Day 5: ACQ, FO^k

ECI 2015
Buenos Aires

Fundamentos lógicos de bases de datos

(Logical foundations of databases)

Diego Figueira **Gabriele Puppis**

CNRS LaBRI

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)

Acyclic CQ's

On graphs: CQ ϕ is **acyclic** if G_ϕ is tree-like

Acyclic CQ's

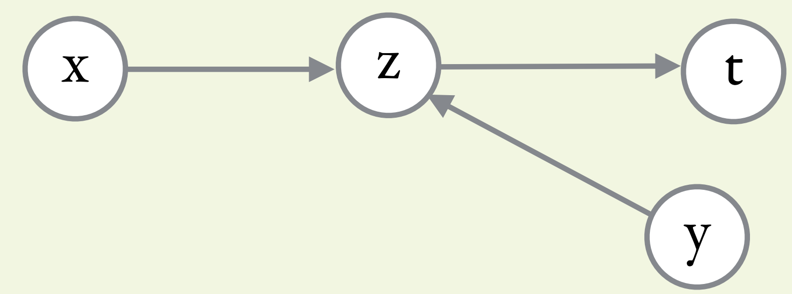
On graphs: CQ ϕ is **acyclic** if G_ϕ is tree-like

underlying undirected graph is acyclic

Acyclic CQ's

On graphs: CQ ϕ is acyclic if G_ϕ is tree-like

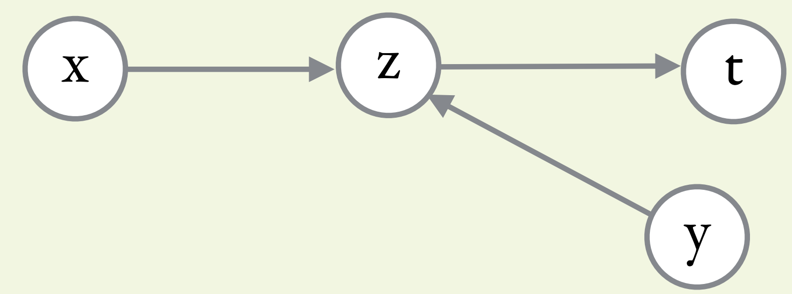
$\phi(x,y) = \exists z \,.\, E(x,z) \wedge E(z,t) \wedge E(y,z)$



Acyclic CQ's

On graphs: CQ ϕ is acyclic if G_ϕ is tree-like

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



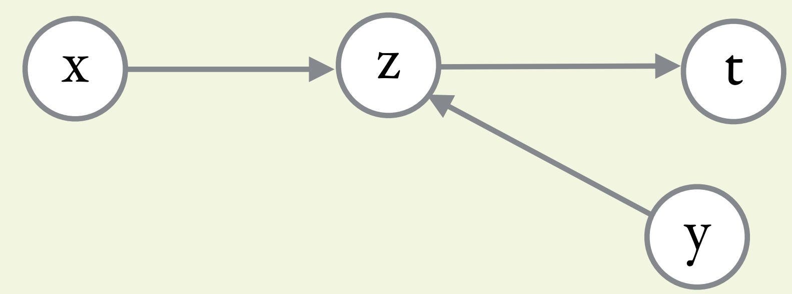
On arbitrary structures: a CQ ϕ is acyclic if it has a join tree

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

Acyclic CQ's

On graphs: CQ ϕ is acyclic if G_ϕ is tree-like

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



On arbitrary structures: a CQ ϕ is acyclic if it has a join tree

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

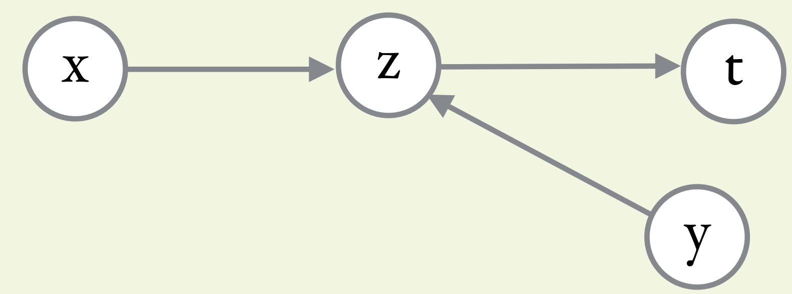
A join tree is a tree T s.t.:

- nodes are the atoms $R_i(\bar{z}_i)$
- for every variable x of ϕ 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 ϕ is acyclic if G_ϕ is tree-like

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



On arbitrary structures: a CQ ϕ is acyclic if it has a join tree

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

If x occurs in two nodes, then it occurs in the path linking the two nodes.

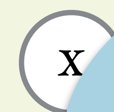
A join tree is a tree T s.t.:

- nodes are the atoms $R_i(\bar{z}_i)$
- for every variable x of ϕ 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 ϕ is **acyclic** if G_ϕ is tree-like

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



Alternatively, if its canonical hyper-graph is **α-acyclic.**

On arbitrary structures: a CQ ϕ is **acyclic** if it has a **join tree**

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

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

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

Acyclic CQ's

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

On graphs: C

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

x

Alternatively, if its canonical hyper-graph is **α-acyclic.**

On arbitrary structures: a CQ ϕ is acyclic if it has a join tree

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

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

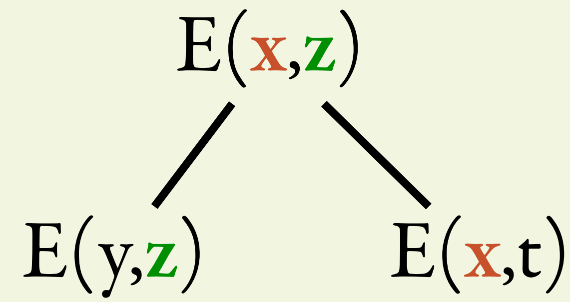
- nodes are the atoms $R_i(\bar{z}_i)$
- for every variable x of ϕ the set of $R_i(\bar{z}_i)$'s with $x \in \bar{z}_i$ forms a subtree of T

Acyclic CQ's

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

Acyclic CQ's

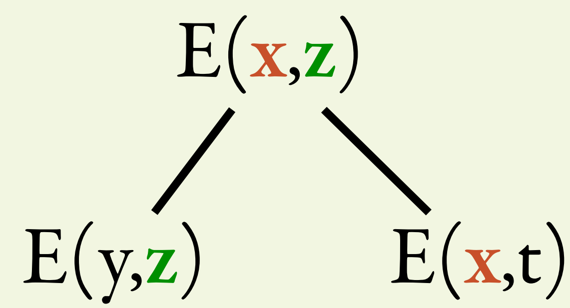
$\phi(x,y) = \exists z \,.\, E(x,z) \wedge E(x,t) \wedge E(y,z)$



join tree

Acyclic CQ's

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

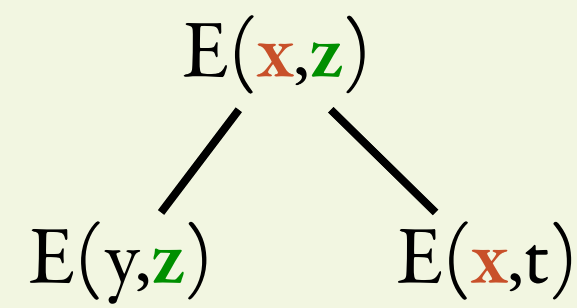


join tree

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

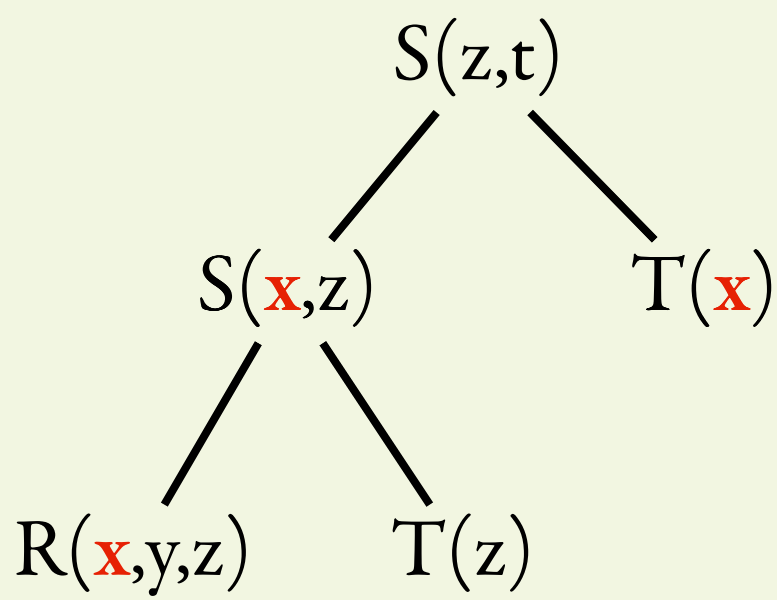
Acyclic CQ's

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



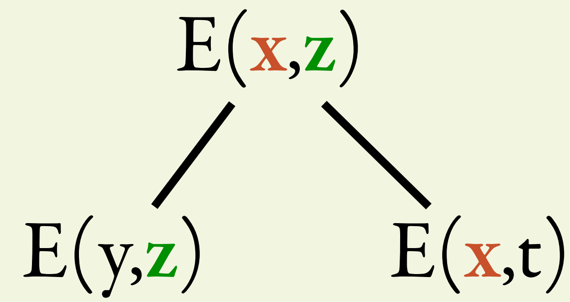
join tree

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



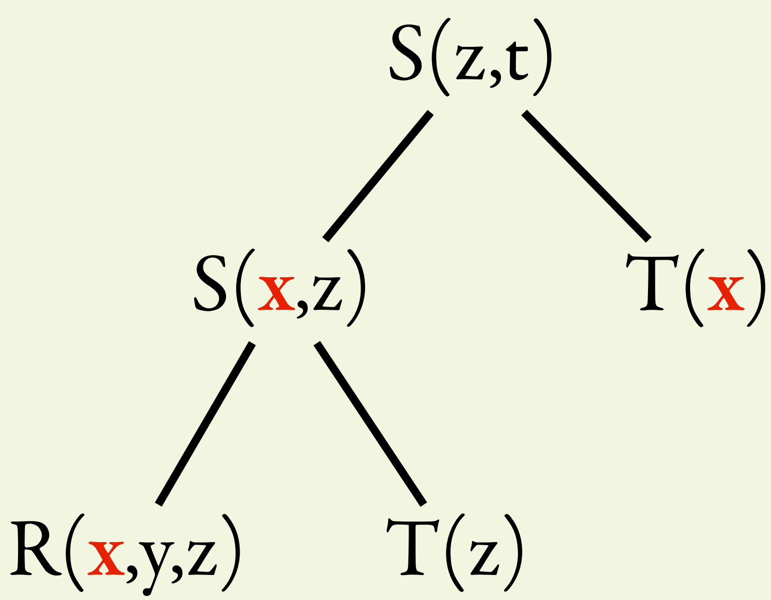
Acyclic CQ's

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



join tree

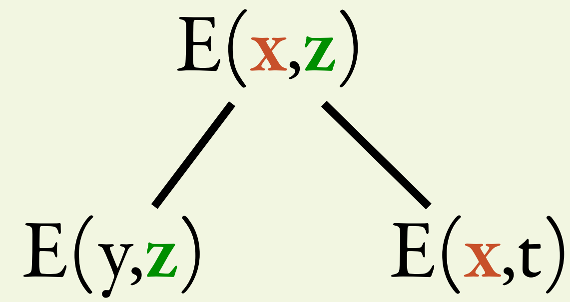
$\phi = \exists x,y,z,t \,.\, R(x,y,z) \wedge S(z,t) \wedge S(x,z) \wedge T(z) \wedge T(x)$



not a join tree

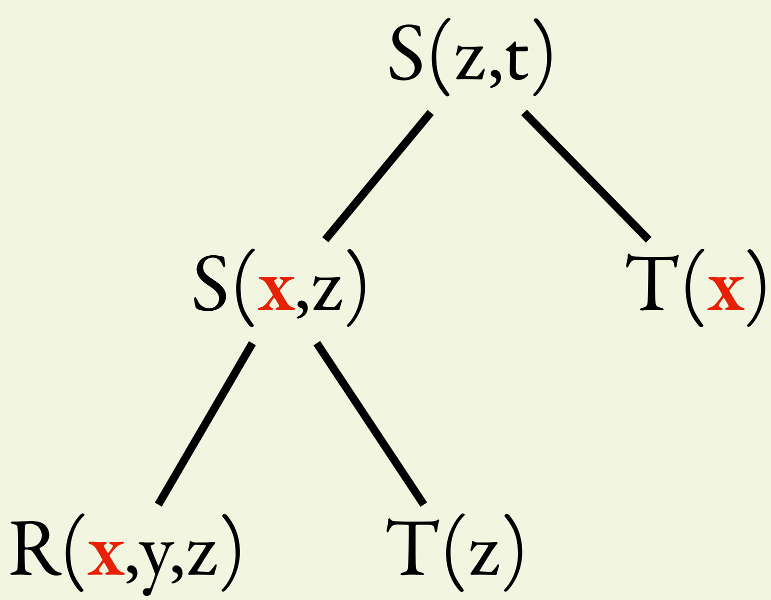
Acyclic CQ's

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

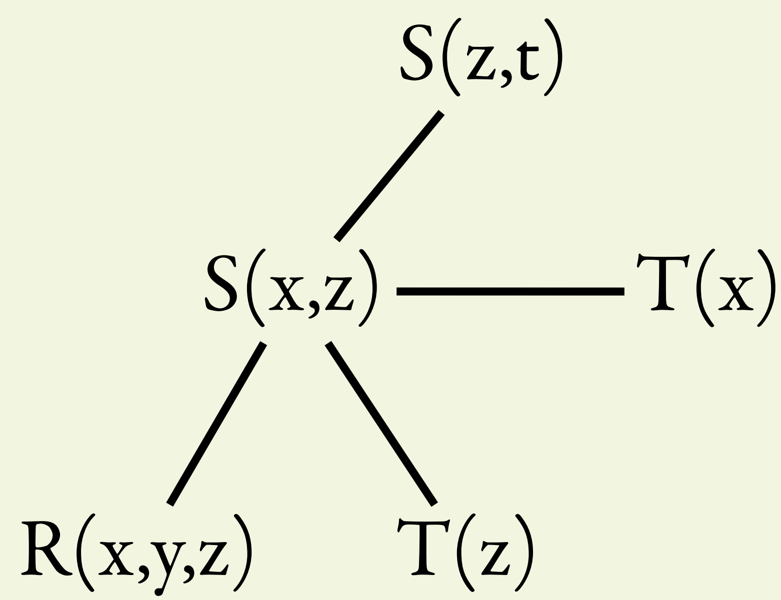


join tree

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

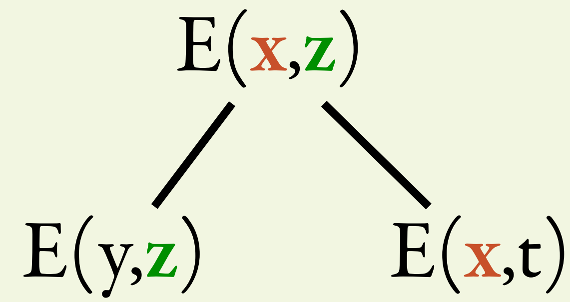


not a join tree



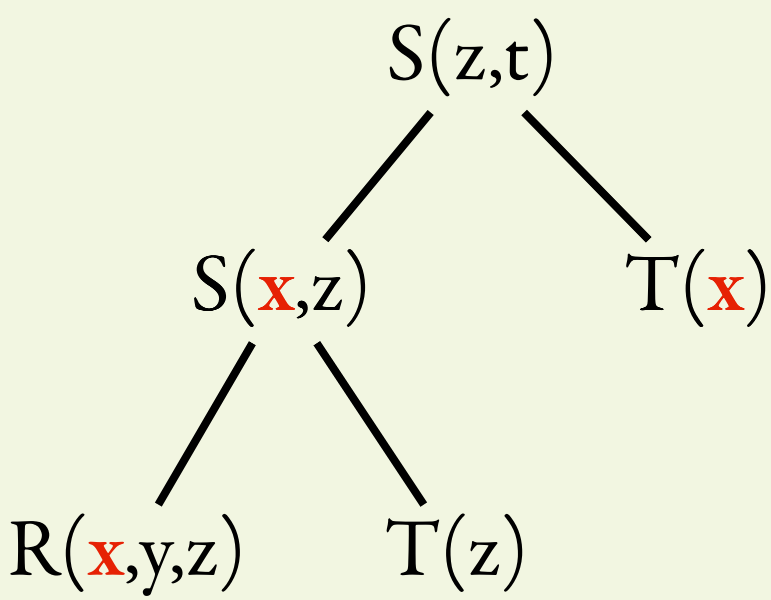
Acyclic CQ's

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

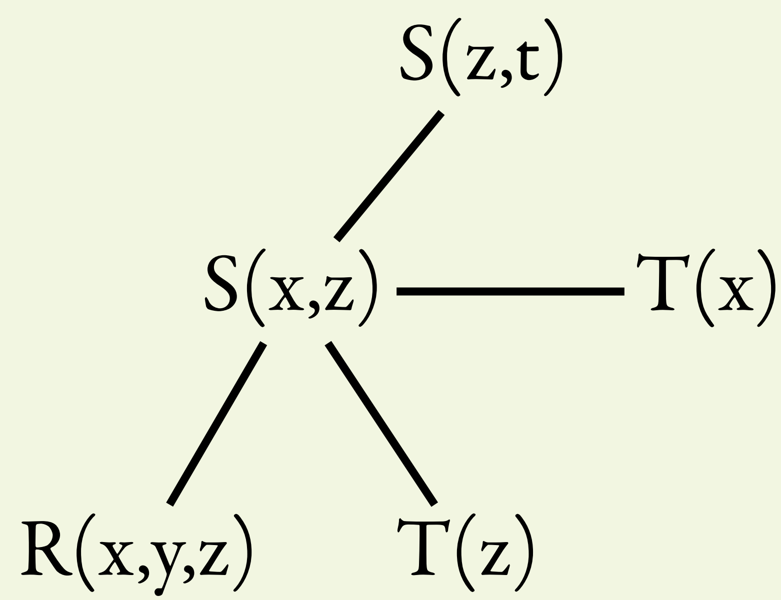


join tree

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

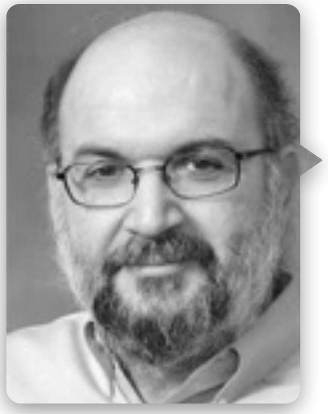


not a join tree



a join tree

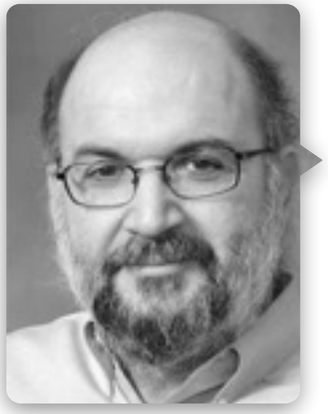
Acyclic CQ's



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 $O(|\phi|.|D|)$

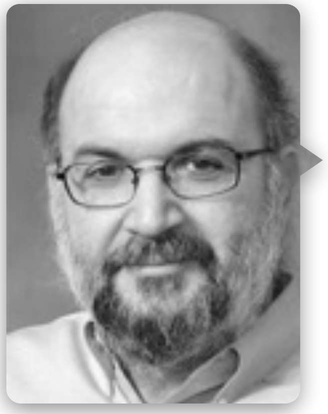
[Yannakakis]

The **semi-join**

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

Note: $R \ltimes_{\{i_1=j_1,\ldots,i_n=j_n\}} S \subseteq R$

Acyclic CQ's

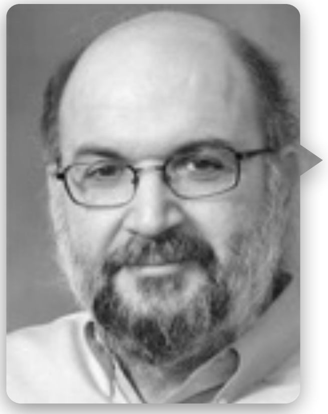


The **evaluation problem** for acyclic CQ sentences is in O(|ϕ|.|D|)

[Yannakakis]

1. Compute the join tree T for ϕ
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
   $$R \ltimes_{\{i=j \mid x_i = y_j\}} S$$
   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 ϕ, otherwise it does not.

Acyclic CQ's



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

[Yannakakis]

in linear time

1. Compute the join tree T for ϕ

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

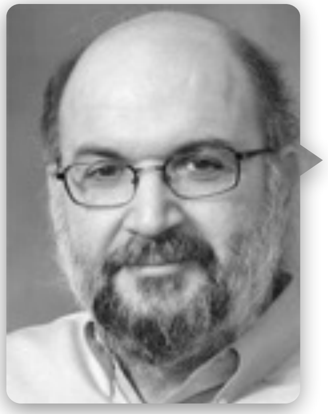
3. For every leaf $S(x_1,\ldots,x_n)$ with parent $R(y_1,\ldots,y_m)$ replace tuples of parent with

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

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 ϕ, otherwise it does not.

Acyclic CQ's



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

[Yannakakis]

remove all the tuples from the parent that do not match a tuple from the child

1. Compute the join tree T for ϕ
2. Populate the nodes of T with tuples from correspon
3. For every leaf $S(x_1,\ldots,x_n)$ with parent $R(y_1,\ldots,y_m)$ replace tuples of parent with

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

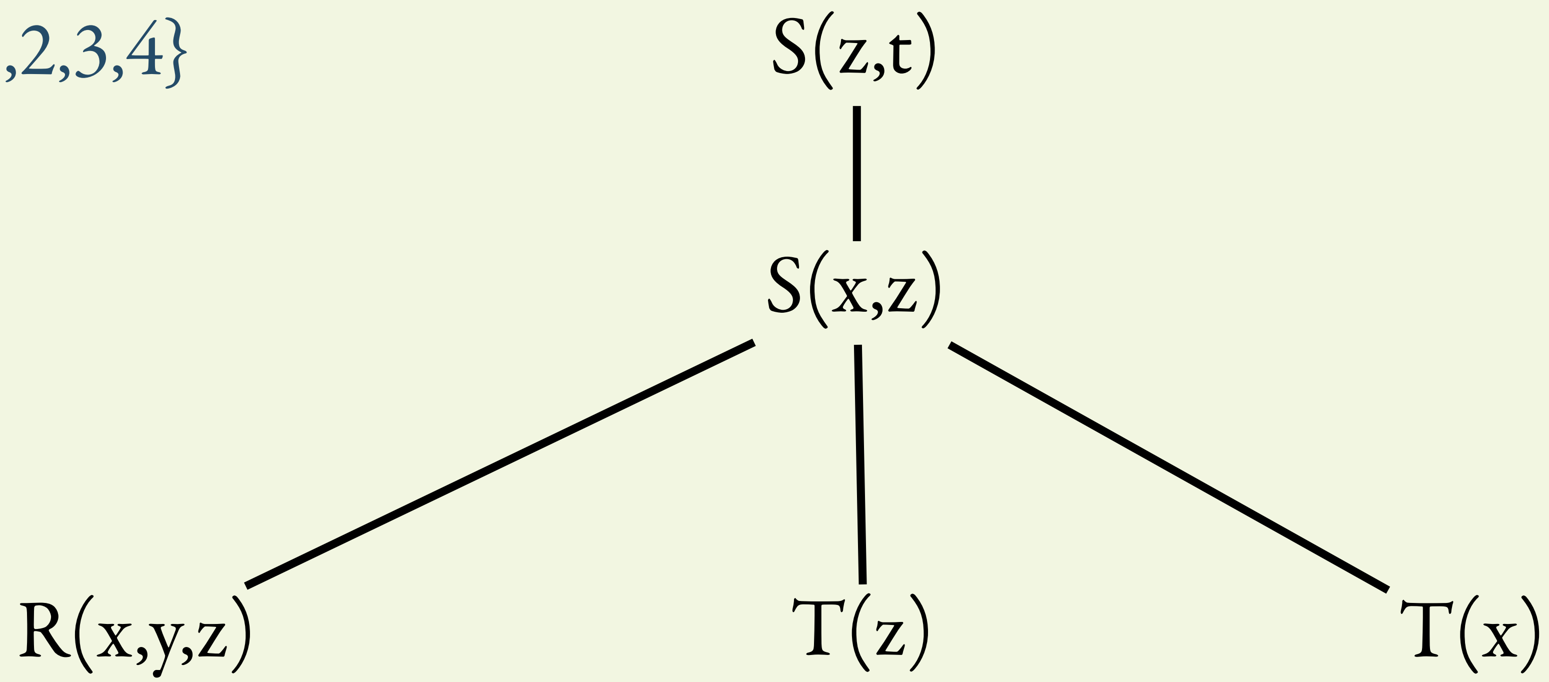
   and delete the leaf $S(x_1,\ldots,x_n)$.
4. Repeat until we are left with one node. If it contains a non-empty relation, then D satisfies ϕ, otherwise it does not.

Acyclic CQ's

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

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

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

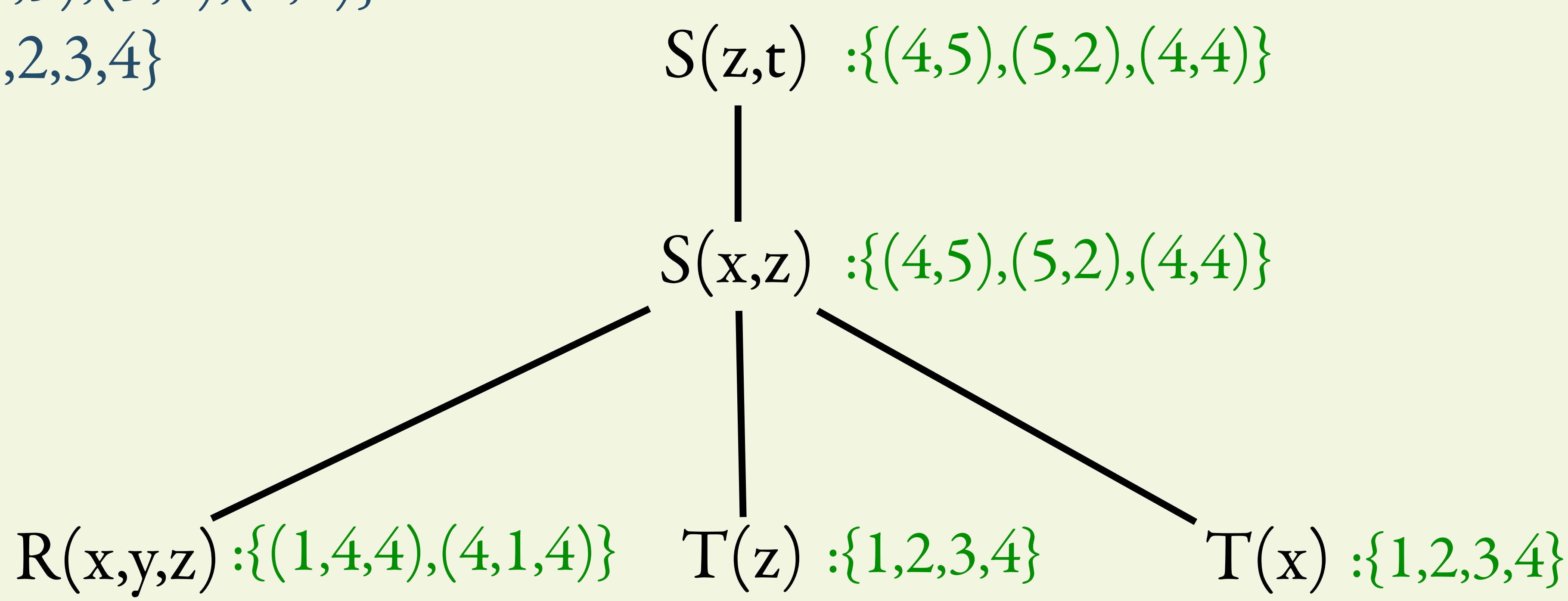


Acyclic CQ's

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

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

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

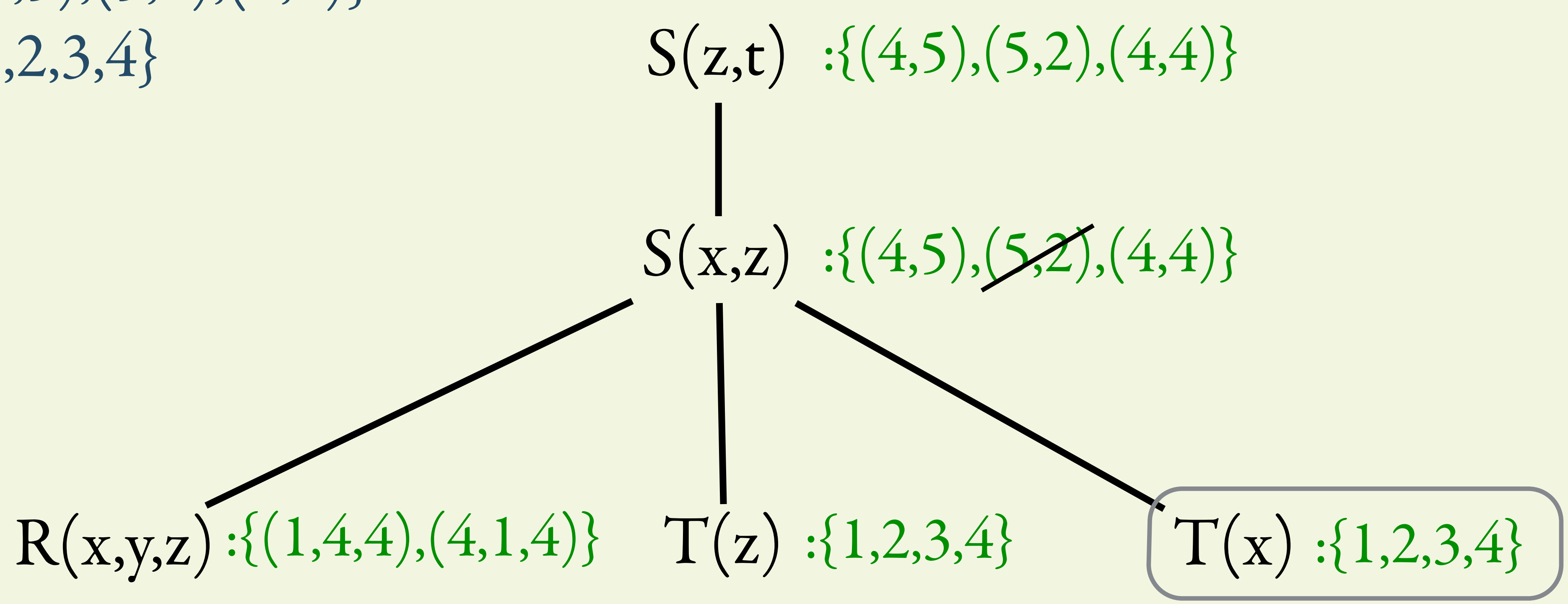


Acyclic CQ's

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

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

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 $O(|\phi|.|D|)$

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

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) :{$\cancel{(4,5)}$,$\cancel{(5,2)}$,(4,4)}

R(x,y,z) :{(1,4,4),(4,1,4)}

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

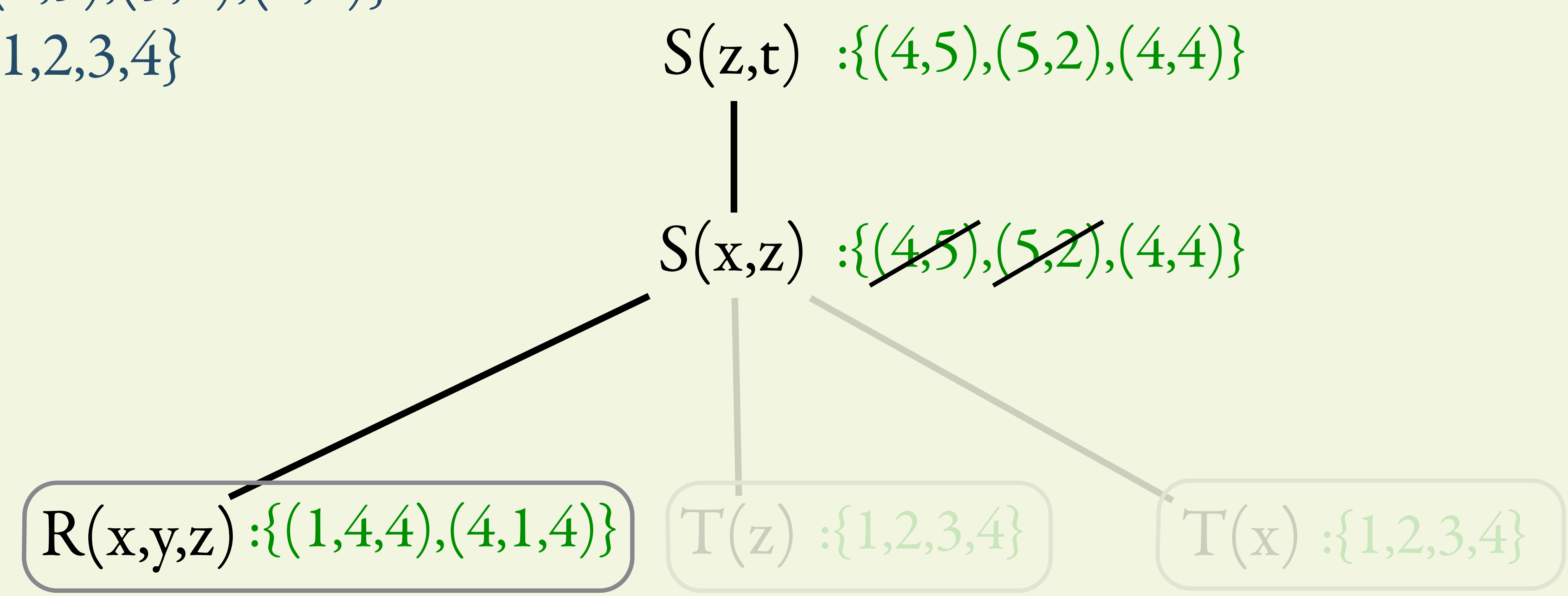
T(x) :{1,2,3,4}

Acyclic CQ's

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

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

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(|ϕ|.|D|)

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

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

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

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

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

Acyclic CQ's

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

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

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

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

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

$R(x,y,z)$:{(1,4,4),(4,1,4)} $T(z)$:{1,2,3,4} $T(x)$:{1,2,3,4}

Acyclic CQ's

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

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

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

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

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

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

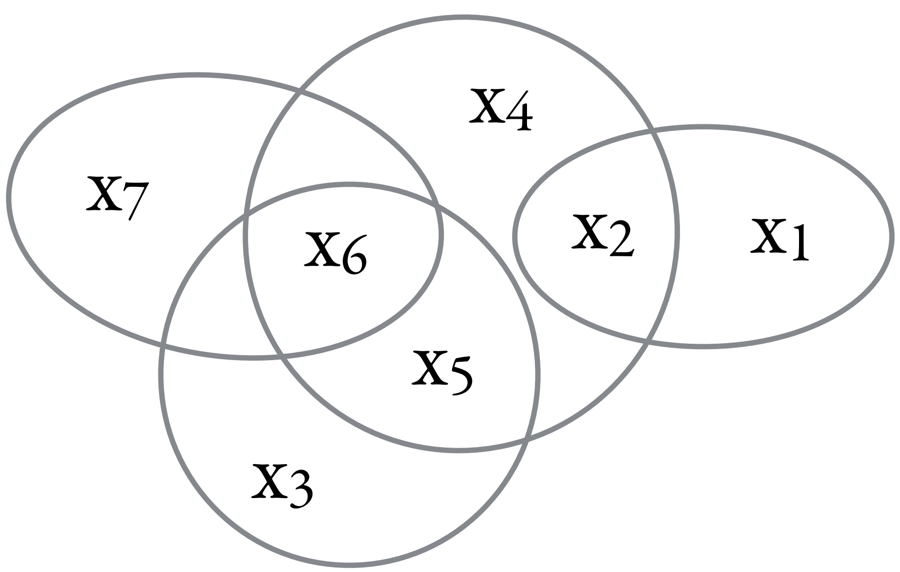
How to compute a join tree?

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:

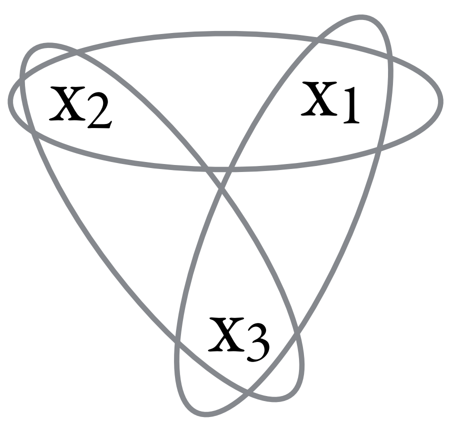
(1) There is a witness e' in E, such that e' ≠ e and each vertex from e is either
  (a) **only** in e or
  (b) in e'; or

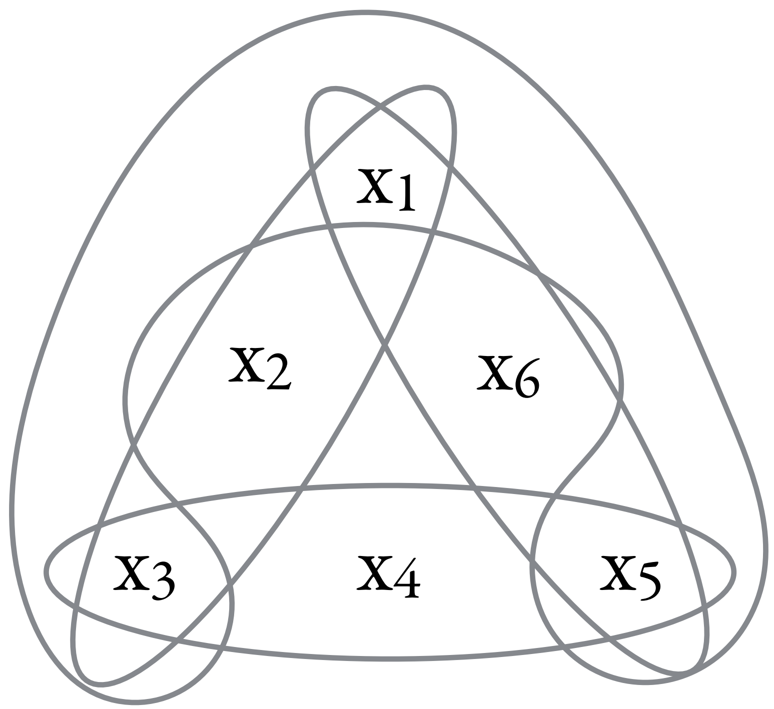
(2) **e** has no intersection with any other hyperedge.



ears?

Ears?





Ears!

Definition: The GYO **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 ϕ having **G** as underlying canonical hyper-graph is acyclic
- The hyper graph **G** is α-acyclic

Ears!

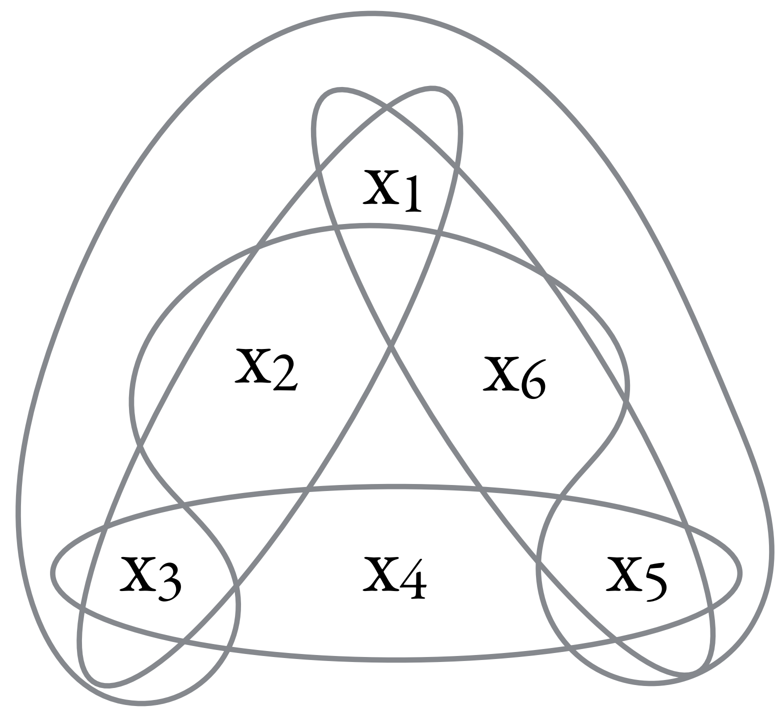
Definition: The GYO **reduct** of a hyper-graph is the result of removing ears until no more ears are left.

Theorem: TFAE

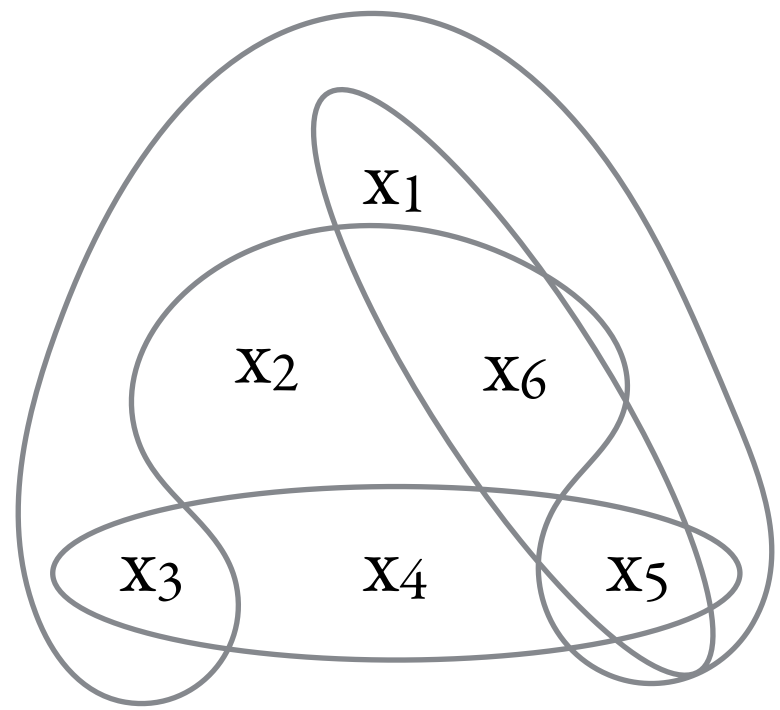
- The GYO **reduct** of a hyper graph **G** is empty
- A CQ ϕ having **G** as underlying canonical hyper-graph is acyclic
- The hyper graph **G** is α-acyclic

We can test acyclicity by computing the GYO reduct!

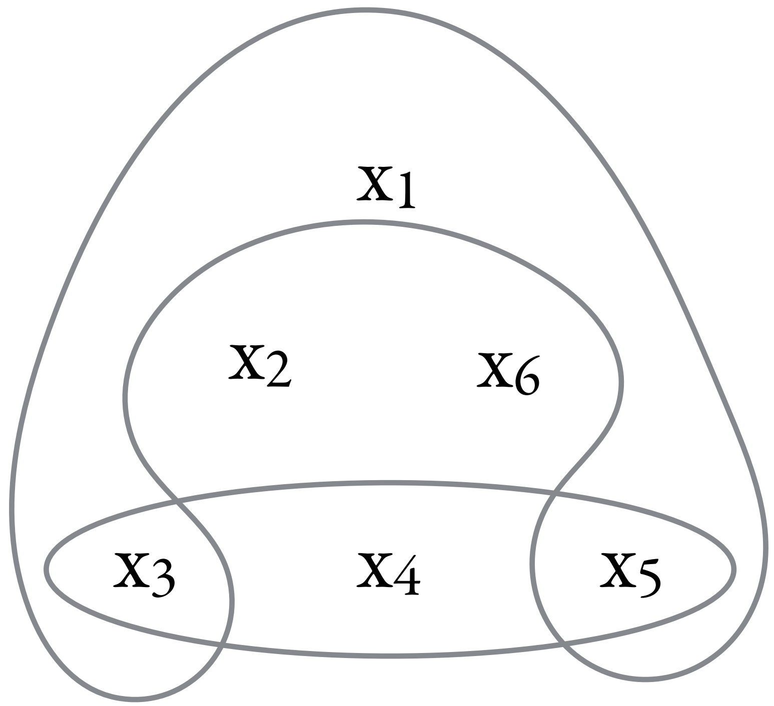
Ears?



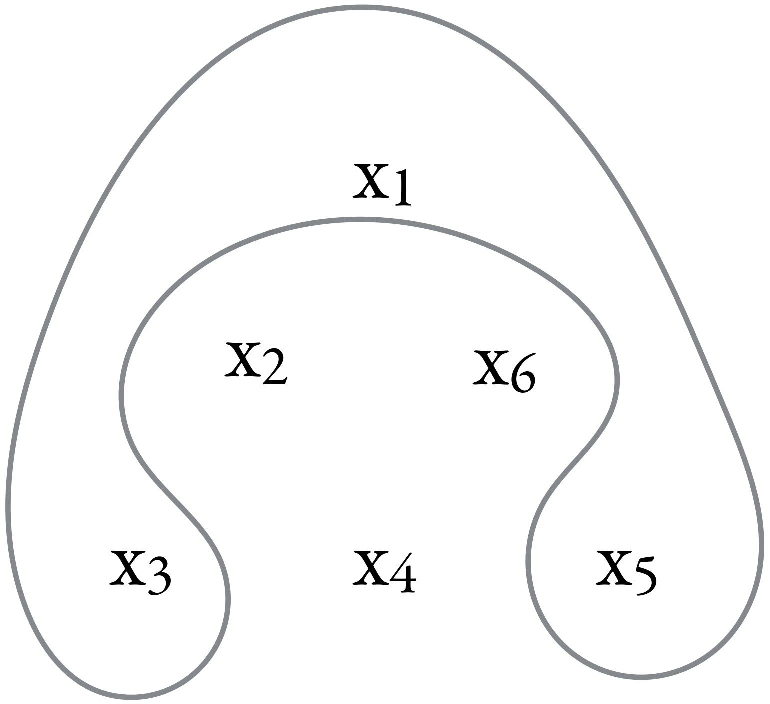
Ears?



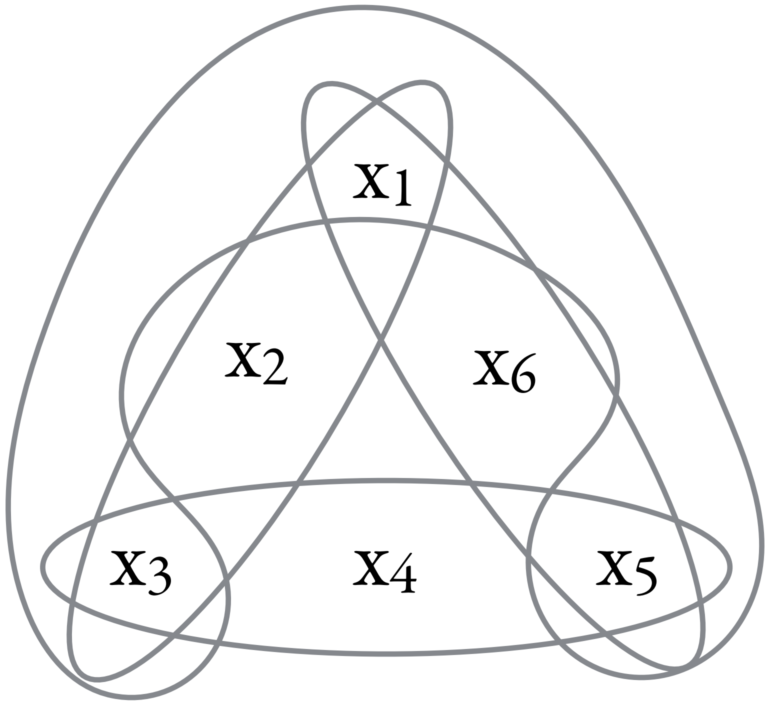
Ears?



Ears?



Ears?



Acyclic!

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.

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)

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)

T(x,x)

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

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

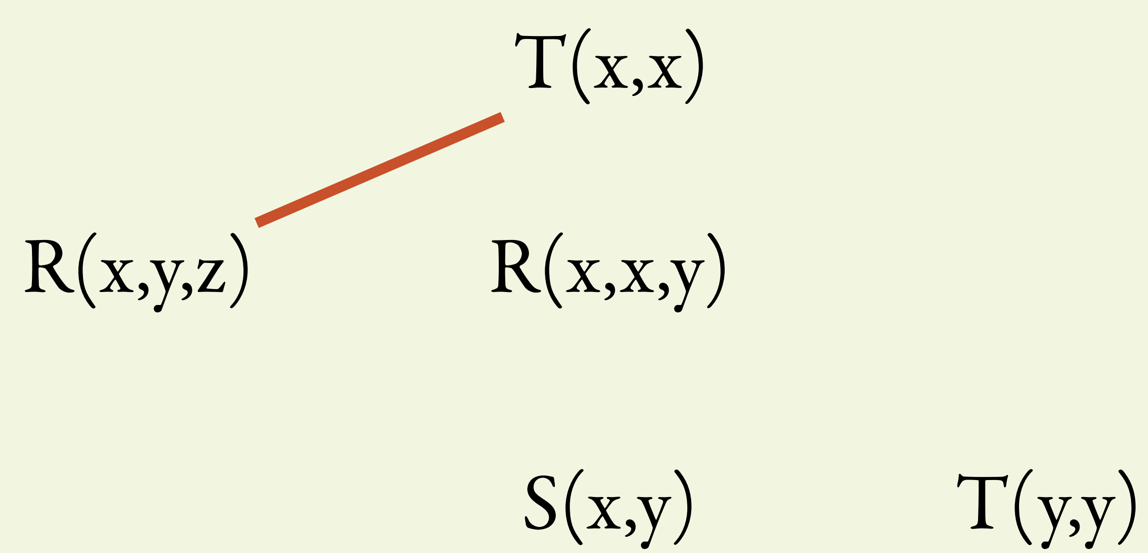
Acyclic CQ's

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)



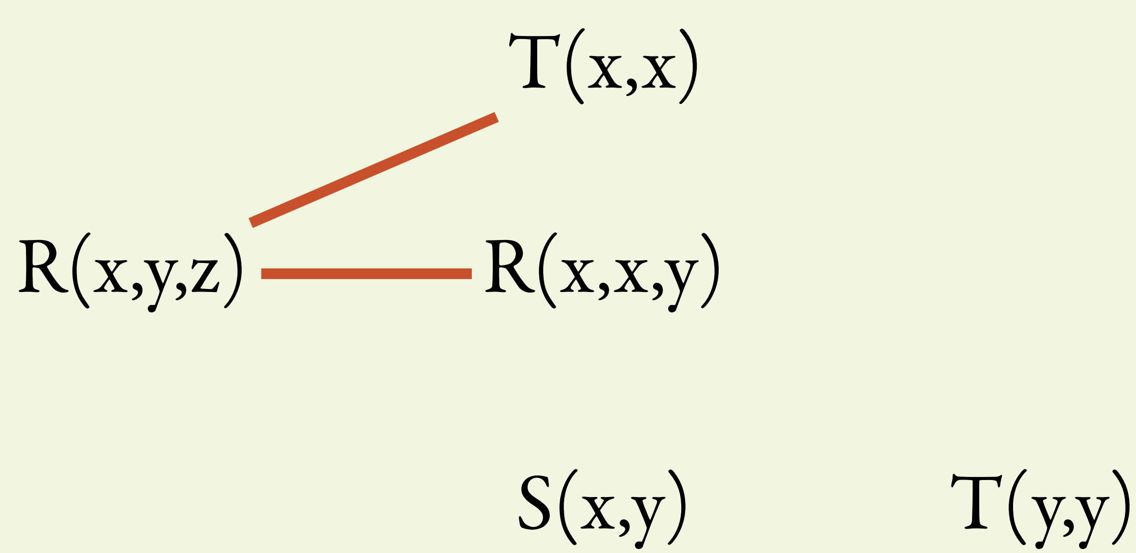
Acyclic CQ's

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)



Acyclic CQ's

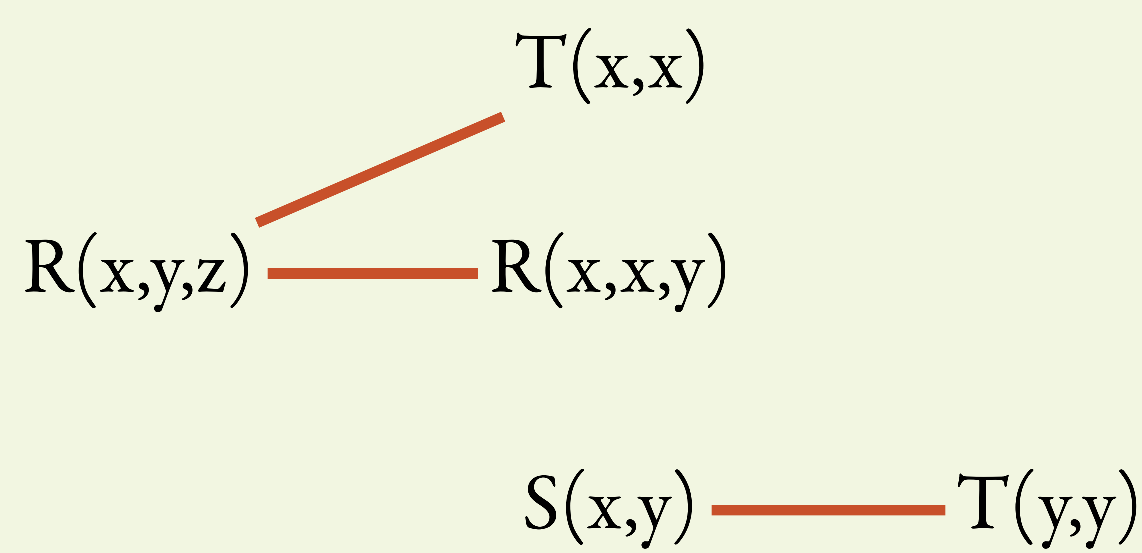
How to compute a join tree?

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)~~



Acyclic CQ's

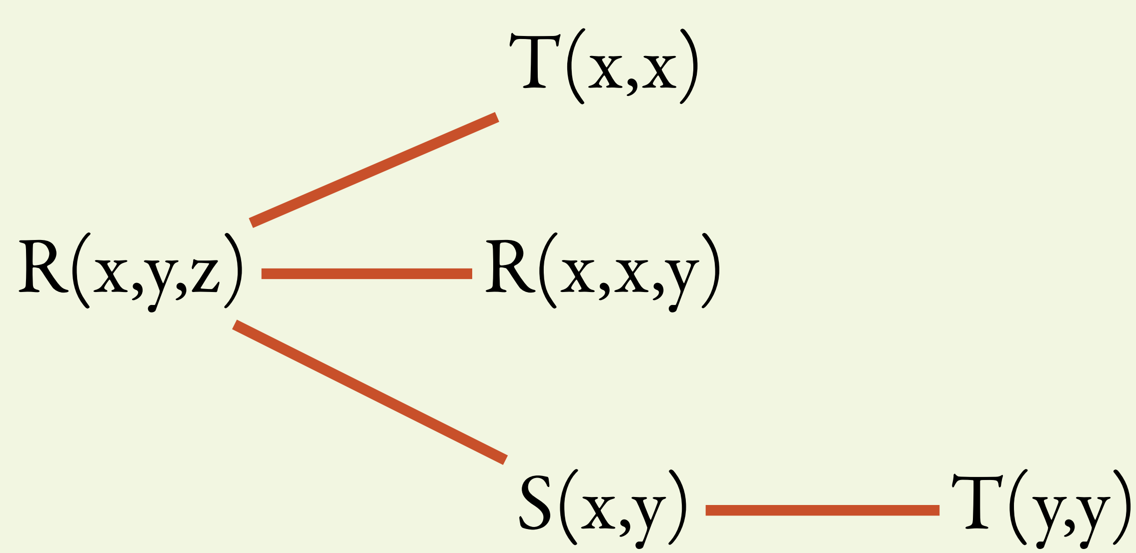
How to compute a join tree?

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)~~



Acyclic CQ's

How to compute a join tree?

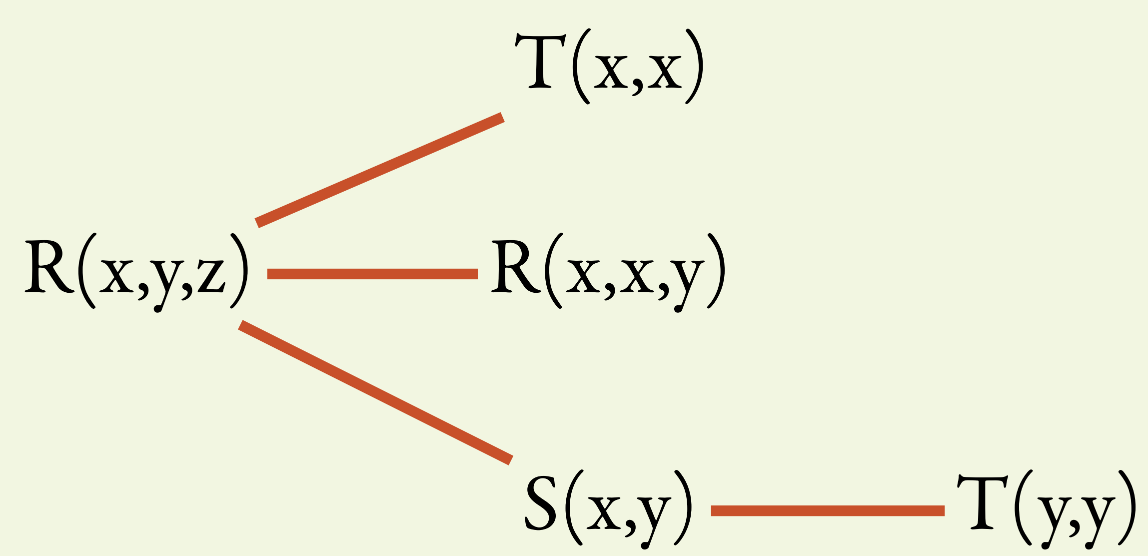
GYO algorithm [Graham, Yu, Ozsoyoglu]

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.

Remove ears until you're left with only one!

E.g.

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



Acyclic CQ's

[Gottlob, Leone, Scarcello]

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

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

  tw(Q) = tw(canonical hypergraph of Q)

  tw = 1 are forests,

  tw = 2 are graphs without K_4 as a minor, ...

- For fixed k,

  computing whether Q has tw ≤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

  tw(Q) = tw(canonical hypergraph of Q)

  tw = 1 are forests,

  tw = 2 are graphs without K_4 as a minor, ...

- For fixed k,

  computing whether Q has tw ≤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 tw ≤ 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 i

$\phi \subseteq \psi$ iff $\psi(G_\phi) \neq \varnothing$

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]≠ FPT, and C a r.e. class of graphs. [Grohe, Schwentick, Segoufin]

Querying with semi-joins

The semi-join

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

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

$\ltimes$, $\cup$, π, σ, $\setminus$, *dupcol*

Querying with semi-joins

The semi-join

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

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

$\ltimes$, $\cup$, π, σ, $\setminus$, *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 $|D|$?

Bounded variable FO

Def.

FO^k = The fragment of FO restricted to k variable names

Bounded variable FO

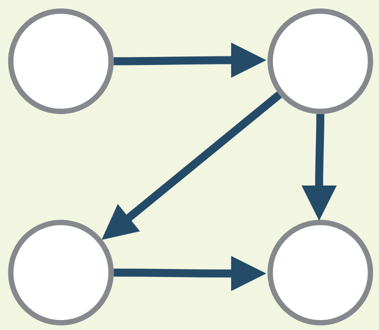
Def.

FO^k = The fragment of FO restricted to k variable names

$\phi(x)$ = "Every neighbour of x has an outgoing path of length 2"

$$= \forall y. \Big(E(x,y) \implies \exists z\, \exists w \,(E(y,z) \wedge E(z,w)) \Big) \in \mathrm{FO}^4$$

G



Bounded variable FO

Def.

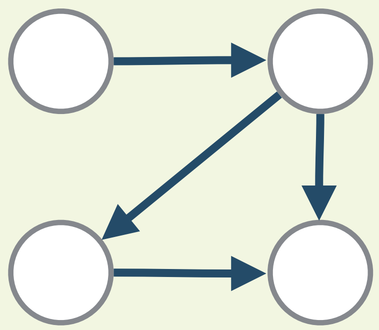
FO^k = The fragment of FO restricted to k variable names

$\phi(x)$ = "Every neighbour of x has an outgoing path of length 2"

$$= \forall y. \Big(E(x,y) \implies \exists z\, \exists w \,\big(E(y,z) \wedge E(z,w) \big) \Big) \in \mathrm{FO}^4$$

Question: in FO^2 ?

G



Bounded variable FO

Def.

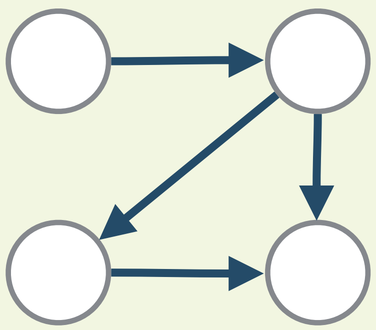
FO^k = The fragment of FO restricted to k variable names

$\phi(x)$ = "Every neighbour of x has an outgoing path of length 2"

$$= \forall y. \Big(E(x,y) \implies \exists z\, \exists w \,\big(E(y,z) \wedge E(z,w) \big) \Big) \quad \in \mathrm{FO}^4$$

$$= \forall y. \Big(E(x,y) \implies \exists x \,\big(E(y,x) \wedge \exists y\, E(x,y) \big) \Big) \quad \in \mathrm{FO}^2$$

G



Bounded variable FO

Def.

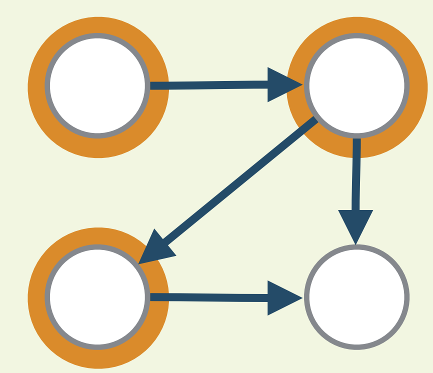
$$\mathrm{FO}^k \;=\; \text{The fragment of FO restricted to } k \text{ variable names}$$

$\phi(x)$ = "Every neighbour of x has an outgoing path of length 2"

$$= \forall y.\, \Big(E(x,y) \implies \exists z\, \exists w \,\big(E(y,z) \wedge E(z,w) \big) \Big) \;\in\; \mathrm{FO}^4$$

$$= \forall y.\, \Big(E(x,y) \implies \exists x \,\big(E(y,x) \wedge \underbrace{\exists y\, E(x,y)}_{\text{PTIME}} \big) \Big) \;\in\; \mathrm{FO}^2$$

G



Bounded variable FO

Def.

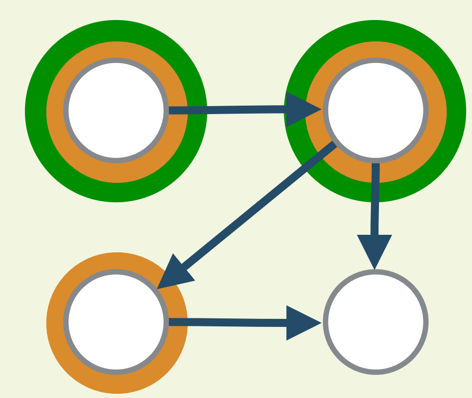
FO^k = The fragment of FO restricted to k variable names

$\phi(x)$ = "Every neighbour of x has an outgoing path of length 2"

$$= \forall y.\, \Big(\, E(x,y) \implies \exists z\, \exists w\, \big(E(y,z) \wedge E(z,w)\big)\Big) \quad \in \mathrm{FO}^4$$

$$= \forall y.\, \Big(\, E(x,y) \implies \underbrace{\exists x\, \big(E(y,x) \wedge \underbrace{\exists y\, E(x,y)}_{\text{PTIME}}\big)}_{\text{PTIME}}\Big) \quad \in \mathrm{FO}^2$$

G



Bounded variable FO

Def.

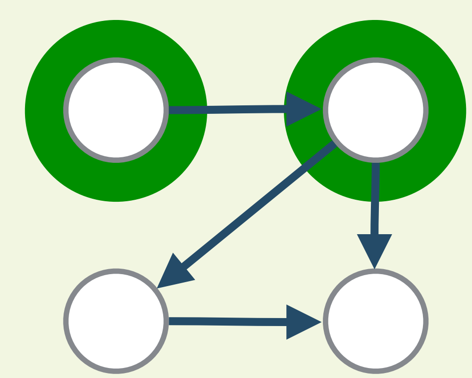
FO^k = The fragment of FO restricted to k variable names

$\phi(x)$ = "Every neighbour of x has an outgoing path of length 2"

$$= \forall y. \Big(E(x,y) \implies \exists z\, \exists w \,\big(E(y,z) \wedge E(z,w) \big) \Big) \quad \in FO^4$$

$$= \forall y. \Big(E(x,y) \implies \underbrace{\exists x \,\big(E(y,x) \wedge \exists y\, E(x,y) \big)}_{\text{PTIME}} \Big) \quad \in FO^2$$

G



Bounded variable FO

Def.

FO^k = The fragment of FO restricted to k variable names

$\phi(x)$ = "Every neighbour of x has an outgoing path of length 2"

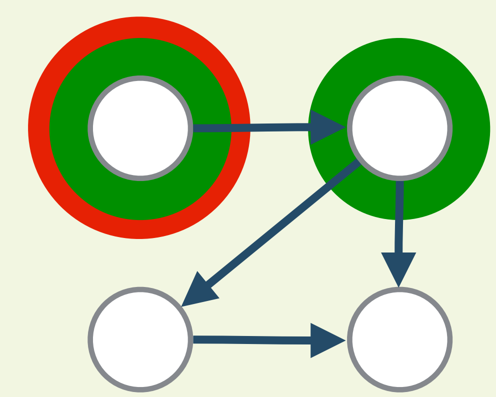
$$= \forall y.\ \Big(E(x,y) \implies \exists z\, \exists w\, \big(E(y,z) \wedge E(z,w) \big) \Big) \ \in \ \mathrm{FO}^4$$

$$= \forall y.\ \Big(E(x,y) \implies \exists x\, \big(E(y,x) \wedge \exists y\, E(x,y) \big) \Big) \ \in \ \mathrm{FO}^2$$

PTIME

PTIME

G



Bounded variable FO

Def.

FO^k = The fragment of FO restricted to k variable names

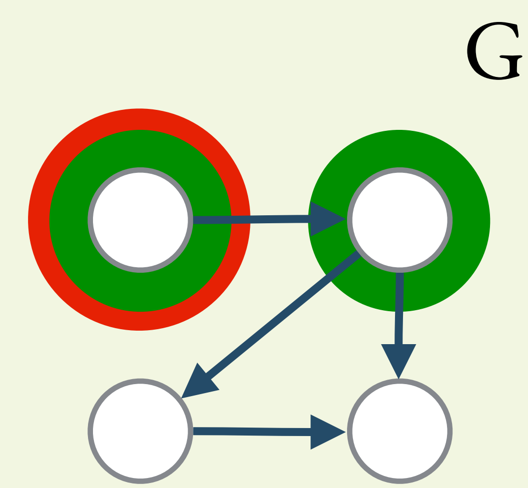
$\phi(x)$ = "Every neighbour of x has an outgoing path of length 2"

$$= \forall y. \Big(E(x,y) \implies \exists z\, \exists w \, \big(E(y,z) \wedge E(z,w) \big) \Big) \quad \in \mathrm{FO}^4$$

$$= \forall y. \Big(E(x,y) \implies \exists x \, \big(E(y,x) \wedge \exists y \, E(x,y) \big) \Big) \quad \in \mathrm{FO}^2$$

PTIME

PTIME



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

Bounded variable FO

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

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

Bounded variable FO

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

Maximum number of nested quantifiers "$\exists x(\ldots \forall y(\ldots \exists x(\ldots \exists z\underbrace{(\ldots)}_{\text{qr } 0})))$"

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

Bounded variable FO

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

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

qr 0

qr 1

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

Bounded variable FO

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

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

qr 0

qr 1

qr 2

$\ddots$

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

Bounded variable FO

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

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

qr 0

qr 1

qr 2

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

1. Evaluate **qr=0** subformulas α and output result in relations $R_{0,\alpha}$
2. Evaluate **qr=1** subformulas β based on $R_{0,\alpha}$ and output in $R_{1,\beta}$
3. Evaluate **qr=2** subformulas γ based on $R_{1,\beta}$ and output in $R_{1,\gamma}$
4. …

⋮

r. …

Bounded variable FO

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

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

qr 0

qr 1

qr 2

$\ddots$

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

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

   $\rightsquigarrow |V|^k \cdot (|\alpha| \cdot |G|)^p$

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

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

4. …

$\vdots$

r. …

Bounded variable FO

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

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

qr 0

qr 1

qr 2

$\ddots$

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

1. Evaluate **qr=0** subformulas α and output result in relations $R_{0,\alpha}$
   $\rightsquigarrow |V|^k \cdot (|\alpha| \cdot |G|)^p$
2. Evaluate **qr=1** subformulas β based on $R_{0,\alpha}$ and output in $R_{1,\beta}$
   $\rightsquigarrow |V|^k \cdot (|\beta| \cdot (|G| + |R_1|))^p$
   $\leq |V|^k$
3. Evaluate **qr=2** subformulas γ based on $R_{1,\beta}$ and output in $R_{1,\gamma}$
4. …

⋮

r. …

Bounded variable FO

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

Maximum number of nested quantifiers "$\exists x(\dots \forall y(\dots \exists x(\dots \exists z(\dots)))) $"

qr 0

qr 1

qr 2

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

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

   $\rightsquigarrow |V|^k \cdot (|\alpha| \cdot |G|)^p$

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

   $\rightsquigarrow |V|^k \cdot (|\beta| \cdot (|G| + \underbrace{|R_1|}_{\leq |V|^k}))^p$

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

   $\rightsquigarrow |V|^k \cdot (|\gamma| \cdot (|G| + \underbrace{|R_2|}_{\leq |V|^k}))^p$

4. …

⋮

r. …

Bounded variable FO

Desirable:

- Given k and a FO query ϕ, is ϕ in FO^k ? ⇝ 💀 Undecidable (even w.o. $\neg$)

Bounded variable FO

Desirable:

- Given k and a FO query ϕ, is ϕ in FO^k ? ⇝ 💀 Undecidable (even w.o. $\neg$)

- Given k and a CQ query ϕ, is ϕ in FO^k ? ⇝ NP-complete

Bounded variable FO

Desirable:

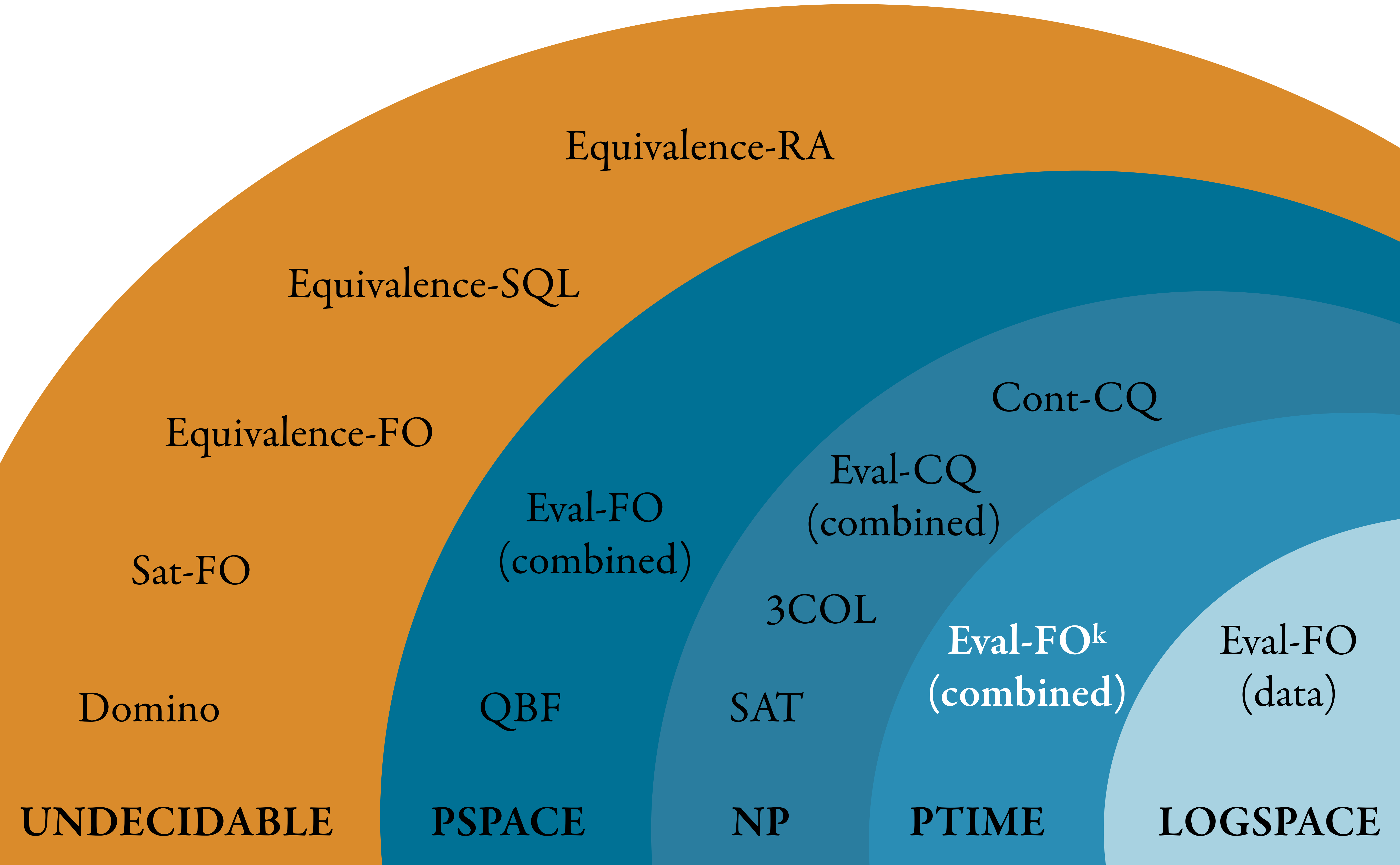
- Given k and a FO query ϕ, is ϕ in FO^k ? ⇝ 💀 Undecidable (even w.o. $\neg$)

- Given k and a CQ query ϕ, is ϕ in FO^k ? ⇝ NP-complete

- Satisfiability for FO^k ⇝ Undecidable if $k \geq 3$ (Domino)

  ⇝ NEXPTIME-complete if $k=2$

Recap



Equivalence-RA
Equivalence-SQL
Equivalence-FO
Sat-FO
Domino
Eval-FO (combined)
QBF
Cont-CQ
Eval-CQ (combined)
3COL
SAT
Eval-FO^k (combined)
Eval-FO (data)
UNDECIDABLE
PSPACE
NP
PTIME
LOGSPACE

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 AC^0"

⇝ "A property is ∃SO-definable iff it can be tested in NP" [Fagin 73]

⇝ Open problem: which logic captures PTIME?

Some more cool stuff…

Recursion

Datalog (semantics based on least fixpoint)

```
Ancestor(X,Y) :- Parent(X,Z), Ancestor(Z,Y)
Ancestor(X,X) :- .
?- Ancestor("Louis XIV",Y)
```

Some more cool stuff…

Recursion

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

Datalog (semantics based on least fixpoint)

```
Ancestor(X,Y) :- Parent(X,Z), Ancestor(Z,Y)
Ancestor(X,X) :- .
?- Ancestor("Louis XIV",Y)
```

Some more cool stuff…

Recursion

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

Datalog (semantics based on least fixpoint)

```
Ancestor(X,Y) :- Parent(X,Z), Ancestor(Z,Y)
Ancestor(X,X) :- .
?- Ancestor("Louis XIV",Y)
```

⇝ Incomparable with FO (has recursion, but is monotone)

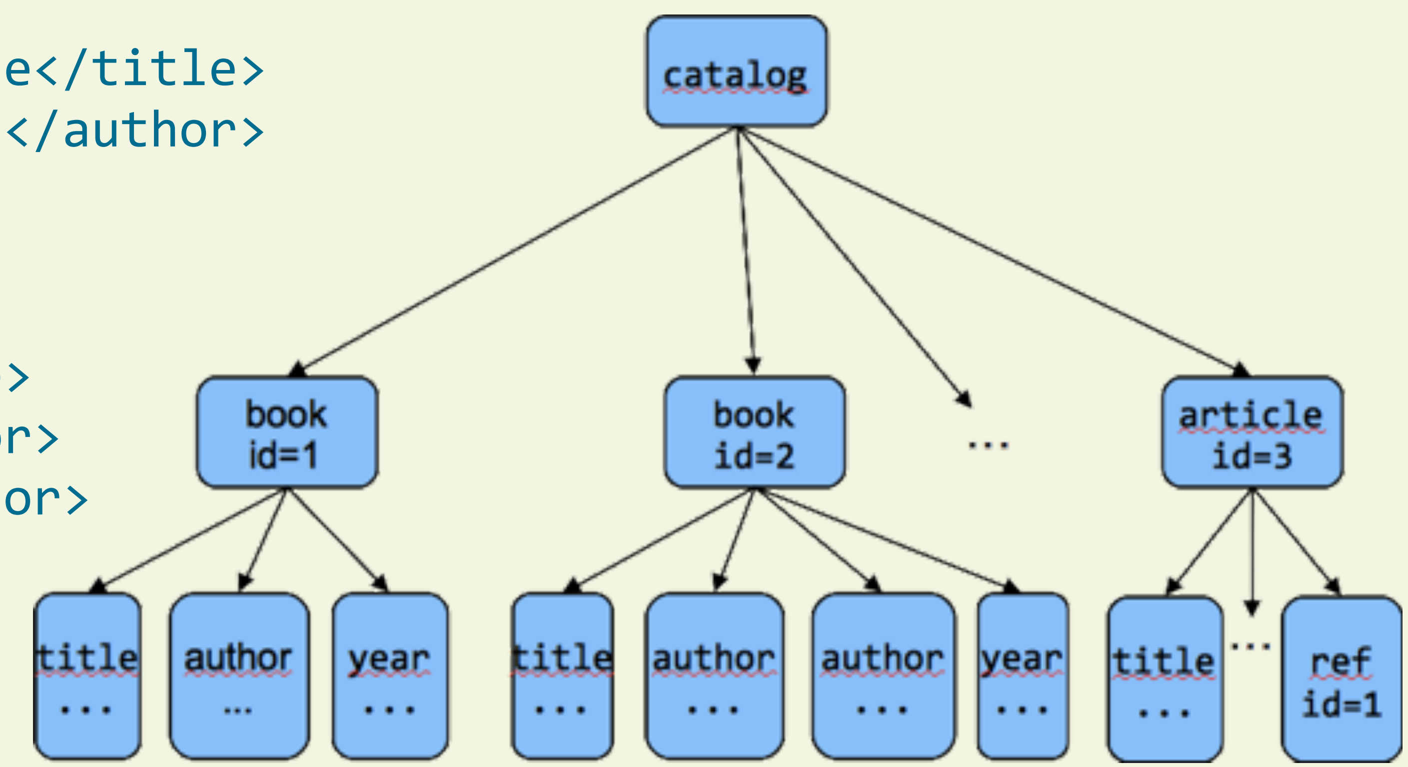
⇝ Evaluation is in PTIME (for data complexity, but also for bounded arity)

Semi-structured data

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

XML, XPath, Stream processing, …

```
<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>
```



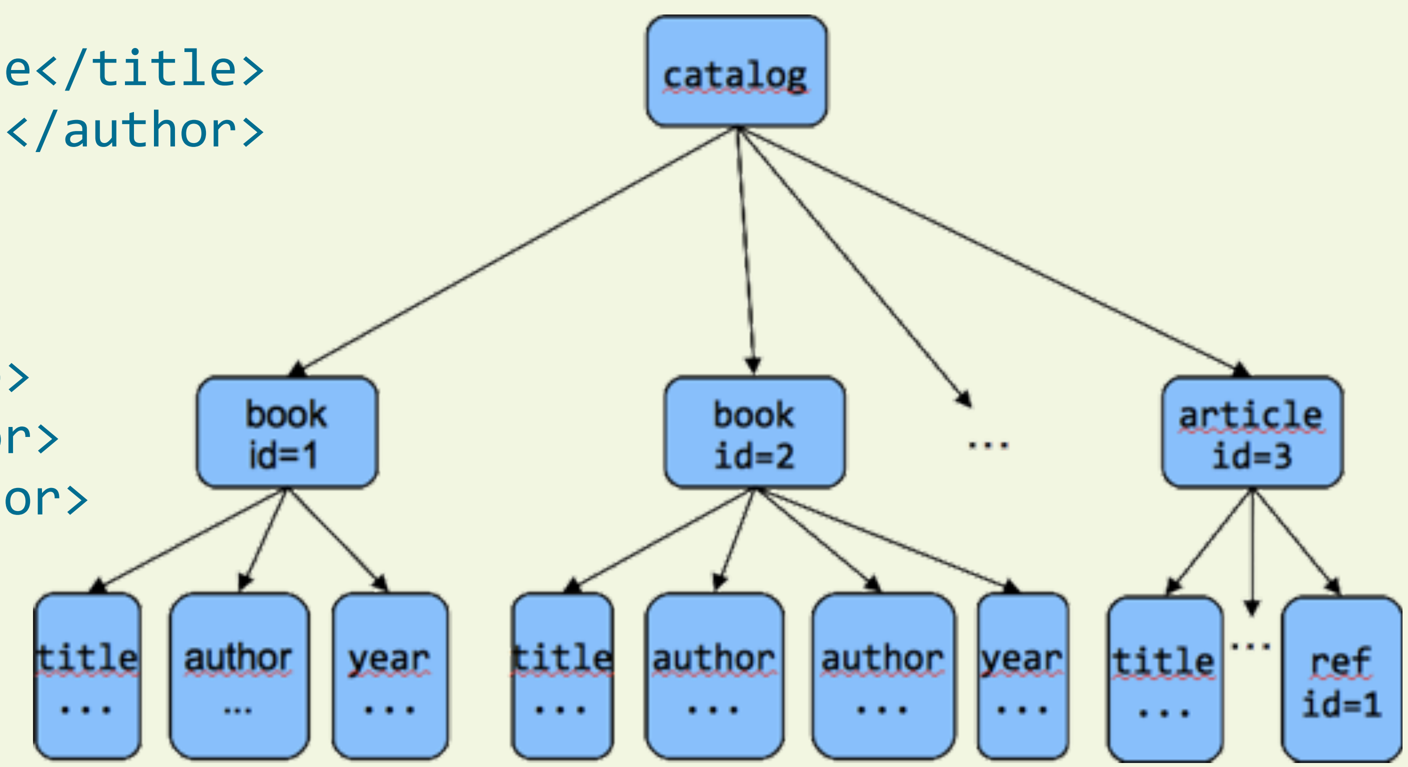
Some more cool stuff…

Semi-structured data

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

XML, XPath, Stream processing, …

```
<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>
```



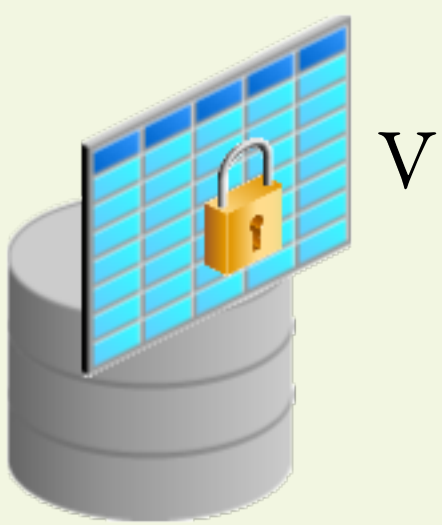
⇝ Evaluation of XPath is in linear time (data complexity) [Bojanczyk, Parys 08]

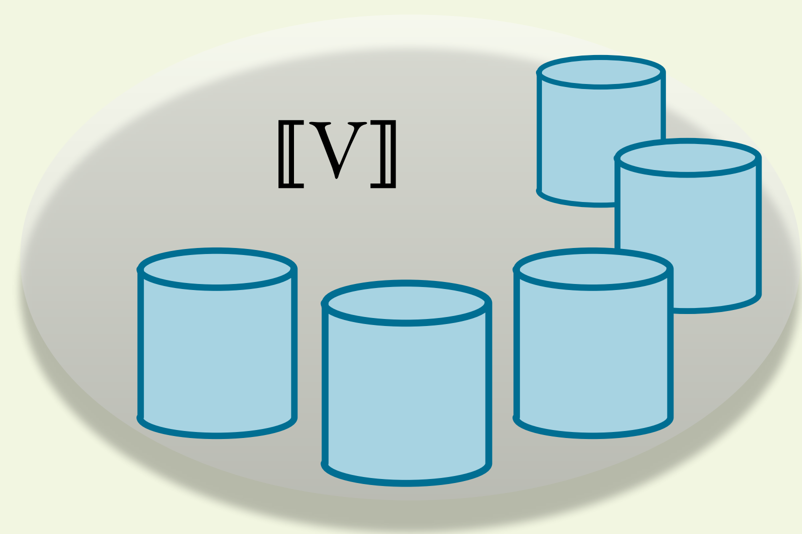
⇝ Satisfiability for $FO^2[\downarrow,\sim]$ is decidable [Bojanczyk, Muscholl, Schwentick, Segoufin 09]

Some more cool stuff…

Incomplete information

Certain Query Answers (CQA)





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

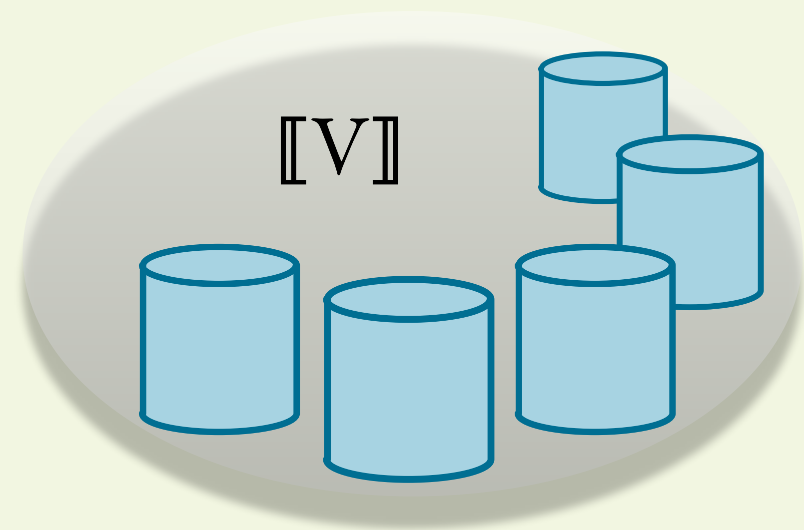
Some more cool stuff…

Incomplete information

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

Certain Query Answers (CQA)

V



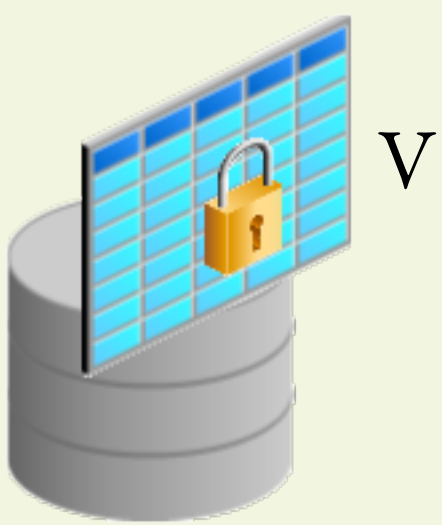
$$\phi[\![V]\!] = \bigcap_{D \in [\![V]\!]} \phi(D)$$

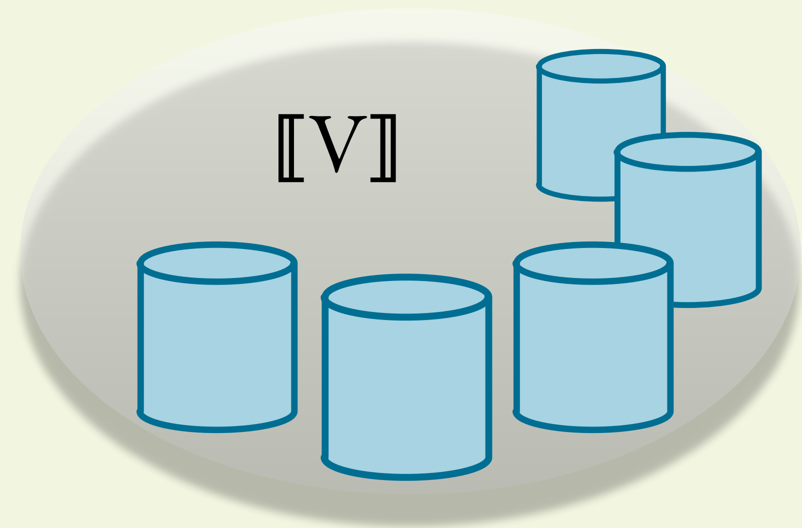
Some more cool stuff...

Incomplete information

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

Certain Query Answers (CQA)





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

⇝ 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

- Abiteboul, Hull, Vianu, “Foundations of Databases”, Addison-Wesley, 1995.

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

- Libkin, “Elements of Finite Model Theory”, Springer, 2004.

- Immerman, “Descriptive Complexity”, Springer, 1999.