day 4

ESSLLI 2016
Bolzano, Italy

Logical foundations of databases

Diego Figueira

Gabriele Puppis

CNRS LaBRI

Recap

- **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 Conjunctive Queries

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

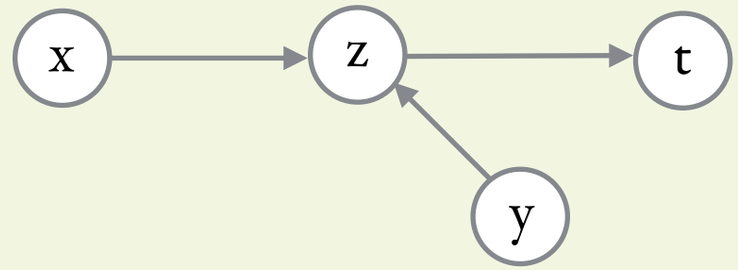
underlying undirected graph is acyclic

Acyclic CQ's : Definition

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

underlying undirected graph is acyclic

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

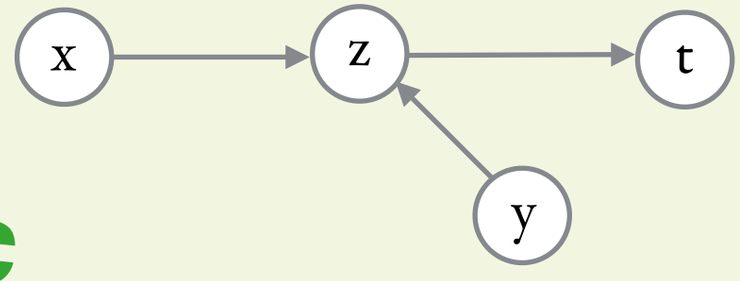


Acyclic CQ's : Definition

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

underlying undirected graph is acyclic

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



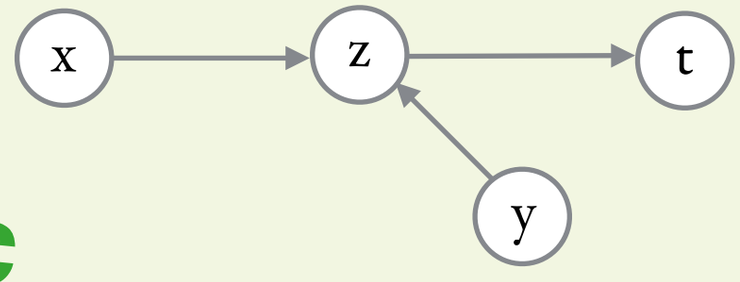
acyclic

Acyclic CQ's : Definition

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

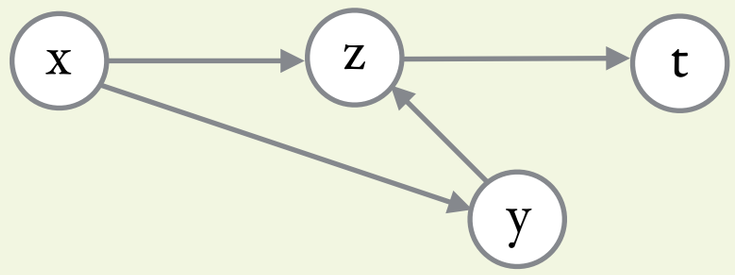
underlying undirected graph is acyclic

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



acyclic

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

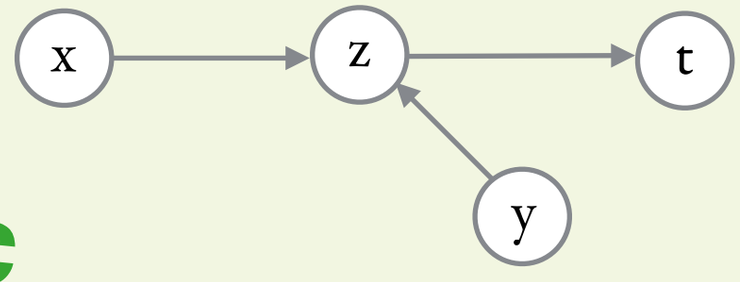


Acyclic CQ's : Definition

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

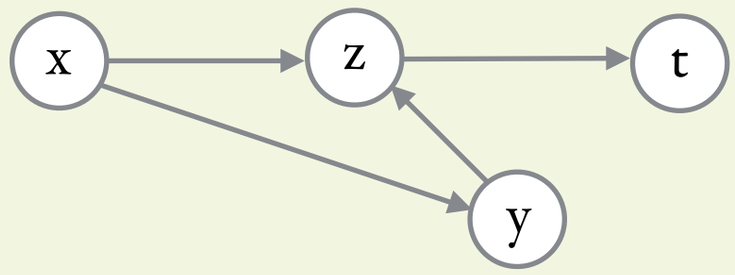
underlying undirected graph is acyclic

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



acyclic

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



non **acyclic**

Acyclic CQ's

On graphs: CQ 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

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

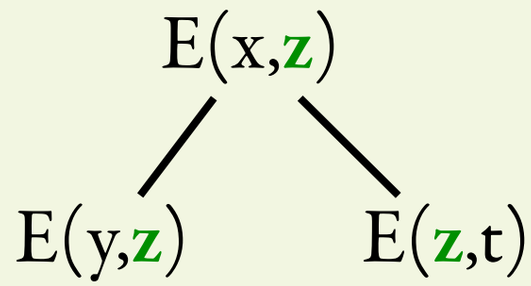
If x occurs in two nodes, then it occurs in the path linking the two nodes.

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

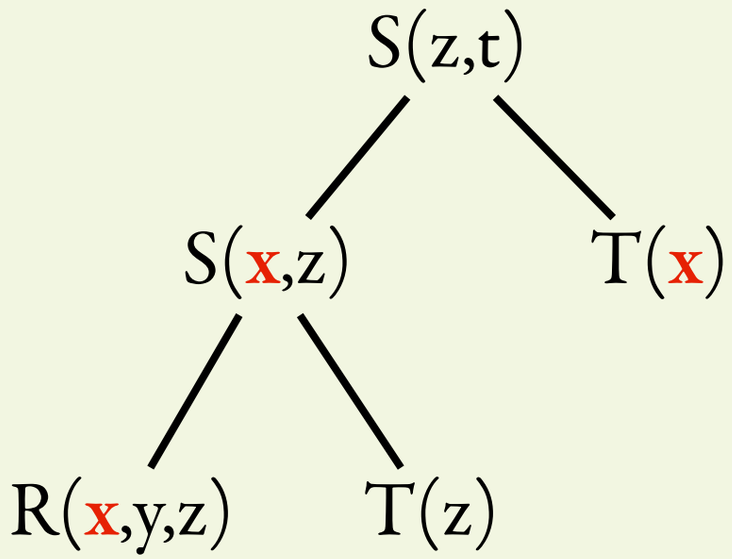
Acyclic CQ's

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

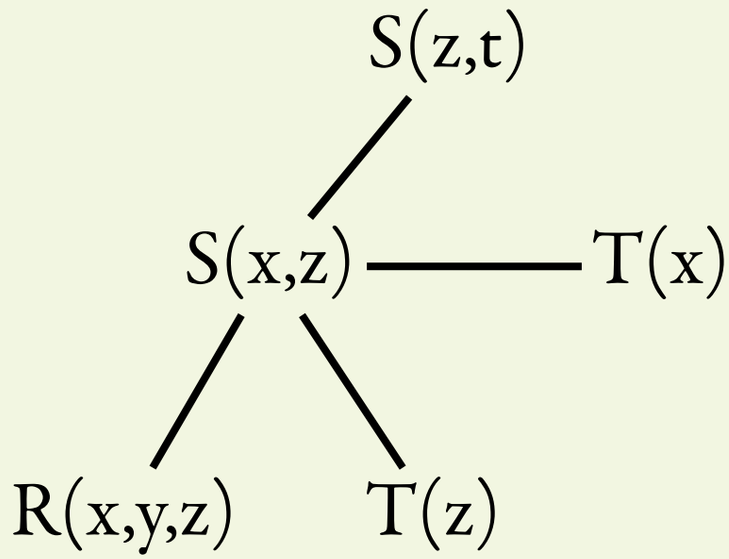


join tree

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



not a join tree



a join tree

Acyclic CQ's

$\phi_1 = R(x_1,x_2,x_3) \wedge R(x_1,x_6,x_5) \wedge R(x_3,x_4,x_5)$

join tree?

Acyclic CQ's

$\phi_1 = R(x_1,x_2,x_3) \wedge R(x_1,x_6,x_5) \wedge R(x_3,x_4,x_5)$ join tree?

$R(x_3,x_4,x_5)$ ——— $R(x_1,x_2,x_3)$ ——— $R(x_1,x_6,x_5)$?

Acyclic CQ's

$\phi_1 = R(x_1,x_2,x_3) \wedge R(x_1,x_6,x_5) \wedge R(x_3,x_4,x_5)$ join tree?

$R(x_3,x_4,x_5)$ ——— $R(x_1,x_2,x_3)$ ——— $R(x_1,x_6,x_5)$?

$R(x_1,x_2,x_3)$ ——— $R(x_1,x_6,x_5)$ ——— $R(x_3,x_4,x_5)$?

Acyclic CQ's

$\phi_1 = R(x_1,x_2,x_3) \wedge R(x_1,x_6,x_5) \wedge R(x_3,x_4,x_5)$ join tree?

$R(x_3,x_4,x_5)$ ——— $R(x_1,x_2,x_3)$ ——— $R(x_1,x_6,x_5)$?

$R(x_1,x_2,x_3)$ ——— $R(x_1,x_6,x_5)$ ——— $R(x_3,x_4,x_5)$?

$\phi_2 = R(x_1,x_2,x_3) \wedge R(x_1,x_6,x_5) \wedge R(x_3,x_4,x_5) \wedge R(x_1,x_3,x_5)$ join tree?

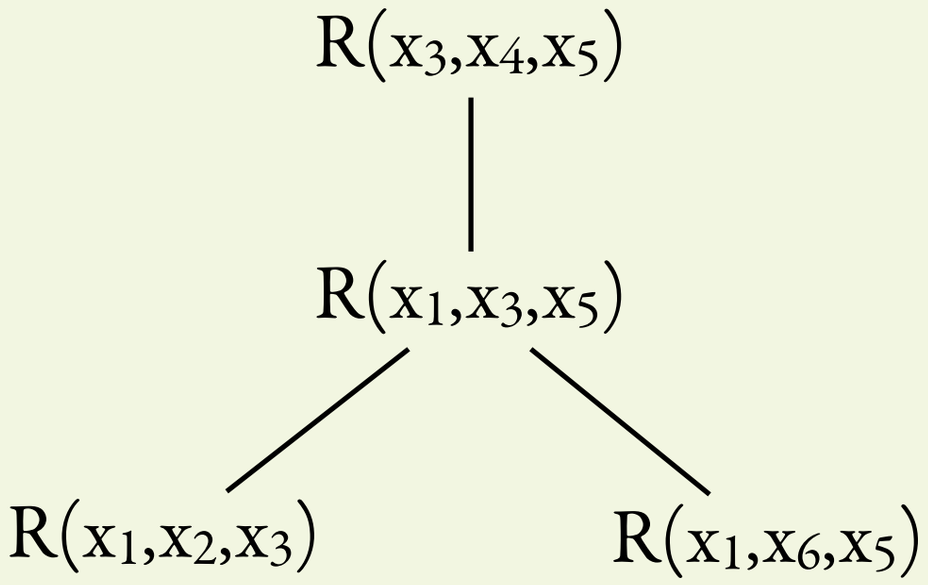
Acyclic CQ's

$\phi_1 = R(x_1,x_2,x_3) \wedge R(x_1,x_6,x_5) \wedge R(x_3,x_4,x_5)$ join tree?

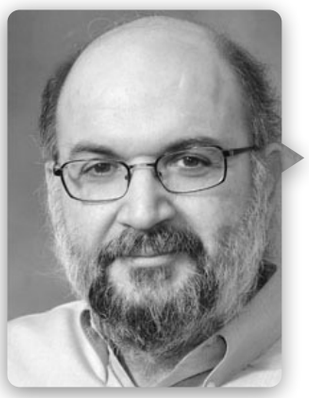
$R(x_3,x_4,x_5)$ ——— $R(x_1,x_2,x_3)$ ——— $R(x_1,x_6,x_5)$?

$R(x_1,x_2,x_3)$ ——— $R(x_1,x_6,x_5)$ ——— $R(x_3,x_4,x_5)$?

$\phi_2 = R(x_1,x_2,x_3) \wedge R(x_1,x_6,x_5) \wedge R(x_3,x_4,x_5) \wedge R(x_1,x_3,x_5)$ join tree?



Acyclic CQ's



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

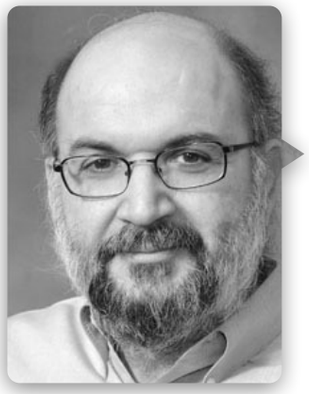
[Yannakakis]

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

Acyclic CQ's



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

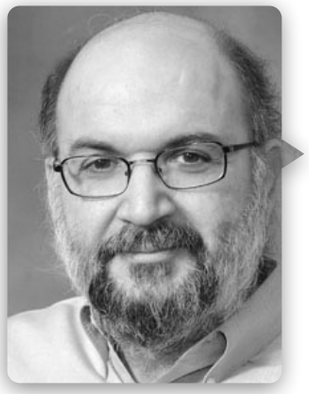
[Yannakakis]

1. Compute the join tree T for ϕ
2. Populate the nodes of T with corresponding relations of D
3. For every leaf $S(x_1,\ldots,x_n)$ with parent $R(y_1,\ldots,y_m)$ perform

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

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

Acyclic CQ's



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

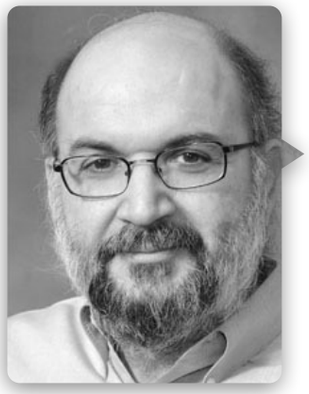
[Yannakakis]

1. Compute the join tree T for ϕ — in linear time
2. Populate the nodes of T with corresponding relations of D
3. For every leaf $S(x_1,\ldots,x_n)$ with parent $R(y_1,\ldots,y_m)$ perform

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

   and delete the leaf $S(x_1,\ldots,x_n)$.
4. Repeat until we are left with one node. If it contains a non-empty relation, then D satisfies 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 correspondi

3. For every leaf $S(x_1,\ldots,x_n)$ with parent $R(v$

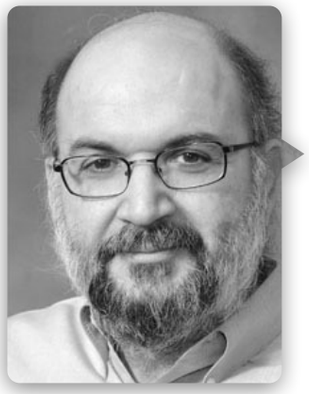
   $R \ltimes_{\{i=j \,|\, x_i = y_j\}} S$

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

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

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]

1. Compute the join tree T for ϕ — in linear time
2. Populate the nodes of T with corresponding relations of D
3. For every leaf $S(x_1,\ldots,x_n)$ with parent $R(y_1,\ldots,y_m)$ perform

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

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

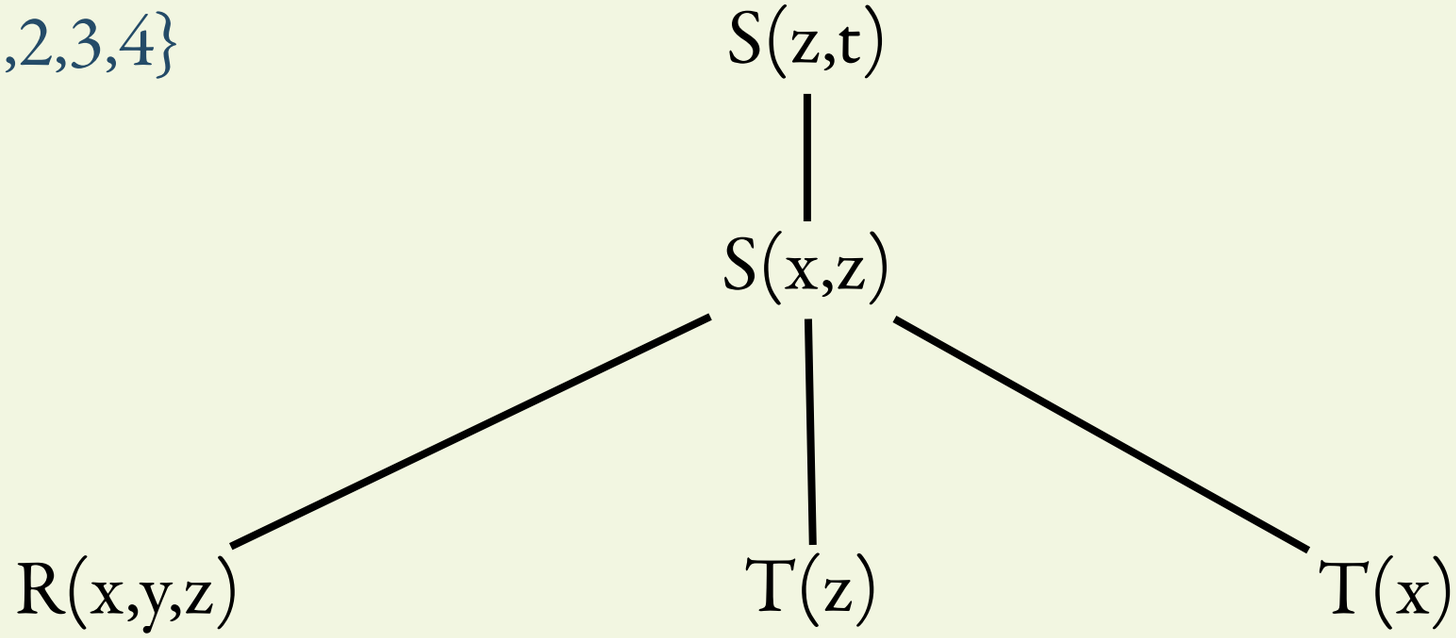
Acyclic CQ's

(combined c.)

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}



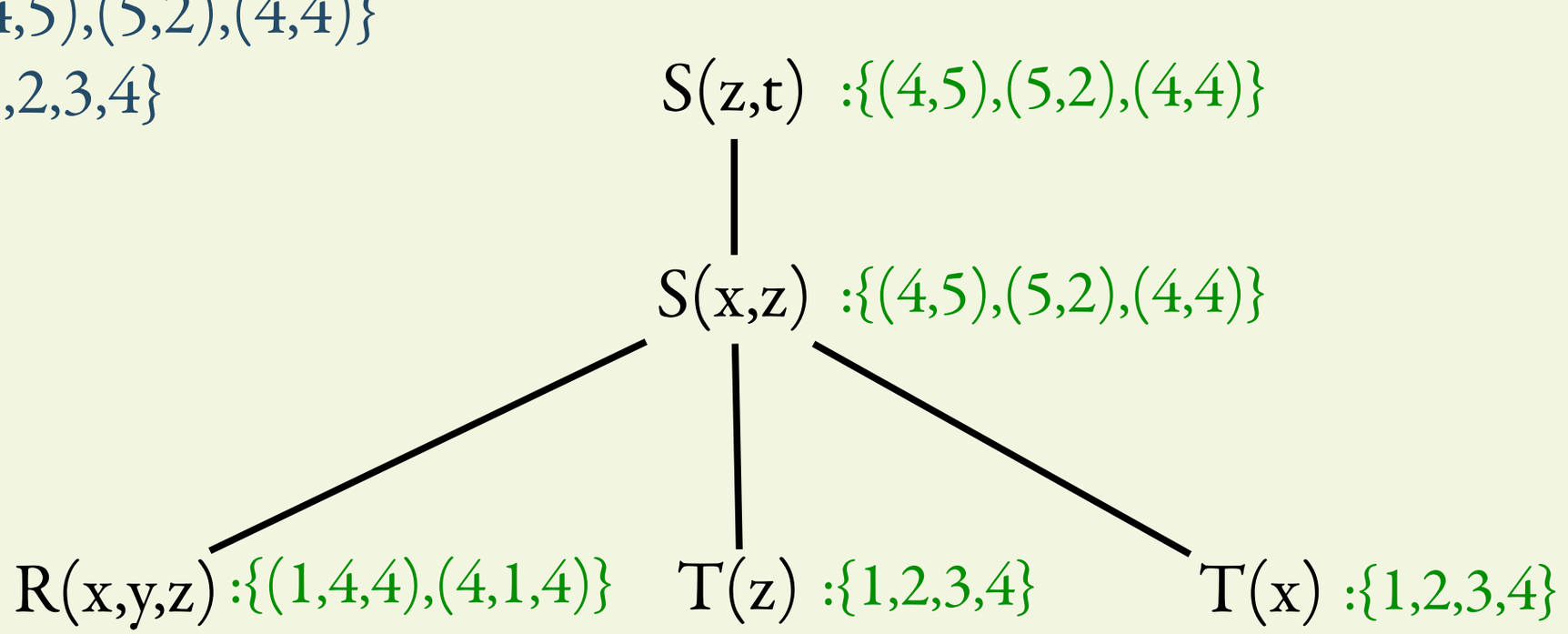
Acyclic CQ's

(combined c.)

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}



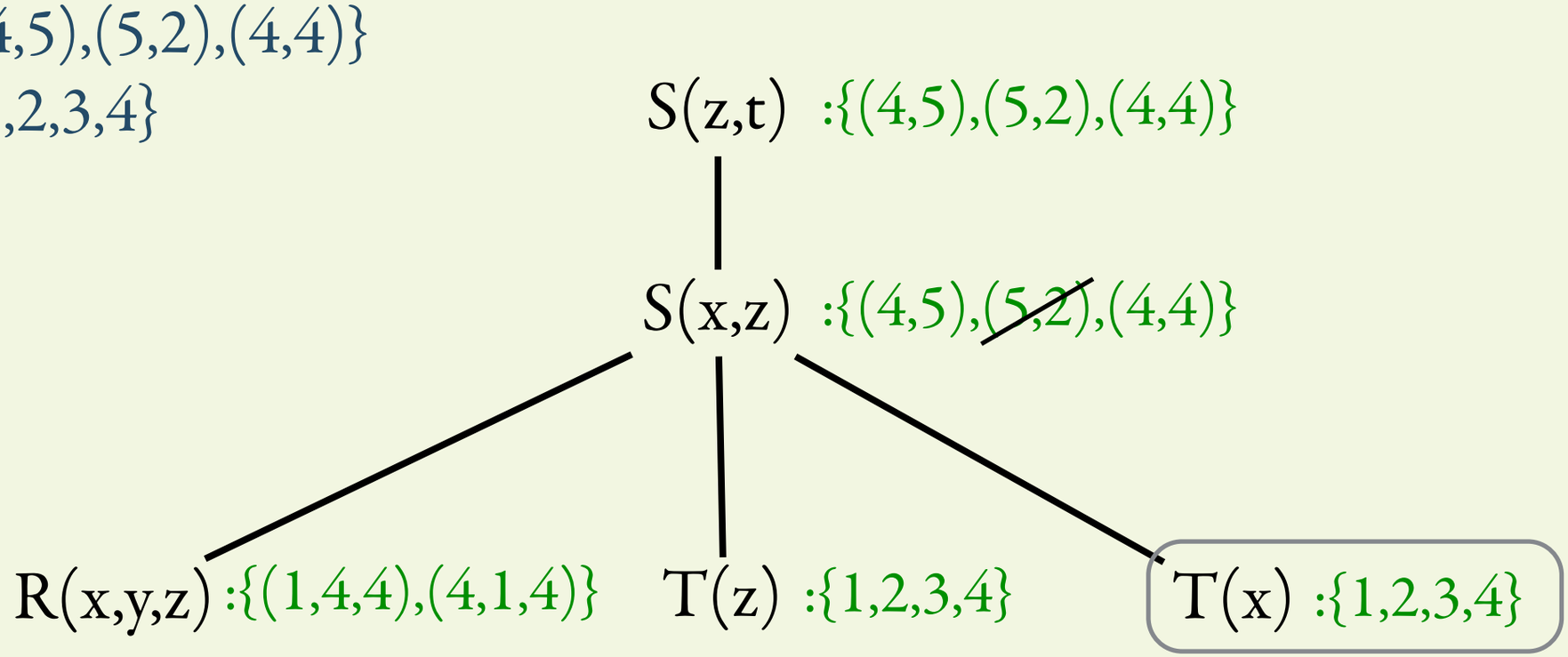
Acyclic CQ's

(combined c.)

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}



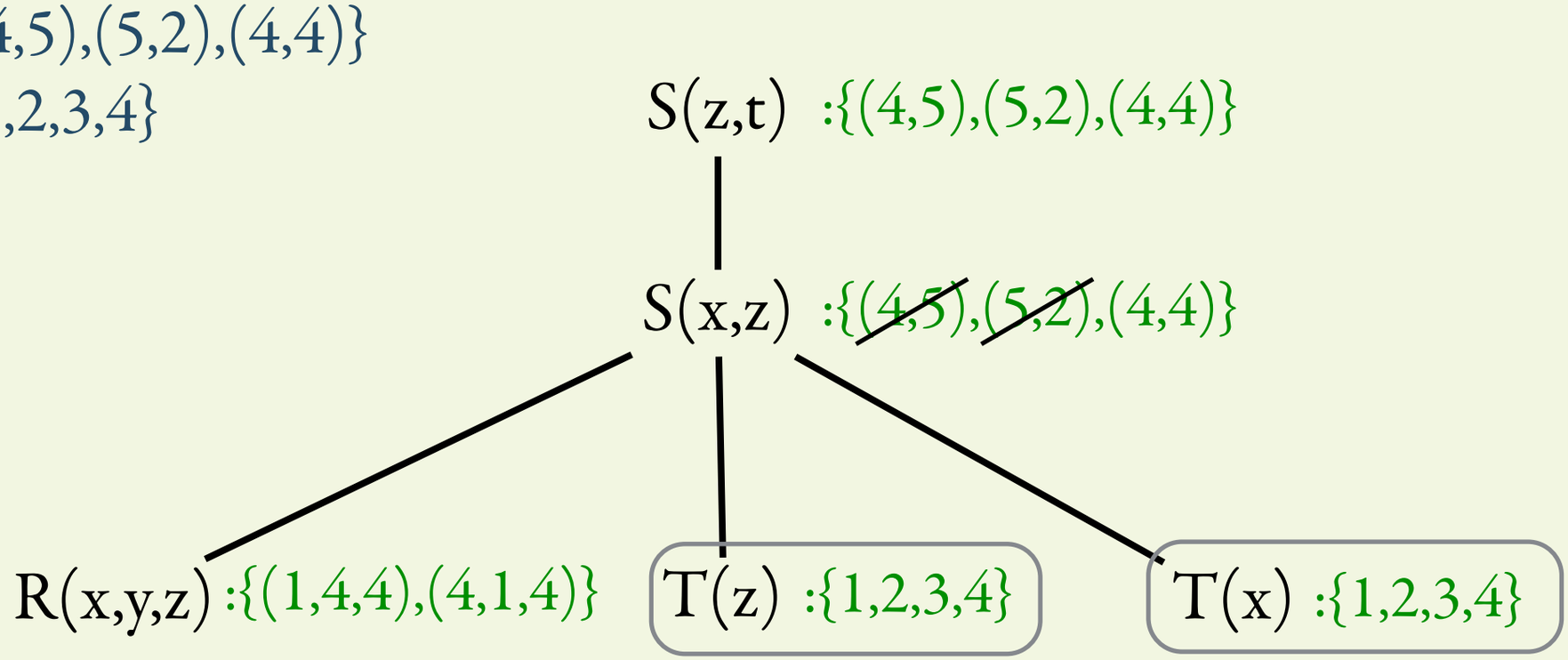
Acyclic CQ's

(combined c.)

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}



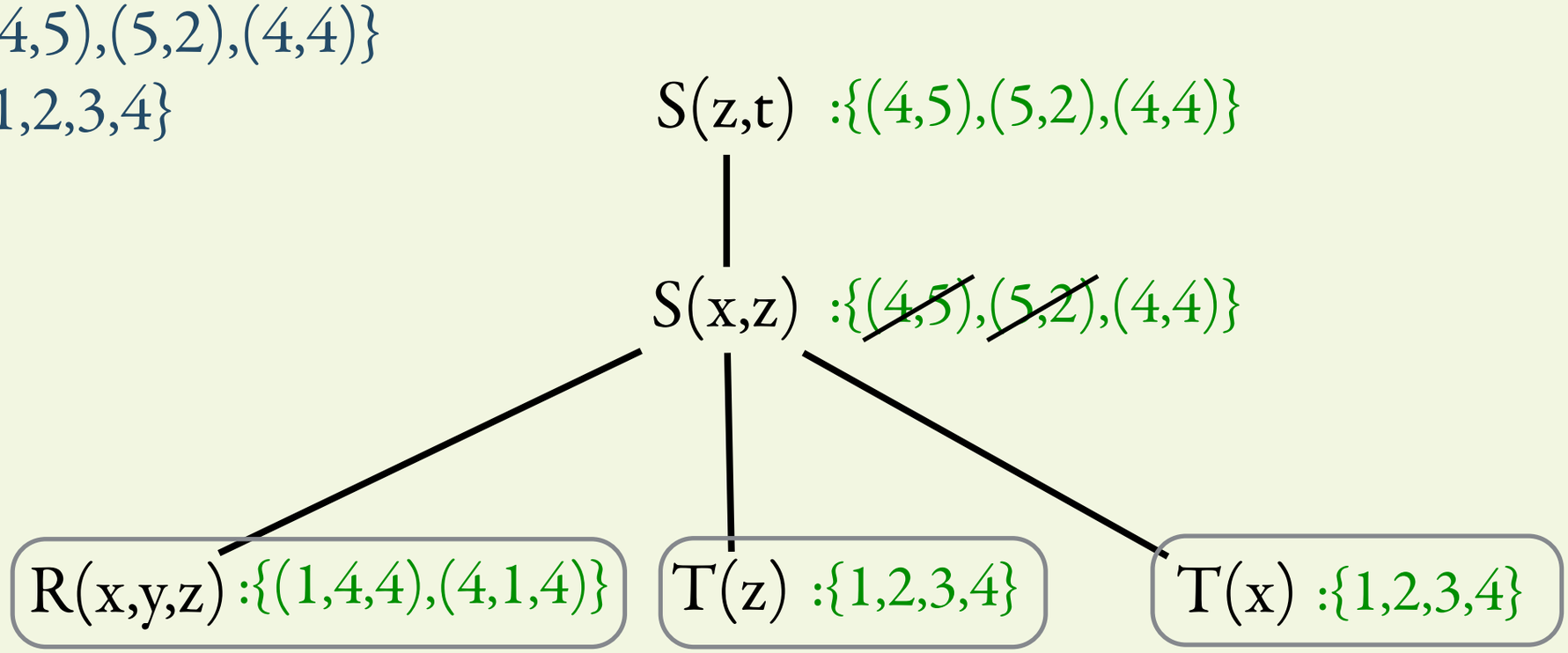
Acyclic CQ's

(combined c.)

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}



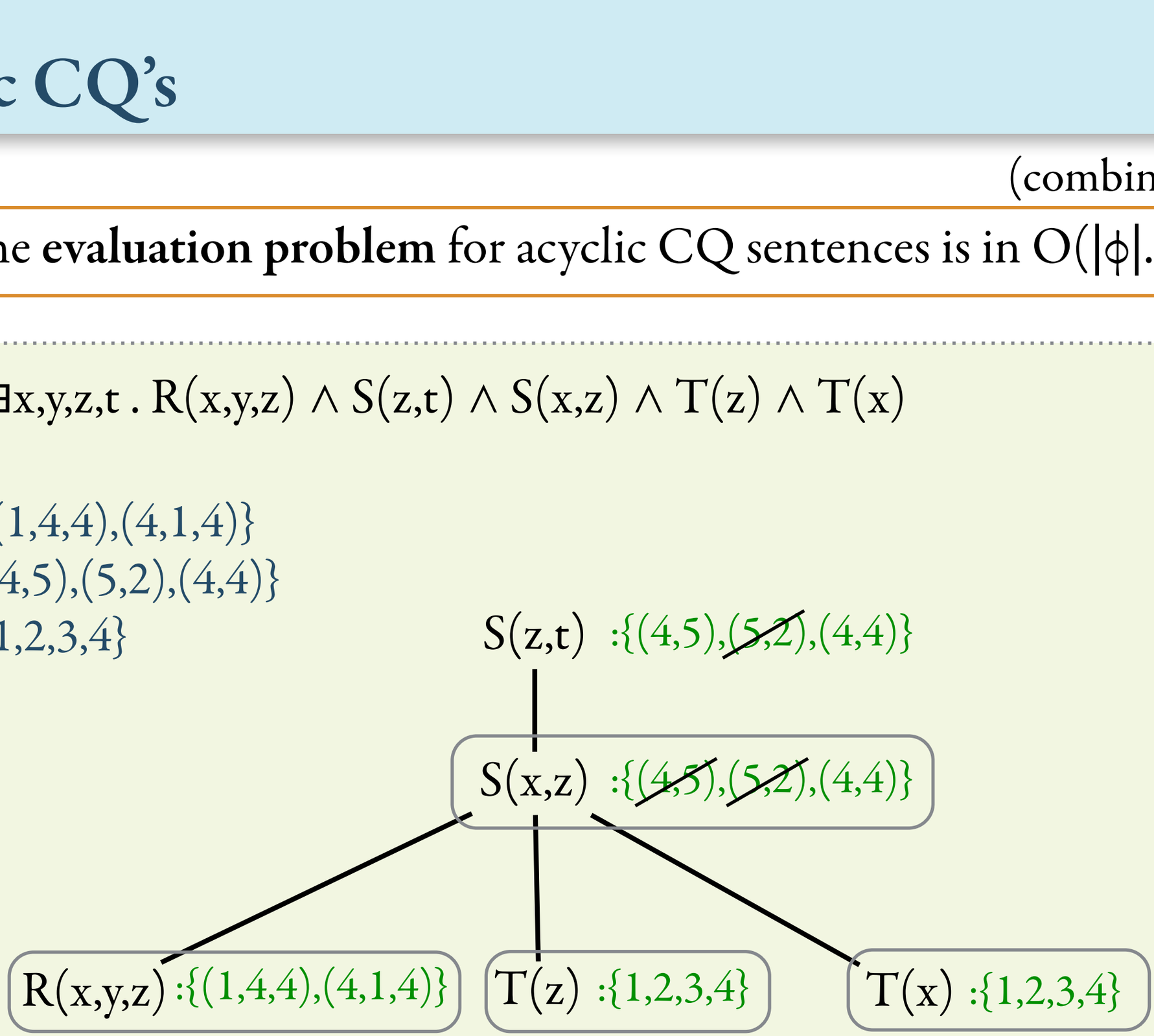
Acyclic CQ's

(combined c.)

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}



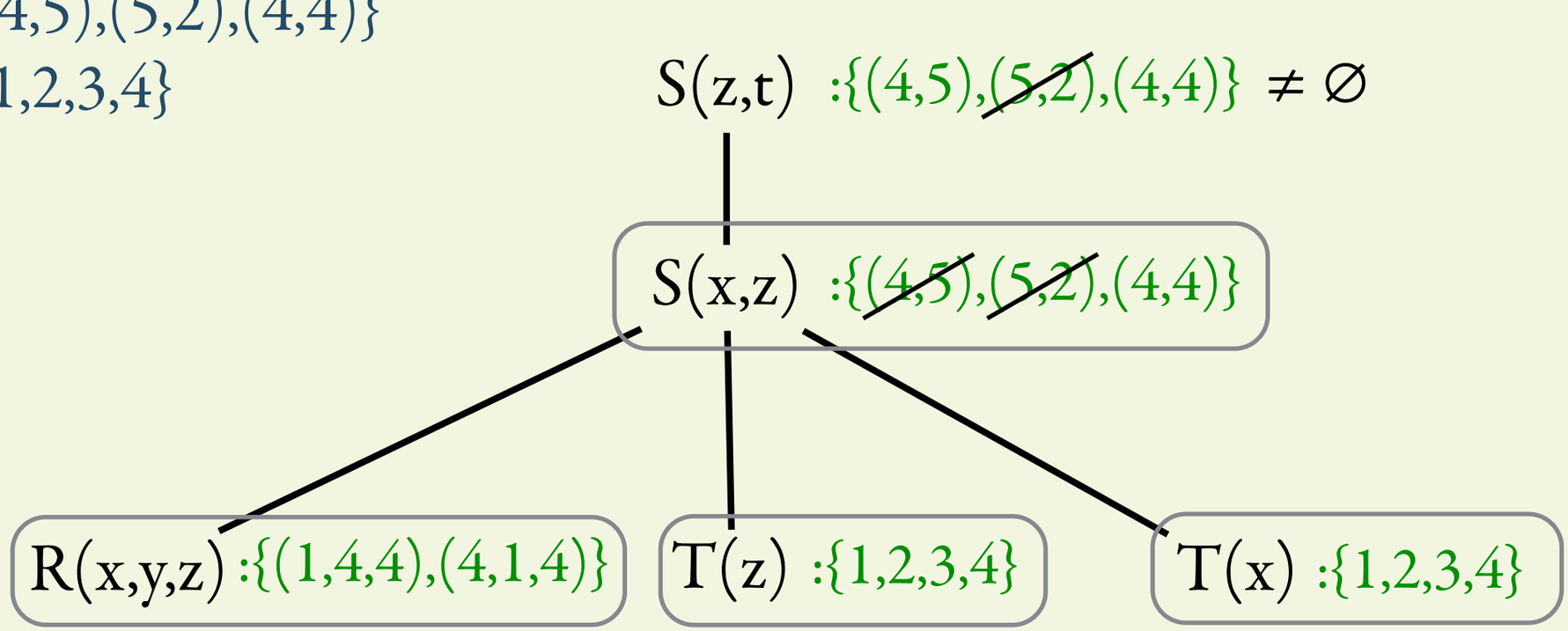
Acyclic CQ's

(combined c.)

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

How to compute a join tree?

GYO reducts [Graham, Yu, Ozsoyoglu]

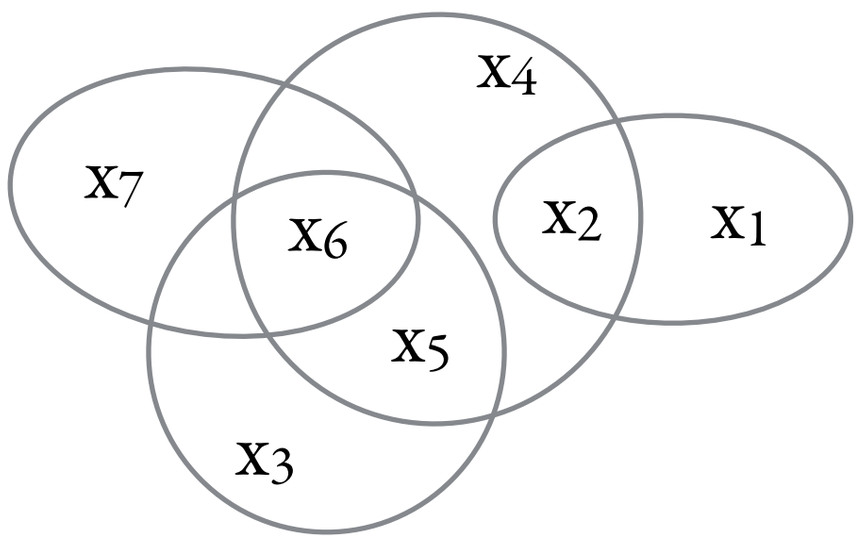
An **ear** of a hypergraph (V,E) is a hyperedge **e** in E such that one of the following conditions holds:

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

(a) **only** in **e** or

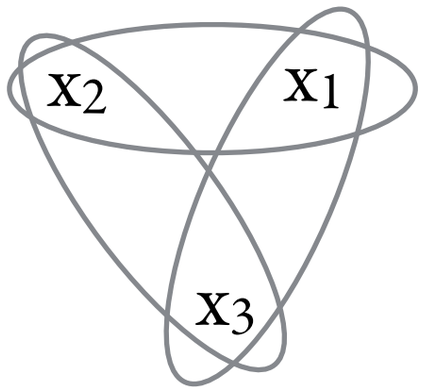
(b) in **e'**; or

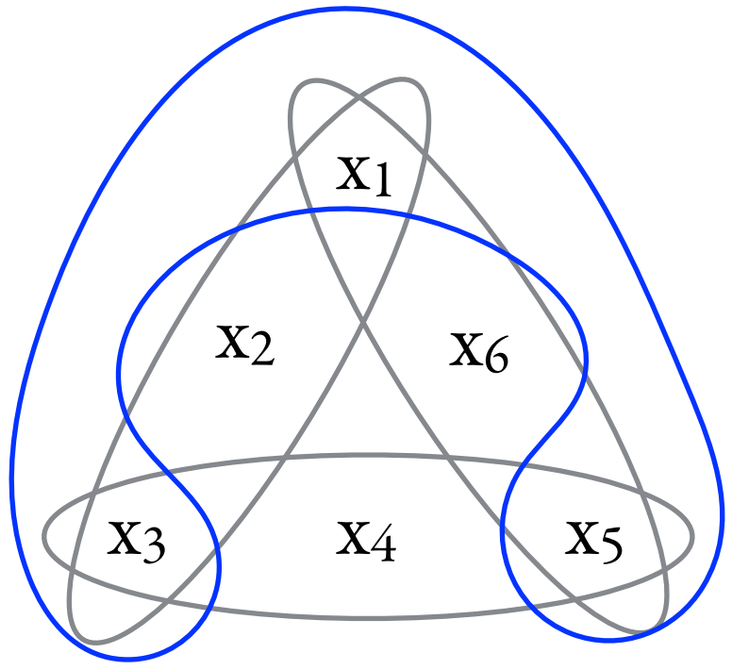
(2) **e** has no intersection with any other hyperedge.



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!

Acyclic CQ's

How to compute a join tree?

GYO algorithm [Graham, Yu, Ozsoyoglu]

Given the query $\phi = R_1(X_1) \wedge \cdots \wedge R_n(X_n)$
Consider its canonical structure G_ϕ
  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 ϕ.
  Repeat.

Acyclic CQ's

How to compute a join tree?

GYO algorithm [Graham, Yu, Ozsoyoglu]

Given the query $\phi = R_1(X_1) \wedge \cdots \wedge R_n(X_n)$
Consider its canonical structure G_ϕ
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 ϕ.
Repeat.

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

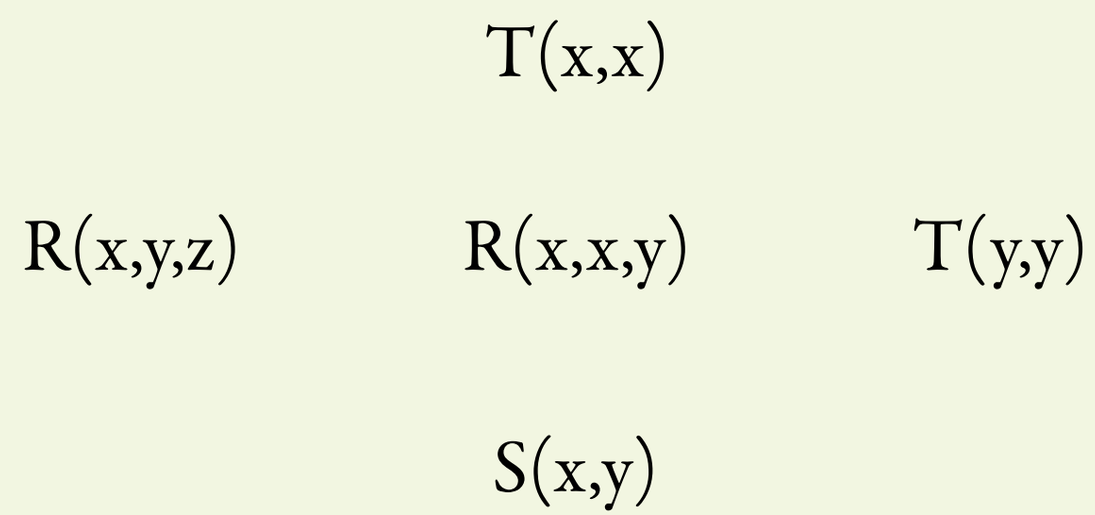
Acyclic CQ's

How to compute a join tree?

GYO algorithm [Graham, Yu, Ozsoyoglu]

Given the query $\phi = R_1(X_1) \wedge \cdots \wedge R_n(X_n)$
Consider its canonical structure G_ϕ
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 ϕ.
Repeat.

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



Acyclic CQ's

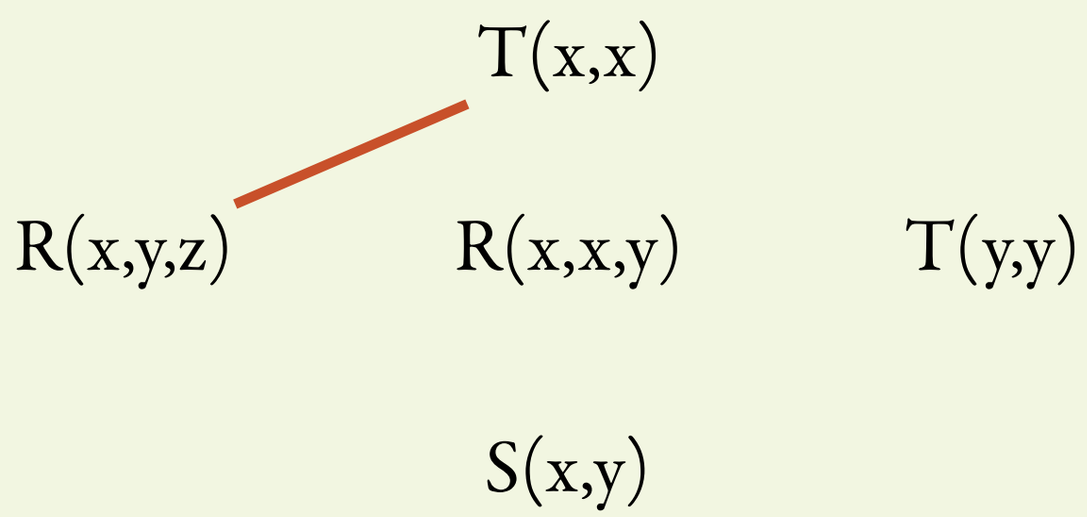
How to compute a join tree?

GYO algorithm [Graham, Yu, Ozsoyoglu]

Given the query $\phi = R_1(X_1) \wedge \cdots \wedge R_n(X_n)$
Consider its canonical structure G_ϕ
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 ϕ.
Repeat.

E.g.

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



Acyclic CQ's

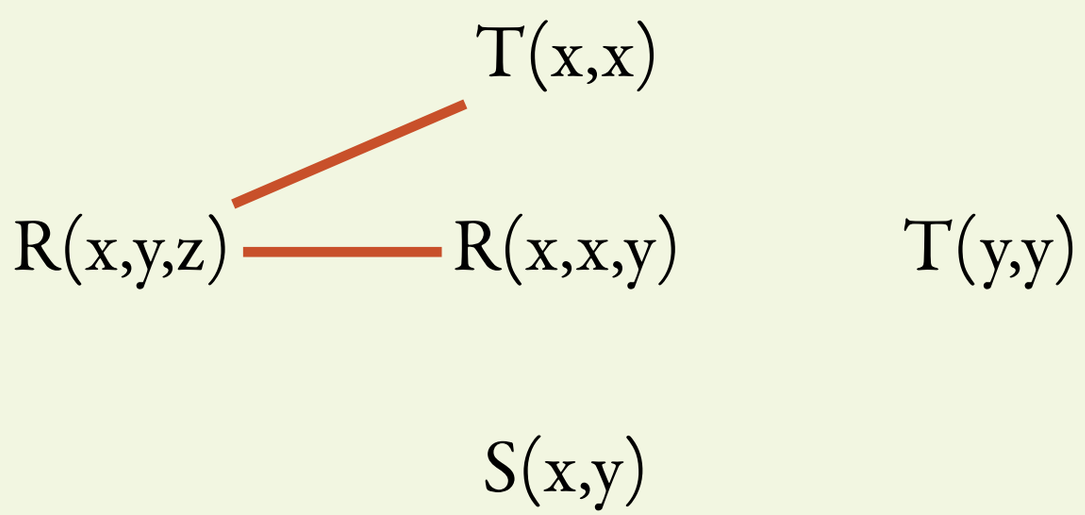
How to compute a join tree?

GYO algorithm [Graham, Yu, Ozsoyoglu]

Given the query $\phi = R_1(X_1) \wedge \cdots \wedge R_n(X_n)$
Consider its canonical structure G_ϕ
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 ϕ.
Repeat.

E.g.

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



Acyclic CQ's

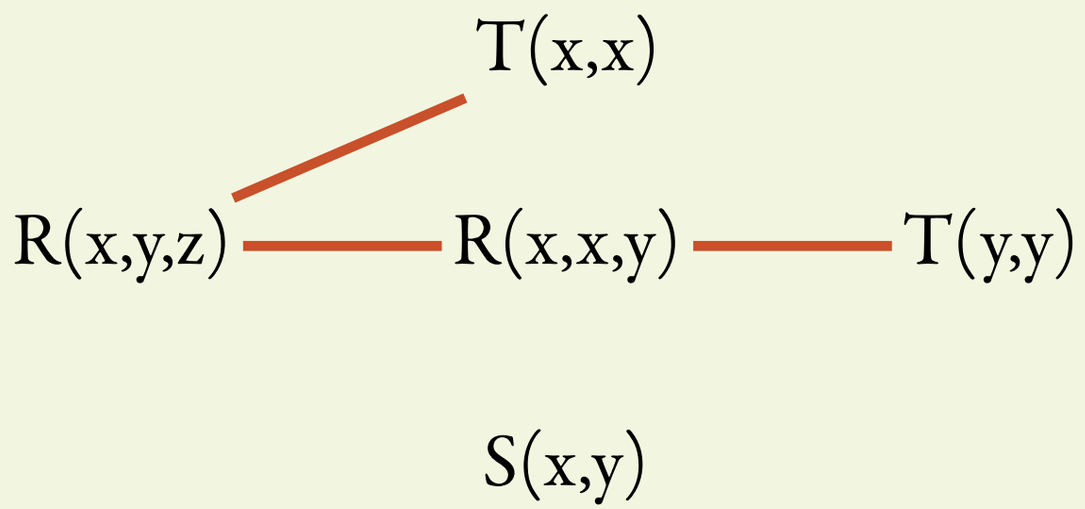
How to compute a join tree?

GYO algorithm [Graham, Yu, Ozsoyoglu]

Given the query $\phi = R_1(X_1) \wedge \cdots \wedge R_n(X_n)$
Consider its canonical structure G_ϕ
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 ϕ.
Repeat.

E.g.

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



Acyclic CQ's

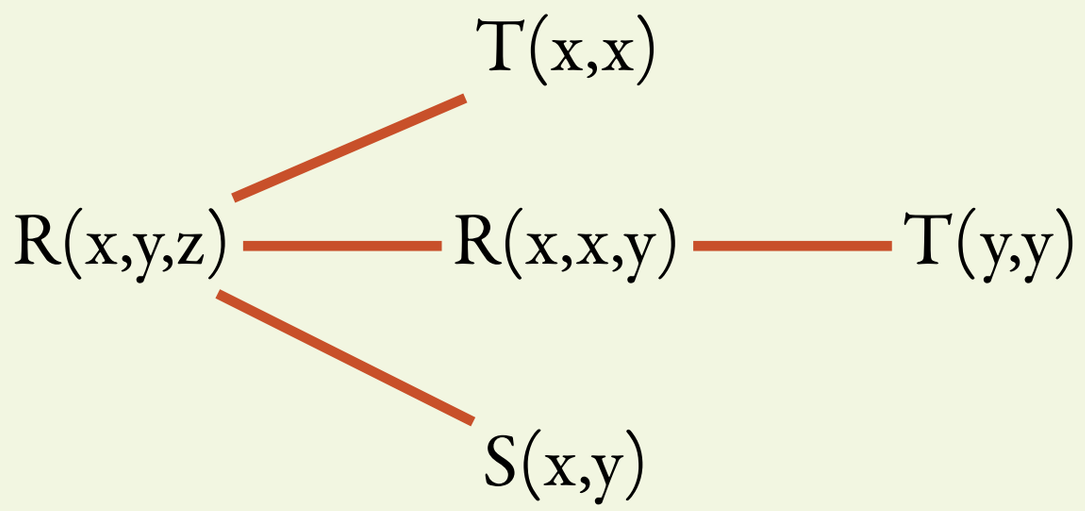
How to compute a join tree?

GYO algorithm [Graham, Yu, Ozsoyoglu]

Given the query $\phi = R_1(X_1) \wedge \cdots \wedge R_n(X_n)$
Consider its canonical structure G_ϕ
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 ϕ.
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 $\phi = R_1(X_1) \wedge \cdots \wedge R_n(X_n)$

Consider its canonical structure G_ϕ

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

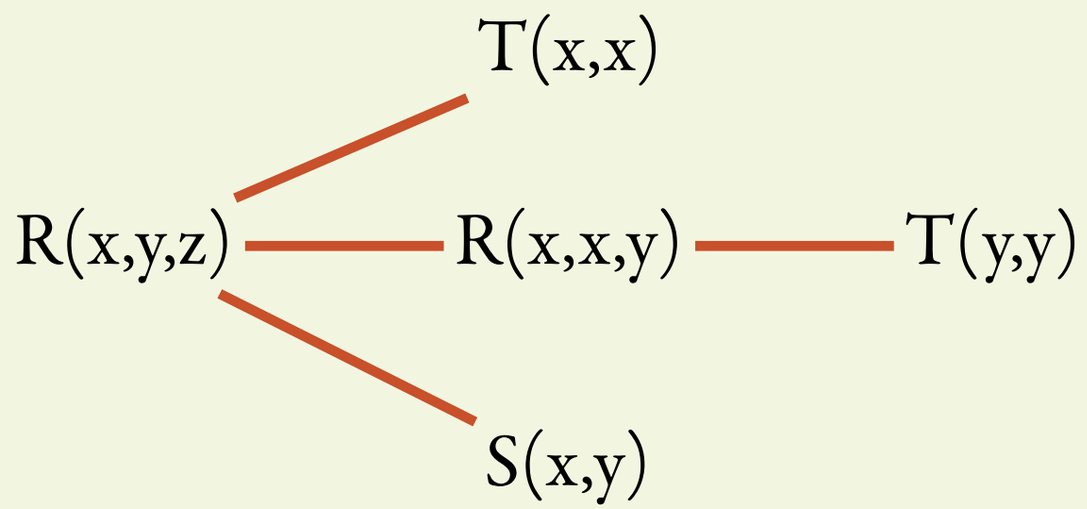
Put an edge between $R_i(X_i)$ and $R_j(X_j)$, and remove R_i from ϕ.

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 ⊆ **LOGCFL** ⊆ AC^1 ⊆ NC^2 ⊆ P

the class of problems logspace-reducible to a context-free language

Beyond acyclic CQ's

Treewidth = a measure of the cyclicity of (hyper-)graphs

$$\mathrm{tw} : \mathrm{CQ} \longrightarrow \mathbf{N}$$

For a fixed k,

the evaluation pb for queries of tw $\leq k$

can be done in **polynomial time.**

[Chekuri, Rajaraman]

Idea: the lower tw(ϕ), the more ϕ resembles a tree

Tree-width, definition

A **tree decomposition** of a graph G:

A bunch of graphs with a special edge "· · · ·" between their nodes so that

1) they have a **tree shape** and

2) collapsing "· · · ·" edges $\rightsquigarrow$ G

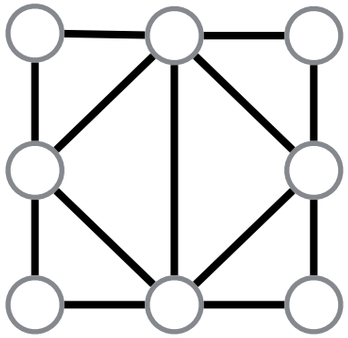
Tree-width, definition

A **tree decomposition** of a graph G:

A bunch of graphs with a special edge "· · · ·" between their nodes so that

1) they have a **tree shape** and

2) collapsing "· · · ·" edges $\rightsquigarrow$ G



G

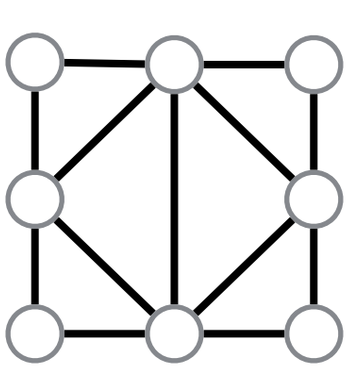
Tree-width, definition

A **tree decomposition** of a graph G:

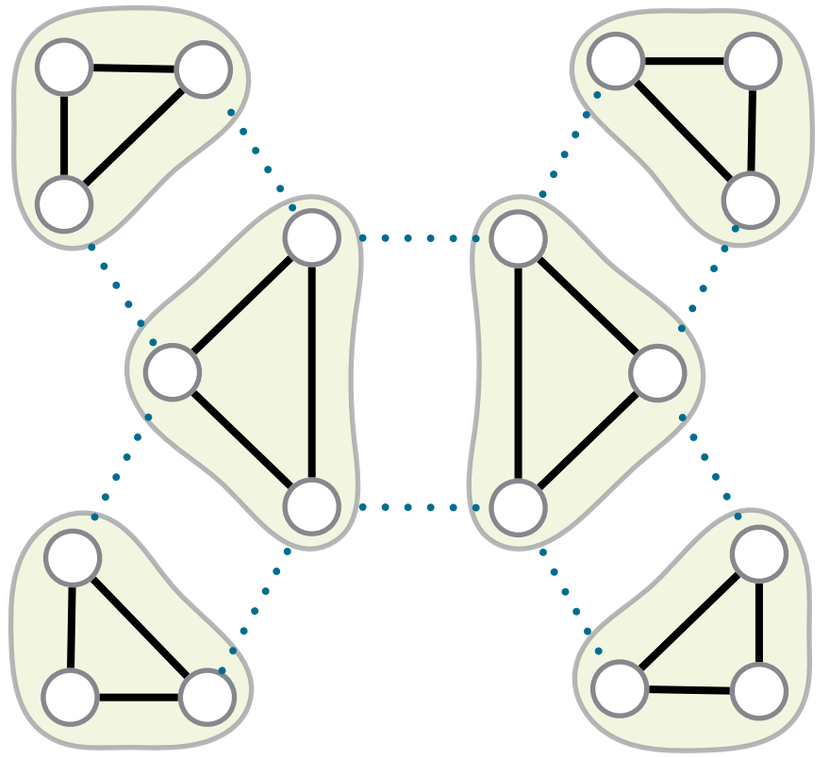
A bunch of graphs with a special edge "· · · ·" between their nodes so that

1) they have a **tree shape** and

2) collapsing "· · · ·" edges ⇝ G



G



tree decomposition of G

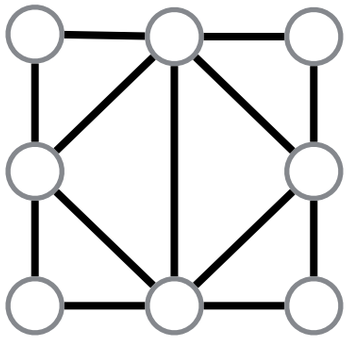
Tree-width, definition

A **tree decomposition** of a graph G:

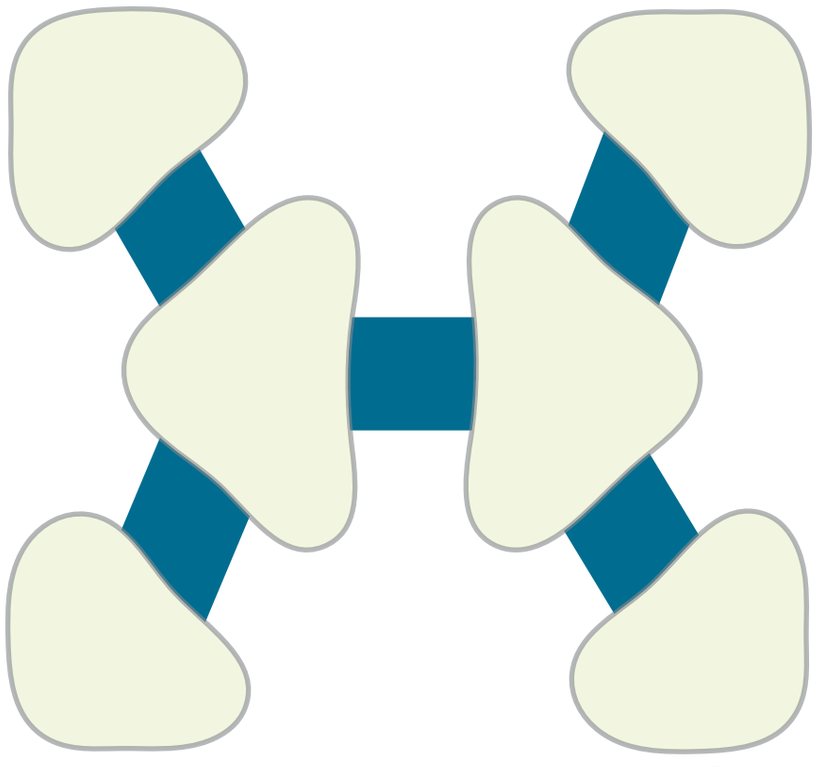
A bunch of graphs with a special edge "· · · ·" between their nodes so that

1) they have a **tree shape** and

2) collapsing "· · · ·" edges ⇝ G



G



tree decomposition of G

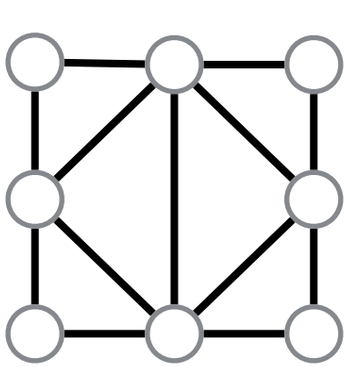
Tree-width, definition

A **tree decomposition** of a graph G:

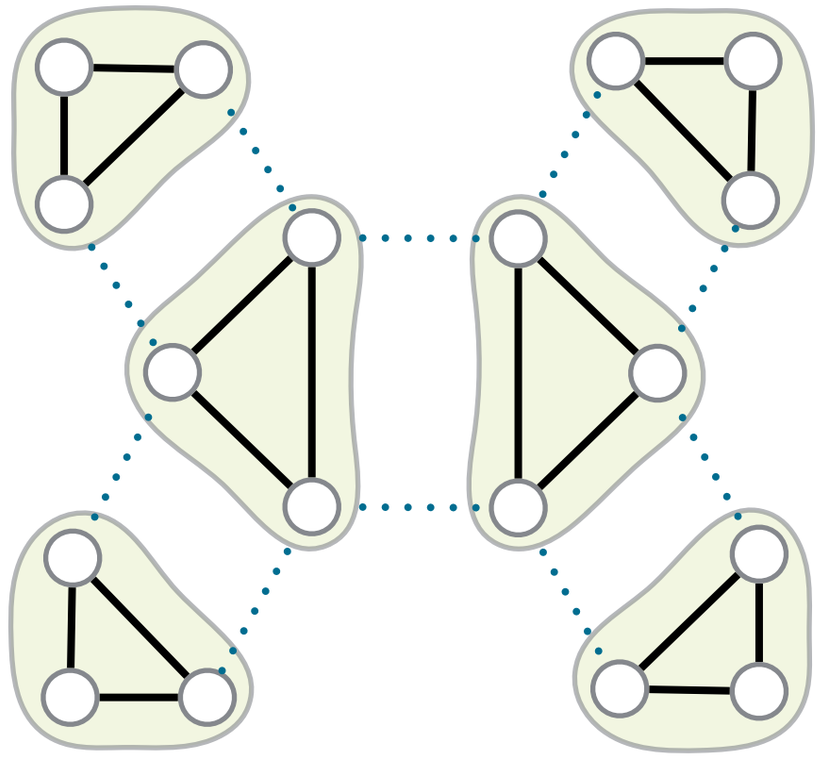
A bunch of graphs with a special edge "· · · ·" between their nodes so that

1) they have a **tree shape** and

2) collapsing "· · · ·" edges ⇝ G



G



tree decomposition of G

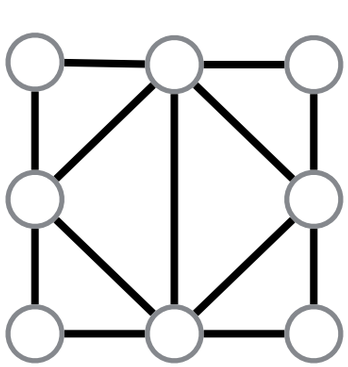
Tree-width, definition

A **tree decomposition** of a graph G:

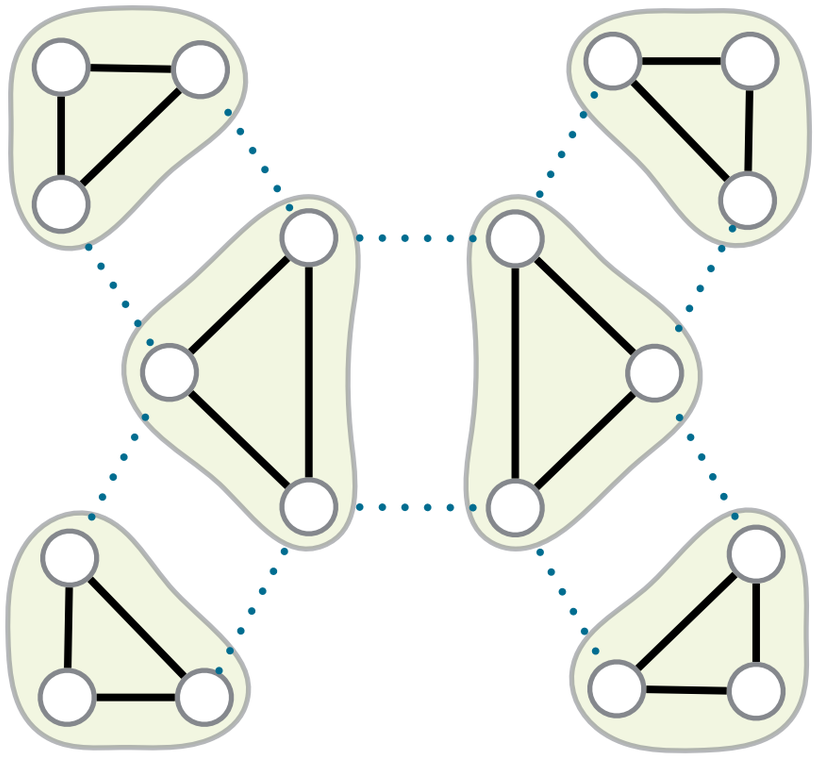
A bunch of graphs with a special edge "· · · ·" between their nodes so that

1) they have a **tree shape** and

2) collapsing "· · · ·" edges ⇝ G



G



tree decomposition of G

width of decomposition = maximum size of graphs −1

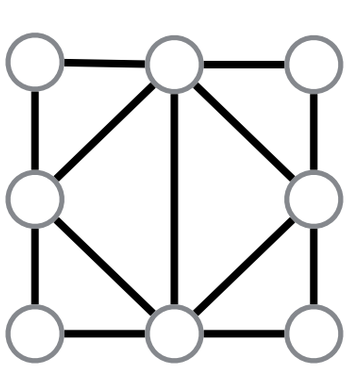
Tree-width, definition

A **tree decomposition** of a graph G:

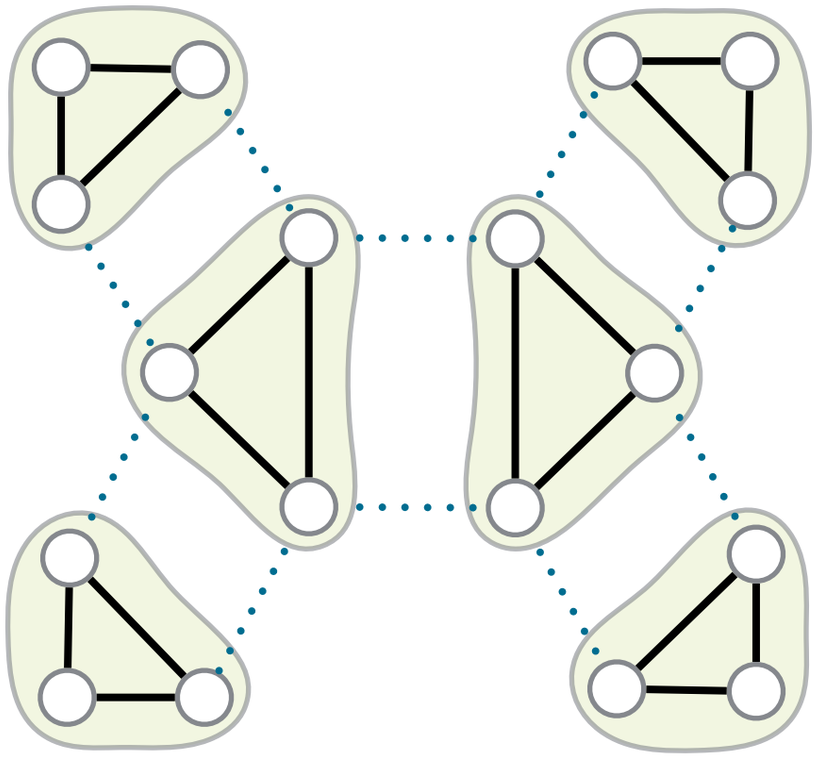
A bunch of graphs with a special edge "· · · ·" between their nodes so that

1) they have a **tree shape** and

2) collapsing "· · · ·" edges ⇝ G



G



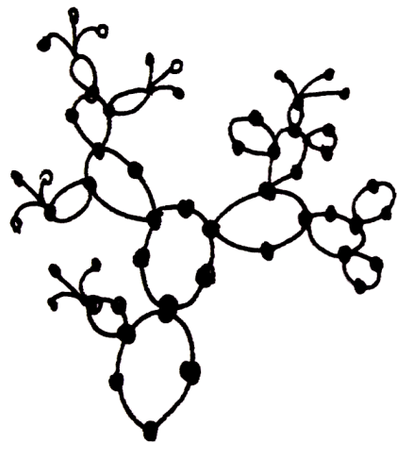
tree decomposition of G

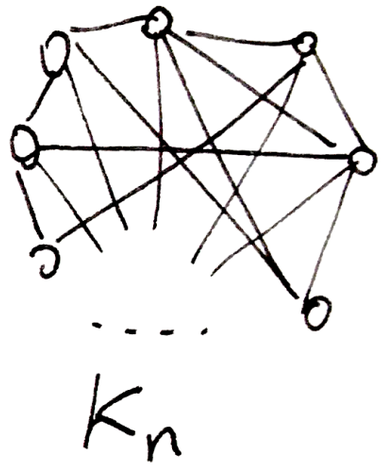
width of decomposition = maximum size of graphs –1

tree-width of G = minimum width of decomposition of G

Tree-width, examples

(a tree)





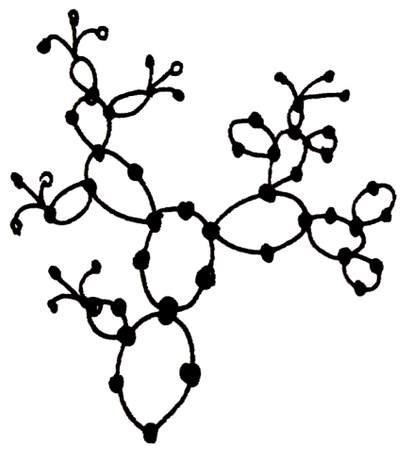
tree-width 1

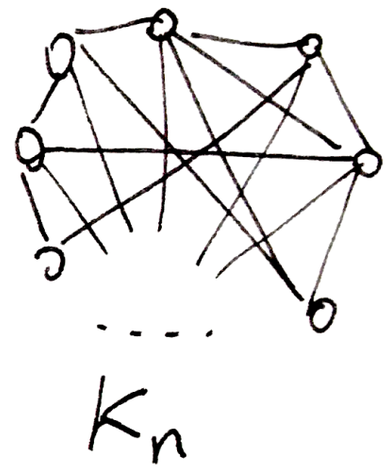
tree-width 2

tree-width $n-1$

Tree-width, examples

(a tree)

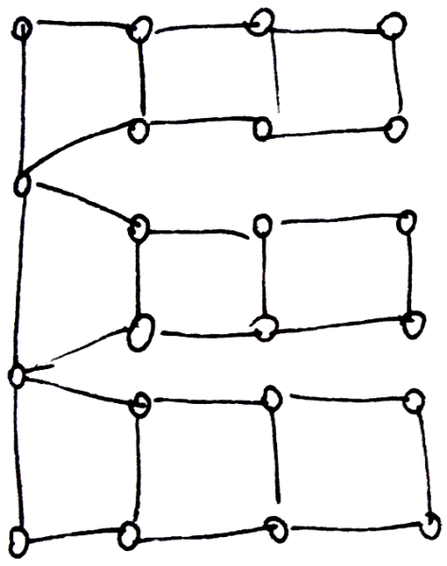


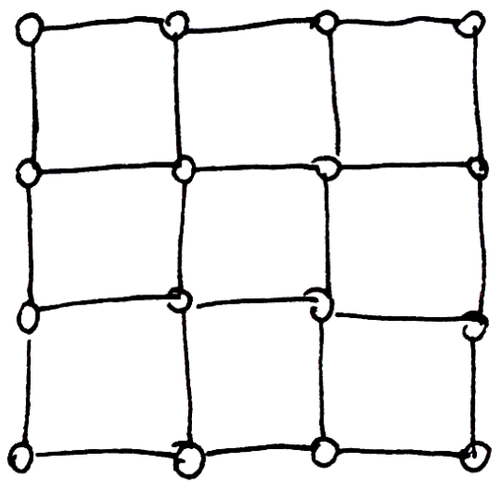


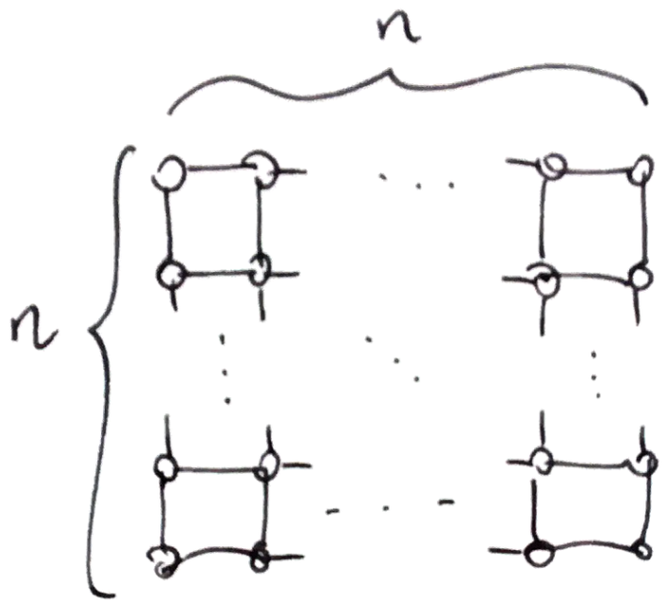
tree-width 1

tree-width 2

tree-width $n-1$







tree-width 2

tree-width 4

tree-width n

Tree-width of structures, queries

tree-width of CQ = tree-width of its canonical structure

tree-width of structure = tree-width of Gaifman graph

Tree-width of structures, queries

tree-width of CQ = tree-width of its canonical structure

tree-width of structure = tree-width of Gaifman graph

e.g.

tree-width($\exists\, x_1,x_2,x_3\, R(x_1,x_2) \wedge S(x_1,x_3) \wedge S(x_2,x_3)$)

Tree-width of structures, queries

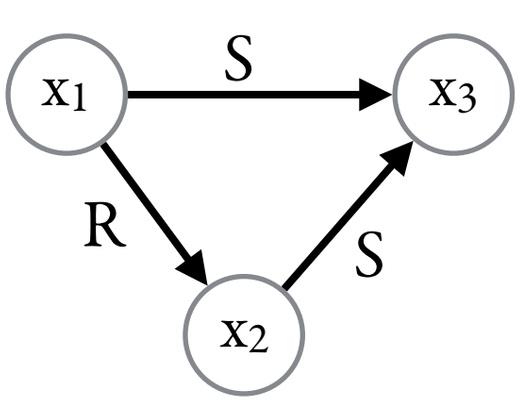
tree-width of CQ = tree-width of its canonical structure

tree-width of structure = tree-width of Gaifman graph

e.g.

tree-width($\exists\, x_1,x_2,x_3\; R(x_1,x_2) \wedge S(x_1,x_3) \wedge S(x_2,x_3)$)

||

tree-width(

)

Tree-width of structures, queries

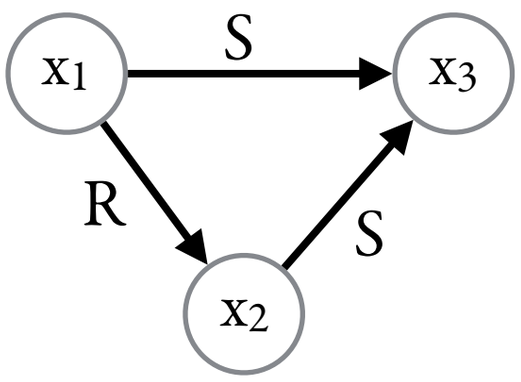
tree-width of CQ = tree-width of its canonical structure

tree-width of structure = tree-width of Gaifman graph

e.g.

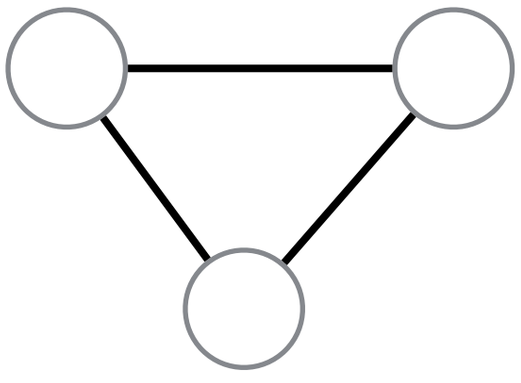
tree-width($\exists\, x_1,x_2,x_3\, R(x_1,x_2) \wedge S(x_1,x_3) \wedge S(x_2,x_3)$)

=

tree-width(

)

=

tree-width() = 2

Beyond acyclic CQ's

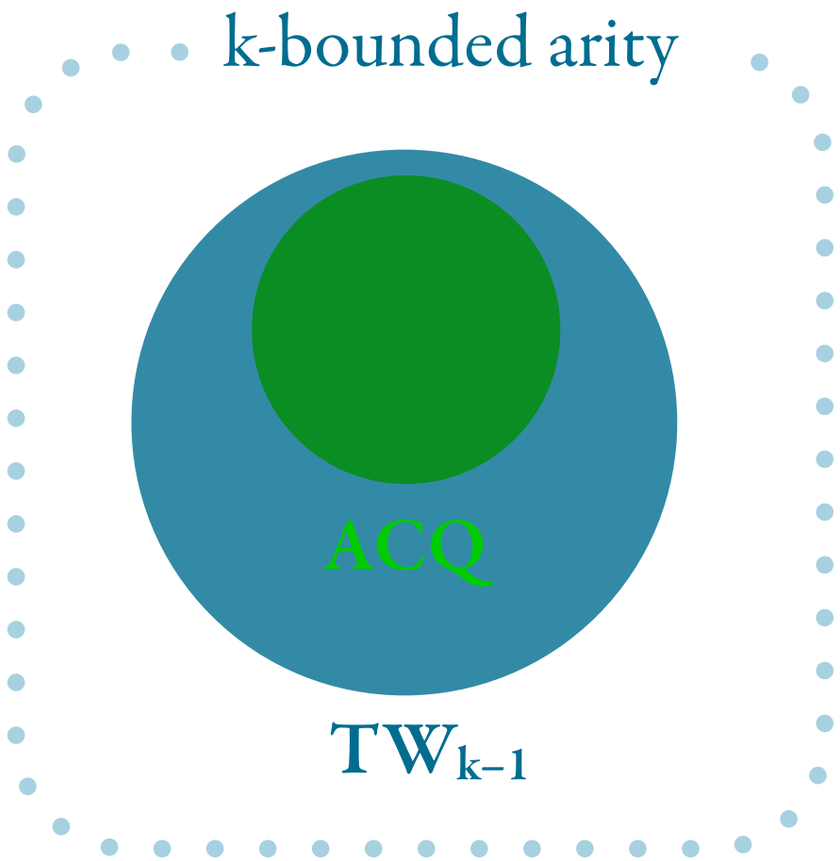
For a fixed k,

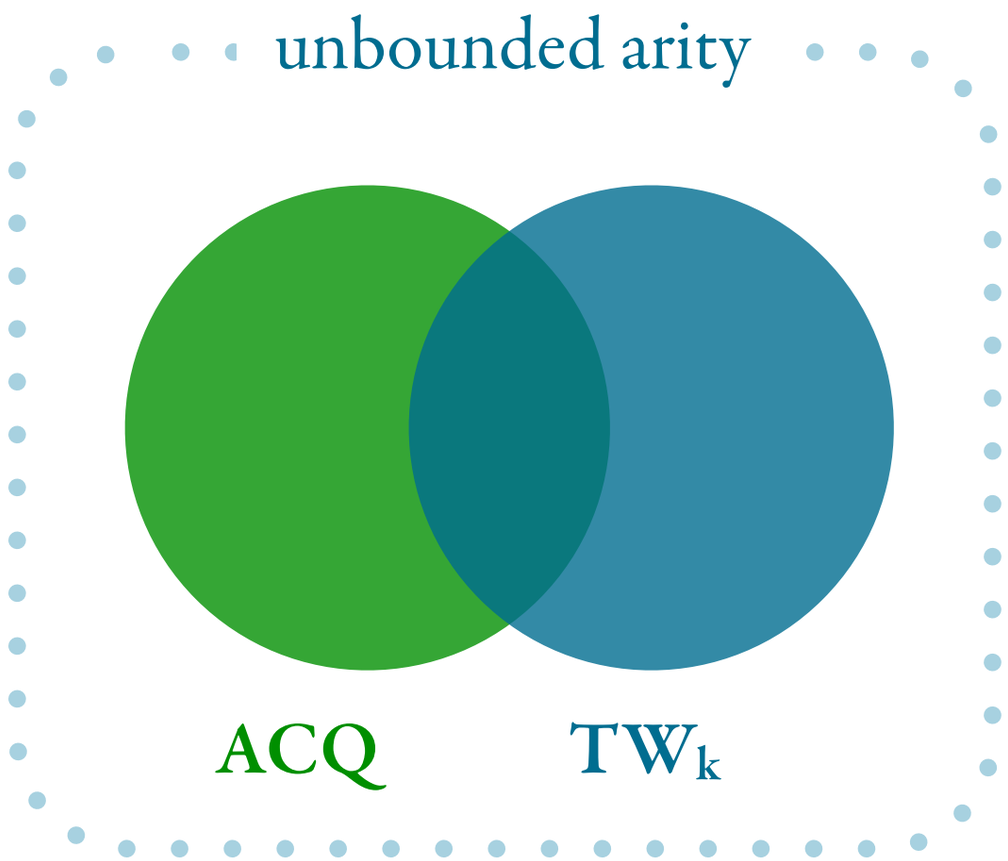
- computing whether $\phi \in CQ$ has tw $\leq k$
- calculating a tree decomposition

can be done in **linear time.**

[Bodlaender]

Tree-width vs. Acyclicity





Beyond acyclic CQ's

CQ's with bounded treewidth can be evaluated in PTIME

[Chekuri, Rajaraman, Gottlob, Leone, Scarcello]

Beyond acyclic CQ's

CQ's with bounded treewidth can be evaluated in PTIME

[Chekuri, Rajaraman, Gottlob, Leone, Scarcello]

CQ's can be evaluated in PTIME iff they have bounded tree width!

[Grohe, Schwentick, Segoufin]

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:

$$\ltimes,\ \cup,\ \pi,\ \sigma,\ \setminus,\ \textit{dupcol}$$

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

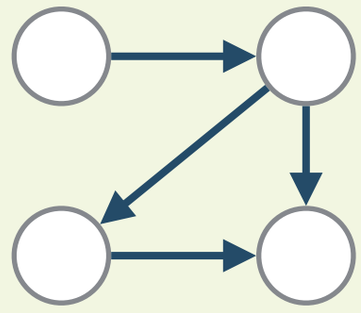
Def.

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

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

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

G



Bounded variable FO

Def.

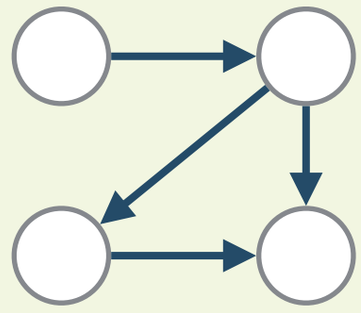
FO^k = The fragment of FO restricted to k variable names

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

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

Question: in FO^2 ?

G



Bounded variable FO

Def.

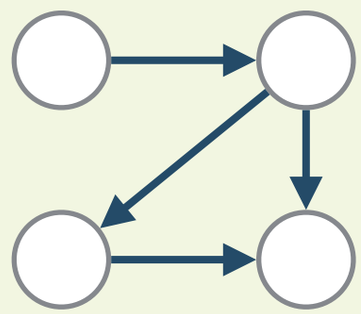
FO^k = The fragment of FO restricted to k variable names

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

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

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

G



Bounded variable FO

Def.

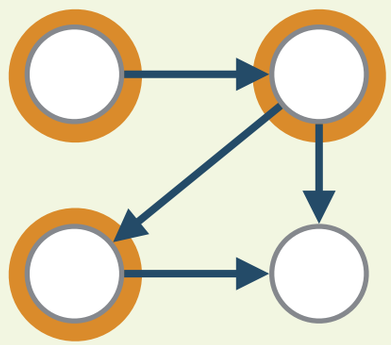
FO^k = The fragment of FO restricted to k variable names

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

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

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

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"

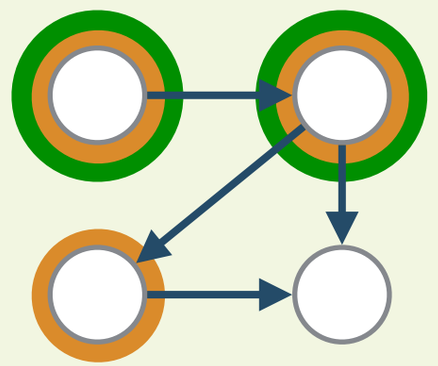
$= \forall y.\ \Big(\ \mathrm{E}(x,y) \implies \exists z\, \exists w\, \big(\ \mathrm{E}(y,z) \wedge \mathrm{E}(z,w)\ \big)\Big) \ \in \mathrm{FO}^4$

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

PTIME

PTIME

G



Bounded variable FO

Def.

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

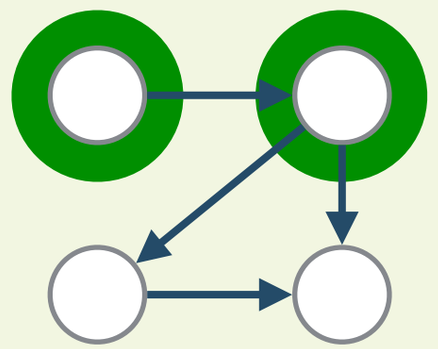
$\phi(x)$ = "Every neighbour of x has an outgoing path of length 2"

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

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

PTIME

G



Bounded variable FO

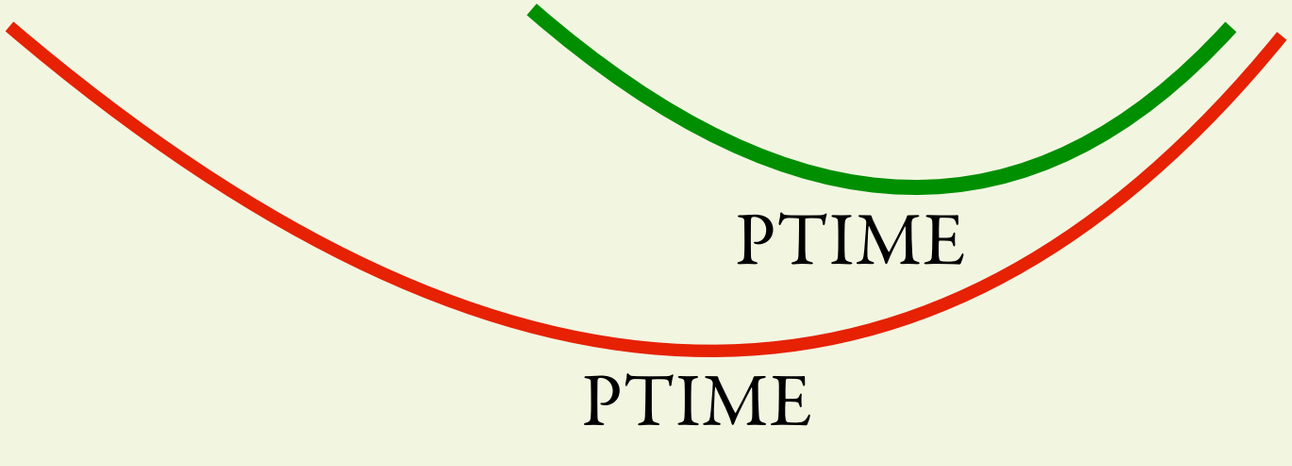
Def.

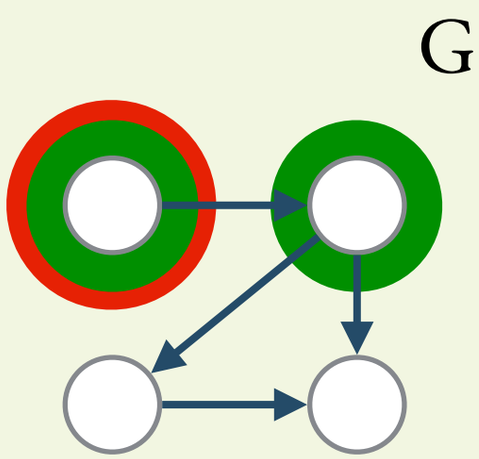
FO^k = The fragment of FO restricted to k variable names

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

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

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





Bounded variable FO

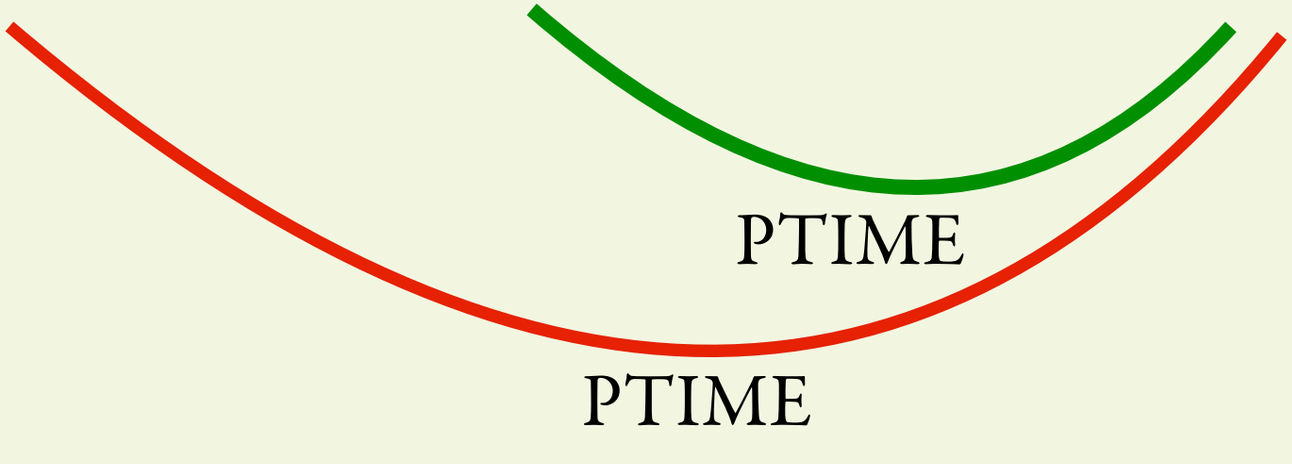
Def.

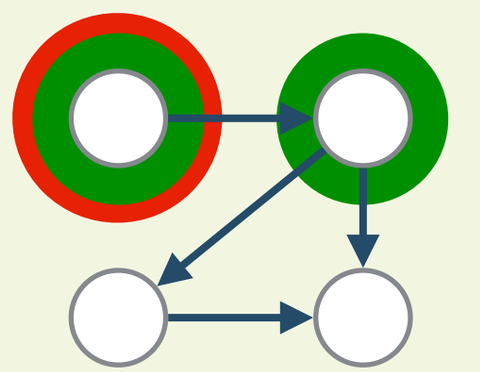
FO^k = The fragment of FO restricted to k variable names

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

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

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





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

⋱

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

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. . . .

⋮

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 $R_{0,\alpha}$

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

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 "$\exists x(\ldots\ \forall y(\ldots\ \exists x(\ldots\ \exists z(\ldots))))$"

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 $R_{0,\alpha}$

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

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 $R_{1,\gamma}$

$\rightsquigarrow |V|^k \cdot (|\gamma| \cdot (|G| + |R_2|))^p$

$\leq |V|^k$

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:

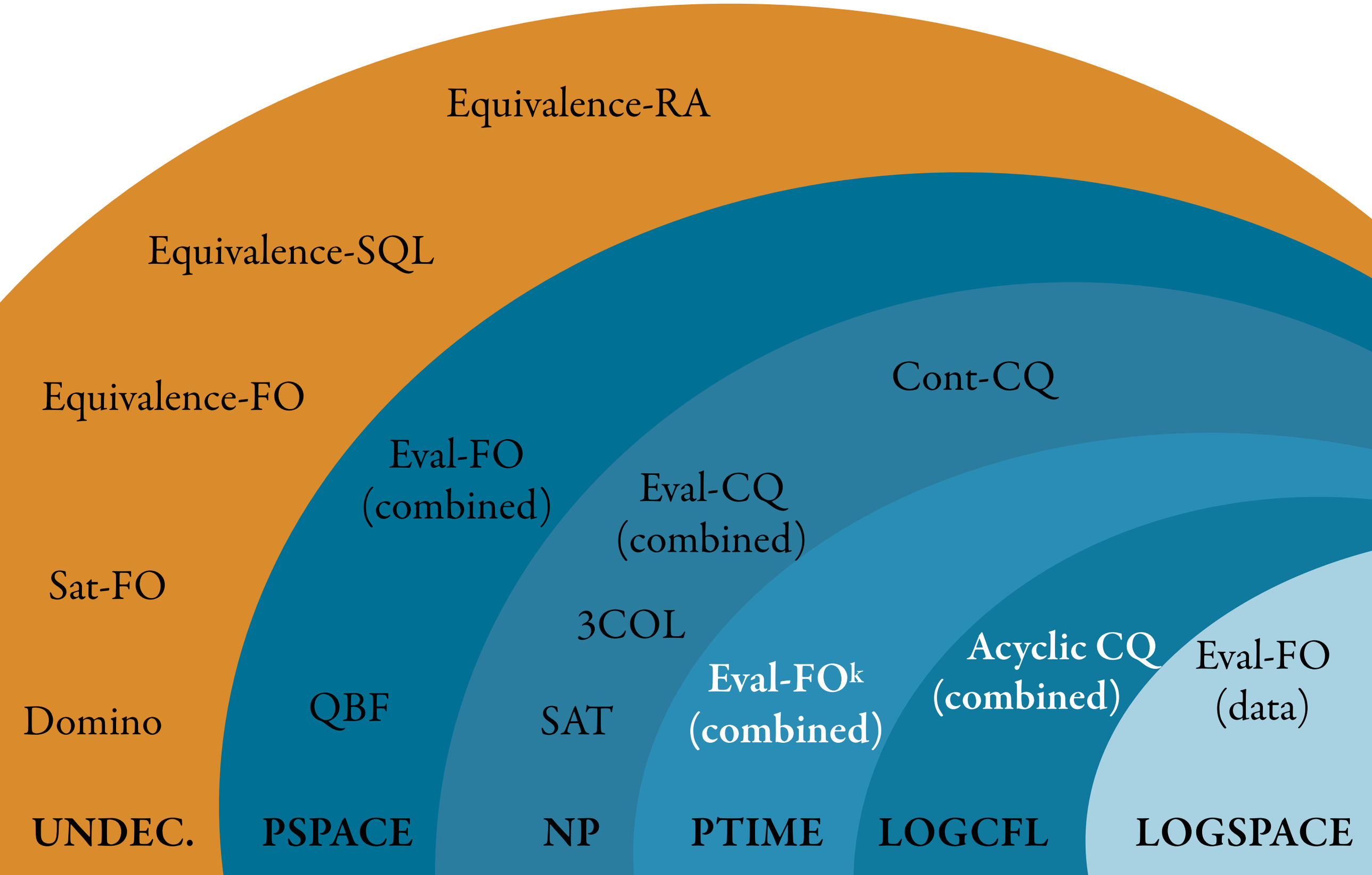
- Given k and a FO query ϕ, is ϕ in FO^k ? ⇝ 💀 Undecidable (even w.o. $\neg$)

- 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
Cont-CQ
Eval-FO
(combined)
Eval-CQ
(combined)
Sat-FO
3COL
Acyclic CQ
(combined)
Eval-FO
(data)
Eval-FO^k
(combined)
QBF
SAT
Domino
UNDEC.
PSPACE
NP
PTIME
LOGCFL
LOGSPACE

Definability in FO

Goal: check which properties / queries are *expressible in FO*

Definability in FO

Goal: check which properties / queries are *expressible in FO*

Example. $Q(G) = \{ (u,v) \mid G \text{ contains a path from } u \text{ to } v \}$

Is Q expressible as a first-order formula?

Definability in FO

Definition. **Quantifier rank** of ϕ = max number of nested quantifiers in ϕ.

Definability in FO

Definition. **Quantifier rank** of ϕ = max number of nested quantifiers in ϕ.

Example. $\phi = \forall x \forall y \Big(\neg E(x,y) \vee \exists z \Big(\big(E(x,z) \vee E(z,x)\big) \wedge \big(E(y,z) \vee E(z,y)\big) \Big) \Big)$

has quantifier rank 3.

Definability in FO

Definition. **Quantifier rank** of ϕ = max number of nested quantifiers in ϕ.

Example. $\phi = \forall x \forall y \Big(\neg E(x,y) \vee \exists z \big(\big(E(x,z) \vee E(z,x)\big) \wedge \big(E(y,z) \vee E(z,y)\big) \big) \Big)$

has quantifier rank 3.

Quantifier rank ≠ quantity of quantifiers

Definability in FO

Definition. **Quantifier rank** of ϕ = max number of nested quantifiers in ϕ.

Example. $\phi = \forall x \forall y \left(\neg E(x,y) \vee \exists z \left(\left(E(x,z) \vee E(z,x) \right) \wedge \left(E(y,z) \vee E(z,y) \right) \right) \right)$

has quantifier rank 3.

Quantifier rank ≠ quantity of quantifiers

Eg, in $\phi_0(x, y) = E(x, y)$, and

$\phi_k(x,y) = \exists z \left(\phi_{k-1}(x, z) \wedge \phi_{k-1}(z, y) \right)$

$qr(\phi_k) = k$ but # quantifiers of ϕ_k is 2^k

Definability in FO

Definition. **Quantifier rank** of ϕ = max number of nested quantifiers in ϕ.

Example. $\phi = \forall x \forall y \Big(\neg E(x,y) \vee \exists z \big((E(x,z) \vee E(z,x)) \wedge (E(y,z) \vee E(z,y)) \big) \Big)$

has quantifier rank 3.

Quantifier rank ≠ quantity of quantifiers

Eg, in $\phi_0(x, y) = E(x, y)$, and

$\phi_k(x,y) = \exists z \, (\phi_{k-1}(x, z) \wedge \phi_{k-1}(z, y))$

$qr(\phi_k) = k$ but # quantifiers of ϕ_k is 2^k

What does it define?

Definability in FO

Definition. **Quantifier rank** of ϕ = max number of nested quantifiers in ϕ.

Example. $\phi = \forall x \forall y \Big(\neg E(x,y) \vee \exists z \big((E(x,z) \vee E(z,x)) \wedge (E(y,z) \vee E(z,y)) \big) \Big)$

has quantifier rank 3.

Quantifier rank ≠ quantity of quantifiers

Eg, in $\phi_0(x, y) = E(x, y)$, and

$\phi_k(x,y) = \exists z \, (\phi_{k-1}(x, z) \wedge \phi_{k-1}(z, y))$

$qr(\phi_k) = k$ but # quantifiers of ϕ_k is 2^k

What does it define?

Quantifier rank is a **measure of complexity** of a formula

Definability in FO

Definition. **Quantifier rank** of ϕ = max number of nested quantifiers in ϕ.

Example. $\phi = \forall x \forall y \Big(\neg E(x,y) \vee \exists z \big((E(x,z) \vee E(z,x)) \wedge (E(y,z) \vee E(z,y)) \big) \Big)$

has quantifier rank 3.

Quantifier rank ≠ quantity of quantifiers

Eg, in $\phi_0(x, y) = E(x, y)$, and

$\phi_k(x,y) = \exists z \, (\phi_{k-1}(x, z) \wedge \phi_{k-1}(z, y))$

$qr(\phi_k) = k$ but # quantifiers of ϕ_k is 2^k

What does it define?

Quantifier rank is a **measure of complexity** of a formula

Sub-goal: Given a property P and a number *n*,
tell whether P is expressible by a sentence of quantifier rank at most *n*.

Definability in FO

Definition. Two structures S_1 and S_2 are ***n*-equivalent** iff they satisfy the same FO sentences of quantifier rank $\leq n$ (i.e. $S_1 \models \phi$ iff $S_2 \models \phi$ for all $\phi \in$ FO with $qr(\phi) \leq n$)

[Tarski '30]

Definability in FO

Definition. Two structures S_1 and S_2 are ***n*-equivalent**
iff
they satisfy the same FO sentences of quantifier rank $\leq n$
(i.e. $S_1 \models \phi$ iff $S_2 \models \phi$ for all $\phi \in$ FO with $qr(\phi) \leq n$)

[Tarski '30]

Consider a property (i.e. a set of structures) $\boldsymbol{P}$.

Suppose that there are $S_1 \in \boldsymbol{P}$, $S_2 \notin \boldsymbol{P}$ *s.t.*

S_1 and S_2 are *n-equivalent*.

Then $\boldsymbol{P}$ is *not expressible* by any sentence of quantifier rank n.

Definability in FO

Definition. Two structures S_1 and S_2 are ***n*-equivalent**
iff
they satisfy the same FO sentences of quantifier rank $\leq n$
(i.e. $S_1 \models \phi$ iff $S_2 \models \phi$ for all $\phi \in$FO with qr(ϕ)$\leq$n)

[Tarski '30]

Consider a property (i.e. a set of structures) ***P***.

Suppose that there are $S_1 \in \boldsymbol{P}$, $S_2 \notin \boldsymbol{P}$ *s.t.*

S_1 and S_2 are *n-equivalent*.

Then ***P*** is *not expressible* by any sentence of quantifier rank *n*.

Note: if the above happens $\forall\, n$, then **P** is not expressible by *any* FO sentence.

Definability in FO

Definition. Two structures S_1 and S_2 are ***n*-equivalent**
iff
they satisfy the same FO sentences of quantifier rank $\leq n$
(i.e. $S_1 \models \phi$ iff $S_2 \models \phi$ for all $\phi \in$FO with qr(ϕ)$\leq$n)

[Tarski '30]

Consider a property (i.e. a set of structures) ***P***.

Suppose that there are $S_1 \in \boldsymbol{P}$, $S_2 \notin \boldsymbol{P}$ *s.t.*

S_1 and S_2 are *n-equivalent*.

Then ***P*** is *not expressible* by any sentence of quantifier rank *n*.

Note: if the above happens $\forall\, n$, then **P** is not expressible by *any* FO sentence.

Example. **P** = { structures of even size } seems to be not FO-definable.
One could then aim at proving that

for all *n* there are $S_1 \in P$ and $S_2 \notin P$ s.t. S_1, S_2 *n*-equivalent...

Expressive power via games

Expressive power via games

Characterisation of the expressive power of FO in terms of **Games**

Expressive power via games

Characterisation of the expressive power of FO in terms of **Games**

Idea: For every two structures (S,S') there is a game where

a player of the game has a **winning strategy**

iff

S,S' are **indistinguishable**

Ehrenfeucht-Fraïssé games

A game between two players

Board: (S_1, S_2)

One player plays in one structure, the other player answers in the other structure.

If Duplicator can ensure not losing after n rounds: S_1, S_2 are n-equivalent

Ehrenfeucht-Fraïssé games

Definition. **Partial isomorphism** between S_1 and S_2 = injective partial map

$$f\colon \text{ nodes of } S_1 \longrightarrow \text{ nodes of } S_2$$

so that $E(x,y)$ **iff** $E(f(x),f(y))$

Ehrenfeucht-Fraïssé games

Definition. **Partial isomorphism** between S_1 and S_2 = injective partial map

$$f\colon \text{ nodes of } S_1 \longrightarrow \text{ nodes of } S_2$$

so that $E(x,y)$ **iff** $E(f(x),f(y))$

Spoiler and *Duplicator* play for n rounds on the board S_1, S_2

Ehrenfeucht-Fraïssé games

Definition. **Partial isomorphism** between S_1 and S_2 = injective partial map

$$f: \text{ nodes of } S_1 \longrightarrow \text{ nodes of } S_2$$

so that $E(x,y)$ **iff** $E(f(x),f(y))$

Spoiler and *Duplicator* play for n rounds on the board S_1, S_2

At each round i :

1. Spoiler chooses a node x_i from S_1

   and Duplicator answers with a node y_i from S_2,

Ehrenfeucht-Fraïssé games

Definition. **Partial isomorphism** between S_1 and S_2 = injective partial map

$$f: \text{ nodes of } S_1 \longrightarrow \text{ nodes of } S_2$$

so that $E(x,y)$ **iff** $E(f(x),f(y))$

Spoiler and *Duplicator* play for n rounds on the board S_1, S_2

At each round i :

1. Spoiler chooses a node x_i from S_1
   and Duplicator answers with a node y_i from S_2,

or

2. Spoiler chooses a node y_i from S_2
   and Duplicator answers with a node x_i from S_1,

Ehrenfeucht-Fraïssé games

Definition. **Partial isomorphism** between S_1 and S_2 = injective partial map

$$f: \text{ nodes of } S_1 \longrightarrow \text{ nodes of } S_2$$

so that $E(x,y)$ **iff** $E(f(x),f(y))$

Spoiler and *Duplicator* play for n rounds on the board S_1, S_2

At each round i :

1. Spoiler chooses a node x_i from S_1
   and Duplicator answers with a node y_i from S_2,

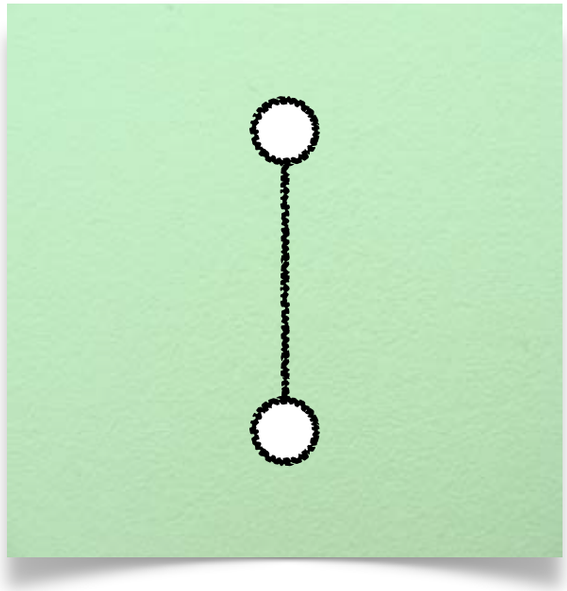
or

2. Spoiler chooses a node y_i from S_2
   and Duplicator answers with a node x_i from S_1,

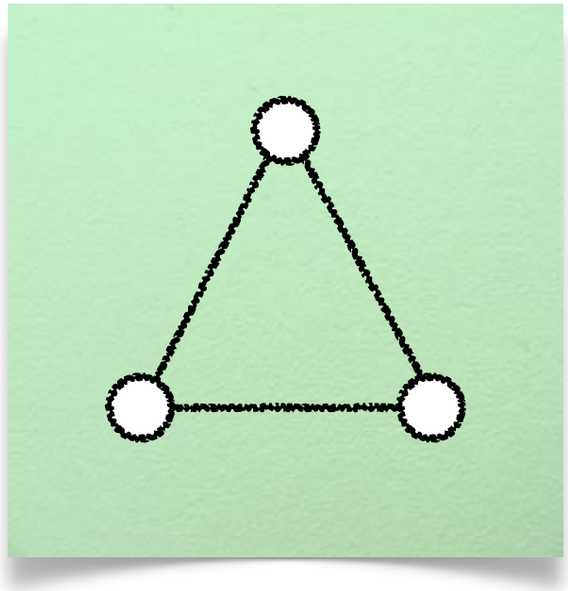
or Spoiler wins if $\{ x_i \mapsto y_i \mid 1 \le i \le n \}$ is **not a partial isomorphism** between S_1 and S_2.

Ehrenfeucht-Fraïssé games

S_1



S_2

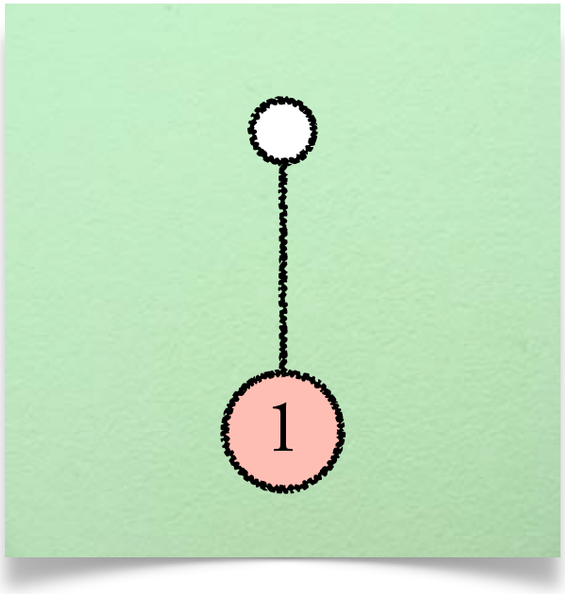


= Spoiler

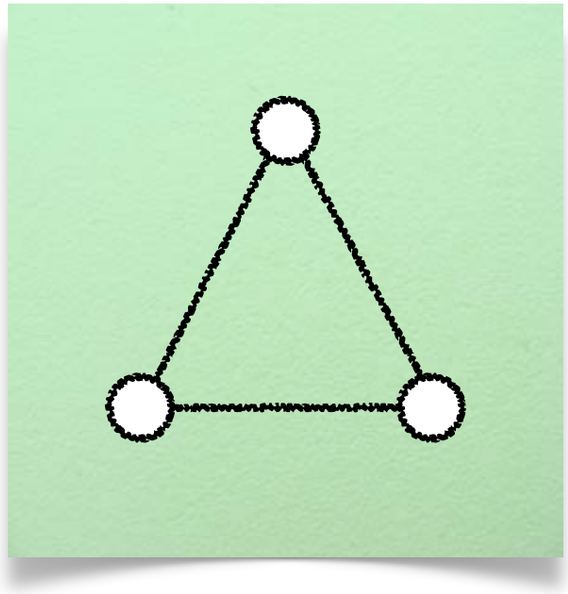
= Duplicator

Ehrenfeucht-Fraïssé games

S_1



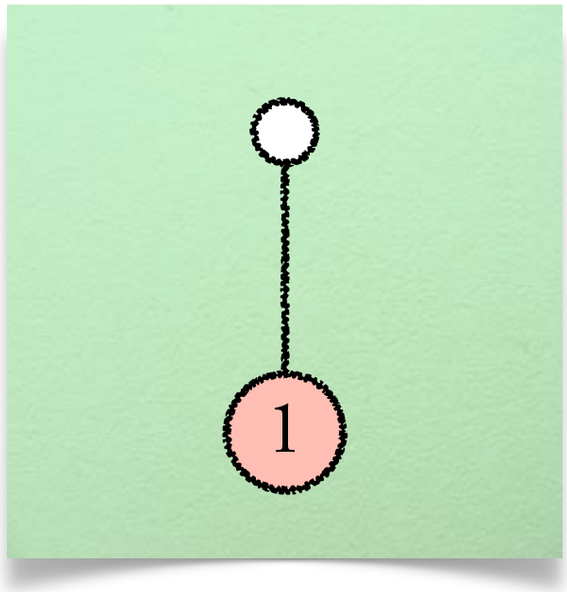
S_2



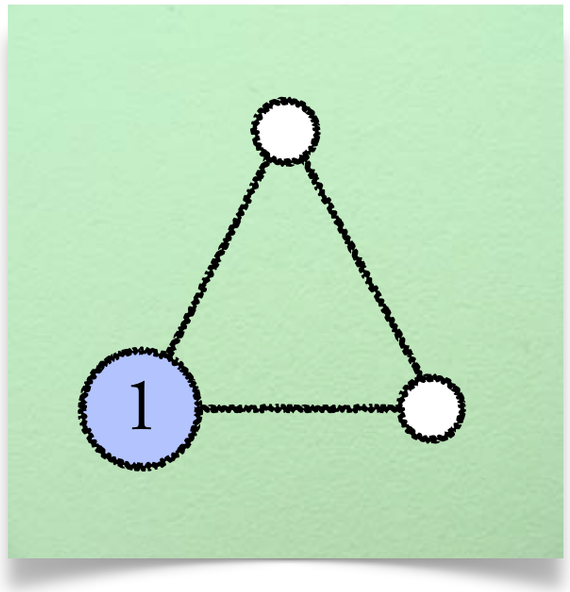
= Spoiler

= Duplicator

Ehrenfeucht-Fraïssé games



S_1



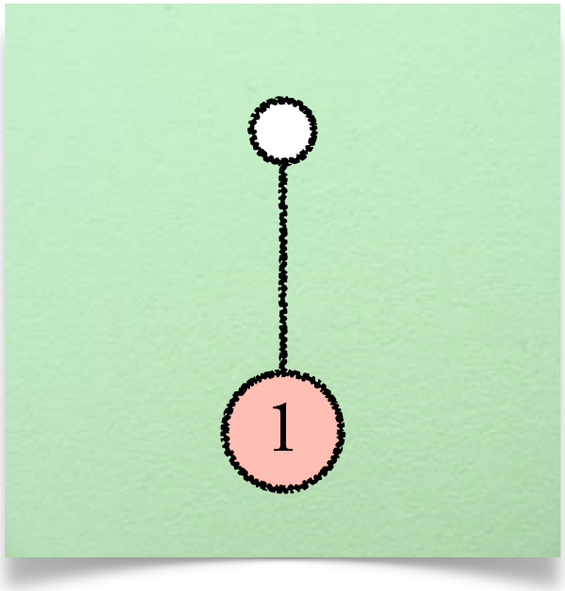
S_2

= Spoiler

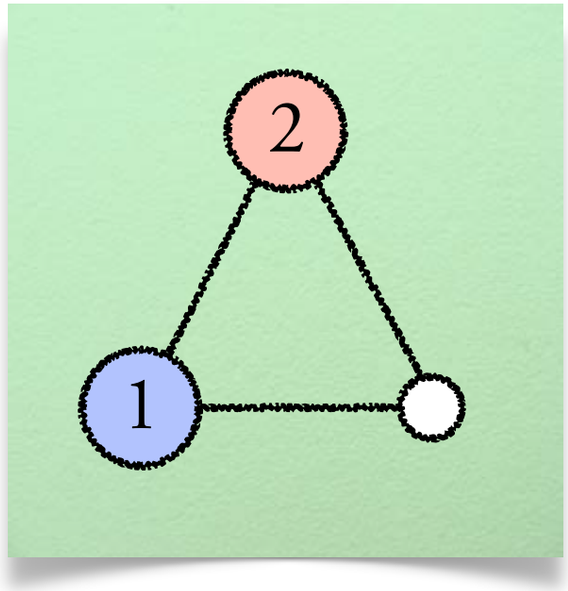
= Duplicator

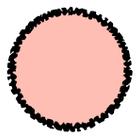
Ehrenfeucht-Fraïssé games

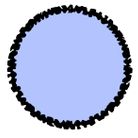
S_1



S_2

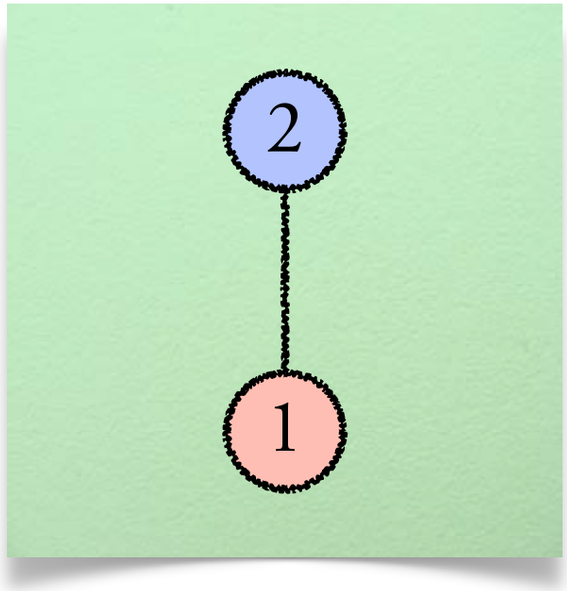



= Spoiler

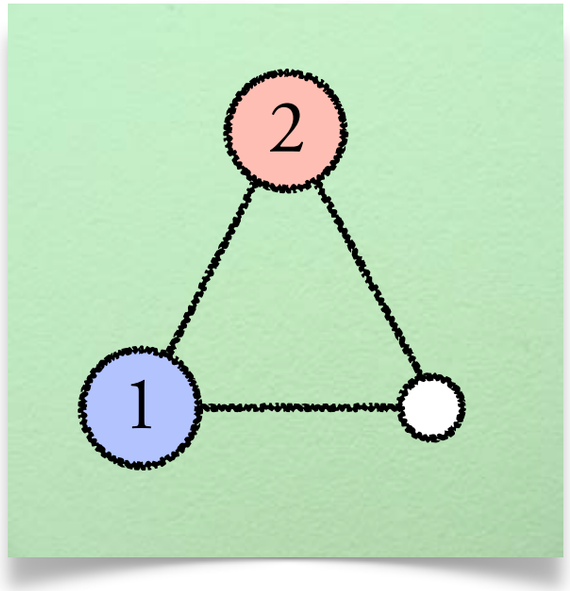

= Duplicator

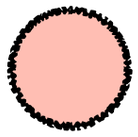
Ehrenfeucht-Fraïssé games

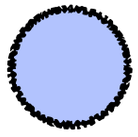
S_1



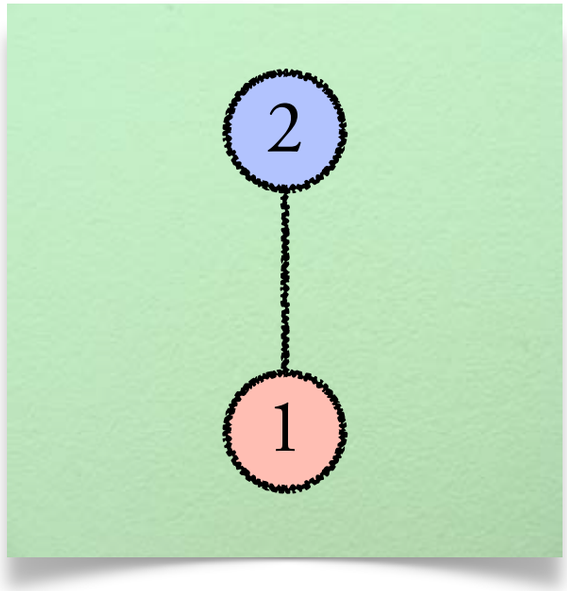
S_2



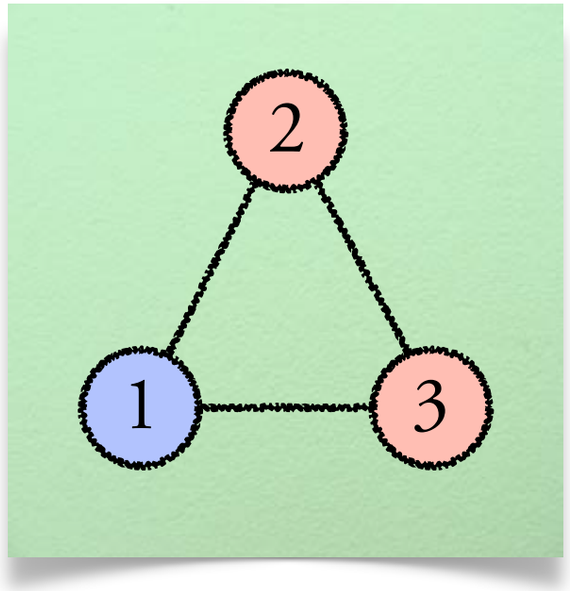

= Spoiler


= Duplicator

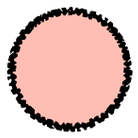
Ehrenfeucht-Fraïssé games



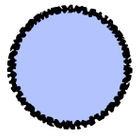
S_1



S_2



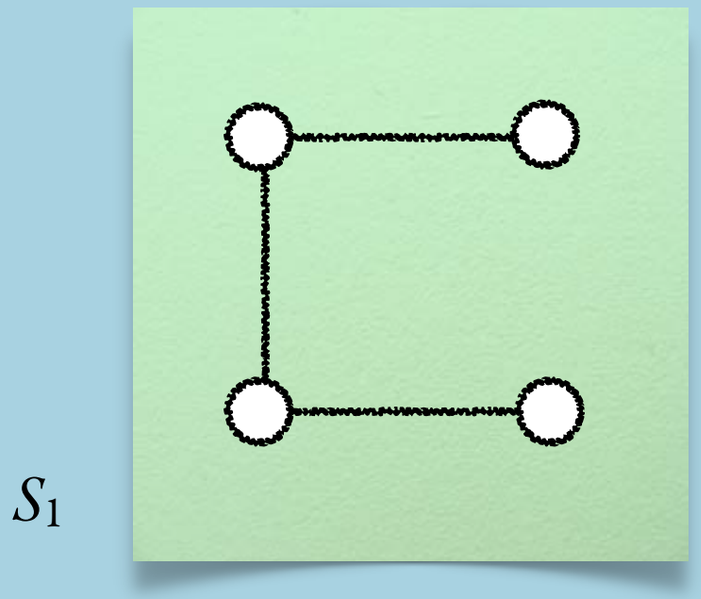
= Spoiler



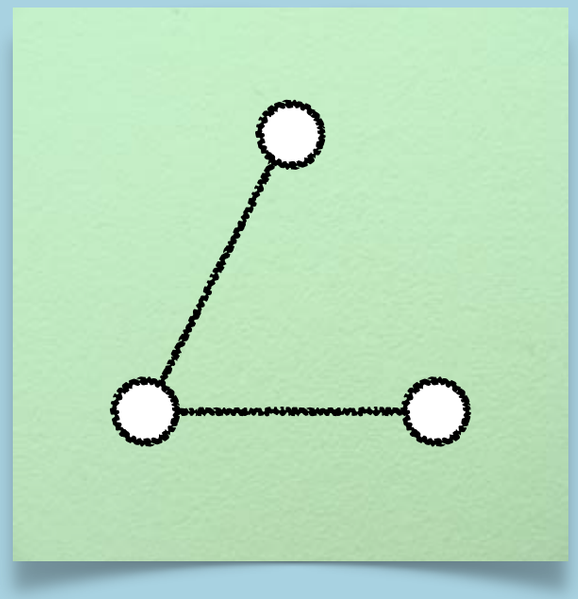
= Duplicator

Ehrenfeucht-Fraïssé games

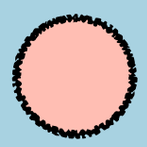
Question: Can Spoiler win in 3 rounds ?



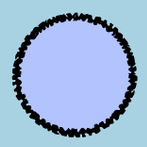
S_1



S_2



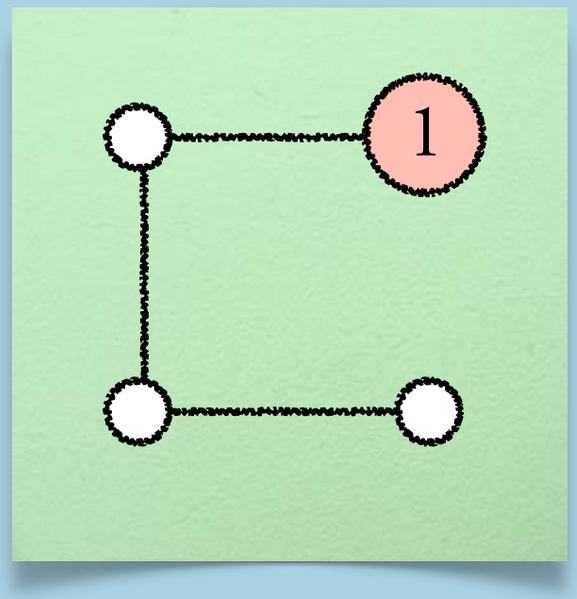
= Spoiler



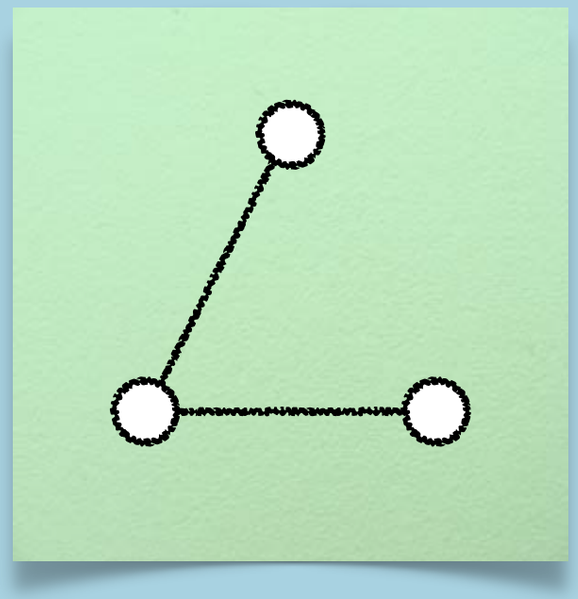
= Duplicator

Ehrenfeucht-Fraïssé games

Question: Can Spoiler win in 3 rounds ?



S_1



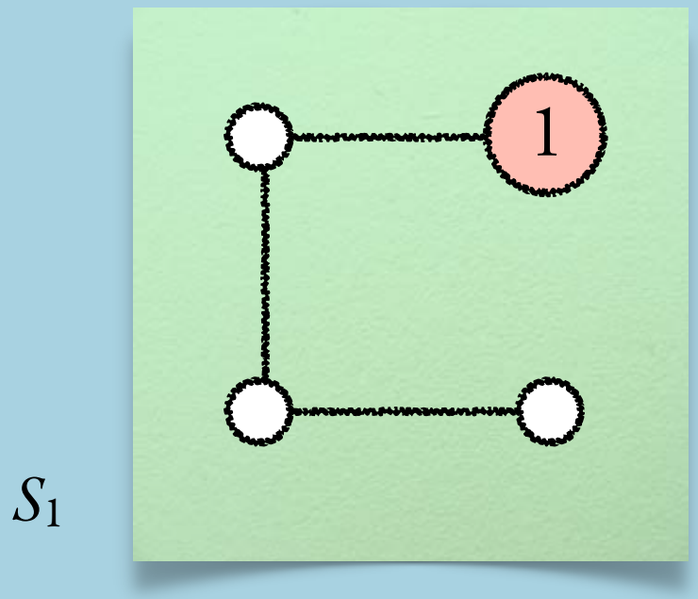
S_2

= Spoiler

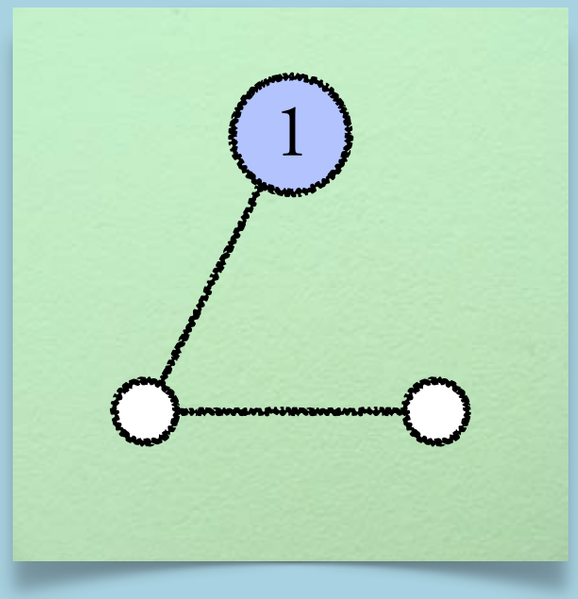
= Duplicator

Ehrenfeucht-Fraïssé games

Question: Can Spoiler win in 3 rounds ?



S_1



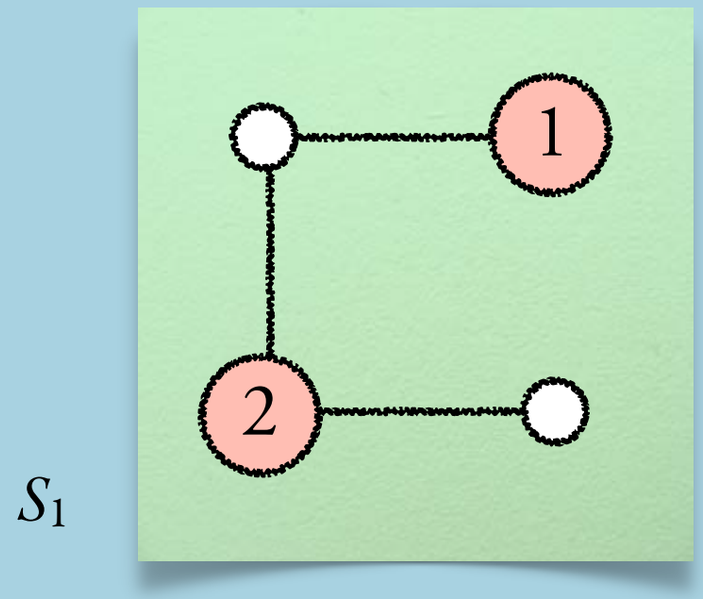
S_2

= Spoiler

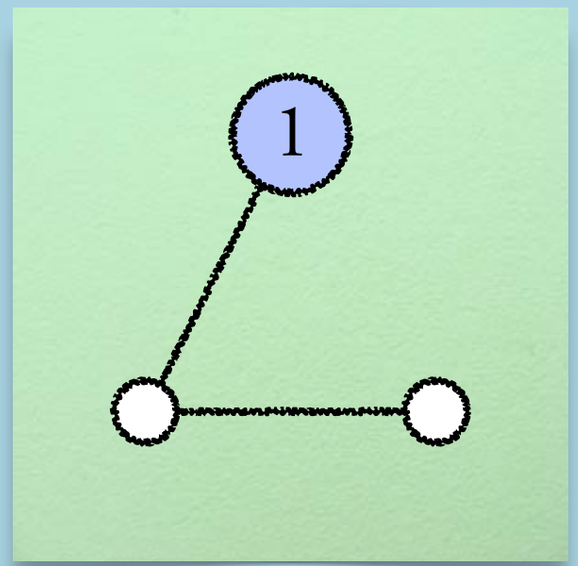
= Duplicator

Ehrenfeucht-Fraïssé games

Question: Can Spoiler win in 3 rounds ?



S_1



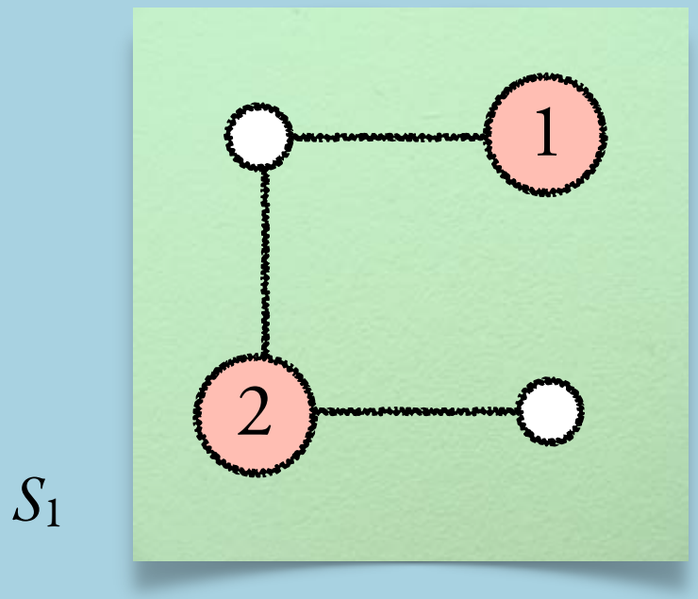
S_2

= Spoiler

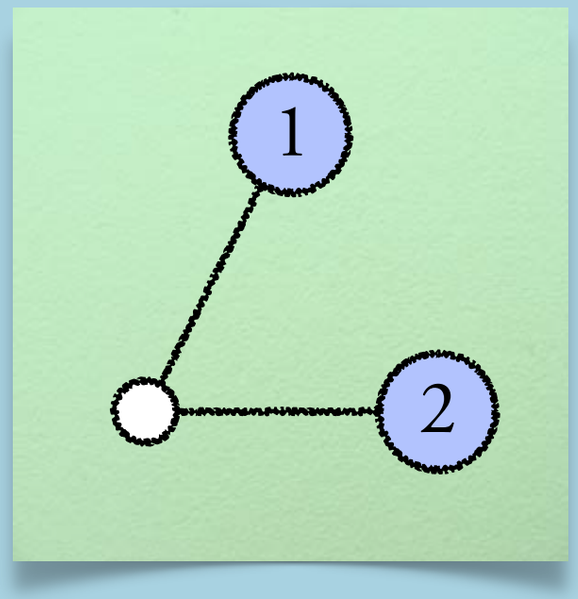
= Duplicator

Ehrenfeucht-Fraïssé games

Question: Can Spoiler win in 3 rounds ?



S_1



S_2

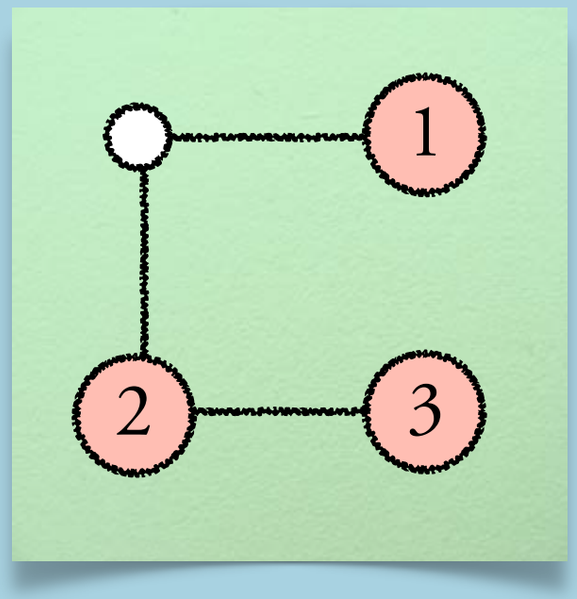
= Spoiler

= Duplicator

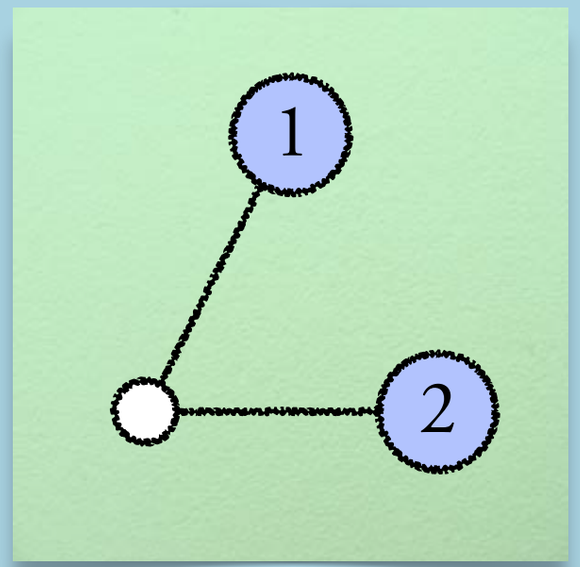
Ehrenfeucht-Fraïssé games

Question: Can Spoiler win in 3 rounds ?

S_1



S_2



= Spoiler

= Duplicator