Linear vs. Polynomial Constraints in Database Query Languages

Foto Afrati[‡] Stavros S. Cosmadakis Stéphane Grumbach[§]
NTU Athens *New York University* *I.N.R.I.A.*

Gabriel M. Kuper
ECRC

Abstract

We prove positive and negative results on the expressive power of the relational calculus augmented with linear constraints. We show non-expressibility of some properties expressed by polynomial constraints. We also show expressibility of some queries involving existence of lines, when the query output has a simple geometrical relation to the input. Finally, we compare the expressive power of linear vs. polynomial constraints in the presence of a discrete order.

1 Introduction

An active area of recent research is concerned with integrating constraints into logical formalisms for programming languages [DG,JL87,Ma87,Sa] and database query languages [BJM93,KKR90, Kup90,Kup93,Re90]. Constraints are incorporated in logic programming systems such as CLP, Prolog III and CHIP. The class of *linear* constraints is of particular interest, because of its applicability and the potential for efficient implementation [HJLL90,JL87,La90].

Kanellakis et.al.[KKR90] describe a methodology to combine constraint programming with database query languages. They propose several generalizations of the traditional relational database calculus (first-order logic). One of the more powerful languages described in [KKR90] is the relational calculus augmented with polynomial constraints, *FO+poly*. This language is powerful enough to express many geometric problems, and has NC data complexity; however, the complexity of quantifier elimination (over real closed fields) makes it impractical for most purposes. A natural question therefore is to ask what happens if constraints are restricted to be linear.

[‡]National Technical University of Athens, Computer Science Division, Heroon Politechniou 9, 157 73 Zographou, Athens, Greece; afrati@theseas.ntua.gr.

[§]I.N.R.I.A., Rocquencourt BP 105, 78153 Le Chesnay, France; Stephane.Grumbach@inria.fr. Supported in part by the Esprit Project BRA FIDE 2.

In this paper we study the expressive power of the relational calculus augmented with linear constraints, *FO+linear*. We first give some negative results, showing that there exist properties in *FO+poly* which are not expressible in *FO+linear*. We use the well-known technique of *Ehrenfeucht-Fraissé* games [Eh61,Fr54]. We show that, when constraints are introduced to first-order logic, games can be appropriately adapted to prove non-definability; by contrast, techniques such as the compactness theorem (from first-order logic) or locality and 0/1 laws (from finite model theory) fail with constraints [GS94].

A natural subset of *FO+poly* singled out in [KKR90] is *FO+lines*; it extends *FO+linear* with variables ranging over lines. We show that there exist properties in *FO+lines* which are not expressible in *FO+linear*. We also show that some natural queries in *FO+lines* can be expressed in *FO+linear* when the output of the query has a simple geometrical relation to the input.

Maybe the most basic query expressed by line variables is "*compute the set of lines contained in the database*". We do not know if it is expressible in *FO+linear*. We show, however, that it is *linear*, i.e., if the input is defined by linear constraints, the output is defined by linear constraints as well. Linearity is a desirable property of query languages with linear constraints, because it makes it possible to cascade queries. It can be shown that queries expressed in *FO+linear* are linear. Also, queries expressed in a fragment of *FO+poly* described in [HJLL90, La90] (the *parametric* queries) are linear. It is an interesting open problem to find the most general fragment of *FO+poly* which expresses only linear queries.

We also compare the expressiveness of linear vs. polynomial constraints in a different context, namely in the presence of a discrete order. We show that including addition in first-order logic increases its expressive power. Adding multiplication increases the expressive power further. Neither is the case for Datalog, because of the availability of recursion. Results in a similar perspective are presented in [NS93], where it is shown that no formula of first order logic using linear ordering and the logical relation $y = 2x$ can define the property that the size of a finite model is divisible by 3.

2 Background

Databases are subsets of the k-dimensional Euclidean space $\mathcal{R}^k$ ($\mathcal{R}$ is the real line). *Queries* are functions from databases to databases; *Boolean* queries are functions from databases to $\{true, false\}$.

FO+poly [KKR90] is the set of first-order formulas (with equality) over atomic formulas as follows:

(i) $S(x_1, \ldots, x_k)$, meaning the point $(x_1, \ldots, x_k)$ is in the database S.
(ii) *Polynomial constraints* of the form

$$f(x_1, \ldots, x_k) \quad \theta \quad 0$$

where f is a k-variable polynomial (with real coefficients) and $\theta \in \{>, =\}$.

Note that $\geq, \neq$ are expressed as Boolean combinations of $>, =$. Also, when writing *FO+poly* formulas we will use abbreviations such as $x < e$ (instead of $-x + e > 0$) and $S(x+1, y)$ (instead of $\exists z.\{z = x + 1 \wedge S(z, y)\}$).

FO+linear is the subset of *FO+poly* obtained be restricting constraints to be linear.

Formulas of *FO+poly* with free variables define queries: the output is 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 [KKR90].

A *linear* database 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. To see this, consider the formula obtained by substituting the definition of the input (by linear constraints) into the query formula. Now the quantifiers can be eliminated, as follows: if $\mathcal{C}$ is a set of linear constraints

$$\begin{aligned} x &> f_i \\ x &< f_j \\ x &= f_k \\ x &= f_l \end{aligned}$$

(where x does not occur in the f's), then the formula $\exists x. \bigwedge \mathcal{C}$ is equivalent to the formula $\bigwedge \mathcal{C}'$, where $\mathcal{C}'$ is the set of linear constraints

$$f_i < f_k = f_l < f_j.$$

Formulas of *FO+poly* do not in general define linear queries, as can be seen by standard geometric arguments (consider, for instance, the set of pairs (x, y) satisfying $x^2 + y^2 = 1$). The *parametric* queries [La90,HJLL90] is a class of formulas of *FO+poly* which define linear queries (by the *Subsumption Theorem* and variable elimination [La90]).

FO+lines is an extension of *FO+linear* with variables ranging over points and lines in $\mathcal{R}^k$. Atomic formulas $S(p), S(l)$ mean the point p (resp. the line l) is contained in the database S; $p \in l$ means the point p lies on the line l. It can be seen that *FO+lines* queries can be expressed in *FO+poly*. More generally, one can consider variables of higher dimension. It can be seen that extending *FO+poly* in this way does not increase its expressive power [KKR90].

3 Linear constraints are less expressive than polynomial

3.1 Games and the expressiveness of *FO+poly*

Definition 1 *The n-round* Ehrenfeucht-Fraissé *game is played between two players on two databases $\mathcal{D}, \mathcal{D}' \subseteq \mathcal{R}^k$. At round r player I picks a point $p_r \in \mathcal{R}$ and associates it to either $\mathcal{D}$ or $\mathcal{D}'$; player II responds by picking $q_r \in \mathcal{R}$ and associating it to the other database.*

For each r let t_r, t'_r be the points associated with $\mathcal{D}, \mathcal{D}'$ respectively; $\{t_r, t'_r\} = \{p_r, q_r\}$. Player II wins *the game iff*

(i) $t_i = t_j$ iff $t'_i = t'_j$ and
(ii) $(t_{i_1}, \ldots, t_{i_k}) \in \mathcal{D}$ iff $(t'_{i_1}, \ldots, t'_{i_k}) \in \mathcal{D}'$.

The above condition is extended, given a set $\mathcal{C}$ of constraints over n variables, by the clause *(iii)* $c(t_{i_1}, \ldots, t_{i_n})$ *iff* $c(t'_{i_1}, \ldots, t'_{i_n})$, *for every constraint* c *in* $\mathcal{C}$.

The well-known theory of Ehrenfeucht-Fraissé games [Eh61,Fr54] gives the following results:

Theorem 2 *Let* Q *be a property of databases. For each* n *and each finite set of linear constraints* $\mathcal{C}$ *(over* n *variables), the following are equivalent:*

(a) Q *is not expressible in* FO+linear *with quantifier depth at most* n *and constraints from* $\mathcal{C}$.
(b) There exist databases $\mathcal{D}_{n,\mathcal{C}}, \mathcal{D}'_{n,\mathcal{C}}$ *which differ wrto* Q *such that player II wins the* n*-round Ehrenfeucht-Fraissé game on* $\mathcal{D}_{n,\mathcal{C}}, \mathcal{D}'_{n,\mathcal{C}}$.

Corollary 3 *Let* Q *be a property of databases. The following are equivalent:*

(a) Q *is not expressible in* FO+linear.
(b) For each n *and each finite set of linear constraints* $\mathcal{C}$ *(over* n *variables), there exist databases* $\mathcal{D}_{n,\mathcal{C}}, \mathcal{D}'_{n,\mathcal{C}}$ *which differ wrto* Q *such that player II wins the* n*-round Ehrenfeucht-Fraissé game on* $\mathcal{D}_{n,\mathcal{C}}, \mathcal{D}'_{n,\mathcal{C}}$.

Consider databases consisting of a subset U of the real line. We will use Corollary 3 to show:

Theorem 4 *The set of databases satisfying*

$$\exists x . \exists y . \quad \{U(x) \wedge U(y) \wedge x^2 + y^2 = 1\}$$

is not expressible in FO+linear.

Proof: (Sketch) Given n and $\mathcal{C}$ as in Corollary 3, we will find points $\delta, \delta', \epsilon$ such that

$$\begin{aligned} \delta^2 + \epsilon^2 &= 1 \\ (\delta')^2 + \epsilon^2 &\neq 1 \end{aligned}$$

and player II has a winning strategy for the game played on the databases

$$\begin{aligned} \mathcal{D} &= \{\delta, \epsilon\} \\ \mathcal{D}' &= \{\delta', \epsilon\}. \end{aligned}$$

Let t_r, t'_r be the points associated (at round r) with $\mathcal{D}, \mathcal{D}'$ respectively (Definition 1). For each r, $0 \leq r \leq n$, we define sets of linear constraints $\mathcal{C}_r, \mathcal{C}'_r$ on the points $\{t_1, \ldots, t_r, \delta, \epsilon\}$ and $\{t'_1, \ldots, t'_r, \delta', \epsilon\}$ respectively. A constraint $c(t_1, \ldots, t_r, \delta, \epsilon)$ is in $\mathcal{C}_r$ iff the *corresponding* constraint $c(t'_1, \ldots, t'_r, \delta', \epsilon)$ is in C'_r. We proceed by induction on r:

$$\begin{aligned} r = n: \ \ \mathcal{C}_n \ &= \ \{t_i = t_j, i, j = 1, \ldots, n\} \cup \\ & \quad\ \ \{t_i = \delta : i = 1, \ldots, n\} \cup \\ & \quad\ \ \{t_i = \epsilon : i = 1, \ldots, n\} \cup \\ & \quad\ \ \{c(t_{i_1}, \ldots, t_{i_n}) : \text{ where } c \in \mathcal{C}, 1 \leq i_j \leq n\}. \end{aligned}$$

$$\begin{array}{lcl} 0 \leq r < n: \ \mathcal{C}_r & = & \{t_i = t_j, i, j = 1, \ldots, r\} \cup \\ & & \{t_i = \delta : i = 1, \ldots, r\} \cup \\ & & \{t_i = \epsilon : i = 1, \ldots, r\} \cup \\ & & \{c(t_{i_1}, \ldots, t_{i_n}) : \text{ where } c \in \mathcal{C}, 1 \leq i_j \leq r\} \cup \\ & & \Delta, \end{array}$$

where Δ is the set of constraints obtained by eliminating t_{r+1} from the set $\mathcal{C}_{r+1}$.

We say that $\mathcal{C}_r, \mathcal{C}'_r$ are *equisatisfied* iff a constraint in $\mathcal{C}_r$ is true just in case the corresponding constraint in $\mathcal{C}'_r$ is true.

Claim: If $\mathcal{C}_r, \mathcal{C}'_r$ are equisatisfied, then for any choice of t_{r+1} (resp. t'_{r+1}) there is a choice of t'_{r+1} (resp. t_{r+1}) such that $\mathcal{C}_{r+1}, \mathcal{C}'_{r+1}$ are equisatisfied.

It follows that, if $\mathcal{C}_0, \mathcal{C}'_0$ are equisatisfied, player II can play so that $\mathcal{C}_n, \mathcal{C}'_n$ are equisatisfied. I.e., by the definition of $\mathcal{C}_n, \mathcal{C}'_n$ player II can win the n-round game, since

$$x \in \mathcal{D} \text{ iff } x = \delta \vee x = \epsilon$$

(resp. $x \in \mathcal{D}'$ iff $x = \delta' \vee x = \epsilon$).

We now show how to pick $\delta, \delta', \epsilon$ so that $\mathcal{C}_0, \mathcal{C}'_0$ are equisatisfied. Write the constraints in $\mathcal{C}_0$ in the form $\delta \ \theta \ f_m(\epsilon)$, where $\theta \in \{>, <, =\}$. Pick ϵ so that $f_m(\epsilon)^2 + \epsilon^2 \neq 1$ for every m. Pick δ so that $\delta^2 + \epsilon^2 = 1$. Now $\delta \neq f_m(\epsilon)$ for every m, and by choosing δ' close enough to δ we can make sure that $\mathcal{C}_0, \mathcal{C}'_0$ are equisatisfied. ■

3.2 The expressiveness of *FO+lines*

We consider databases consisting of a binary relation S. The Boolean query *in-line* asks whether S is contained in a line. It is expressible in *FO+poly*, by the formula

$$\exists u.\exists v.\exists w.\forall x.\forall y.\{S(x, y) \rightarrow ux + vy + w = 0\}.$$

In particular, it is expressible in *FO+lines* by the formula

$$\exists l.\forall p.\{S(p) \rightarrow p \in l\}.$$

Theorem 5 *The* in-line *query is not expressible in* FO+linear.

Proof: We show that, if the *in-line* query is expressible in *FO+linear*, then the set of tuples (x, y, z) satisfying $z = xy$ is definable in *FO+linear*; the latter can be shown to be false.

Given x, y, z, let S be a binary relation containing three tuples:

$$S = \{[1, x], [0, 0], [y, z]\}.$$

It is easy to verify that the three points are on a line if and only if $z = xy$. ■

4 Expressibility of some *FO+lines* queries

We consider databases consisting of a binary relation S. The Boolean query *exists-line* asks whether S contains a line; it is expressible in *FO+lines* by the formula

$$\exists l. S(l).$$

The more general *lines* query returns the set of lines contained in S. The output of *lines* is a set of tuples (u, v, w), each specifying the set of points (x, y) satisfying $ux + vy + w = 0$. Both *lines* and *exists-line* are expressible in *FO+poly*, using the formula

$$\forall x. \forall y. \{ux + vy + w = 0 \rightarrow S(x, y)\}.$$

The *line-intersection* query returns all points p which are intersections of (pairs of distinct) lines contained in S; it is expressible in *FO+lines* by the formula

$$\begin{aligned} \exists l_1. \exists l_2. \quad & \{S(l_1) \wedge S(l_2) \wedge l_1 \neq l_2 \\ & \wedge p \in l_1 \wedge p \in l_2\}. \end{aligned}$$

We will show that *exists-line* and *line-intersection* can be expressed in *FO+linear* for databases of certain geometric shapes. We will also show that *lines* is a linear query.

Definition 6 *A* two-slope *database $S(x, y)$ has the form*

$$\begin{array}{rccc} & (x \geq \beta_1 & \wedge & y \leq \alpha_1 x + s_1) \\ \vee & (\beta_2 < x < \beta_1 & \wedge & y \leq \gamma) \\ \vee & (x \leq \beta_2 & \wedge & y \leq \alpha_2 x + s_2) \end{array}$$

(see Figure 1).

A two-slope database contains a line iff $\alpha_1 \geq \alpha_2$.

Theorem 7 *The* exists-line *query is expressible in* FO+linear *for two-slope databases.*

Proof: (Sketch) Let φ be the formula

$$\begin{aligned} \exists z. \exists w. \quad & \{\neg S(x, z) \wedge \neg S(-x, w) \wedge z + w \leq y \\ & \wedge x > b_1 \wedge -x < b_2\}. \end{aligned}$$

Suppose S is a two-slope database with parameters $\beta_1, \beta_2, \alpha_1, \alpha_2, s_1, s_2, \gamma$ as in Definition 6. For $b_1 > \beta_1$, $b_2 < \beta_2$, the formula φ is equivalent to

$$\begin{aligned} y &> (\alpha_1 - \alpha_2)x + s_1 + s_2 \\ x &> b_1, -b_2. \end{aligned}$$

It follows that, for $b_1 > \beta_1$, $b_2 < \beta_2$, the formula $\forall y. \exists x. \varphi$ is true iff $\alpha_1 - \alpha_2 < 0$ (since $b_2 < b_1$ implies $x > 0$).

Now the formula

$$\exists B_1. \exists B_2. \forall b_1. \forall b_2. \{(b_1 > B_1 \wedge b_2 < B_2) \rightarrow (\forall y. \exists x. \varphi)\}$$

is true iff $\alpha_1 - \alpha_2 < 0$, i.e., iff S does not contain a line. ■

Theorem 8 *The* line-intersection *query is expressible in* FO+linear *for databases consisting of at most two lines.*

Proof: (Sketch) Suppose S consists of exactly two lines, neither parallel to the x-axis, intersecting at (a,b) (it is easy to remove these assumptions). The database

$$S'(x,y) \stackrel{\text{def}}{=} S(x,y) \wedge S(x+1,y)$$

consists of two points (x_1,y_1), (x_2,y_2) (see Figure 2). By a simple geometrical argument,

$$\begin{aligned} x_1 + x_2 &= 2a-1 \\ y_1 + y_2 &= 2b. \end{aligned}$$

Therefore, the formula

$$\begin{aligned} \exists x_1.\exists y_1.\exists x_2.\exists y_2.\quad & \{S'(x_1,y_1) \wedge S'(x_2,y_2) \wedge (x_1 \neq x_2 \vee y_1 \neq y_2) \\ & \wedge\ u = \tfrac{x_1+x_2+1}{2} \wedge v = \tfrac{y_1+y_2}{2}\} \end{aligned}$$

is true iff $(u,v) = (a,b)$. ■

Theorem 9 *The* lines *query is linear.*

Proof: (Sketch) Write $S(x,y)$ in conjunctive normal form: $\bigwedge_i \bigvee_j C_{ij}(x,y)$, where C_{ij} is a linear constraint. Write the formula

$$\forall x.\forall y.\{ux + vy + w = 0 \rightarrow S(x,y)\}$$

in the form

$$\forall x.\{\bigwedge_i \bigvee_j C_{ij}(x, -\frac{ux+w}{v})\},$$

equivalently

$$\bigwedge_i \{\neg\exists x.(\bigwedge_j \neg C_{ij}(x, -\frac{ux+w}{v}))\}$$

(it is easy to deal with the case $v = 0$). Now consider eliminating x from the set of linear constraints $\bigwedge_j \neg C_{ij}(x, -\frac{ux+w}{v})$. Eliminating x from

$$\begin{aligned} d_1 x + d_2(-\frac{ux+w}{v}) + d_3 \quad & \theta_1 \quad 0 \\ d'_1 x + d'_2(-\frac{ux+w}{v}) + d'_3 \quad & \theta_2 \quad 0 \end{aligned}$$

gives, after simplification and cancellation of a common factor v, a constraint

$$(d_2 d'_3 - d'_2 d_3)u + (d_3 d'_1 - d'_3 d_1)v + (d_1 d'_2 - d'_1 d_2)w \quad \theta \quad 0$$

which is linear in the free variables of the query, u, v, w. ■

5 Addition, multiplication, and discrete order

In this section we consider first-order logic and Datalog with a discrete (linear) order. We denote by FO (FO($\leq$), FO($\leq$,+), FO($\leq, +, \times$)) first-order logic with equality (and order, and addition, and multiplication). We use corresponding notation for Datalog and the corresponding extensions. The version of Datalog we are considering allows first-order queries on the input predicates.

It is easy to see that Datalog($\leq$,+) = Datalog($\leq$). We first use $\leq$ and negation to define a successor relation *succ*. Addition can then be defined as a ternary predicate, **PLUS**, as follows:

$$\begin{array}{l}\mathbf{PLUS}(0, x, x) \leftarrow, \\ \mathbf{PLUS}(x', y, z') \leftarrow succ(x, x') \wedge succ(z, z') \wedge \mathbf{PLUS}(x, y, z).\end{array}$$

Further, Datalog($\leq$,+,$\times$) = Datalog($\leq$,+). Multiplication can be easily defined as a ternary predicate, **MULT**, using + as follows:

$$\begin{array}{l}\mathbf{MULT}(0, x, 0) \leftarrow, \\ \mathbf{MULT}(x', y, z') \leftarrow succ(x, x') \wedge z' = z + y \wedge \mathbf{MULT}(x, y, z).\end{array}$$

Therefore, in the presence of discrete order, recursion can be used to show that addition and multiplication do not add expressive power to Datalog. We next see that this is not the case in first-order logic. The following query is (i) not expressible in FO($\leq$), but (ii) expressible in FO($\leq$,+).

Example 10 *Consider the schema $\sigma = (R)$, where R is a binary relation. The universe is the set of natural numbers. The query answers true if and only if (i) the cardinality of the projection of R on the first attribute, R_1, is even, and (ii) the second projection of R, R_2, contains the order of x in R_1 (i.e. $R(x, y)$ iff x is the y^{th} element of R_1).*

It is easy to express the query in FO($\leq$,+).

$$\Big(\forall x_1\ x_2\ y_1\ y_2\ \big(\neg\exists x\ ((x_1 < x < x_2) \wedge R_1(x)) \\ \wedge R_1(x_1) \wedge R_1(x_2) \wedge R(x_1, y_1) \wedge R(x_2, y_2)\big) \rightarrow (y_2 = y_1 + 1)\Big)$$

$$\wedge min_{R_2}(1) \wedge \exists n\ (max_{R_2}(n) \wedge \exists m\ (n = m + m)).$$

Here $min_{R_2}(1)$ expresses the fact that the smallest element in the second column of R is 1 and $max_{R_2}(n)$ the fact that the largest element in the second column of R is n. The proof that it cannot be expressed in FO($\leq$) is based on Ehrenfeucht-Fraïssé games.

The query "is the cardinality of the domain a prime number" is expressible in FO($\leq, +, \times$) but not in FO($\leq, +$). We can therefore conclude with the following result.

Theorem 11

$$\begin{array}{ccccccc} FO & \subset & FO(\leq) & \subset & FO(\leq,+) & \subset & FO(\leq,+,\times) \\ Datalog & \subset & Datalog(\leq) & = & Datalog(\leq,+) & = & Datalog(\leq,+,\times) \end{array}$$

Acknowledgments

We wish to thank Serge Abiteboul, Alex Brodsky, Christophe Tollu and Victor Vianu for helpful discussions, and Paris Kanellakis for providing some of the initial motivation.

References

[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.

[DG] J. Darlington and Y-K. Guo. Constraint Functional Programming. Tech. Report, Dept. of Computing, Imperial College, to appear.

[Eh61] A. Ehrenfeucht. An Application of Games to the Completeness Problem for Formalized Theories. *Fund. Math.*, 49:129–141, 1961.

[Fr54] R. Fraissé. Sur quelques classifications des systèmes de relations. *Publications Scientifiques de l'Université d'Algerie, Séries A*, 1:35–182, 1954.

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

[HJLL90] T. Huynh, L. Joskowicz, C. Lassez and J-L. Lassez. Reasoning About Linear Constraints Using Parametric Queries. *Foundations of Software Technology and Theoretical Computer Science.* Lecture Notes in Computer Science, Springer-Verlag vol. 472, 1990.

[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.

[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.

[NS93] D. Niwinski and A. Stolboushkin. y=2x vs. y=3x. In *Proc. IEEE Symp. of Logic in Computer Science*, pages 172–178, Montreal, June 1993.

[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.

[Sa] V. Saraswat. Concurrent Constraint Logic Programming. MIT Press, to appear.