day 3

ESSLLI 2016
Bolzano, Italy

Logical foundations of databases

Diego Figueira

Gabriele Puppis

CNRS LaBRI

Recap

- **Active domain semantics and expressiveness:** $FO^{act} =^{*} RA$

- **Undecidable problems** (Halting ≤ Domino ≤ FO-Satisfiability ≤ FO-Equivalence)

- **Data complexity / Combined complexity**

- **Evaluation problem for FO:** in PSPACE (combined comp.)
  in PSPACE (query comp.)
  in LOGSPACE (data comp.)

- **Positive FO:** evaluation in NP (combined comp.)

- Conjunctive Queries

Conjunctive Queries

Def.

CQ = FO without $\forall, \neg, \vee$

Eg: $\phi(x, y) = \exists z . (\text{Parent}(x, z) \wedge \text{Parent}(z, y))$

Usual notation: "Grandparent(X,Y) : – Parent(X,Z), Parent(Z,Y)"

Conjunctive Queries

Def.

CQ = FO without $\forall, \neg, \vee$

Normal form: " $\exists\, x_1, \ldots, x_n \,.\, \phi(x_1, \ldots, x_n)$ "

quantifier-free and no equalities!

Eg: $\phi(x, y) = \exists\, z \,.\, (\text{Parent}(x, z) \wedge \text{Parent}(z, y))$

Usual notation: "Grandparent(X,Y) : – Parent(X,Z), Parent(Z,Y)"

Conjunctive Queries

Def.

CQ = FO without ∀,¬,∨

Normal form: " $\exists x_1, \ldots, x_n . \phi(x_1, \ldots, x_n)$ "

quantifier-free and no equalities!

Eg: $\phi(x, y) = \exists z . (\text{Parent}(x, z) \wedge \text{Parent}(z, y))$

Usual notation: "Grandparent(X,Y) : – Parent(X,Z), Parent(Z,Y)"

It corresponds to positive "SELECT-FROM-WHERE" SQL queries

```
Select ...
From ...
Where Z
```

no negation or disjunction

It corresponds to "π-σ-×" RA queries

$\pi_X(\sigma_Z(R_1 \times \cdots \times R_n))$

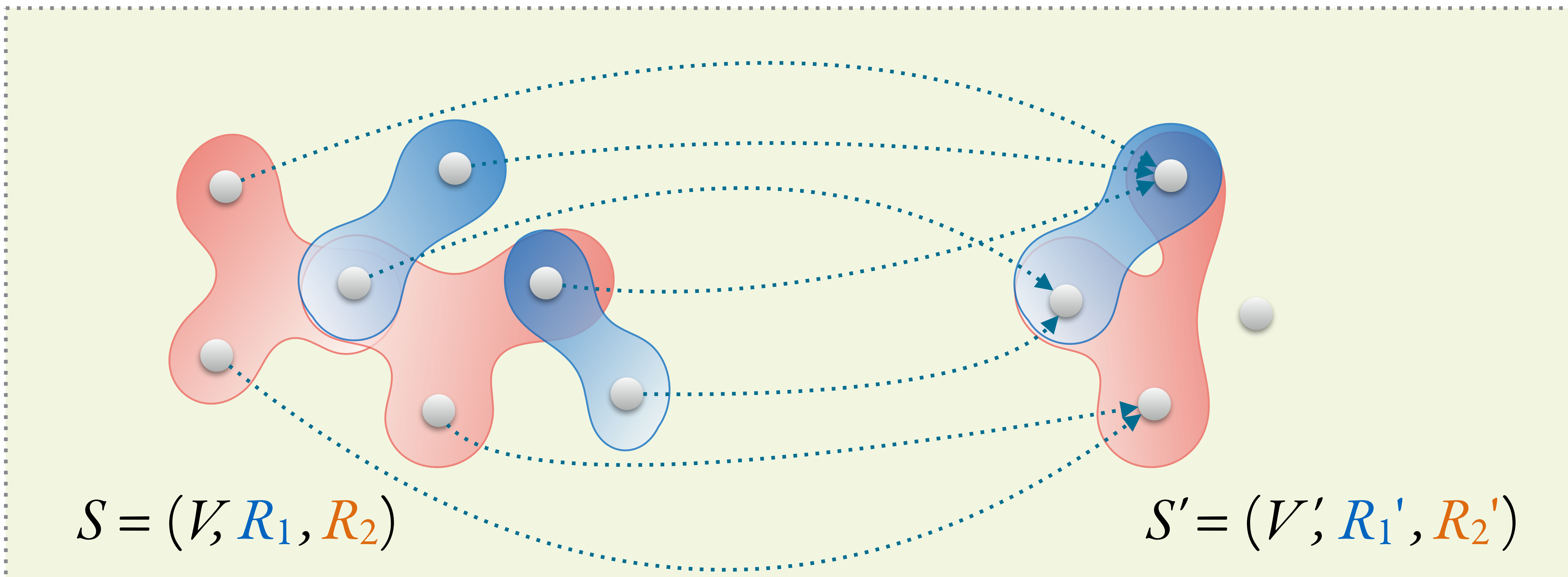
no negation

Homomorphisms

Homomorphism between structures $S=(V,R_1,\ldots,R_n)$ and $S'=(V',R_1',\ldots,R_n')$

is a function $h: V \longrightarrow V'$ such that

$(x_1,\ldots,x_n) \in R_i$ implies $(h(x_1),\ldots,h(x_n)) \in R_i'$

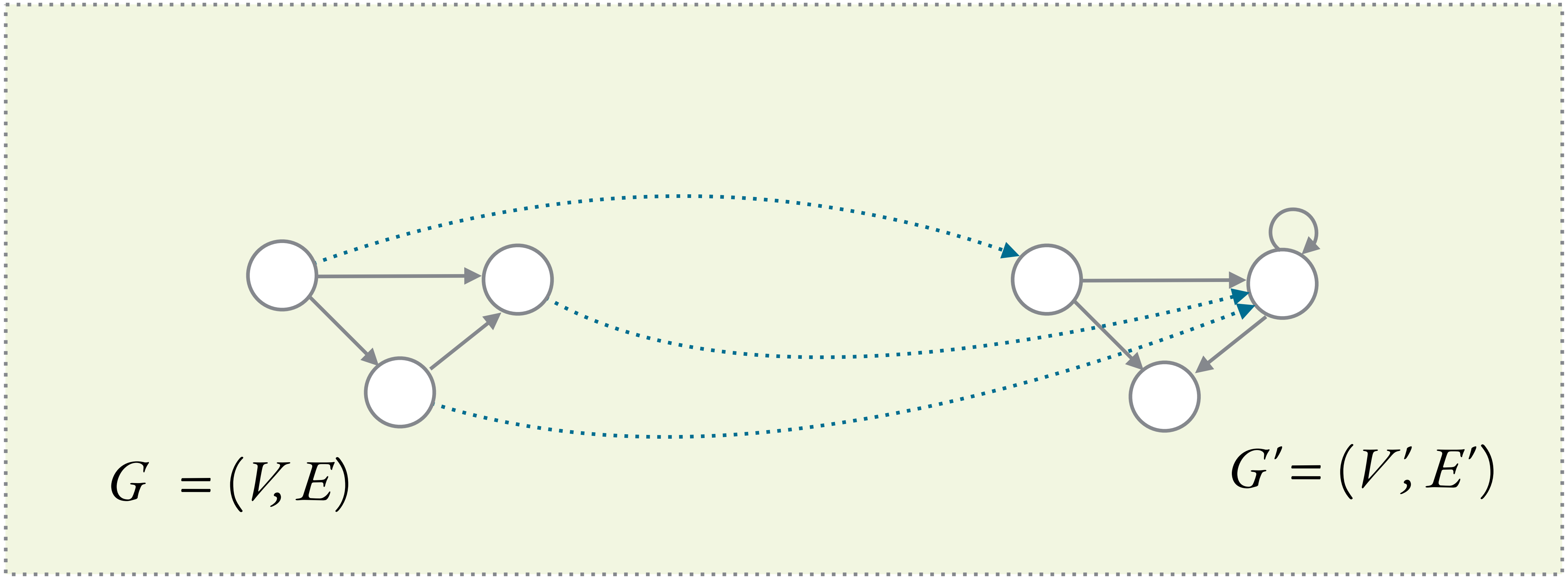


Homomorphisms

Homomorphism between structures $S=(V, R_1, \ldots, R_n)$ and $S'=(V', R_1', \ldots, R_n')$

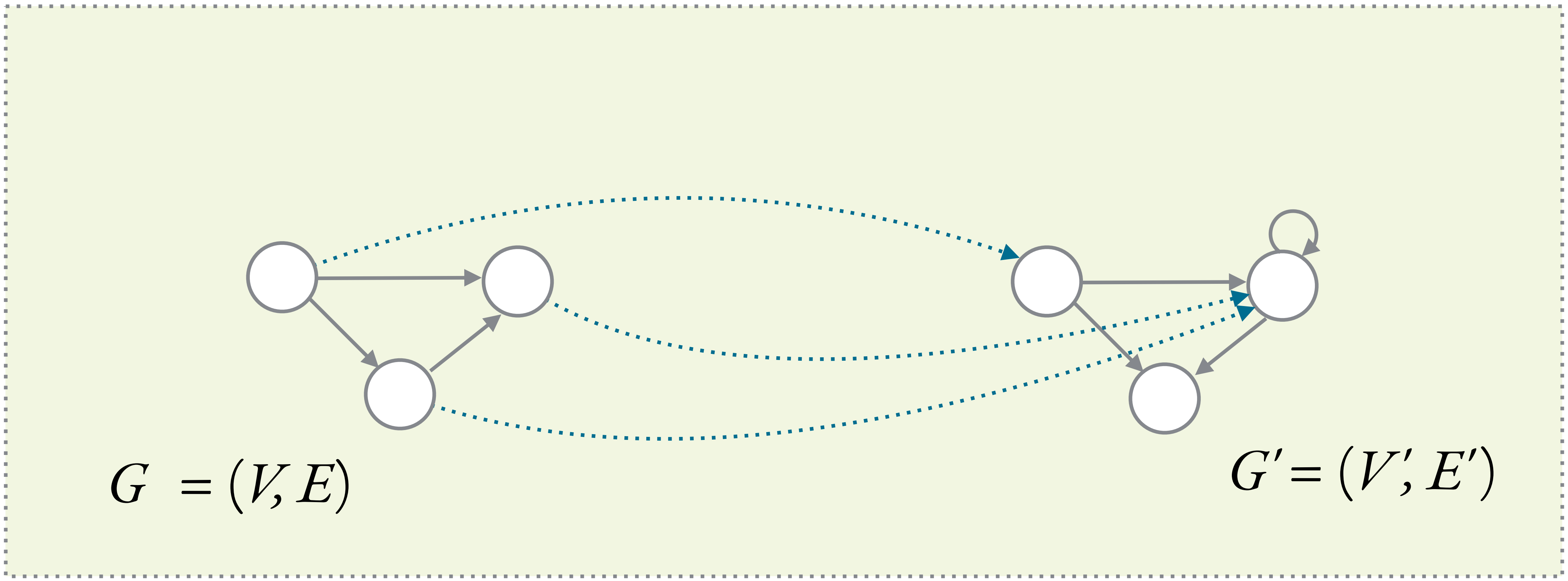
is a function $h : V \longrightarrow V'$ such that

$(x_1, \ldots, x_n) \in R_i$ implies $(h(x_1), \ldots, h(x_n)) \in R_i'$



Canonical structures

Canonical structure G_ϕ of a Conjunctive Query ϕ has
- variables as nodes
- tuples $(x_1, \ldots, x_n) \in R_i$ for all atomic sub-formulas $R_i(x_1, \ldots, x_n)$ of ϕ

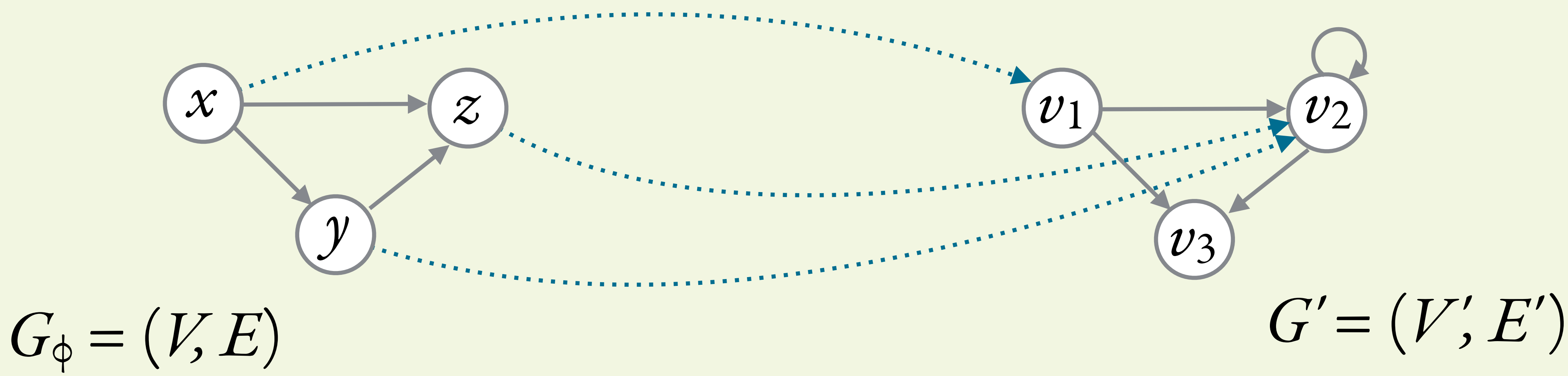


Canonical structures

Canonical structure G_ϕ of a Conjunctive Query ϕ has

- variables as nodes
- tuples $(x_1, \ldots, x_n) \in R_i$
  for all atomic sub-formulas $R_i(x_1, \ldots, x_n)$ of ϕ

E.g.: $\phi = \exists x \, \exists y \, \exists z \, . \, (\mathrm{E}(x,y) \wedge \mathrm{E}(y,z) \wedge \mathrm{E}(x,z))$



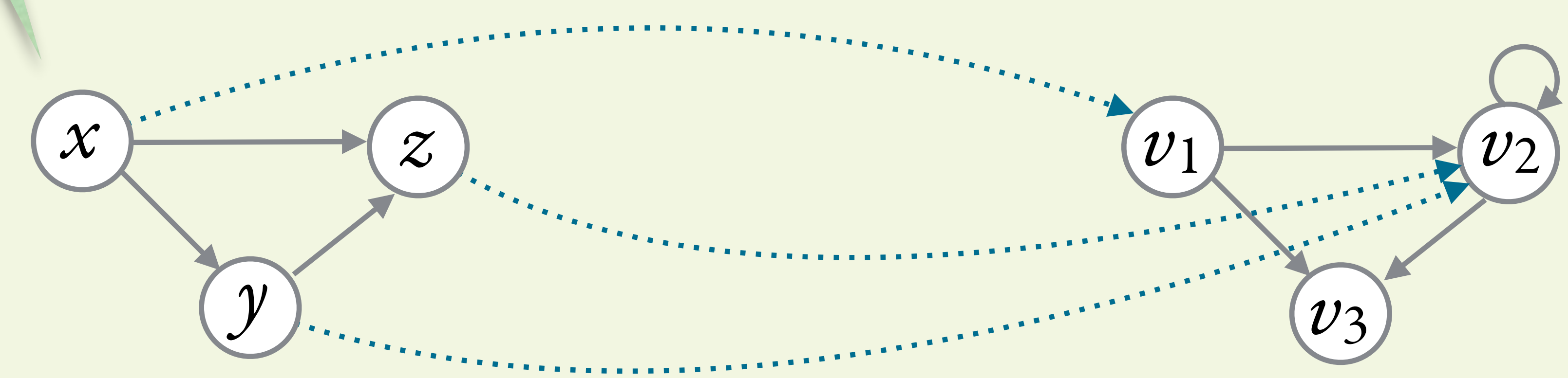
Canonical structures

Canonical structure G_ϕ of a Conjunctive Query ϕ has

- variables as nodes
- tuples $(x_1, \dots, x_n) \in R_i$ for all atomic sub-formulas $R_i(x_1, \dots, x_n)$ of ϕ

Fact 1: $G_\phi \models \phi$

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



$G_\phi = (V, E)$

$G' = (V', E')$

Canonical structures

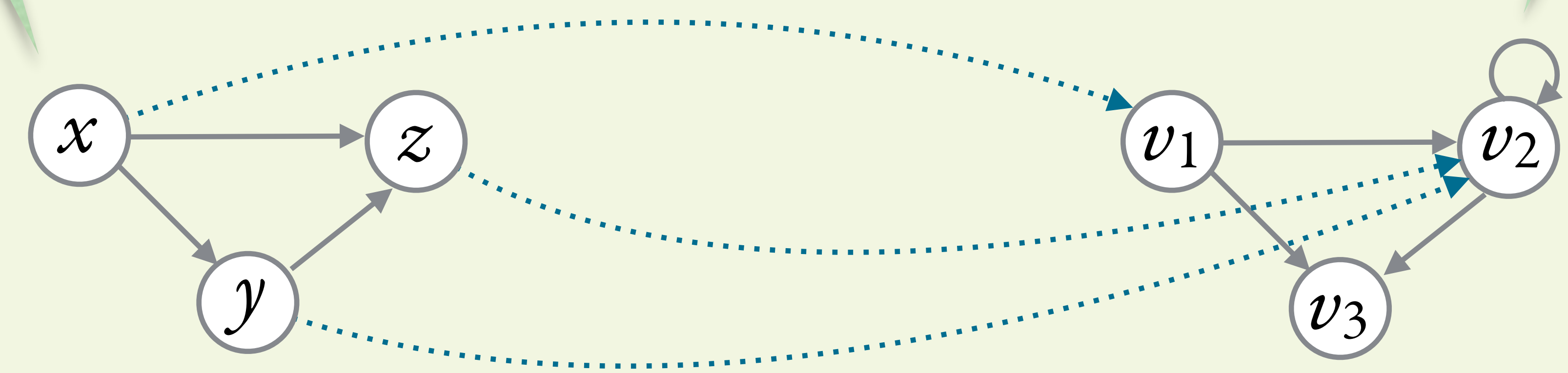
Canonical structure G_ϕ of a Conjunctive Query ϕ has

- variables as nodes
- tuples $(x_1, \ldots, x_n) \in R_i$
  for all atomic sub-formulas $R_i(x_1, \ldots, x_n)$ of ϕ

Fact 1: $G_\phi \models \phi$

Fact 2: $h(G_\phi) \models \phi$

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



$G_\phi = (V, E)$

$G' = (V', E')$

Canonical structures

Canonical structure G_ϕ of a Conjunctive Query ϕ has

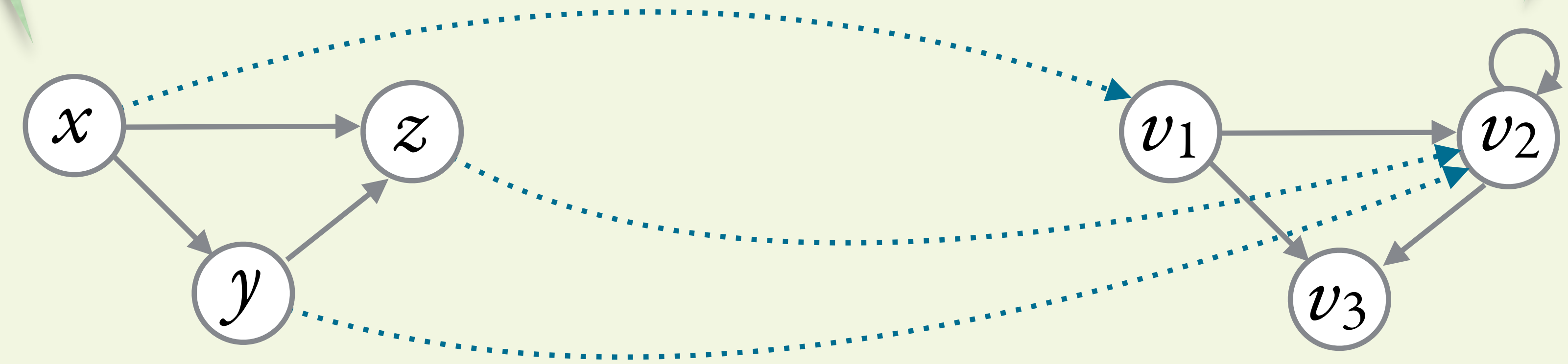
- variables as nodes
- tuples $(x_1, \ldots, x_n) \in R_i$ for all atomic sub-formulas $R_i(x_1, \ldots, x_n)$ of ϕ

Fact 1: $G_\phi \models \phi$

Fact 3: $G'' \models \phi$ iff $\exists$ h: $G_\phi \longrightarrow G''$

Fact 2: $h(G_\phi) \models \phi$

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



$G_\phi = (V, E)$

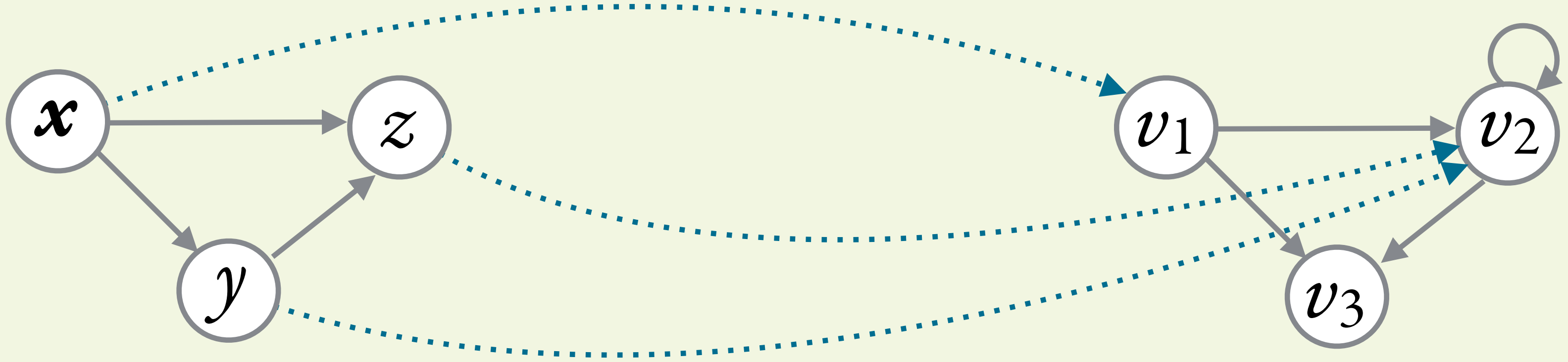
$G' = (V', E')$

Evaluation via homomorphisms

Lemma. The evaluation of a CQ $\phi(x_1, \ldots, x_n)$ on S' returns the set

$$\phi(S') = \{ (\mathrm{h}(x_1), \ldots, \mathrm{h}(x_n)) \mid \mathrm{h} : G_\phi \longrightarrow S' \}$$

E.g.: $\phi(x) = \exists y \, \exists z \, . \, (\mathrm{E}(x,y) \wedge \mathrm{E}(y,z) \wedge \mathrm{E}(x,z))$



$G_\phi = (V, E)$

$G' = (V', E')$

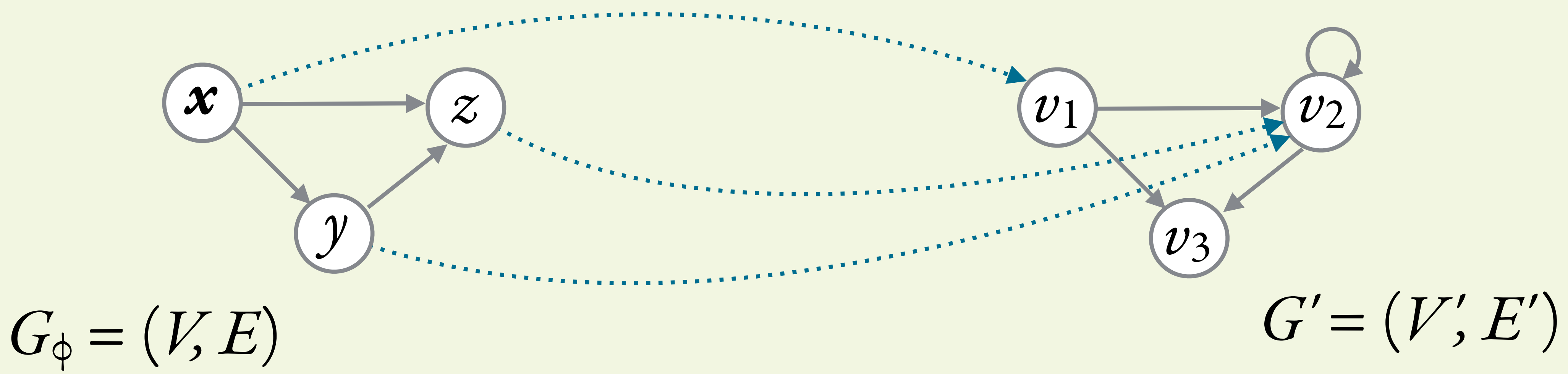
Evaluation via homomorphisms

Lemma. The evaluation of a CQ $\phi(x_1, \ldots, x_n)$ on S' returns the set

$$\phi(S') = \{ (h(x_1), \ldots, h(x_n)) \mid h : G_\phi \longrightarrow S' \}$$

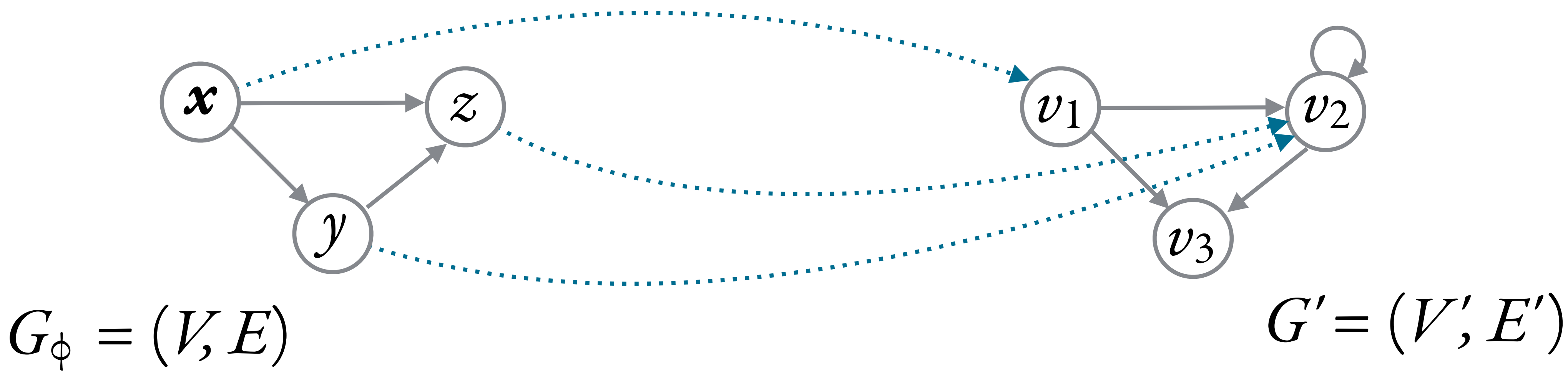
Question: Which are the homomorphisms $G_\phi \longrightarrow G'$?
What is the result of $\phi(G')$?

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



Evaluation via homomorphisms

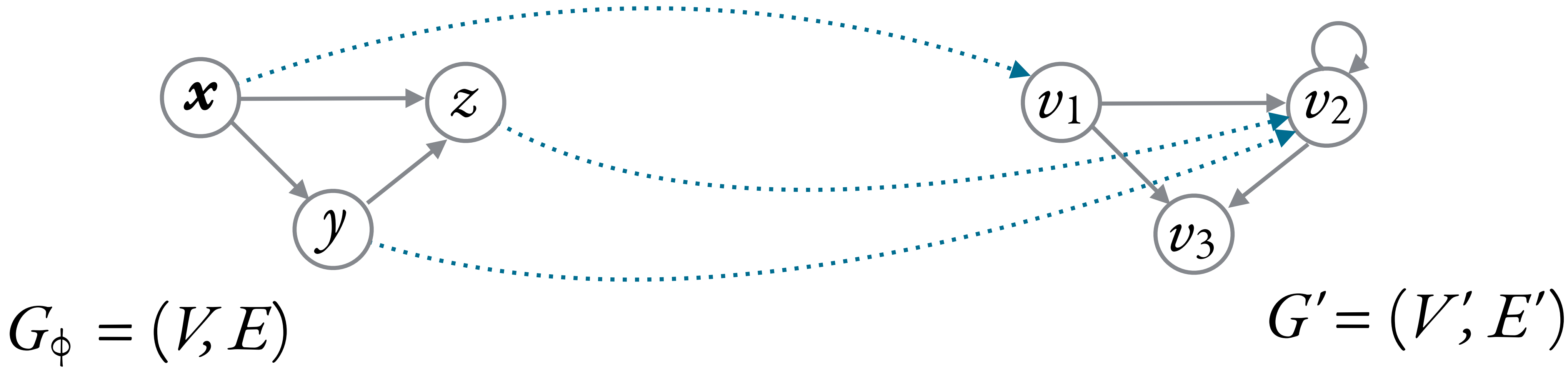
Theorem. Evaluation of CQ is in NP (combined complexity)



Input: A CQ $\phi(x_1, \ldots, x_n)$, a graph G, a tuple $(a_1, \ldots, a_n)$
Output: Is $(a_1, \ldots, a_n) \in \phi(G)$?

Evaluation via homomorphisms

Theorem. Evaluation of CQ is in NP (combined complexity)



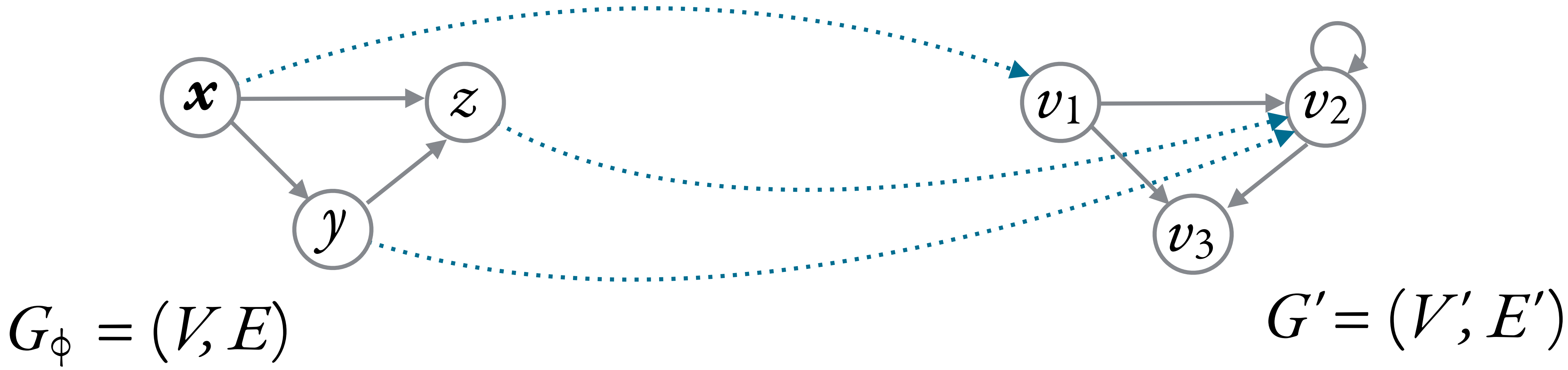
Input: A CQ $\phi(x_1, \ldots, x_n)$, a graph G, a tuple $(a_1, \ldots, a_n)$

Output: Is $(a_1, \ldots, a_n) \in \phi(G)$?

Ideas?

Evaluation via homomorphisms

Theorem. Evaluation of CQ is in NP (combined complexity)



Input: A CQ $\phi(x_1, \ldots, x_n)$, a graph G, a tuple $(a_1, \ldots, a_n)$

Output: Is $(a_1, \ldots, a_n) \in \phi(G)$?

Ideas?

1. Guess h: $G_\phi \longrightarrow G$
2. Check that it is a homomorphism
3. Output YES if $(h(x_1), \ldots, h(x_n)) = (a_1, \ldots, a_n)$; NO otherwise.

Evaluation via homomorphisms

Theorem. Evaluation of CQ is NP-complete (combined complexity)

Evaluation via homomorphisms

Theorem. Evaluation of CQ is NP-complete (combined complexity)

NP-complete problem: **3-COLORABILITY**

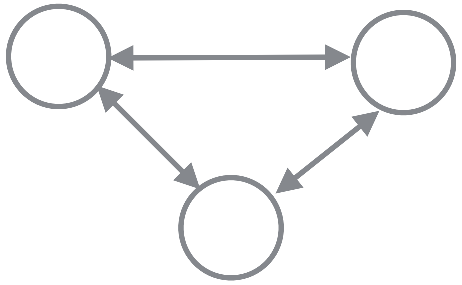
Input: A graph G

Output: Can we assign a colour from $\{R,G,B\}$ to each node so that adjacent nodes have always different colours ?

=

Is there a *homomorphism* from G to K_3 ?

K_3



Evaluation via homomorphisms

Theorem. Evaluation of CQ is NP-complete (combined complexity)

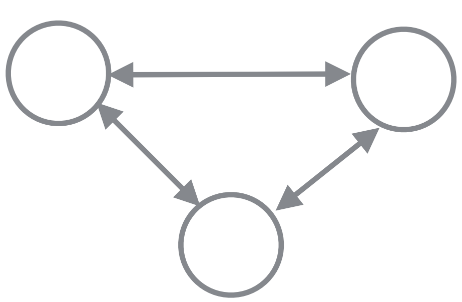
NP-complete problem: **3-COLORABILITY**

Input: A graph G

Output: Can we assign a colour from $\{R,G,B\}$ to each node so that adjacent nodes have always different colours ?

=

Is there a *homomorphism* from G to K_3 ?



Reduction 3COL ⇝ CQ-EVAL: 1. Given G, build a CQ ϕ such that $G_\phi = G$.
2. Test if $() \in \phi(K_3)$.

Monotonicity and preservation theorems

Lemma. Every CQ is **monotone**:

$S \subseteq S'$ implies $\phi(S) \subseteq \phi(S')$

Monotonicity and preservation theorems

Lemma. Every CQ is **monotone**:

$$S \subseteq S' \text{ implies } \phi(S) \subseteq \phi(S')$$

Proof:

$$\begin{aligned}\phi(S) &= \{ (h(x_1), ..., h(x_n)) \mid h : G_\phi \longrightarrow S \} \\ &\subseteq \{ (h'(x_1), ..., h'(x_n)) \mid h' : G_\phi \longrightarrow S' \} \\ &= \phi(S')\end{aligned}$$

Monotonicity and preservation theorems

Lemma. Every CQ is **monotone**:

$S \subseteq S'$ implies $\phi(S) \subseteq \phi(S')$

Proof:

$$\phi(S) = \{ (h(x_1), ..., h(x_n)) \mid h : G_\phi \longrightarrow S \}$$
$$\subseteq \{ (h'(x_1), ..., h'(x_n)) \mid h' : G_\phi \longrightarrow S' \}$$
$$= \phi(S')$$

- "The relation R has at most 2 elements" $\notin$ CQ
- "There is a node connected to every other node" $\notin$ CQ
- "The radius of the graph is 5" $\notin$ CQ

Monotonicity and preservation theorems

Theorem. If an FO query ϕ is **monotone**
then $\phi \in \mathrm{UCQ}$ [Rossman '08]

Monotonicity and preservation theorems

Theorem. If an FO query ϕ is **monotone**

then $\phi \in$ UCQ [Rossman '08]

Finite unions of CQs

$\approx$ $\exists \wedge \vee$ fragment of FO

Monotonicity and preservation theorems

Theorem. If an FO query ϕ is **monotone**

then $\phi \in$ UCQ [Rossman '08]

Finite unions of CQs

$\approx$ $\exists \wedge \vee$ fragment of FO

Equally expressive, but
UCQ are less succinct

Monotonicity and preservation theorems

Theorem. If an FO query ϕ is **monotone**

then $\phi \in \mathrm{UCQ}$ [Rossman '08]

Finite unions of CQs

$\approx$ $\exists \wedge \vee$ fragment of FO

Equally expressive, but
UCQ are less succinct

- One example of the few properties which still hold on finite structures.
- Proof in the finite is difficult and independent.

Static analysis with CQs

The satisfiability problem for CQ is decidable…

Question: What is the algorithm for CQ-SAT? What is the complexity?

Static analysis with CQs

The satisfiability problem for CQ is decidable…

Question: What is the algorithm for CQ-SAT? What is the complexity?

Answer: CQs are always satisfiable by their canonical structure!

$$G_\phi \models \phi$$

Static analysis with CQs

problem: **CQ-CONTAINMENT**

Input: Two CQs ϕ, ψ

Output: Does $\phi(S) \subseteq \psi(S)$ holds for every structure S ?

Static analysis with CQs

problem: **CQ-CONTAINMENT**

Input: Two CQs ϕ, ψ

Output: Does $\phi(S) \subseteq \psi(S)$ holds for every structure S ?

Theorem. The containment problem for CQ is NP-complete

Static analysis with CQs

problem: **CQ-CONTAINMENT**

Input: Two CQs ϕ, ψ
Output: Does $\phi(S) \subseteq \psi(S)$ holds for every structure S ?

Theorem. The containment problem for CQ is NP-complete

Question: Is this combined or data complexity?

Static analysis with CQs

problem: **CQ-CONTAINMENT**

Input: Two CQs ϕ, ψ
Output: Does $\phi(S) \subseteq \psi(S)$ holds for every structure S ?

Theorem. The containment problem for CQ is NP-complete

Question: Is this combined or data complexity? **Answer:** None!

Static analysis with CQs

problem: **CQ-CONTAINMENT**

Input: Two CQs ϕ, ψ

Output: Does $\phi(S) \subseteq \psi(S)$ holds for every structure S ?

Theorem. The containment problem for CQ is NP-complete

$\phi(x_1, \ldots, x_n)$ is contained in $\psi(y_1, \ldots, y_m)$ iff

1. $n = m$
2. There is $g \colon G_\psi \longrightarrow G_\phi$
3. $g(y_i) = x_i$ for all i

Question: Is this combined or data complexity? **Answer:** None!

Static analysis with CQs

problem: **CQ-CONTAINMENT**

Input: Two CQs ϕ, ψ

Output: Does $\phi(S) \subseteq \psi(S)$ holds for every structure S ?

Theorem. The containment problem for CQ is NP-complete

$\phi(x_1, ..., x_n)$ is contained in $\psi(y_1, ..., y_m)$ iff

1. $n = m$
2. There is $g: G_\psi \longrightarrow G_\phi$
3. $g(y_i) = x_i$ for all i

Why?

Question: Is this combined or data complexity? **Answer:** None!

Static analysis with CQs

$\phi(x_1, \ldots, x_n)$ is contained in $\psi(y_1, \ldots, y_m)$ iff

1. $n = m$
2. There is $g: G_\psi \longrightarrow G_\phi$
3. $g(y_i) = x_i$ for all i

Static analysis with CQs

$\phi(x_1, \ldots, x_n)$ is contained in $\psi(y_1, \ldots, y_m)$ iff

1. $n = m$
2. There is g: $G_\psi \longrightarrow G_\phi$
3. $g(y_i) = x_i$ for all i

[$\Longrightarrow$] Suppose $\forall S \quad \phi(S) \subseteq \psi(S)$

Static analysis with CQs

$\phi(x_1, \ldots, x_n)$ is contained in $\psi(y_1, \ldots, y_m)$ iff

1. $n = m$
2. There is g: $G_\psi \longrightarrow G_\phi$
3. $g(y_i) = x_i$ for all i

$\{ (h(x_1), \ldots, h(x_n)) \mid h : G_\phi \longrightarrow S \}$

[$\Longrightarrow$] Suppose $\forall S \quad \phi(S) \subseteq \psi(S)$

Static analysis with CQs

$\phi(x_1, ..., x_n)$ is contained in $\psi(y_1, ..., y_m)$ iff

1. $n = m$
2. There is g: $G_\psi \longrightarrow G_\phi$
3. $g(y_i) = x_i$ for all i

$\{ (h(x_1), ..., h(x_n)) \mid h : G_\phi \longrightarrow S \}$

[$\Longrightarrow$] Suppose $\forall S \quad \phi(S) \subseteq \psi(S)$

$\{ (g(y_1), ..., g(y_n)) \mid g : G_\psi \longrightarrow S \}$

Static analysis with CQs

$\phi(x_1, \ldots, x_n)$ is contained in $\psi(y_1, \ldots, y_m)$ iff
1. $n = m$
2. There is $g: G_\psi \longrightarrow G_\phi$
3. $g(y_i) = x_i$ for all i

$\{ (h(x_1), \ldots, h(x_n)) \mid h : G_\phi \longrightarrow S \}$

[$\Longrightarrow$] Suppose $\forall S \quad \phi(S) \subseteq \psi(S)$

$\{ (g(y_1), \ldots, g(y_n)) \mid g : G_\psi \longrightarrow S \}$

If there is $h: G_\phi \longrightarrow S$

Then there is $g: G_\psi \longrightarrow S$ such that $h(x_1, \ldots, x_n) = g(y_1, \ldots, y_m)$

Static analysis with CQs

$\phi(x_1, ..., x_n)$ is contained in $\psi(y_1, ..., y_m)$ iff
1. $n = m$
2. There is $g: G_\psi \longrightarrow G_\phi$
3. $g(y_i) = x_i$ for all i

$\{ (h(x_1), ..., h(x_n)) \mid h : G_\phi \longrightarrow S \}$

[$\Longrightarrow$] Suppose $\forall S \quad \phi(S) \subseteq \psi(S)$

$\{ (g(y_1), ..., g(y_n)) \mid g : G_\psi \longrightarrow S \}$

If there is $h: G_\phi \longrightarrow S$

Then there is $g: G_\psi \longrightarrow S$ such that $h(x_1, ..., x_n) = g(y_1, ..., y_m)$

Take $S = G_\phi$ and $h = \textit{identity}$.

Static analysis with CQs

$\phi(x_1, \ldots, x_n)$ is contained in $\psi(y_1, \ldots, y_m)$ iff

1. $n = m$
2. There is $g\colon G_\psi \longrightarrow G_\phi$
3. $g(y_i) = x_i$ for all i

$\{ (h(x_1), \ldots, h(x_n)) \mid h : G_\phi \longrightarrow S \}$

[$\Longrightarrow$] Suppose $\forall S \quad \phi(S) \subseteq \psi(S)$

$\{ (g(y_1), \ldots, g(y_n)) \mid g : G_\psi \longrightarrow S \}$

If there is $h\colon G_\phi \longrightarrow S$

Then there is $g\colon G_\psi \longrightarrow S$ such that $h(x_1, \ldots, x_n) = g(y_1, \ldots, y_m)$

Take $S = G_\phi$ and $h = identity$.

[$\Longleftarrow$] Consider S and $(v_1,\ldots,v_n) \in \phi(S)$.

Then, $(v_1,\ldots,v_n) = (h(x_1), \ldots, h(x_n))$ for some $h : G_\phi \longrightarrow S$.

Static analysis with CQs

$\phi(x_1, ..., x_n)$ is contained in $\psi(y_1, ..., y_m)$ iff

1. $n = m$
2. There is $g: G_\psi \longrightarrow G_\phi$
3. $g(y_i) = x_i$ for all i

$\{ (h(x_1), ..., h(x_n)) \mid h : G_\phi \longrightarrow S \}$

[$\Longrightarrow$] Suppose $\forall S \quad \phi(S) \subseteq \psi(S)$

$\{ (g(y_1), ..., g(y_n)) \mid g : G_\psi \longrightarrow S \}$

If there is $h: G_\phi \longrightarrow S$

Then there is $g: G_\psi \longrightarrow S$ such that $h(x_1, ..., x_n) = g(y_1, ..., y_m)$

Take $S = G_\phi$ and $h = \mathit{identity}$.

[$\Longleftarrow$] Consider S and $(v_1,...,v_n) \in \phi(S)$.

Then, $(v_1,...,v_n) = (h(x_1), ..., h(x_n))$ for some $h: G_\phi \longrightarrow S$.

Since $g(y_1, ..., y_n) = (x_1, ..., x_n)$, then $(v_1, ..., v_n) = h(x_1, ..., x_n) = h(g(y_1, ..., y_n))$.

Static analysis with CQs

$\phi(x_1, \ldots, x_n)$ is contained in $\psi(y_1, \ldots, y_m)$ iff 1. $n = m$

2. There is $g: G_\psi \longrightarrow G_\phi$

3. $g(y_i) = x_i$ for all i

$\{ (h(x_1), \ldots, h(x_n)) \mid h : G_\phi \longrightarrow S \}$

[$\Longrightarrow$] Suppose $\forall S \quad \phi(S) \subseteq \psi(S)$

$\{ (g(y_1), \ldots, g(y_n)) \mid g : G_\psi \longrightarrow S \}$

If there is $h: G_\phi \longrightarrow S$

Then there is $g: G_\psi \longrightarrow S$ such that $h(x_1, \ldots, x_n) = g(y_1, \ldots, y_m)$

Take $S = G_\phi$ and $h = \mathit{identity}$.

[$\Longleftarrow$] Consider S and $(v_1, \ldots, v_n) \in \phi(S)$.

Then, $(v_1, \ldots, v_n) = (h(x_1), \ldots, h(x_n))$ for some $h : G_\phi \longrightarrow S$.

Since $g(y_1, \ldots, y_n) = (x_1, \ldots, x_n)$, then $(v_1, \ldots, v_n) = h(x_1, \ldots, x_n) = h(g(y_1, \ldots, y_n))$.

$h \circ g$ is a homomorphism from G_ψ to S. Hence, $(v_1, \ldots, v_n) \in \psi(S)$.

Static analysis with CQs

problem: **CQ-EQUIVALENCE**

Input: Two CQs ϕ, ψ

Output: Does $\phi(S) = \psi(S)$ holds for every S ? (we write "$\phi \equiv \psi$")

Static analysis with CQs

problem: **CQ-EQUIVALENCE**

Input: Two CQs ϕ, ψ

Output: Does $\phi(S) = \psi(S)$ holds for every S ? (we write "$\phi \equiv \psi$")

Theorem. The equivalence problem for CQ is NP-complete

Static analysis with CQs

problem: **CQ-EQUIVALENCE**

Input: Two CQs ϕ, ψ

Output: Does $\phi(S) = \psi(S)$ holds for every S ? (we write "$\phi \equiv \psi$")

Theorem. The equivalence problem for CQ is NP-complete

$\phi \equiv \psi$ iff

1. $n = m$

2a. There is g: $G_\psi \longrightarrow G_\phi$

2b. There is h: $G_\phi \longrightarrow G_\psi$

3a. $g(y_i) = x_i$ for all i

3b. $h(x_i) = y_i$ for all i

Static analysis with CQs

problem: **CQ-EQUIVALENCE**

Input: Two CQs ϕ, ψ

Output: Does $\phi(S) = \psi(S)$ holds for every S ? (we write "$\phi \equiv \psi$")

Theorem. The equivalence problem for CQ is NP-complete

$\phi \equiv \psi$ iff

1. $n = m$
2a. There is g: $G_\psi \longrightarrow G_\phi$
2b. There is h: $G_\phi \longrightarrow G_\psi$
3a. $g(y_i) = x_i$ for all i
3b. $h(x_i) = y_i$ for all i

Amounts to testing if G_ϕ and G_ψ are **hom-equivalent**
(homomorphisms in both senses)

Static analysis with CQs

Query optimisation: Can I **simplify** the query?

Static analysis with CQs

Query optimisation: Can I **simplify** the query?

problem: **CQ-MINIMIZATION**

Input: A CQ ϕ

Output: Is there a smaller CQ ψ such that $\psi \equiv \phi$?

smaller = with less number of atoms

Static analysis with CQs

problem: **CQ-MINIMIZATION**

Input: A CQ ϕ

Output: Is there a smaller CQ ψ such that $\psi \equiv \phi$?

Static analysis with CQs

problem: **CQ-MINIMIZATION**

Input: A CQ ϕ

Output: Is there a smaller CQ ψ such that $\psi \equiv \phi$?

Theorem. The minimization problem for CQ is NP-complete

Static analysis with CQs

problem: **CQ-MINIMIZATION**

Input: A CQ ϕ

Output: Is there a smaller CQ ψ such that $\psi \equiv \phi$?

Theorem. The minimization problem for CQ is NP-complete

Amounts to testing if there is a smaller structure hom-equivalent to G_ϕ

≈ testing if there is a **non-injective endomorphism**

$$g \colon G_\phi \longrightarrow G_\phi$$

The **smallest structure hom-equivalent** to S is called the **core** of S, and it is unique.

Static analysis with CQs

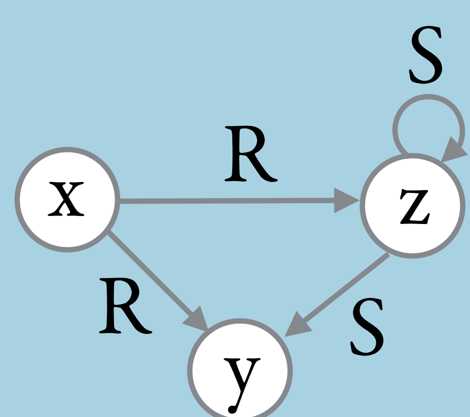
Question:
- Is $\phi = \exists\, x,y,z\ \ R(x,y) \wedge R(x,z) \wedge S(z,z) \wedge S(z,y)$ minimal?
- What is its minimal equivalent query?

Answer:

Static analysis with CQs

Question:
- Is $\phi = \exists\, x,y,z\ \ R(x,y) \wedge R(x,z) \wedge S(z,z) \wedge S(z,y)$ minimal?
- What is its minimal equivalent query?

Answer:



G_ϕ

Static analysis with CQs

Question: • Is $\phi = \exists\, x,y,z\ \ R(x,y) \wedge R(x,z) \wedge S(z,z) \wedge S(z,y)$ minimal?

• What is its minimal equivalent query?

Answer:

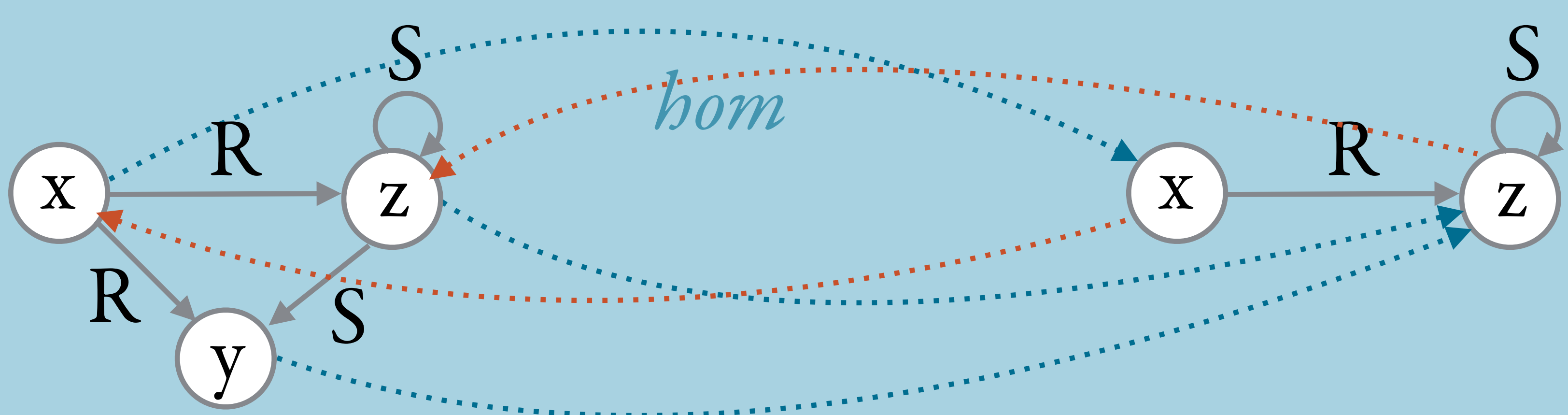
G_ϕ core(G_ϕ)

Static analysis with CQs

Question: • Is $\phi = \exists\, x,y,z\;\; R(x,y) \wedge R(x,z) \wedge S(z,z) \wedge S(z,y)$ minimal?

• What is its minimal equivalent query?

Answer:



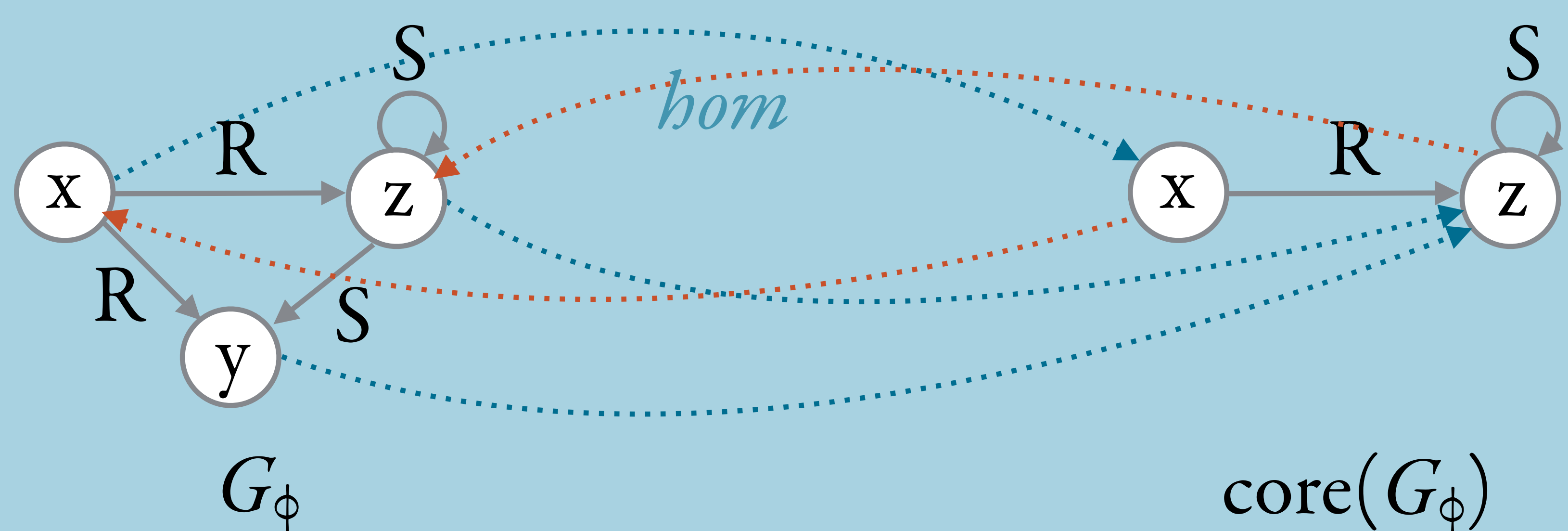
G_ϕ core(G_ϕ)

Static analysis with CQs

Question: • Is $\phi = \exists\, x,y,z\;\; R(x,y) \wedge R(x,z) \wedge S(z,z) \wedge S(z,y)$ minimal?

• What is its minimal equivalent query?

Answer:



No! $\psi = \exists\, x,z\;\; R(x,z) \wedge S(z,z)$ is the minimal query s.t. $\phi \equiv \psi$

Adding functional dependencies

Adding functional dependencies

e.g.

key constraints like

"column *SSN* determines column *Name* in the table Employees"

(component *i*) (component *j*) (relation)

Adding functional dependencies

A **unary functional dependency** is a sentence of the form

$$\forall x_1,\ldots,x_n,y_1,\ldots,y_n \,.\, R(x_1,\ldots,x_n) \land R(y_1,\ldots,y_n) \land (x_i = y_i) \Rightarrow (x_j = y_j)$$

"**R[i⟶j]**" : in relation R the i-th component determines the j-th component

Adding functional dependencies

A **unary functional dependency** is a sentence of the form

$$\forall x_1,\ldots,x_n,y_1,\ldots,y_n \,.\, R(x_1,\ldots,x_n) \wedge R(y_1,\ldots,y_n) \wedge (x_i = y_i) \Rightarrow (x_j = y_j)$$

"**R[i⟶j]**" : in relation R the i-th component determines the j-th component

Example: In the following relation we may enforce the functional dependency

$$\boldsymbol{\gamma} = \forall x, y, z, x', y', z' \;\; R(x,y,z) \wedge R(x',y',z') \wedge (x = x') \Rightarrow (y = y')$$

Agent	Name	Drives
007	James Bond	Aston Martin
200	Mr Smith	Cadillac
201	Mrs Smith	Mercedes
3	Jason Bourne	BMW

Adding functional dependencies

A **unary functional dependency** is a sentence of the form

$$\forall x_1,\ldots,x_n,y_1,\ldots,y_n \,.\, R(x_1,\ldots,x_n) \wedge R(y_1,\ldots,y_n) \wedge (x_i = y_i) \Rightarrow (x_j = y_j)$$

"**R[i⟶j]**" : in relation R the i-th component determines the j-th component

Adding functional dependencies

A **unary functional dependency** is a sentence of the form

$$\forall x_1,\ldots,x_n,y_1,\ldots,y_n \,.\, R(x_1,\ldots,x_n) \land R(y_1,\ldots,y_n) \land (x_i = y_i) \Rightarrow (x_j = y_j)$$

"**R[i⟶j]**" : in relation R the i-th component determines the j-th component

A structure S **verifies** a set of UFD $\{\phi_1, \ldots, \phi_n\}$ if $S \vDash \phi_1 \land \cdots \land \phi_n$.

Adding functional dependencies

A **unary functional dependency** is a sentence of the form

$$\forall x_1,\ldots,x_n,y_1,\ldots,y_n\,.\,R(x_1,\ldots,x_n) \wedge R(y_1,\ldots,y_n) \wedge (x_i = y_i) \Rightarrow (x_j = y_j)$$

"**R[i⟶j]**" : in relation R the i-th component determines the j-th component

All the previous problems:

- CQ-CONTAINMENT
- CQ-EQUIVALENCE
- CQ-MINIMIZATION

remain in NP if we further restrict finite structures so as to satisfy any set of functional dependencies

Adding functional dependencies

A **unary functional dependency** is a sentence of the form

$$\forall\, x_1,\ldots,x_n,y_1,\ldots,y_n \,.\, R(x_1,\ldots,x_n) \wedge R(y_1,\ldots,y_n) \wedge (x_i = y_i) \Rightarrow (x_j = y_j)$$

"**R[i⟶j]**" : in relation R the i-th component determines the j-th component

All the previous problems:

Modify the canonical structure G_ϕ ...

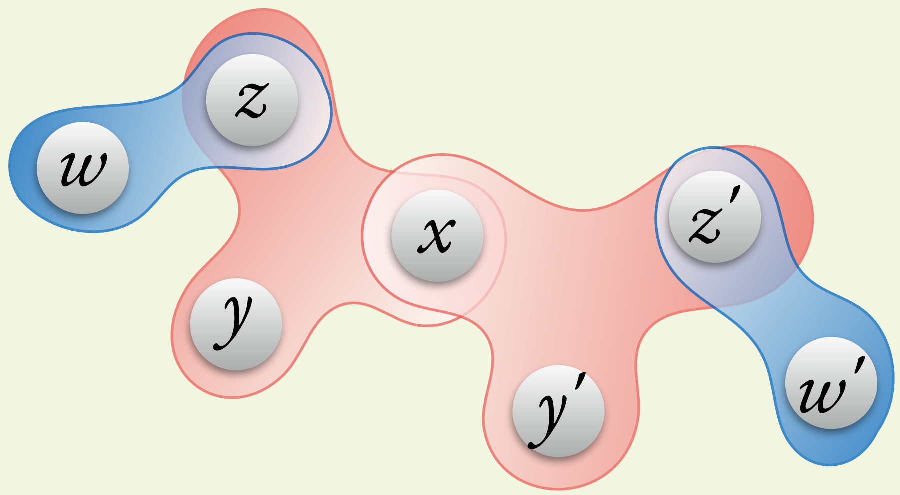
- CQ-CONTAINMENT
- CQ-EQUIVALENCE
- CQ-MINIMIZATION

remain in NP if we further restrict finite structures so as to satisfy any set of functional dependencies

Adding functional dependencies

CQ $\phi = R_2(x, y, z) \wedge R_2(x, y', z') \wedge R_1(z, w) \wedge R_1(z', w')$

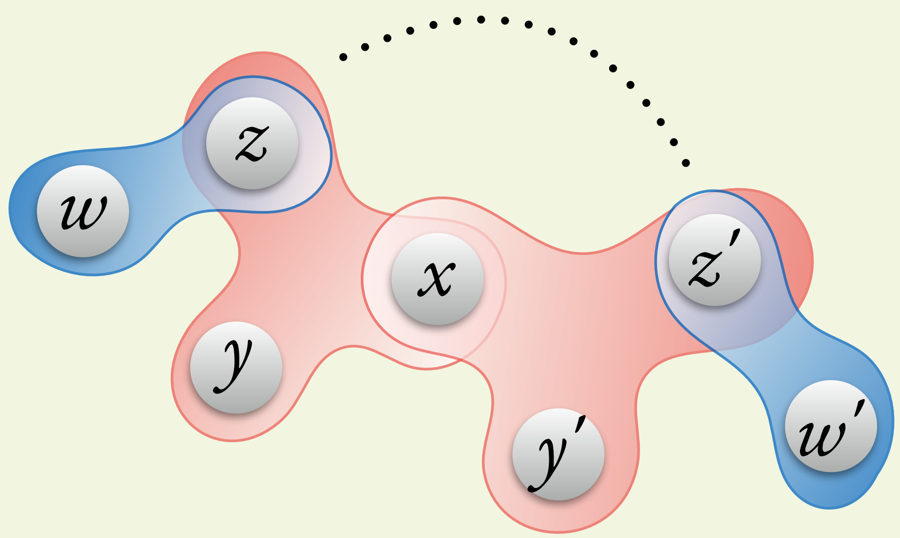
under functional dependencies $F=\{\, R_1[1 \longrightarrow 2],\ R_2\!: [1 \longrightarrow 3]\,\}$



Adding functional dependencies

CQ $\phi = R_2(x, y, z) \wedge R_2(x, y', z') \wedge R_1(z, w) \wedge R_1(z', w')$

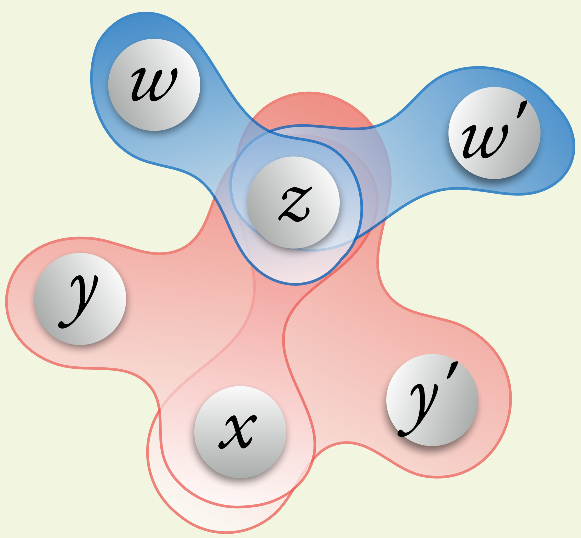
under functional dependencies $F = \{ R_1[1 \longrightarrow 2],\ R_2{:}\ [1 \longrightarrow 3] \}$



Adding functional dependencies

CQ $\phi = R_2(x, y, z) \wedge R_2(x, y', z') \wedge R_1(z, w) \wedge R_1(z', w')$

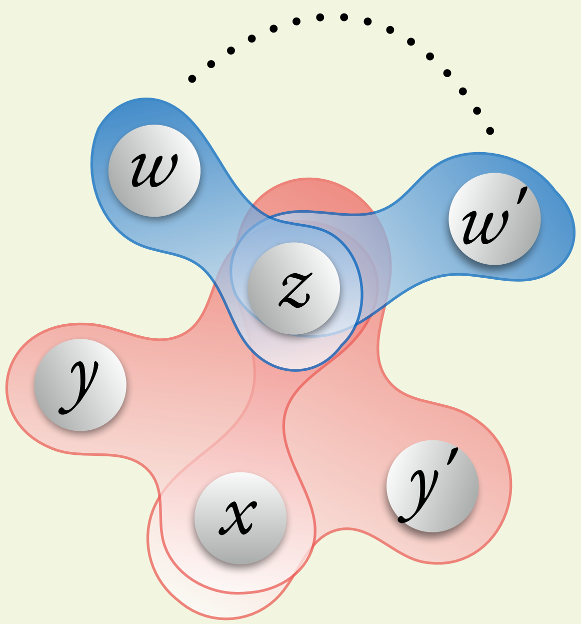
under functional dependencies F={ $R_1[1 \longrightarrow 2]$, R_2: $[1 \longrightarrow 3]$ }



Adding functional dependencies

CQ $\phi = R_2(x,y,z) \wedge R_2(x,y',z') \wedge R_1(z,w) \wedge R_1(z',w')$

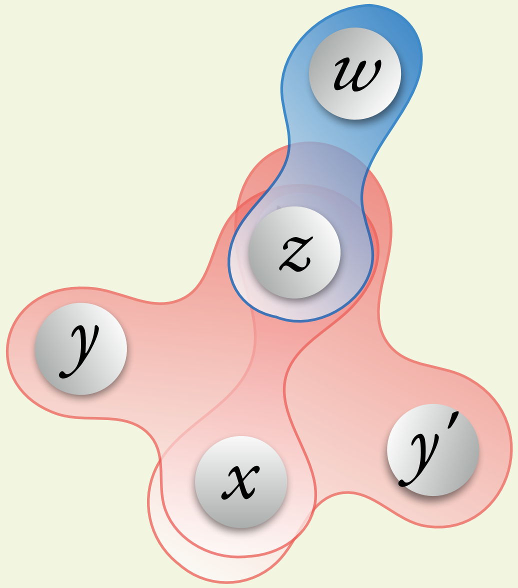
under functional dependencies $F = \{ R_1[1 \longrightarrow 2],\ R_2\!: [1 \longrightarrow 3] \}$



Adding functional dependencies

CQ $\phi = R_2(x, y, z) \wedge R_2(x, y', z') \wedge R_1(z, w) \wedge R_1(z', w')$

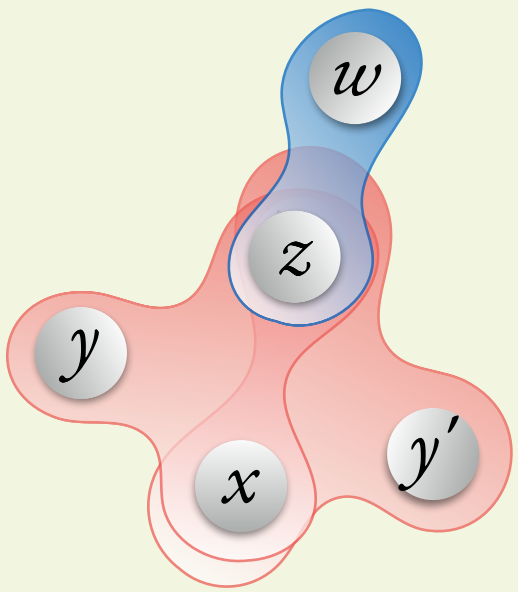
under functional dependencies $F = \{ R_1[1 \longrightarrow 2],\ R_2\!: [1 \longrightarrow 3] \}$



Adding functional dependencies

CQ $\phi = R_2(x, y, z) \wedge R_2(x, y', z') \wedge R_1(z, w) \wedge R_1(z', w')$

under functional dependencies $F = \{\, R_1[1 \longrightarrow 2],\ R_2{:}\ [1 \longrightarrow 3] \,\}$



$= \text{chase}_F(G_\phi)$ (the **chased canonical structure**)

- $\text{chase}_F(G_\phi)$ is unique and can be constructed in polynomial time

Adding functional dependencies

$\phi \in CQ$
FD's $F=\{fd_1, \ldots, fd_n\}$

chase →

$chase_F(\phi) \in CQ$

Adding functional dependencies

$\phi \in CQ$
FD's $F=\{fd_1, \ldots, fd_n\}$ —chase→ $chase_F(\phi) \in CQ$

The static analysis problems restricted to FD's can now be also shown in NP

- CQ-Containment: $\phi \subseteq_F \psi$ iff $chase_F(\phi) \subseteq chase_F(\psi)$
- CQ-Equivalence: $\phi \equiv_F \psi$ iff $chase_F(\phi) \equiv chase_F(\psi)$
- CQ-Minimization: ϕ is minimal wrt structures verifying F iff $chase_F(\phi)$ is minimal

Acyclic CQ's : Definition

On graphs: CQ ϕ is **acyclic** if G_{ϕ} is tree-like

Acyclic CQ's : Definition

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

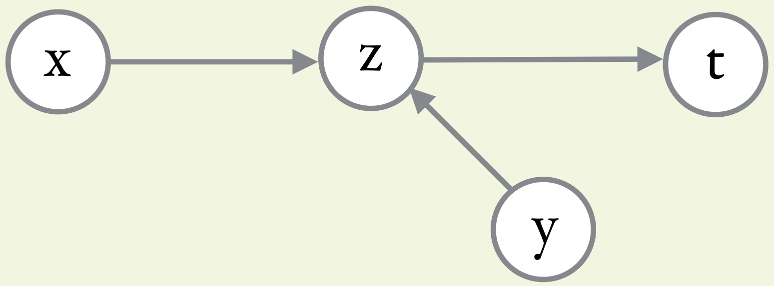
underlying undirected graph is acyclic

Acyclic CQ's : Definition

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

underlying undirected graph is acyclic

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



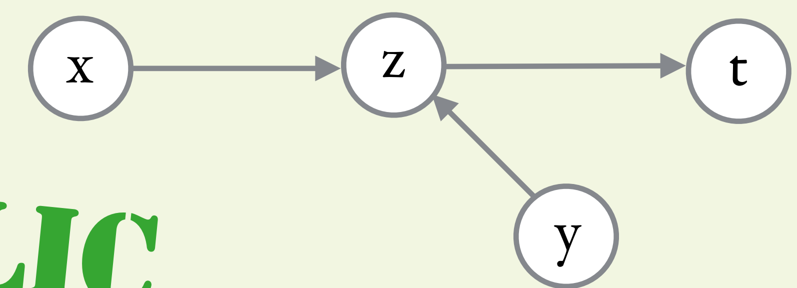
Acyclic CQ's : Definition

underlying undirected graph is acyclic

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

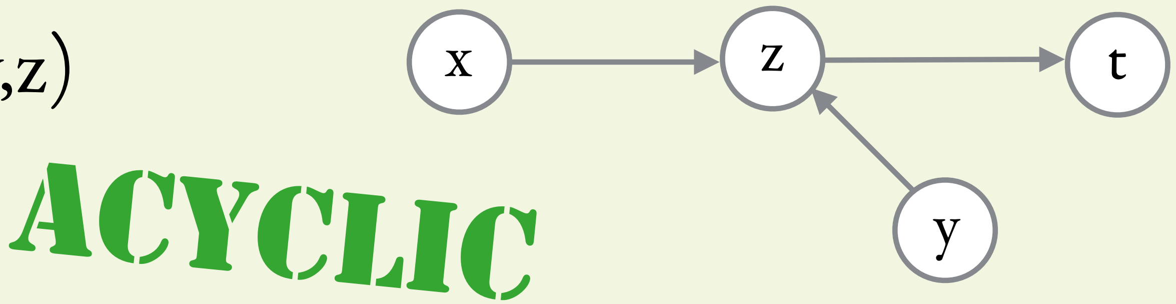


Acyclic CQ's : Definition

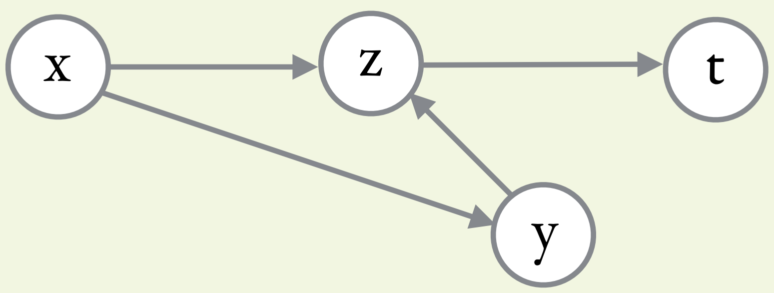
underlying undirected graph is acyclic

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

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



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

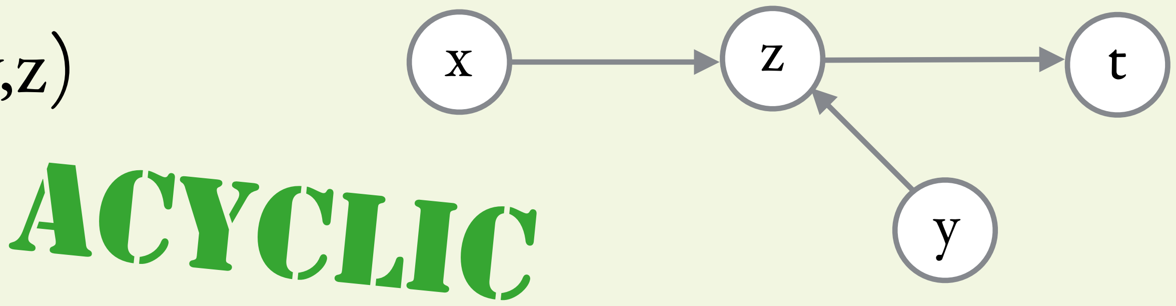


Acyclic CQ's : Definition

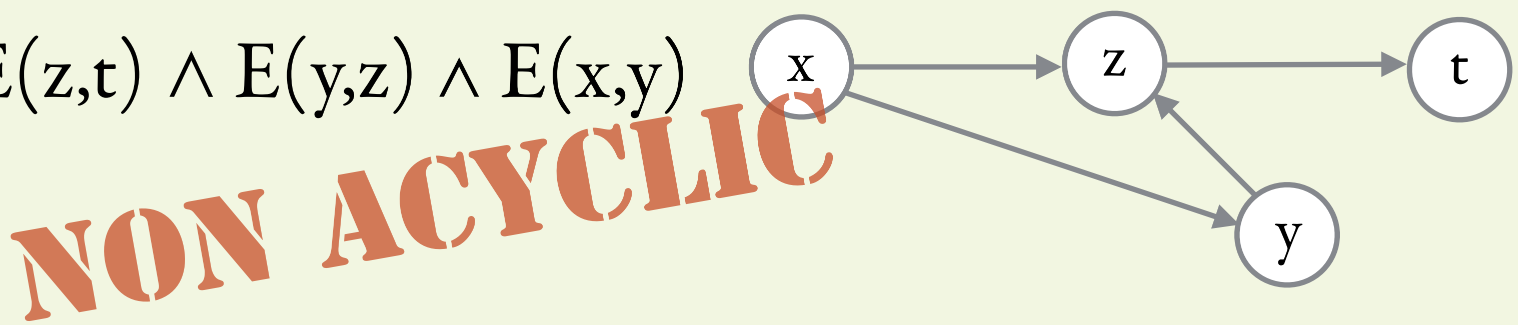
On graphs: CQ ϕ is acyclic if G_ϕ is tree-like

underlying undirected graph is acyclic

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



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



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

On general 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

On general 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 st:

- 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

On general 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 st:

- 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

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

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

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

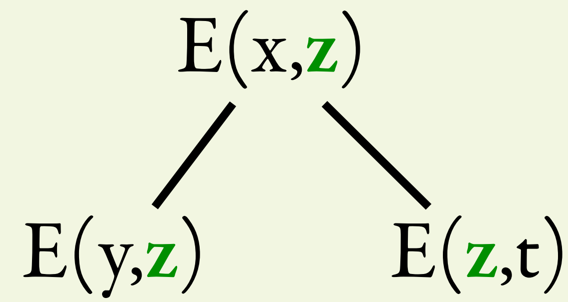
If x occurs in two nodes, then it occurs in the path linking the two nodes.

A **join tree** is a tree T st:

- 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

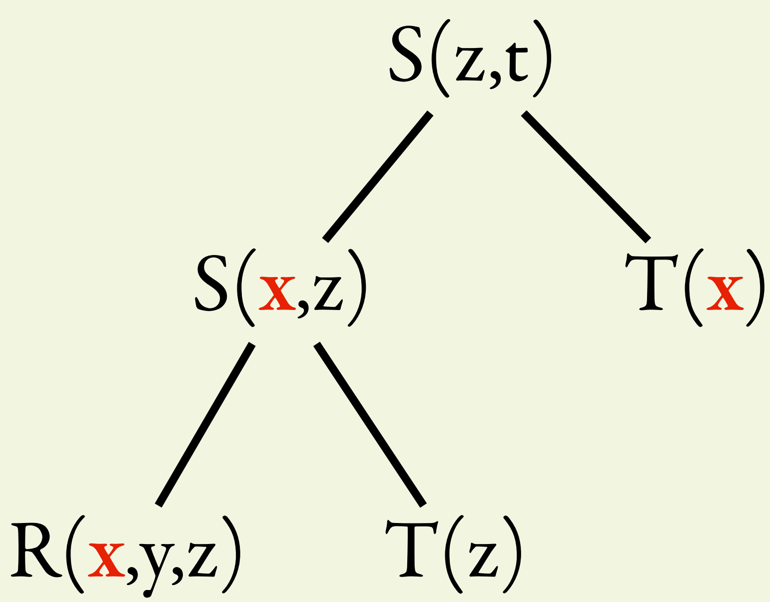
Acyclic CQ's

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

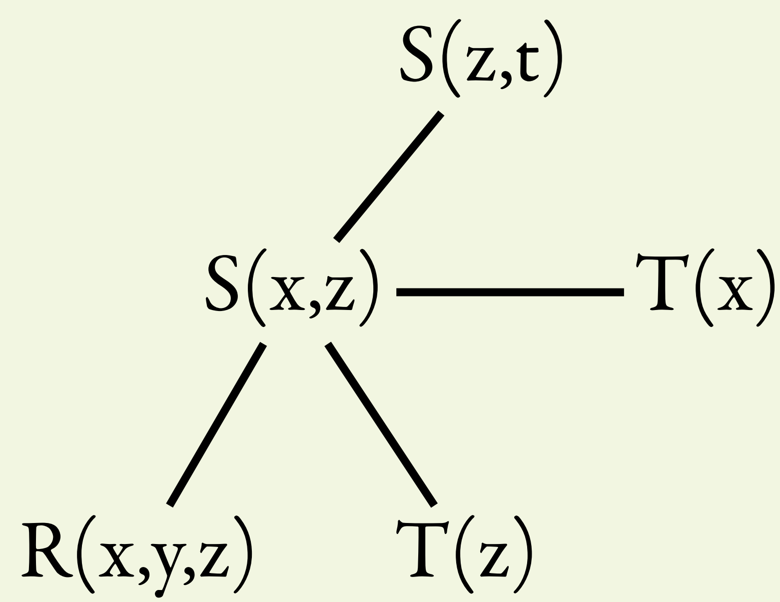


join tree

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

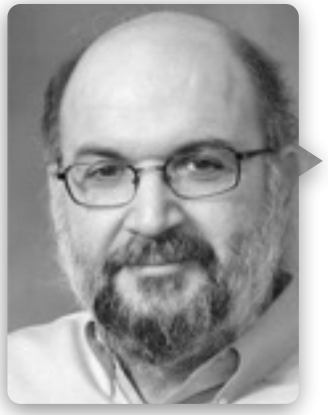


not a join tree



a join tree

Acyclic CQ's



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

[Yannakakis]

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