# Constraint Databases

# and

# Quantifier Elimination

Bart Kuijpers
Hasselt University, Belgium
bart.kuijpers@uhasselt.be

## Conclusion

- Constraint databases have a well-developed theory
  - data model
  - many query languages (expressive power)
  - applications (spatial databases, geographic information systems)
- but have not been developed into real systems
  - efficient query evaluation is bottleneck for developing real systems
  - efficient query evaluation requires efficient quantifier-elimination algorithms

# Outline

## Part I: Basic ideas behind constraint databases

- data model
- query languages: FO, FO+TC, FO+While
- flavour of results in the constraint database model

## Part II: The problem of efficient query evaluation

- algorithms for query evaluation
- the importance of data structures
- open problems

## In the beginning there was ... the relational database model

- a relational database is a finite set of tables

- a table has a finite number of tuples

**TaxTable**

| *Income* | *Tax* |
|---|---|
| 0 | 0 |
| ... | ... |
| 5000 | 0 |
| ... | ... |
| 10000 | 1000 |
| ... | ... |
| 2000000 | 697500 |
| ... | ... |

**TaxRecord**

| *Name* | *PhoneNr* | *Income* |
|---|---|---|
| Bart | 2305950 | 5000 |
| Joos | 47715760 | 10000 |
| Bill | 666 | 2000000 |
| ... | ... | ... |

## Logic as a relational query language

The relational calculus (first-order logic) is used to query relational databases [Codd, 1970]:

$\varphi_R(z) \equiv \exists x \exists y \mathbf{TaxRecord}(x, y, z)$ defines a unary relation $R$

**TaxRecord**

| *Name* | *PhoneNr* | *Income* |
|---|---|---|
| Bart | 2305950 | 5000 |
| Joos | 47715760 | 10000 |
| Bill | 666 | 2000000 |
| ... | ... | ... |

$R$

| *Income* |
|---|
| 5000 |
| 10000 |
| 2000000 |
| ... |

## More formally: relational database theory

- Relational database: finite collection of finite relations $R, S, T, ...$ over a universe $\mathbf{U}$ of atomic values.
  E.g., $\mathbf{U} = \mathbf{N} \cup \{A, B, ..., Z, a, b, ..., z\}^*$.

- First-order logic over $(R, S, T, ...)$ is used as a query language.

- But! in SQL we can write:

  select $x + y$
  from $R$
  where $x < y$

- $\psi(z) \equiv \exists x \exists y (R(x, y) \wedge x < y \wedge z = x + y)$

- The universe $\mathbf{U}$ typically has a structure of its own.
  *E.g.*:
  - Numbers with $<$, $+$, $\times$, ...
  - Strings with *length*, *concat*, ...

# Relational database viewed as structure

- $\mathbf{U}$ can be viewed as a structure in mathematical logic: a set with functions, predicates and constants on it. *E.g.*: $\mathbf{U} = (\mathbf{U}; Number, <, +, \times, 0, 1, String, length, concat)$

- We can look at a relational database as an extension of $\mathbf{U}$ with finite relations: $db = (\mathbf{U}; R, S, T, ...)$.

- We can use classical logic-based languages over the extended alphabet of the structure $\mathbf{U}$ to query (FO, Datalog). E.g., $\psi(z) \equiv \exists x \exists y (R(x, y) \wedge x < y \wedge z = x + y)$

- select $x + y$ variables range over $\mathbf{U}$!!
  from $R$
  where $x < y$

## Some problems with this approach: safety

- Safety problem: The FO-queries

  - $\varphi(z) \equiv \exists x \exists y (z = x + y \wedge (S(x) \vee T(y)))$ and

  - $\psi(x) \equiv \exists y (Number(y) \wedge x = y + y)$

  return infinite outputs (on finite inputs).

- Idea!: We can represent these infinite sets by their defining formulas. The string

$$\exists y (Number(y) \wedge x = y + y)$$

  finitely represents the unary relation

$$\{x \in \mathbf{U} \mid \exists y (Number(y) \wedge x = y + y)\}.$$

## Some problems with this approach: closure, compositionality

- Closure: output relations can be used later as input to other queries (compositionality, views).

- Idea! (continued): The output of a query

$$\varphi(z) \equiv \exists x \exists y (R(x,y) \wedge x < y \wedge z = x + y)$$

applied to the finite input $R = \{(1,2),(3,4)\}$ can be obtained by plugging in the defining formula

$$(x = 1 \wedge y = 2) \vee (x = 3 \wedge y = 4)$$

of $R$ in the query-formula $\varphi(z)$. This gives a formula defining a set $S$:

$$\psi_S(z) \equiv \exists x \exists y (((x = 1 \wedge y = 2) \vee (x = 3 \wedge y = 4)) \wedge x < y \wedge z = x + y)$$

- $\psi_S(z)$ can in turn be plugged in in query-formulas that talk about $S$.

## Constraint databases (1st definition)

- A constraint database over $\mathbf{U} = (\mathrm{U}; Number, <, +, ...)$ is a finite collection of FO-formulas over $\mathbf{U}$: $(\varphi_R, \varphi_S, \varphi_T, \ldots)$.

- Each formula defines a (possible infinite) relation over $\mathbf{U}$: $(R, S, T, \ldots)$.

- The constraint database represents the infinite structure $(\mathrm{U}; Number, <, +, \ldots, R, S, T, \ldots)$.

- Relational databases are a trivial case of constraint databases: **TaxRecord**$= \{(n, p, i) \mid (n = \text{Bart} \wedge p = 2305950 \wedge i = 5000) \vee (n = \text{Joos} \wedge p = 47715760 \wedge i = 10000) \vee \cdots\}$

## From the relational to the constraint data model

- Key idea: we allow relations that contain infinitely many tuples, but that are finitely representable

*"Finite relations are generalized to finitely representable relations"*

- Query evaluation: To evaluate a FO-query

$$\forall x \exists y (x = y + 1 \rightarrow S(x))$$

on a database $(\mathbf{U}; Number, <, +, \ldots, S, \ldots)$, where $S, \ldots$ are given by formulas $\varphi_S, \ldots$, we simply plug in these formulas in the query formula and get

$$\forall x \exists y (x = y + 1 \rightarrow \varphi_S(x)).$$

## What do we have?

- We allow
  - not only finite relations over $\mathbf{U}$
  - also definable relations over $\mathbf{U}$

- We have the closure property.

- But what can we do with these defining formulas?

- What would we like to use these defining formulas for?

- Testing membership: Does $(1, 2)$ belong to the set $R$ given by $\varphi_R(u, v) \equiv \exists x \exists y (u = x + y \wedge ((x = 1 \vee x = 2) \vee y = 3)) \vee u = v$?

- Testing emptyness: Is the set $S$ given by $\varphi_S(z) \equiv \exists x \exists y (z = x + y \wedge ((x = 1 \vee x = 2) \vee y = 3))$ empty?

# Testing membership/emptiness of definable relations

- Can we decide the truth of sentences?

- If the first-order theory of $\mathbf{U}$ is decidable, then these properties can be decided!

- Some examples theories:

| Decidable | Undecidable |
|---|---|
| $(\mathbf{Z}, +, 0, 1, <)$ | $(\mathbf{N}, +, \times, 0, 1, <)$ |
| $(\mathbf{R}, +, \times, 0, 1, <)$ | $(\mathbf{Q}, +, \times, 0, 1, <)$ |
| $(\mathbf{R}, +, 0, 1, <)$ | |
| $(\mathbf{Q}, +, 0, 1, <)$ | |
| Boolean Algebra | $(\Sigma^*, (a)_{a \in \Sigma}, concat)$ |

- Usually huge complexity!, ... in the number of quantifiers.

## Quantifier elimination

- Originally developed by logicians to test membership.

- Idea: Try to express every formula over $\mathbf{U}$ equivalently as a Boolean combination of certain base formulas.

- $\mathbf{U}$ has quantifier elimination: base formulas are atomic formulas.

  *E.g.*: $(\mathbf{R}, +, 0, 1, <)$
  $(\mathbf{R}, +, \times, 0, 1, <)$ both have q.e.

- Sometimes: base formulas are atomic formulas + extra formulas

  *E.g.*: $(\mathbf{Z}, +, 0, 1, <)$
  - $(\exists x)(y = x + x + x + x + 2)$
  - add all mod $n$ and you have q.e.

## Constraint databases (2nd definition)

- Assume the structure $\mathbf{U}$ has quantifier elimination.

- So, we can assume that formulas describing a constraint database are quantifier-free (in DNF).

- Query evaluation:

| | |
|---|---|
| query | $\psi(R, S, T, ...)$ |
| database | $\varphi_R, \varphi_S, \varphi_T, \cdots$ |
| plug in | $\psi(\varphi_R, \varphi_S, \varphi_T, ...)$ |
| apply q.e. | $\psi'$ represents output relation |

## Research in constraint databases in the 1990s

- New topics made possible by new possibilities of representing infinite relations (spatial, spatio-temporal databases).

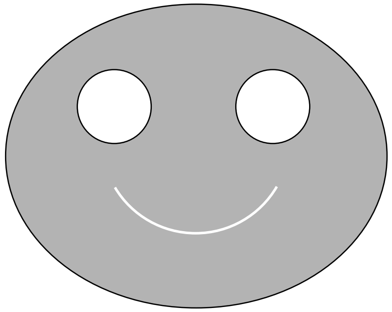
- Classical database theory problems
(about finite databases over $\mathbf{U}$) can be *reconsidered*.
E.g.: are parity, connectivity,... FO-expressible?.

- And the links between the two!

## New topics: spatial databases

- For $\mathbf{U} = (\mathbf{R}, +, \times, 0, 1, <)$, the definable $n$-ary relations are the *semi-algebraic sets* in $\mathbf{R}^n$.

- E.g.: $x^2/25 + y^2/16 \leq 1 \wedge x^2 + 4x + y^2 - 2y \geq -4$
  $\wedge x^2 - 4x + y^2 - 2y \geq -4 \wedge (x^2 + y^2 - 2y \neq 8 \vee y > -1)$.



- Topological and geometrical properties of semi-algebraic sets are well-known [Real Algebraic Geometry]

- Can be extended with classical information.

## FO-queries on spatial databases

- "Is the spatial relation $S$ a straight line?"

$$\exists a \exists b \exists c \big(\neg(a = 0 \wedge b = 0) \wedge ((\forall x)(\forall y)(S(x,y) \leftrightarrow ax + by + c = 0))\big)$$

- "Return the topological interior of $S$."

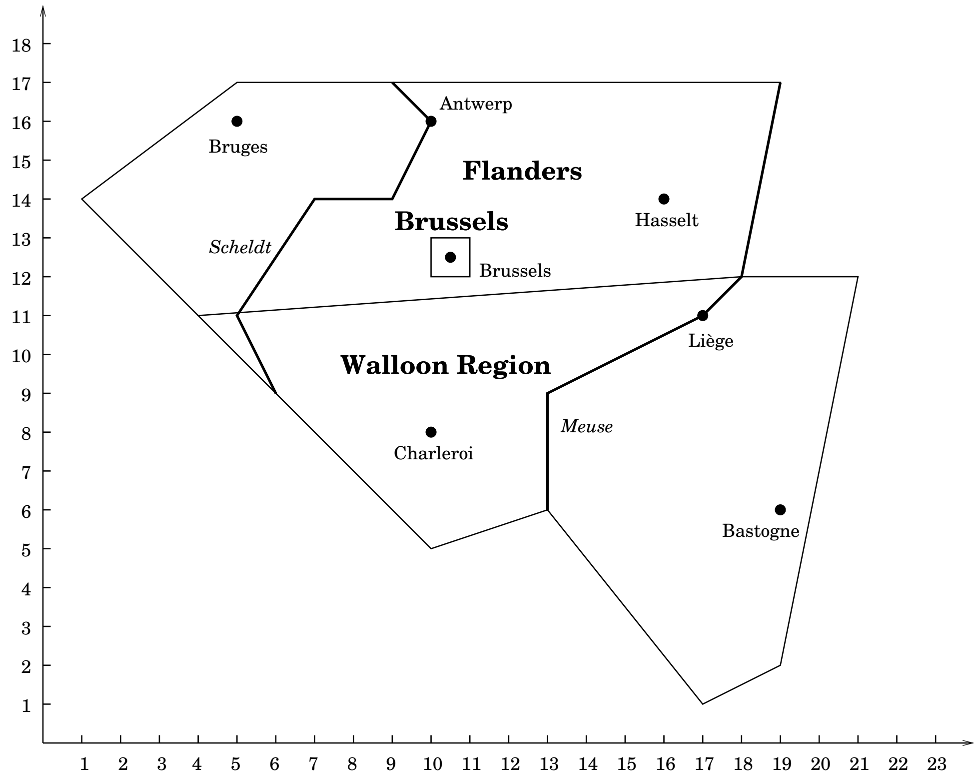
$$\exists \varepsilon (\varepsilon \neq 0 \wedge \forall x' \forall y' ((x - x')^2 + (y - y')^2 < \varepsilon^2 \rightarrow S(x', y'))$$

- "Are the spatial relations $S$ and $T$ overlapping?"

$$\exists x \exists y (S(x,y) \wedge T(x,y))$$

## Linear spatial databases: geographic information systems (GIS)

For $\mathbf{U} = (\mathbf{R}, +, 0, 1, <)$, we get definable $n$-ary relations are the *semi-linear sets* in $\mathbf{R}^n$.




**Regions**

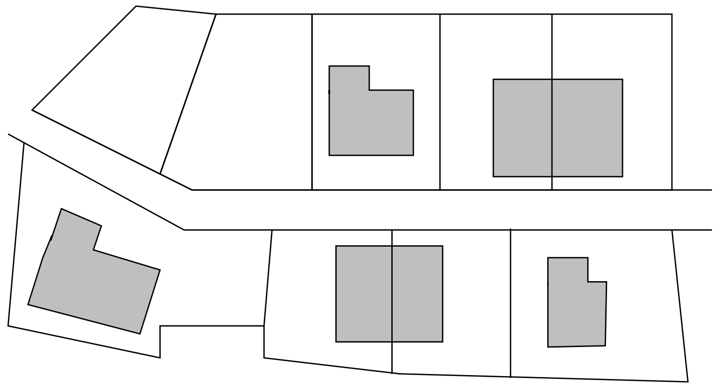
| *Name* | $x$ | $y$ | *Geometry* |
|---|---|---|---|
| Brussels | $x$ | $y$ | $(y \leq 13) \wedge (x \leq 11) \wedge (y \geq 12) \wedge (x \geq 10)$ |
| Flanders | $x$ | $y$ | $(y \leq 17) \wedge (5x - y \leq 78) \wedge (x - 14y \leq -150) \wedge (x + y \geq 45) \wedge$ $(3x - 4y \geq -53) \wedge (\neg((y \leq 13) \wedge (x \leq 11) \wedge (y \geq 12) \wedge (x \geq 10)))$ |
| Walloon Region | $x$ | $y$ | $((x - 14y \geq -150) \wedge (y \leq 12) \wedge (19x + 7y \leq 375) \wedge (x - 2y \leq 15) \wedge$ $(5x + 4y \geq 89) \wedge (x \geq 13)) \vee ((-x + 3y \geq 5) \wedge (x + y \geq 45) \wedge$ $(x - 14y \geq -150) \wedge (x \geq 13))$ |

## FO-queries on linear spatial databases

?• "Is the spatial relation $S$ a straight line?"
This is not expressible in $\mathrm{FO}(\mathbf{R}, +, 0, 1, <, S)$. [E-F-game]

• "Return the topological interior of $S$."
$$\exists \varepsilon (\varepsilon > 0 \wedge \forall x' \forall y' ((| x - x' | < \varepsilon \wedge | y - y' | < \varepsilon) \rightarrow S(x', y'))$$

## More classical approaches to geographic information systems



- Polyhedral subdivisions of $\mathbf{R}^n$.

- Finite number of abstract spatial data types:

  - point, line segment, polyline, polyhedron

  - circle, arc segment, ...

- Elegant, flexible, closed, logical query languages are harder to get here.

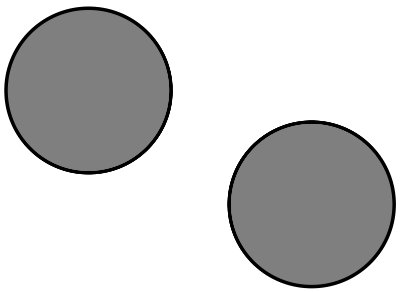
- *But* more efficient implementations of specific operators.

## Classical database theory problems: expressive power of FO on finite databases

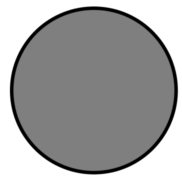
- **Generic collapse**: any formula in FO$(\times, +, 0, 1, <, S, T, ...)$ that is invariant under monotone bijections from $\mathbf{R}$ to $\mathbf{R}$ is equivalently expressible on *finite* db in FO$(<, S, T, ...)$.

- Connectivity of a finite graph (embedded in $\mathbf{R}$) is not FO-expressible.

## Links between finite and spatial: Expressiveness results



and



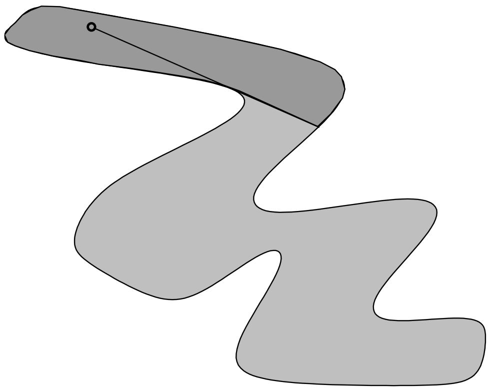
are indistinguishable
by a topological FO-query

**Theorem:**
Topological connectivity of spatial databases is *not* FO-expressible.

⇒ More powerful query languages are needed to express topological connectivity (e.g., FO+While, Datalog, FO+TC).

# Spatial Datalog

*Spatial Datalog* = Datalog + polynomial inequalities in the body of rules (with the underlying domain is $\mathbf{R}$; the only EDB predicate is $S$; relations can be infinite).



$$
\begin{array}{rcl}
\mathit{Obstr}(x,y,x',y') & \longleftarrow & \neg S(\bar{x},\bar{y}), S(x,y), S(x',y'), \\
 & & \bar{x} = a_1 t + b_1, \\
 & & \bar{y} = a_2 t + b_2, 0 \leq t, t \leq 1, \\
 & & b_1 = x, b_2 = y, \\
 & & a_1 + b_1 = x', \\
 & & a_2 + b_2 = y' \\
\mathit{Path}(x,y,x',y') & \longleftarrow & \neg \mathit{Obstr}(x,y,x',y') \\
\mathit{Path}(x,y,x',y') & \longleftarrow & \mathit{Path}(x,y,x'',y''), \\
 & & \mathit{Path}(x'',y'',x',y') \\
\mathit{Disconnected} & \longleftarrow & S(x,y),\ S(x',y'), \\
 & & \neg \mathit{Path}(x,y,x',y') \\
\mathit{Connected} & \longleftarrow & \neg \mathit{Disconnected}.
\end{array}
$$

# Extensions of FO with transitive closure: FO+TC

- FO extended with

$$[\mathsf{TC}_{\vec{x};\vec{y}}\psi(\vec{x},\vec{y})](\vec{s},\vec{t})$$

  with $\vec{x}$, $\vec{y}$ $k$-tuples of real variables.

- Evaluation on input database $A$:

  – $X_0 := \psi(A)$,

  – $X_{i+1} := X_i \cup \{(\vec{x},\vec{y}) \in \mathbf{R}^{2k} \mid (\exists \vec{z})\,(X_i(\vec{x},\vec{z}) \wedge X_0(\vec{z},\vec{y}))\}$,

  – and stop as soon as $X_{i+1} = X_i$.

- **Example**: "Is the linear relation $S$ connected?"
  $(\forall\vec{x})(\forall\vec{y})(S(\vec{x}) \wedge S(\vec{y}) \rightarrow [TC_{\vec{r},\vec{s}}(Seg(\vec{r},\vec{s})](\vec{x},\vec{y}))$ with
  $Seg(\vec{r},\vec{s}) \equiv (\exists\lambda)(0 \leq \lambda \leq 1 \wedge (\forall\vec{t})((\vec{t} = \lambda\cdot\vec{r} + (1-\lambda)\cdot\vec{s}) \rightarrow S(\vec{t}))$

## Extension of FO with While-loop: FO+$While$

$R_1 := \{(x, y) \mid S(x, y)\}$
$R_2 := \{(x, y) \mid (\exists z)(R_1(x, z) \wedge S(z, y))\}$
**while** $R_1 \neq R_2$
**do**
$R_1 := \{(x, y) \mid S(x, y)\}$
$R_2 := \{(x, y) \mid (\exists z)(R_1(x, z) \wedge S(z, y))\}$
**od**

- Programming language with assignment and while-loop

- FO+$While$ is computationally complete

## A short history of constraint databases theory

- 1990-2004: mainstream database research (JACM, SICOMP, JCSS, JSL, ...; PODS, LICS, ICDT, ...);
  practical and mathematical motivations.

- State of the art book (400+ pages): "Constraint databases" (eds. Kuper, Libkin, Paredaens), Springer-Verlag, 2000.


- Textbook: Revesz, "Introduction to Constraint Databases", Springer, 2002.

## Outline

Part I: Basic ideas behind constraint databases

- data model
- query languages: FO, FO+TC, FO+While
- flavour of results in the constraint database model

Part II: The problem of efficient query evaluation

- algorithms for query evaluation
- the importance of data structures
- open problems

# Constraint databases in practice

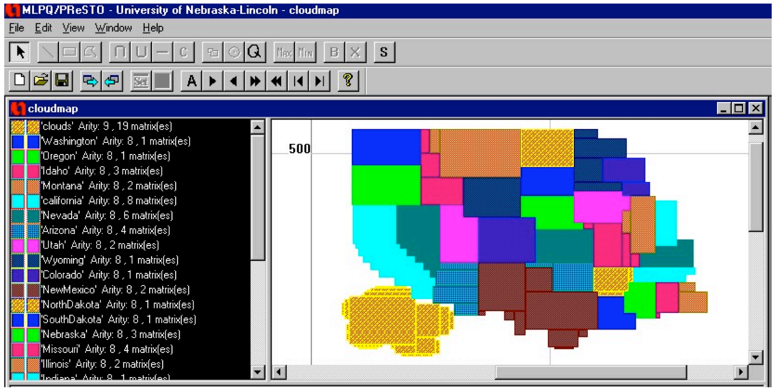
- Fourier-Motzkin quantifier elimination for $(\mathbf{R}, +, 0, 1, <)$:

  $\varphi(x_1, ..., x_{m-1}) \equiv (\exists x_m)\psi(x_1, ..., x_m)$ with $\psi(x_1, ..., x_m)$ a q.f.f. is equivalent to

$$\bigvee_{x_m = t_i \text{ or } x_m = \frac{1}{2}(t_i + t_j) \text{ or } x_m = \pm\infty} \psi(x_1, ..., x_m)$$

- Used in DEDALE [Grumbach et al. at INRIA, Paris]

- Used in PRESTO/MLPQ (didactical) [Revesz in Nebraska],



# A QEPCAD-based constraint database system

Three test queries:

- Topological interior: $\exists \varepsilon(\varepsilon \neq 0 \wedge \forall u \forall v((x-u)^2+(y-v)^2<\varepsilon^2 \rightarrow S(u,v)))$
- Translation: $\exists u \exists v(S(u,v) \wedge x=u+1 \wedge y=v+1)$
- Buffer: $\exists u \exists v(S(u,v) \wedge (x-u)^2+(y-v)^2 \leq 1)$

Three inputs:

- Line segment: $x=0 \wedge -1<y \wedge y<1$
- Square: $-1<x \wedge x<1 \wedge -1<y \wedge y<1$
- Disk: $x^2+y^2 \leq 4$

| | Int ($\exists\forall\forall$) | Trans ($\exists\exists$) | Buffer ($\exists\exists$) |
|---|---|---|---|
| Segm | 330 milliSec | 110 milliSec | 240 milliSec |
| Square | 9,5 Sec | 110 milliSec | 800 milliSec |
| Disk | 9 Min 26 Sec | 190 milliSec | 8 Sec |

## Thoughts about the data structures for constraint databases

- **Q**: Why are we representing constraint database relations by quantifier-free formulas?
  **A**: Membership testing!

- **Q**: Why quantifier-free formulas in DNF? Remark that for a formula $\varphi(x_1, ..., x_n)$ in DNF, $\neg\varphi$ can become of size $O(2^n)$.
  **A**: OK, let's not insist on DNF.

- **Q**: How do you represent a quantifier-free formula?
  **A**: Using dense or sparse representation of polynomials.

# Dense and sparse representations are unsuitable

- Consider the queries given by the formulas
  $\exists x_1 \cdots \exists x_n (R(a_{11}, \ldots, a_{nn}, x_1, \ldots, x_n) \wedge \bigvee_{i=1}^{n} x_i \neq 0)$.

- When applied to

  $$A_n = \{(\alpha_{11}, \ldots, \alpha_{nn}, v_1, \ldots, v_n) \in \mathbf{R}^{n^2+n} \mid \begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1n} \\ \vdots & & \vdots \\ \alpha_{n1} & \cdots & \alpha_{nn} \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \},$$

- we obtain a formula expressing that
  $\det(\alpha_{ij}) = \sum_{\sigma \in \mathcal{S}_n} (-1)^{sgn(\sigma)} \alpha_{1\sigma(1)} \cdots \alpha_{n\sigma(n)} = 0$

- The result is dense (even for moderate degrees)!

$\Rightarrow$ dense or sparse representation of polynomials is unsuitable for query evaluation.

## Alternative data structures: arithmetic boolean circuits

- The problem of exploding representations, suggests changing data structure.

- Expressing that $\det(\alpha_{ij})_{1 \leq i,j \leq n} = 0$ can be done more efficient.

- Using arithmetic boolean circuits (with divisions) of size $O(n^3)$ we can implement Gauss elimination algorithm.

- Idea: complexity theory for geometric elimination requires *simultaneous* optimization of data structures *and* algorithms.

## Arithmetic boolean circuits

- $\exists x_1 \cdots \exists x_n (x_1 = t+1 \wedge R(x_1, x_2) \wedge \cdots \wedge R(x_{n-1}, x_n) \wedge y = x_n^2)$, applied to $A = \{(v_1, v_2) \in \mathbf{R}^2 \mid v_1^2 = v_2\}$, gives formulas $\varphi(t,y) \equiv \exists x_1 \cdots \exists x_n (x_1 = t+1 \wedge x_1^2 = x_2 \wedge \cdots \wedge x_{n-1}^2 = x_n \wedge y = x_n^2)$

- which is is logically equivalent to the q.f.f. $\psi(t,y) \equiv y = \sum_{i=0}^{2^n} \binom{2^n}{i} t^i = (t+1)^{2^n}$.

- **Length**:

| | Dense/Sparse | ABC |
|---|---|---|
| $\varphi(x,t)$ | $O(n)$ | $O(n)$ |
| $\psi(x,t)$ | $O(2^n)$ | $O(n)$ |

- But general-purpose elimination algorithms cannot always guarantee polynomial output descriptions (even using ABCs).

## Upper bounds

- $\exists x_1 \cdots \exists x_n (R(x_1) \wedge \cdots \wedge R(x_n) \wedge y = u_1 x_1 + \cdots + u_n x_n)$, for $n = 1, 2, \ldots$ applied to $A = \{v \in \mathbf{R} \mid v^2 - v = 0\}$,

- gives the formulas $\phi_n(y, u_1, \ldots, u_n) \equiv$ $\exists x_1 \cdots \exists x_n (x_1^2 - x_1 = 0 \wedge \cdots \wedge x_n^2 - x_n = 0 \wedge y = u_1 x_1 + \cdots + u_n x_n)$. Remark: $\phi_n(\beta, \alpha_1, \ldots, \alpha_n)$, $\beta, \alpha_1, \ldots, \alpha_n \in \mathbf{N}$, knapsack problem.

- This elimination problem has the following canonical quantifier-free output formula $\prod_{(\varepsilon_1, \ldots, \varepsilon_n) \in \{0,1\}^n} (y - (\varepsilon_1 u_1 + \cdots + \varepsilon_n u_n)) = 0$.

- In dense or sparse representation this takes $O(2^{n^2})$ space; checking membership requires $O(2^n)$ arithmetic operations.

- Traditional elimination algorithms requite $O(2^{n^2})$ time, whereas ABC based algorithms $O(2^n)$ time.

## Why is elimination exponential?

- Elimination of a block of existential quantifiers is polynomial in the *system degree* (which may be exponential in the size of the input formula).

(1) $\exists x_1 \cdots \exists x_n (x_1 = t + 1 \wedge R(x_1, x_2) \wedge \cdots \wedge R(x_{n-1}, x_n) \wedge y = x_n^2)$,
applied to $A = \{(v_1, v_2) \in \mathbf{R}^2 \mid v_1^2 = v_2\}$, gives formulas
$\exists x_1 \cdots \exists x_n (x_1 = t + 1 \wedge x_1^2 = x_2 \wedge \cdots \wedge x_{n-1}^2 = x_n \wedge y = x_n^2)$
Red part defines one point of $\mathbf{R}^n$: its system degree is 1.

(2) $\exists x_1 \cdots \exists x_n (R(x_1) \wedge \cdots \wedge R(x_n) \wedge y = u_1 x_1 + \cdots + u_n x_n)$,
applied to $A = \{v \in \mathbf{R} \mid v^2 - v = 0\}$, gives
$\exists x_1 \cdots \exists x_n (x_1^2 - x_1 = 0 \wedge \cdots \wedge x_n^2 - x_n = 0 \wedge y = u_1 x_1 + \cdots + u_n x_n)$.
Red part has $2^n$ roots: : its system degree is $2^n$.

- Problem: the system degree may come from the query formula and the database formula.
$\Rightarrow$ Find a complexity invariant in the spirit of the system degree.

## A new data model for constraint databases

- quantifier-free formula (in DNF) in dense/sparse or ABC
  - supports membership test
  - doesn't support visualisation
- Geometric figures
  - implicit representation $y = x^2$ supports membership testing
  - parametric reprentation $x = t,\ y = t^2$ supports visualisation
- Intermediate solution: extend the data model with sample points.

## Sample point

A sample point of a set $A$ is a q.f. formula that defines one point $(a_1, .., a_n) \in A$ such that for any $p \in \mathbf{Z}[x_1, ..., x_n]$ the sign of $p(a_1, .., a_n)$ can be determined by a finite number of arithmetic operations $(+, \times)$ and comparisons $(=, <, ...)$ in $\mathbf{Q}$.

- expressible in FO
- Encoding based on Thom's lemma: a real algebraic number can be given by sign conditions on $p, p', p'', p''', ...$.

## Example of the use of sample points: optimization

- Given a system of linear inequalities $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ $(1 \leq i \leq n)$, determine whether it has a solution, i.e., decide whether the formula

$$\exists x_1 \cdots \exists x_n \bigwedge_{i=1}^{m} \sum_{j=1}^{n} a_{ij}x_j \geq b_i$$

  is true.

- If a system of linear inequalities $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ $(1 \leq i \leq n)$ has a non-empty solution set $V$, decide whether a given affine target function $f$ defined by $(x_1, \ldots, x_n) \mapsto \sum_{i=1}^{n} c_i x_i + d$ reaches a finite maximum on $V$.

- If $f$ reaches such a maximum on $V$, give an example of a point in $V$ that realizes this maximum.

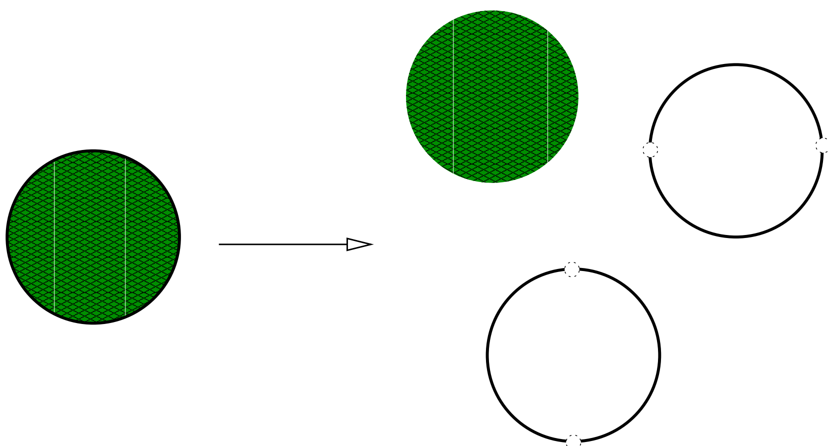
## Constraint databases over $(+, \times, 0, 1, <)$ (3rd definition)

- A set $A \subseteq \mathbf{R}^n$ is given by a cell decomposition $\mathcal{F}_1, ..., \mathcal{F}_n$,

- where each $\mathcal{F}_k$ is given by polynomial conditions $f_1 = 0, ..., f_{s_k} = 0$, $g_1 > 0, ..., g_{t_k} > 0$, $\rho_k \neq 0$ (given by ABC) plus sample points.

- close to stratification, with as few branchings (i.e., divisions) as possible

- Gauss $O(n^3)$ algorithm vs. Berkowitz polynomial-time algorithm



## Lower bound theorems

- It turns out that the constraint database formalism can be used as a meta-language for elimination theory

- exponential lower bounds

## Open problems

- Find a complexity invariant in the spirit of the system degree (fragments of FO).

- How can we deal with approximation in the constraint database model?

- Do languages like FO($Between$), FO($Between, Eqdist, Unitdist$), ... have quantifier elimination?

- ...

- Is a constraint database system feasible in practice?

## Conclusion

- Constraint databases have a well-developed theory
  - data model
  - many query languages (expressive power)
  - applications (spatial databases, geographic information systems)
- but have not been developed into real systems
  - efficient query evaluation is bottleneck for developing real systems
  - efficient query evaluation requires efficient quantifier-elimination algorithms