



# Logical foundations of databases

Diego Figueira

Gabriele Puppis

CNRS LaBRI



# Recap

- Active domain semantics and expressiveness:  $\text{FO}_{\text{act}} =^* \text{RA}$
- Undecidable problems ( $\text{Halting} \leq \text{Domino} \leq \text{FO-Satisfiability} \leq \text{FO-Equivalence}$ )
- 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: “ $\exists x_1, \dots, x_n . \phi(x_1, \dots, x_n)$ ”

..... 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  $\forall, \neg, \vee$**

Normal form: “ $\exists x_1, \dots, x_n. \phi(x_1, \dots, x_n)$ ”

..... 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

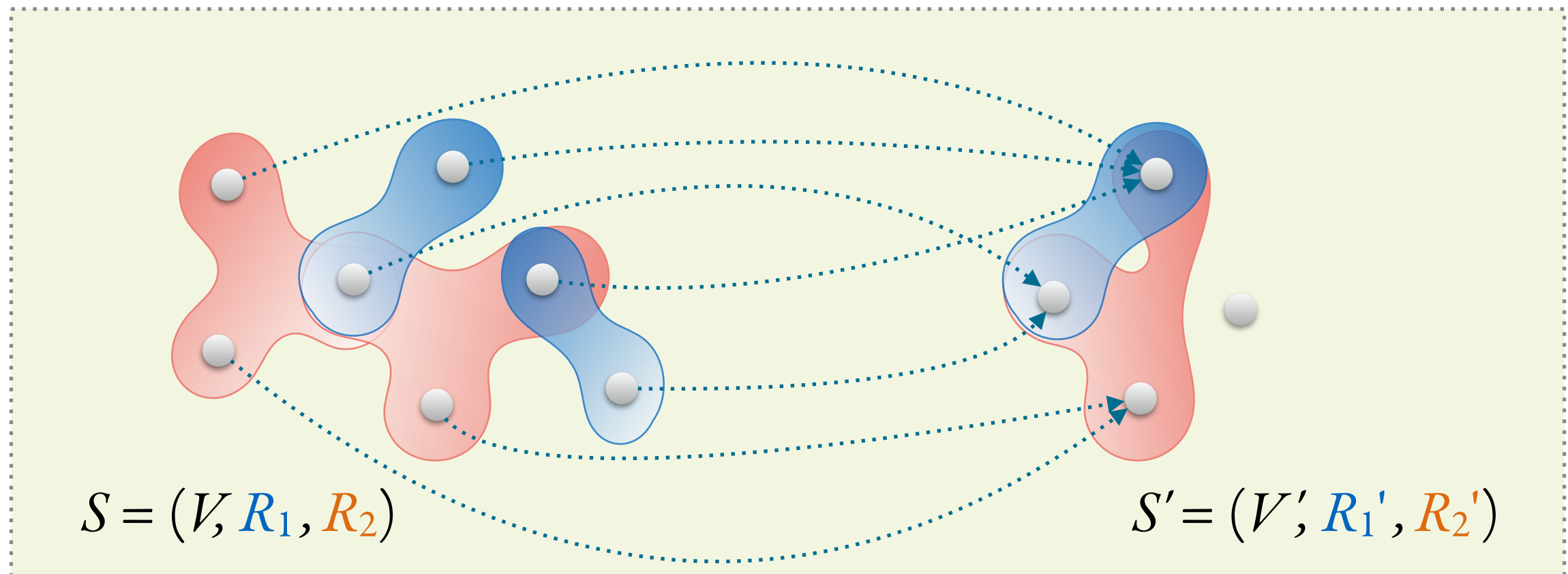
It corresponds to “ $\pi$ - $\sigma$ - $\times$ ” RA queries

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

..... no negation

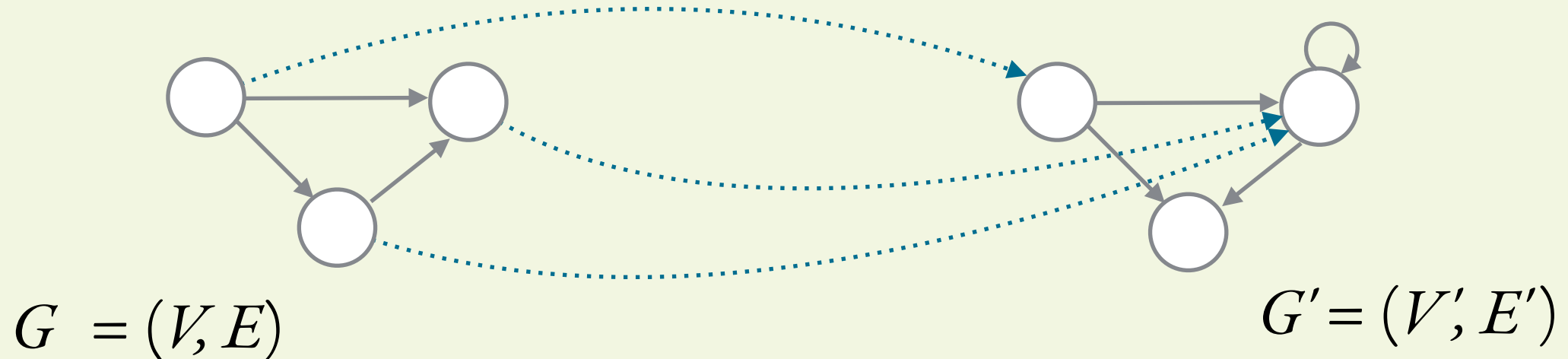
# Homomorphisms

**Homomorphism** between structures  $S=(V, R_1, \dots, R_n)$  and  $S'=(V', R_1', \dots, R_n')$  is a function  $h : V \longrightarrow V'$  such that

$$(x_1, \dots, x_n) \in R_i \text{ implies } (h(x_1), \dots, h(x_n)) \in R_i'$$


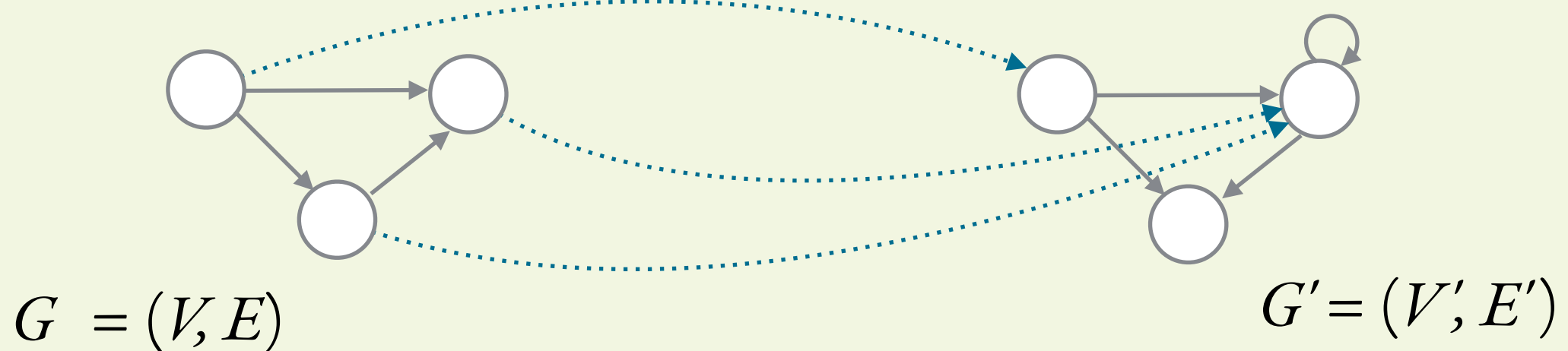
# Homomorphisms

**Homomorphism** between structures  $S=(V, R_1, \dots, R_n)$  and  $S'=(V', R_1', \dots, R_n')$  is a function  $h : V \longrightarrow V'$  such that

$$(x_1, \dots, x_n) \in R_i \text{ implies } (h(x_1), \dots, h(x_n)) \in R_i'$$


# Canonical structures

- Canonical structure**  $G_\phi$  of a Conjunctive Query  $\phi$  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  $\phi$

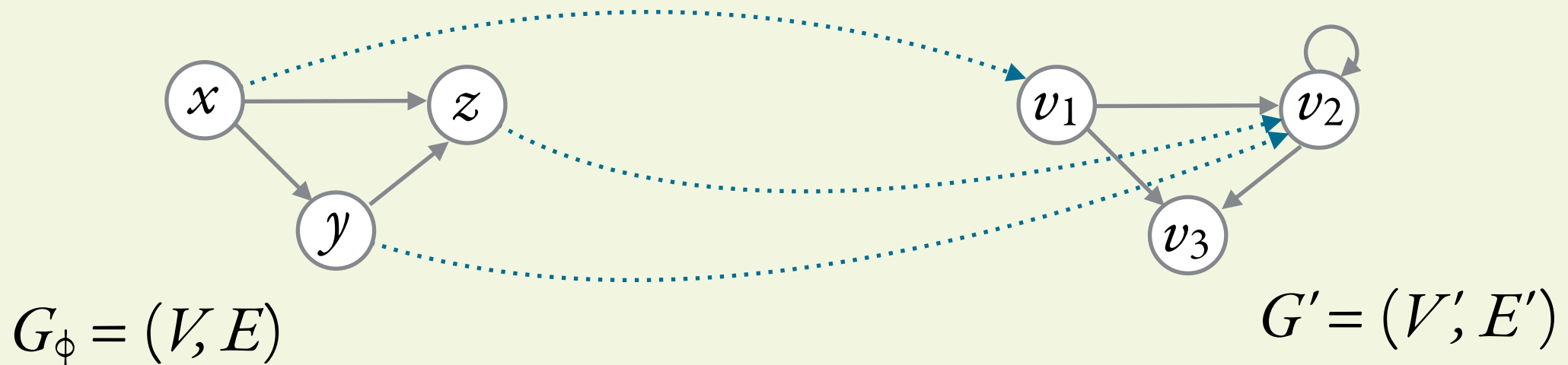




# Canonical structures

- Canonical structure**  $G_\phi$  of a Conjunctive Query  $\phi$  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  $\phi$

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



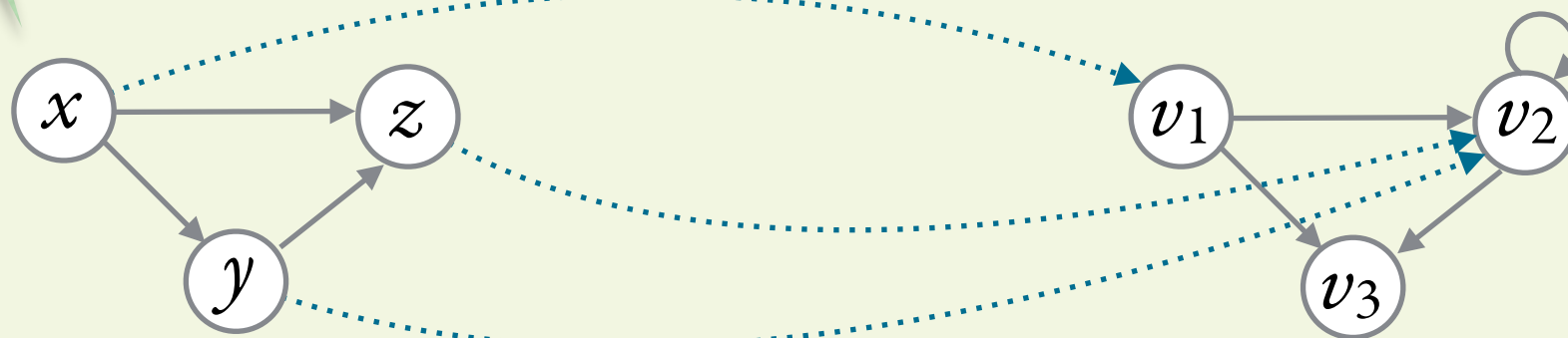
# Canonical structures

**Canonical structure**  $G_\phi$  of a Conjunctive Query  $\phi$  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  $\phi$

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

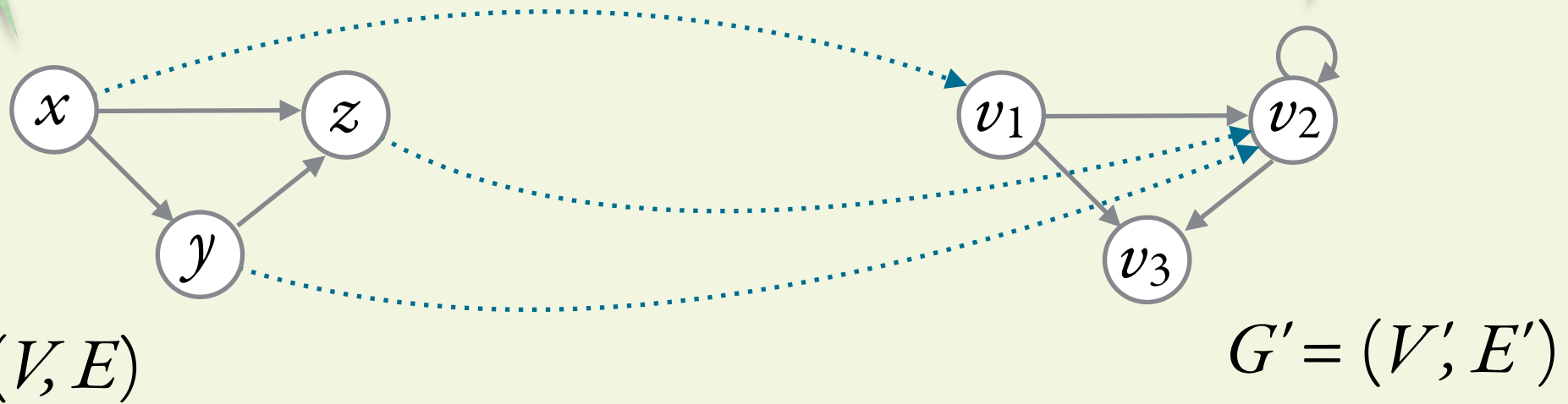
**Canonical structure**  $G_\phi$  of a Conjunctive Query  $\phi$  has

- 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  $\phi$

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

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

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



# Canonical structures

**Canonical structure**  $G_\phi$  of a Conjunctive Query  $\phi$  has

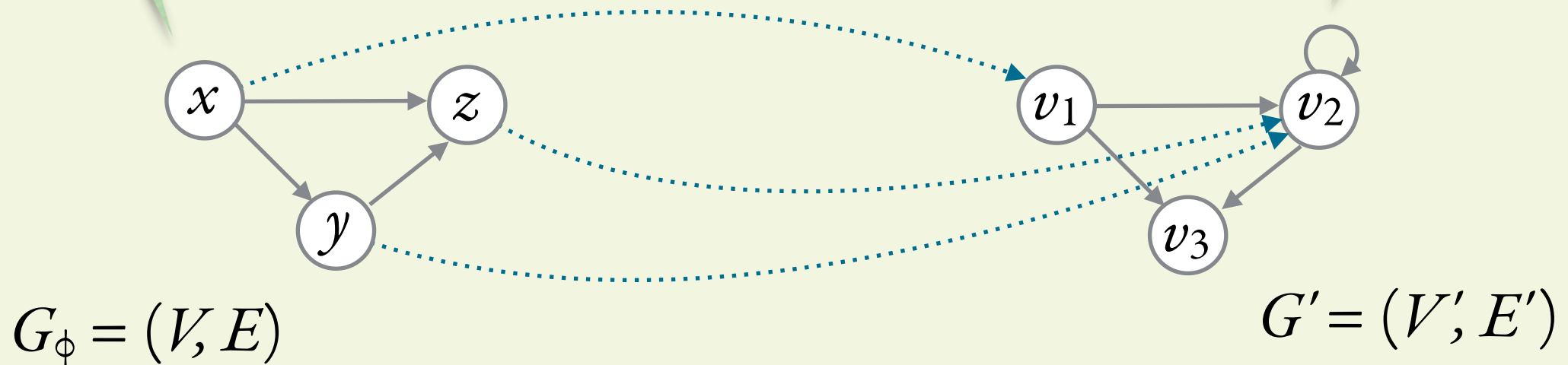
- variables as nodes
- tuples  $(x_1, \dots, x_n) \in R_i$   
for all atomic sub-formulas  $R_i(x_1, \dots, x_n)$  of  $\phi$

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

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

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

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

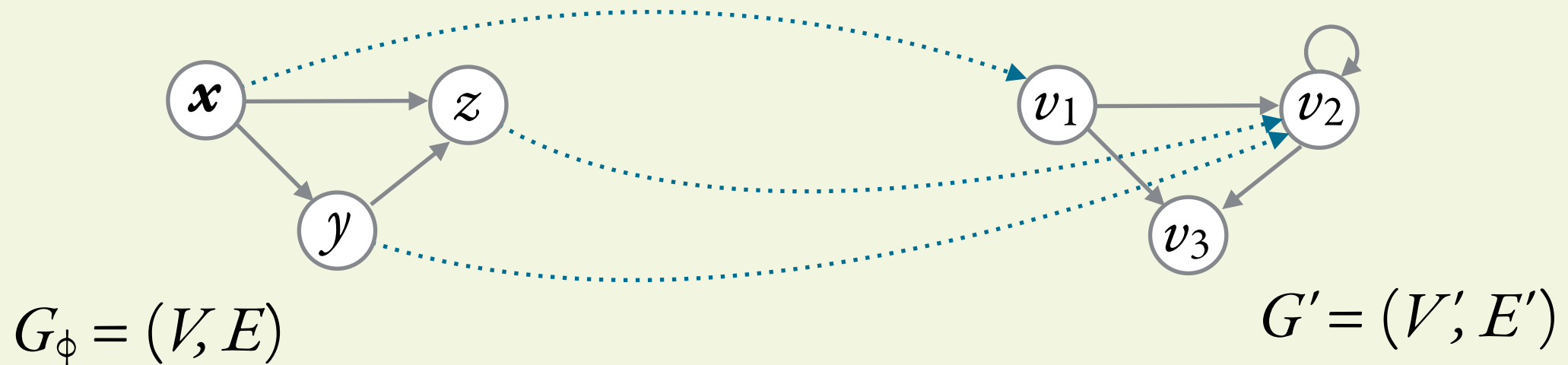


# Evaluation via homomorphisms

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

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

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



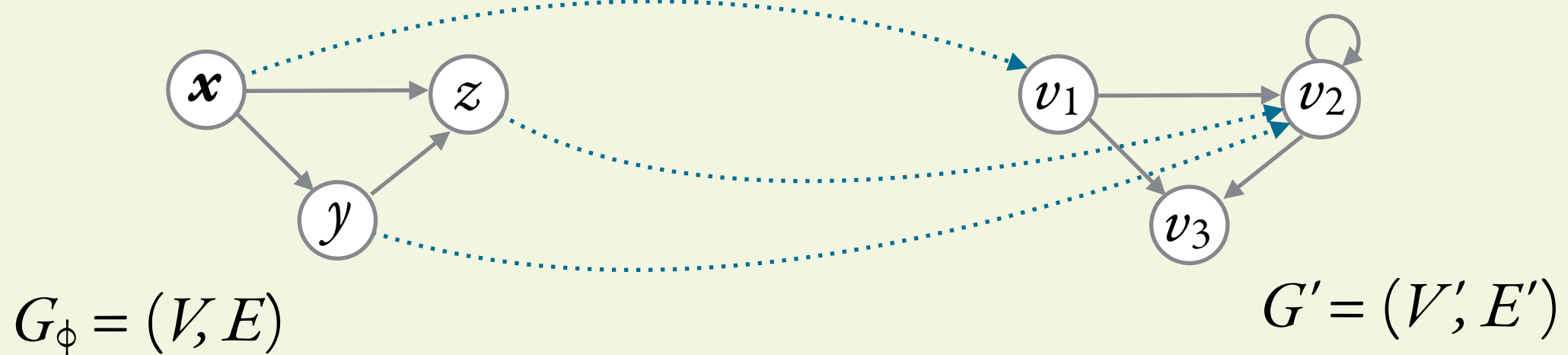
# Evaluation via homomorphisms

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

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

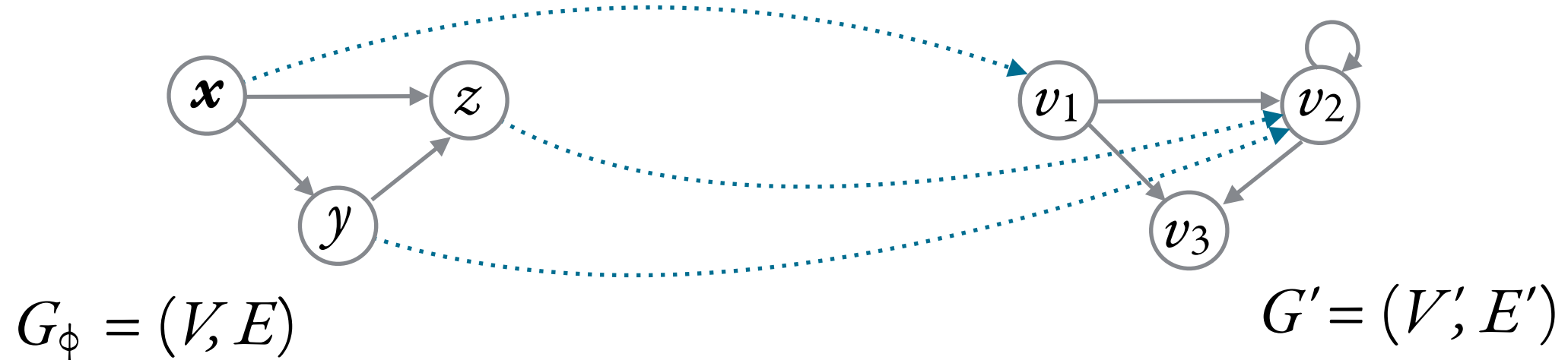
**Question:** Which are the homomorphisms  $G_\phi \longrightarrow G'$  ?  
What is the result of  $\phi(G')$  ?

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)

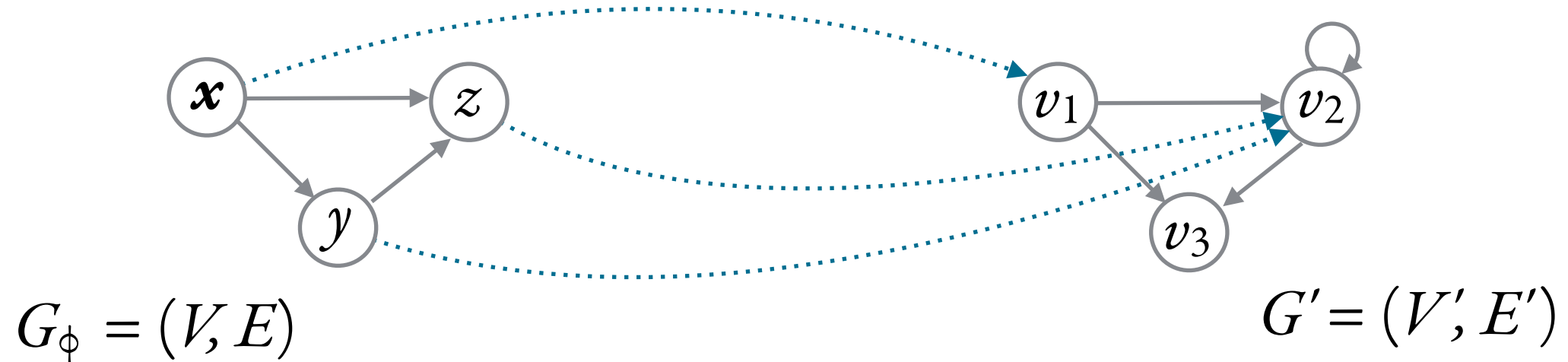


**Input:** A CQ  $\phi(x_1, \dots, x_n)$ , a graph  $G$ , a tuple  $(a_1, \dots, a_n)$

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

# Evaluation via homomorphisms

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



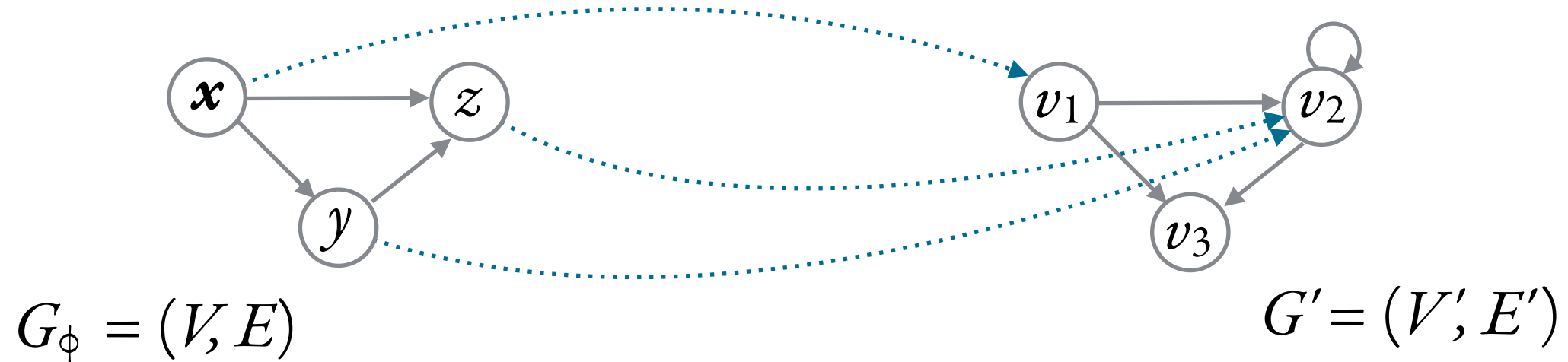
**Input:** A CQ  $\phi(x_1, \dots, x_n)$ , a graph  $G$ , a tuple  $(a_1, \dots, a_n)$   
**Output:** Is  $(a_1, \dots, a_n) \in \phi(G)$  ?

Ideas?



# Evaluation via homomorphisms

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



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

Ideas?

1. Guess  $h: G_\phi \longrightarrow G$
2. Check that it is a homomorphism
3. Output YES if  $(h(x_1), \dots, h(x_n)) = (a_1, \dots, a_n)$ ; 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)

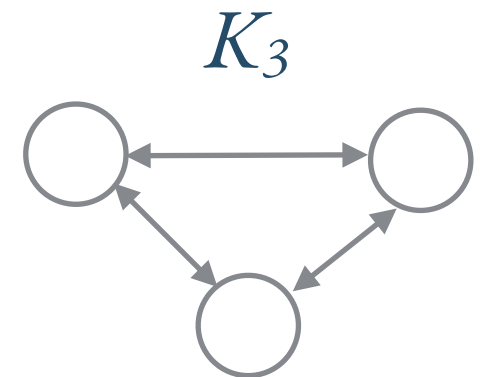
NP-complete problem: **3-COLORABILITY**

**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  $K_3$  ?



# Evaluation via homomorphisms

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

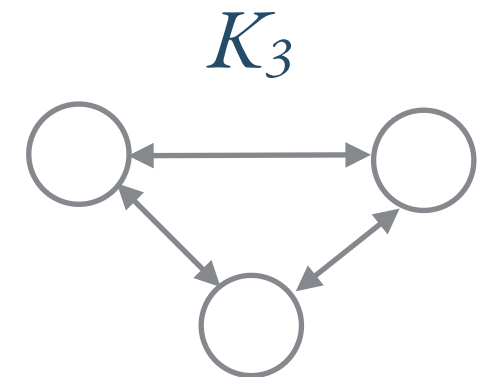
NP-complete problem: **3-COLORABILITY**

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

# Monotonicity and preservation theorems

**Lemma.** Every CQ is **monotone**:

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

# Monotonicity and preservation theorems

**Lemma.** Every CQ is **monotone**:

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

Proof:

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

# Monotonicity and preservation theorems

**Lemma.** Every CQ is **monotone**:

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

Proof:

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



“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  $\phi$  is **monotone**  
then  $\phi \in \text{UCQ}$  [Rossman '08]



# Monotonicity and preservation theorems

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

Finite unions of CQs  
 $\approx \exists \wedge \vee$  fragment of FO

# Monotonicity and preservation theorems

**Theorem.** If an FO query  $\phi$  is **monotone**  
then  $\phi \in \text{UCQ}$  [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  $\phi$  is **monotone**  
then  $\phi \in \text{UCQ}$  [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  $\phi, \psi$

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

# Static analysis with CQs

problem: **CQ-CONTAINMENT**

**Input:** Two CQs  $\phi, \psi$

**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  $\phi, \psi$

**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  $\phi, \psi$

**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  $\phi, \psi$

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

**Theorem.** The containment problem for CQ is NP-complete

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

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

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  $\phi, \psi$

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

**Theorem.** The containment problem for CQ is NP-complete

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

Why?

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

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

**Question:** Is this combined or data complexity?

**Answer:** None!

# Static analysis with CQs

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

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

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

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

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

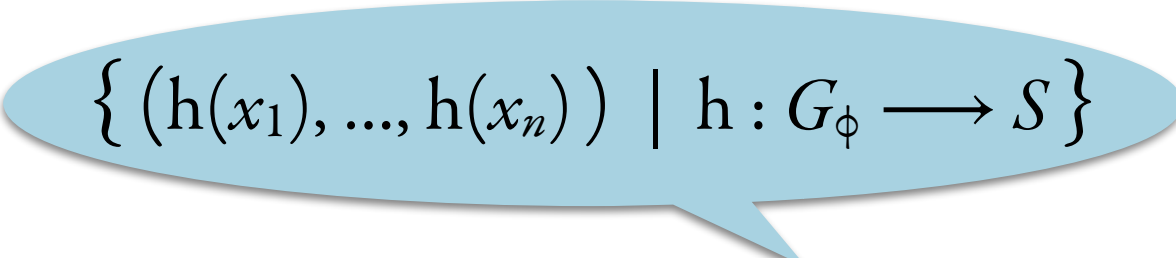
# Static analysis with CQs

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

1.  $n = m$

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

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


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

$\Rightarrow$  Suppose  $\forall S \quad \phi(S) \subseteq \psi(S)$

# Static analysis with CQs

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

1.  $n = m$

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

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

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

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

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

# Static analysis with CQs

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

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

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

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

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

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

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

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



# Static analysis with CQs

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

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

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

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

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

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

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

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

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

# Static analysis with CQs

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

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

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

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

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

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

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

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

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

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

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

# Static analysis with CQs

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

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

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

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

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

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

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

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

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

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

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

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

# Static analysis with CQs

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

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

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

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

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

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

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

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

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

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

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

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

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

# Static analysis with CQs

problem: **CQ-EQUIVALENCE**

**Input:** Two CQs  $\phi, \psi$

**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  $\phi, \psi$

**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  $\phi, \psi$

**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  $\phi, \psi$

**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  $G_\phi$  and  $G_\psi$  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  $\phi$

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

*smaller* = with less number of atoms

# Static analysis with CQs

problem: **CQ-MINIMIZATION**

**Input:** A CQ  $\phi$

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

# Static analysis with CQs

problem: **CQ-MINIMIZATION**

**Input:** A CQ  $\phi$

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

**Theorem.** The minimization problem for CQ is NP-complete

# Static analysis with CQs

problem: **CQ-MINIMIZATION**

**Input:** A CQ  $\phi$

**Output:** Is there a smaller CQ  $\psi$  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  $G_\phi$

$\approx$  testing if there is a **non-injective endomorphism**

$$g: G_\phi \longrightarrow G_\phi$$

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?

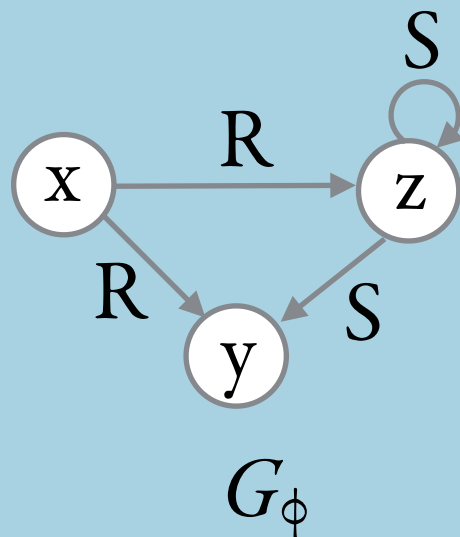
**Answer:**

# 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:**

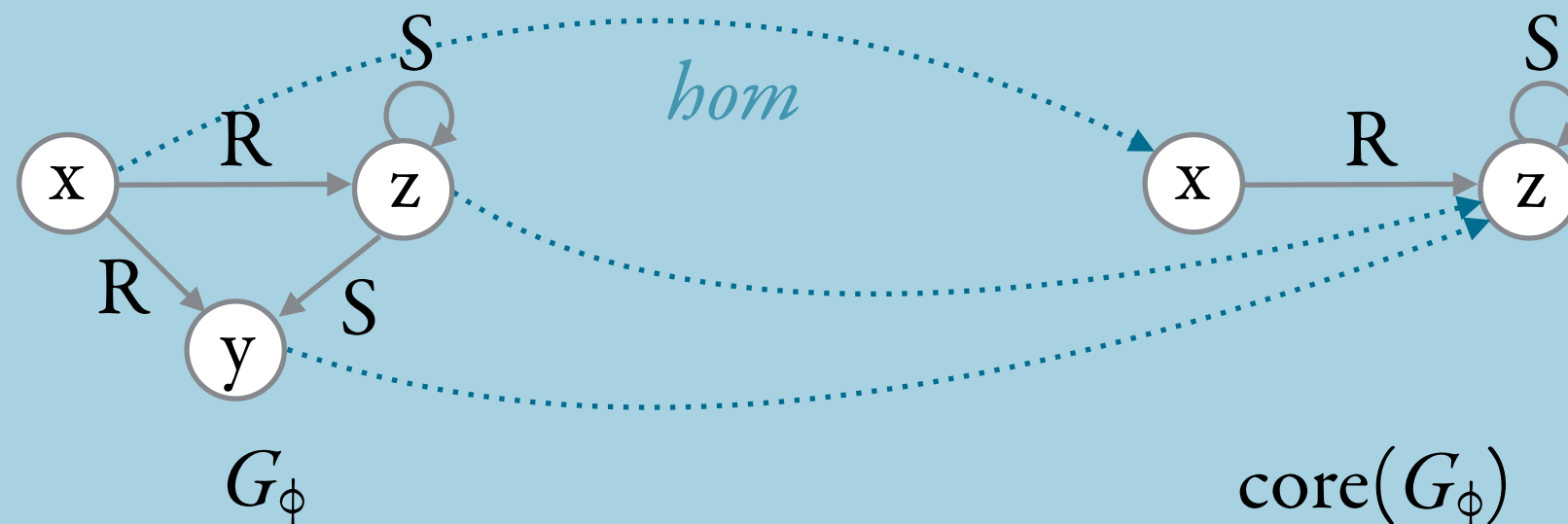


# Static analysis with CQs

**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:**



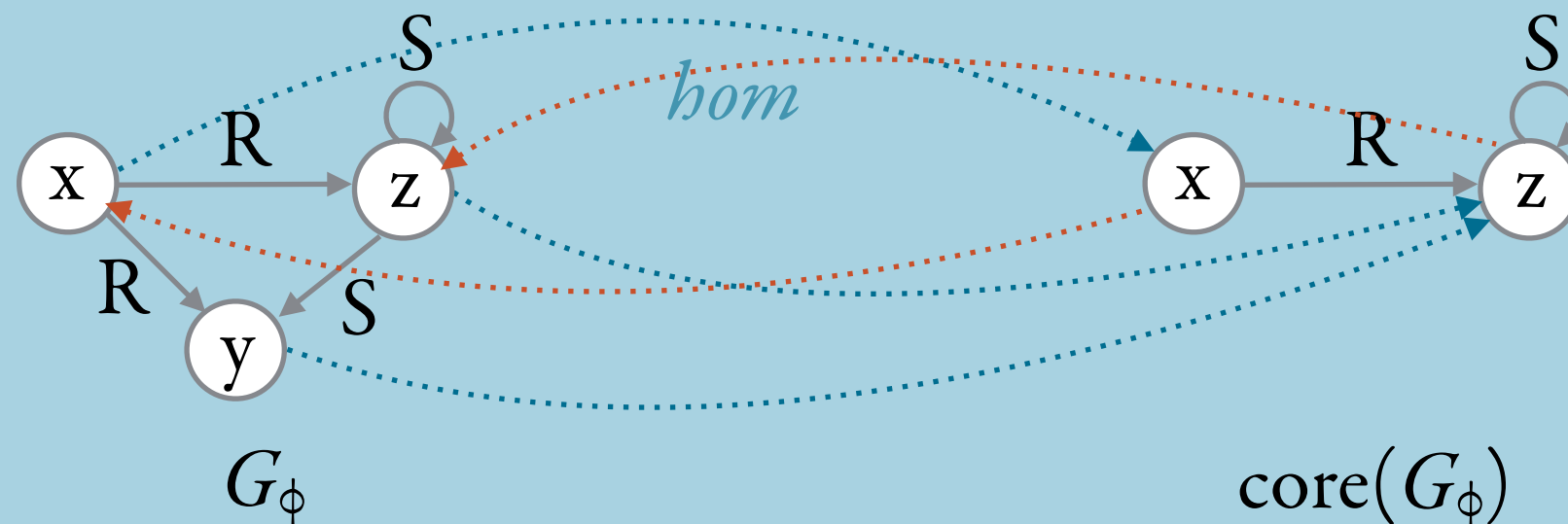


# Static analysis with CQs

**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:**

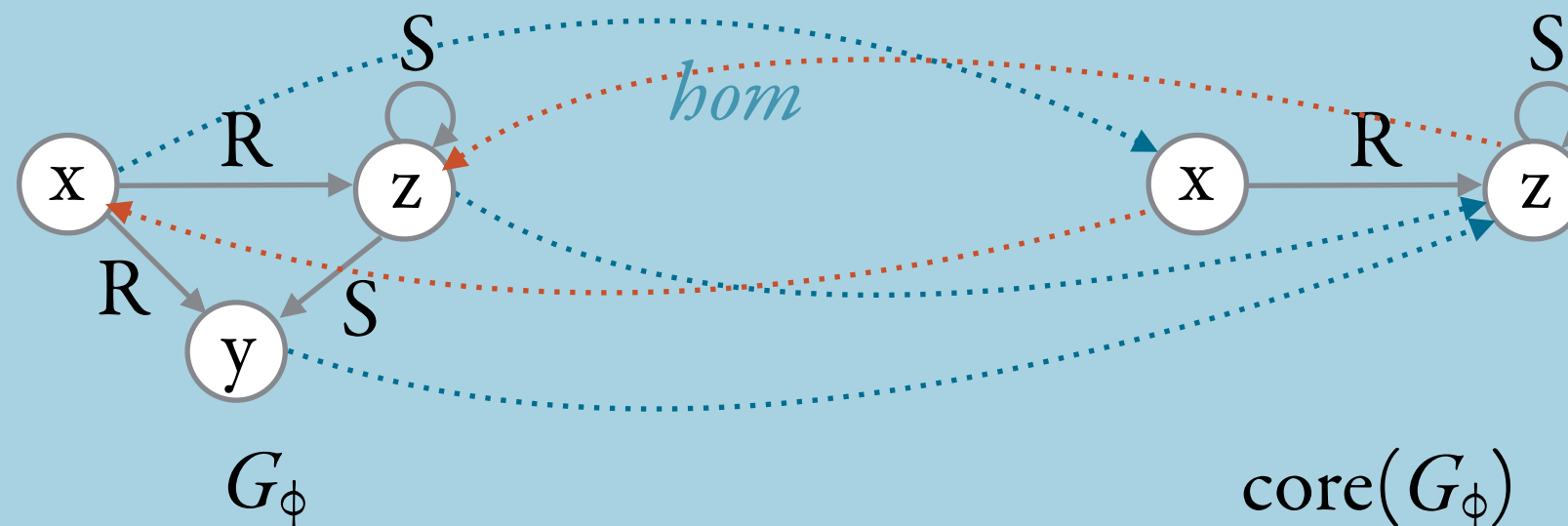


# Static analysis with CQs

**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

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

"**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

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

"**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

$$\gamma = \forall x, y, z, x', y', z' . R(x, y, z) \wedge R(x', y', z') \wedge (x = x') \Rightarrow (y = y')$$

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

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

"**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

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

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

A structure  $S$  *verifies* a set of UFD  $\{\phi_1, \dots, \phi_n\}$  if  $S \models \phi_1 \wedge \dots \wedge \phi_n$ .



# Adding functional dependencies

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

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

"**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

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

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

All the previous problems:



Modify the canonical structure  $G_\phi \dots$

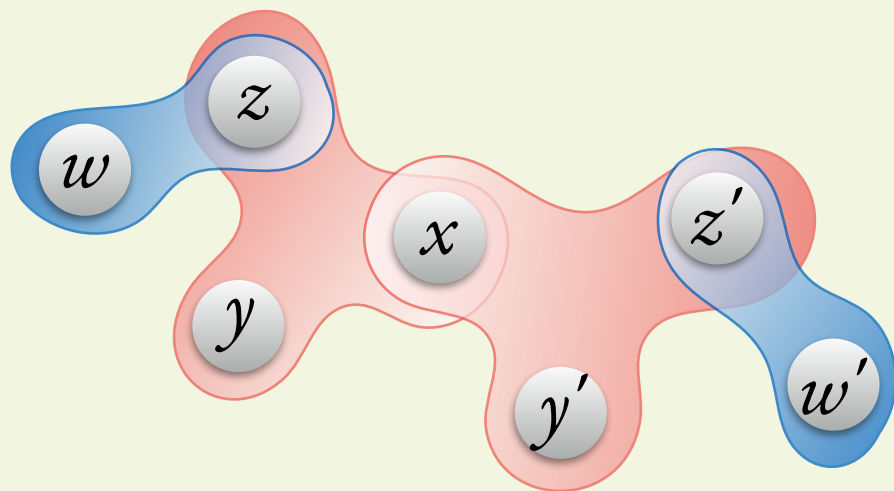
- CQ-CONTAINMENT
- CQ-EQUIVALENCE
- CQ-MINIMIZATION

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

# Adding functional dependencies

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

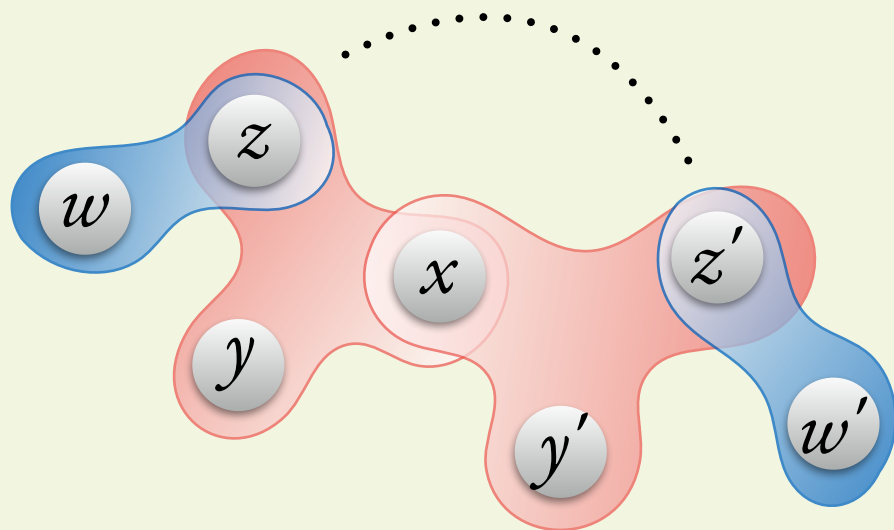
under functional dependencies  $F = \{ R_1[1 \longrightarrow 2], R_2: [1 \longrightarrow 3] \}$



# Adding functional dependencies

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

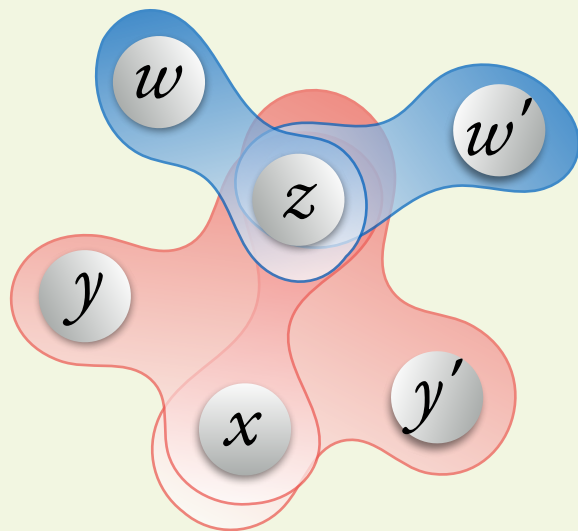
under functional dependencies  $F = \{ R_1[1 \longrightarrow 2], R_2: [1 \longrightarrow 3] \}$



# Adding functional dependencies

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

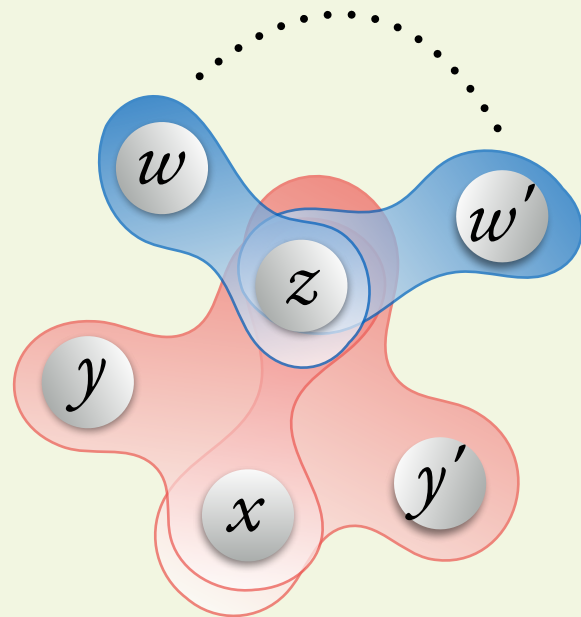
under functional dependencies  $F = \{ R_1[1 \longrightarrow 2], R_2: [1 \longrightarrow 3] \}$



# Adding functional dependencies

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

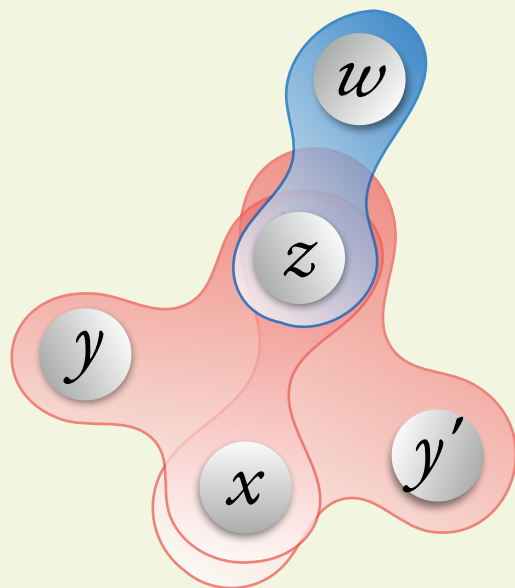
under functional dependencies  $F = \{ R_1[1 \longrightarrow 2], R_2: [1 \longrightarrow 3] \}$



# Adding functional dependencies

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

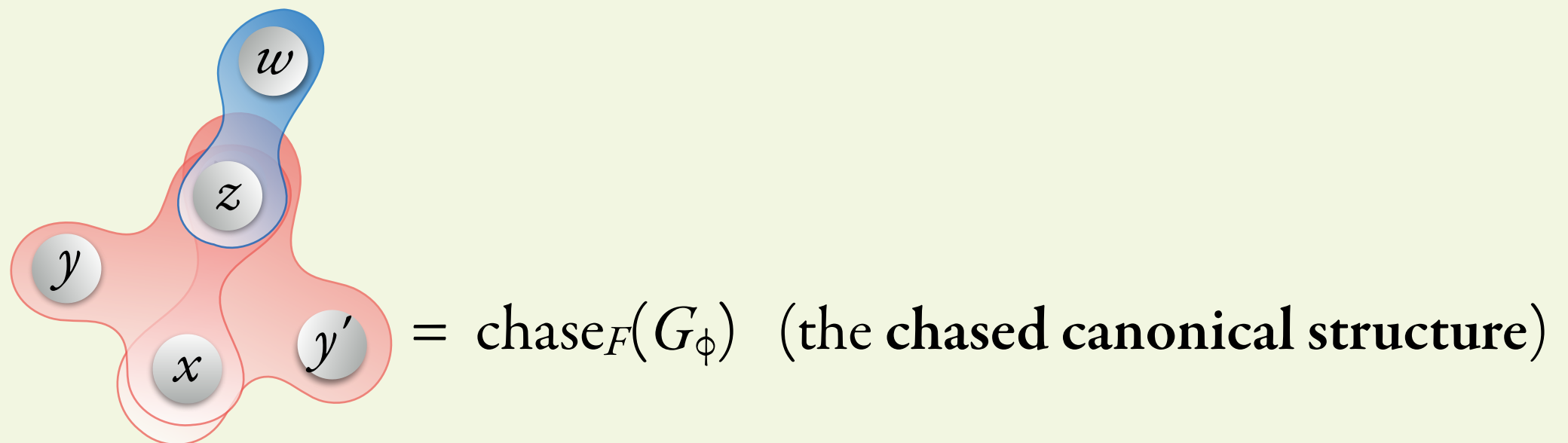
under functional dependencies  $F = \{ R_1[1 \longrightarrow 2], R_2: [1 \longrightarrow 3] \}$



# Adding functional dependencies

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

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



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



# Adding functional dependencies

$$\begin{array}{l} \phi \in \text{CQ} \\ \text{FD's } F = \{fd_1, \dots, fd_n\} \end{array} \xrightarrow{\text{chase}} \text{chase}_F(\phi) \in \text{CQ}$$

# Adding functional dependencies

$$\begin{array}{c} \phi \in \text{CQ} \\ \text{FD's } F = \{fd_1, \dots, fd_n\} \end{array} \xrightarrow{\text{chase}} \text{chase}_F(\phi) \in \text{CQ}$$

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

- **CQ-Containment**       $\phi \subseteq_F \psi$  iff  $\text{chase}_F(\phi) \subseteq \text{chase}_F(\psi)$
- **CQ-Equivalence**       $\phi \equiv_F \psi$  iff  $\text{chase}_F(\phi) \equiv \text{chase}_F(\psi)$
- **CQ-Minimization**       $\phi$  is minimal wrt  
structures verifying  $F$  iff  $\text{chase}_F(\phi)$  is minimal

# Acyclic CQ's : Definition

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

# Acyclic CQ's : Definition

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

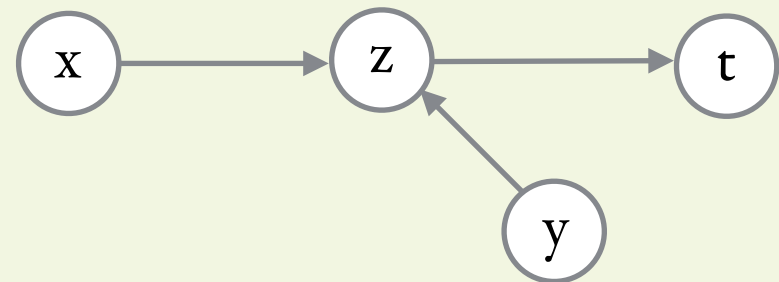
underlying  
undirected graph is  
acyclic

# Acyclic CQ's : Definition

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

underlying  
undirected graph is  
acyclic

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

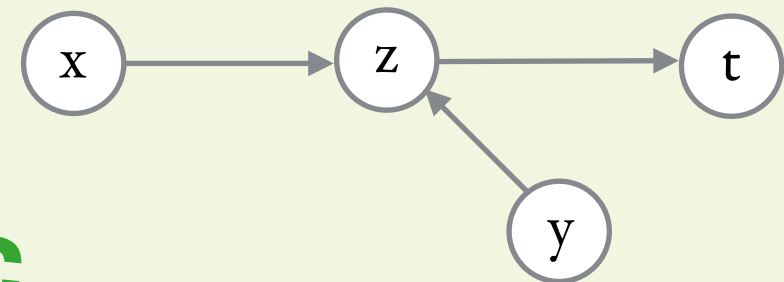


# Acyclic CQ's : Definition

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

underlying  
undirected graph is  
acyclic

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



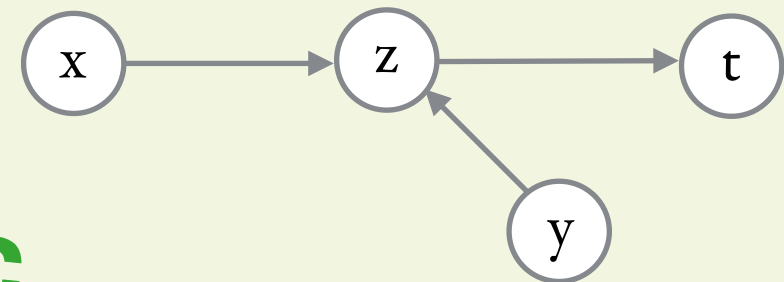
**acyclic**

# Acyclic CQ's : Definition

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

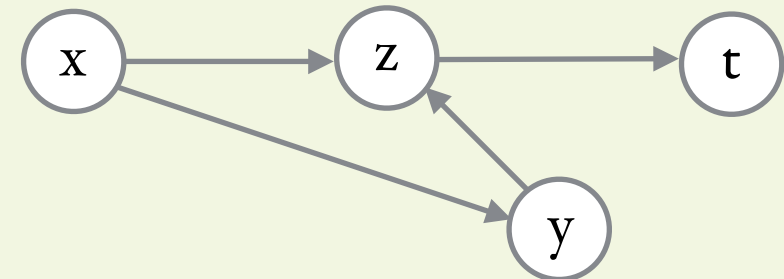
underlying  
undirected graph is  
acyclic

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



**acyclic**

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

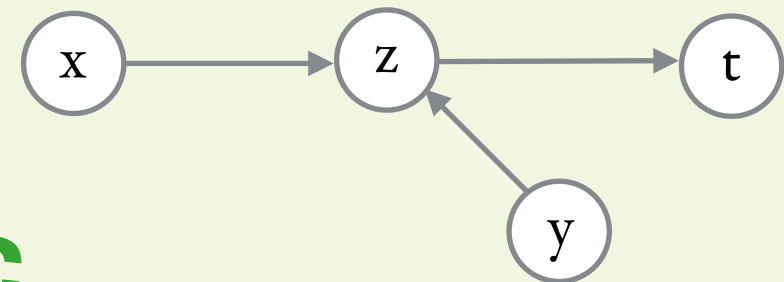


# Acyclic CQ's : Definition

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

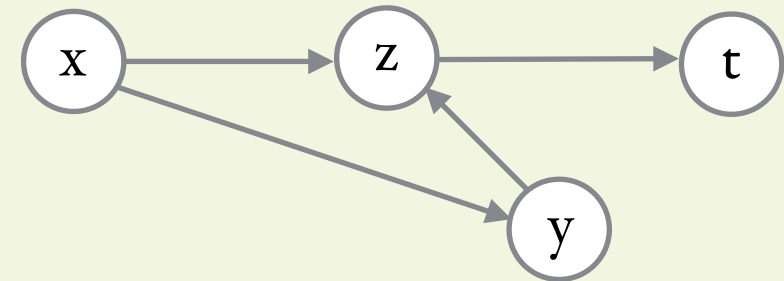
underlying  
undirected graph is  
acyclic

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



**acyclic**

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



**non acyclic**



# Acyclic CQ's

On graphs: 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

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

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

# Acyclic CQ's

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

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

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

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

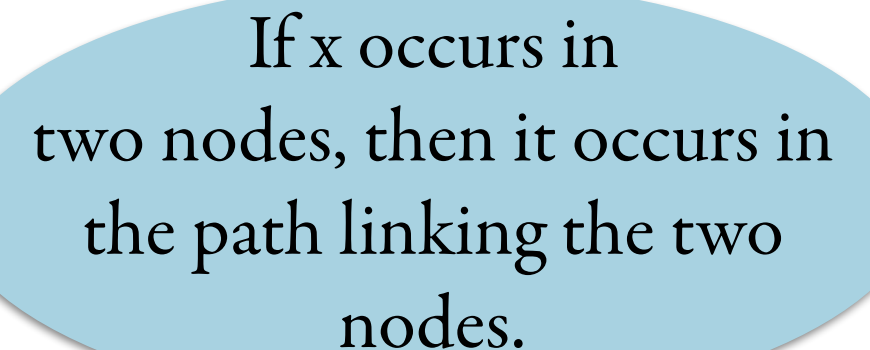
- nodes are the atoms  $R_i(\bar{z}_i)$
- for every variable  $x$  of  $\phi$  the set of  $R_i(\bar{z}_i)$ 's with  $x \in \bar{z}_i$  forms a subtree of  $T$

# Acyclic CQ's

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

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

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



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  $\phi$  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

Alternatively, if its canonical hyper-graph is  $\alpha$ -acyclic.

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

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

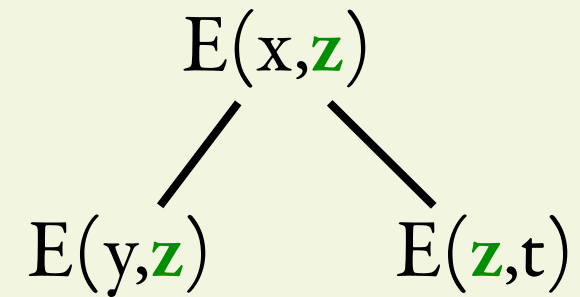
If  $x$  occurs in two nodes, then it occurs in the path linking the two nodes.

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

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

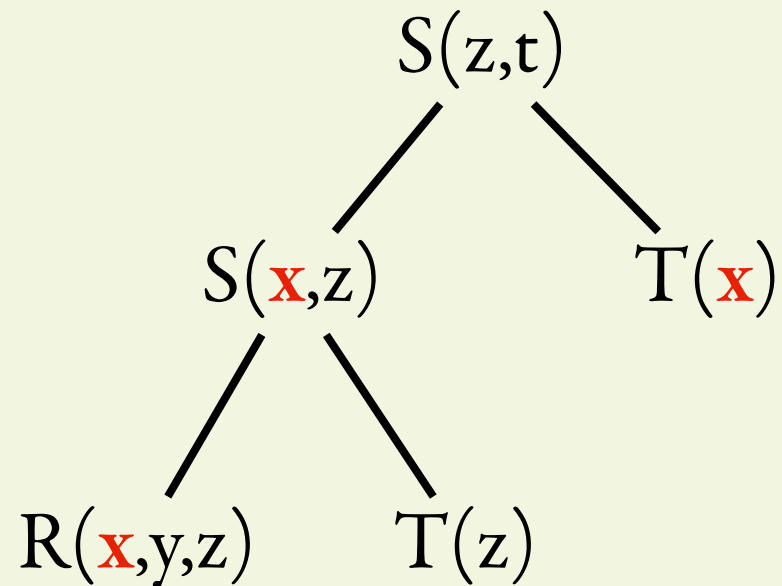
# Acyclic CQ's

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

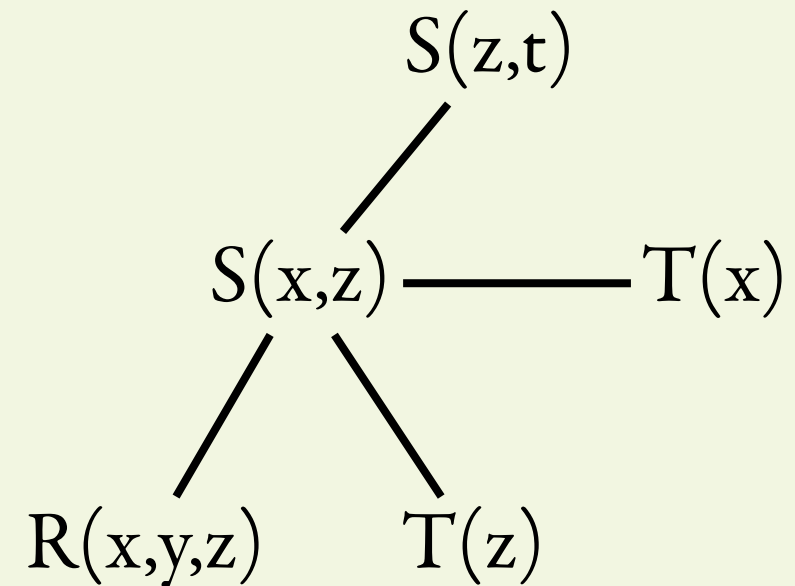


join tree

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

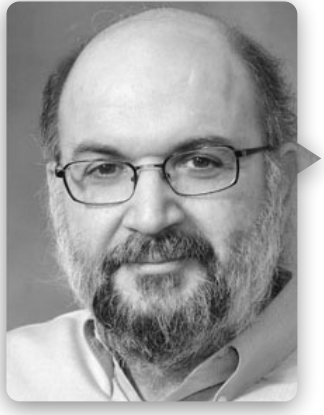


not a join tree



a join tree

# Acyclic CQ's



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

[Yannakakis]

The **semi-join**

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

Note:  $R \bowtie_{\{i_1=j_1, \dots, i_n=j_n\}} S \subseteq R$