Updated 2020/2/9

T2: Complexity of Query Evaluation
L10: Query minimization & nested queries

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp20)

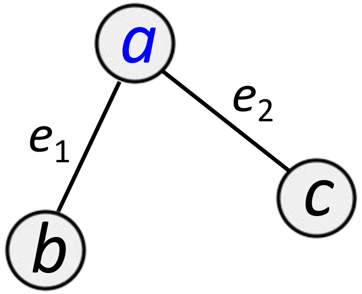
https://northeastern-datalab.github.io/cs7240/sp20/

Date: 2020/2/7

Absorption (or the challenge with self-joins)

f is true if there is an edge

$f = \exists x,y.\ x \land y \land (x,y) \in \mathrm{E}$

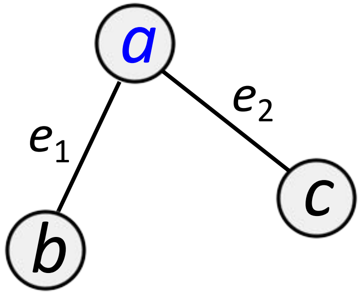


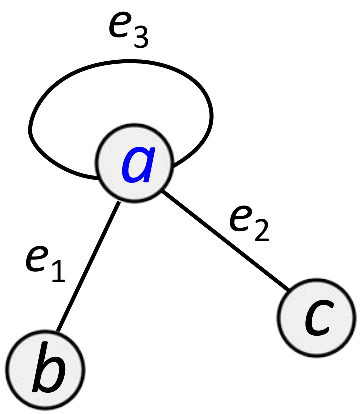
$f =$?

Absorption (or the challenge with self-joins)

f is true if there is an edge

$f = \exists x,y.\ x \wedge y \wedge (x,y) \in \mathrm{E}$





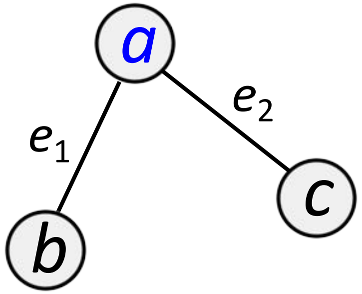
$f = ab \vee ac$

$f =$?

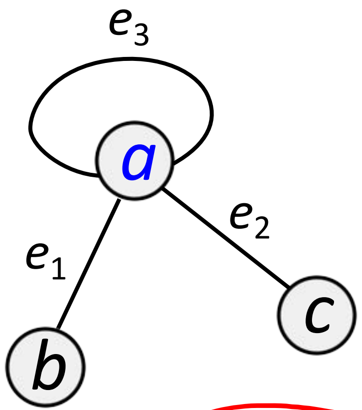
Absorption (or the challenge with self-joins)

f is true if there is an edge

$f = \exists x,y.\ x \wedge y \wedge (x,y) \in E$



$f = ab \vee ac$



$f = a = ab \vee ac \vee aa$ *absorption*

$(\cancel{\varphi_1} \vee \varphi_2) \wedge (\varphi_1 \Rightarrow \varphi_2)$

$\cancel{a}c \vee a \quad \wedge (ac \Rightarrow a)$

Absorption

Absorption

$(A \vee B) \wedge A = A$

$(A \vee B) \wedge (A \Rightarrow B) = A$

$(A \min B) \max A = A$

$(A \min B) = A$, if $A \leq B$

$A \leq B \Leftrightarrow A \min B = A$

$\sim A \wedge B = A$

Two binary operations, ∨ and ∧, are said to be connected by the absorption law if:

$a \vee (a \wedge b) = a \wedge (a \vee b) = a$.

A set equipped with two **commutative**, **associative** and **idempotent** binary operations ∨ ("join") and ∧ ("meet") that are connected by the absorption law is called a **lattice**.
Examples of lattices include Boolean algebras, the set of sets with union and intersection operators, and ordered sets with min and max operations.
https://en.wikipedia.org/wiki/Absorption_law

Outline: Complexity of Query Equivalence

- Query equivalence and query containment
  - Graph homomorphisms
  - Homomorphism beyond graphs
  - CQ containment
  - Beyond CQs
  - CQ equivalence under bag semantics
  - CQ minimization
  - Nested queries
  - Tree pattern queries

Exercise: Find the Homomorphisms

Order of subgoals in the query does not matter (thus ~sets)

q_1: {E(x,y),E(y,z),E(z,w)}

q_2: {E(x,y),E(y,z),E(z,x)}

q_3: {E(x,y),E(y,x)}

What is the containment relation between these queries ?

q_4: {E(x,y),E(y,x),E(y,y)}

q_5: {E(x,x)}

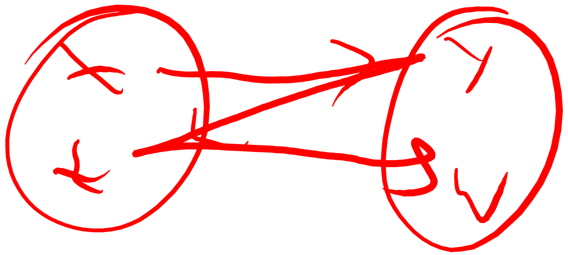
Example by Andreas Pieris

Exercise: Find the Homomorphisms

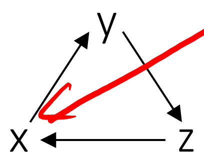
$q_2 \Rightarrow q_1$

q_1: {E(x,y),E(y,z),E(z,w)}

$x \rightarrow y \rightarrow z \rightarrow w$



q_2: {E(x,y),E(y,z),E(z,x)}

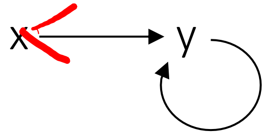


q_3: {E(x,y),E(y,x)}

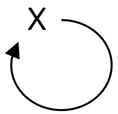
$x \leftrightarrow y$

What is the containment relation between these queries ?

q_4: {E(x,y),E(y,x),E(y,y)}



q_5: {E(x,x)}



Exercise: Find the Homomorphisms

q_1: {E(x,y),E(y,z),E(z,w)}

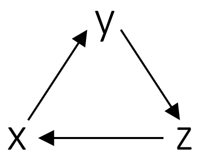
$x \longrightarrow y \longrightarrow z \longrightarrow w$

{x⟶x, y⟶y, z⟶z, w⟶x}

or {x⟶y, y⟶z, z⟶x, w⟶y}, etc.

{x⟶x, y⟶y, z⟶x, w⟶y}

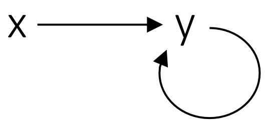
q_2: {E(x,y),E(y,z),E(z,x)}



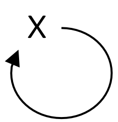
q_3: {E(x,y),E(y,x)}

$x \longleftrightarrow y$

q_4: {E(x,y),E(y,x),E(y,y)}

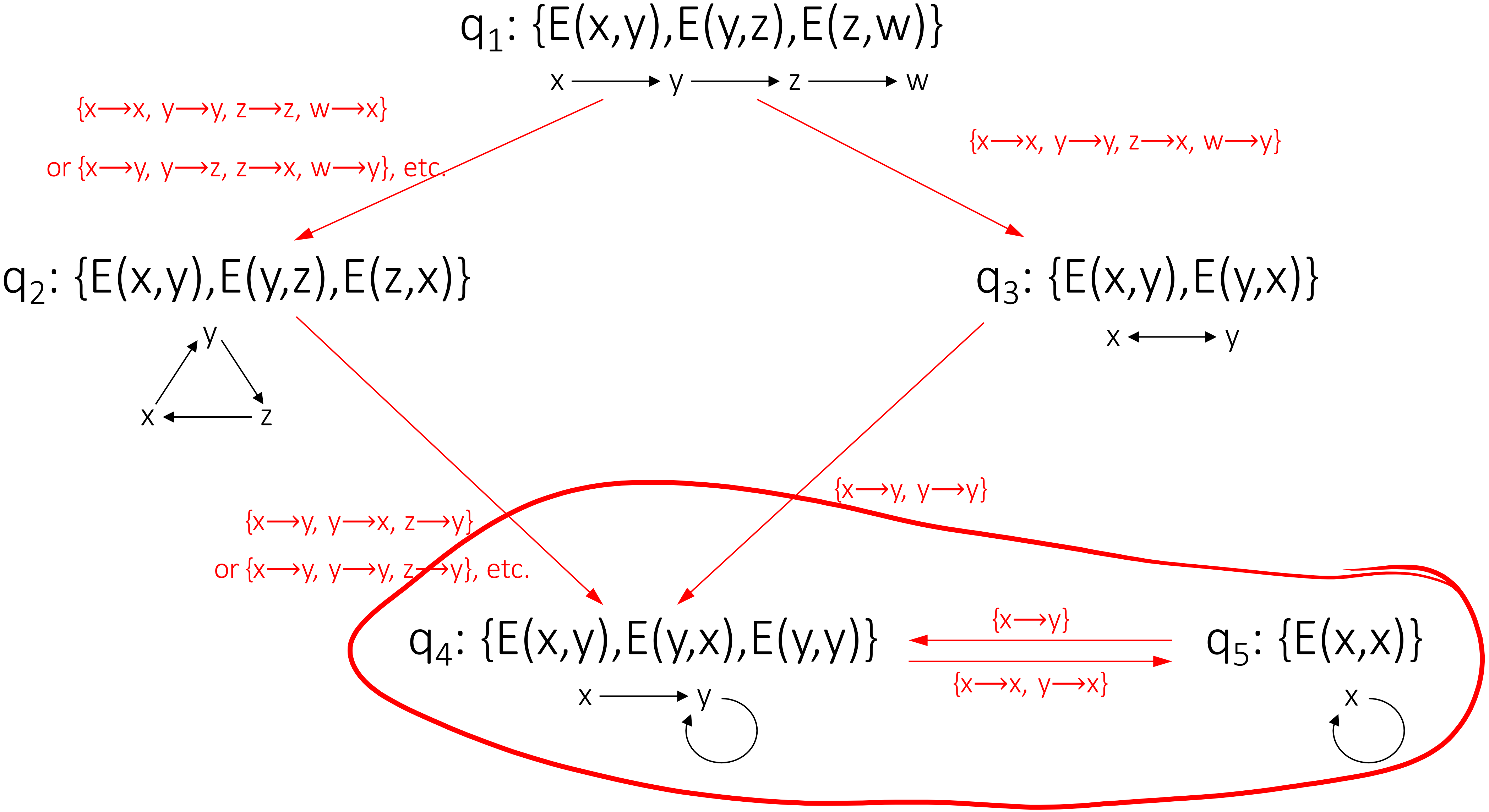


q_5: {E(x,x)}



Example by Andreas Pieris

Exercise: Find the Homomorphisms



Example by Andreas Pieris

Exercise: Find the Homomorphisms

q_1: {E(x,y),E(y,z),E(z,w)}

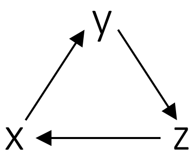
$x \longrightarrow y \longrightarrow z \longrightarrow w$

{x⟶x, y⟶y, z⟶z, w⟶x}

or {x⟶y, y⟶z, z⟶x, w⟶y}, etc.

{x⟶x, y⟶y, z⟶x, w⟶y}

q_2: {E(x,y),E(y,z),E(z,x)}



q_3: {E(x,y),E(y,x)}

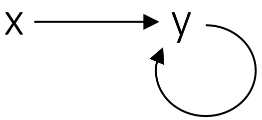
$x \longleftrightarrow y$

{x⟶y, y⟶y}

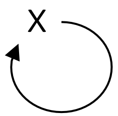
{x⟶y, y⟶x, z⟶y}

or {x⟶y, y⟶y, z⟶y}, etc.

q_4: {E(x,y),E(y,x),E(y,y)}



q_5: {E(x,x)}



Query Homeomorphism Practice

$q_1(x,y)$:- $R(x,u),R(v,u),R(v,y)$ $\quad$ $var(q_1) = \{x, u, v, y\}$

$q_2(x,y)$:- $R(x,u),R(v,u),R(v,w),R(t,w),R(t,y)$ $\quad$ $var(q_2) = \{x, u, v, w, t, y\}$

Are these queries equivalent ?

Query Homeomorphism Practice

$q_1 \subseteq q_2$

$\boxed{R \; A \; R}$

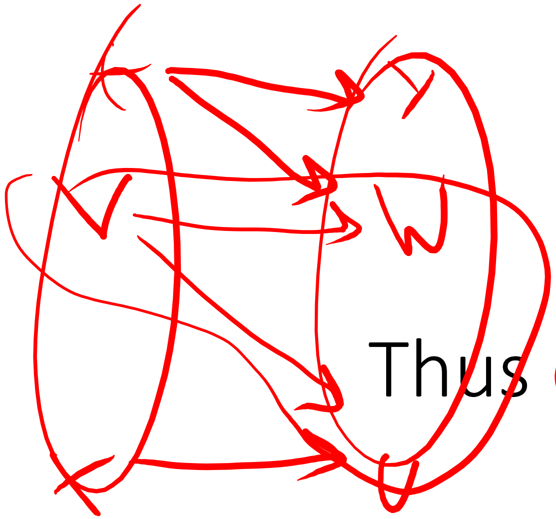
$q_1(x,y)$:- $R(x,u), R(v,u), R(v,y)$

$q_2(x,y)$:- $R(x,u), R(v,u), R(v,w), R(t,w), R(t,y)$

var(q_1) = {x, u, v, y}

var(q_2) = {x, u, v, w, t, y}

$q_2 \subseteq q_1$



$1 \to 2$

$\searrow 3$

Thus $q_1 \subseteq q_2$

:- R(x,u)

Minimizing Conjunctive Queries

- Goal: minimize the number of joins in a query
- Definition: A conjunctive query Q is minimal if there is no conjunctive query Q’ such that:
    1. Q ≡ Q’
    2. Q’ has fewer atoms than Q
- The task of CQ minimization is, given a conjunctive query Q, to compute a minimal one that is equivalent to Q

Minimization by Deletion

Theorem: Consider a conjunctive query $Q_1(x_1,\ldots,x_k)$:- $body_1$.

If Q_1 is equivalent to a conjunctive query $Q_2(y_1,\ldots,y_k)$:- $body_2$

where $|body_2| < |body_1|$, then Q_1 is equivalent to a query

$Q_3(x_1,\ldots,x_k)$:- $body_3$ such that $body_3 \subseteq body_1$

The above theorem says that to minimize a conjunctive query $Q_1(\mathbf{x})$:- body we simply need to remove some atoms from body

Can we shown by exploiting the homomorphism theorem…

Conjunctive query minimization algorithm

Minimize(Q(**x**) :- body)

- Repeat {
  - Choose an atom α ∈ body
  - Remove α from Q; let Q' be the new query
  - If there is a homomorphism from Q' to Q, then body := body \ {α}
- Until no atom can be removed}

$Q \subseteq Q'$

$Q' \subseteq Q$

Notice: the order in which we inspect subgoals doesn't matter

Minimization Procedure: Example

a,b,c,d are constants

Q(x) :- R(x,y), R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d)

Is this query minimal ?

Minimization Procedure: Example

a,b,c,d are constants

Q(x) :- R(x,y), R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d)

$\{y \longrightarrow b\}$

Q(x) :- R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d)

Is this query minimal ?

Minimization Procedure: Example

a,b,c,d are constants

Q(x) :- R(x,y), R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d)

{y⟶b}

Q(x) :- R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d)

{v⟶c}

Q(x) :- R(x,b), R(a,b), R(u,c), S(a,c,d)

Is this query minimal ?

Minimization Procedure: Example

a,b,c,d are constants

Q(x) :- R(x,y), R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d)

{y⟶b}

Q(x) :- R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d)

{v⟶c}

Q(x) :- R(x,b), R(a,b), R(u,c), S(a,c,d)

{x⟶a}

Q(a) :- R(a,b), R(u,c), S(a,c,d)

Is this query minimal ?

Minimization Procedure: Example

a,b,c,d are constants

Q(x) :- R(x,y), R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d)

{y⟶b}

Q(x) :- R(x,b), R(a,b), R(u,c), R(u,v), S(a,c,d)

{v⟶c}

Q(x) :- R(x,b), R(a,b), R(u,c), S(a,c,d)

Minimal query

{x⟶a}

Q(a) :- ~~R(a,b), R(u,c), S(a,c,d)~~

Mapping x⟶a is not valid since x is a distinguished variable

Uniqueness of Minimal Queries

$Q^1 \text{ :- } R(x,y)$ R 1 2

$Q^2 \text{ :- } R(y,z)$

~~$Q^1 \text{ :- } S(x,y)$~~   S 3 4

Natural question: does the order in which we remove atoms from the body of the input conjunctive query matter?

Theorem: Consider a conjunctive query Q. Let Q_1 and Q_2 be minimal conjunctive queries such that $Q_1 \equiv Q$ and $Q_2 \equiv Q$. Then, Q_1 and Q_2 are isomorphic (i.e., they are the same up to variable renaming)

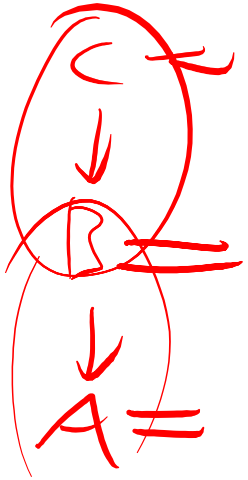
Therefore, given a conjunctive query Q, the result of Minimization(Q) is unique (up to variable renaming) and is called the core of Q

Query Minimization for Views

Employee(name, university, manager)

This query is minimal

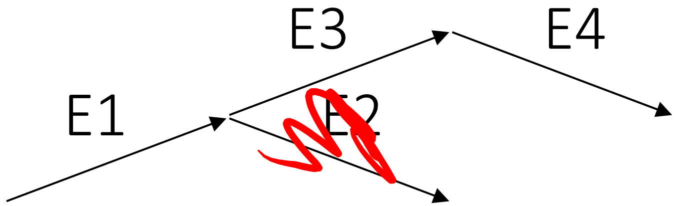
```
CREATE VIEW NeuMentors

SELECT DISTINCT E1.name,E1.manager
FROM Employee E1, Employee E1
WHERE E1.manger = E2.name
AND E1.university = ‘Northeastern’
AND E2.university= ‘Northeastern’
```



This query is no longer redundant!

This query is minimal

```
SELECT DISTINCT N1.name
FROM NeuMentors N1, NeuMentors N1
WHERE N1.manger = N2.name
```



View expansion (when you run a SQL query on a view)

SELECT DISTINCT E1.name
FROM Employee E1, Employee E2, Employee E3, Employee E4
WHERE ~~E1.manger = E2.name AND~~ E1.manger = E3.name AND E3.manger = E4.name
AND E1.university = ‘Northeastern’ AND ~~E2.university = ‘Northeastern’~~
AND E3.university = ‘Northeastern’ AND E4.university = ‘Northeastern’

Example adopted from Dan Suciu

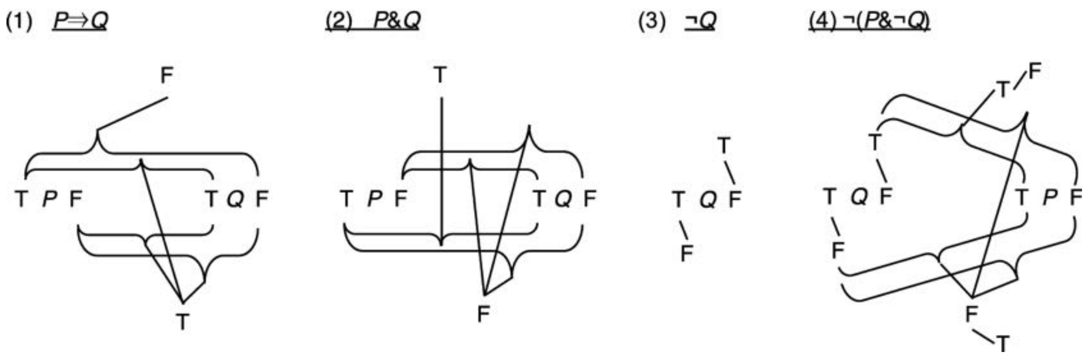
Outline: Complexity of Query Equivalence

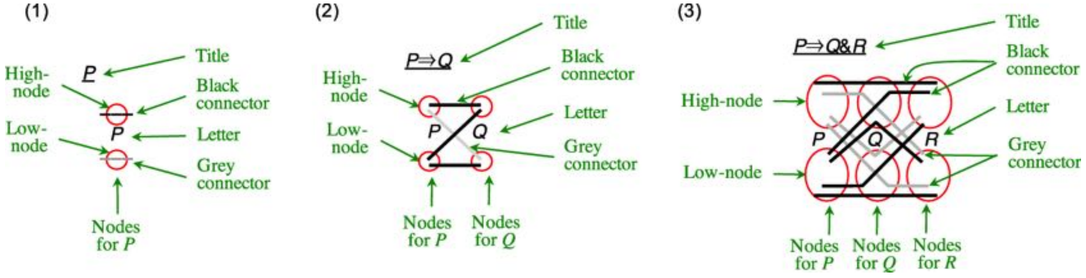
- Query equivalence and query containment
  - Graph homomorphisms
  - Homomorphism beyond graphs
  - CQ containment
  - Beyond CQs
  - CQ equivalence under bag semantics
  - CQ minimization
  - Nested queries
  - Tree pattern queries

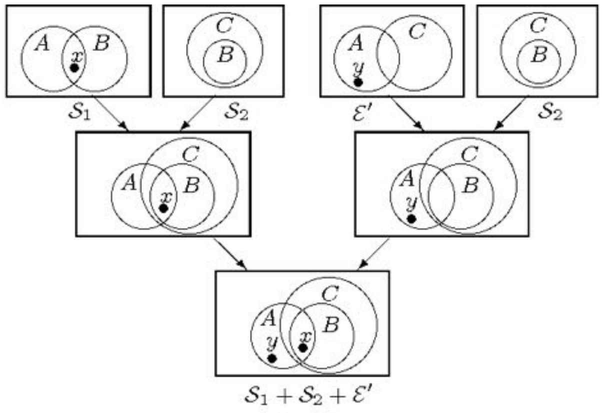
Equivalence of nested queries

- Query equivalence is one of the foundational questions in database theory (and practice?)
  - touches on logics and decidability
  - what modifications allow tractability
- Lots of work (and open questions) on query equivalence
  - But not so much on nested queries!
- Related to QueryViz project (http://queryviz.com) and two foundational questions on visual formalism:
  1. When can visual formalism *unambiguously* express logical statements?
  2. When can equivalent logical statements be transformed to each other by a sequence of visual transformations? (*Query equivalence*)

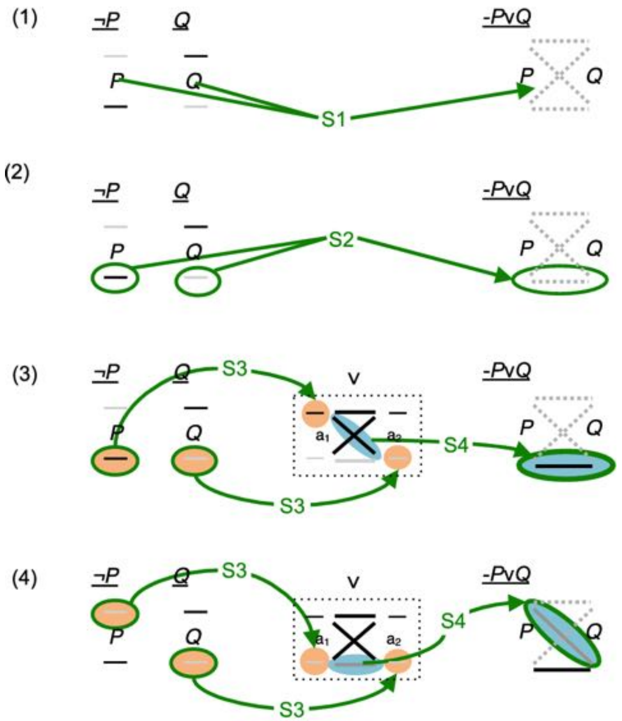
Diagrammatic reasoning systems and their expressiveness

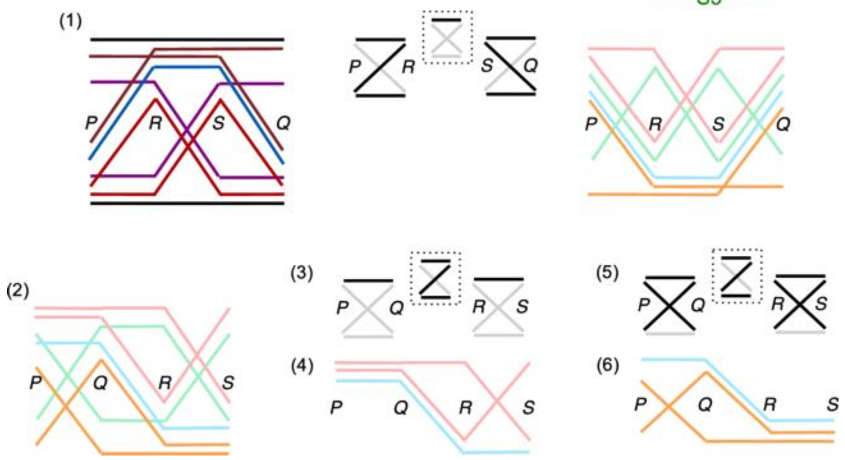




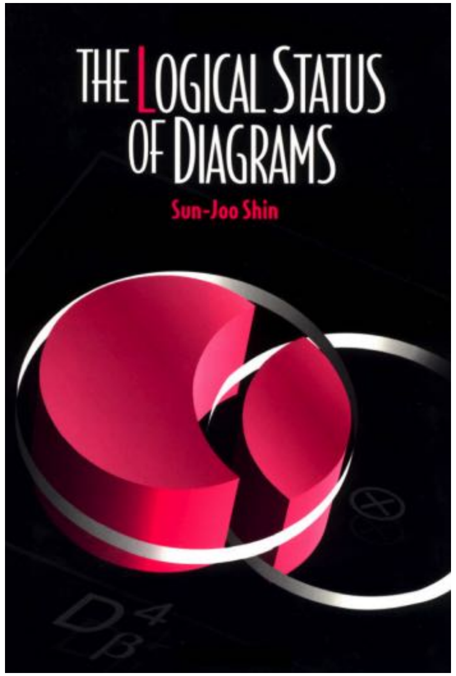


onstruction of a counter-diagram (Example 11)





Diagrammatic reasoning systems and their expressiveness



Diagrams are widely used in reasoning about problems in physics, mathematics, and logic, but have traditionally been considered to be only heuristic tools and not valid elements of mathematical proofs. This book challenges this prejudice against visualization in the history of logic and mathematics and provides a formal foundation for work on natural reasoning in a visual mode.

The author presents Venn diagrams as a formal system of representation equipped with its own syntax and semantics and specifies rules of transformation that make this system sound and complete. The system is then extended to the equivalent of a first-order monadic language. The soundness of these diagrammatic systems refutes the contention that graphical representation is misleading in reasoning. The validity of the transformation rules ensures that the correct application of the rules will not lead to fallacies. The book concludes with a discussion of some fundamental differences between graphical systems and linguistic systems.

This groundbreaking work will have important influence on research in logic, philosophy, and knowledge representation.

objects. Conjunctive information is more naturally represented by diagrams than by linguistic formulæ. For example, a single Venn diagram can

Still, not all relations can be viewed as membership or inclusion. Shin has been careful throughout her book to restrict herself to monadic systems. Relations per se (polyadic predicates) are not considered. And while it may be true that the formation of a system (such as Venn-II) that is provably both sound and complete would help mitigate the prejudice

perception. In her discussion of perception she shows that disjunctive information is not representable in *any* system. In doing so she relies on

The logical status of diagrams, Sun-Joo Shin, Cambridge university press 1994. https://doi.org/10.1017/CBO9780511574696
Sun-Joo Shin at Yale: https://philosophy.yale.edu/people/sun-joo-shin

QueryViz

- Motivation: Can we create an automatic system that:
  - unambiguously visualizes the logical intent of a SQL query (thus no two different queries lead to an "identical" visualization; with "identical" to be formalized correctly)
  - for some important subset of nested queries
  - with visual diagrams that allow us to reason about SQL design patterns
- Related:
  - Lot's of interest on conjunctive queries equivalence. Now: For what fragment of nested queries is equivalence decidable (under set semantics)?
- Suggestion:
  - nested queries, with inequalities, without any disjunctions
  - Strict superset of conjunctive queries

What is the intend of this query?


```
SELECT   L1.drinker
FROM     Likes L1
WHERE    NOT EXISTS
         (SELECT  *
         FROM     Likes L2
         WHERE    L1.drinker <> L2.drinker
         AND      NOT EXISTS
                  (SELECT  *
                  FROM     Likes L3
                  WHERE    L3.drinker = L2.drinker
                  AND      NOT EXISTS
                           (SELECT  *
                           FROM     Likes L4
                           WHERE    L4.drinker = L1.drinker
                           AND      L4.beer = L3.beer))
         AND      NOT EXISTS
                  (SELECT  *
                  FROM     Likes L5
                  WHERE    L5. drinker = L1. drinker
                  AND      NOT EXISTS
                           (SELECT  *
                           FROM     Likes L6
                           WHERE    L6.drinker = L2.drinker
                           AND      L6.beer= L5.beer)))
```

Likes
drinker
beer

What is the intend of this query?

Likes
drinker
beer

Nesting Depth

```
0   SELECT  L1.drinker
    FROM    Likes L1
    WHERE   NOT EXISTS
1           (SELECT  *
            FROM     Likes L2
            WHERE    L1.drinker <> L2.drinker
            AND      NOT EXISTS
2                    (SELECT  *
                     FROM     Likes L3
                     WHERE    L3.drinker = L2.drinker
                     AND      NOT EXISTS
3                             (SELECT  *
                              FROM     Likes L4
                              WHERE    L4.drinker = L1.drinker
                              AND      L4.beer = L3.beer))
            AND      NOT EXISTS
2                    (SELECT  *
                     FROM     Likes L5
                     WHERE    L5. drinker = L1. drinker
                     AND      NOT EXISTS
3                             (SELECT  *
                              FROM     Likes L6
                              WHERE    L6.drinker = L2.drinker
                              AND      L6.beer= L5.beer)))
```

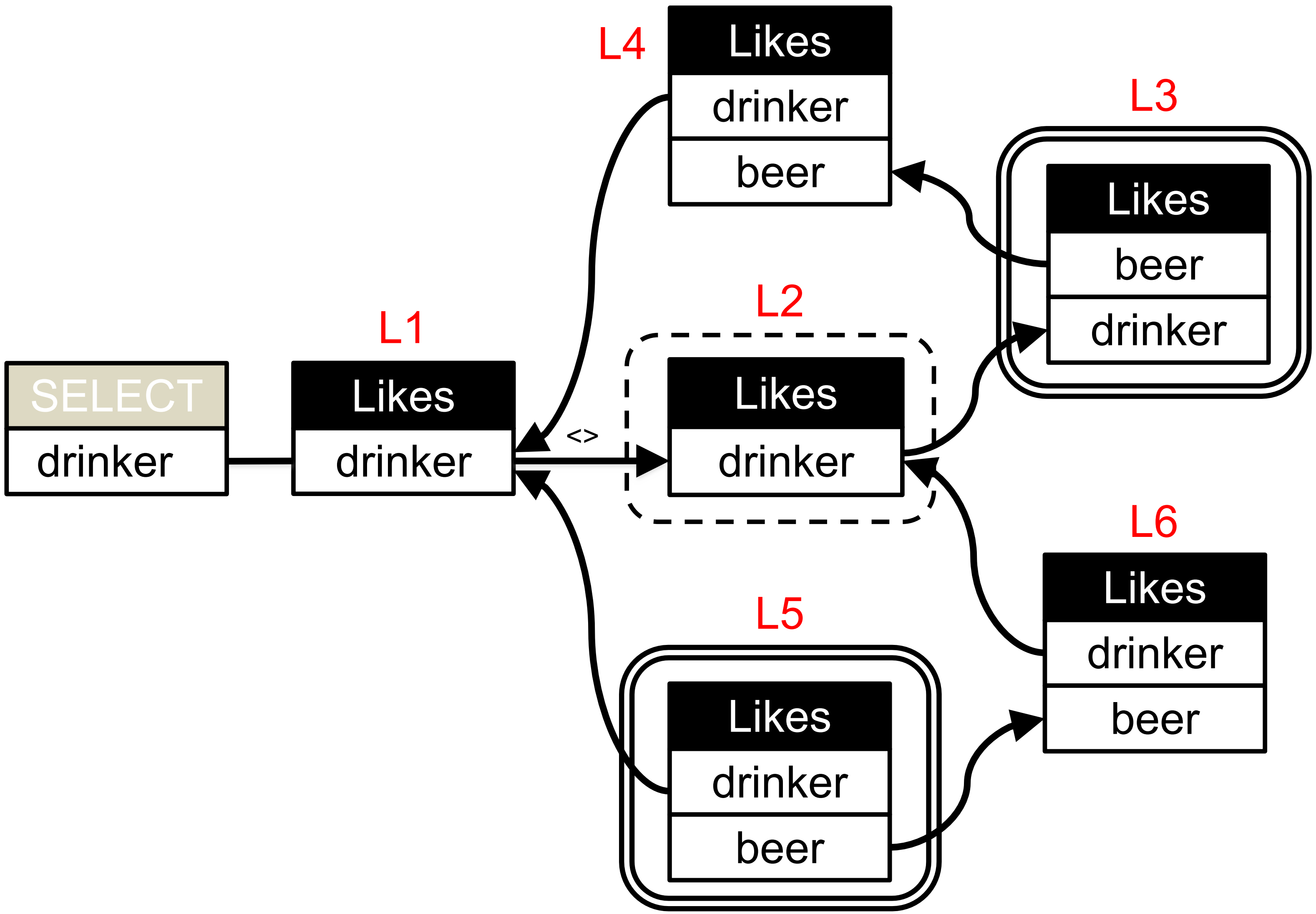
Unique set query: "Find drinkers that like a unique set of beers."

Likes
drinker
beer

Nesting Depth

```
0   SELECT    L1.drinker
    FROM      Likes L1
    WHERE     NOT EXISTS
1             (SELECT   *
              FROM      Likes L2
              WHERE     L1.drinker <> L2.drinker
              AND       NOT EXISTS
2                       (SELECT   *
                        FROM      Likes L3
                        WHERE     L3.drinker = L2.drinker
                        AND       NOT EXISTS
3                                 (SELECT   *
                                  FROM      Likes L4
                                  WHERE     L4.drinker = L1.drinker
                                  AND       L4.beer = L3.beer))
              AND       NOT EXISTS
2                       (SELECT   *
                        FROM      Likes L5
                        WHERE     L5. drinker = L1. drinker
                        AND       NOT EXISTS
3                                 (SELECT   *
                                  FROM      Likes L6
                                  WHERE     L6.drinker = L2.drinker
                                  AND       L6.beer= L5.beer)))
```

Unique set query: "Find drinkers that like a unique set of beers."



“Return any drinker, s.t. there does not exist any other drinker, s.t. there does not exist any beer liked by that other drinker that is not also liked by the returned drinker and there does not exist any beer liked by the returned drinker that is not also liked by the same other drinker.”

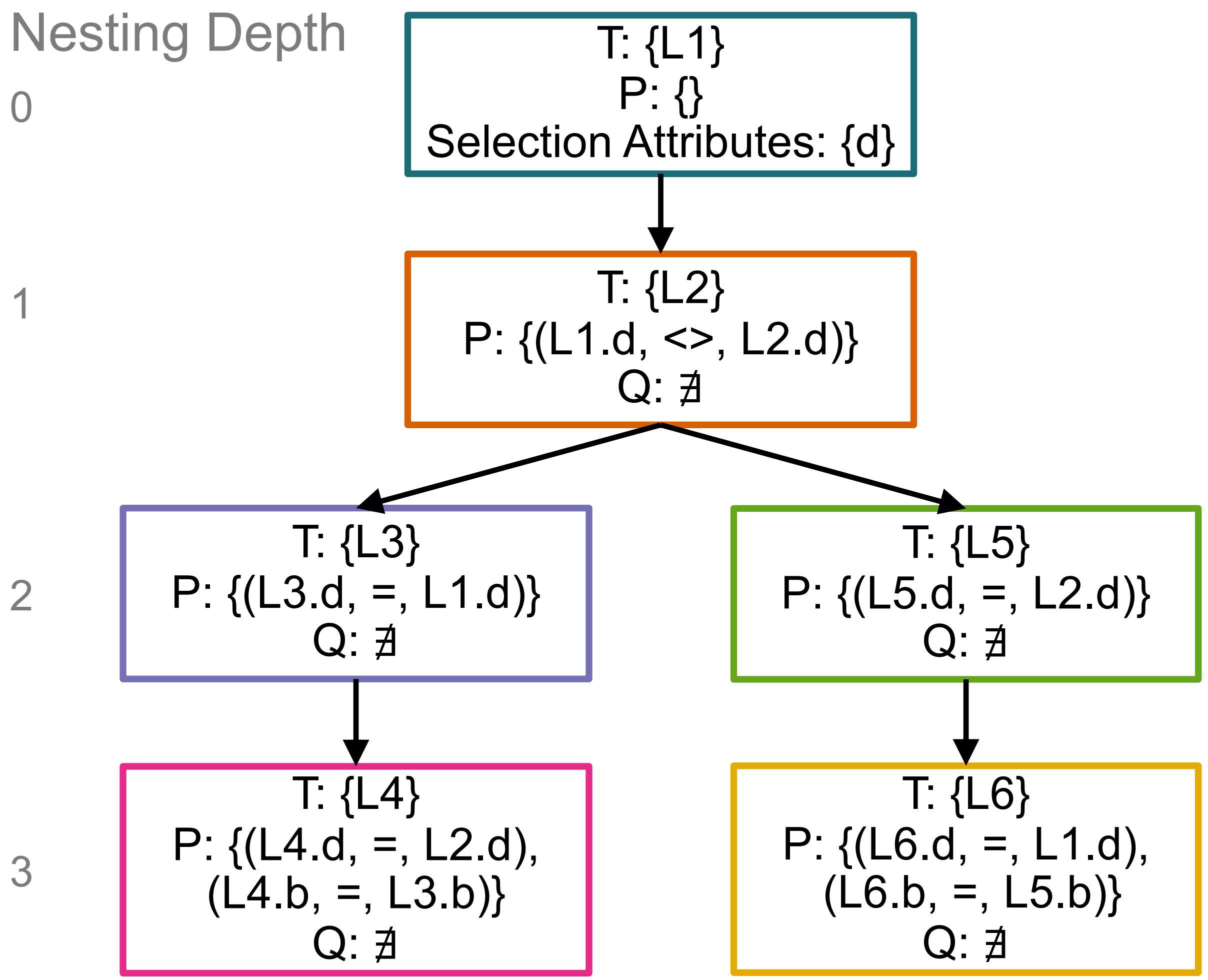
Let x be a drinker, and S(x) be the set of liked beers by drinker x.
Find any drinker x, s.t. there does not exist another drinker x ' , x for which:
$S(x') \subseteq S(x)$ *and* $S(x') \supseteq S(x)$

Unique set query: "Find drinkers that like a unique set of beers."

Likes
drinker
beer

Likes
d
b

$$
\begin{aligned}
\{\, L1.d \mid\ & \exists L1 \in Likes \wedge \\
& \quad \nexists L2 \in Likes\ [L2.d <> L1.d \wedge \\
& \quad\quad \nexists L3 \in Likes\ [L3.d = L1.\ d \wedge \\
& \quad\quad\quad \nexists L4 \in Likes\ [L4.d = L2.d \wedge L4.b = L3.b]] \wedge \\
& \quad\quad \nexists L5 \in Likes\ [L5.d = L2.d \wedge \\
& \quad\quad\quad \nexists L6 \in Likes\ [L6.d = L1.d \wedge L6.b = L5.b]]]\}
\end{aligned}
$$



Notice how the logic tree portrays the nesting hierarchy shown in the FOL (TRC) representation of the SQL query.

Each node in the LT represents the root of a scope in the FOL representation. The predicates in each node are the predicates in the root of the scope of a given node (thus the predicates which do not use any additionally quantified variables).

Atomic predicate classification

type

scope

	selection p.	**join p.**
local *(all C are local)*	C O V	C O C
connecting *(one C is local, another one is foreign*		C O C
foreign *(all C are foreign)*		

Our simple rule: **every predicate needs to have at least one local table identifier.**

Allowed:
- local op value (local selection pred.)
- local op local (local join pred.)
- local op ancestor (connecting join pred.)

Not allowed:
- ancestor op value (foreign selectio pred.)
- ancestor op ancestor (foreign join pred.)

Focus: one single nesting level

- We first restrict ourselves to
  - equi-joins (no inequalities like T.A < T.B)
  - paths (no siblings = every node can have only one nested child)
  - one single nesting level
  - Boolean queries
  - no foreign predicates
  - only binary relations (thus can be represented as graphs)
  - only one single relation R
  - (and as before only conjunctions)
- Given two such queries, what is a generalization of the homomorphism procedure that works for that fragment?

207

Simplifying notation

Schema: R(A,B)

What will become handy, is a short convenient notation for queries

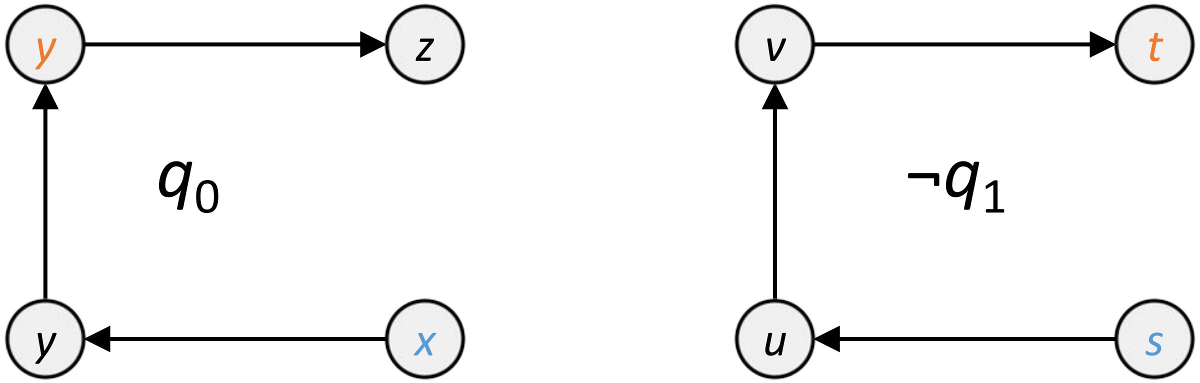
```
SELECT   TRUE
FROM     R R1, R R2, R R3
WHERE    R1.B = R2.A
AND      R2.B = R3.A
NOT EXISTS
         (SELECT  *
         FROM     R R4, R R5, R R6
         WHERE    R4.B = R5.A
         AND      R5.B = R6.A
         AND      R4.A = R1.A
         AND      R6.A = R2.B)
```

$$q_0 \text{ :- } R(x,y),\ R(y,z),\ R(z,w)$$

$$q_1(\mathrm{s,t})\text{:- } R(s,u),\ R(u,v),\ R(v,t),\ \mathrm{s=x},\ \mathrm{t=y}$$

$$q_0 \text{ :- } R(x,y),\ R(y,z),\ R(z,w),\ \neg q_1(\mathrm{x,z})$$

∃ R1, R2, R3 ∈ R
(R1.B=R2.A ∧ R2.B=R3.A ∧
∄ R4, R5, R6 ∈ R
(R4.B=R5.A ∧ R5.B=R6.A ∧
R4.A=R1.A ∧ R6.A = R2.B)
)



s=x, t=y

Simplifying notation

Schema: R(A,B)

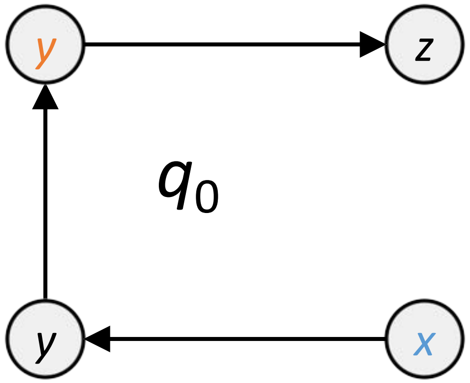
What will become handy, is a short convenient notation for queries

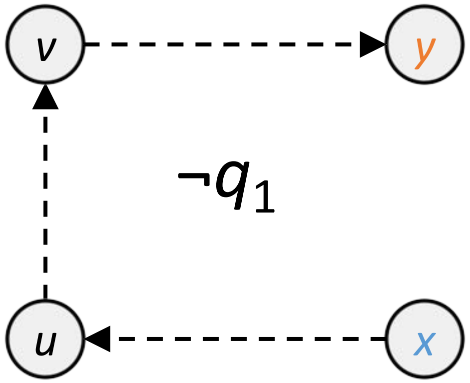
```
SELECT   TRUE
FROM     R R1, R R2, R R3
WHERE    R1.B = R2.A
AND      R2.B = R3.A
NOT EXISTS
         (SELECT  *
         FROM     R R4, R R5, R R6
         WHERE    R4.B = R5.A
         AND      R5.B = R6.A
         AND      R4.A = R1.A
         AND      R6.A = R2.B)
```

q_0 :- $R(x,y), R(y,z), R(z,w)$

$\neg q_1$:- $R(x,u), R(u,v), R(v,y)$

∃ R1, R2, R3 ∈ R
(R1.B=R2.A ∧ R2.B=R3.A ∧
∄ R4, R5, R6 ∈ R
(R4.B=R5.A ∧ R5.B=R6.A ∧
R4.A=R1.A ∧ R6.A = R2.B)
)





Simplifying notation

Schema: R(A,B)

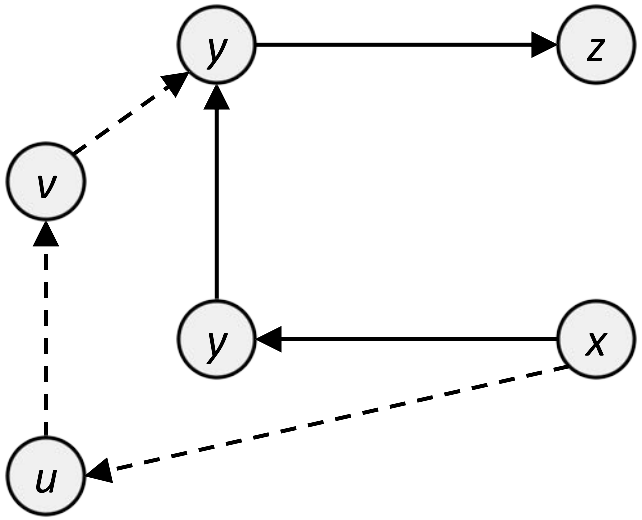
What will become handy, is a short convenient notation for queries

```
SELECT  TRUE
FROM    R R1, R R2, R R3
WHERE   R1.B = R2.A
AND     R2.B = R3.A
NOT EXISTS
        (SELECT  *
        FROM     R R4, R R5, R R6
        WHERE    R4.B = R5.A
        AND      R5.B = R6.A
        AND      R4.A = R1.A
        AND      R6.A = R2.B)
```

$$q_0 \text{ :- } R(x,y), R(y,z), R(z,w)$$

$$\neg q_1 \text{ :- } R(x,u), R(u,v), R(v,y)$$

∃ R1, R2, R3 ∈ R
(R1.B=R2.A ∧ R2.B=R3.A ∧
∄ R4, R5, R6 ∈ R
(R4.B=R5.A ∧ R5.B=R6.A ∧
R4.A=R1.A ∧ R6.A = R2.B)
)



Cartesian product: R'(x,y,z,w)=
R(x,y), R(y,z), R(z,w)?
can be expressed in guarded
fragment of FOL (with negation)?
But single join already not guarded

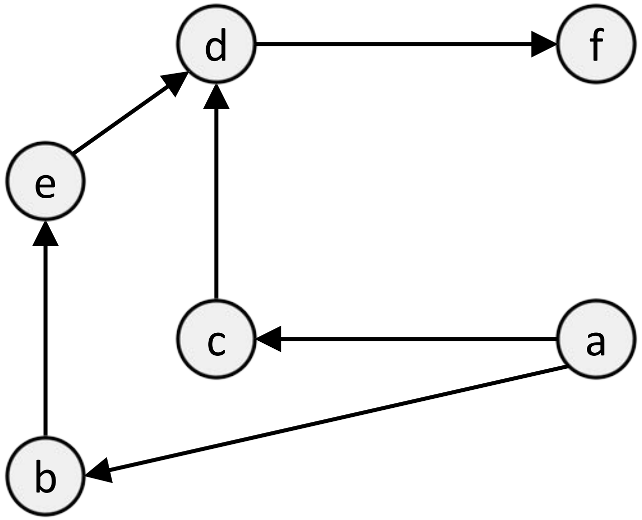
See Barany, Cate, Segoufin,
"Guarded negatation", JACM 2015

guardeness

Exercise

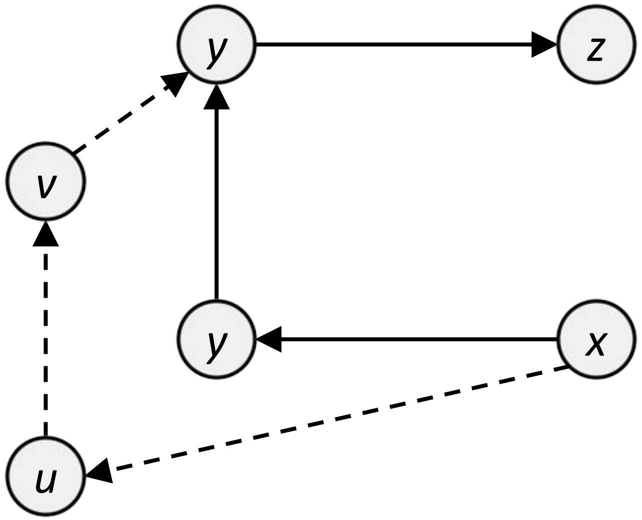
Schema: R(A,B)

Database *D*



Does the query below evaluate to true on above database?

Query *q*

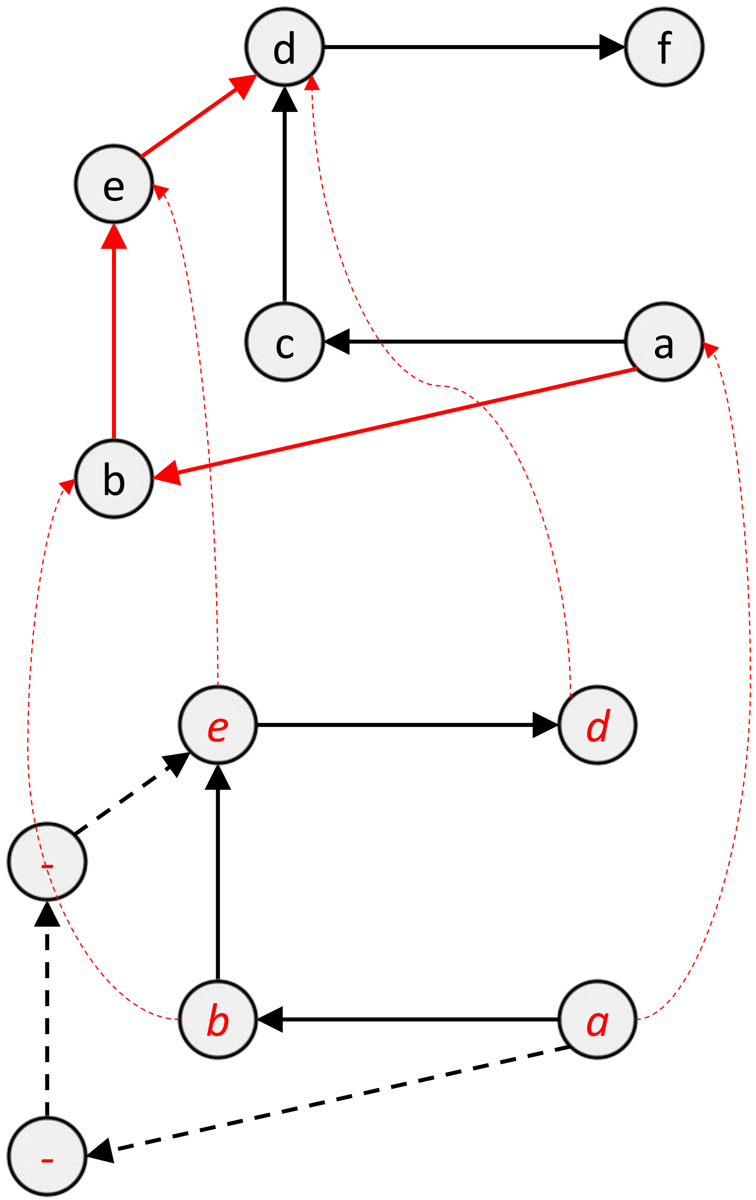


Exercise

Schema: R(A,B)

Database *D*

Query *q*



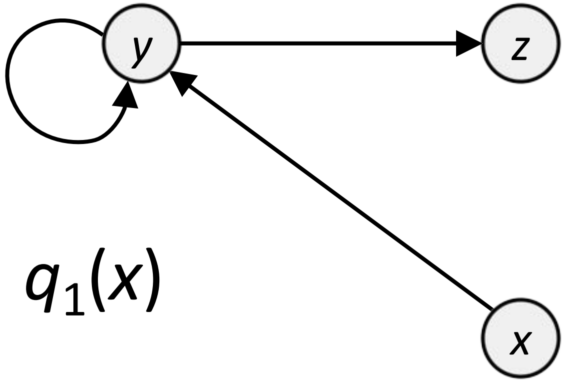
Question

- Find two such nested queries (somehow leveraging the example below) that are equivalent (based on some simple reasoning)
- What is then the *structured* procedure to prove equivalence?

<u>Example</u>

$q_1(x)$:- $R(x,y), R(y,y), R(y,z)$

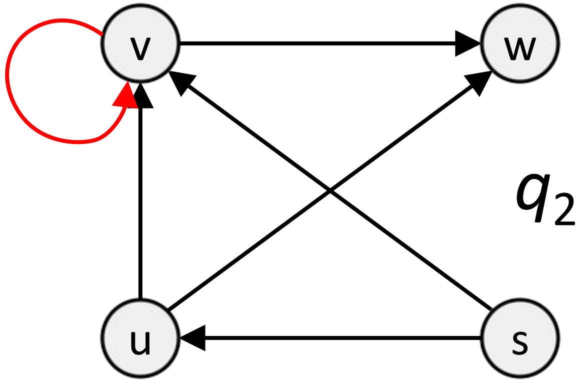
$q_2(s)$:- $R(s,u), R(u,w), R(s,v), R(u,w), R(u,v)$, $R(v,v)$



$q_1(x)$

$\cancel{h_{1\to 2}: \{(x,s),(y,v),(z,w)\}}$ $\quad q_1 \nsubseteq q_2$

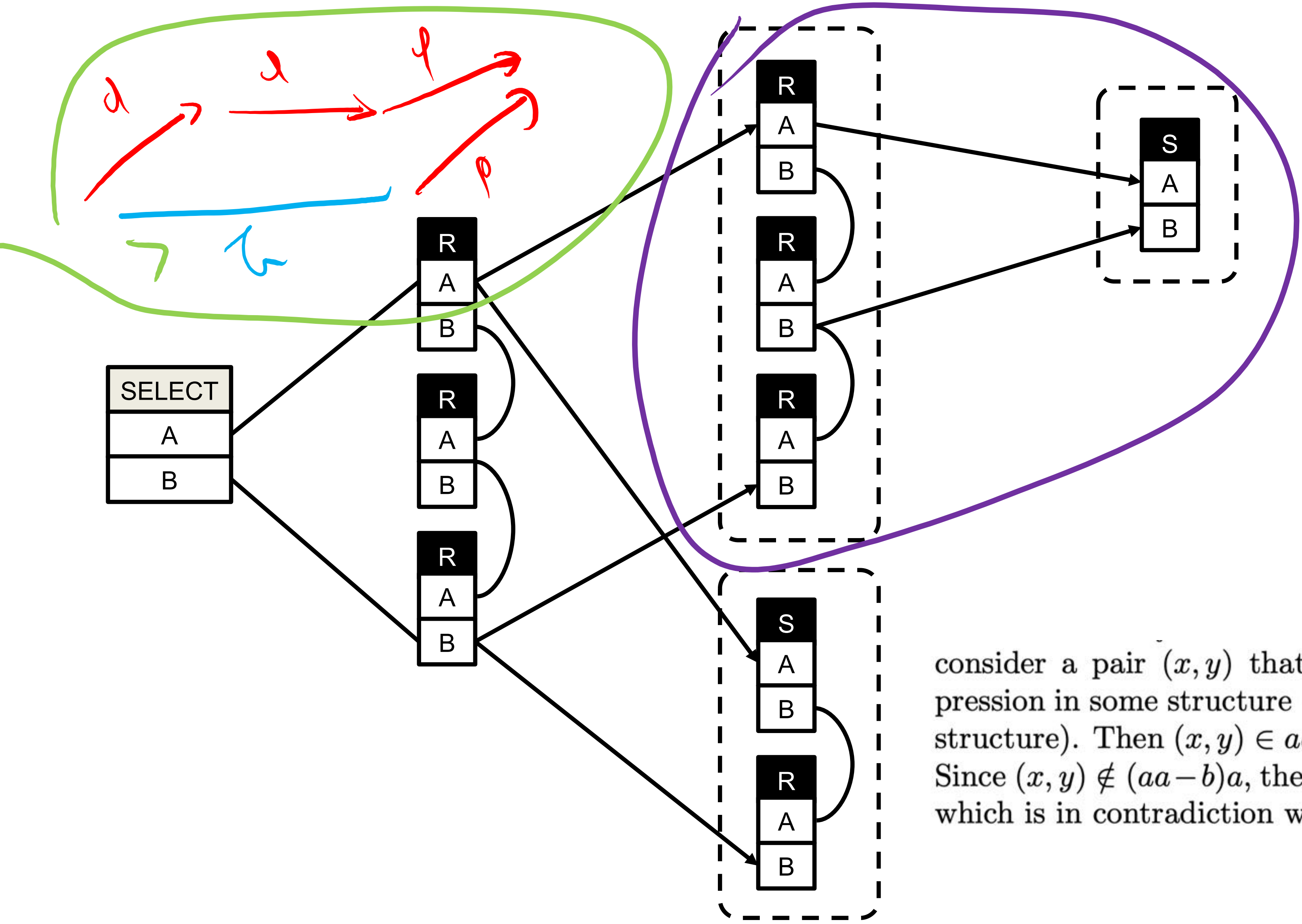
$h_{2\to 1}: \{(s,x),(u,y),(v,y),(w,z)\}$ $\quad q_1 \subseteq q_2$



$q_2(x)$

Undecidability ☹

- Unfortunately, the following problem is already undecidable
  - Consider the class of nested queries with maximal nesting level 2, no disjunctions, our safety restrictions from earlier, set semantics, arbitrary number of siblings
  - Deciding whether any given query is finitely satisfiable is undecidable.
- This follows non-trivially from from following Arxiv paper:
  - "**Undecidability of satisfiability in the algebra of finite binary relations with union, composition, and difference**" by Tony Tan, Jan Van den Bussche, Xiaowang Zhang, Corr 1406.0349. https://arxiv.org/abs/1406.0349



a → R(A,B)

b → S(C,D)

$$= aaa - (aa - b)a - ba$$

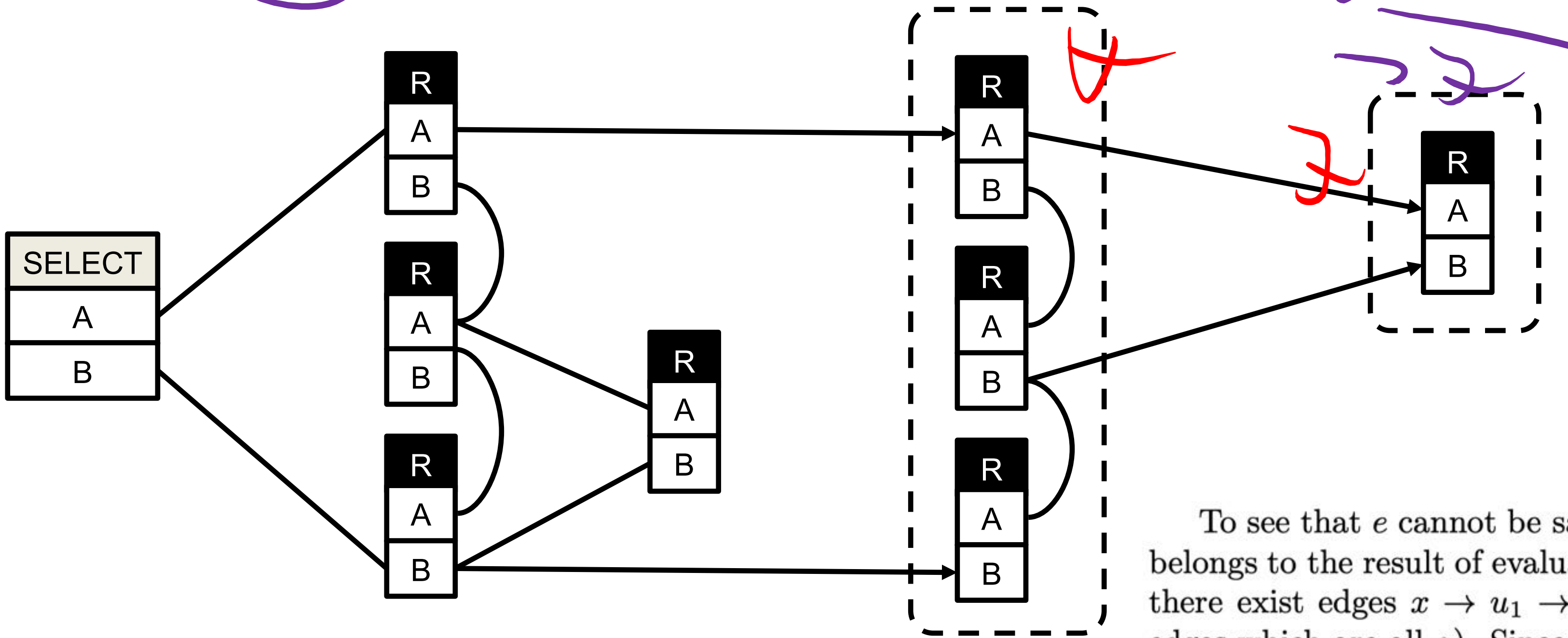
$$= aef - (ae - b)f - bf$$

$$= aef - aef \cup bf - bf$$

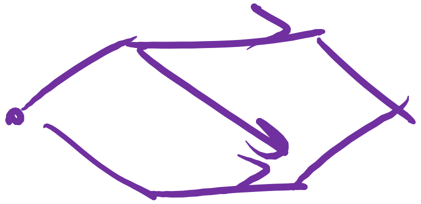
consider a pair (x, y) that would belong to the result of evaluating this expression in some structure (for brevity we are omitting explicit reference to this structure). Then $(x, y) \in aaa$ so there exist a-edges (x, x_1), (x_1, x_2), and (x_2, y). Since $(x, y) \notin (aa - b)a$, the b-edge (x, x_2) must be present. But then $(x, y) \in ba$, which is in contradiction with the last part of the expression. □

$$aaa - ((aa - b)a \cup ba) = aaa - (aa - b)a - ba \qquad X - (Y \cup Z) = X - Y - Z$$

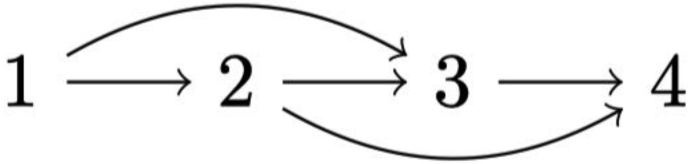
R B

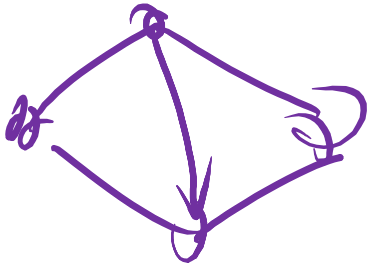


To see that e cannot be satisfied by any series-parallel graph, suppose (x, y) belongs to the result of evaluating e on some structure. Since $(x, y) \in a(a \cap aa)$, there exist edges $x \to u_1 \to u_2 \to y$ and $u_1 \to y$ (we omit the labels on the edges which are all a). Since $(x, y) \notin (aa - a)a$, there must be an edge $x \to u_2$. If at least two of the four elements x, u_1, u_2 and y are identical, the graph contains a cycle and is not series-parallel. If all four elements are distinct, we have a subgraph isomorphic to W above, so the structure is not series-parallel



$$a(aa \cap a) - (aa - a)a$$





See "Undecidability of satisfiability in the algebra of finite binary relations with union, composition, and difference" by Tan, Van den Bussche, Zhang. https://arxiv.org/abs/1406.0349

Open question

NESTING \ (SIBLINGS) ~ OUTDEGREE	1	2	3+
0	CQ	—	—
1			?
2			
3+			

Pointers to related work

- Kolaitis. *Logic and Databases*. Logical Structures in Computation Boot Camp, Berkeley 2016. https://simons.berkeley.edu/talks/logic-and-databases
- Abiteboul, Hull, Vianu. *Foundations of Databases*. Addison Wesley, 1995. http://webdam.inria.fr/Alice/, Ch 2.1: Theoretical background, Ch 6.2: Conjunctive queries & homomorphisms & query containment, Ch 6.3: Undecidability of equivalence for calculus.
- Chandra, Merlin. *Optimal implementation of conjunctive queries in relational data bases*. STOC 1977. https://doi.org/10.1145/800105.803397
- Tan, Van den Bussche, Zhang. *Undecidability of satisfiability in the algebra of finite binary relations with union, composition, and difference*. Corr 1406.0349. https://arxiv.org/abs/1406.0349
- Gatterbauer. *Databases will visualize queries too*. PVLDB 2011. http://www.vldb.org/pvldb/vol4/p1498-gatterbauer.pdf