\bar{x},y,\bar{j})$. Suppose that $(a_1,\dots,a_k) \in \psi^{\mathfrak{B}}$. This is the case if and only in there is an a_{m+1} with $(a_1,\dots,a_m,a_{m+1}) \in \varphi^{\mathfrak{B}}$. By induction $\varphi^{\mathfrak{B}} = \pi((f\varphi)^{\mathfrak{B}'})$. Thus there is a tuple $([a_1],\dots,[a_{m+1}],\bar{j}) \in (f\varphi)^{\mathfrak{B}'}$ and $(a_1,\dots,a_{m+1})$ satisfies the $(m+1)$-order type ρ denoted by $\bar{j}$. This is the case if and only if there is a tuple $([a_1],\dots,[a_m],\bar{i}) \in (f\psi)^{\mathfrak{B}'}$ such that ρ extends the order type denoted by $\bar{i}$. Thus we get that $(a_1,\dots,a_m) \in \psi^{\mathfrak{B}}$ if and only if $([a_1],\dots,[a_m],\bar{i}) \in (f\psi)^{\mathfrak{B}'}$, where $(a_1,\dots,a_m)$ satisfies the order type denoted by $\bar{i}$. This implies that $\psi^{\mathfrak{B}} = \pi((f\psi)^{\mathfrak{B}'})$.
- Finally, let $\psi(\bar{u}) := [\mathrm{LFP}_{R,\bar{x}}\varphi(R,\bar{x})](\bar{u})$. straightforward. We can assume that φ does not contain an LFP-operator. The proof then is straightforward. □

Now all parts of the evaluation method are defined. We illustrate the method in the following figure.

$$\begin{tikzcd}
\mathfrak{B} \arrow[r, "Q"] \arrow[d, shift right, "inv"'] & (\mathfrak{B}, Q(\mathfrak{B})) \arrow[d, shift right, "inv"'] \\
\mathfrak{B}' \arrow[r, "Q'"] \arrow[u, shift right, "\pi"'] & (\mathfrak{B}', Q'(\mathfrak{B}')) \arrow[u, shift right, "\pi"']
\end{tikzcd}$$

To evaluate the query Q (considered as being fixed) in the database $\mathfrak{B}$, the invariant $\mathfrak{B}' := inv(\mathfrak{B})$ is constructed, the transformed query $Q' := f(Q)$ is evaluated in $\mathfrak{B}'$, and the result is transformed back via the map $\hat{\pi}$. By Corollary 13 the mappings inv and $\hat{\pi}$ are LOGSPACE-computable. Thus we get the following theorem.

Theorem 16. *Suppose $\mathcal{L} \in \{\mathrm{FO}, \mathrm{FO{+}DTC}, \mathrm{FO{+}TC}, \mathrm{FO{+}LFP}, \mathrm{FO{+}IFP}, \mathrm{FO{+}PFP}\}$ is a logic and $\mathcal{C}$ a complexity class so that the evaluation problem for $\mathcal{L}$ on finite databases is in $\mathcal{C}$. Then the evaluation problem for $\mathcal{L}$ on dense linear order databases is also in $\mathcal{C}$.*

4.3 Capturing complexity classes

We now use the invariant to lift the capturing results of descriptive complexity theory from finite ordered structures to dense linear order databases. The crucial observation is that $inv(\mathfrak{B})$ is interpretable in $\mathfrak{B}$. In particular, this will give us a transformation from formulae over the invariant to formulae over the database. See [9] for background on interpretations.

Definition 17. Let $\mathfrak{B} := (\mathbb{R}, <, R_1^{\mathfrak{B}}, \ldots, R_k^{\mathfrak{B}})$ a database with signature σ over $(\mathbb{R}, <)$, let $\mathfrak{B}' = inv(\mathfrak{B})$ its invariant, and τ be the signature of the invariant. The interpretation Γ interpreting $\mathfrak{B}'$ in $\mathfrak{B}$ is given by

(1) a surjective function $f_\Gamma : \mathbb{R} \to U$ defined as $f_\Gamma(x) := [x]$, and
(2) for each atomic τ-formula $\psi(x_1, \ldots, x_m)$ a formula $\psi_\Gamma(x_1, \ldots, x_m) \in \mathrm{FO}[\sigma]$ such that for all tuples $\bar{a} \in \mathbb{R}^m$: $\mathfrak{B}' \models \psi(f_\Gamma(\bar{a}))$ if and only if $\mathfrak{B} \models \psi_\Gamma(\bar{a})$.

An equality $u = v \in \mathrm{FO}[\tau]$ corresponds to $u \sim v$, where u, v denote either variables or parameters from $P^{\mathfrak{B},\psi}$ (recall that $\sim$ is first-order definable). The translations for all other atomic formulae is given according to Definition 10. That is, a formula $u < v \in \mathrm{FO}[\tau]$ corresponds to $u < v \wedge \neg u \sim v$ and $R'_s\bar{x}\bar{i}$ to $\exists \bar{y}\, R_s\bar{y} \wedge \sigma_{ar(R_s)}(\bar{y}, \bar{i}) \wedge \bigwedge_j (x_j \sim y_j)$. (Recall the definition of σ_k from Definition 11).

We can now replace in any formula ψ of vocabulary τ in first-order logic, transitive closure logic or fixed point logic the atomic formula by their corresponding formulae and obtain a σ-formula ψ_Γ. The equivalence between ψ and ψ_Γ in part (2) of the definition thus extends to arbitrary formula in these logics.

We are now ready to lift the capturing results from finite ordered structures to dense linear order databases. Clearly, every, say, FO+LFP-query ψ is invariant under automorphisms on $\mathfrak{A}$ that preserve the constants in ψ. Thus we can only hope to capture those PTIME-queries which are invariant under such automorphisms. This is made precise in the following definition.

Definition 18. A complexity class $\mathcal{C}$ is captured by a logic $\mathcal{L}$ on the class of dense order databases, if for all queries Q in $\mathcal{C}$ for which we can choose a finite set $S \subset \mathbb{R}$ such that Q commutes with every automorphism on $(\mathbb{R}, <, S)$, there is a formula ψ in $\mathcal{L}$ satisfying the following property: For all dense order databases $\mathfrak{B}$ we have that $Q(\mathfrak{B})$ is true iff $\mathfrak{B} \models \psi$.

Theorem 19. *Let $\mathcal{L}$ be a logic as in Theorem 16 and $\mathcal{C}$ be a complexity class such that $\mathcal{L}$ captures $\mathcal{C}$ on the class of finite ordered structures. Then $\mathcal{L}$ captures $\mathcal{C}$ on the class of dense order databases.*

Proof. We give the proof explicitly only for FO+LFP. The other cases can be proven analogously. We have already shown that FO+LFP $\subseteq$ PTIME. For the other direction, suppose that Q a polynomial-time computable query on dense order constraint databases of signature σ. We show that there is an FO+LFP $[\sigma]$-formula ψ_Q defining Q.

Again let τ denote the signature of the corresponding invariants. Let Q' be the query that takes invariants $inv(\mathfrak{B})$ of databases $\mathfrak{B}$ as inputs and returns as output the set $Q'(\mathfrak{B}') := \{f_\Gamma(\bar{a}) : \bar{a} \in Q(\mathfrak{B})\}$. Clearly Q' can be computed in polynomial time, since a representation of the database $\mathfrak{B}$ whose invariant is given as the input can be computed in LOGSPACE and since Q is a PTIME-query. (Note that in contrast to the algorithm of the previous section this algorithm constructs the database from the invariant and evaluates the query in the database, whereas the algorithm in the previous section constructs the invariant from the database and then operates on the invariant.)

Since Q' is a PTIME-query on finite ordered structures, there exists by the Theorem of Immerman and Vardi (see [1, 10]) an FO+LFP $[\tau]$-formula φ that defines Q'. By the remarks above, there exists a formula $\varphi_\Gamma \in$ FO+LFP $[\sigma]$ such that for all $\bar{a} \in \mathbb{R}^m$, $inv(\mathfrak{B}) \models \varphi(f_\Gamma(\bar{a}))$ iff $\mathfrak{B} \models \varphi_\Gamma(\bar{a})$. Thus $\mathfrak{B} \models \varphi_\Gamma(\bar{a})$ if and only if $\bar{a} \in Q(\mathfrak{B})$. This proves the theorem. □

The following table summarizes the relations between logics and complexity classes in the context of dense linear orders.

Logics and complexity classes in the context of dense linear orders.

FO+DTC	=	LOGSPACE
FO+TC	=	NLOGSPACE
FO+LFP	=	PTIME
FO+PFP	=	PSPACE

5 Summary and Further Results

In the main result of this paper we presented a general method to prove complexity bounds for query languages over dense order databases. The idea was to code the finitely represented database as a finite database and then use the evaluation algorithms available for the query language on finite databases. It turned out that this encoding can be defined by first-order formulae using only the order predicate and some very limited kind of arithmetic. It can therefore be done with very low data complexity. This method enabled us to evaluate queries for various query languages within the same complexity classes as for finite databases.

This method also works for databases defined by inequality constraints over a countable infinite set. By a simple argument based on Ehrenfeucht-Fraïssé games we can also prove that the various fixed-point logics considered before are too weak to express all LOGSPACE-computable queries.

Unfortunately the good results for dense order databases cannot be extended to linear constraint databases over the reals. As soon as we admit recursion in the query language the arithmetic over $\mathbb{N}$ becomes definable and thus the query language undecidable.

The situation changes drastically if structures with a discrete order as universe are considered. It is known that positive DATALOG-queries on discrete order databases can be evaluated in closed form (see [14]) but the data complexity is still unknown. For first-order queries a better result can be shown.

Theorem 20. *First-order queries on discrete order databases can be evaluated in* LOGSPACE.

See [13] for a proof of the theorem. In Section 3 we have shown that the data complexity of first-order queries over $(\mathbb{N}, <, +)$ is in the polynomial time hierarchy and that there are complete first-order queries for all levels of PH.

As in the case of the context structure $(\mathbb{R}, <, +)$, adding recursion to the query language leads to undecidable query languages. Of course, even first-order queries are undecidable if we also add multiplication to the context structure.

The following table summarizes the results. The NC bound for first-order queries on databases over the field of reals comes from [12]. Note that only in the case of $(\mathbb{R}, <)$ we have precise capturing results. The other cases are just complexity bounds.

	inequality	$(\mathbb{R}, <)$	$(\mathbb{R}, <, +)$	$(\mathbb{R}, <, +, \cdot)$	$(\mathbb{N}, <_c)$	$(\mathbb{N}, <, +)$	$(\mathbb{N}, <, +, \cdot)$
FO	AC^0	AC^0	NC	NC	LOGSPACE	PH	n.d.
FO+DTC	LOGSPACE	LOGSPACE	n.d.	n.d.	n.d.	n.d.	n.d.
FO+TC	NLOGSPACE	NLOGSPACE	n.d.	n.d.	n.d.	n.d.	n.d.
FO+LFP	PTIME	PTIME	n.d.	n.d.	n.d.	n.d.	n.d.
FO+PFP	PSPACE	PSPACE	n.d.	n.d.	n.d.	n.d.	n.d.

n.d. = not decidable

References

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

[2] E. Grädel. Subclasses of Presburger arithmetic and the polynomial-time hierarchy. *Theoretical Computer Science*, 56:289–301, 1988.

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

[4] E. Grädel and K. Meer. Descriptive complexity theory over the real numbers. In *Mathematics of Numerical Analysis: Real Number Algorithms*, volume 32 of *AMS Lectures in Applied Mathematics*, pages 381–403. 1996.

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

[6] S. Grumbach and J. Su. Queries with arithmetical constraints. *Theoretical Computer Science*, 173:151–181, 1997.

[7] S. Grumbach, J. Su, and C. Tollu. Linear constraint query languages: Expressive power and complexity. *Lecture Notes in Computer Science*, 960:426–446, 1995.

[8] D. Harel. Towards a theory of recursive structures. volume 1450 of *Lecture Notes in Computer Science*, pages 36–53. Springer, 1998.

[9] W. Hodges. *Model Theory.* Cambridge University Press, 1993.

[10] N. Immerman. *Descriptive complexity.* Graduate Texts in Computer Science. Springer, 1998.

[11] P. Kanellakis and D. Goldin. Constraint programming and database query languages. volume 789 of *Lecture Notes in Computer Science*, pages 96–120. Springer, 1994.

[12] P. Kanellakis, G. Kuper, and P. Revesz. Constraint query languages. *Journal of Computer and Systems Sciences*, 51:26–52, 1995.

[13] S. Kreutzer. Descriptive complexity theory for constraint databases. Diplomarbeit, RWTH Aachen, 1999.

[14] P. Revesz. A closed form evaluation for datalog queries with integer (gap)-order constraints. *Theoretical Computer Science*, 116:117–149, 1993.

[15] U. Schöning. Complexity of Presburger arithmetic with fixed quantifier dimension. *Theory of Computing Systems*, 30:423–428, 1997.