# On the Expressiveness of First-Order Constraint Languages

Foto Afrati, Theodore Andronikos, Theodore Kavalieros

National Technical University of Athens, Greece

**Abstract.** We investigate the expressive power of query languages for constraint databases. We examine languages over first order logic enhanced with the inequality constraint and linear constraints. We show that the query "one-line" is not expressible in *FO+order* (i.e. inequality constraint). In fact, the result is stronger in that we prove that it is not expressible in any higher order logic either. Furthermore, we show that the same query can be expressed in *FO+linear* by giving a non-trivial sentence that characterizes the class of databases consisting of exactly one line.

## 1 Introduction

In classical database theory, databases are modeled as relational structures and they are considered to be finite collections of data. Today, many applications such as geographical information systems, that involve representation and handling of spatial data, prove that this traditional view of a database is not adequate. The need for more general data models that allow infinite collections of items to be stored in the database is very intense. Of course we would like to have a finite representation of these infinite collections, otherwise this would not facilitate our work.

Kanellakis, Kuper and Revesz in [KKR90], introduced constraint query languages and proposed the integration of database and constraint solving methods as a solution to the problems that arise in modern applications. This idea creates a very promising research area in the field of Constraint Databases ([BJM93], [GS94], [Kup90], [Kup93], [PBG95], [Re90]). The basic idea is the generalization of the relations of the relational model ([Cod70]) by defining generalized tuples as conjunctions of constraints. For instance, $(x = y \wedge x < 3)$ defines a generalized tuple of arity 2 and a tuple $(a, b)$ is a shorthand for the formula $(x = a \wedge y = b)$. Consequently, a generalized relation is a finite set of such generalized tuples. In this way we are able to define an infinite set of points over $R^2$.

The declarative style of database query languages is an important aspect of database systems, that has been at the core of the relational data model since Codd's pioneering work. Indeed, having such a language for ad-hoc database querying is a requirement in today's commercial relational technology. Querying in Codd's relational calculus is a form of limited (but very successful) constraint programming. It is rather surprising that other advances in constraint programming have not yet influenced database query language design. However, it is believed that this will soon change and there will be integration of constraint programming and databases.

The relational calculus over finitely representable relations, proposed in [KKR90], constitutes a constraint query language which admits an efficient bottom-up evaluation and declarative semantics. In [KG94] an algebra for dense order constraints is presented that is even simpler to evaluate than the calculus in [KKR90].

The data complexity of both the relational calculus and inflationary Datalog with negation over constraint databases is studied in [KKR90]. Different types of constraints are considered, such as dense linear order inequalities, real polynomial inequalities, etc. These constraints are based on decidable theories which admit elimination of quantifiers. It is proved that, although the decision problem of the underlying theories must be high, the data complexity of the query languages is tractable. This shows the practical interest of this approach.

The main contribution of the CQL framework is to demonstrate that it is possible to bridge the gap between: bottom-up, efficient, declarative database programming and efficient constraint solving. The intuition comes from CLP ([JL87]): a conjunction of quantifier-free constraints over k-variables, where k depends on the database schema and not the instance, is the correct generalization of the k-tuple or record or ground fact. The technical tools for this integration are: data complexity from database theory, and quantifier elimination methods from mathematical logic.

In this paper we consider databases consisting of a binary relation $L$. For this class of databases many interesting queries have been examined in [ACGK94] such as: the *in-line* query, that asks whether $L$ is contained in a line, the *exists-line* query that asks whether $L$ contains a line, the *lines* query that returns the set of lines contained in $L$, and the *line-intersection* query that returns all points $P$ which are intersections of (pairs of distinct) lines contained in $L$. These queries are used as benchmarks to study the expressive power of various versions of first order query languages with constraints.

Here, we carry on investigation which concerns the expressiveness of *FO+order* and *FO+linear*. Relevant expressivenes questions are studied in [GS95]. We examine the Boolean query *one-line* that asks whether $L$ (the database) is precisely one line. Our first result states that we cannot express this query in *FO+order*. In fact we prove something stronger, in that we show an isomorphism between a database which consists of a single line and a database which consists of a curve which represents a function but is not a line. Therefore this query is not expressible even in any higher order logic. Furthermore, we have shown that the same query can be expressed in *FO+linear* by giving a non-trivial sentence that characterizes the class of databases consisting of exactly one line.

The rest of the paper is organized as follows: In Section 2 we present some basic definitions from the fields of constraint databases and mathematical logic. In section 3 we review some of the previous results in the specific field and we prove our result about the expressiveness of *FO+order*. In Section 4 we explain how the *one-line* query can be expressed in an extended language $\mathcal{L}$ in which the set of predicate symbols is $\{=, <, +\} \cup \sigma$. In other words we show that it is expressible in *FO+linear*. Section 5 concludes and presents future research directions.

## 2 Preliminaries

In this section we present some basic definitions. First of all, we define *databases* as subsets of the $k$-dimensional Euclidean space $R^k$ ($R$ is the real line). A *query* is a function from databases to databases and *Boolean* queries are functions from databases to $\{true, false\}$.

- A very interesting and powerful class of constraints is *FO+poly* [KKR90] which is the set of first-order formulas (with equality) over atomic formulas as follows:
  (i) $S(x_1, \ldots, x_k)$, (which states that point $(x_1, \ldots, x_k)$ is in the database $S$).
  (ii) *Polynomial constraints* of the form
  $$f(x_1, \ldots, x_k)\ \theta\ 0$$
  where $f$ is a $k$-variable polynomial (with real coefficients) and $\theta \in \{>, =\}$.
- *FO+linear* is the subset of *FO+poly* obtained by restricting constraints to be linear.
- $FO + order$ is the subset of *FO+linear* obtained by restricting constraints to contain only $<, =$.
- $FO + lines$ is an extension of *FO+linear* with variables ranging over points and lines.

Formulas of *FO+poly* with free variables define queries: the output of such a query is exactly the set of tuples satisfying the formula. Sentences define Boolean queries. If the input to a *FO+poly* query is defined by a Boolean combination of polynomial constraints, the output is also defined by such a combination, as shown in [KKR90]. We say that a database is a *linear* database if it is a subset of $\mathcal{R}^k$ defined by a Boolean combination of linear constraints. A *linear* query is a function from linear databases to linear databases. Formulas of *FO+linear* with free variables define linear queries.

- Two structures $\mathcal{U}$ and $\mathcal{B}$ in a language $\mathcal{L}$ are *elementarily equivalent* ($\mathcal{U} \equiv \mathcal{B}$) ([En72]) iff for any first-order sentence $\phi$
  $$\models_{\mathcal{U}} \phi \Leftrightarrow \models_{\mathcal{B}} \phi$$
- Two structures $\mathcal{U}$ and $\mathcal{B}$ in a language $\mathcal{L}$ are *isomorphic* ($\mathcal{U} \cong \mathcal{B}$) iff there exists a function $h : \mid \mathcal{U} \mid \rightarrow \mid \mathcal{B} \mid$ which is 1-1 and onto such that for each $n$-place predicate symbol $P$ of the language and each $n$-tuple $(a_1, \ldots, a_n)$ of elements of $\mid \mathcal{U} \mid$:
  $$P^{\mathcal{U}}(a_1, \ldots, a_n) \Leftrightarrow P^{\mathcal{B}}(h(a_1), \ldots, h(a_n))$$

– If two structures $\mathcal{U}$ and $\mathcal{B}$ in a language $\mathcal{L}$ are *isomorphic* ([En72]) then for any sentence $\phi$

$$\models_{\mathcal{U}} \phi \Leftrightarrow \models_{\mathcal{B}} \phi$$

## 3 Query Languages with Order

We consider a *first order language* $\mathcal{L}_1$ that consists of:

1. A set of logical symbols; these are all the usual first order logical symbols, plus the equality symbol $=$ and the order symbol $<$, which, being logical symbols, are regarded as *built-in predicates*, i.e. they have a fixed a priori meaning (the intuitive meaning) and are not open to any other interpretation.
2. A set of nonlogical symbols, that contains the binary predicate symbol $L$. This set, denoted $\sigma$, is called the *the database schema* (or *signature*), i.e. $\sigma = \{L\}$.

In this paper we consider databases consisting of a binary relation $L$. For this class of databases we can define several queries (see also [ACGK94]). In the following queries, $\Gamma$ denotes the graph of exactly one line.

– The Boolean query *in-line*; this query asks whether $L$ is contained in a line, i.e whether $L \subseteq \Gamma$.
– The Boolean query *exists-line*; this query asks whether $L$ contains a line, i.e whether $\Gamma \subseteq L$.
– The general query *lines* returns the set of lines contained in $L$.
– The general query *line-intersection* returns all points $P$ which are intersections of (pairs of distinct) lines contained in $L$.

The above queries have been studied in [ACGK94] where important results have been obtained.

In this work we focus on the the Boolean query *one-line*, denoted $QL_1$. This query asks whether $L$ (the database) is precisely one line, i.e. whether $L = \Gamma$. The main result of this section is Theorem 9 which states that this query cannot be expressed in logic of any order in any language $\mathcal{L}$ that has only the predicate symbols $\{=, <\} \cup \sigma$. In Section 4 we explain how this query can be expressed in the extended language $\mathcal{L}_2$ in which the set of predicate symbols is $\{=, <, +\} \cup \sigma$. In other words we show that it is expressible in *FO+linear*.

We must say that this last result does not contradict the result in [ACGK94] that the Boolean query *in-line* is not expressible in *FO+linear*. It should be clear that the class of databases characterized by the *one-line* query is a subclass of the class of databases characterized by the *in-line* query; for instance, a database consisting of just 3 points lying on a line belongs to the latter class but does not belong to the former class.

In order to derive this result, we shall first show that in the language $\mathcal{L}_1$ it is impossible to distinguish between a database consisting of exactly one line passing through the origin and a database consisting of a continuous curve which is identical to the first line except within a specific interval where it is slightly disfigured.

We proceed to define the problem formally:

We have two structures $D_1 =< R, =, <, L^1 >$ and $D_2 =< R, =, <, L^2 >$ over $\sigma$. In both structures the universe is the set of reals $R$ and the symbols $=$ and $<$ are interpreted as the usual *"equality"* and *"less than"* relations over $R^2$. Therefore in order to simplify notation we have dropped the superscripts from the symbols $=$ and $<$ and instead of $=^1$ $(=^2)$ and $<^1$ $(<^2)$ we simply write $=$ and $<$. The structures differ on the interpretation of $L$.

- In $D_1$, $L^1$ is the graph of line $y = cx$, where $c$ is an arbitrary real $> 1$. Intuitively, $D_1$ is a database that contains *exactly one line*, passing through the origin.
- In $D_2$, $L^2$ is the graph of the following curve:

$$y = \begin{cases} cx, & x \in (-\infty, 1] \\ (c+\varepsilon)x - \varepsilon, & x \in (1, (1+c)/2] \\ (c-\varepsilon)x + c\varepsilon, & x \in ((1+c)/2, c] \\ cx, & x \in (c, \infty) \end{cases}$$

  where $c$ is the same real as in $D_1$ and $\varepsilon$ is an arbitrary real such that $0 < \varepsilon < 1$. Intuitively, this means that $D_2$ is "almost" the same with $D_1$, except a small portion where it is distorded.

**Definition 1.** We define two sequences of intervals in $R$:

- $I_0, I_1^f, I_1^g, I_2^f, I_2^g, \ldots$
- $J_0, J_1^f, J_1^g, J_2^f, J_2^g, \ldots$ as follows:

$$\begin{array}{ll} I_0 = (-\infty, c] & J_0 = (-\infty, c] \\ I_n^f = (c^n, c^{n-1}x_M], n \geq 1 & J_n^f = (c^n, c^{n-1}x_K], n \geq 1 \\ I_n^g = (c^{n-1}x_M, c^{n+1}], n \geq 1 & J_n^g = (c^{n-1}x_K, c^{n+1}], n \geq 1 \end{array}$$

where $x_M = c(1+c)/2$ and $x_K = (c^2 + c + c\varepsilon - \varepsilon)/2$ (constants $c$ and $\varepsilon$ are the same as in $D_2$).

It should be clear that

- $I_0 \cap I_n^f = \emptyset, \forall n \in N$.
- $I_0 \cap I_n^g = \emptyset, \forall n \in N$.
- $I_m^f \cap I_n^f = \emptyset, \forall m \forall n \in N$ such that $m \neq n$.
- $I_m^g \cap I_n^g = \emptyset, \forall m \forall n \in N$ such that $m \neq n$.
- $I_m^f \cap I_n^g = \emptyset, \forall m \forall n \in N$.
- $I_0 \cup (\cup_{n \geq 1} I_n^f) \cup (\cup_{m \geq 1} I_m^g) = R$.

Analogous properties hold for the other sequence of intervals. For our convenience we can give the following definitions:

– $SI_n^f = (\cup_{k\geq n} I_k^f) \cup (\cup_{k\geq n} I_k^g) = (c^n, \infty)$ and $SI_n^g = (\cup_{k\geq n} I_k^g) \cup (\cup_{k\geq n+1} I_k^f) = (c^{n-1}x_M, \infty)$.
– $SJ_n^f = (\cup_{k\geq n} J_k^f) \cup (\cup_{k\geq n} J_k^g) = (c^n, \infty)$ and $SJ_n^g = (\cup_{k\geq n} J_k^g) \cup (\cup_{k\geq n+1} I_k^f) = (c^{n-1}x_K, \infty)$.

We also extend the notion of order between intervals as follows:

– Given two intervals $I$ and $J$, we write $I < J$ iff $\forall x \in I, \forall y \in J, x < y$.

Under this notion we see the following relations hold:

– $I_0 < I_n^f, I_0 < I_n^g, I_n^f < I_n^g, I_n^g < I_{n+1}^f, n \geq 1$, i.e. $I_0 < I_1^f < I_1^g < I_2^f < I_2^g, \ldots$ $(R_1)$.
– $J_0 < J_n^f, J_0 < J_n^g, J_n^f < J_n^g, J_n^g < J_{n+1}^f, n \geq 1$, i.e. $J_0 < J_1^f < J_1^g < J_2^f < J_2^g, \ldots$ $(R_2)$.

**Definition 2.** We define the function $h : R \to R$ as follows:

$$h(x) = \begin{cases} h_0(x) = x & x \in I_0 \\ f_1(x) = (c+\varepsilon)(x/c) - \varepsilon & x \in I_1^f \\ g_1(x) = (c-\varepsilon)(x/c) + c\varepsilon & x \in I_1^g \\ f_n(x) = cf_{n-1}(x/c) = c^{n-1}f_1(x/c^{n-1}), n \geq 2 & x \in I_n^f, n \geq 2 \\ g_n(x) = cg_{n-1}(x/c) = c^{n-1}g_1(x/c^{n-1}), n \geq 2 & x \in I_n^g, n \geq 2 \end{cases}$$

The inverse function $h^{-1} : R \to R$ is defined as follows:

$$h^{-1}(x) = \begin{cases} h_0^{-1}(x) = x & x \in J_0 \\ f_1^{-1}(x) = c(x+\varepsilon)/(c+\varepsilon) & x \in J_1^f \\ g_1^{-1}(x) = c(x-c\varepsilon)/(c-\varepsilon) & x \in J_1^g \\ f_n^{-1}(x) = cf_{n-1}^{-1}(x/c) = c^{n-1}f_1^{-1}(x/c^{n-1}), n \geq 2 & x \in J_n^f, n \geq 2 \\ g_n^{-1}(x) = cg_{n-1}^{-1}(x/c) = c^{n-1}g_1^{-1}(x/c^{n-1}), n \geq 2 & x \in J_n^g, n \geq 2 \end{cases}$$

**Lemma 3.** *The following hold:*

1. *$h$ is one-to-one.*
2. *$h_0 \uparrow (h_0^{-1} \uparrow), f_n \uparrow, (f_n^{-1} \uparrow)$ and $g_n \uparrow, (g_n^{-1} \uparrow), n \geq 1$, i.e. they are strictly increasing.*
3. *$h[I_0] = J_0, h[I_n^f] = J_n^f, h[I_n^g] = J_n^g, \forall m \forall n \in N$.*
4. *$h$ and $h^{-1}$ are one-to-one and onto (bijections).*

*Proof.* (Sketch)
(1) In order to prove that $h$ is one-to-one, it suffices to prove that:

- $h_0$ is 1-1, which is trivial.
- $f_n$ is 1-1, $n \geq 1$. We can prove this by induction on $n$.
  - $f_1$ is 1-1; $f_1(x_1) \neq f_1(x_2) \Rightarrow (c+\varepsilon)(x_1/c) - \varepsilon \neq (c+\varepsilon)(x_2/c) - \varepsilon \Rightarrow x_1 \neq x_2$.
  - We assume that $f_n$ is 1-1 and we shall prove that $f_{n+1}$ is 1-1: $f_{n+1}(x_1) \neq f_{n+1}(x_2) \Rightarrow cf_n(x_1/c) \neq cf_n(x_2/c) \Rightarrow f_n(x_1/c) \neq f_n(x_2/c) \Rightarrow x_1 \neq x_2$ by the inductive hypothesis.
- $g_n$ is 1-1, $n \geq 1$. We can prove this by induction on $n$, exactly as before.

(2) Monotonicity is easily proved in all cases by taking the first derivative of the appropriate function, which is positive in all cases.
(3) We now prove that the image of any interval $I$ under $h$ is the corresponding interval $J$.

- $h[I_0] = J_0$, which is obvious according to the definitions.
- $h[I_n^f] = J_n^f$, i.e. $f_n[I_n^f] = J_n^f, n \geq 1$. According to the definition of $f_n$, we get that $\forall x \in I_n^f, c^n < f_n(x) \leq c^{n-1}x_K$, i.e. $f_n[I_n^f] \subseteq J_n^f$. Let us assume now that there exists a $y \in J_n^f$, such that $y \neq f_n(x), \forall x \in I_n^f$. This is impossible because we can solve the equation $y = f_n(x)$ for $x$ and obtain the corresponding $x$ in the interval $I_n^f$.
- In an analogous manner we prove that $h[I_n^g] = J_n^g$ i.e. $g_n[I_n^g] = J_n^g, n \geq 1$.

(4) From (1) we know that $h$ is 1-1. From (3) we get that the image of $h$ is the whole of $R$ because $J_0 \cup (\cup_{n \geq 1} J_n^f) \cup (\cup_{m \geq 1} J_m^g) = R$. Therefore, $h$ is a bijection and its inverse function $h^{-1}$ exists and is also a bijection.

**Lemma 4.** *$a_i = a_j \Leftrightarrow h(a_i) = h(a_j), \forall a_i \forall a_j \in R$.*

*Proof.* Trivial because $h$ is one-to-one.

**Lemma 5.** *$a_i < a_j \Leftrightarrow h(a_i) < h(a_j), \forall a_i \forall a_j \in R$.*

*Proof.* ($\Rightarrow$) Initially, we shall prove that $a_i < a_j \Rightarrow h(a_i) < h(a_j), \forall a_i \forall a_j \in R$
We shall do that by considering all the possible intervals in which $a_i$ may belong.

- $a_i \in I_0$. We have the following obvious implications:
  $a_i < a_j \Rightarrow h(a_i) = a_i \wedge a_j \in I_0 \cup SI_1^f$,
  - $a_j \in I_0 \Rightarrow h(a_j) = a_j \Rightarrow h_0(a_i) < h_0(a_j)$ by Lemma 3(2).

- $a_j \in SI_1^f \Rightarrow a_j \leq h(a_j) \Rightarrow h(a_i) = a_i < a_j \leq h(a_j)$.
- $a_i \in I_n^f$, for some $n \geq 1$. Then $h(a_i) \in J_n^f \wedge a_j \in I_n^f \cup SI_n^g$,
  - $a_j \in I_n^f \Rightarrow f_n(a_i) < f_n(a_j)$ by Lemma 3(2).
  - $a_j \in SI_n^g \Rightarrow h(a_j) \in SJ_n^g \Rightarrow h(a_i) < h(a_j)$ by $(R_2)$.
- $a_i \in I_n^g$, for some $n \geq 2$. Then $h(a_i) \in J_n^g \wedge a_j \in I_n^g \cup SI_{n+1}^f$
  - $a_j \in g_n \Rightarrow g_n(a_i) < h_2^n(a_j)$ by Lemma 3(2).
  - $a_j \in SI_{n+1}^f \Rightarrow h(a_j) \in SJ_{n+1}^f \Rightarrow h(a_i) < h(a_j)$ by $(R_2)$.

($\Leftarrow$) We shall now prove the opposite direction, i.e. $h(a_i) < h(a_j) \Rightarrow a_i < a_j, \forall a_i \forall a_j \in R$. We shall do that by considering all the possible intervals in which $h(a_i)$ may belong.

- $h(a_i) \in J_0$. We have the following implications:
  $h(a_i) < h(a_j) \Rightarrow a_i = h(a_i) \in I_0 \wedge h(a_j) \in J_0 \cup SJ_1^f$,
  - $h(a_j) \in J_0 \Rightarrow a_j = h(a_j) \Rightarrow a_i < a_j$
  - $h(a_j) \in SJ_1^f \Rightarrow a_j \in SI_1^f \Rightarrow a_i < a_j$
- $h(a_i) \in J_n^f$, for some $n \geq 1$. We then have:
  $h(a_i) < h(a_j) \Rightarrow a_i \in I_n^f \wedge h(a_j) \in J_n^f \cup SJ_n^g$,
  - $h(a_j) \in J_n^f \Rightarrow a_i < a_j$ by Lemma 3(2).
  - $h(a_j) \in SJ_n^g \Rightarrow a_j \in SI_n^g \Rightarrow a_i < a_j$
- $h(a_i) \in J_n^g$, for some $n \geq 1$. We then have:
  $h(a_i) < h(a_j) \Rightarrow a_i \in I_n^g \wedge h(a_j) \in J_n^g \cup SJ_{n+1}^f$,
  - $h(a_j) \in J_n^g \Rightarrow a_i < a_j$ by Lemma 3(2).
  - $h(a_j) \in SJ_{n+1}^f \Rightarrow a_j \in SI_{n+1}^f \Rightarrow a_i < a_j$

This concludes the proof that $a_i < a_j \Leftrightarrow h(a_i) < h(a_j)$.

**Lemma 6.** $L^1(a_i, a_j) \Leftrightarrow L^2(h(a_i), h(a_j)), \forall a_i \forall a_j \in R$.

*Proof.* ($\Rightarrow$)As usual, we shall first prove that $L^1(a_i, a_j) \Rightarrow L^2(h(a_i), h(a_j)), \forall a_i \forall a_j \in R$.
Again, we shall do so by considering all the possible intervals in which $a_i$ may belong.

- $a_i \in I_0$. Then we have the following obvious implications:
  $h(a_i) = a_i \wedge L^1(a_i, a_j) \Rightarrow a_j = ca_i \Rightarrow a_j \in I_0 \cup I_1^f \cup I_1^g$
  - $a_j \in I_0 \Rightarrow a_i \leq 1 \wedge h(a_j) = a_j \Rightarrow h_0(a_j) = ch_0(a_i) \Rightarrow L^2(h(a_i), h(a_j))$
  - $a_j \in I_1^f \Rightarrow 1 < a_i \leq (1+c)/2 \wedge f_1(a_j) = (c+\varepsilon)(a_j/c) - \varepsilon = (c+\varepsilon)a_i - \varepsilon = (c+\varepsilon)h(a_i) - \varepsilon \Rightarrow L^2(h(a_i), h(a_j))$
  - $a_j \in I_1^g \Rightarrow (1+c)/2 < a_i \leq c \wedge g_1(a_j) = (c-\varepsilon)(a_j/c) + c\varepsilon = (c-\varepsilon)a_i + c\varepsilon = (c-\varepsilon)h(a_i) + c\varepsilon \Rightarrow L^2(h(a_i), h(a_j))$
- $a_i \in I_n^f$, for some $n \geq 1$. Then $f_n(a_i) = c^{n-1}[(c+\varepsilon)(a_i/c^n) - \varepsilon] \in J_n^f$.
  $L^1(a_i, a_j) \Rightarrow a_j = ca_i \Rightarrow a_j \in I_{n+1}^f \Rightarrow f_{n+1}(a_j) = c^n[(c+\varepsilon)(a_j/c^{n+1}) - \varepsilon] = c[c^{n-1}(c+\varepsilon)(a_i/c^n) - \varepsilon] = ch(a_i) \Rightarrow L^2(h(a_i), h(a_j))$
- $a_i \in I_n^g$, for some $n \geq 1$. Then $g_n(a_i) = c^{n-1}[(c-\varepsilon)(a_i/c^n) + c\varepsilon] \in J_n^g$.
  $L^1(a_i, a_j) \Rightarrow a_j = ca_i \Rightarrow a_j \in I_{n+1}^g \Rightarrow g_{n+1}(a_j) = c^n[(c-\varepsilon)(a_j/c^{n+1}) + c\varepsilon] = c[c^{n-1}(c-\varepsilon)(a_i/c^n) + c\varepsilon] = ch(a_i) \Rightarrow L^2(h(a_i), h(a_j))$

($\Leftarrow$) We now prove the opposite direction, i.e. $L^2(h(a_i), h(a_j)) \Rightarrow L^1(a_i, a_j), \forall a_i \forall a_j \in R$.
As before, we shall do that by considering all the possible intervals in which $h(a_i)$ may belong.

– $h(a_i) \in J_0$. Then we have:
  $a_i = h(a_i) \wedge L^2(h(a_i), h(a_j)) \Rightarrow h(a_j) \in J_0 \cup J_1^f \cup J_1^g$
  - $h(a_j) \in J_0 \Rightarrow h(a_i) \leq 1 \wedge h(a_j) \leq c \Rightarrow h_0(a_j) = a_j = ch_0(a_i) = ca_i \Rightarrow L^1(a_i, a_j)$
  - $h(a_j) \in J_1^f \Rightarrow 1 < h(a_i) \leq (1+c)/2 \wedge f_1(a_j) = (c+\varepsilon)h(a_i) - \varepsilon \Rightarrow a_j = f_1^{-1}(f_1(a_j)) = c(f_1(a_j)+\varepsilon)/(c+\varepsilon) = ch(a_i) = ca_i \Rightarrow L^1(a_i, a_j)$.
  - $h(a_j) \in J_1^g \Rightarrow (1+c)/2 < h(a_i) \leq c \wedge g_1(a_j) = (c-\varepsilon)h(a_i) + c\varepsilon \Rightarrow a_j = g_1^{-1}(g_1(a_j)) = c(g_1(a_j) - c\varepsilon)/(c-\varepsilon) = ch(a_i) = ca_i \Rightarrow L^1(a_i, a_j)$

– $h(a_i) \in J_n^f$, for some $n \geq 1$. Then $a_i = f_n^{-1}(h(a_i)) = c^n[(h(a_i)/c^{n-1}+\varepsilon)/(c+\varepsilon)]$.
  $L^2(h(a_i), h(a_j)) \Rightarrow h(a_j) = ch(a_i) \Rightarrow h(a_j) \in J_{n+1}^f \Rightarrow a_j = f_{n+1}^{-1}(h(a_j)) = c^n[c(h(a_j)/c^n+\varepsilon)/(c+\varepsilon)] = c[c^n(h(a_i)/c^{n-1}+\varepsilon)/(c+\varepsilon)] = ca_i \Rightarrow L^1(a_i, a_j)$.

– $h(a_i) \in J_n^g$, for some $n \geq 1$. Then $a_i = g_n^{-1}(h(a_i)) = c^n[(h(a_i)/c^{n-1} - c\varepsilon)/(c-\varepsilon)]$.
  $L^2(h(a_i), h(a_j)) \Rightarrow h(a_j) = ch(a_i) \Rightarrow h(a_j) \in J_{n+1}^g \Rightarrow a_j = g_{n+1}^{-1}(h(a_j)) = c^n[c(h(a_j)/c^n - c\varepsilon)/(c-\varepsilon)] = c[c^n(h(a_i)/c^{n-1} - c\varepsilon)/(c-\varepsilon)] = ca_i \Rightarrow L^1(a_i, a_j)$.

This completes the proof of $L^1(a_i, a_j) \Leftrightarrow L^2(h(a_i), h(a_j)), \forall a_i \forall a_j \in R$.

**Theorem 7.** *$D_1 \cong D_2$, i.e. these two structures are isomorphic.*

*Proof.* The function $h$ is a bijection and by Lemmata 3, 5 and 6, we know that it defines an isomorphism between the structures $D_1$ and $D_2$.

**Corollary 8.** *$D_1 \equiv D_2$ , i.e. they are elementarily equivalent.*

We have proved that the structures $D_1$ and $D_2$ cannot be distinguished by any formula of the language $\mathcal{L}_1$. We emphasize that we do not mean just a first order formula but also a second order formula, and in general a formula of any order. The importance of this theorem is that it proves that we cannot distinguish between these two structures, using only the set of predicate symbols $\{=, <\} \cup \sigma$, even if we resort to higher order logic. Therefore, we have no alternative but to extend our language with more predicate symbols.

**Theorem 9.** *The Boolean query* one-line *is not expressible in logic of any order in any language $\mathcal{L}$ that has only the set of predicate symbols $\{=, <\} \cup \sigma$.*

*Proof.* Suppose to the contrary that the Boolean query *one-line* is expressible in $\mathcal{L}$. This means that there is a sentence $\phi$ of this language such that given any database $D$, $D$ satisfies $\phi$ iff $D$ is a single line. Then $\phi$ would distinguish $D_1$ from $D_2$, which is impossible as we have shown above.

**Corollary 10.** *The Boolean query* exists-line *is not expressible in the logic of any order in any language $\mathcal{L}$ that has only the set of predicate symbols $\{=,<\} \cup \sigma$.*

Once again the conclusion is that if we want to express this query, we have no alternative but to extend our language with more symbols.

## 4 Monotone Curves

In this section we shall consider a *first order language* $\mathcal{L}_2$ (an extension of $\mathcal{L}_1$) that has one more built-in predicate symbol, the ternary predicate symbol $PLUS$. As we did with the symbols $=$ and $<$, we will fix the following standard interpretation for $PLUS$: $PLUS(x,y,z) \Leftrightarrow x+y=z$
This is of course the familiar addition relation, formulated in terms of predicates. For our convenience, in what follows we shall write $x+y=z$ instead of $PLUS(x,y,z)$, and $L(x+y,z)$ instead of $\exists w(x+y=w \wedge L(w,z))$.

With the symbols we have at our disposal we can define:

- the abbreviation $MINUS(x,y,z) \equiv PLUS(y,z,x)$
  Since the intuition behind the MINUS abbreviation is the subtraction over the reals, in the rest of the paper we shall write $x-y=z$ instead of $MINUS(x,y,z)$ and $L(x-y,z)$ instead of $\exists w(x-y=w \wedge L(w,z))$.
- the set $\{0\}$, where 0 is the familiar real zero, with the following defining formula (that has one free variable $x$) $\mathbf{0}(x) \equiv x+x=x : \{0\} = \{x \mid < R, =, <, +, L^D > \models \mathbf{0}(x)\}$
  In the rest of the paper we shall be using a symbol $\mathbf{0}$, which does not really exist in the language, to denote 0. It should be obvious that we do not have to include this symbol in the language, since any formula containing $\mathbf{0}$ can be transformed (using formula $\mathbf{0}(x)$) into an equivalent formula that does not contain 0.
- Finally, we shall also need the following logical symbol $\exists!$ , which asserts the existence of *exactly one* element from the universe of our structure. This symbol can be defined from the usual logical symbols in the following way : $\exists! x\phi(x) \equiv \exists x \forall y(\phi(x) \wedge (\phi(y) \rightarrow y=x))$.

Consider now the following four sentences of our language:

- $\forall x \exists! y L(x,y)$ $(BQ_1)$
- $\forall y \exists! x L(x,y)$ $(BQ_2)$
- $\forall x_i \forall x_j \forall y_i \forall y_j (L(x_i,y_i) \wedge L(x_j,y_j) \rightarrow L(x_i+x_j, y_i+y_j))$ $(BQ_3)$
- $\forall x \forall y (L(x,y) \rightarrow \forall \varepsilon(\varepsilon > 0 \rightarrow \exists \delta(\delta > 0 \wedge \forall x_i \forall y_i (x-\delta < x_i \wedge x_i < x+\delta \wedge L(x_i,y_i) \rightarrow y-\varepsilon < y_i \wedge y_i < y+\varepsilon))))$ $(BQ_4)$

We define the query $BQL_0$ to be the conjuction: $(BQ_1) \wedge (BQ_2) \wedge (BQ_3) \wedge (BQ_4)$.
Note:

1. In order to see that we do not need the symbol $\mathbf{0}$ in our language, it suffices to consider the following sentence which is logically equivallent to $BQ_4$ but lacks the intuitive meaning that $BQ_4$ conveys:
   $\forall z \forall x \forall y (\mathbf{0}(z) \rightarrow (L(x,y) \rightarrow \forall \varepsilon(\varepsilon > z \rightarrow \exists \delta(\delta > z \wedge \forall x_i \forall y_i (x - \delta < x_i \wedge x_i < x + \delta \wedge L(x_i, y_i) \rightarrow y - \varepsilon < y_i \wedge y_i < y + \varepsilon)))))$
2. If we assume that sentences $BQ_1, BQ_2$ and $BQ_3$ hold, then by accepting sentence $BQ_4$ (and for the class of structures we study) we have at our disposal the following sentence:
   $\forall x \forall y (L(x,y) \leftrightarrow \forall \varepsilon(\varepsilon > 0 \rightarrow \exists \delta(\delta > 0 \wedge \forall x_i \forall y_i (x - \delta < x_i \wedge x_i < x + \delta \wedge L(x_i, y_i) \rightarrow y - \varepsilon < y_i \wedge y_i < y + \varepsilon)))$.

We shall discuss this fact after Lemma 17.
*Remarks*

- Sentence $BQ_1$ can serve as a Boolean query that characterizes total functional relations. A relation $L$ is a *total functional relation* iff it is the graph of a total function $h$:
  $L(x,y) \Leftrightarrow y = h(x)$, where $h : R \rightarrow R$.
- Sentence $BQ_2$ can serve as a Boolean query that characterizes the inverse relation $L^{-1}(x,y)$ (where $L^{-1}(x,y) \Leftrightarrow L(y,x)$) as a total functional relation. Taken together sentences $BQ_1$ and $BQ_2$ characterize relations $L$ that are graphs of functions $h : R \rightarrow R$, 1-1 and onto (bijections).
- Taken together sentences $BQ_1$, $BQ_2$ and $BQ_3$ imply that infinitely many points of $L$ belong in a line passing through the origin. The possibility of having "infinite" number of lines each having an infinite number of "holes" still remains.
- Sentence $BQ_4$ simply asserts the "continuity" of $L$. However, taken together sentences $BQ_1$, $BQ_2$, $BQ_3$ and $BQ_4$ characterize the class of databases consisting of *exactly one line passing through the origin*. It should be clear that with a few minor substitutions $BQL_0$ can also capture the class of databases consisting of exactly one line not passing through the origin and not being parallel to the coordinate axes. The remaining case, i.e. when the database consists of exactly one line parallel to any of the coordinate axes is trivial.

**Definition 11.** Let $\mathbf{K}_R$ be the class of structures for language $\mathcal{L}_2$ that are models of $BQL_0$ and their universe is the set of reals, i.e. $\mathbf{K}_R = \{D =< R, =, <, +, L^D > | D \models BQL_0\}$.
(We impose this restiction because we are only interested in structures that have $R$ as domain.)

**Definition 12.** Let $\mathbf{K}_R^*$ be $\{D =< R, =, <, +, L^D > | L^D(x, cx)\}$, where $c$ is real and $c \neq 0$, i.e. $\mathbf{K}_R^*$ is the class of structures for $L_2$ with domain the set of reals and where the binary predicate symbol $L$ is interpreted as the graph of the line $y = cx$, with $c$ real and $c \neq 0$.

**Lemma 13.** *$L^D(x,y) \Rightarrow L^D(-x,-y)$, for every structure $D \in \mathbf{K}_R$.*

*Proof.* Let $P = (x,y)$ be an arbitrary point of $R^2$ that belongs to $L^D$ and consider $-x$. We know from $BQ_1$ that there is a unique $y'$ such that $(-x,y') \in L^D$. Using $BQ_3$ we get that $(0, y+y') \in L^D$ (1). In a similar way we get that $(x+x', 0) \in L^D$ (2). Due to $BQ_3$, (1) and (2) give us that $(x+x', y+y') \in L^D$ (3). From (1), (2) and (3) because of the imposed uniqueness we take that $x+x' = 0$ and $y+y' = 0$ and finally $x' = -x$ and $y' = -y$.

**Corollary 14.** *$L^D(0,0)$ for every structure $D \in \mathbf{K}_R$.*

**Lemma 15.** *Let $P_1$ be $(x,y)$ and let $P_k$ be $(kx,ky), k \in Z$. $P_1 \in L^D \Rightarrow P_k \in L^D$, for every structure $D \in \mathbf{K}$.*

*Proof.* Because of Lemma 13 and Corollary 14, given that $P_1 \in L^D$, it suffices to prove that $P_n \in L^D$ for $n \in N$. We can prove that easily by induction. For $n = 1$ it is true by our hypothesis. We assume that it is true for every natural $< n$ and we must show that it is true for $n$. But this is trivial; since $P_1 \in L^D$ and $P_{n-1} \in L^D$, we get from sentence $BQ_3$ that $P_n \in L^D$.

**Lemma 16.** *Let $P_1$ be $(x,y)$ and let $P_{1/k}$ be $(x/k, y/k), k \in Z-\{0\}$. $P_1 \in L^D \Rightarrow P_{1/k} \in L^D$, for every structure $D \in \mathbf{K}_R$.*

*Proof.* Because of Lemma 13, given that $P_1 \in L^D$, it suffices to prove that $P_{1/n} \in L^D$ for $n \in N - \{0\}$. We know from $BQ_1$ that there is a unique $y'$ such that $(x/n, y') \in L^D$. By Lemma 15 we deduce that $(x, ny') \in L^D$ (1). Combining (1) with our hypothesis that $P_1 \in L^D$ and because uniqueness is imposed, we conclude that $y = ny'$, or $y' = y/n$, which proves that $P_{1/n} \in L^D$.

**Lemma 17.** *Let $P_1$ be $(x,y)$ and let $P_q$ be $(qx,qy), q \in Q.P_1 \in LD \Rightarrow P_q \in L^D$, for every structure $D \in \mathbf{K}_R$.*

*Proof.* Being rational $q = k/m$ for some $k \in Z$ and $m \in Z - \{0\}$. From Lemma 15 we know that $(kx,ky) \in L^D$ and by applying Lemma 16 to $(kx,ky)$ we get that $(kx/m, ky/m) = (qx,qy) \in L^D$.

*Note:*
In order to derive Lemmata 13, 15, 16, 17 only sentences $BQ_1$, $BQ_2$ and $BQ_3$ have been used. There was no need to use sentence $BQ_4$. Lemma 17 implies that in every structure $D \in \mathbf{K}_R$ the following sentence is true: $\forall x \forall y (\forall \varepsilon (\varepsilon > \mathbf{0} \rightarrow \exists \delta (\delta > \mathbf{0} \wedge \forall x_i \forall y_i (x - \delta < x_i \wedge x_i < x + \delta \wedge L(x_i, y_i) \rightarrow y - \varepsilon < y_i \wedge y_i < y + \varepsilon))) \rightarrow L(x, y))$. Here is a short proof of this fact.

Let us assume that $(x, y)$ is point such that $\forall \varepsilon (\varepsilon > \mathbf{0} \rightarrow \exists \delta (\delta > \mathbf{0} \wedge \forall x_i \forall y_i (x - \delta < x_i \wedge x_i < x + \delta \wedge L^D(x_i, y_i) \rightarrow y - \varepsilon < y_i \wedge y_i < y + \varepsilon)))$ but $\neg L^D(x, y)$. From $BQ_1$ we know that there exists a unique $y'$ such that $(x, y') \in L^D$. To keep things simple, but without loss of generality we assume that $y' - y = \varepsilon > 0$, and we examine the case $x > 0$ and $y > 0$. If we use our hypothesis for $(x, y)$ and for this specific $\varepsilon$, we deduce that $\exists \delta (\delta > 0 \wedge \forall x_i \forall y_i (x - \delta < x_i \wedge x_i < x + \delta \wedge L^D(x_i, y_i) \rightarrow y - \varepsilon < y_i \wedge y_i < y + \varepsilon))$. From Lemma 17, we know that the point $(qx, qy') \in L^D$, where $q \in Q$ and $1 < q < 1 + (\delta / x)$. This means that $x < qx < x + \delta$, which in turn implies that $qy' < y + \varepsilon = y'$, which is absurd, because we have taken $q > 1$. If $y - y' = \varepsilon > 0$ then we take $1 - (\delta / x) < q < 1$ and arrive at the contradiction $qy' > y - \varepsilon$. The remaining cases, i.e. when $x < 0$ or $y < 0$.
If we combine this last result with sentence $(BQ_4)$, we see that every structure $D \in \mathbf{K}_R$ satisfies the following sentence: $\forall x \forall y (L(x, y) \leftrightarrow \forall \varepsilon (\varepsilon > \mathbf{0} \rightarrow \exists \delta (\delta > \mathbf{0} \wedge \forall x_i \forall y_i (x - \delta < x_i \wedge x_i < x + \delta \wedge L(x_i, y_i) \rightarrow y - \varepsilon < y_i \wedge y_i < y + \varepsilon))))$

**Lemma 18.** *$L^D(x_i, y_i)$ and $L^D(x_j, y_j) \Rightarrow L^D(q_i x_i + q_j x_j, q_i y_i + q_j y_j)$, where $q_i, q_j \in Q$, for every structure $D \in \mathbf{K}_R$.*

*Proof.* From Lemma 17 we get that $L^D(q_i x_i, q_i y_i)$ and $L^D(q_j x_j, q_j y_j)$ and by applying $BQ_3$ we obtain $L^D(q_i x_i + q_j x_j, q_i y_i + q_j y_j)$.

**Corollary 19.** *$L^D(x_i, y_i)$ and $L^D(x_j, y_j) \Rightarrow L^D((x_i + x_j)/2, (y_i + y_j)/2)$, for every structure $D \in \mathbf{K}_R$.*

**Theorem 20.** $\mathbf{K}_R = \mathbf{K}_R^*$

*Proof.* i) $\mathbf{K}_R^* \subseteq \mathbf{K}_R$
This is trivial. Consider an arbitrary structure $D \in \mathbf{K}_R^*$. Obviously $D \models BQL_0$, and therefore $D \in \mathbf{K}_R$.
ii) $\mathbf{K}_R \subseteq \mathbf{K}_R^*$
This direction is not trivial and in order to prove it we shall use $BQ_4$, which we have not used up to now. Let $D$ be a structure that belongs to $\mathbf{K}_R$, i.e. $D \models BQL_0$. We must show that $D \in \mathbf{K}_R^*$, i.e. prove that $L^D$ is the graph of a line $y = cx$, with $c$ real and $c \neq 0$.

Let $P_1 = (x_1, y_1)$, where $x_1 \neq 0$ and $y_1 \neq 0$, be a point that belongs to $L^D$. Clearly, $BQL_0$ asserts that such a point exists. Now let $c$ be $y_1/x_1$. We can then define the following set $\Gamma = \{(x, y) | x, y \in R \text{ and } y = cx\}$; $\Gamma$ is the graph of exactly one line that passes through the origin and is not parallel to any of the coordinate axes. We shall prove that $\Gamma = L^D$. Every point $P_q = (qx_1, qy_1), q \in Q$, belongs to $\Gamma$. By Lemma 17, $P_q$ also belongs to $L^D$. Up to this moment we have not proved that every point of the form $P_r = (rx_1, ry_1)$, with $r \in R$, which in effect means every point of $\Gamma$, also belongs to $L^D$.
a) First, we shall show that $\Gamma \subseteq L^D$. Let $P = (x, y)$ be an arbitrary point of $\Gamma$. Then, there is an integer $k$, such that $P$ lies within the line segment of $\Gamma$ defined by $P_k$ and $P_{k+1}$. In order to keep things simple, we assume that $kx_1 < (k+1)x_1$ and $ky_1 < (k+1)y_1$, which implies $c > 0$, which in turn implies that $x \in [kx_1, (k+1)x_1]$ and $y \in [ky_1, (k+1)y_1]$. It should be clear that this assumption is without loss of generality, because the other three cases can be dealt with an analogous manner, with appropriate modifications in the definitions that follow.
We now define two infinite sequences of closed intervals and two infinite sequences of points, as follows:

- $lx_0 = kx_1, ux_0 = (k+1)x_1, ly_0 = ky_1, uy_0 = (k+1)y_1$
- $lx_{i+1} = lx_i, ux_{i+1} = (lx_i + ux_i)/2$, if $x \in [lx_i, (lx_i + ux_i)/2]$
- $lx_{i+1} = (lx_i + ux_i)/2, ux_{i+1} = ux_i$, if $x \in ((lx_i + ux_i)/2, ux_i]$
- $ly_{i+1} = ly_i, uy_{i+1} = (ly_i + uy_i)/2$, if $y \in [ly_i, (ly_i + uy_i)/2]$
- $ly_{i+1} = (ly_i + uy_i)/2, uy_{i+1} = uy_i$, if $y \in ((ly_i + uy_i)/2, uy_i]$
- $X_i = [lx_i, ux_i], Y_i = [ly_i, uy_i]$
- $LP_i = (lx_i, ly_i), UP_i = (ux_i, uy_i)$

By the definitions, we deduce that $lim_n(ux_n - lx_n) = 0$ and $lim_n(uy_n - ly_n) = 0$. This implies that $lim_n lx_n = lim_n ux_n = \cap X_n = \{x\}$ and $lim_n ly_n = lim_n uy_n = \cap Y_n = \{y\}$ (1). From (1) we get that $lim_n LP_n = lim_n UP_n = P = (x, y)$ (2). Therefore, $\forall\delta > 0$ there exist points $LP_n$ and $UP_n$ such that $lx_n \in (x - \delta, x]$ and $ly_n \in (y - \delta, y]$ and $ux_n \in [x, x + \delta)$ and $uy_n \in [y, y + \delta)$ (3).
We now prove prove that $\forall n \in N$ points $LP_n$ and $UP_n$ belong to $\Gamma$ and $L^D$. This can be done by an easy induction. For $n = 0, LP_0 = (kx_1, ky_1) = P_k$ and $UP_0 = ((k + 1)x_1, (k + 1)y_1) = P_{k+1}$ which belong to $\Gamma$ and $L^D$ (by Lemma 15). We assume that $LP_n$ and $UP_n$ belong to $\Gamma$ and $L^D$, and we must show that $LP_{n+1}$ and $UP_{n+1}$ belong to $\Gamma$ and $L^D$. Because $P, LP_n$ and $UP_n$ are points of $\Gamma$, we know that $lx_{i+1} = lx_i$ iff $ly_{i+1} = ly_i$. This means that either $LP_{n+1} = (lx_n, ly_n) = LP_n$ and $UP_{n+1} = ((lx_n + ux_n)/2, (ly_n + uy_n)/2)$ or $LP_{n+1} = ((lx_n + ux_n)/2, (ly_n + uy_n)/2)$ and $UP_{n+1} = (ux_n, uy_n) = UP_n$. In both cases, we see that $LP_{n+1}$ and $UP_{n+1}$ belong to $\Gamma$ and $L^D$ because of the inductive hypothesis and Corollary 19.
Let us assume that $(x, y) \notin L^D$; then from $(BQ_1)$ we know that there exists a unique $y'$ such that $(x, y') \in L^D$. Clearly, this does not lie on the line $\Gamma$. Suppose $y' - y = \varepsilon > 0, (y - y') = \varepsilon > 0)$. If we take this particular $\varepsilon$, then $(BQ_4)$ asserts that $\exists\delta(\delta > 0 \wedge \forall x_i \forall y_i(x - \delta < x_i \wedge x_i < x + \delta \wedge L^D(x_i, y_i) \rightarrow y' - \varepsilon < y_i \wedge y_i < y' + \varepsilon))$ (4). Let us consider one of the above points $LP_n = (lx_n, ly_n)(UP_n = (ux_n, uy_n))$, such that $x - \delta < lx_n < x(x < ux_n < x + \delta)$ and of course $ly_n < y$ $(x < ux_n < x + \delta$ and $y < uy_n)$ - and we have shown above that such points do exist. From (4) we get that $y' - ly_n < \varepsilon = y' - y(uy_n - y' < \varepsilon = y - y')$, or equivalently $ly_n > y(y > uy_n)$ which is absurd. By assuming that $(x, y) \notin L^D$ we have reached a contradiction; thus $P \in L^D$. This proves that $\Gamma \subseteq L^D$.
b) We complete the proof by showing that $L^D \subseteq \Gamma$.
Let $P = (x, y)$ be an arbitrary point of $L^D$. Let $x'$ be such that $x_1 + x' = x$. Let $(x', y')$ be the corresponding point of $\Gamma$, i.e. $y' = cx'$ (4). We have already proven that $\Gamma \subseteq L^D$, which means that $(x', y') \in L^D$ (5). From (4) and (5) by $BQ_3$ we have that $(x_1 + x', y_1 + y') = (x, y_1 + y') \in L^D$ (6). From (6) by the uniqueness property of $L^D$ we deduce that $y = y_1 + y' = cx_1 + cx' = c(x_1 + x') = cx$. This means that $(x, y) \in \Gamma$ and $L^D \subseteq \Gamma$.
Thus, $\mathbf{K}_R \subseteq \mathbf{K}_R^*$ which completes the proof that $\mathbf{K}_R = \mathbf{K}_R^*$

**Theorem 21.** *The Boolean query $BQL_0$ describes the databases consisting of a single line passing through the origin.*

As an example the structure $D_2$ from the previous section (slightly disfigured line) is not a member of the class $\mathbf{K}_R$. It is easy to see that $D_2$ does not satisfy $BQL_0$ and in particular sentence $BQ_3$.

The sentence $BQL_0$ fails to detect lines that are either parallel to the co-ordinate axes or lines that do not pass through the origin. If we want to cover the case of one line parallel to one of the axes the following two sentences are sufficient:

- $\exists! y \forall x L(x, y)$ $(BQ_5)$
- $\exists! x \forall y L(x, y)$ $(BQ_6)$

The proof is straightforward.

The latter case, i.e. of a line intersecting the axes at a point other than the origin, is also easy to deal with. Instead of using $(BQ_3)$, we will use sentence $(BQ_3')$ which is:
$\exists w \forall x_i \forall x_j \forall y_i \forall y_j (L(x_i, y_i) \wedge L(x_j, y_j) \rightarrow L(x_i + x_j, y_i + y_j - w))$

We define $BQL_1$ to be the conjunction $(BQ_1) \wedge (BQ_2) \wedge (BQ_3') \wedge (BQ_4)$. The following theorem is obvious:

**Theorem 22.** *The Boolean query $BQL_1$ characterizes the class of databases constisting of exactly one line, not parallel to either of the coordinate axes.*

Combining all the previous theorems, we can easily see that:

**Theorem 23.** $QL_1 \equiv BQL_1 \vee (BQ_5) \vee (BQ_6)$

## 5 Conclusions

We investigated the problem of characterizing the class of databases consisting of exactly one line. We proved that we cannot distinguish between a database consisting of exactly one line passing through the origin and a curve that is identical to the line except within a small region. This result means that the important query *one-line* is not expressible in any language with order.

We also examined the consequences of extending our language with linear constraints. We proved that this query can be expressed in *FO+linear* by giving a sentence that characterizes the class of databases consisting of exactly one line.

A straightforward extension of our problem is the investigation of queries capturing more general classes of databases, e.g. databases consisting of conse-qutive line segments. Another interesting problem in the same context is the study of databases consisting of more than one, possibly intersecting, lines.

## Acknowledgments

We wish to thank Stavros Cosmadakis, Gabriel Kuper and George Stavrinos for helpful discussions.

## References


[ACGK94] F. Afrati, S. Cosmadakis, S. Grumbach, G. Kuper. Linear vs. Polynomial Constraints in Database Query Languages. *PPCP 1994*.

[BJM93] A. Brodsky, J. Jaffar and M.J. Maher. Toward Practical Constraint Databases. *Proc. 19th International Conference on Very Large Data Bases*, Dublin, Ireland, 1993.

[En72] H. Enderton. A Mathematical Introduction to Logic. *Academic Press Inc.*, 1972.

[GS94] S. Grumbach and J. Su. Finitely representable databases. In *Manuscript*, 1994.

[GS95] S. Grumbach and J. Su. Dense-Order Constraint Databases. *Proc. 14th ACM PODS*, 1995

[JL87] J. Jaffar and J.L. Lassez. Constraint Logic Programming. *Proc. 14th ACM POPL*, 111–119, 1987.

[KKR90] P. Kanellakis, G. Kuper and P. Revesz. Constraint Query Languages. *Proc. 9th ACM PODS*, pp. 299–313, 1990. To appear in JCSS.

[KG94] P. Kanellakis, D. Goldin. Constraint Programming and Database Query Languages. *Technical Report*, Brown University, CS-94-31.

[Kup90] G.M. Kuper. On the expressive power of the relational calculus with arithmetic constraints. In *Proc. Int. Conf. on Database Theory*, pages 202–211, 1990.

[Kup93] G.M. Kuper. Aggregation in constraint databases. In *Proc. First Workshop on Principles and Practice of Constraint Programming*, 1993.

[La90] J.L. Lassez. Querying Constraints. *Proc. 9th ACM PODS*, 1990.

[Ma87] M. Maher. A Logic Semantics for a class of Committed Choice Languages. *Proc. ICLP4*, MIT Press 1987.

[PBG95] J. Paredaens, J. Van den Bussche, D. Van Gucht. First-order queries on finite structures over the reals.*LICS '95*.

[Re90] P.Z. Revesz. A Closed Form for Datalog Queries with Integer Order. *Proc. 3rd International Conference on Database Theory*, 1990. To appear in TCS.