Updated 2/21/2023

Topic 2: Complexity of Query Evaluation
Unit 1: Conjunctive Queries
Lecture 13

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp23)

https://northeastern-datalab.github.io/cs7240/sp23/

2/21/2023

Topic 2: Complexity of Query Evaluation & Reverse Data Management

- **Lecture 12 (Fri 2/17):** T2-U1 Conjunctive Queries
- **Lecture 13 (Tue 2/21):** T2-U2 Beyond Conjunctive Queries
- **Lecture 14 (Fri 2/24):** T2-U3 Provenance
- **Lecture 15 (Tue 2/28):** T2-U3 Provenance
- **Lecture 16 (Fri 3/3):** T2-U4 Reverse Data Management

Pointers to relevant concepts & supplementary material:

- **Unit 1. Conjunctive Queries**: Query evaluation of conjunctive queries (CQs), data vs. query complexity, homomorphisms, constraint satisfaction, query containment, query minimization, absorption: [Kolaitis, Vardi'00], [Vardi'00], [Kolaitis'16], [Koutris'19] L1 & L2
- **Unit 2. Beyond Conjunctive Queries**: unions of conjunctive queries, bag semantics, nested queries, tree pattern queries: [Kolaitis'16], [Tan+'14], [G.'11], [Martens'17]
- **Unit 3. Provenance**: [Buneman+02], [Green+07], [Cheney+09], [Green,Tannen'17], [Kepner+16], [Buneman, Tan'18]
- **Unit 4. Reverse Data Management**: update propagation, resilience: [Buneman+02], [Kimelfeld+12], [Freire+15]

Three Fundamental Algorithmic Problems about Queries

Let L be a database query language.

- The Query Evaluation Problem:

?

- The Query Equivalence Problem:

?

- The Query Containment Problem:

?

Three Fundamental Algorithmic Problems about Queries

Let L be a database query language.

- The Query Evaluation Problem:
  - "Given a query q in L and a database instance D, evaluate $q(D)$"
  - That's the main problem in query processing.

- The Query Equivalence Problem:

?

- The Query Containment Problem:

?

Three Fundamental Algorithmic Problems about Queries

Let L be a database query language.

- The Query Evaluation Problem:
  - "Given a query q in L and a database instance D, evaluate $q(D)$"
  - That's the main problem in query processing.

- The Query Equivalence Problem:
  - "Given two queries q and q' in L, is it the case that $q \equiv q'$?"
    - i.e., is it the case that, for all (infinitely many) database instances D, we have that $q(D) = q'(D)$?
  - This problem underlies query optimization: transform a given query to an equivalent more efficient one.

- The Query Containment Problem:

?

Three Fundamental Algorithmic Problems about Queries

Is some answer A contained in the query answer?

	$q(D)$	$q'(D)$
case 1	A	A
case 2	–	–
case 3	–	A
~~case 4~~	~~A~~	~~–~~

Let L be a database query language.

- The Query Evaluation Problem:
  - "Given a query q in L and a database instance D, evaluate $q(D)$"
  - That's the main problem in query processing.
- The Query Equivalence Problem:
  - "Given two queries q_1 and q_2 in L, is it the case that $q_1 \equiv q_2$?"
    - i.e., is it the case that, for all (infinitely many) database instances D, we have that $q_1(D) = q_2(D)$?
  - This problem underlies query optimization: transform a given query to an equivalent more efficient one.
- The Query Containment Problem:
  - "Given two queries q_1 and q_2 in L, is it the case that $q_1(D) \subseteq q_2(D)$ for every D?"

Boolean variant $q_1 \Rightarrow q_2$:
for all D: if $D \models q_1$, then $D \models q_2$

Outline: T2-1/2: Query Evaluation & Query Equivalence

- T2-1: Conjunctive Queries (CQs)
  - CQ equivalence and containment
  - Graph homomorphisms
  - Homomorphism beyond graphs
  - CQ containment
  - CQ minimization
- T2-2: Equivalence Beyond CQs
  - Union of CQs, and inequalities
  - Union of CQs equivalence under bag semantics
  - Tree pattern queries
  - Nested queries

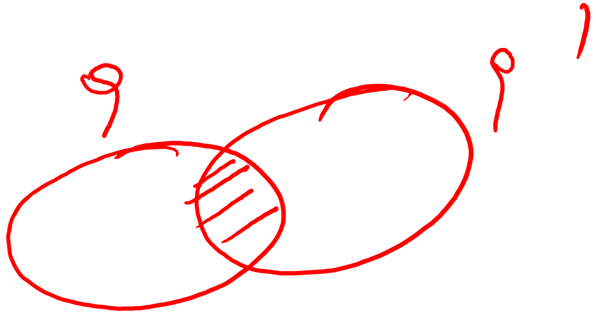
Why bother about Query Containment

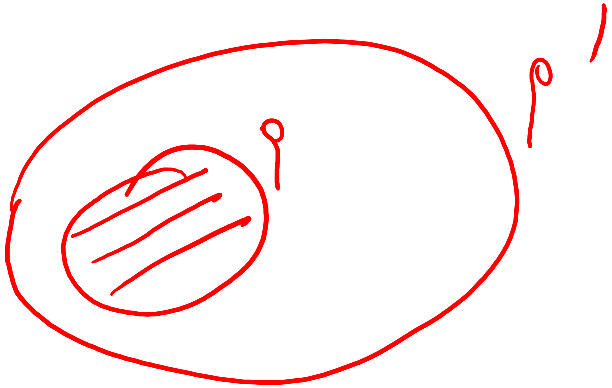
- The Query Containment Problem and Query Equivalence Problem are closely related to each other:
  - $q \equiv q'$ if and only if

    ?

  - $q \subseteq q'$ if and only if

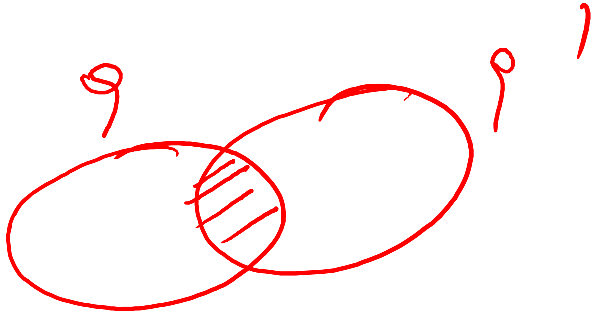
    ?





Why bother about Query Containment

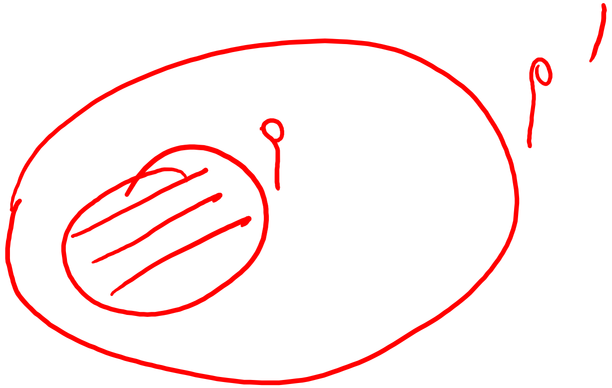
- The Query Containment Problem and Query Equivalence Problem are closely related to each other:



  - $q \equiv q'$ if and only if
    - $q \subseteq q'$ and $q \supseteq q'$

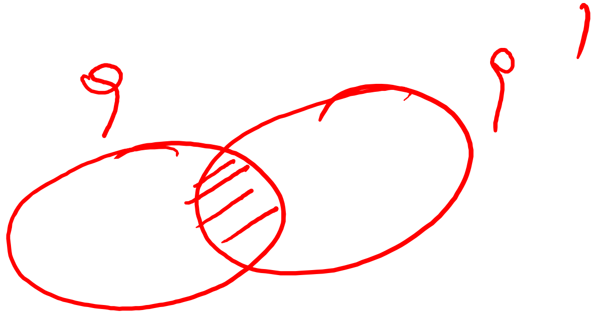
  - $q \subseteq q'$ if and only if

?

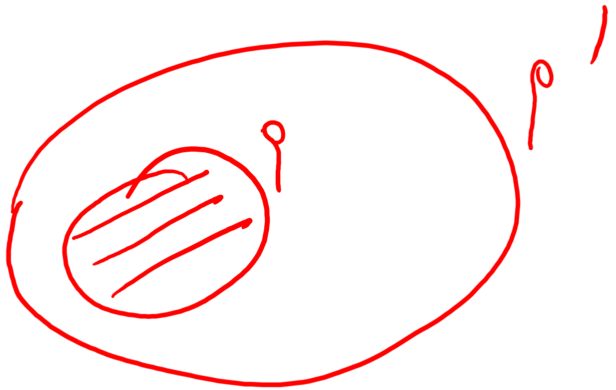


Why bother about Query Containment

- The Query Containment Problem and Query Equivalence Problem are closely related to each other:



  - $q \equiv q'$ if and only if
    - $q \subseteq q'$ and $q \supseteq q'$



  - $q \subseteq q'$ if and only if
    - $q \equiv (q \cap q')$

Complexity of Equivalence and Containment

- Thm: The Query Equivalence Problem for relational calculus (RC) queries is...

?

Complexity of Equivalence and Containment

- Thm: The Query Equivalence Problem for relational calculus (RC) queries is...

  ... undecidable ☹

  A decision problem is undecidable if it is impossible to construct an algorithm that always leads to a correct yes-or-no answer.

- Proof: using Trakhtenbrot's Theorem (1949):
  - The Finite Validity Problem (problem of validity in FOL on the class of all finite models) is undecidable.

    a formula is valid if it comes out as true (or "satisfied") under all admissible assignments of meaning to that formula within the intended semantics for the logical language

what problem do we have to reduce to what other problem ?

Tip: $A \preccurlyeq B$: reduction from A to B.
Means: B could be used to solve A. But A is hard ...

Complexity of Equivalence and Containment

- Thm: The Query Equivalence Problem for relational calculus (RC) queries is...

  ... undecidable ☹

  A decision problem is undecidable if it is impossible to construct an algorithm that always leads to a correct yes-or-no answer.

- Proof: using Trakhtenbrot's Theorem (1949):
  - The Finite Validity Problem (problem of validity in FOL on the class of all finite models) is undecidable.

    a formula is valid if it comes out as true (or "satisfied") under all admissible assignments of meaning to that formula within the intended semantics for the logical language
  - Finite Validity Problem ≼ Query Equivalence Problem

how ?

Tip: A ≼ B: reduction from A to B.
Means: B could be used to solve A. But A is hard ...

- Corollary: The Query Containment Problem for RC is undecidable.

how ?

Complexity of Equivalence and Containment

- Thm: The Query Equivalence Problem for relational calculus (RC) queries is...

  ... undecidable ☹ A decision problem is undecidable if it is impossible to construct an algorithm that always leads to a correct yes-or-no answer.

- Proof: using Trakhtenbrot's Theorem (1949):
  - The Finite Validity Problem (problem of validity in FOL on the class of all finite models) is undecidable. a formula is valid if it comes out as true (or "satisfied") under all admissible assignments of meaning to that formula within the intended semantics for the logical language
  - Finite Validity Problem $\preccurlyeq$ Query Equivalence Problem
    - Take a fixed finitely valid RC sentence ψ, and assume you can solve the query equivalence problem. Then for every RC sentence φ, we could solve validity: φ is finitely valid $\Leftrightarrow \varphi \equiv \psi$.

Tip: $A \preccurlyeq B$: reduction from A to B. Means: B could be used to solve A. But A is hard ...

- Corollary: The Query Containment Problem for RC is undecidable.

how ?

Complexity of Equivalence and Containment

- Thm: The Query Equivalence Problem for relational calculus (RC) queries is...

  ... undecidable ☹

  A decision problem is undecidable if it is impossible to construct an algorithm that always leads to a correct yes-or-no answer.

- Proof: using Trakhtenbrot's Theorem (1949):
  - The Finite Validity Problem (problem of validity in FOL on the class of all finite models) is undecidable.

    a formula is valid if it comes out as true (or "satisfied") under all admissible assignments of meaning to that formula within the intended semantics for the logical language
  - Finite Validity Problem $\preccurlyeq$ Query Equivalence Problem
    - Take a fixed finitely valid RC sentence ψ, and assume you can solve the query equivalence problem. Then for every RC sentence φ, we could solve validity: φ is finitely valid $\Leftrightarrow \varphi \equiv \psi$.

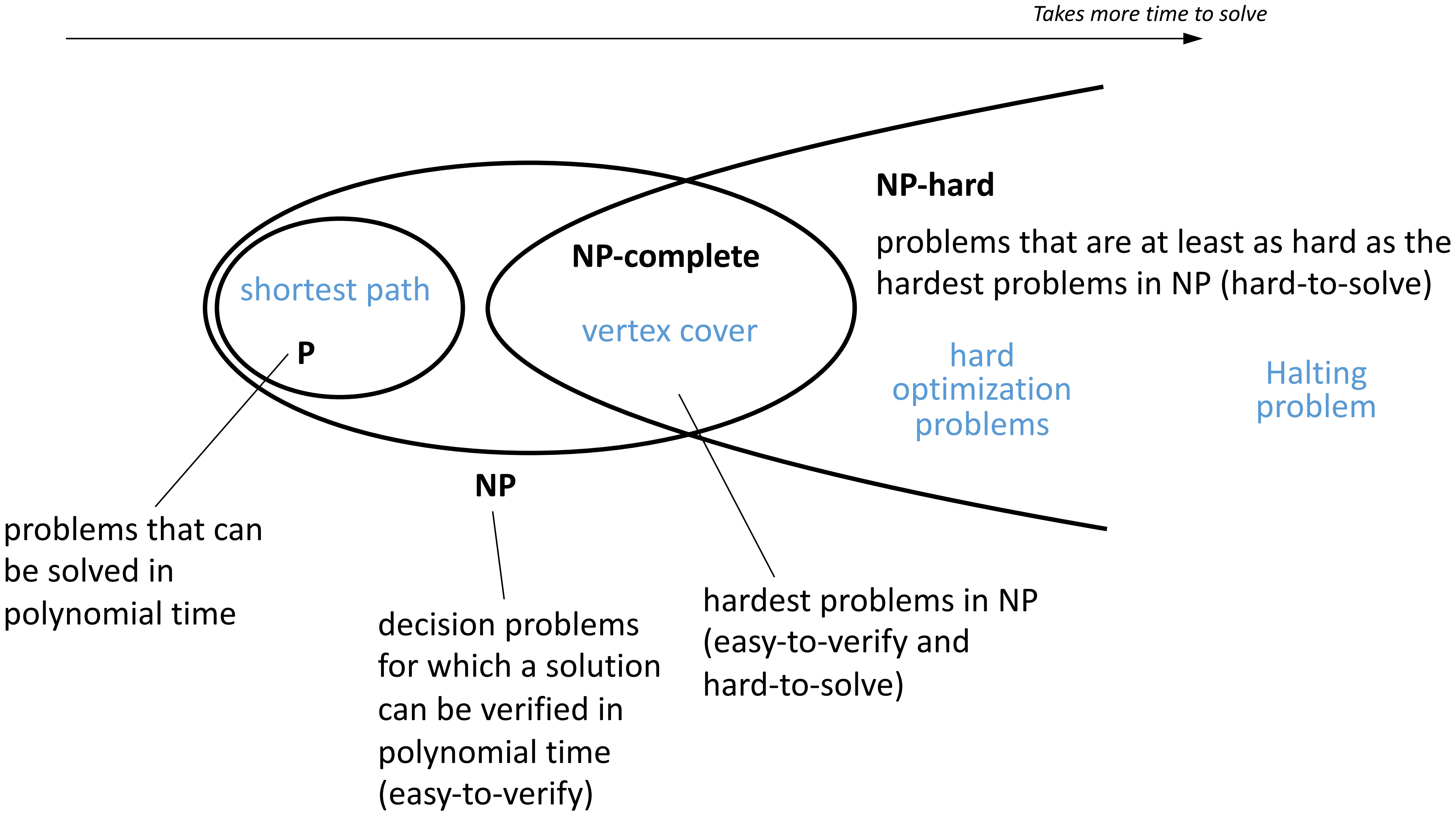
Tip: $A \preccurlyeq B$: reduction from A to B.
Means: B could be used to solve A. But A is hard ...

- Corollary: The Query Containment Problem for RC is undecidable.
  - Proof: Query Equivalence $\preccurlyeq$ Query Containment, since

$$q \equiv q' \Leftrightarrow (q \subseteq q' \text{ and } q' \supseteq q)$$

NP-hardness (assuming P≠NP)

Repeated Slide



Outline: T2-1/2: Query Evaluation & Query Equivalence

- T2-1: Conjunctive Queries (CQs)
  - CQ equivalence and containment
  - Graph homomorphisms
  - Homomorphism beyond graphs
  - CQ containment
  - CQ minimization
- T2-2: Equivalence Beyond CQs
  - Union of CQs, and inequalities
  - Union of CQs equivalence under bag semantics
  - Tree pattern queries
  - Nested queries

Complexity of the Query Evaluation Problem

- The Query Evaluation Problem for Relational Calculus (RC):
  - Given a RC formula φ and a database instance D, find $\varphi^{adom}(D)$.

- Theorem: The Query Evaluation Problem for Relational Calculus is ...

  ... PSPACE-complete.

  - PSPACE: decision problems, can be solved using an amount of memory that is polynomial in the input length (~ in polynomial amount of space).
  - PSPACE-complete: PSPACE + every other PSPACE problem can be transformed to it in polynomial time (PSPACE-hard)

- Proof: We need to show both
  - This problem is in PSPACE.
  - This problem is PSPACE-hard. (We only focus on this task for Boolean RC queries)

Complexity of the Query Evaluation Problem

- Theorem: The Query Evaluation Problem for Boolean RC is PSPACE-hard.
- Reduction uses QBF (Quantified Boolean Formulas):
  - Given QBF $\forall x_1 \exists x_2 \; \ldots . \; \forall x_k \; \psi$, is it true or false
  - (notice every variable is quantified = bound at beginning of sentence; no free variables)
- Proof shows that QBF $\preccurlyeq$ Query Evaluation for Relational Calculus
  - Given QBF $\forall x_1 \exists x_2 \; \ldots . \; \forall x_k \; \psi$,
  - Let V and P be two unary relations and D be the database instance with V(0), V(1), P(1)
  - Obtain ψ^* from ψ by replacing every occurrence of x_i by $P(x_i)$, and $\neg x_i$ by $\neg P(x_i)$
  - Then the following statements are equivalent:
    - $\forall x_1 \exists x_2 \; \ldots . \; \forall x_k \; \psi$ is true
    - $\forall x_1 \, [V(x_1) \rightarrow \exists x_2 \, [V(x_2) \wedge \ldots \forall x_k \, [V(x_k) \rightarrow \psi^*]]\ldots]$ is true on D

Sublanguages of Relational Calculus

- Question: Are there interesting sublanguages of relational calculus for which the Query Containment Problem and the Query Evaluation Problem are "easier" than the full relational calculus?

- Answer:
  - Yes, the language of Conjunctive Queries (CQs) is such a sublanguage.
  - Moreover, conjunctive queries are the most frequently asked queries against relational databases.

Conjunctive Queries (CQs)

- Definition:
  - A CQ is a query expressible by a RC formula in prenex normal form built from atomic formulas $R(y_1,\ldots,y_n)$, and $\wedge$ and $\exists$ only.

    $$\{ (x_1,\ldots,x_k): \ \exists z_1 \ldots \exists z_m \ \phi(x_1, \ldots,x_k, z_1,\ldots,z_k) \},$$

  - where $\phi(x_1, \ldots,x_k, z_1,\ldots,z_k)$ is a conjunction of atomic formulas of the form $R(y_1,\ldots,y_m)$.
  - Prenex formula: prefix (quantifiers & bound variables), then quantifier-free part
- Equivalently, a CQ is a query expressible by a RA expression of the form
  - $\pi_X(\sigma_\Theta(R_1 \times \ldots \times R_n))$, where
  - Θ is a conjunction of equality atomic formulas (equijoin).
- Equivalently, a CQ is a query expressible by an SQL expression of the form
  - SELECT <list of attributes>
FROM <list of relation names>
WHERE <conjunction of equalities>

Conjunctive Queries (CQs)

- Definition:
  - A CQ is a query expressible by a RC formula in prenex normal form built from atomic formulas $R(y_1,\ldots,y_n)$, and $\wedge$ and $\exists$ only.

    $\{ (x_1,\ldots,x_k): \exists z_1 \ldots \exists z_m \ \phi(x_1, \ldots,x_k, z_1,\ldots,z_k) \}$,
  - where $\phi(x_1, \ldots,x_k, z_1,\ldots,z_k)$ is a conjunction of atomic formulas of the form $R(y_1,\ldots,y_m)$.

- Equivalently, a CQ can be written as a logic-programming rule:

  $Q(x_1,\ldots,x_k)$:- $R_1(\mathbf{u}_1), \ldots, R_n(\mathbf{u}_n)$, where
  - Each variable x_i occurs in the right-hand side of the rule.
  - Each $\mathbf{u}_i$ is a tuple of variables (not necessarily distinct)
  - The variables occurring in the right-hand side (the body), but not in the left-hand side (the head) of the rule are existentially quantified (but the quantifiers are not displayed).

Conjunctive Queries (CQs)

- Every natural join is a conjunctive query with ...

  ... no existentially quantified variables
- Example: Given R(A,B,C), S(B,C,D)
  - $R \bowtie S = \{(x,y,z,w): R(x,y,z) \wedge S(y,z,w)\}$
  - q(x,y,z,w) :- R(x,y,z), S(y,z,w)

    (no variables are existentially quantified)
  - SELECT R.A, R.B, R.C, S.D
    FROM R, S
    WHERE R.B = S.B AND R.C = S.C
- Conjunctive queries are also known as SPJ-queries (SELECT-PROJECT-JOIN queries)

Examples of Conjunctive Queries

$E(\mathrm{v}_1, \mathrm{v}_2)$

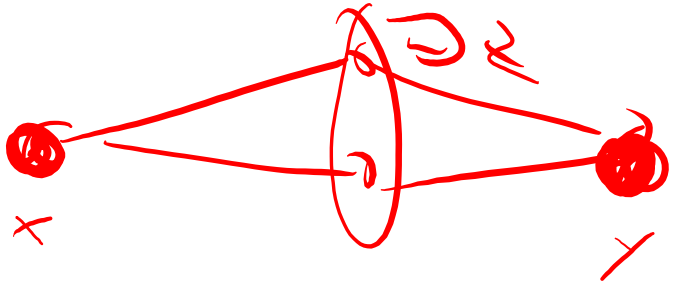
- Return paths of Length 2: (binary output)

  DRC: ?

  TRC: ?

  RA: ?

  Datalog: ?



Examples of Conjunctive Queries

$E(\mathrm{v}_1, \mathrm{v}_2)$

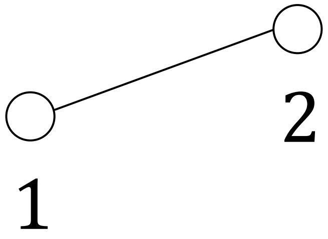
- Return paths of Length 2: (binary output)

DRC: $\{(x, y) \mid \exists z\, [E(x, z) \wedge E(z, y)]\}$

TRC: ?

RA: ?

Datalog: ?



Is there a path of length 2 ?

Examples of Conjunctive Queries

$E(\mathrm{v}_1, \mathrm{v}_2)$

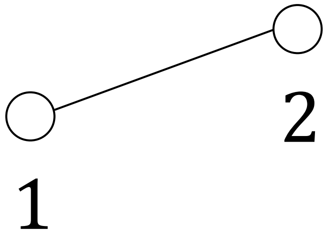
- Return paths of Length 2: (binary output)

DRC: $\{(x, y) \mid \exists z\, [E(x, z) \wedge E(z, y)]\}$

TRC: ?

RA: ?

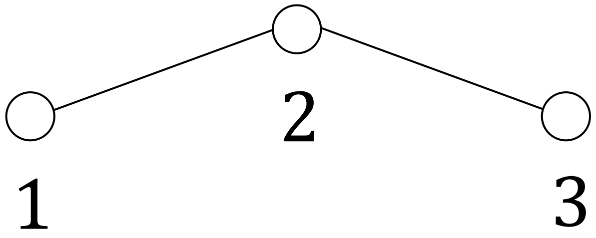
Datalog: ?



E

1	2
2	1

Is there a path of length 2 ?



$x \neq y$ not required ?

HOMOM.

ISOM.

Examples of Conjunctive Queries

$E(\mathrm{v}_1, \mathrm{v}_2)$

- Return paths of Length 2: (binary output)

DRC: $\{(x, y) \mid \exists z\, [E(x, z) \wedge E(z, y)]\}$

TRC: $\{q \mid \exists e_1, e_2 \in E[e_1.B = e_2.A \wedge q.A = e_1.A \wedge q.B = e_2.B]\}$

RA: ?

Datalog: ?

Examples of Conjunctive Queries

$E(\mathrm{v}_1, \mathrm{v}_2)$ 

- Return paths of Length 2: (binary output)

DRC: $\{(x, y) \mid \exists z\, [E(x, z) \wedge E(z, y)]\}$

TRC: $\{q \mid \exists e_1, e_2 \in E[e_1.B = e_2.A \wedge q.A = e_1.A \wedge q.B = e_2.B]\}$

RA: $\pi_{\$1,\$4}(\sigma_{\$2=\$3}(E \times E))$ unnamed perspective

Datalog: ?

Examples of Conjunctive Queries

$E(\mathrm{v}_1, \mathrm{v}_2)$ 

- Return paths of Length 2: (binary output)

DRC: $\{(x, y) \mid \exists z\, [E(x, z) \wedge E(z, y)]\}$

TRC: $\{q \mid \exists e_1, e_2 \in E[e_1.B = e_2.A \wedge q.A = e_1.A \wedge q.B = e_2.B]\}$

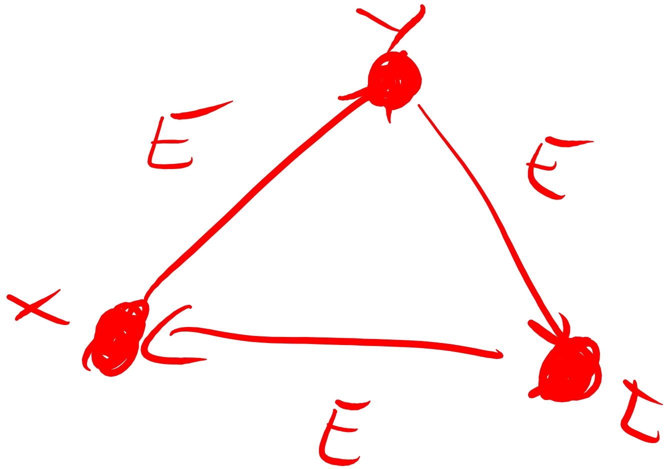
RA: $\pi_{\$1,\$4}(\sigma_{\$2=\$3}(E \times E))$ unnamed perspective

Datalog: $Q(\mathrm{x,y}) \text{ :- } E(x, z), E(z, y)$

- Is there a cycle of Length 3: (Boolean query)

DRC: ?

Datalog: ?



Examples of Conjunctive Queries

$E(\mathrm{v}_1, \mathrm{v}_2)$

- Return paths of Length 2: (binary output)

DRC: $\{(x, y) \mid \exists z\,[E(x, z) \wedge E(z, y)]\}$

TRC: $\{q \mid \exists e_1, e_2 \in E[e_1.B = e_2.A \wedge q.A = e_1.A \wedge q.B = e_2.B]\}$

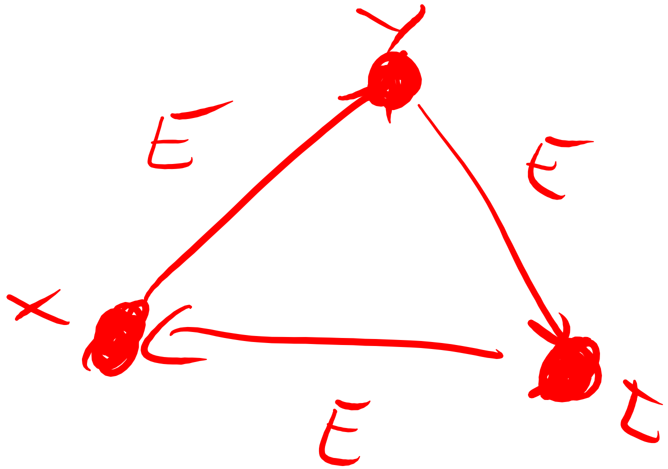
RA: $\pi_{\$1,\$4}(\sigma_{\$2=\$3}(E \times E))$ unnamed perspective

Datalog: $Q(\mathrm{x,y}) :\!-\ E(x, z), E(z, y)$

- Is there a cycle of Length 3: (Boolean query)

DRC: $\exists x\, \exists y\, \exists z\, [E(x, y) \wedge E(y, z) \wedge E(z, x)]\}$

Datalog: ?



Examples of Conjunctive Queries

$E(\mathrm{v}_1, \mathrm{v}_2)$

- Return paths of Length 2: (binary output)

  DRC: $\{(x, y) \mid \exists z\, [E(x, z) \land E(z, y)]\}$

  TRC: $\{q \mid \exists e_1, e_2 \in E[e_1.B = e_2.A \land q.A = e_1.A \land q.B = e_2.B]\}$

  RA: $\pi_{\$1,\$4}(\sigma_{\$2=\$3}(E \times E))$ unnamed perspective

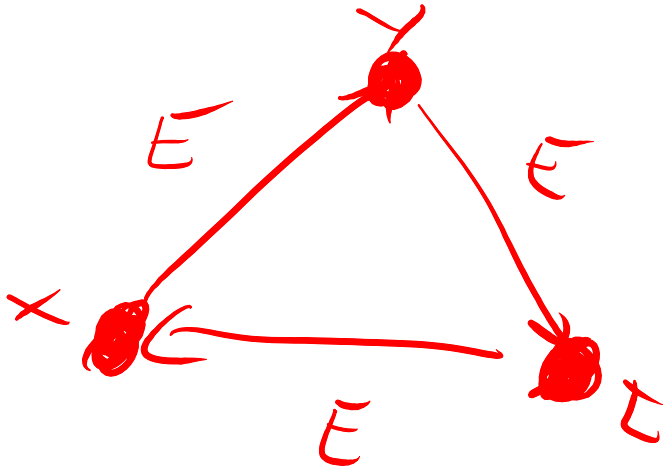
  Datalog: $Q(\mathrm{x,y})$:- $E(x, z), E(z, y)$

- Is there a cycle of Length 3: (Boolean query)

  DRC: $\exists x\, \exists y\, \exists z\, [E(x, y) \land E(y, z) \land E(z, x)]\}$

  Datalog: Q :- $E(x, y), E(y, z), E(z, x)$

alternatiave variant with "directed" cycle and arcs



Vardi's Taxonomy of the Query Evaluation Problem

Definition: Let **L** be a database query language.

- The combined complexity of **L** is the decision problem $\mathbf{P}_{\varphi,\mathbf{D}}$:
  - given an **L**-sentence φ and a database instance **D**, is φ true on **D**?
  - In symbols, does $\mathbf{D} \vDash \varphi$ (does **D** satisfy φ)?

D φ

- The data complexity of **L** is the family of the following decision problems $\mathbf{P}_{\varphi}$, where φ is a <u>fixed</u> L-sentence:
  - given a database instance **D**, does $\mathbf{D} \vDash \varphi$?
- The query complexity of **L** is the family of the following decision problems $\mathbf{P}_{\mathbf{D}}$, where **D** is a <u>fixed</u> database instance:
  - given an **L**-sentence φ, does $\mathbf{D} \vDash \varphi$?

Moshe Vardi. "The Complexity of Relational Query Languages." STOC 1982. https://doi.org/10.1145/800070.802186

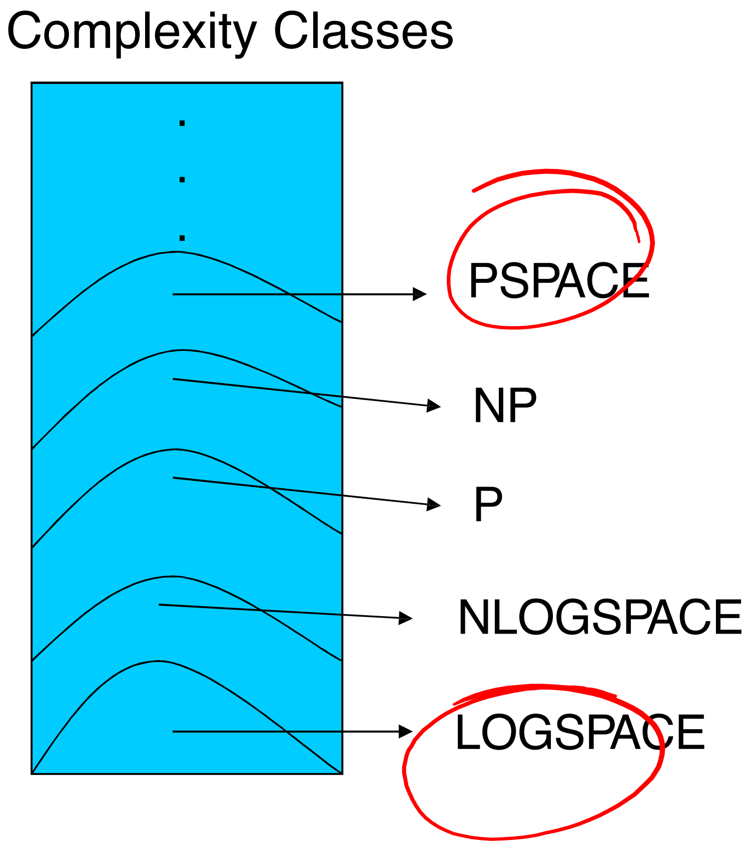
Vardi's Taxonomy of the Query Evaluation Problem

Vardi's "empirical" discovery:

- For most query languages L:
  - The data complexity of L is of lower complexity than both the combined complexity of L and the query complexity of L.
  - The query complexity of L can be as hard as the combined complexity of L.

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases

Taxonomy of the Query Evaluation Problem for Relational Calculus



The Query Evaluation Problem
for Relational Calculus

Problem	**Complexity**
Combined Complexity	PSPACE-complete
Query Complexity	• in PSPACE • can be PSPACE-complete
Data Complexity	In LOGSPACE

Source: Phokion Kolaitis

Summary

- Relational Algebra and Relational Calculus have "essentially" the same expressive power.

- The Query Equivalence Problem for Relational Calculus is undecidable.
    - Therefore also the Query Containment Problem

- The Query Evaluation Problem for Relational Calculus:
    - Data Complexity is in LOGSPACE (and thus very efficient)
    - Query Complexity is PSPACE-complete
    - Combined Complexity is PSPACE-complete