Fundamentos lógicos de bases de datos
(Logical foundations of databases)

Diego Figueira
Gabriele Puppis

CNRS LaBRI

Day 5: ACQ, $FO^k$
Recap

- NP complexity class (SAT, 3COL, ....)

- Conjunctive Queries (correspondence with SQL and Relational Algebra)

- Homomorphisms and canonical structure

- Evaluation of CQ (NP-completeness)

- Containment, Equivalence, Minimisation of CQ (NP-completeness)

- Extension to functional dependencies (chased canonical structure)
On graphs: $\text{CQ } \phi$ is **acyclic** if $G_{\phi}$ is tree-like
Acyclic CQ’s

On graphs: $\text{CQ } \phi$ is **acyclic** if $G_\phi$ is tree-like
Acyclic CQ’s

On graphs: CQ $\phi$ is **acyclic** if $G_{\phi}$ is tree-like

$$\phi(x,y) = \exists z . E(x,z) \land E(z,t) \land E(y,z)$$
Acyclic CQ’s

On graphs: CQ $\phi$ is acyclic if $G_\phi$ is tree-like

$\phi(x,y) = \exists z . E(x,z) \land E(z,t) \land E(y,z)$

On arbitrary structures: a CQ $\phi$ is acyclic if it has a join tree

$\phi(\bar{y}) = \exists \bar{z} . R_1(\bar{z}_1) \land ... \land R_m(\bar{z}_m)$
Acyclic CQ’s

On graphs: CQ $\phi$ is **acyclic** if $G_\phi$ is tree-like

$$\phi(x,y) = \exists z \cdot E(x,z) \land E(z,t) \land E(y,z)$$

On arbitrary structures: a CQ $\phi$ is **acyclic** if it has a **join tree**

$$\phi(\bar{y}) = \exists \bar{z} \cdot R_1(\bar{z}_1) \land ... \land R_m(\bar{z}_m)$$

A **join tree** is a tree $T$ s.t.:
- nodes are the atoms $R_i(\bar{z}_i)$
- for every variable $x$ of $\phi$ the set of $R_i(\bar{z}_i)$’s with $x \in \bar{z}_i$ forms a subtree of $T$
Acyclic CQ’s

On graphs: CQ $\phi$ is **acyclic** if $G_\phi$ is tree-like

$$\phi(x,y) = \exists z . E(x,z) \land E(z,t) \land E(y,z)$$

On arbitrary structures: a CQ $\phi$ is **acyclic** if it has a **join tree**

$$\phi(\bar{y}) = \exists \bar{z} . R_1(\bar{z}_1) \land \cdots \land R_m(\bar{z}_m)$$

A **join tree** is a tree $T$ s.t.:

- nodes are the atoms $R_i(\bar{z}_i)$
- for every variable $x$ of $\phi$ the set of $R_i(\bar{z}_i)$’s with $x \in \bar{z}_i$ forms a subtree of $T$

If $x$ occurs in two nodes, then it occurs in the path linking the two nodes.
Acyclic CQ’s

On graphs: CQ $\phi$ is **acyclic** if $G_\phi$ is tree-like

$\phi(x,y) = \exists z . E(x,z) \land E(z,t) \land E(y,z)$

On arbitrary structures: a CQ $\phi$ is **acyclic** if it has a **join tree**

$\phi(\bar{y}) = \exists \bar{z} . R_1(\bar{z}_1) \land ... \land R_m(\bar{z}_m)$

A **join tree** is a tree $T$ s.t.:
- nodes are the atoms $R_i(\bar{z}_i)$
- for every variable $x$ of $\phi$ the set of $R_i(\bar{z}_i)$’s with $x \in \bar{z}_i$ forms a subtree of $T$
Acyclic CQ’s

On graphs: CQ \( \phi \) is acyclic if \( G_\phi \) is tree-like

\[
\phi(x,y) = \exists z . E(x,z) \land E(z,t) \land E(y,z)
\]

On arbitrary structures: a CQ \( \phi \) is acyclic if it has a join tree

\[
\phi(\vec{y}) = \exists \vec{z} . R_1(\vec{z}_1) \land \ldots \land R_m(\vec{z}_m)
\]

A join tree is a tree \( T \) s.t.:

- nodes are the atoms \( R_i(\vec{z}_i) \)
- for every variable \( x \) of \( \phi \) the set of \( R_i(\vec{z}_i) \)'s with \( x \in \vec{z}_i \) forms a subtree of \( T \)

A reduced hypergraph \( F = (V,E) \) is \( \alpha \)-acyclic if for each \( U \subseteq V \), if \( F|_U \) is connected and has more than one edge, then it has an articulation set.

Alternatively, if its canonical hyper-graph is \( \alpha \)-acyclic.
Acyclic CQ’s

\[ \phi(x,y) = \exists z . E(x,z) \land E(x,t) \land E(y,z) \]
Acyclic CQ’s

\[ \phi(x,y) = \exists z . E(x,z) \land E(x,t) \land E(y,z) \]

E(x,z)  
E(y,z)  E(x,t) 

join tree
Acyclic CQ’s

\[ \phi(x, y) = \exists z \cdot (E(x, z) \land E(x, t) \land E(y, z)) \]

join tree

\[ \phi = \exists x, y, z, t \cdot (R(x, y, z) \land S(z, t) \land S(x, z) \land T(z) \land T(x)) \]
**Acyclic CQ's**

\[ \phi(x,y) = \exists z . E(x,z) \land E(x,t) \land E(y,z) \]

**Join tree**

\[ \phi = \exists x,y,z,t . R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x) \]

**Join tree**
Acyclic CQ’s

\[ \phi(x,y) = \exists z . E(x,z) \land E(x,t) \land E(y,z) \]

\[ \phi = \exists x,y,z,t . R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x) \]

join tree

not a join tree
Acyclic CQ’s

\[ \phi(x,y) = \exists z . E(x,z) \land E(x,t) \land E(y,z) \]

\[ \phi = \exists x,y,z,t . R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x) \]

**join tree**

**not a join tree**
Acyclic CQ’s

\[ \phi(x,y) = \exists z . \ E(x,z) \land E(x,t) \land E(y,z) \]

join tree

\[ \phi = \exists x,y,z,t . \ R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x) \]

not a join tree

a join tree
The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

[Yannakakis]
Acyclic CQ’s

The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

[Yannakakis]

The semi-join

$R \bowtie_{\{i_1=j_1, \ldots, i_n=j_n\}} S = \{ (x_1, \ldots, x_n) \in R \mid \text{there is } (y_1, \ldots, y_m) \in S \text{ where } x_{i_k} = y_{j_k} \text{ for all } k \}$

Note: $R \bowtie_{\{i_1=j_1, \ldots, i_n=j_n\}} S \subseteq R$
1. Compute the join tree $T$ for $\phi$

2. Populate the nodes of $T$ with tuples from corresponding relations of $D$

3. For every leaf $S(x_1,\ldots,x_n)$ with parent $R(y_1,\ldots,y_m)$ replace tuples of parent with

   $$R \bowtie \{i=j \mid x_i = y_j\} \ S$$

   and delete the leaf $S(x_1,\ldots,x_n)$.

4. Repeat until we are left with one node. If it contains a non-empty relation, then $D$ satisfies $\phi$, otherwise it does not.

---

The **evaluation problem** for acyclic CQ sentences is in $O(|\phi|.|D|)$

[Yannakakis]
Acyclic CQ’s

The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

[Yannakakis]

1. Compute the join tree $T$ for $\phi$

2. Populate the nodes of $T$ with tuples from corresponding relations of $D$

3. For every leaf $S(x_1,\ldots,x_n)$ with parent $R(y_1,\ldots,y_m)$ replace tuples of parent with $R \bowtie \{i=j | x_i = y_j\} S$
   and delete the leaf $S(x_1,\ldots,x_n)$.

4. Repeat until we are left with one node. If it contains a non-empty relation, then $D$ satisfies $\phi$, otherwise it does not.

in linear time
Acyclic CQ’s

The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

[Yannakakis]

1. Compute the join tree $T$ for $\phi$
2. Populate the nodes of $T$ with tuples from corresponding relations of $D$
3. For every leaf $S(x_1,...,x_n)$ with parent $R(y_1,...,y_m)$ replace tuples of parent with $R \bowtie \{i=j \mid x_i = y_j\} \ S$
    and delete the leaf $S(x_1,...,x_n)$.
4. Repeat until we are left with one node. If it contains a non-empty relation, then $D$ satisfies $\phi$, otherwise it does not.
The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

$$\phi = \exists x,y,z,t . R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x)$$

$R = \{(1,4,4),(4,1,4)\}$
$S = \{(4,5),(5,2),(4,4)\}$
$T = \{1,2,3,4\}$
Acyclic CQ’s

The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

$\phi = \exists x,y,z,t . \ R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x)$

$R=\{(1,4,4),(4,1,4)\}$
$S=\{(4,5),(5,2),(4,4)\}$
$T=\{1,2,3,4\}$

$S(z,t) : \{(4,5),(5,2),(4,4)\}$
$S(x,z) : \{(4,5),(5,2),(4,4)\}$

$R(x,y,z) : \{(1,4,4),(4,1,4)\}$
$T(z) : \{1,2,3,4\}$
$T(x) : \{1,2,3,4\}$
The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

$$\phi = \exists x,y,z,t . R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x)$$

$R = \{(1,4,4),(4,1,4)\}$
$S = \{(4,5),(5,2),(4,4)\}$
$T = \{1,2,3,4\}$
Acyclic CQ’s

The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

$$\phi = \exists x,y,z,t . R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x)$$

$R=\{(1,4,4),(4,1,4)\}$
$S=\{(4,5),(5,2),(4,4)\}$
$T=\{1,2,3,4\}$

$S(z,t) :\{(4,5),(5,2),(4,4)\}$

$S(x,z) :\{(4,5),(5,2),(4,4)\}$

$T(z) :\{1,2,3,4\}$

$T(x) :\{1,2,3,4\}$
Acyclic CQ’s

The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

$$\phi = \exists x,y,z,t . \ R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x)$$

$R=\{(1,4,4),(4,1,4)\}$
$S=\{(4,5),(5,2),(4,4)\}$
$T=\{1,2,3,4\}$

$$R(x,y,z) :\{(1,4,4),(4,1,4)\}$$
$$S(z,t) :\{(4,5),(5,2),(4,4)\}$$
$$S(x,z) :\{(4,5),(5,2),(4,4)\}$$
$$T(z) :\{1,2,3,4\}$$
$$T(x) :\{1,2,3,4\}$$
Acyclic CQ’s

The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

$\phi = \exists x,y,z,t . R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x)$

$R=\{(1,4,4),(4,1,4)\}$
$S=\{(4,5),(5,2),(4,4)\}$
$T=\{1,2,3,4\}$

$S(z,t) : \{(4,5),(5,2),(4,4)\}$
$S(x,z) : \{(4,5),(5,2),(4,4)\}$
$R(x,y,z) : \{(1,4,4),(4,1,4)\}$
$T(z) : \{1,2,3,4\}$
$T(x) : \{1,2,3,4\}$
The evaluation problem for acyclic CQ sentences is in $O(|\phi|.|D|)$

$$\phi = \exists x,y,z,t . R(x,y,z) \land S(z,t) \land S(x,z) \land T(z) \land T(x)$$

$R=\{(1,4,4),(4,1,4)\}$  
$S=\{(4,5),(5,2),(4,4)\}$  
$T=\{1,2,3,4\}$  

$S(z,t):\{(4,5),(5,2),(4,4)\}$  
$S(x,z):\{(4,5),(5,2),(4,4)\}$
The evaluation problem for acyclic CQ sentences is in \(O(|\phi|.|D|)\)

\[
\phi = \exists x, y, z, t \ . \ R(x, y, z) \land S(z, t) \land S(x, z) \land T(z) \land T(x)
\]

\(R = \{(1,4,4),(4,1,4)\}\)
\(S = \{(4,5),(5,2),(4,4)\}\)
\(T = \{1,2,3,4\}\)

\(S(z, t) : \{(4,5),(5,2),(4,4)\} \neq \emptyset\)
How to compute a join tree?

**GYO reducts** [Graham, Yu, Ozsoyoglu]

An **ear** of a hypergraph \((V,E)\) is a hyperedge \(e\) in \(E\) such that one of the following conditions holds:

1. There is a **witness** \(e'\) in \(E\), such that \(e' \neq e\) and each vertex from \(e\) is either
   (a) **only** in \(e\) or
   (b) in \(e'\); or
2. \(e\) has no intersection with any other hyperedge.

![Diagram of ears in a hypergraph](image.png)
Ears?
Definition: The GYO \textit{reduct} of a hyper-graph is the result of removing ears until no more ears are left.
Ears!

Definition: The GYO reduct of a hyper-graph is the result of removing ears until no more ears are left.

Theorem: TFAE

- The GYO reduct of a hyper graph $G$ is empty
- A CQ $\phi$ having $G$ as underlying canonical hyper-graph is acyclic
- The hyper graph $G$ is $\alpha$-acyclic
Definition: The GYO reduct of a hyper-graph is the result of removing ears until no more ears are left.

Theorem: TFAE

- The GYO reduct of a hyper graph $G$ is empty
- A CQ $\phi$ having $G$ as underlying canonical hyper-graph is acyclic
- The hyper graph $G$ is $\alpha$-acyclic

We can test acyclicity by computing the GYO reduct!
Ears?
Ears?
Ears?
Ears?
Ears?

Acyclic!
How to compute a join tree?

GYO algorithm [Graham, Yu, Ozsoyoglu]

Given the query $Q = \{ R_1(X_1),...,R_n(X_n) \}$
Consider its canonical structure $G_Q$
  For $R_i(X_i)$ an ear with witness $R_j(Y_j)$
  Put an edge between $R_i(X_i)$ and $R_j(X_j)$, and remove $R_i$ from $Q$.
  Repeat.
How to compute a join tree?

**GYO algorithm** [Graham, Yu, Ozsoyoglu]

Given the query $Q = \{ R_1(X_1), \ldots, R_n(X_n) \}$

Consider its canonical structure $G_Q$

For $R_i(X_i)$ an ear with witness $R_j(Y_j)$

Put an edge between $R_i(X_i)$ and $R_j(X_j)$, and remove $R_i$ from $Q$.

Repeat.

**E.g.**

$R(x,y,z)$, $S(x,y)$, $T(x,x)$, $R(x,x,y)$, $T(y,y)$
Acyclic CQ’s

How to compute a join tree?

**GYO algorithm** [Graham, Yu, Ozsoyoglu]

Given the query $Q = \{ R_1(X_1),...,R_n(X_n) \}$
Consider its canonical structure $G_Q$
  For $R_i(X_i)$ an ear with witness $R_j(Y_j)$
  Put an edge between $R_i(X_i)$ and $R_j(X_j)$, and remove $R_i$ from $Q$.
  Repeat.

**E.g.**

$R(x,y,z), S(x,y), T(x,x), R(x,x,y), T(y,y)$

```
T(x,x)
```

```
R(x,y,z)  R(x,x,y)
```

```
S(x,y)  T(y,y)
```
Acyclic CQ’s

How to compute a join tree?

**GYO algorithm** [Graham, Yu, Ozsoyoglu]

Given the query $Q = \{ R_1(X_1), \ldots, R_n(X_n) \}$

Consider its canonical structure $G_Q$

For $R_i(X_i)$ an ear with witness $R_j(Y_j)$

Put an edge between $R_i(X_i)$ and $R_j(X_j)$, and remove $R_i$ from $Q$.

Repeat.

E.g.

$R(x,y,z), S(x,y), T(x,x), R(x,x,y), T(y,y)$

```
R(x,y,z)  R(x,x,y)
    \--------\--
    |        |
    |        |
    T(x,x)
```

```
S(x,y)  T(y,y)
```

12
Acyclic CQ’s

How to compute a join tree?

**GYO algorithm** [Graham, Yu, Ozsoyoglu]

Given the query $Q = \{ R_1(X_1), \ldots, R_n(X_n) \}$

Consider its canonical structure $G_Q$

For $R_i(X_i)$ an ear with witness $R_j(Y_j)$

Put an edge between $R_i(X_i)$ and $R_j(X_j)$, and remove $R_i$ from $Q$.

Repeat.

---

**E.g.**

$R(x,y,z)$, $S(x,y)$, $T(x,x)$, $R(x,x,y)$, $T(y,y)$

[Diagram showing the join tree construction with edges and nodes labeled accordingly.]
Acyclic CQ’s

How to compute a join tree?

GYO algorithm [Graham, Yu, Ozsoyoglu]

Given the query $Q = \{ R_1(X_1), \ldots, R_n(X_n) \}$
Consider its canonical structure $G_Q$
  For $R_i(X_i)$ an ear with witness $R_j(Y_j)$
  Put an edge between $R_i(X_i)$ and $R_j(X_j)$, and remove $R_i$ from $Q$.
  Repeat.

E.g.
$R(x,y,z), S(x,y), T(x,x), R(x,x,y), T(y,y)$

Diagram:

- $R(x,y,z)$
- $R(x,x,y)$
- $T(x,x)$
- $S(x,y)$
- $T(y,y)$
How to compute a join tree?

**GYO algorithm** [Graham, Yu, Ozsoyoglu]

Given the query $Q = \{ R_1(X_1), \ldots, R_n(X_n) \}$

Consider its canonical structure $G_Q$

For $R_i(X_i)$ an ear with witness $R_j(Y_j)$

Put an edge between $R_i(X_i)$ and $R_j(X_j)$, and remove $R_i$ from $Q$.

Repeat.

**E.g.**

$R(x,y,z)$, $S(x,y)$, $T(x,x)$, $R(x,x,y)$, $T(y,y)$

![Diagram of join tree](attachment:image.png)
Acyclic CQ’s

How to compute a join tree?

GYO algorithm [Graham, Yu, Ozsoyoglu]

Given the query \( Q = \{ R_1(X_1), \ldots, R_n(X_n) \} \)
Consider its canonical structure \( G_Q \)
   For \( R_i(X_i) \) an ear with witness \( R_j(Y_j) \)
   Put an edge between \( R_i(X_i) \) and \( R_j(X_j) \), and remove \( R_i \) from \( Q \).
   Repeat.

E.g.
\( R(x,y,z), S(x,y), T(x,x), R(x,x,y), T(y,y) \)

Remove ears until you’re left with only one!
Acyclic CQ’s

- Evaluation problem for boolean ACQ’s is LOGCFL-complete
- $NL \subseteq LOGCFL \subseteq AC^1 \subseteq NC^2 \subseteq P$

[Gottlob, Leone, Scarcello]

the class of problems logspace-reducible to a context-free language
Beyond acyclic CQ’s

- **(Hyper-)Treewidth** = a measure of the cyclicity of (hyper-)graphs
  
  \[ \text{tw}(Q) = \text{tw}(\text{canonical hypergraph of } Q) \]

  \( \text{tw} = 1 \) are forests,

  \( \text{tw} = 2 \) are graphs without \( K_4 \) as a minor, ...

- For fixed \( k \),
  
  computing whether \( Q \) has \( \text{tw} \leq k \) and 

  calculating a tree decomposition

  can be done in **linear time**
Beyond acyclic CQ’s

- **(Hyper-)Treewidth** = a measure of the cyclicity of (hyper-)graphs
  \[ \text{tw}(Q) = \text{tw( canonical hypergraph of } Q) \]
  \[ \text{tw} = 1 \text{ are forests,} \]
  \[ \text{tw} = 2 \text{ are graphs without } K_4 \text{ as a minor, ...} \]
- For fixed \( k \),
  computing whether \( Q \) has \( \text{tw} \leq k \) and
  calculating a tree decomposition
  can be done in **linear time**

Bounded tree width queries = a class of CQ’s so that for some \( k \),
every query has \( \text{tw} \leq k \)
Beyond acyclic CQ’s

CQ’s with bounded treewidth can be evaluated in PTIME
Beyond acyclic CQ’s

CQ’s with bounded treewidth can be evaluated in PTIME

Containment of CQ’s with bounded treewidth is in PTIME
Beyond acyclic CQ’s

CQ’s with bounded treewidth can be evaluated in PTIME.

Containment of CQ’s with bounded treewidth is in PTIME.
Beyond acyclic CQ’s

CQ’s with bounded treewidth can be evaluated in PTIME

Containment of CQ’s with bounded treewidth is in PTIME

A class $C$ of CQ’s can be evaluated in PTIME iff they have bounded tree width!

For graphs, assuming $W[1] \neq FPT$, and $C$ a r.e. class of graphs. [Grohe, Schwentick, Segoufin]
Querying with semi-joins

The semi-join

\[ R \bowtie_{\{i_1=j_1,\ldots,i_n=j_n\}} S = \{ (x_1,\ldots,x_n) \in R \mid \text{there is } (y_1,\ldots,y_m) \in S \text{ where } x_{i_k} = y_{j_k} \text{ for all } k \} \]

The semi-join algebra (SA): variant of RA with operations:

\( \bowtie, \cup, \pi, \sigma, \setminus, \text{dupcol} \)
Querying with semi-joins

The semi-join

\[ R \Join_{\{i_1=j_1, \ldots, i_n=j_n\}} S = \{ (x_1, \ldots, x_n) \in R \mid \text{there is } (y_1, \ldots, y_m) \in S \text{ where } x_{i_k} = y_{j_k} \text{ for all } k \} \]

The semi-join algebra (SA): variant of RA with operations:

\[ \Join, \cup, \pi, \sigma, \setminus, \text{dupcol} \]

Output at most linear in the database. Further,

**The evaluation problem** for SA is in \( O(|\phi|.|D|) \)

Logical characterisation: “stored-tuples guarded fragment of FO”
Acyclic CQs:

- every intermediate relation is **linear** in $|D|$
- we apply $|\phi|$ semi-joins

What if we allow intermediate relations to be **polynomial** in $|D|$?
Def. \( \text{FO}^k = \text{The fragment of FO restricted to} \  k \ \text{variable names} \)
\( \phi(x) = \text{“Every neighbour of } x \text{ has an outgoing path of length 2”} \)

\[
\forall y. \left( E(x, y) \implies \exists z \exists w \left( E(y, z) \land E(z, w) \right) \right) \in FO^4
\]
Def. \[ \text{FO}^k = \text{The fragment of FO restricted to } k \text{ variable names} \]

\[ \phi(x) = \text{“Every neighbour of } x \text{ has an outgoing path of length 2”} \]

\[ = \forall y. \left( E(x, y) \implies \exists z \exists w \left( E(y, z) \land E(z, w) \right) \right) \in \text{FO}^4 \]

Question: in \text{FO}^2?
\[ \phi(x) = \text{“Every neighbour of } x \text{ has an outgoing path of length 2”} \]

= \forall y. \left( E(x, y) \implies \exists z \exists w \left( E(y, z) \land E(z, w) \right) \right) \in FO^4

= \forall y. \left( E(x, y) \implies \exists x \left( E(y, x) \land \exists y E(x, y) \right) \right) \in FO^2

Def.
\[ FO^k = \text{The fragment of FO restricted to } k \text{ variable names} \]
Bounded variable FO

Def.

\[ \text{FO}^k = \text{The fragment of FO restricted to } k \text{ variable names} \]

For example:

\[ \phi(x) = \text{“Every neighbour of } x \text{ has an outgoing path of length 2”} \]

\[ = \forall y. \left( E(x, y) \Rightarrow \exists z \exists w \left( E(y, z) \land E(z, w) \right) \right) \in \text{FO}^4 \]

\[ = \forall y. \left( E(x, y) \Rightarrow \exists x \left( E(y, x) \land \exists y E(x, y) \right) \right) \in \text{FO}^2 \]

PTIME

G
Bounded variable FO

**Def.**

\[ \text{FO}^k = \text{The fragment of FO restricted to } k \text{ variable names} \]

\[ \phi(x) = \text{“Every neighbour of } x \text{ has an outgoing path of length 2”} \]

\[ = \forall y. \left( E(x, y) \implies \exists z \exists w \left( E(y, z) \land E(z, w) \right) \right) \in \text{FO}^4 \]

\[ = \forall y. \left( E(x, y) \implies \exists x \left( E(y, x) \land \exists y E(x, y) \right) \right) \in \text{FO}^2 \]
\[ \phi(x) = \text{“Every neighbour of } x \text{ has an outgoing path of length 2”} \]

\[ = \forall y. \left( E(x, y) \implies \exists z \exists w \left( E(y, z) \land E(z, w) \right) \right) \in \text{FO}^4 \]

\[ = \forall y. \left( E(x, y) \implies \exists x \left( E(y, x) \land \exists y E(x, y) \right) \right) \in \text{FO}^2 \]
Bounded variable FO

Def. \( FO^k = \) The fragment of FO restricted to \( k \) variable names

\[
\phi(x) = \text{“Every neighbour of } x \text{ has an outgoing path of length 2”}
\]
\[
= \forall y. \left( E(x, y) \Rightarrow \exists z \exists w \left( E(y, z) \land E(z, w) \right) \right) \in FO^4
\]
\[
= \forall y. \left( E(x, y) \Rightarrow \exists x \left( E(y, x) \land \exists y E(x, y) \right) \right) \in FO^2
\]
Bounded variable FO

Def.

$\phi(x) = \text{“Every neighbour of } x \text{ has an outgoing path of length 2”}$

$= \forall y. \left( E(x, y) \Rightarrow \exists z \exists w \left( E(y, z) \land E(z, w) \right) \right) \in \text{FO}^4$

$= \forall y. \left( E(x, y) \Rightarrow \exists x \left( E(y, x) \land \exists y E(x, y) \right) \right) \in \text{FO}^2$

The evaluation problem for $\text{FO}^k$ is in PTIME (combined c.)
The evaluation problem for $\text{FO}^k$ is in PTIME (combined c.)

Algorithm for a $\text{FO}^k$ formula $\psi$ of quantifier rank $r$: 
Bounded variable FO

The evaluation problem for $\text{FO}^k$ is in PTIME (combined c.)

Maximum number of nested quantifiers “$\exists x(... \forall y(... \exists x(... \exists z(...)) ...))$”

Algorithm for a $\text{FO}^k$ formula $\psi$ of quantifier rank $r$: 
Bounded variable FO

The evaluation problem for $\text{FO}^k$ is in PTIME (combined c.)

Maximum number of nested quantifiers “$\exists x(... \forall y(... \exists x(... \exists z(...)))$”

Algorithm for a $\text{FO}^k$ formula $\psi$ of quantifier rank $r$:
Bounded variable FO

The evaluation problem for FO\(^k\) is in PTIME (combined c.)

Maximum number of nested quantifiers “∃x(... ∀y(... ∃x(... ∃z(…))))”

Algorithm for a FO\(^k\) formula \(\psi\) of quantifier rank \(r\):

...
Bounded variable FO

The evaluation problem for FO^k is in PTIME (combined c.)

Maximum number of nested quantifiers “∃x(... ∀y(... ∃x(... ∃z(...))))”

Algorithm for a FO^k formula ψ of quantifier rank r:

1. Evaluate qr=0 subformulas α and output result in relations R_{0,α}

2. Evaluate qr=1 subformulas β based on R_{0,α} and output in R_{1,β}

3. Evaluate qr=2 subformulas γ based on R_{1,β} and output in R_{1,γ}

4. ...

r. ...

...
Bounded variable FO

The evaluation problem for FO$^k$ is in PTIME (combined c.)

Maximum number of nested quantifiers \( \exists x(\ldots \forall y(\ldots \exists x(\ldots \exists z(\ldots )))) \)

Algorithm for a FO$^k$ formula \( \psi \) of quantifier rank \( r \):

1. Evaluate \( qr=0 \) subformulas \( \alpha \) and output result in relations \( R_{0,\alpha} \)

2. Evaluate \( qr=1 \) subformulas \( \beta \) based on \( R_{0,\alpha} \) and output in \( R_{1,\beta} \)

3. Evaluate \( qr=2 \) subformulas \( \gamma \) based on \( R_{1,\beta} \) and output in \( R_{1,\gamma} \)

4. \( \ldots \)

\( \vdots \)

\( r. \ldots \)
Bounded variable FO

The evaluation problem for $\text{FO}^k$ is in PTIME (combined c.)

Maximum number of nested quantifiers “$\exists \ldots \forall \ldots (\exists \ldots (\exists \ldots (\ldots) \ldots))$”

Algorithm for a $\text{FO}^k$ formula $\varphi$ of quantifier rank $r$:

1. Evaluate $qr=0$ subformulas $\alpha$ and output result in relations $R_{0,\alpha}$

2. Evaluate $qr=1$ subformulas $\beta$ based on $R_{0,\alpha}$ and output in $R_{1,\beta}$

3. Evaluate $qr=2$ subformulas $\gamma$ based on $R_{1,\beta}$ and output in $R_{1,\gamma}$

4. ...
Bounded variable FO

Algorithm for a FO^k formula \( \psi \) of quantifier rank \( r \):

1. Evaluate \( qr=0 \) subformulas \( \alpha \) and output result in relations \( R_{0,\alpha} \)
   \[ \implies |V|^k \cdot (|\alpha| \cdot |G|)^p \]
2. Evaluate \( qr=1 \) subformulas \( \beta \) based on \( R_{0,\alpha} \) and output in \( R_{1,\beta} \)
   \[ \implies |V|^k \cdot (|\beta| \cdot (|G| + |R_1|))^p \leq |V|^k \]
3. Evaluate \( qr=2 \) subformulas \( \gamma \) based on \( R_{1,\beta} \) and output in \( R_{1,\gamma} \)
   \[ \implies |V|^k \cdot (|\gamma| \cdot (|G| + |R_2|))^p \leq |V|^k \]
4. \ldots
   
   \vdots
   
   \( r \). \ldots
Bounded variable FO

Desirable:

- Given \( k \) and a FO query \( \phi \), is \( \phi \) in \( \text{FO}^k \)? 💀 Undecidable (even w.o. \( \lnot \)
Bounded variable FO

Desirable:

- Given $k$ and a FO query $\phi$, is $\phi$ in $\text{FO}^k$? 👻 Undecidable (even w.o. $\neg$)

- Given $k$ and a CQ query $\phi$, is $\phi$ in $\text{FO}^k$? 🧊 NP-complete
Bounded variable FO

Desirable:

- Given $k$ and a FO query $\phi$, is $\phi$ in $\text{FO}^k$? \(\rightarrow\) Undecidable (even w/o $\neg$)

- Given $k$ and a CQ query $\phi$, is $\phi$ in $\text{FO}^k$? \(\rightarrow\) NP-complete

- Satisfiability for $\text{FO}^k$ \(\rightarrow\) Undecidable if $k \geq 3$ (Domino)
  \(\rightarrow\) NEXPTIME-complete if $k = 2$
Some more cool stuff...

Descriptive complexity

What properties can be checked efficiently? E.g. 3COL can be tested in NP

**Metatheorem**

“A property can be expressed in [insert some logic here] iff it can be checked in [some complexity class here]”
Some more cool stuff...

Descriptive complexity

What properties can be checked efficiently? E.g. 3COL can be tested in NP

Metatheorem

“A property can be expressed in [insert some logic here] iff it can be checked in [some complexity class here]”

⇒ “A property is FO-definable iff it can be tested in AC₀”

⇒ “A property is ∃SO-definable iff it can be tested in NP” [Fagin 73]

⇒ Open problem: which logic captures PTIME?
Recursion

Datalog (semantics based on least fixpoint)

Ancestor(X,Y) :- Parent(X,Z), Ancestor(Z,Y)
Ancestor(X,X) :- .
?- Ancestor(“Louis XIV”,Y)
Recursion

Can we enhance query languages with recursion? E.g. express reachability properties

Datalog (semantics based on least fixpoint)

\[
\text{Ancestor}(X,Y) : \text{Parent}(X,Z), \text{Ancestor}(Z,Y) \\
\text{Ancestor}(X,X) : \text{.} \\
?\text{- Ancestor(“Louis XIV”,Y)}
\]
Recursion

Can we enhance query languages with recursion?  
E.g. express reachability properties

Datalog  

\begin{align*}
\text{Ancestor}(X,Y) & \leftarrow \text{Parent}(X,Z), \text{Ancestor}(Z,Y) \\
\text{Ancestor}(X,X) & \leftarrow \\
?- \text{Ancestor}\left(\text{"Louis XIV"}, Y\right) 
\end{align*}

\(\Rightarrow\) Incomparable with FO (has recursion, but is monotone)

\(\Rightarrow\) Evaluation is in PTIME (for data complexity, but also for bounded arity)
Some more cool stuff...

Semi-structured data

Tree-structured or graph-structures dbs in place of relational dbs.

XML, XPath, Stream processing, ...

```xml
<catalog>
  <book id="1">
    <title>XML Developer's Guide</title>
    <author>Matthew Gambardella</author>
    <year>2000</year>
  </book>
  <book id="2">
    <title>Beginning XML</title>
    <author>David Hunter</author>
    <author>David Gibbons</author>
    <year>2007</year>
  </book>
  ...
</catalog>
```
Some more cool stuff...

Semi-structured data

Tree-structured or graph-structures dbs in place of relational dbs.

XML, XPath, Stream processing, ...

 Evaluation of XPath is in linear time (data complexity)  
 Satisfiability for $\text{FO}^2[\downarrow, \sim]$ is decidable

[Bojanczyk, Parys 08]  
[Bojanczyk, Muscholl, Schwentick, Segoufin 09]
Certain Query Answers (CQA)

\[ \phi[V] = \bigcap_{D \in [V]} \phi(D) \]
Some more cool stuff...

Incomplete information

How to correctly treat NULL values, missing tuples, noisy data?

Certain Query Answers (CQA)

\[ \phi[V] = \bigcap_{D \in [V]} \phi(D) \]
Some more cool stuff...

Incomplete information

How to correctly treat NULL values, missing tuples, noisy data?

Certain Query Answers (CQA)

\[ \phi[V] = \bigcap_{D \in [V]} \phi(D) \]

\[\Rightarrow\] CQA computable in PTIME w.r.t. view size.  
[Abiteboul, Kanellakis, Grahne 91]
Recap

- Relational Algebra = simple SQL = FO on active domain
- Evaluation, Satisfiability, Equivalence, Containment
- Data / Combined complexity
- Expressiveness of FO: EF games, 0-1 law, Rado structure, Locality
- Conjunctive Queries: Homomorphism lemma, Canonical structures
- Acyclic CQ
- FO^k
Bibliography

  (freely available at http://webdam.inria.fr/Alice/)
