Updated 2/23/2022

Topic 2: Complexity of Query Evaluation
Unit 1: Conjunctive Queries
Lecture 11

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp22)

https://northeastern-datalab.github.io/cs7240/sp22/

2/22/2022

Updated 2/23/2022

Topic 2: Complexity of Query Evaluation
Unit 1: Conjunctive Queries
Lecture 11

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp22)

https://northeastern-datalab.github.io/cs7240/sp22/

2/22/2022

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:

?

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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:

?

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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 every database instance 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:

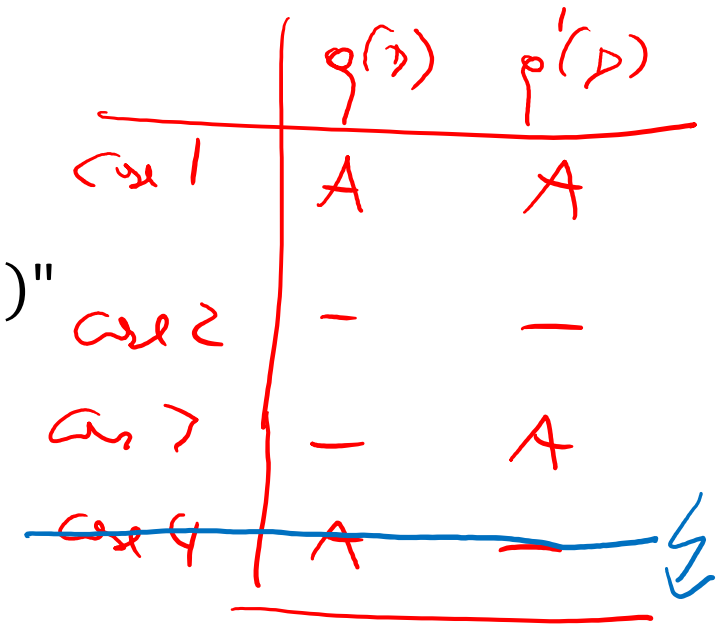
?

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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 every database instance 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:
  - "Given two queries q and q' in L, is it the case that $q \subseteq q'$?"

$q(D) \subseteq q'(D)$

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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

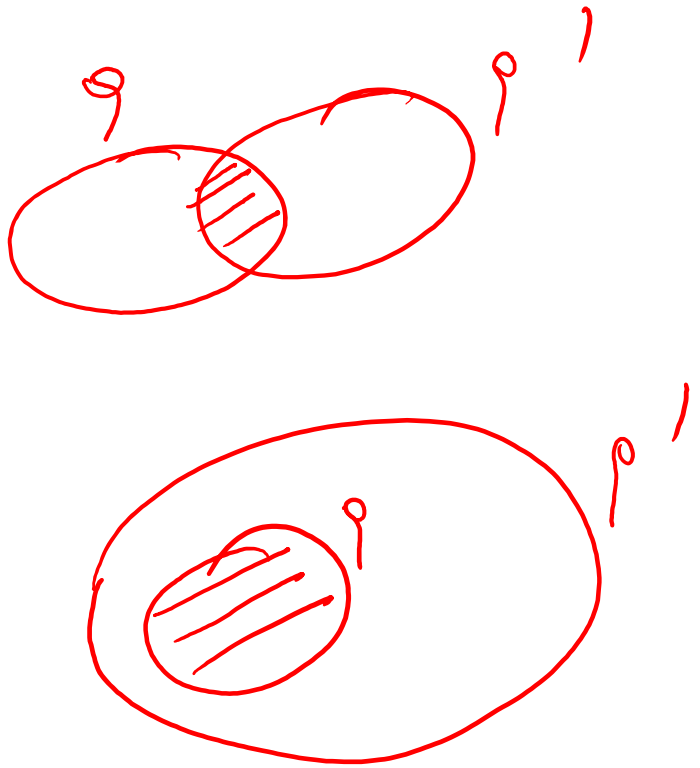
- T2-1: Conjunctive Queries (CQs)
  - CQ equivalence and containment
  - Graph homomorphisms
  - Homomorphism beyond graphs
  - CQ containment
  - CQ minimization
- T2-2: Equivalence Beyond CQs
  - Union of CQs, and inequalities
  - Union of CQs equivalence under bag semantics
  - Nested queries
  -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

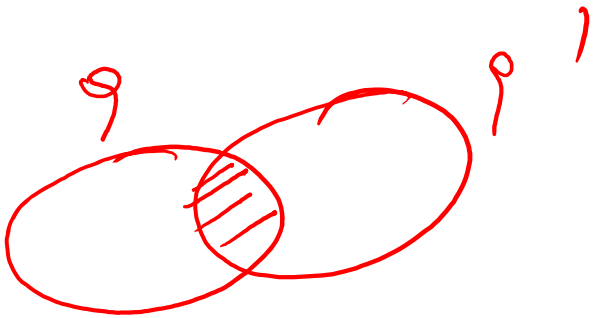
    ?

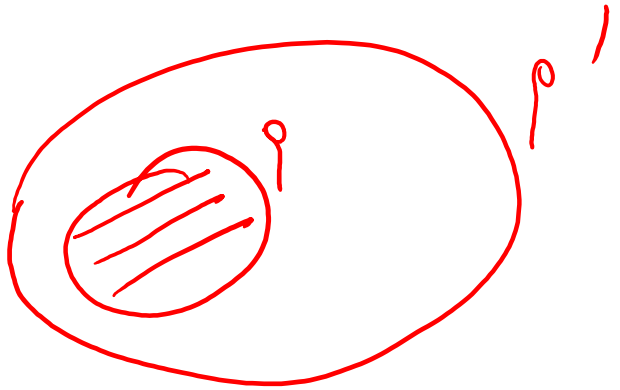


Why bother about Query Containment

- 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

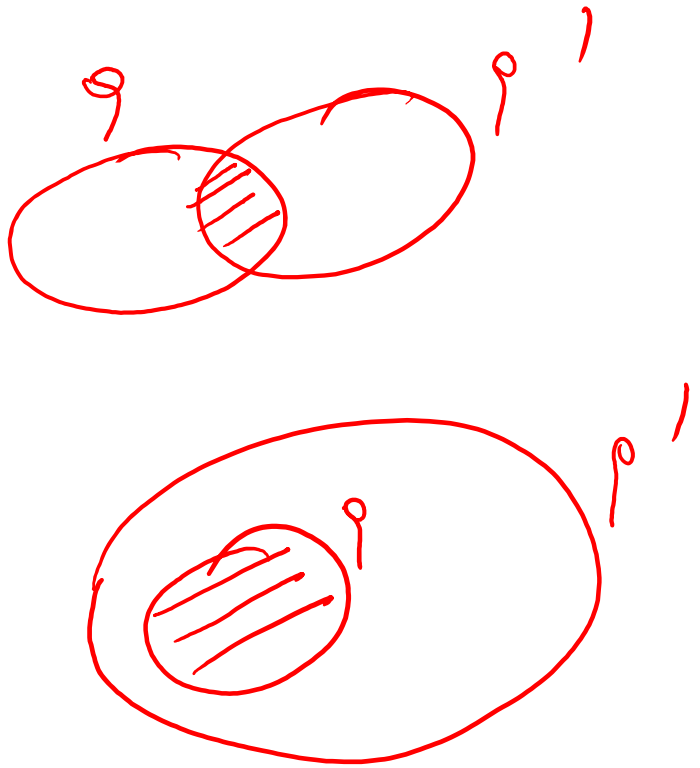
?





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...

?

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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 $\preccurlyeq$ Query Equivalence Problem

    how ?

    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 ?

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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

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

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Updated 2/25/2022

Topic 2: Complexity of Query Evaluation
Unit 1: Conjunctive Queries (continued)
Lecture 12

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp22)

https://northeastern-datalab.github.io/cs7240/sp22/

2/25/2022

Complexity of the Query Evaluation Problem

- The Query Evaluation Problem for Relational Calculus:
  - 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)

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Complexity of the Query Evaluation Problem

- Theorem: The Query Evaluation Problem for Relational Calculus 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, there are no free variables)
- Proof
  - Show that QBF $\preccurlyeq_p$ Query Evaluation for Relational Calculus

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Complexity of the Query Evaluation Problem

Proof: Show that QBF $\preccurlyeq$p 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 relation symbols
- Obtain ψ^* from ψ by replacing x_i by $P(x_i)$, and $\neg x_i$ by $\neg P(x_i)$
- Let D be the database instance with V = {0,1}, P={1}.
- 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.

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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.

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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>

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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:

  $$Q(x_1,\ldots,x_k) \text{ :- } R_1(\mathbf{u}_1), \ldots, R_n(\mathbf{u}_n), \text{ where}$$

  - 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).

Based on Phokion Kolaitis' "Logic and Databases" series at Simons Institute, 2016. https://simons.berkeley.edu/talks/logic-and-databases
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Examples of Conjunctive Queries

$E(\text{from}, \text{to})$

- Return paths of Length 2: (binary output)

  RC: ?

  RA: ?

  Datalog: ?

Examples of Conjunctive Queries

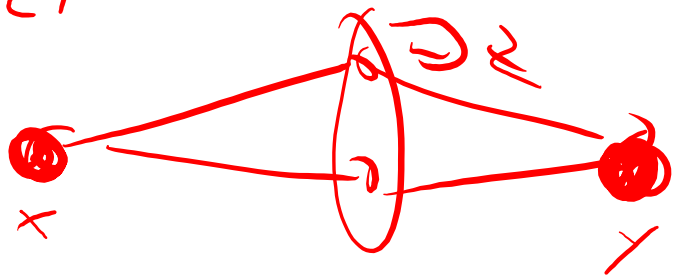
$E(\text{from}, \text{to})$

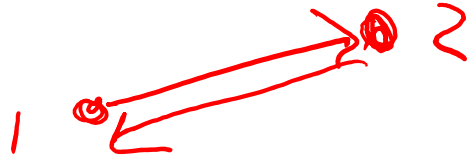
- Return paths of Length 2: (binary output)

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

RA: ?

Datalog: ?





(1, 2)

(1, 1)

Examples of Conjunctive Queries

$E(\text{from}, \text{to})$

- Return paths of Length 2: (binary output)

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

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

Datalog: ?

Examples of Conjunctive Queries

$E(\text{from}, \text{to})$

- Return paths of Length 2: (binary output)

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

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

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

- Cycle of Length 3: (Boolean query)

  RC: ?

  Datalog: ?

Examples of Conjunctive Queries

$E(\text{from}, \text{to})$

- Return paths of Length 2: (binary output)

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

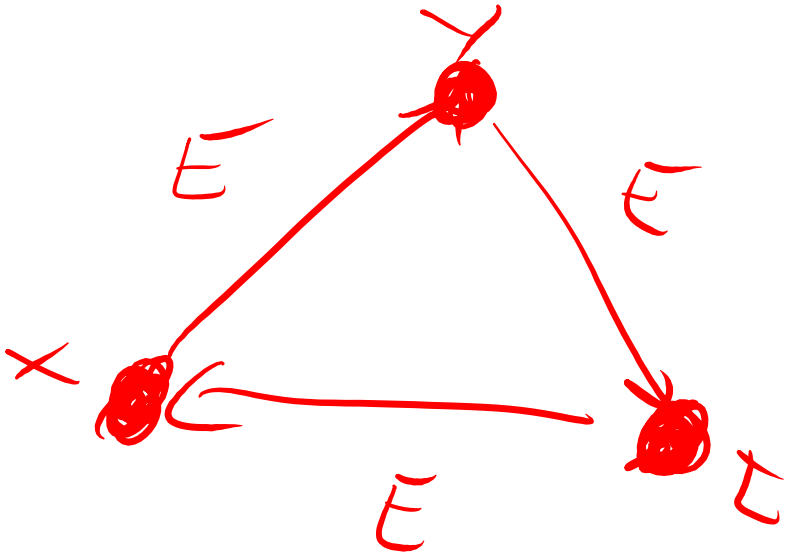
  RA: $\pi_{1,4}(\sigma_{\$2=\$3}(E \times E))$ unnamed perspective

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

- Cycle of Length 3: (Boolean query)

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

  Datalog: ?



Examples of Conjunctive Queries

$E(\text{from}, \text{to})$

- Return paths of Length 2: (binary output)

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

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

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

- Cycle of Length 3: (Boolean query)

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

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

Conjunctive Queries

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

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

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

Conjunctive Query Evaluation and Containment

- Definition: Two fundamental problems about CQs
  - Conjunctive Query Evaluation (CQE):
    - Given a conjunctive query q and an instance D, find q(D).
  - Conjunctive Query Containment (CQC):
    - Given two k-ary conjunctive queries q_1 and q_2, is it true that $q_1 \subseteq q_2$?
      (i.e., for every instance D, we have that q1(D) ⊆ q2(D))
    - Given two Boolean conjunctive queries q1 and q2, is it true that $q_1 \models q_2$? (that is, for all D, if $D \models q_1$, then $D \models q_2$)?

- Notice that CQC is logical implication.
- Later today: connection to homomorphisms

Vardi's Taxonomy of the Query Evaluation Problem

M.Y Vardi, "The Complexity of Relational Query Languages", 1982

- Definition: Let L be a database query language.
  - The combined complexity of L is the decision problem:
    - given an L-sentence and a database instance D, is φ true on D?
    - In symbols, does $D \models \varphi$ (does D satisfy φ)?
  - The data complexity of L is the family of the following decision problems P_φ, where φ is an L-sentence:
    - given a database instance D, does $D \models \varphi$?
  - The query complexity of L is the family of the following decision problems P_D, where D is a database instance:
    - given an L-sentence φ, does $D \models \varphi$?

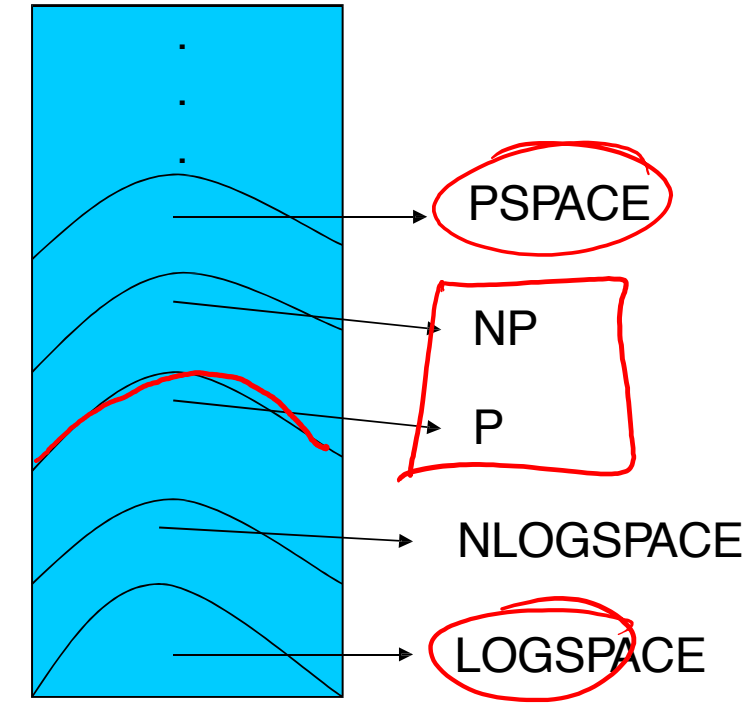
Vardi's Taxonomy of the Query Evaluation Problem

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.

Source: Phokion Kolaitis

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.

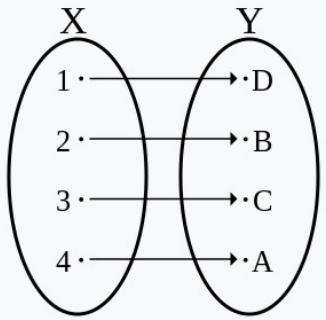
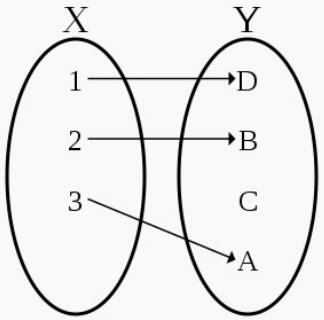
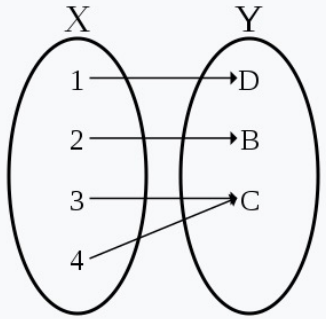
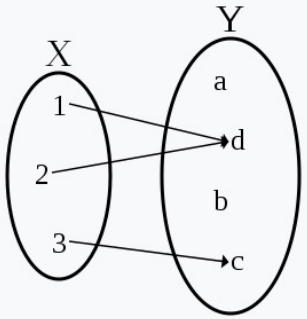
- The Query Equivalence Problem for Relational Calculus is undecidable.
  - Therefore also the Query Containment Problem

- The Query Evaluation Problem for Relational Calculus:
  - Data Complexity is in LOGSPACE
  - Combined Complexity is PSPACE-complete
  - Query Complexity is PSPACE-complete.

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
  - Nested queries
  - Tree pattern queries

Injective, Surjective, and Bijective functions $f: X \rightarrow Y$

	surjective	non-surjective
injective	 **bijective**	 **injective-only**
non-injective	 **surjective-only**	 **general**

Function ?

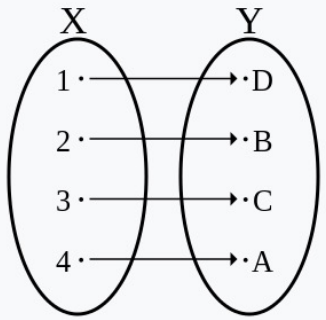
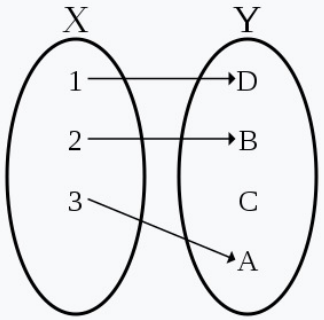
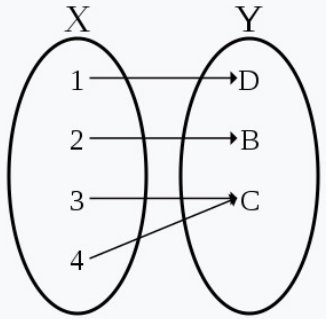
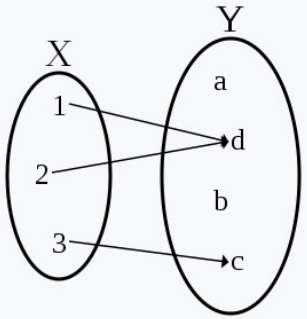
Injective function ?

Surjective function ?

Bijective function ?

Sources: https://en.wikipedia.org/wiki/Bijection,_injection_and_surjection

Injective, Surjective, and Bijective functions $f: X \to Y$

	surjective	non-surjective
injective	 bijective	 injective-only
non-injective	 surjective-only	 general

Function maps each argument (element from its domain) to exactly one image (element in its codomain)

$$\forall x \in X, \exists! \, y \in Y[y = f(x)]\}$$

Injective function ?

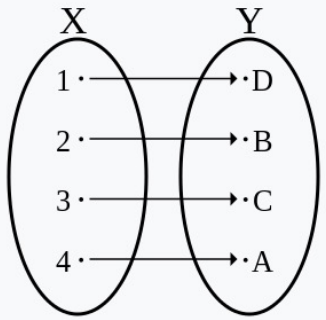
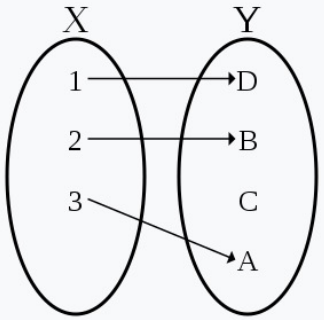
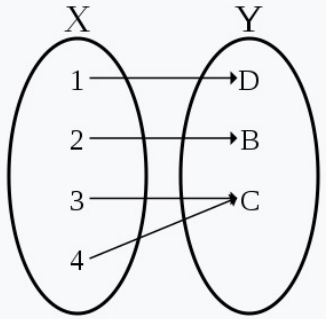
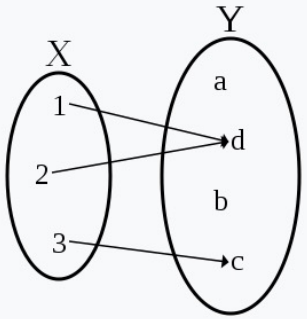
logical transpose

Surjective function ?

Bijective function ?

Sources: https://en.wikipedia.org/wiki/Bijection,_injection_and_surjection

Injective, Surjective, and Bijective functions $f: X \to Y$

	surjective	non-surjective
injective	 bijective	 injective-only
non-injective	 surjective-only	 general

Function maps each argument (element from its domain) to exactly one image (element in its codomain)

$$\forall x \in X, \exists! \, y \in Y[y = f(x)]\}$$

Injective function ("one-to-one"): each element of the codomain is mapped to by at most one element of the domain (i.e. distinct elements of the domain map to distinct elements in the codomain)

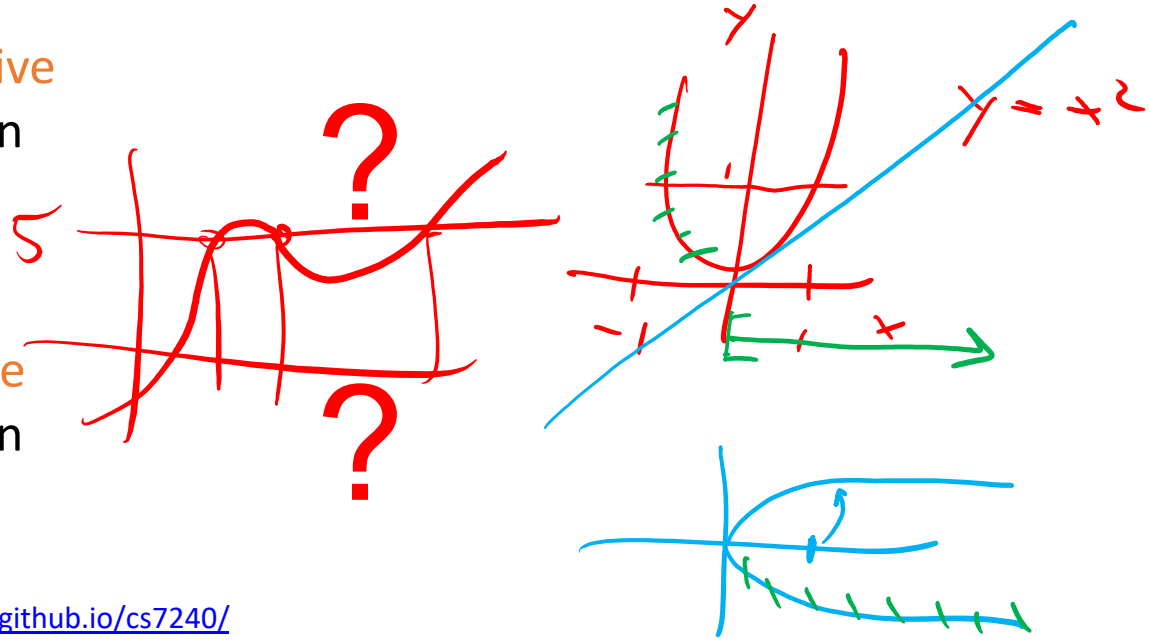
$$\forall x, x' \in X. [f(x) = f(x') \Rightarrow x = x']$$

logical transpose

$$\forall x, x' \in X. [x \neq x' \Rightarrow f(x) \neq f(x')]$$

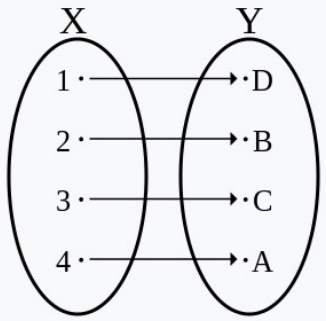
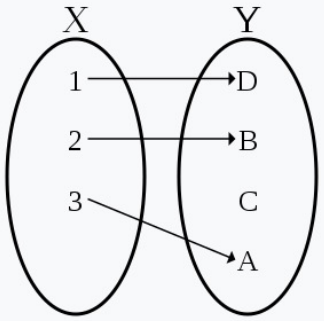
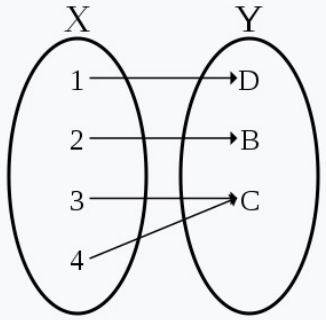
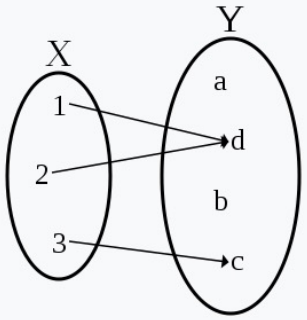
Surjective function

Bijective function



Sources: https://en.wikipedia.org/wiki/Bijection,_injection_and_surjection

Injective, Surjective, and Bijective functions $f: X \to Y$

	surjective	non-surjective
injective	 bijective	 injective-only
non-injective	 surjective-only	 general

Function maps each argument (element from its domain) to exactly one image (element in its codomain)

$$\forall x \in X, \exists! \, y \in Y[y = f(x)]\}$$

Injective function ("one-to-one"): each element of the codomain is mapped to by at most one element of the domain (i.e. distinct elements of the domain map to distinct elements in the codomain)

$$\forall x, x' \in X. [f(x) = f(x') \Rightarrow x = x']$$

logical transpose

$$\forall x, x' \in X. [x \neq x' \Rightarrow f(x) \neq f(x')]$$

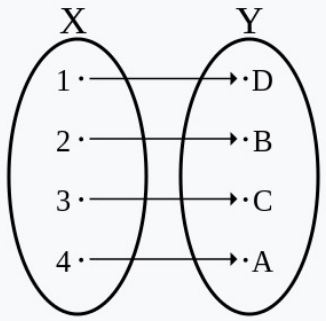
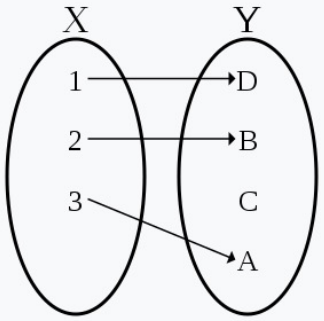
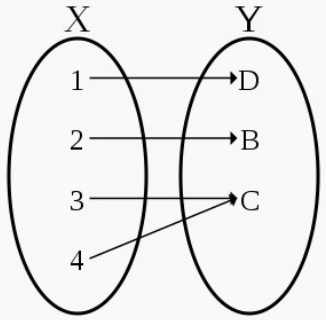
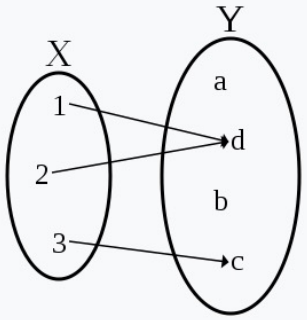
Surjective function ("onto"): each element of the codomain is mapped to by at least one element of the domain (i.e. the image and the codomain of the function are equal)

$$\forall y \in Y, x \in X[y = f(x)]$$

Bijective function ?

Sources: https://en.wikipedia.org/wiki/Bijection,_injection_and_surjection

Injective, Surjective, and Bijective functions $f: X \to Y$

	surjective	non-surjective
injective	 bijective	 injective-only
non-injective	 surjective-only	 general

Function maps each argument (element from its domain) to exactly one image (element in its codomain)

$$\forall x \in X, \exists! \, y \in Y [y = f(x)]\}$$

Injective function ("one-to-one"): each element of the codomain is mapped to by at most one element of the domain (i.e. distinct elements of the domain map to distinct elements in the codomain)

$$\forall x, x' \in X . [f(x) = f(x') \Rightarrow x = x']$$

logical transpose

$$\forall x, x' \in X . [x \neq x' \Rightarrow f(x) \neq f(x')]$$

Surjective function ("onto"): each element of the codomain is mapped to by at least one element of the domain (i.e. the image and the codomain of the function are equal)

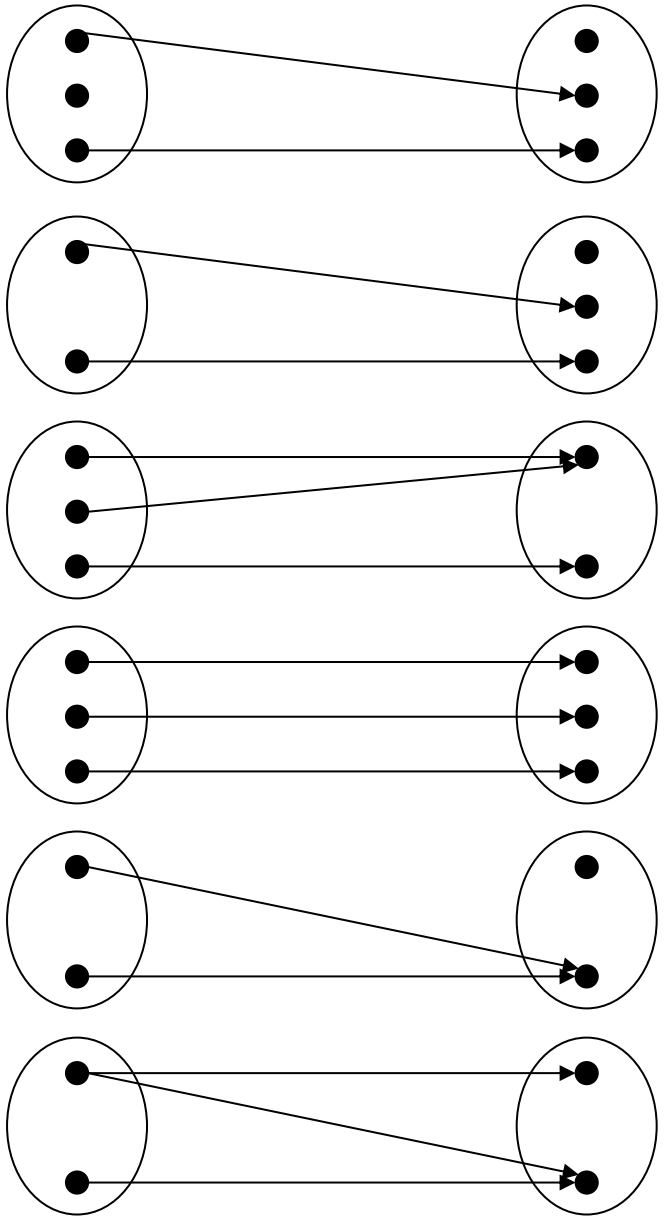
$$\forall y \in Y, x \in X [y = f(x)]$$

Bijective function ("invertible"): each element of the codomain is mapped to by exactly one element of the domain (both injective and surjective)

$$\forall y \in Y, \exists! \, x \in X [y = f(x)]\}$$

Sources: https://en.wikipedia.org/wiki/Bijection,_injection_and_surjection

Mappings: Injection, Surjection, and Bijection



?

?

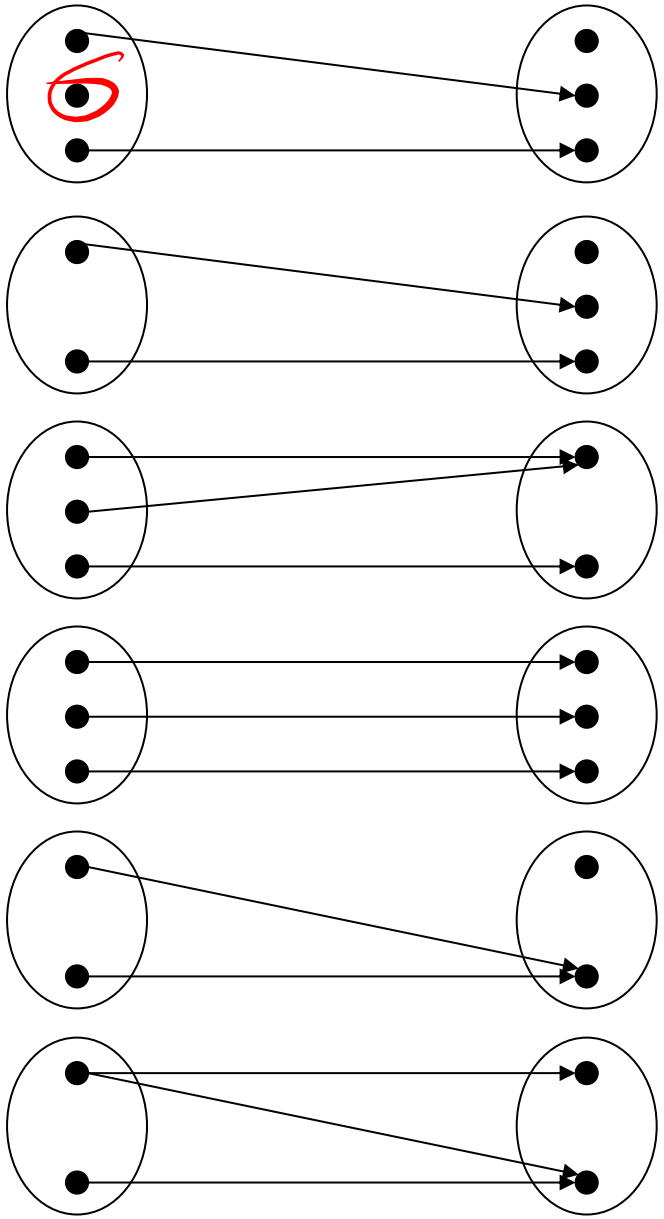
?

?

?

?

Mappings: Injection, Surjection, and Bijection



not a mapping (or function)!

?

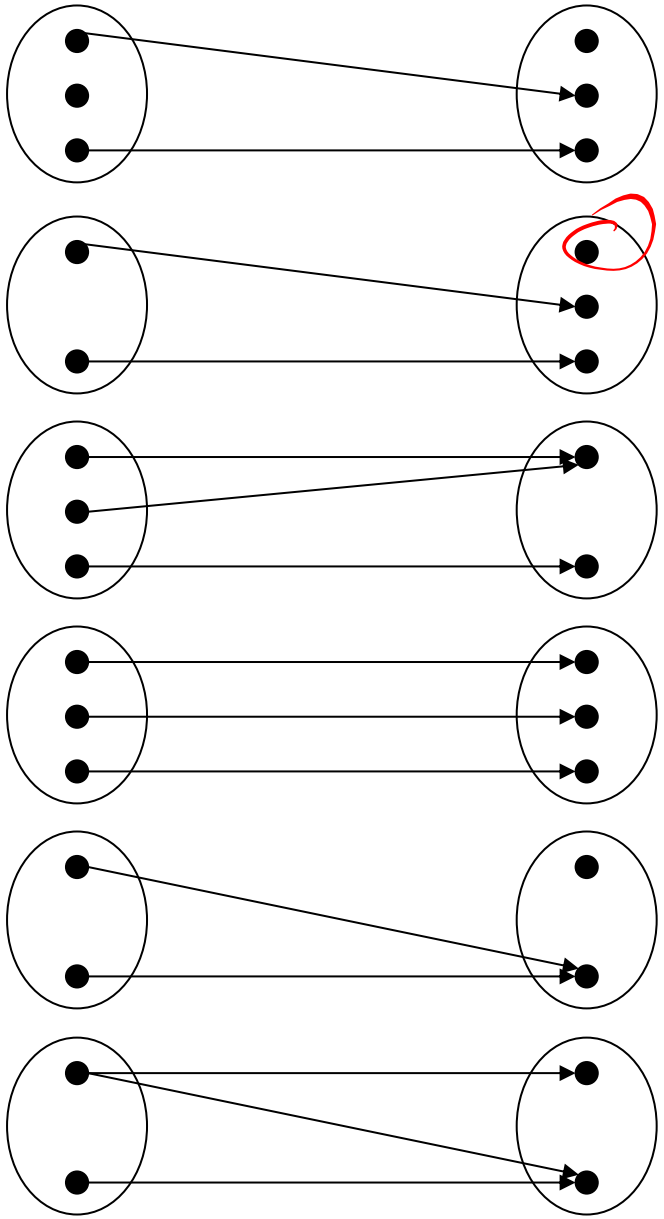
?

?

?

?

Mappings: Injection, Surjection, and Bijection



not a mapping (or function)!

injective function (or one-to-one): maps distinct elements of its domain to distinct elements of its codomain

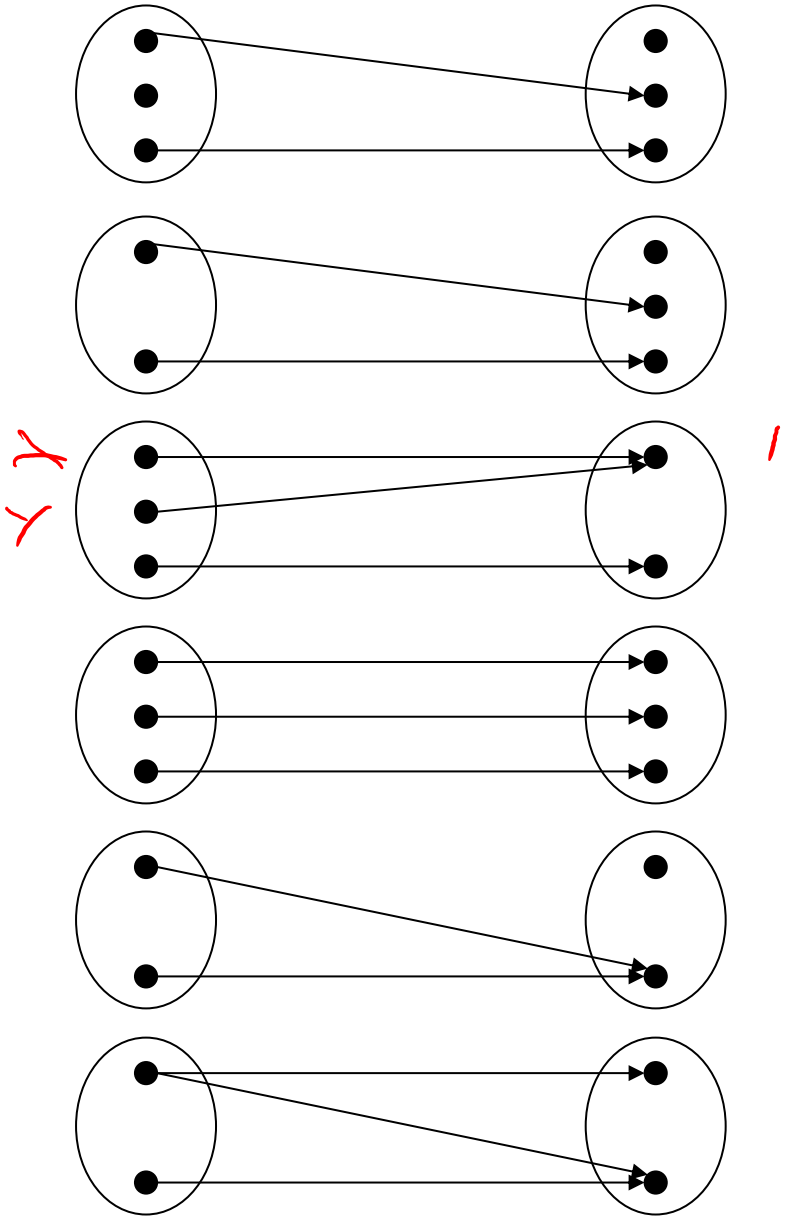
?

?

?

?

Mappings: Injection, Surjection, and Bijection



not a mapping (or function)!

injective function (or one-to-one): maps distinct elements of its domain to distinct elements of its codomain

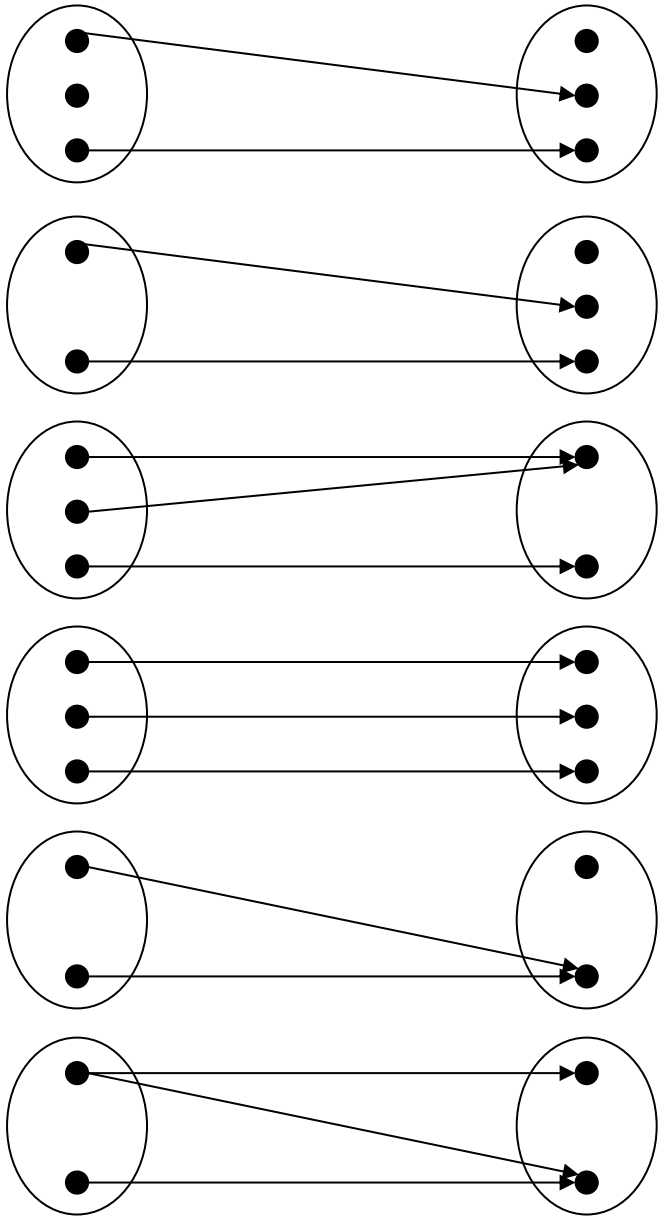
surjective (or onto): every element y in the codomain Y of f has at least one element x in the domain that maps to it

?

?

?

Mappings: Injection, Surjection, and Bijection



not a mapping (or function)!

injective function (or one-to-one): maps distinct elements of its domain to distinct elements of its codomain

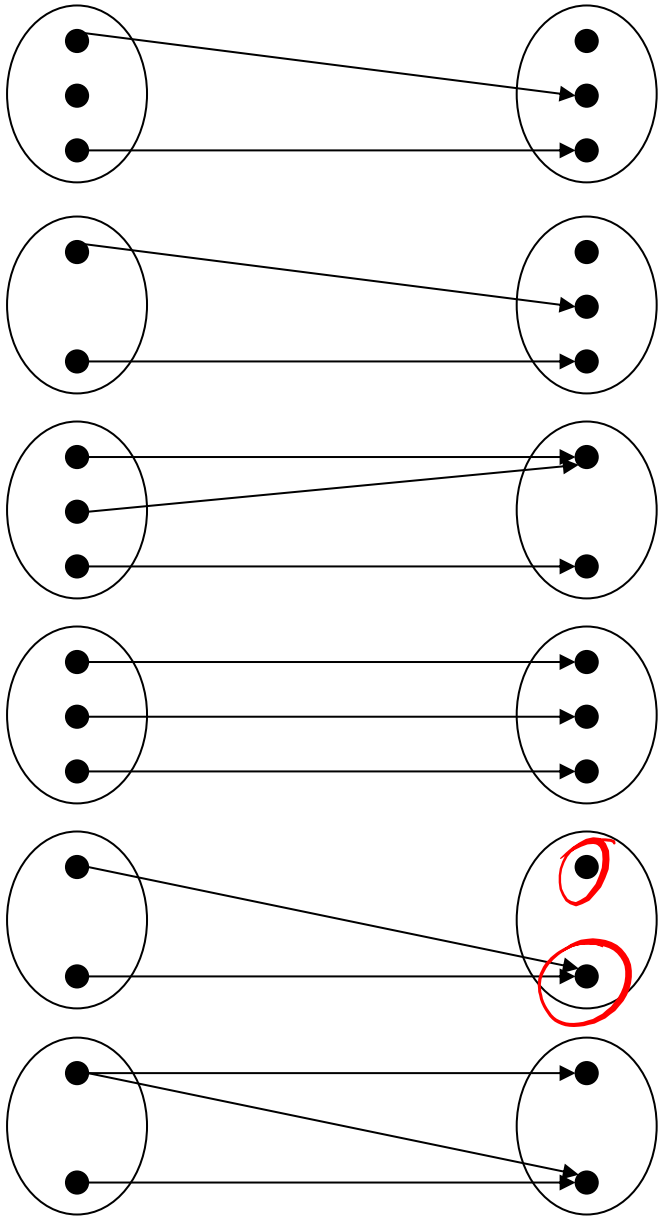
surjective (or onto): every element y in the codomain Y of f has at least one element x in the domain that maps to it

injective & surjective

?

?

Mappings: Injection, Surjection, and Bijection



not a mapping (or function)!

injective function (or one-to-one): maps distinct elements of its domain to distinct elements of its codomain

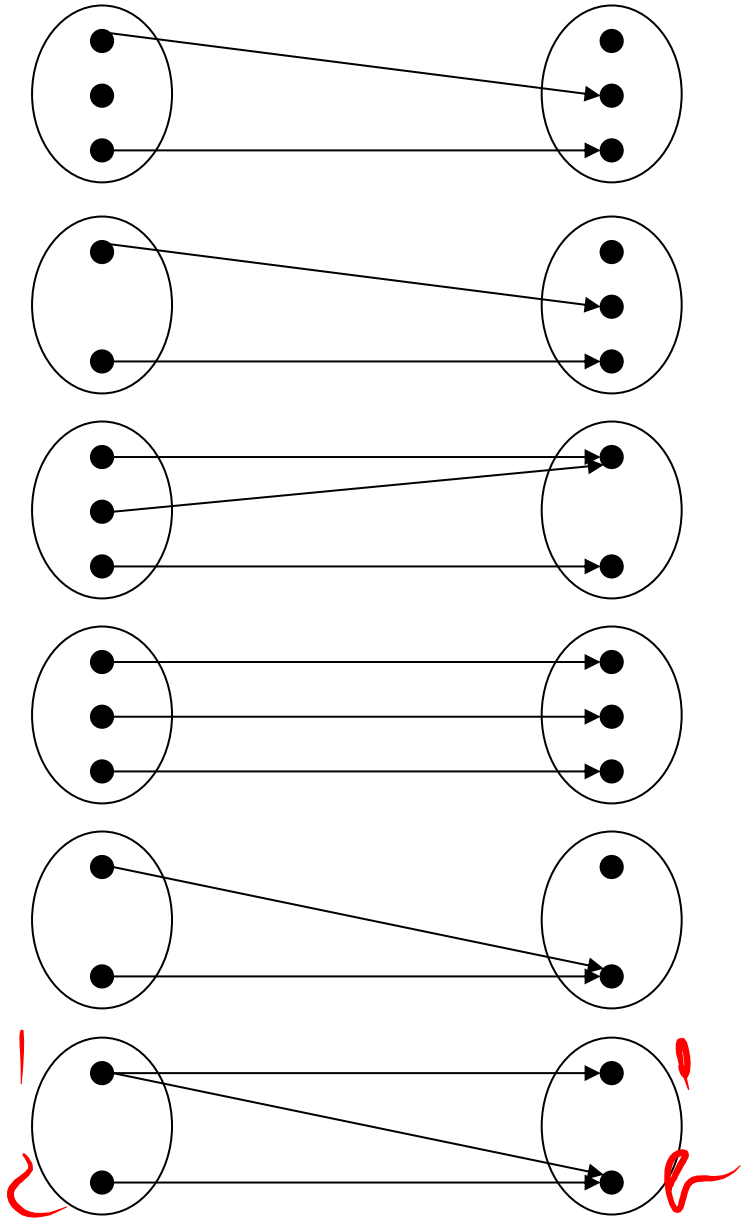
surjective (or onto): every element y in the codomain Y of f has at least one element x in the domain that maps to it

injective & surjective

neighter

?

Mappings: Injection, Surjection, and Bijection



not a mapping (or function)!

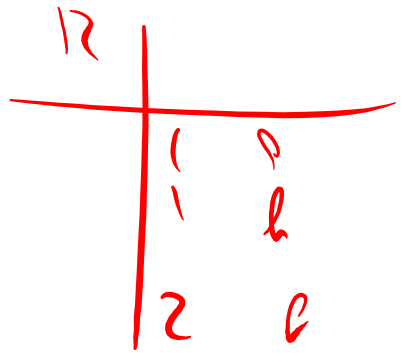
injective function (or one-to-one): maps distinct elements of its domain to distinct elements of its codomain

surjective (or onto): every element y in the codomain Y of f has at least one element x in the domain that maps to it

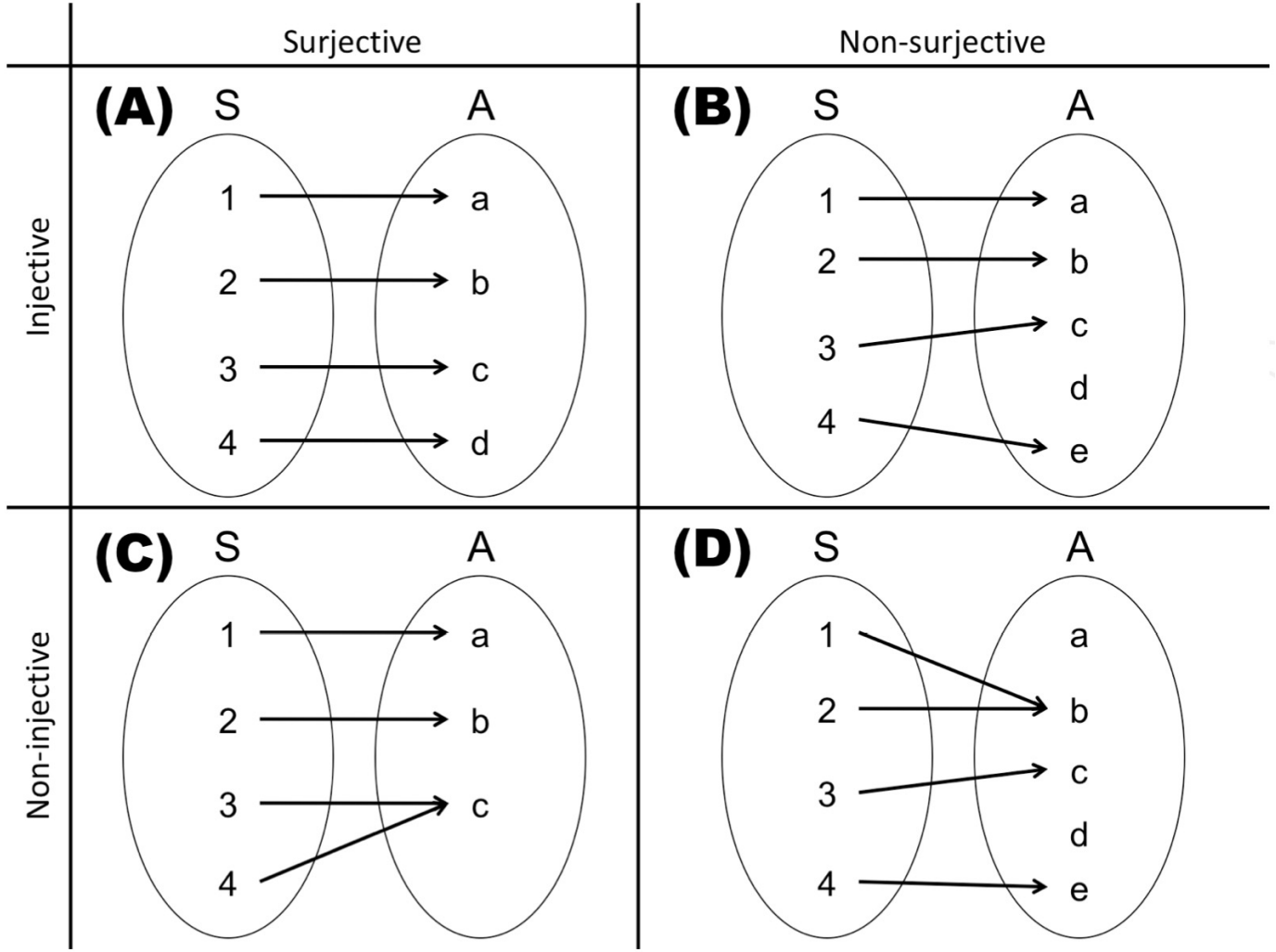
injective & surjective

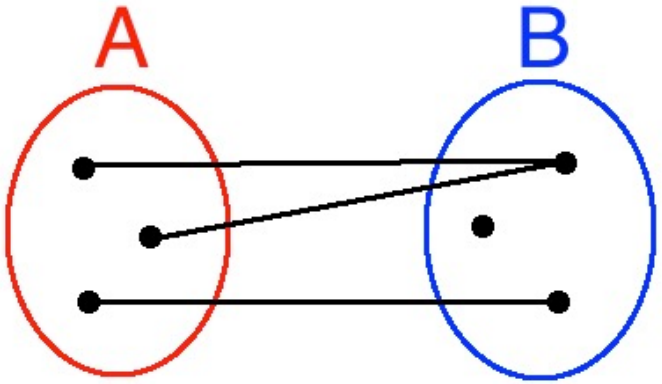
neighter

not even a mapping!

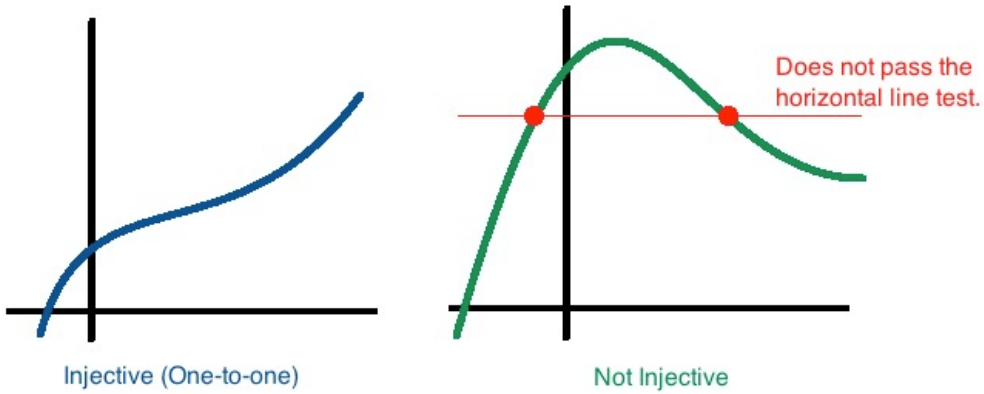


Bijection, Injection, and Surjection



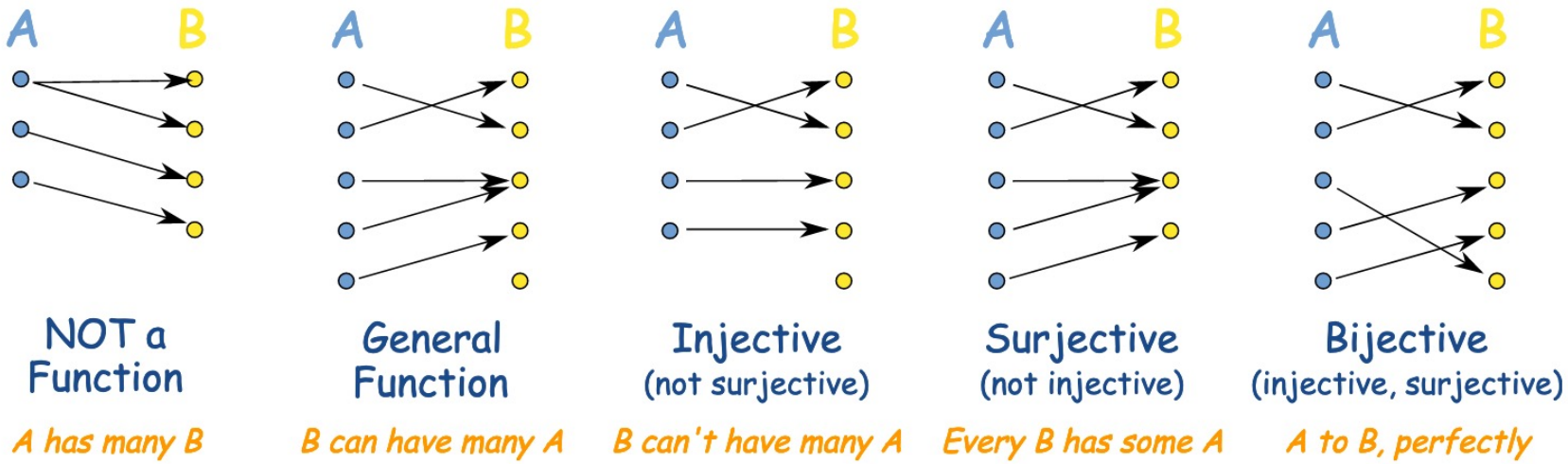


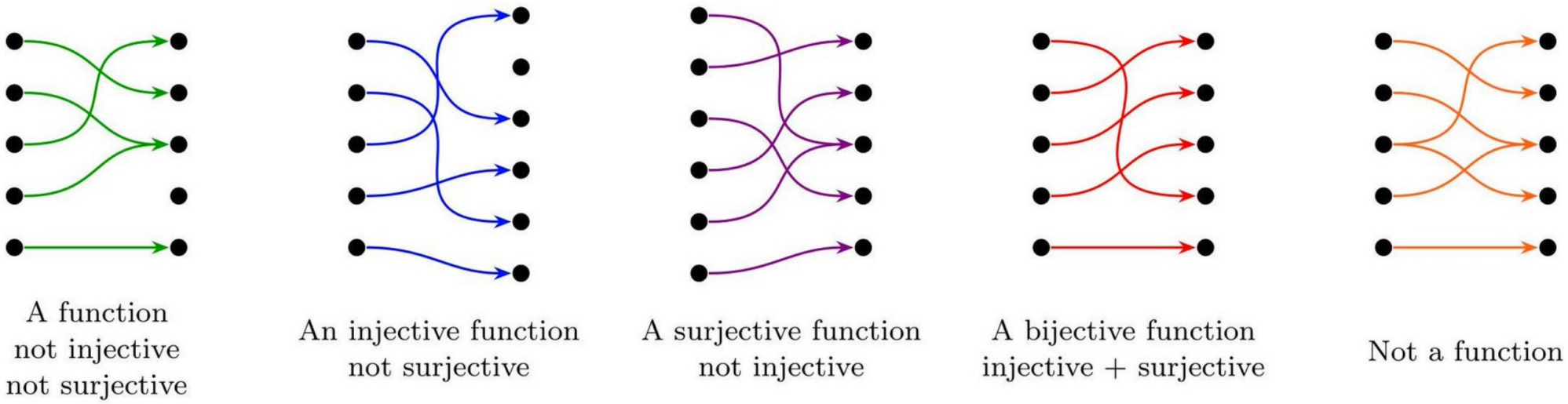
Neither Injective or Surjective
Two elements in set A maps to the same element in set B (not injective), and one element in set B is not in the image or range of the function that maps set A to B (not surjective).



Sources: https://www.intechopen.com/books/protein-interactions/relating-protein-structure-and-function-through-a-bijection-and-its-implications-on-protein-structur, http://mathonline.wikidot.com/injections-surjections-and-bijections, https://northeastern-datalab.github.io/cs7240/

Bijection, Injection, and Surjection





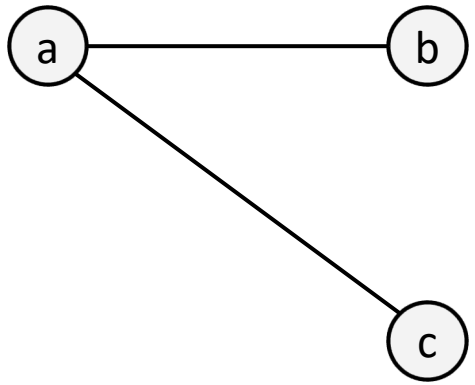
Sources: https://www.mathsisfun.com/sets/injective-surjective-bijective.html, https://twitter.com/jdhamkins/status/841318019397779456,

We make a detour to Graph matching

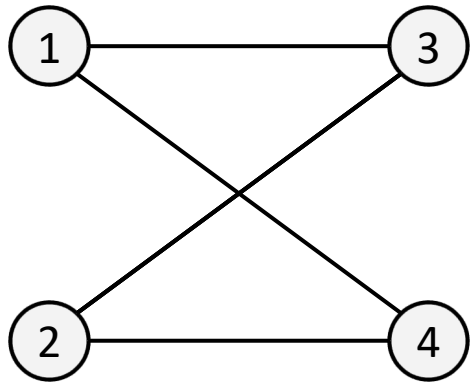
- Finding a correspondence between the nodes and the edges of two graphs that satisfies some (more or less stringent) constraints

Homomorphism

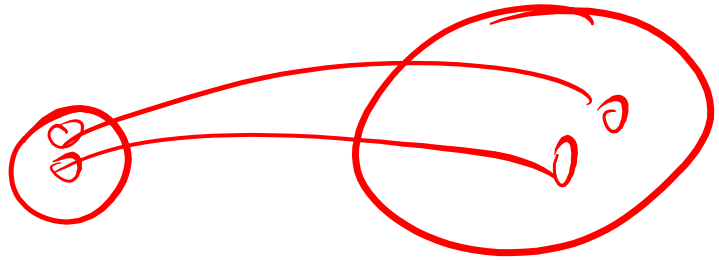
- A graph homomorphism h from graph $G(V_G,E_G)$ to $H(V_H,E_H)$, is a mapping from V_G to V_H such that $\{x,y\} \in E_G$ implies $\{h(x),h(y)\} \in E_H$
  - "edge-preserving": if two nodes in G are linked by an edge, then they are mapped to two nodes in H that are also linked



G



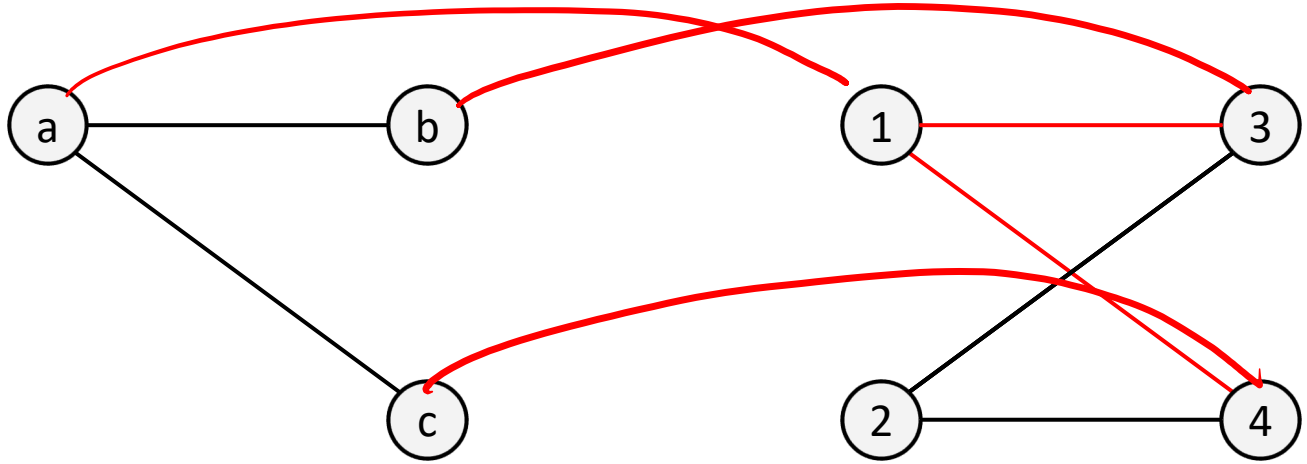
H



Is there a homomorphism from G to H ?

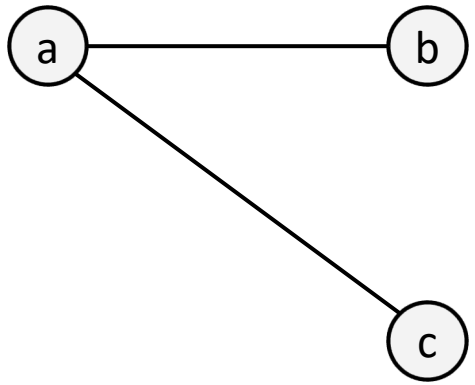
Homomorphism

- A graph homomorphism *h* from graph $G(V_G,E_G)$ to $H(V_H,E_H)$, is a mapping from V_G to V_H such that $\{x,y\} \in E_G$ implies $\{h(x),h(y)\} \in E_H$
  - "edge-preserving": if two nodes in *G* are linked by an edge, then they are mapped to two nodes in *H* that are also linked

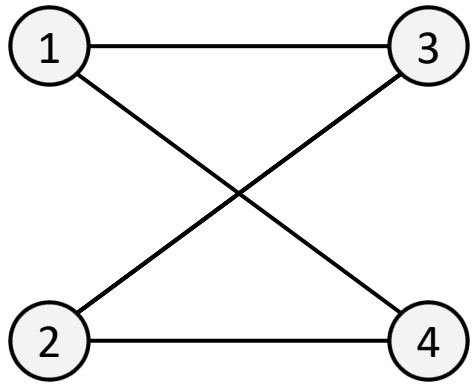


Homomorphism

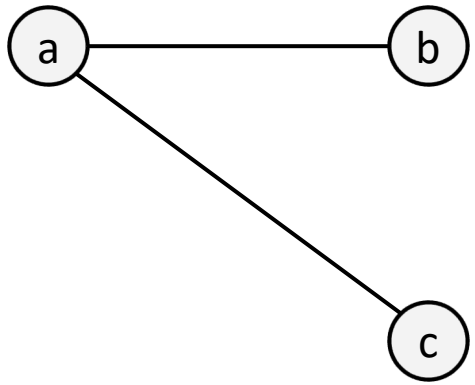
- A graph homomorphism h from graph $G(V_G,E_G)$ to $H(V_H,E_H)$, is a mapping from V_G to V_H such that $\{x,y\} \in E_G$ implies $\{h(x),h(y)\} \in E_H$
  - "edge-preserving": if two nodes in G are linked by an edge, then they are mapped to two nodes in H that are also linked



G



H



G

h: {(a,1), (b,3), (c,4)}

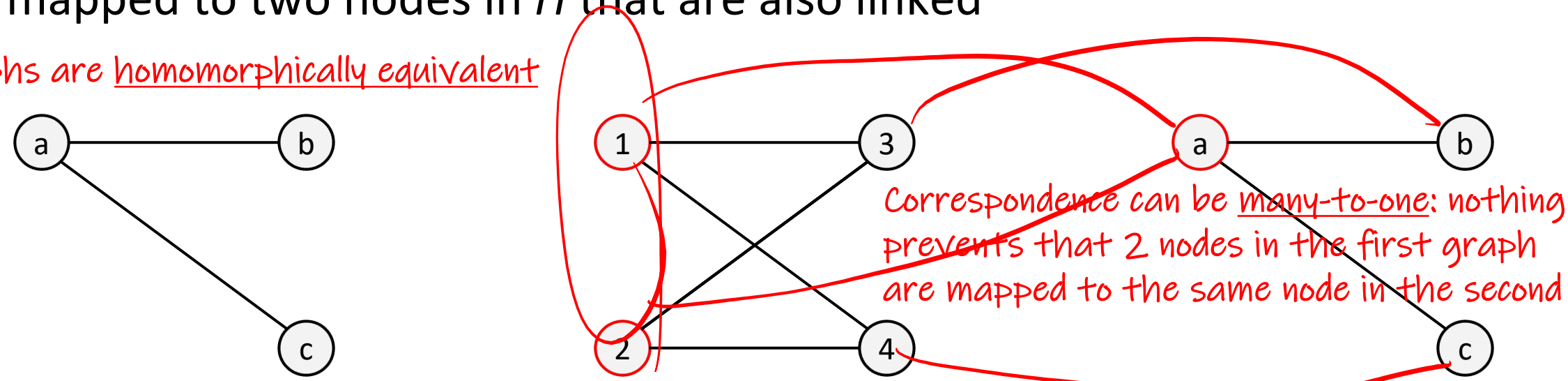
does not need to be surjective!

Is there a homomorphism from H to G ?

Homomorphism

- A graph homomorphism *h* from graph $G(V_G,E_G)$ to $H(V_H,E_H)$, is a mapping from V_G to V_H such that $\{x,y\} \in E_G$ implies $\{h(x),h(y)\} \in E_H$
  - "edge-preserving": if two nodes in *G* are linked by an edge, then they are mapped to two nodes in *H* that are also linked

Graphs are homomorphically equivalent



G H G

h: {(a,1), (b,3), (c,4)}

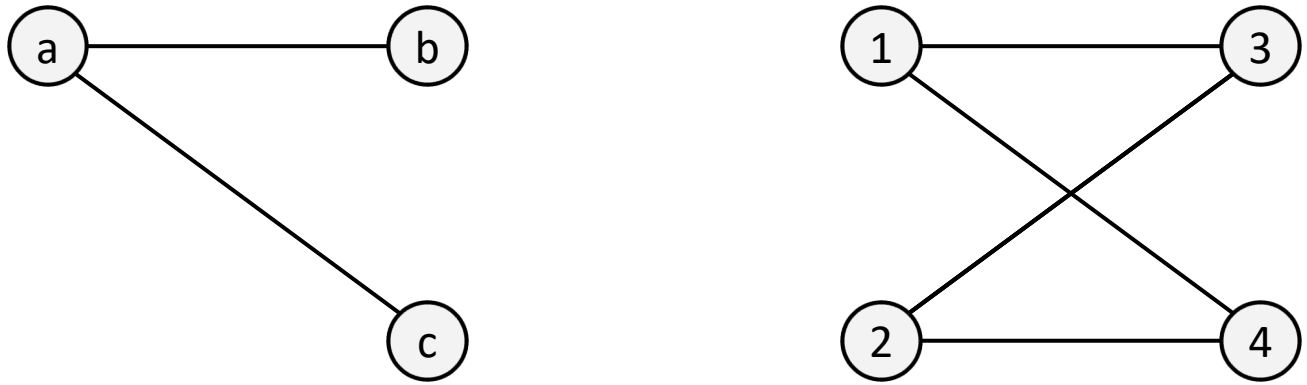
does not need to be surjective!

h: {(1,a), (2,a), (3,b), (4,c)}

does not need to be injective!

Graph Isomorphism

- Graphs $G(V_G,E_G)$ and $H(V_H,E_H)$ are isomorphic iff there is an invertible h from V_G to V_H s.t. $\{x,y\} \in E_G$ iff $\{h(u),h(v)\} \in E_H$
  - We need to find a one-to-one correspondence

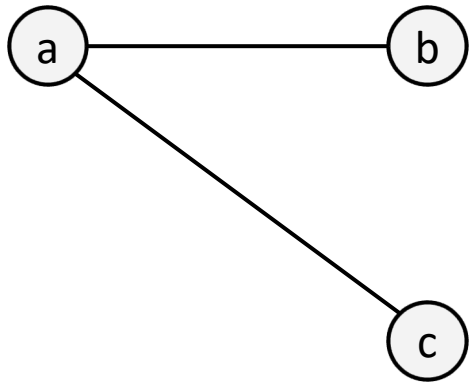


G H

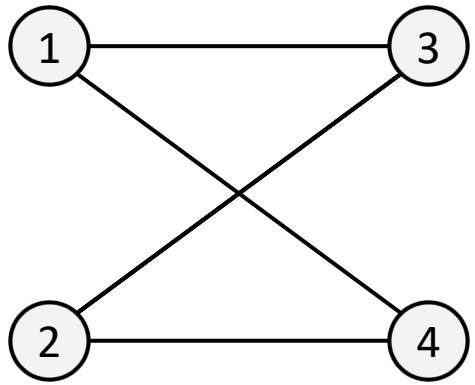
Is there an isomorphism from G to H ?

Graph Isomorphism

- Graphs $G(V_G,E_G)$ and $H(V_H,E_H)$ are isomorphic iff there is an invertible h from V_G to V_H s.t. $\{x,y\} \in E_G$ iff $\{f(u),f(v)\} \in E_H$
  - We need to find a one-to-one correspondence



G



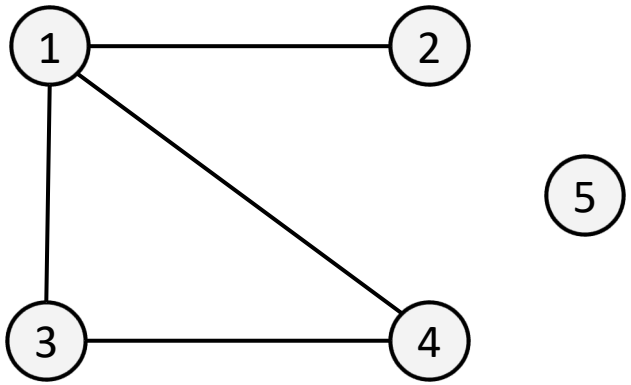
H

Is there an isomorphism from G to H?

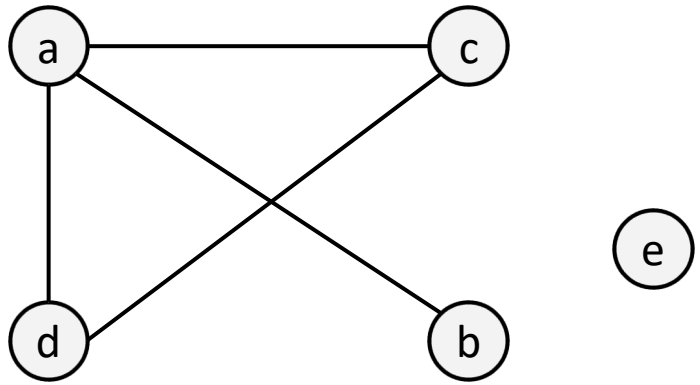
They are homomorphically equivalent, but not isomorphic!

Graph Isomorphism

- Graphs $G(V_G,E_G)$ and $H(V_H,E_H)$ are isomorphic iff there is an invertible h from V_G to V_H s.t. $\{x,y\} \in E_G$ iff $\{f(u),f(v)\} \in E_H$
  - We need to find a one-to-one correspondence



G

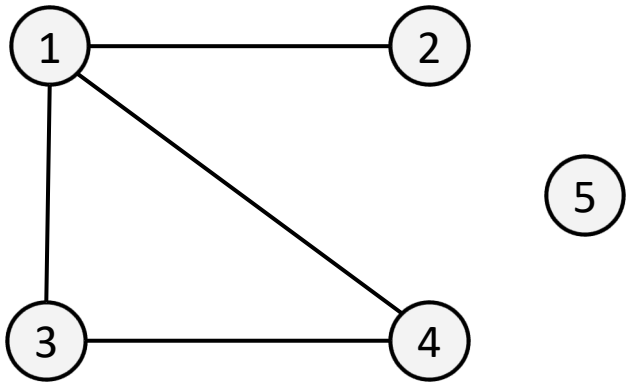


H

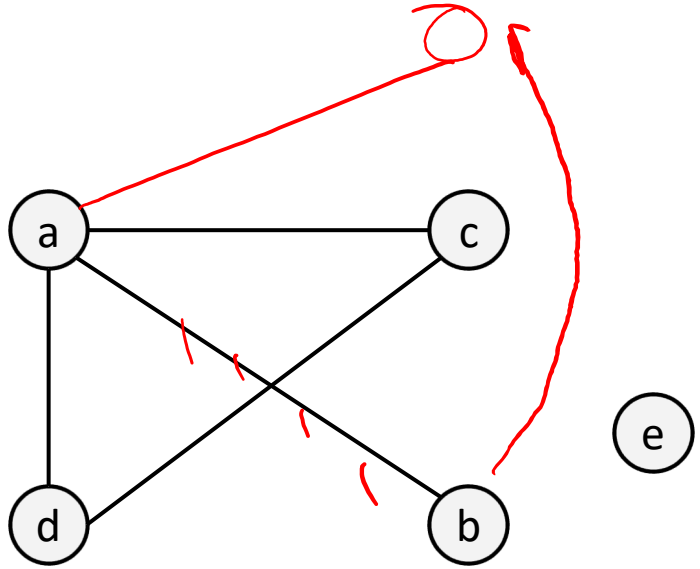
Is there an isomorphism from G to H?

Graph Isomorphism

- Graphs $G(V_G,E_G)$ and $H(V_H,E_H)$ are isomorphic iff there is an invertible h from V_G to V_H s.t. $\{x,y\} \in E_G$ iff $\{f(u),f(v)\} \in E_H$
  - We need to find a one-to-one correspondence



G



H

Is there an isomorphism from G to H?

Yes:

h: {(1,a), (2,b), (3,d), (4,c), (5,e)}

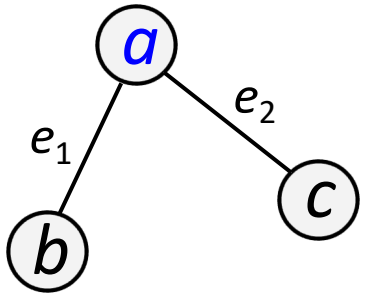
bijection = surjective and injective mapping

Another perspective on absorption and subsumption

Absorption (or the challenge with self-joins)

f is true if there is an edge

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

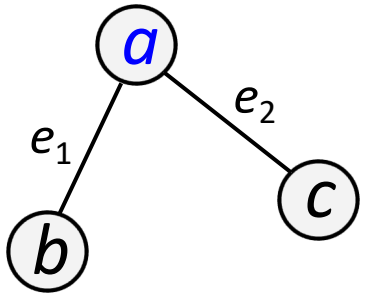


$f =$?

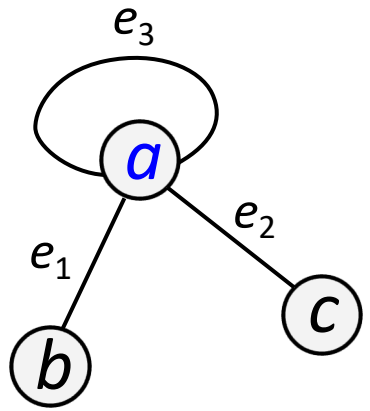
Absorption (or the challenge with self-joins)

f is true if there is an edge

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



$f = ab \vee ac$

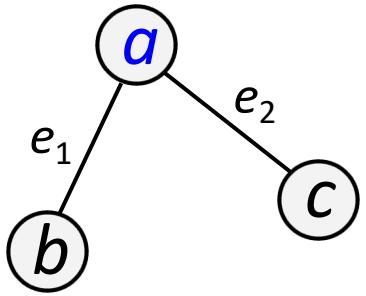


$f =$?

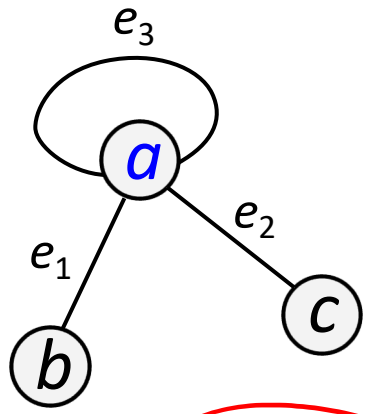
Absorption (or the challenge with self-joins)

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*

$(\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \Rightarrow \varphi_2)$

$ac \vee a \quad \wedge (ac \Rightarrow a)$

Absorption

Absorption

$(A \vee B) \wedge A = A$ $\quad$ $(A \min B) \max A = A$ $\quad$ $A \leq B \Leftrightarrow A \min B = A$

$(A \vee B) \wedge (A \Rightarrow B) = A$ $\quad$ $(A \min B) = A$, if $A \leq B$ $\quad$ $\sim A \wedge B = A$

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

Updated 3/2/2021

Topic 2: Complexity of Query Evaluation
Unit 1: Conjunctive Queries
Lecture 12

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp21)

https://northeastern-datalab.github.io/cs7240/sp21/

3/2/2021

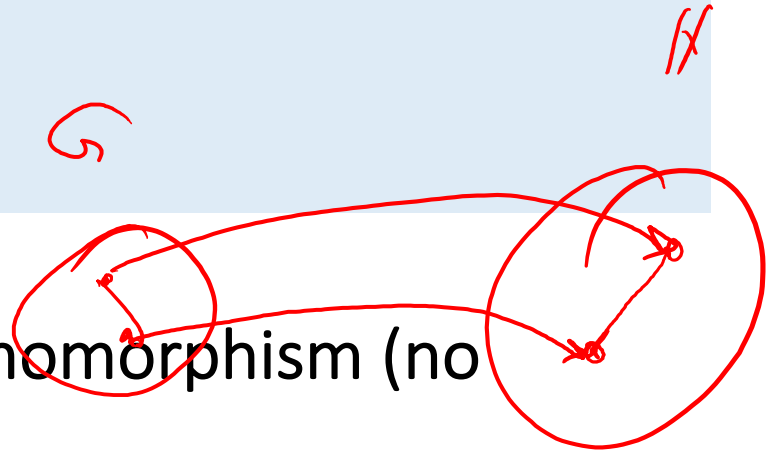
Outline: T2-1/2: Query Evaluation & Query Equivalence

- 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
  - Nested queries
  - Tree pattern queries

Graph Homomorphism beyond graphs

Definition : *Let G and H be graphs. A homomorphism of G to H is a function* $f: V(G) \to V(H)$ such that

$$(x,y) \in E(G) \Rightarrow (f(x),f(y)) \in E(H).$$

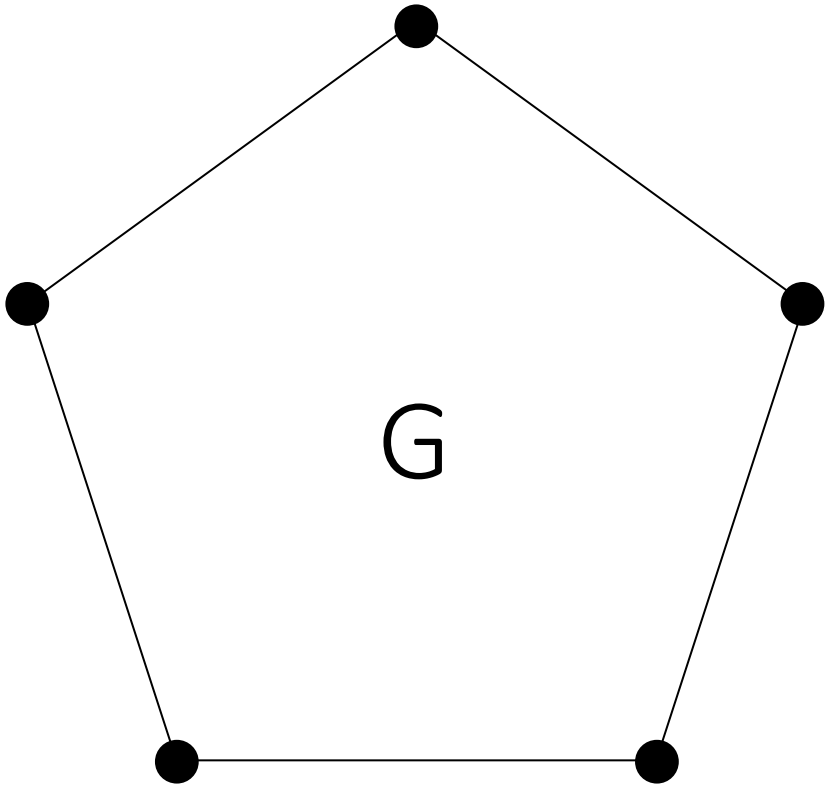


We sometimes write G → H (G ↛ H) if there is a homomorphism (no homomorphism) of G to H

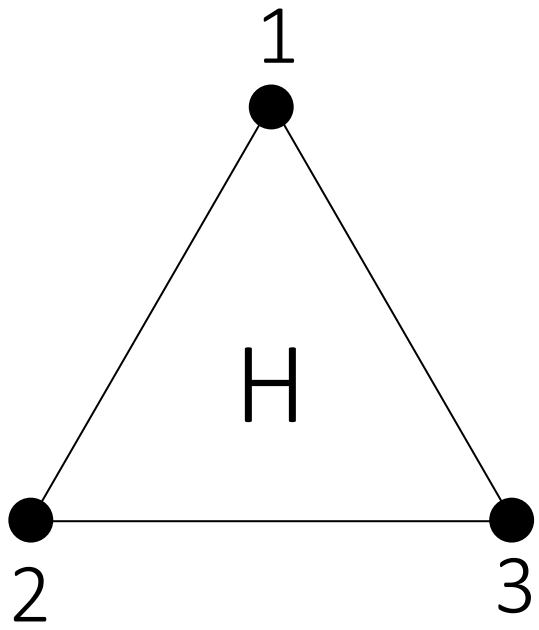
Definition of a homomorphism naturally extends to:

- digraphs (directed graphs)
- edge-colored graphs
- relational systems
- constraint satisfaction problems (CSPs)

An example

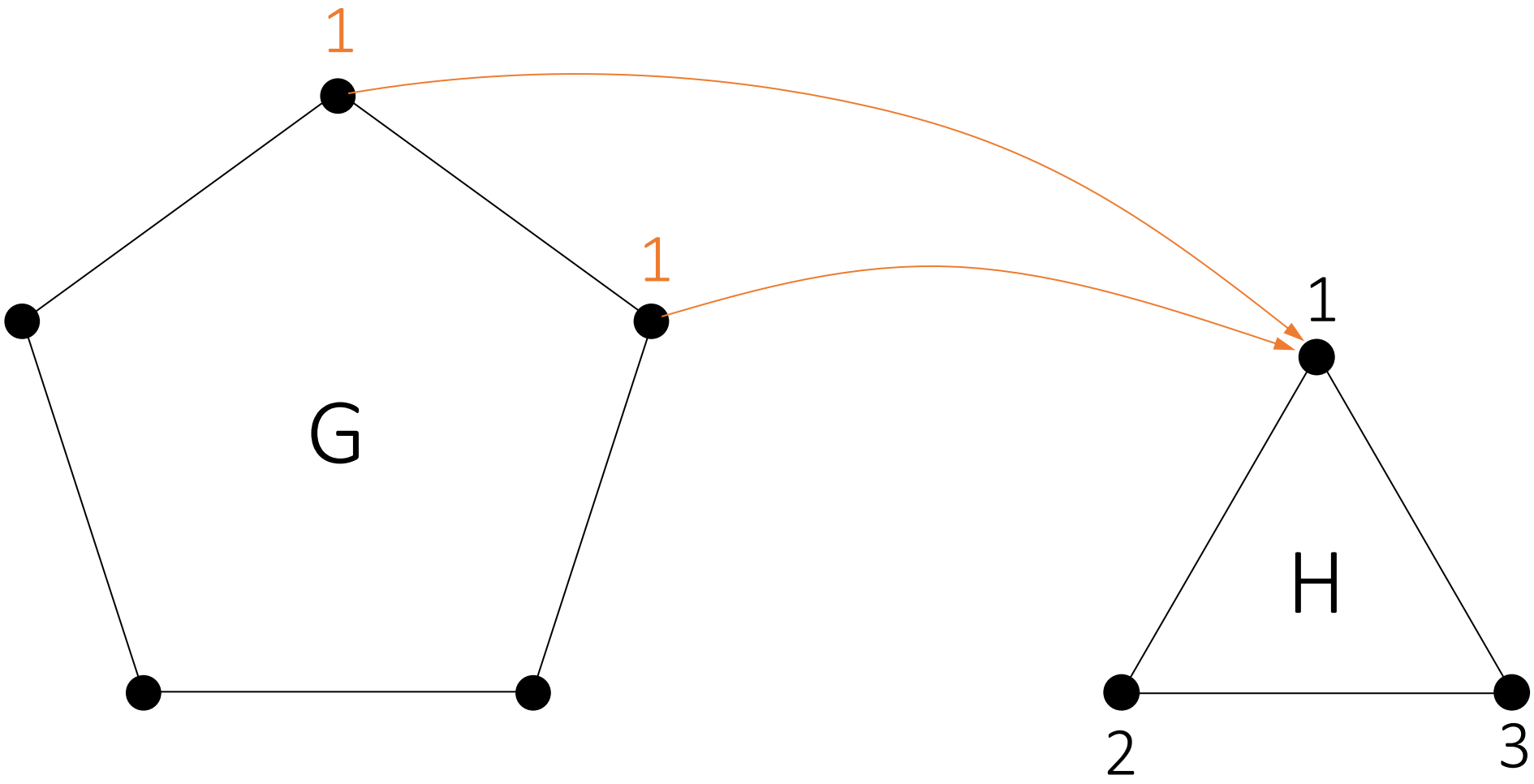


3 "colors" of the vertices



Example by Rick Brewster, Graph homomorphism tutorial, 2006

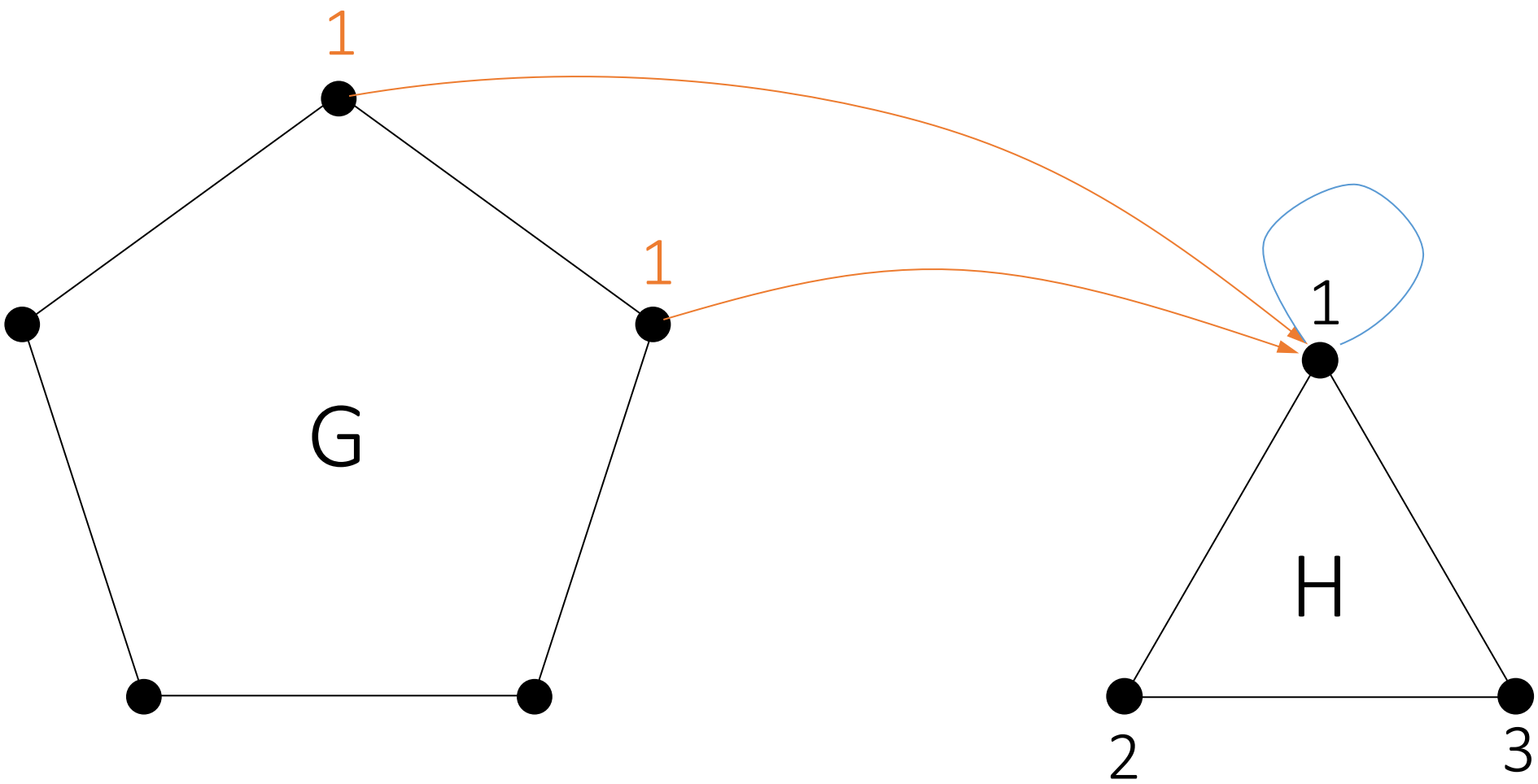
An example



Can this assignment be extended to a homomorphism? ?

Example by Rick Brewster, Graph homomorphism tutorial, 2006

An example

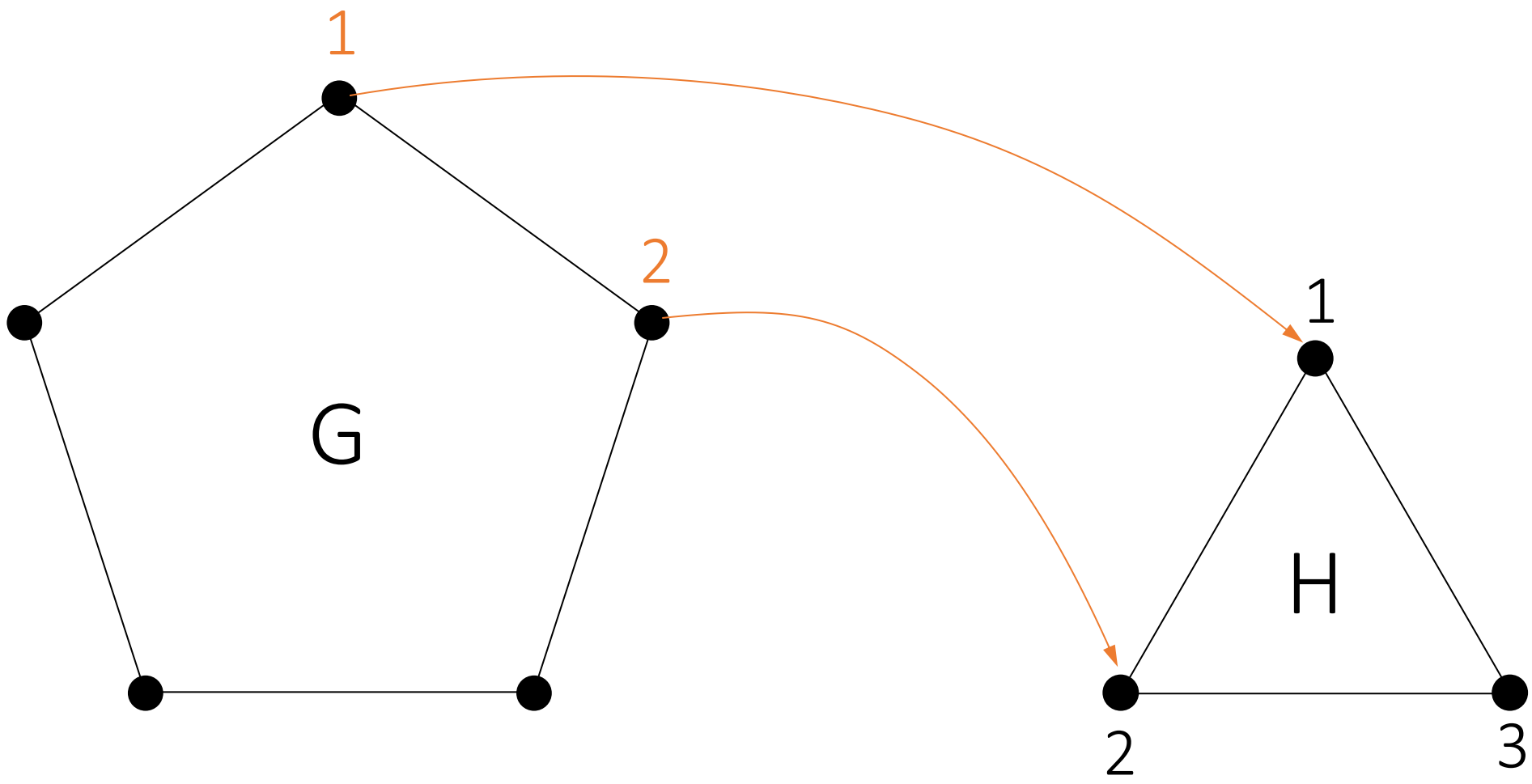


Can this assignment be extended to a homomorphism?

No, this assignment requires a loop on vertex 1 (in H)

Example by Rick Brewster, Graph homomorphism tutorial, 2006

An example



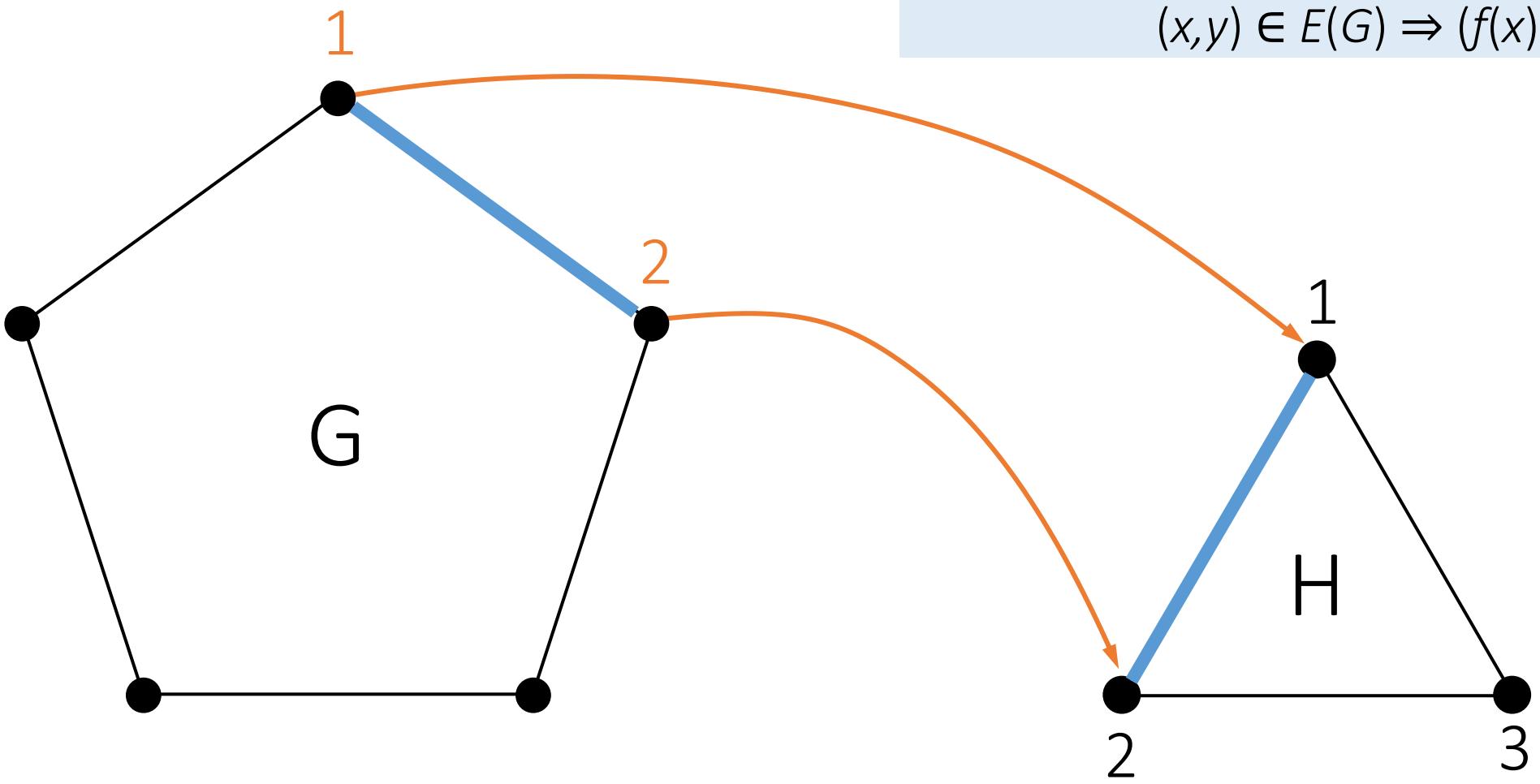
Can this assignment be extended to a homomorphism? ?

Example by Rick Brewster, Graph homomorphism tutorial, 2006

An example

Definition: *Let G and H be graphs. A homom. of G to H is a function f: V(G) → V(H)* s.t. that

$(x,y) \in E(G) \Rightarrow (f(x),f(y)) \in E(H)$.



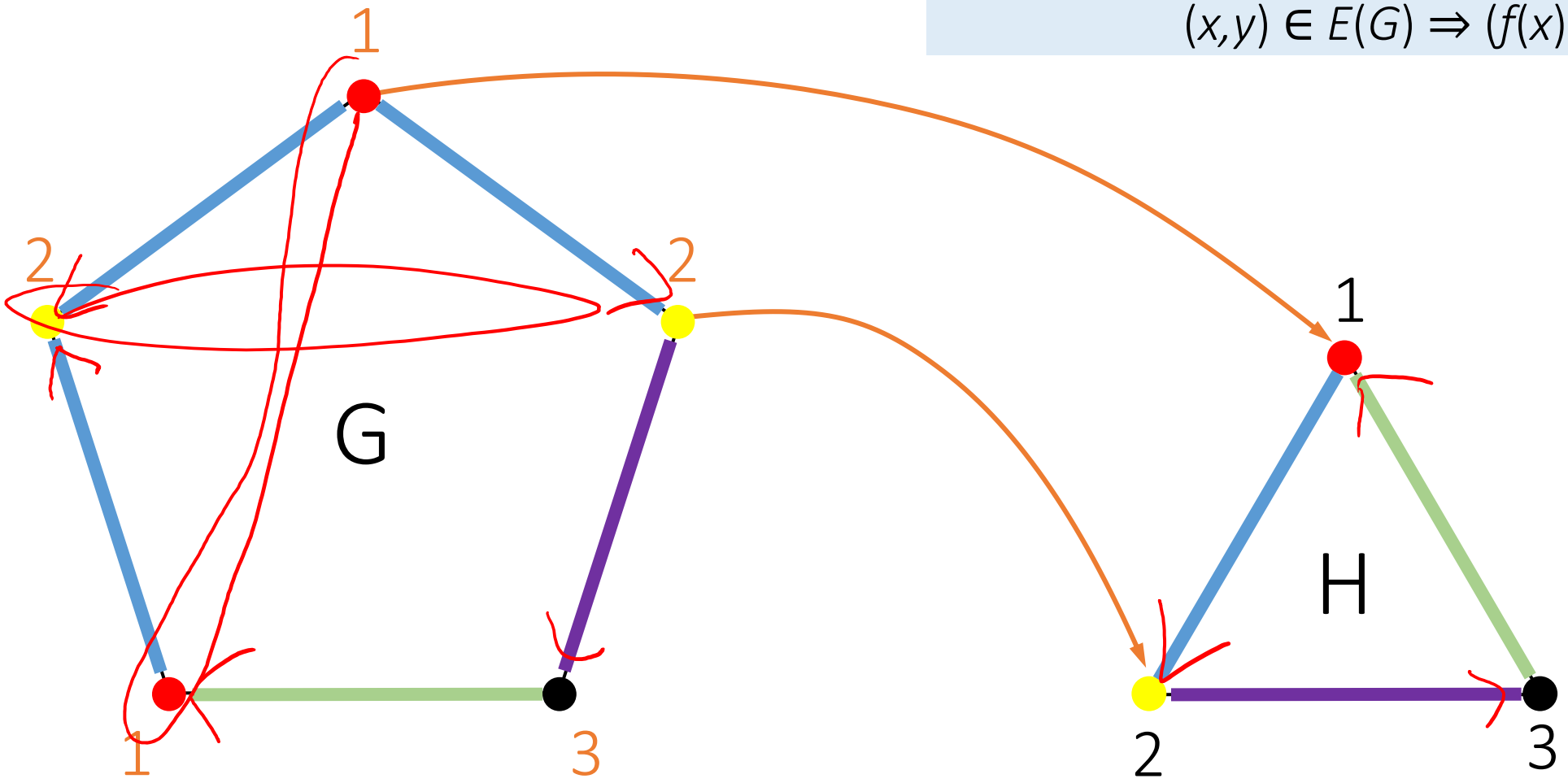
Can this assignment be extended to a homomorphism? ?

Example by Rick Brewster, Graph homomorphism tutorial, 2006

An example

Definition: *Let G and H be graphs. A homom. of G to H is a function f: V(G) → V(H)* s.t. that

$(x,y) \in E(G) \Rightarrow (f(x),f(y)) \in E(H)$.

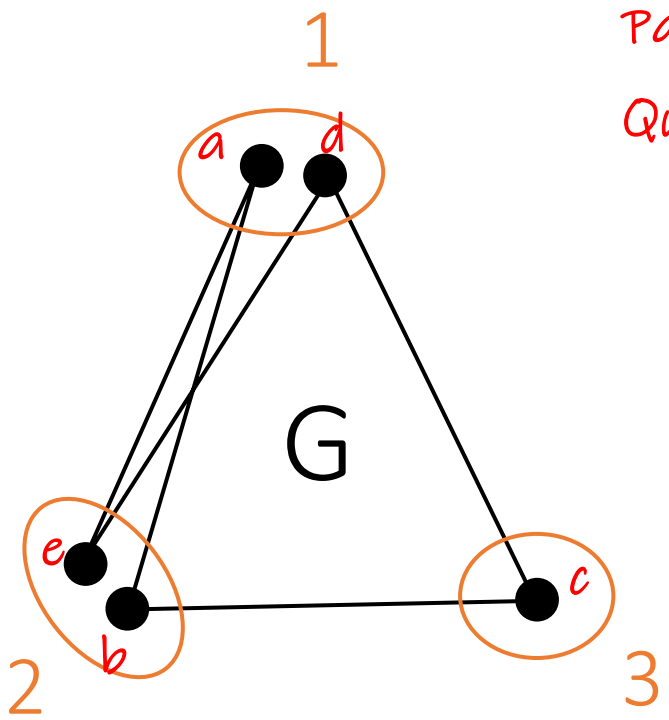


Example by Rick Brewster, Graph homomorphism tutorial, 2006

An example

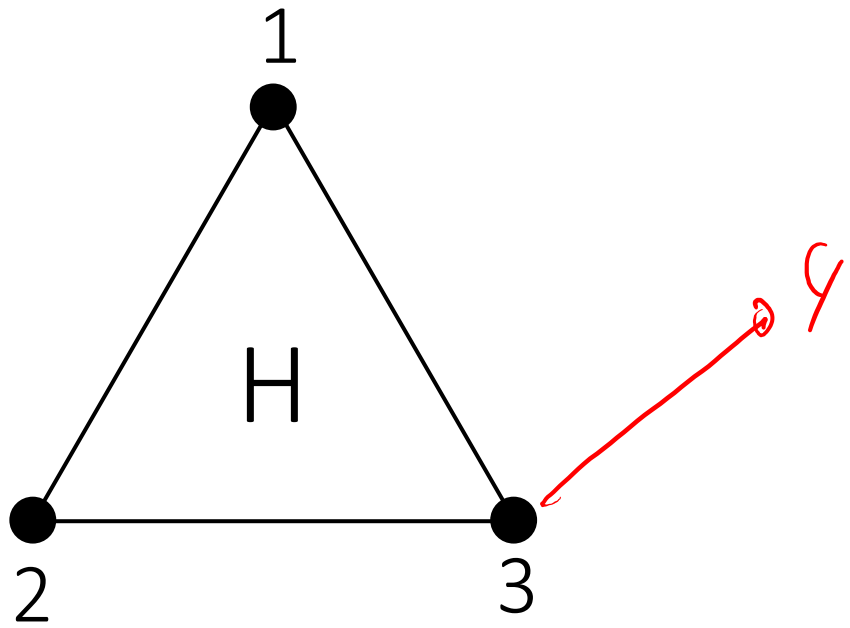
Basically a partitioning problem!

The quotient set of the partition (set of equivalence classes of the partition) is a subgraph of H.



Partition: {{a,d}, {b,e}, {c}}

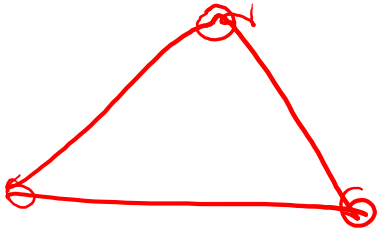
Quotient set: {[a], [b], [c]}



Example by Rick Brewster, Graph homomorphism tutorial, 2006

Some observations

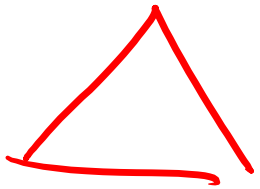
When does G → K_3 hold? (K_3 = 3-clique = triangle)



?

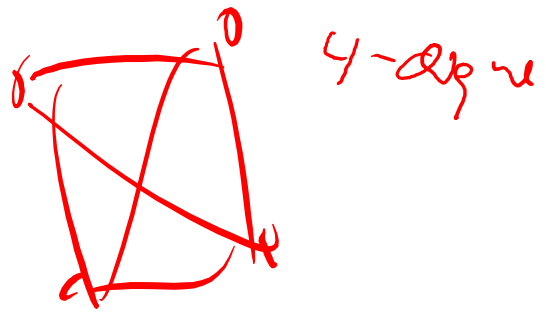
Some observations

When does G → K_3 hold? (K_3 = 3-clique = triangle)



iff G is 3-colorable

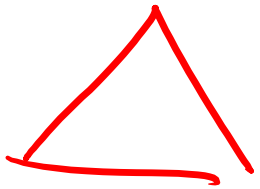
When does G → K_n hold? (K_n = n-clique)



?

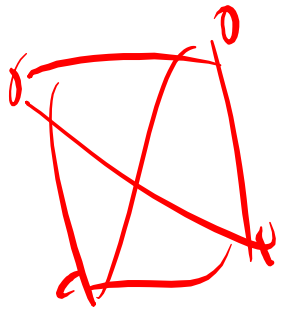
Some observations

When does G → K_3 hold? (K_3 = 3-clique = triangle)



iff G is 3-colorable

When does G → K_n hold? (K_n = n-clique)



iff G is n-colorable

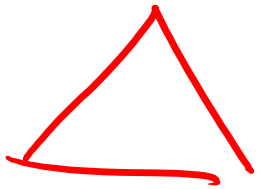
Thus homomorphisms generalize colorings:
Notation: G → H is an H-coloring of G.

What is the complexity of testing for the existence of a homomorphism?

?

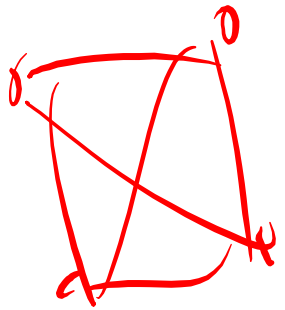
Some observations

When does G → K_3 hold? (K_3 = 3-clique = triangle)



iff G is 3-colorable

When does G → K_n hold? (K_n = n-clique)



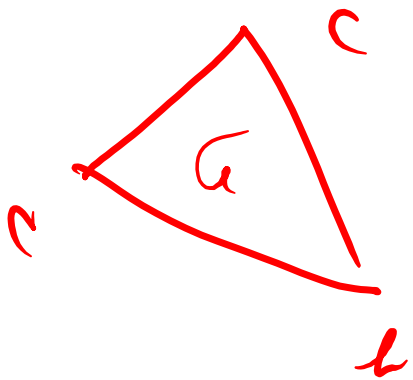
iff G is n-colorable

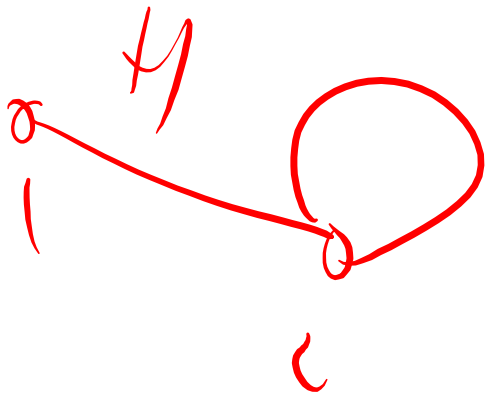
Thus homomorphisms generalize colorings:
Notation: G → H is an H-coloring of G.

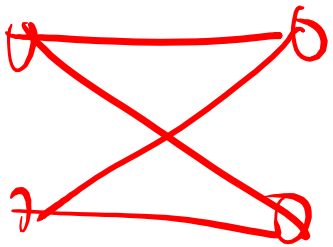
What is the complexity of testing for the existence of a homomorphism?

NP-complete

The complexity of H-coloring

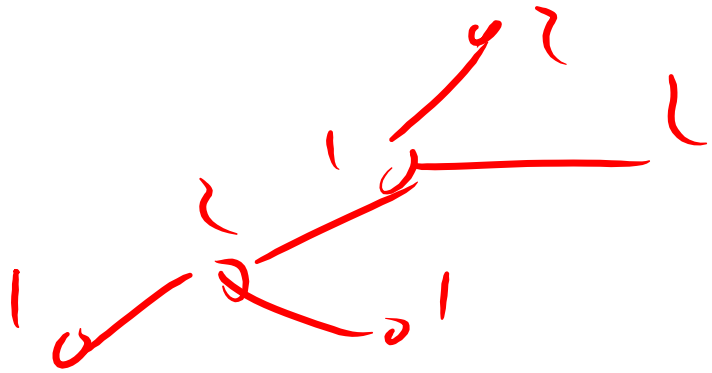






Let H be a fixed graph.
H-coloring
Instance: A graph G.
Question: Does G admit an H-coloring?

Theorem [Hell, Nesetril'90]:
If H is bipartite or contains a loop, then H-colouring is polynomial time solvable; otherwise, H is NP-complete.



[Hell, Nesetril'90]: Hell, Nešetřil. On the complexity of H-coloring. Journal of Combinatorial Theory, 1990. https://doi.org/10.1016/0095-8956(90)90132-J

Repeated variable names

REPEATED SLIDE

In sentences with multiple quantifiers, distinct variables do not need to range over distinct objects! (cp. homomorphism vs. isomorphism)

? which of the following formulas imply each other?

$\forall x.\forall y.\ E(x,y)$

$\forall x.\ E(x,x)$

$\exists x.\exists y.\ E(x,y)$

$\exists x.\ E(x,x)$

Repeated variable names

REPEATED SLIDE

In sentences with multiple quantifiers, distinct variables do not need to range over distinct objects! (cp. homomorphism vs. isomorphism)

$$\forall x.\forall y.\ E(x,y) \quad \Rightarrow \quad \forall x.\ E(x,x)$$

$$\exists x.\exists y.\ E(x,y) \quad \Leftarrow \quad \exists x.\ E(x,x)$$

E

s	t
1	1
1	2
2	1
2	2

E

s	t
1	1

Repeated variable names

In sentences with multiple quantifiers, distinct variables do not need to range over distinct objects! (cp. homomorphism vs. isomorphism)

$$\forall x.\forall y.\ E(x,y) \quad \Rightarrow \quad \forall x.\ E(x,x)$$

$$\Downarrow \qquad \Downarrow$$

Only if domain is not empty!
Dom ≠ ∅

$$\exists x.\exists y.\ E(x,y) \quad \Leftarrow \quad \exists x.\ E(x,x)$$

E

s	t

A more abstract (general) view on homomorphisms

Homomorphisms on Binary Structures

- **Definition (Binary algebraic structure):** A binary algebraic structure is a set together with a binary operation on it. This is denoted by an ordered pair $(S,\star)$ in which S is a set and $\star$ is a binary operation on S.

- **Definition (homomorphism of binary structures):** Let $(S,\star)$ and $(S',\circ)$ be binary structures. A homomorphism from $(S,\star)$ to $(S',\circ)$ is a map $h: S \longrightarrow S'$ that satisfies, for all x, y in S:

  $$h(x \star y) = h(x) \circ h(y)$$

- We can denote it by $h: (S,\star) \longrightarrow (S',\circ)$.

Example: from addition to multiplication

- Let $f(x) = e^x$. Is f a homomorphism b/w two binary structures?

?

Example: from addition to multiplication

- Let $f(x) = e^x$. Is f a homomorphism b/w two binary structures?
  - Yes, from the real numbers with addition $(\mathbb{R},+)$ to $f(x+y) = f(x) \cdot f(y)$
  - the positive real numbers with multiplication $(\mathbb{R}^+,\cdot)$ $f:(\mathbb{R},+) \longrightarrow (\mathbb{R}^+,\cdot)$
  - It is even an isomorphism!

The exponential map exp : $\mathbb{R} \to \mathbb{R}^+$ defined by $\exp(x) = e^x$, where e is the base of the natural logarithm, is an isomorphism from $(\mathbb{R}, +)$ to $(\mathbb{R}^+, \times)$. Exp is a bijection since it has an inverse function (namely $\log_e$) and exp preserves the group operations since $e^{x+y} = e^x e^y$. In this example both the elements and the operations are different yet the two groups are isomorphic, that is, as groups they have identical structures.

- Let $g(x) = e^{ix}$. Is g also a homomorphism?

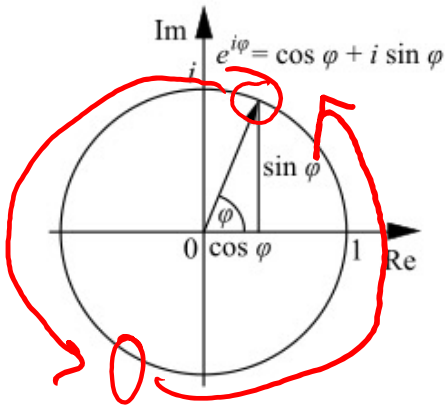
?

Example: from addition to multiplication

- Let $f(x) = e^x$. Is f a homomorphism b/w two binary structures?
  - Yes, from the real numbers with addition $(\mathbb{R},+)$ to $f(x+y) = f(x) \cdot f(y)$
  - the positive real numbers with multiplication $(\mathbb{R}^+,\cdot)$ $f:(\mathbb{R},+) \longrightarrow (\mathbb{R}^+,\cdot)$
  - It is even an isomorphism!

The exponential map exp : $\mathbb{R} \to \mathbb{R}^+$ defined by $\exp(x) = e^x$, where e is the base of the natural logarithm, is an isomorphism from $(\mathbb{R}, +)$ to $(\mathbb{R}^+, \times)$. Exp is a bijection since it has an inverse function (namely $\log_e$) and exp preserves the group operations since $e^{x+y} = e^x e^y$. In this example both the elements and the operations are different yet the two groups are isomorphic, that is, as groups they have identical structures.

- Let $g(x) = e^{ix}$. Is g also a homomorphism?
  - Yes, from the real numbers with addition $(\mathbb{R},+)$ to
  - the unit circle in the complex plane with rotation



Example: from addition to multiplication

$G = \mathbb{R}$ under $+$

$H = \{ z \in \mathbb{C} : |z| = 1 \}$

$= \text{Group under } \times$

Hint:

Every $z \in \mathbb{C}$ with $|z| = 1$ can be written as $z = e^{i\theta}$.

$f : G \longrightarrow H$

$x \longmapsto e^{ix}$

Show $f(x+y) = f(x) \times f(y)$

$$e^{i(x+y)} = e^{ix} \times e^{iy}$$

$$e^{ix+iy} = e^{ix} \times e^{iy}$$

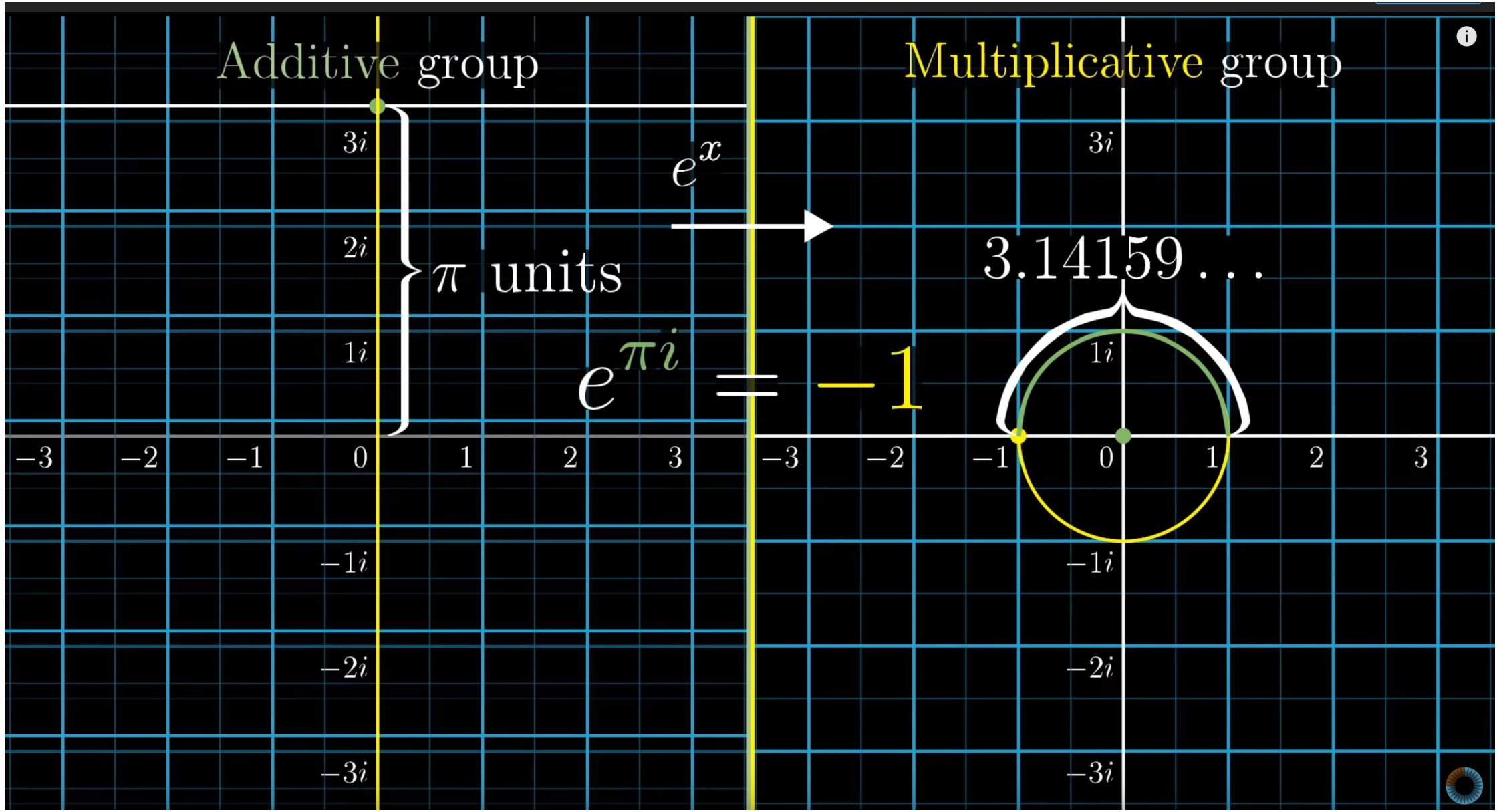
$$e^{ix} \times e^{iy} = e^{ix} \times e^{iy}$$

$f(0) = f(2\pi) = 1, \quad f(2\pi n) = 1$

f is not 1-1

Source: Socratica, Homomorphisms, 2014: https://www.youtube.com/watch?v=cYzp5IWqCsg

Example: from addition to multiplication

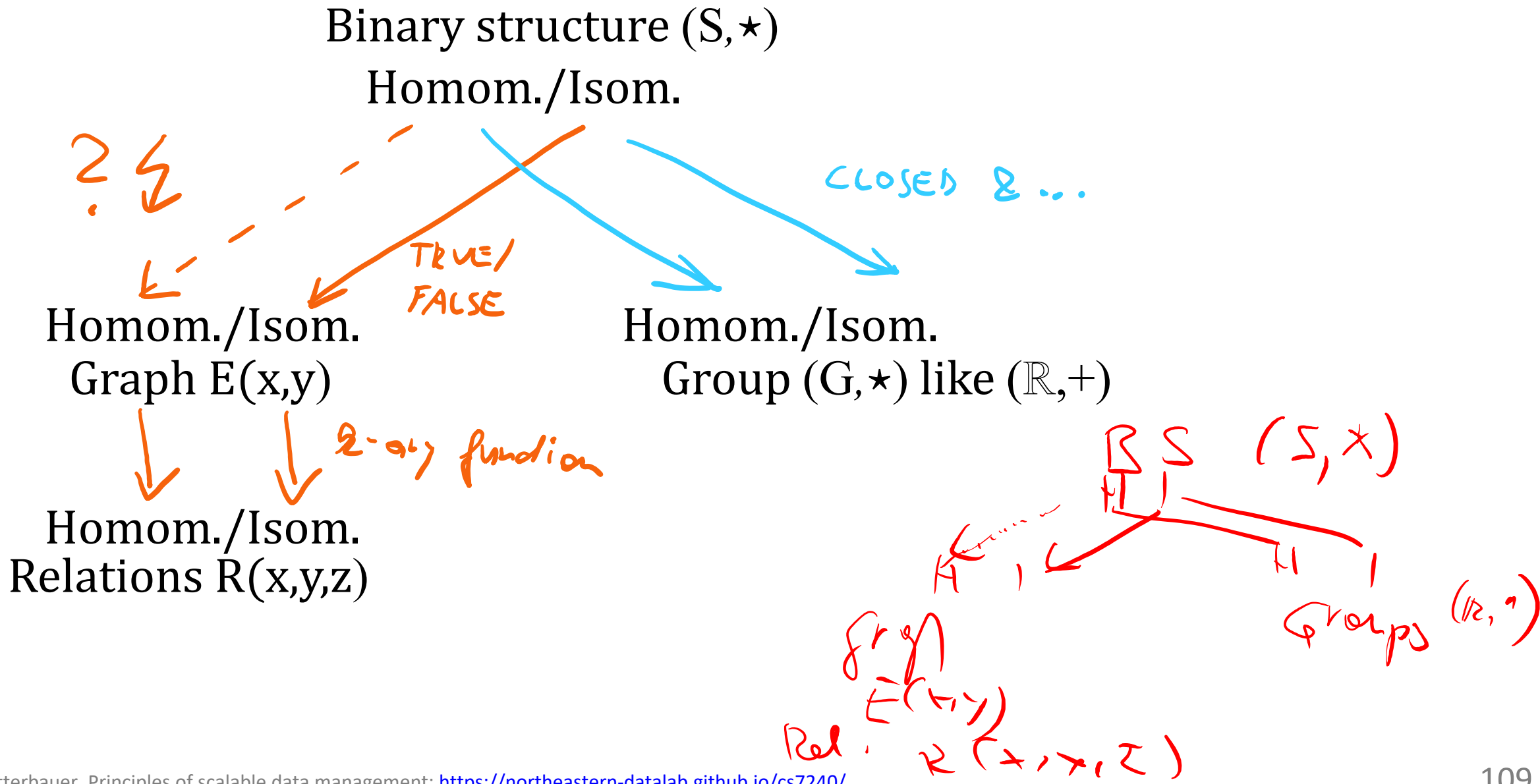


Source: 3blue1brown: Euler's formula with introductory group theory, 2017, https://www.youtube.com/watch?v=mvmuCPvRoWQ

Isomorphism

- **Definition**: A homomorphism of binary structures is called an isomorphism iff the corresponding map of sets is:
  - one-to-one (injective) and
  - onto (surjective).

Some homomorphisms



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.
- Kolaitis, Vardi. *Conjunctive-Query Containment and Constraint Satisfaction*. JCSS 2000. https://doi.org/10.1006/jcss.2000.1713
- Vardi. The Complexity of Relational Query Languages. STOC 1982. https://doi.org/10.1145/800070.802186
- Vardi. *Constraint satisfaction and database theory: a tutorial*. PODS 2000. https://doi.org/10.1145/335168.335209

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
  - Nested queries
  - Tree pattern queries

Query Equivalence

Two queries q_1, q_2 are equivalent, denoted $q_1 \equiv q_2$, if for every database instance D, we have $q_1(D) = q_2(D)$.

Query q_1 is contained in query q_2 , denoted $q_1 \subseteq q_2$, if for every database instance D, we have $q_1(D) \subseteq q_2(D)$

Corollary

$q_1 \equiv q_2$ is equivalent to ($q_1 \subseteq q_2$ and $q_1 \supseteq q_2$)

If queries are Boolean, then query containment = logical implication:

$q_1 \Leftrightarrow q_2$ is equivalent to ?

Query Equivalence

Two queries q_1, q_2 are equivalent, denoted $q_1 \equiv q_2$, if for every database instance D, we have $q_1(D) = q_2(D)$.

Query q_1 is contained in query q_2 , denoted $q_1 \subseteq q_2$, if for every database instance D, we have $q_1(D) \subseteq q_2(D)$

Corollary

$q_1 \equiv q_2$ is equivalent to ($q_1 \subseteq q_2$ and $q_1 \supseteq q_2$)

If queries are Boolean, then query containment = logical implication:

$q_1 \Leftrightarrow q_2$ is equivalent to ($q_1 \Rightarrow q_2$ and $q_1 \Leftarrow q_2$)

Homomorphisms

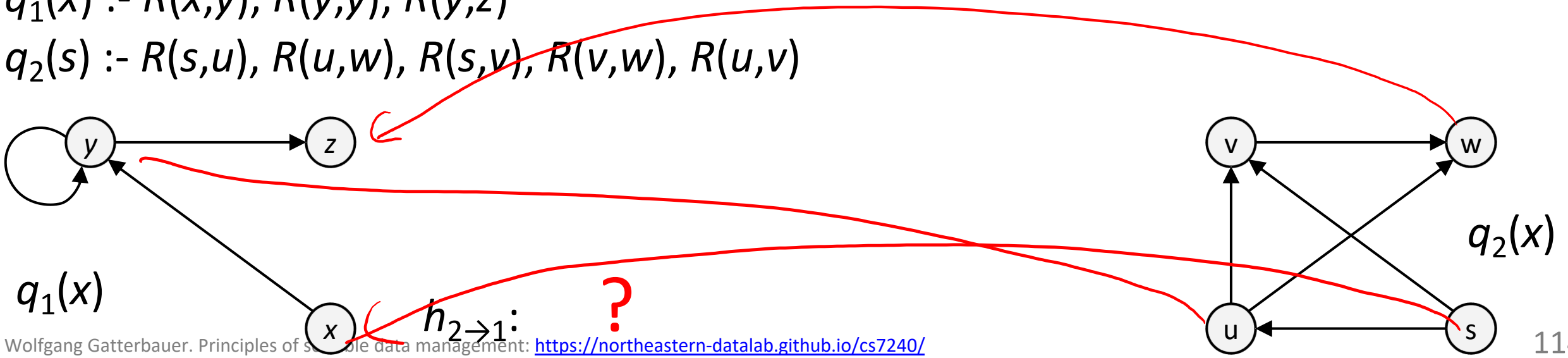
A homomorphism h from Boolean q_2 to q_1 is a function
h: var(q_2) → var(q_1) ∪ const(q_1) such that:
for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

SAME

Example

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

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



Homomorphisms

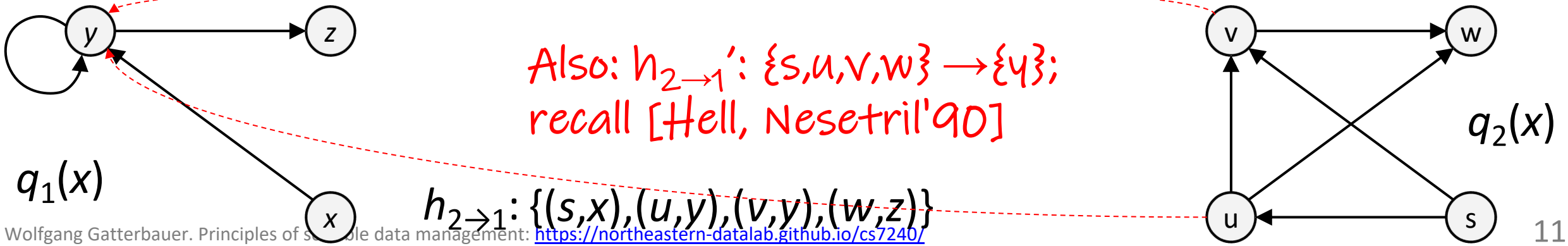
A homomorphism h from Boolean q_2 to q_1 is a function h: var(q_2) → var(q_1) ∪ const(q_1) such that:

for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

Example

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

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



Homomorphisms

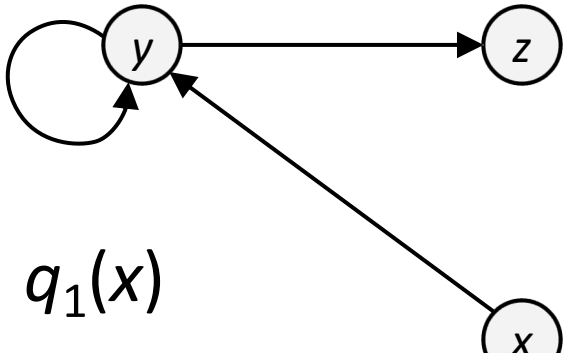
A homomorphism h from Boolean q_2 to q_1 is a function h: var(q_2) → var(q_1) ∪ const(q_1) such that:

for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

Example

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

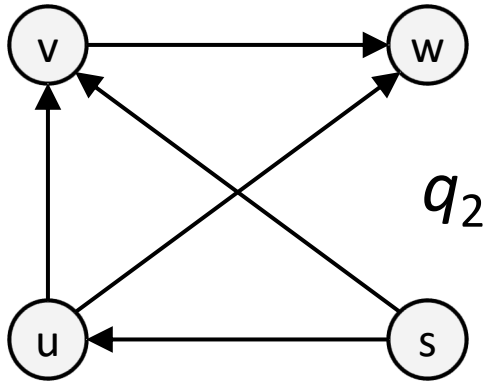
$q_2(s)$:- $R(s,u)$, $R(u,w)$, $R(s,v)$, $R(v,w)$, $R(u,v)$



$q_1(x)$

$h_{1\to2}$: ?

$h_{2\to1}$: {(s,x),(u,y),(v,y),(w,z)}



$q_2(x)$

Homomorphisms

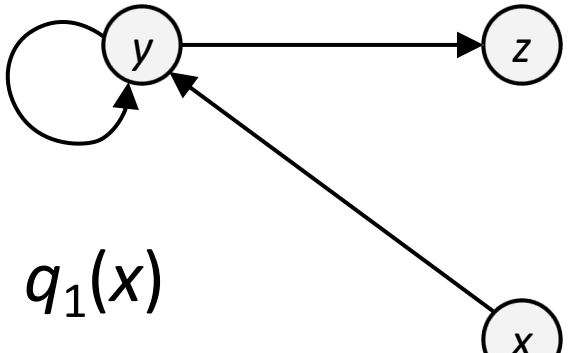
A homomorphism h from Boolean q_2 to q_1 is a function h: var(q_2) → var(q_1) ∪ const(q_1) such that:

for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

Example

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

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

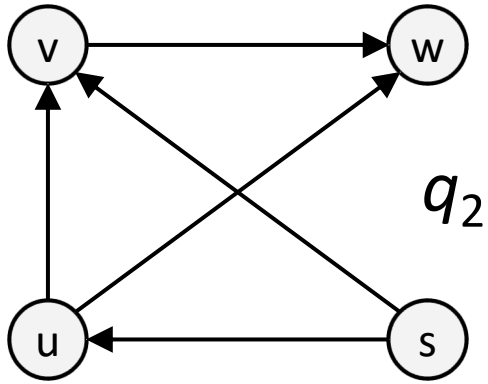


$q_1(x)$

What about:

$h_{1\to2}$: {(x,s),(y,v),(z,w)} ?

$h_{2\to1}$: {(s,x),(u,y),(v,y),(w,z)}



$q_2(x)$

Homomorphisms

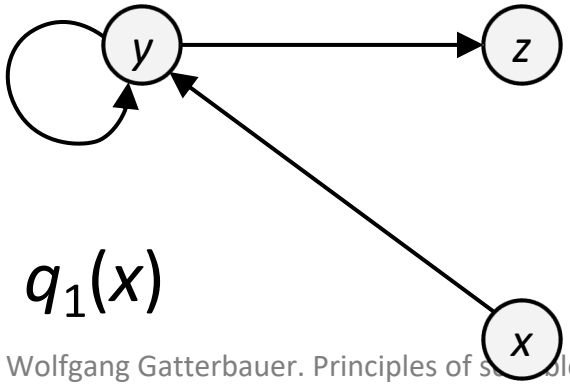
A homomorphism h from Boolean q_2 to q_1 is a function h: var(q_2) → var(q_1) ∪ const(q_1) such that:

for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

Example

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

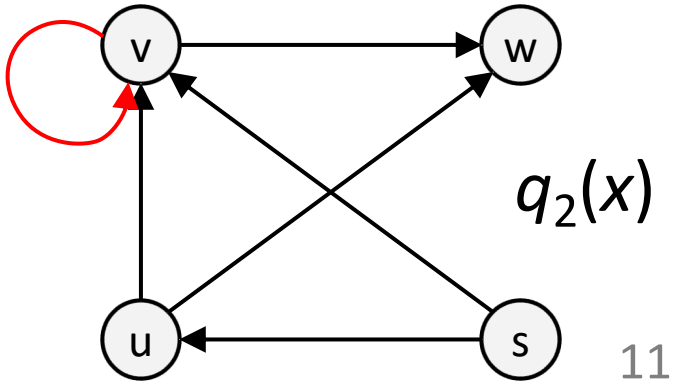
$q_2(s)$:- $R(s,u)$, $R(u,w)$, $R(s,v)$, $R(v,w)$, $R(u,v)$, $R(v,v)$



$q_1(x)$

$h_{1\to2}$: {(x,s),(y,v),(z,w)}

$h_{2\to1}$: {(s,x),(u,y),(v,y),(w,z)}



$q_2(x)$

Homomorphisms

A homomorphism h from Boolean q_2 to q_1 is a function h: var(q_2) → var(q_1) ∪ const(q_1) such that:

for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

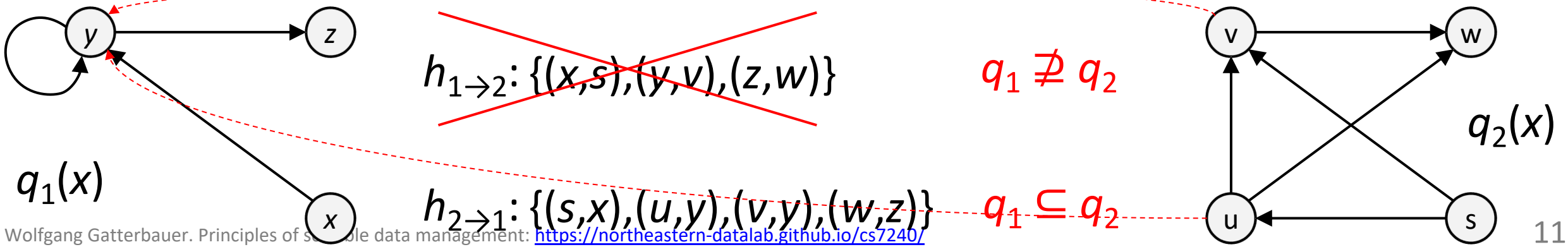
Compare to our earlier example:

$\exists x.\ E(x,x)$? $\exists x.\exists y.\ E(x,y)$

Example

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

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



Homomorphisms

A homomorphism h from Boolean q_2 to q_1 is a function h: var(q_2) → var(q_1) ∪ const(q_1) such that:

for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

E(2,2)

Compare to our earlier example:

$$\exists x.\ E(x,x) \quad \Rightarrow \quad \exists x.\exists y.\ E(x,y)$$

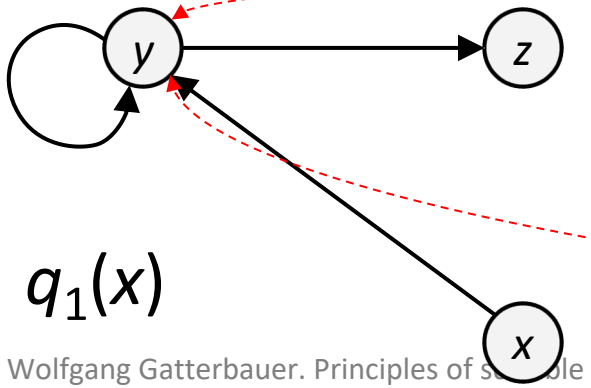
F T

E(1,2)

Example

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

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



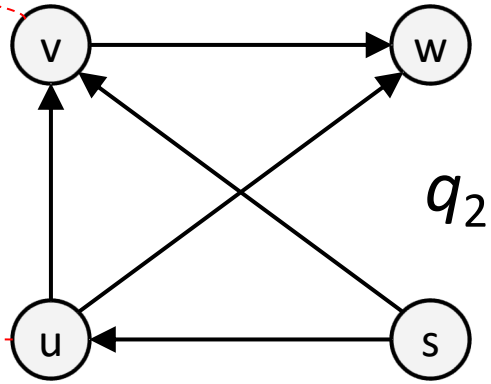
$q_1(x)$

$h_{1\to2}$: {(x,s),(y,v),(z,w)}

$h_{2\to1}$: {(s,x),(u,y),(v,y),(w,z)}

$q_1 \not\supseteq q_2$

$q_1 \subseteq q_2$



$q_2(x)$

Canonical database

Definition (Canonical database)

Given a conjunctive query q, the canonical database $D_c[q]$ is the database instance where each atom in q becomes a fact in the instance.

Example

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

$D_c[q]$ = ?

Canonical database

Definition (Canonical database)

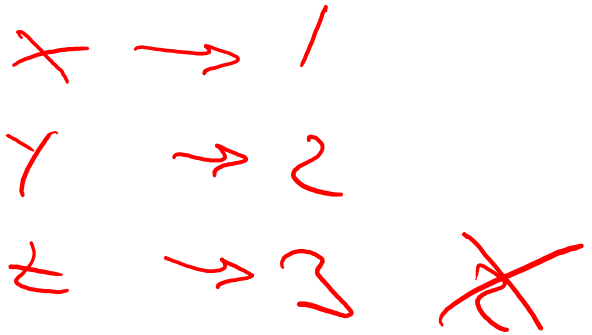
Given a conjunctive query q, the canonical database $D_c[q]$ is the database instance where each atom in q becomes a fact in the instance.

Example

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

$D_c[q]$ = {R('x','y'), R('y','y'), R('y','z')}

≡ {R(a,b), R(b,b), R(b,c)}



Just treat each variable as different constant ☺

[Chandra and Merlin 1977]

THEOREM (Query Containment)

Given two Boolean CQs q_1, q_2, the following statements are equivalent:

1) $q_1 \subseteq q_2$

2) There is a homomorphism $h_{2\to1}$ from q_2 to q_1

3) $q_2(D_C[q_1])$ is true

We will only look at 2) ⇒ 1)

[Chandra and Merlin 1977]

If there is a homomorphism h *from* q_2 *to* q_1, then $q_1 \subseteq q_2$

1. Given $h=h_{2\to1}$, we will show that for any D: $q_1(D) \Rightarrow q_2(D)$
2. For $q_1(D)$ to hold, there is a valuation v s.t. $v(q_1) \in D$
3. We will show that the composition $g = v \circ h$ is a valuation for q_2

$g=v \circ h$

$g(x)=v(h(x))$

[Chandra and Merlin 1977]

If there is a homomorphism h *from* q_2 *to* q_1, then $q_1 \subseteq q_2$

1. Given $h=h_{2\to1}$, we will show that for any D: $q_1(D) \Rightarrow q_2(D)$
2. For $q_1(D)$ to hold, there is a valuation v s.t. $v(q_1) \in D$
3. We will show that the composition $g = v \circ h$ is a valuation for q_2
   - 3a. By definition of h, for every $R(x_1,x_2,...)$ in q_2, $R(h(x_1),h(x_2),...)$ in q_1
   - 3b. By definition of v, for every $R(x_1,x_2,...)$ in q_2, $R(v(h(x_1)),v(h(x_2)),...)$ in D

$g=v \circ h$

$g(x)=v(h(x))$

[Chandra and Merlin 1977]

If there is a homomorphism h *from* q_2 *to* q_1, then $q_1 \subseteq q_2$

1. Given $h=h_{2\to1}$, we will show that for any D: $q_1(D) \Rightarrow q_2(D)$
2. For $q_1(D)$ to hold, there is a valuation v s.t. $v(q_1) \in D$
3. We will show that the composition $g = v \circ h$ is a valuation for q_2
   - 3a. By definition of h, for every $R(x_1,x_2,...)$ in q_2, $R(h(x_1),h(x_2),...)$ in q_1
   - 3b. By definition of v, for every $R(x_1,x_2,...)$ in q_2, $R(v(h(x_1)),v(h(x_2)),...)$ in D

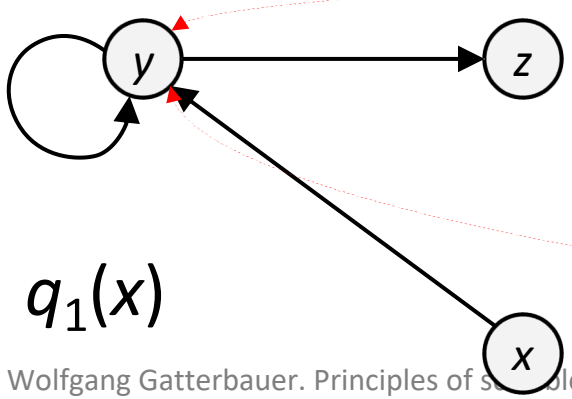
$g=v \circ h$

$g(x)=v(h(x))$

Example

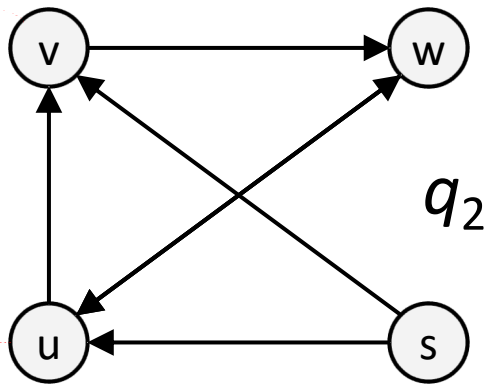
$q_1() :- R(x,y), R(y,y), R(y,z)$

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



$q_1(x)$

$h_{2\to1}$: {(s,x),(u,y),(v,y),(w,z)}



$q_2(x)$

[Chandra and Merlin 1977]

If there is a homomorphism h *from* q_2 *to* q_1, then $q_1 \subseteq q_2$

1. Given $h=h_{2\to1}$, we will show that for any D: $q_1(D) \Rightarrow q_2(D)$
2. For $q_1(D)$ to hold, there is a valuation v s.t. $v(q_1) \in D$
3. We will show that the composition $g = v \circ h$ is a valuation for q_2
   - 3a. By definition of h, for every $R(x_1,x_2,...)$ in q_2, $R(h(x_1),h(x_2),...)$ in q_1
   - 3b. By definition of v, for every $R(x_1,x_2,...)$ in q_2, $R(v(h(x_1)),v(h(x_2)),...)$ in D

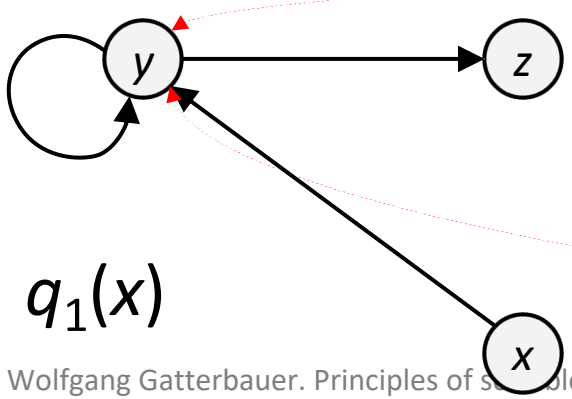
$g=v \circ h$

$g(x)=v(h(x))$

Example

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

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

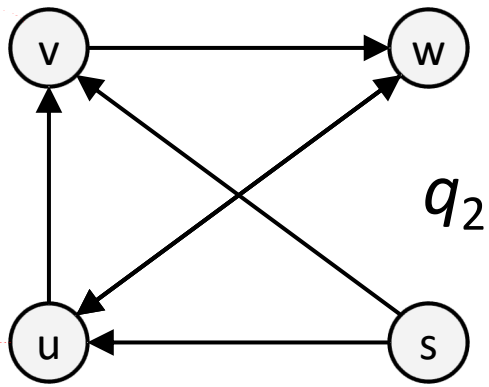


$q_1(x)$

$v=\{(x,a),(y,b),(z,c)\}$

R

A	B
a	b
b	b
b	c



$q_2(x)$

$h_{2\to1}$: {(s,x),(u,y),(v,y),(w,z)}

[Chandra and Merlin 1977]

If there is a homomorphism h *from* q_2 *to* q_1, then $q_1 \subseteq q_2$

1. Given $h=h_{2\to1}$, we will show that for any D: $q_1(D) \Rightarrow q_2(D)$
2. For $q_1(D)$ to hold, there is a valuation v s.t. $v(q_1) \in D$
3. We will show that the composition $g = v \circ h$ is a valuation for q_2
   - 3a. By definition of h, for every $R(x_1,x_2,...)$ in q_2, $R(h(x_1),h(x_2),...)$ in q_1
   - 3b. By definition of v, for every $R(x_1,x_2,...)$ in q_2, $R(v(h(x_1)),v(h(x_2)),...)$ in D

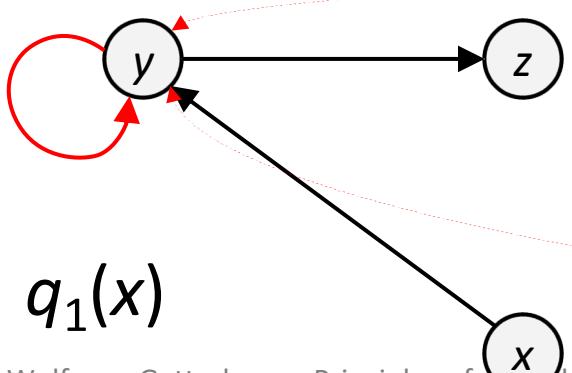
$g=v \circ h$

$g(x)=v(h(x))$

Example

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

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



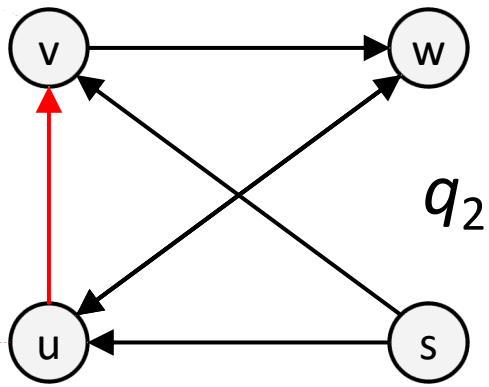
$q_1(x)$

$v=\{(x,a),(y,b),(z,c)\}$

$g=\{(s,a),(u,b),(v,b),(w,c)\}$

$h_{2\to1}$: $\{(s,x),(u,y),(v,y),(w,z)\}$

R	A	B
	a	b
	b	b
	b	c



$q_2(x)$

Combined complexity of CQC and CQE

Corollary:

The following problems are NP-complete:

1) Given two (Boolean) conjunctive queries Q and Q', is $Q \subseteq Q'$?

2) Given a Boolean conjunctive query Q and an instance D, does $D \models Q$?

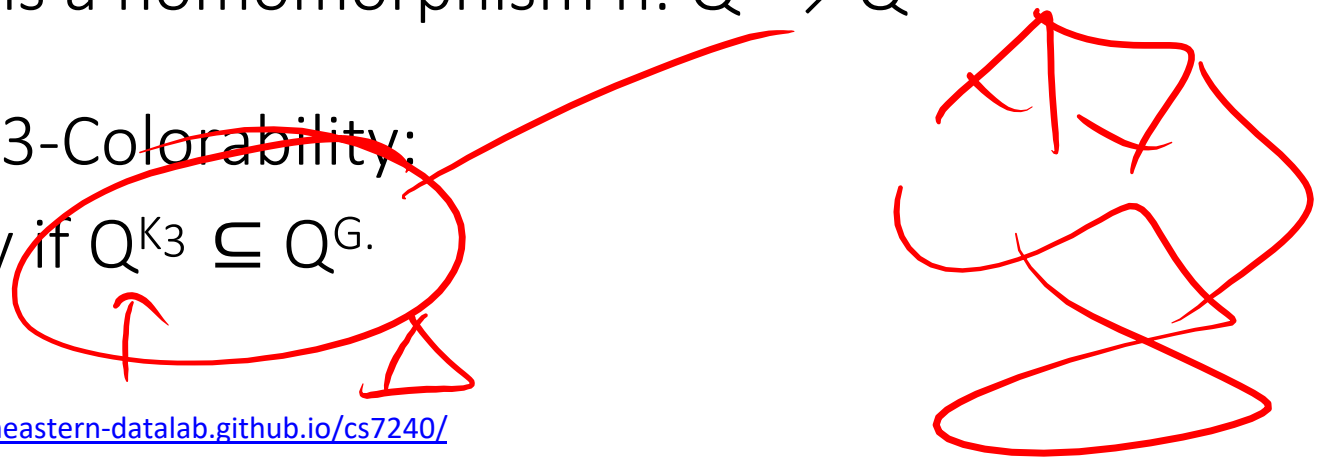
Proof:

(a) Membership in NP follows from the Homom. Theorem:

$Q \subseteq Q'$ if and only if there is a homomorphism $h: Q' \rightarrow Q$

(b) NP-hardness follows from 3-Colorability:

G is 3-colorable if and only if $Q^{K3} \subseteq Q^{G}$.



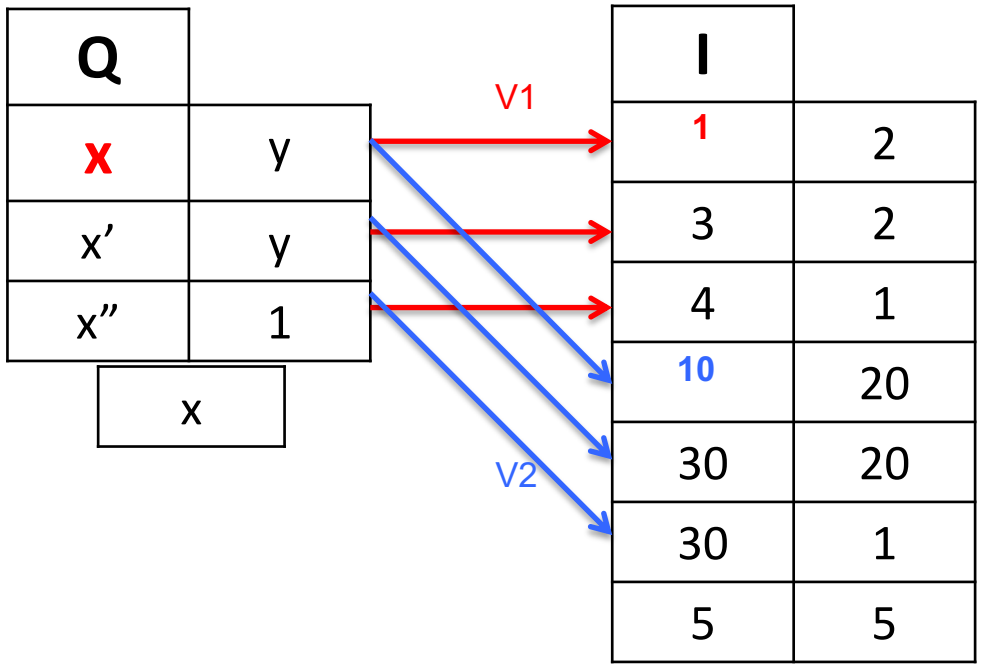
The Complexity of Database Query Languages

	Relational Calculus	CQs
Query Eval.: Data Complexity	In LOGSPACE (hence, in P)	In LOGSPACE (hence, in P)
Query Eval.: Combined Compl.	PSPACE-complete	NP-complete
Query Equivalence & Containment	Undecidable	NP-complete

Source: Phokion Kolaitis

Containment of conjunctive queries

$$Q = \{ x \mid \exists x', x'', y \; (R(xy) \wedge R(x'y) \wedge R(x'',1)) \}$$

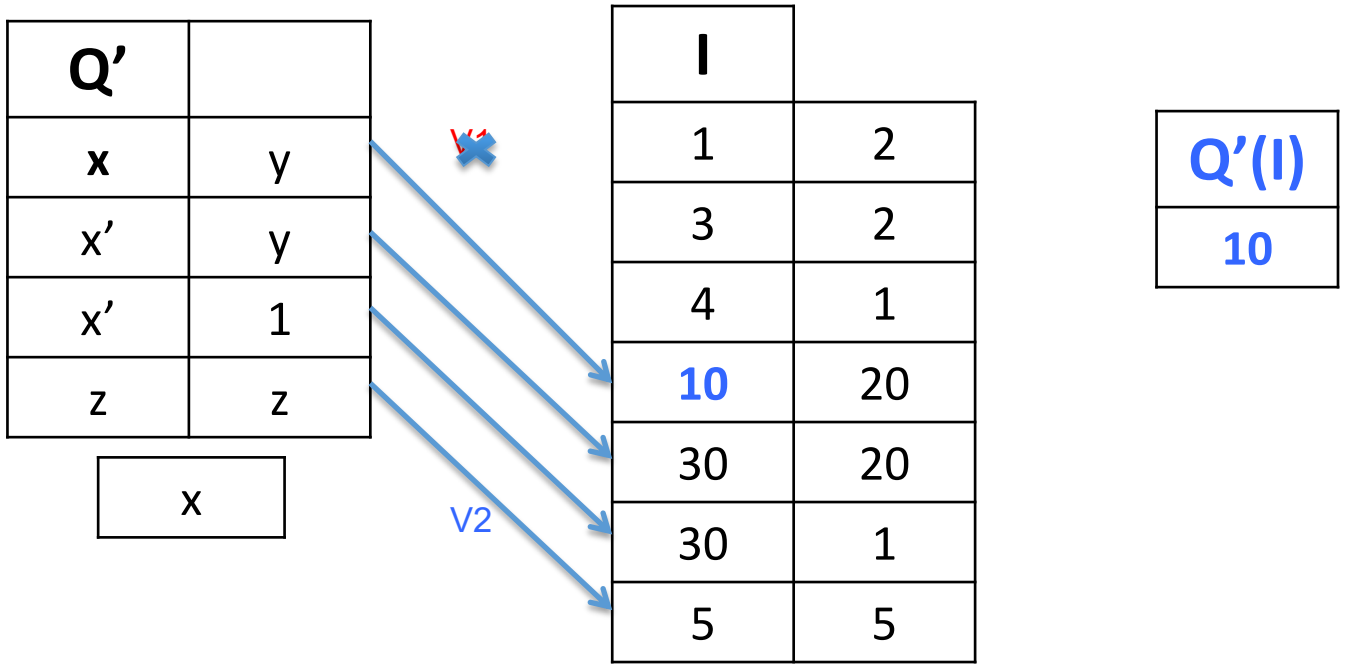


Q(I)
1
10

Another query

$\exists x', x'', y \; (R(xy) \wedge R(x'y) \wedge R(x'',1) \wedge x' = x'' \wedge R(z,z))$

$Q' = \exists x', y, z \; (R(xy) \wedge R(x'y) \wedge R(x',1) \wedge R(z,z))$



Question

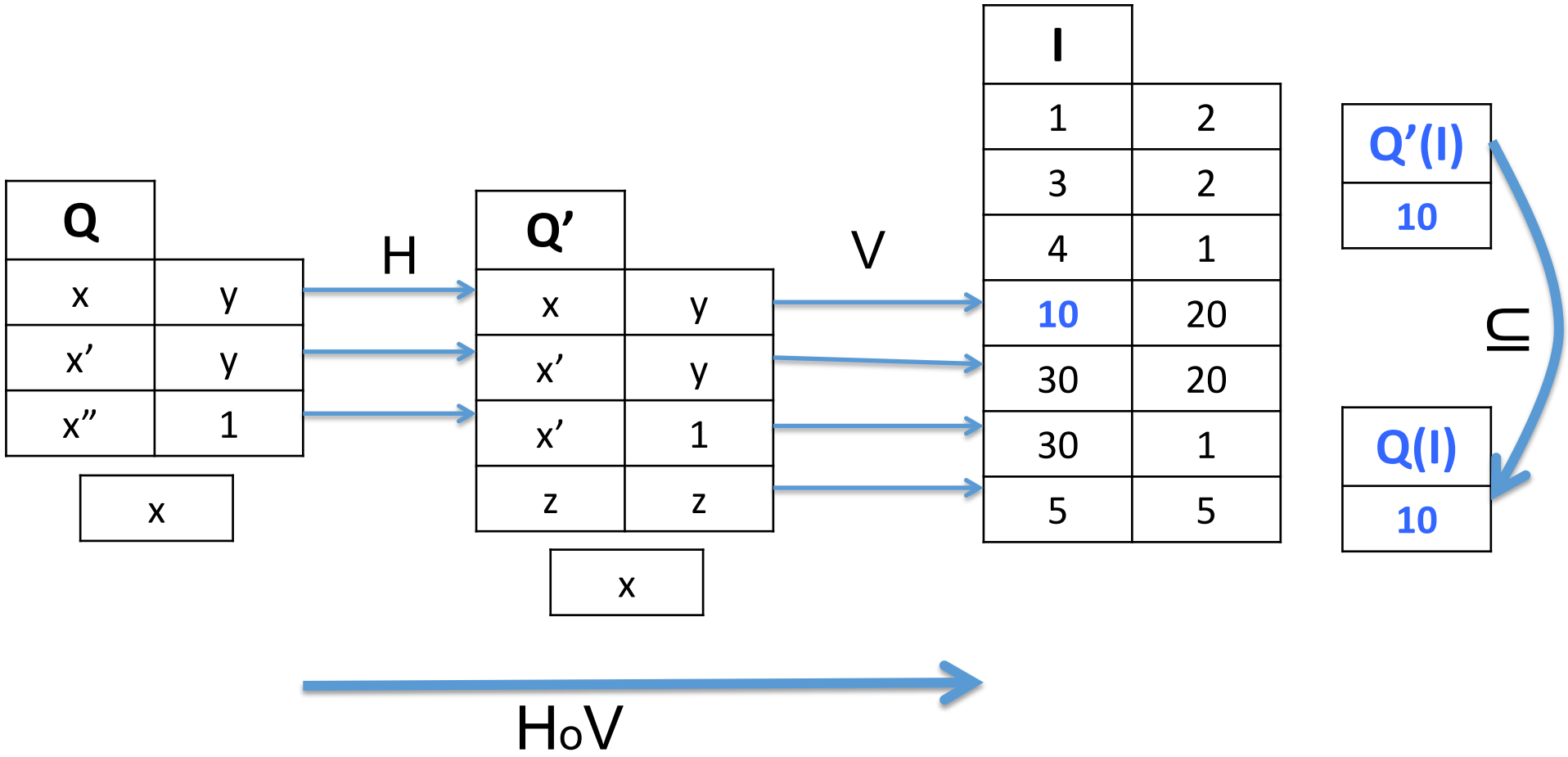
Definition: $Q' \subseteq Q$ if for all I, $Q'(I) \subseteq Q(I)$

$Q' \equiv Q$ if $Q' \subseteq Q$ and $Q \subseteq Q'$

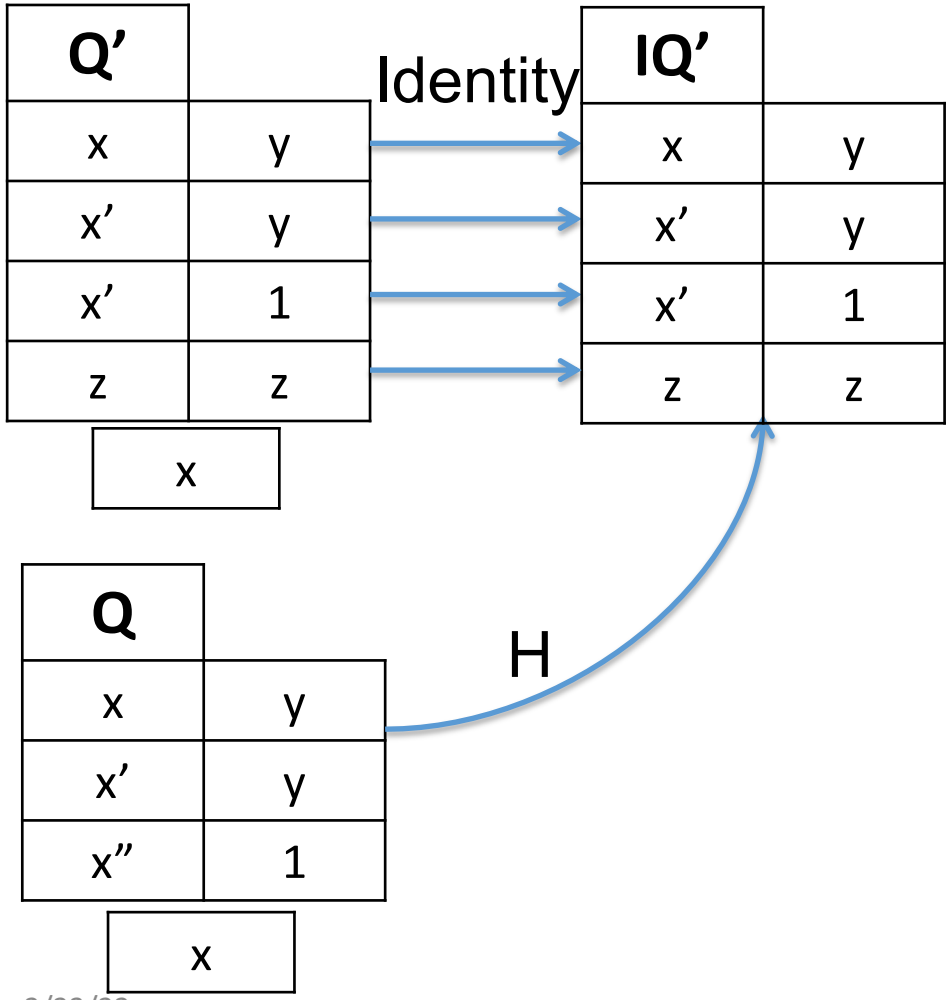
Problem: given Q', Q, test whether $Q' \subseteq Q$

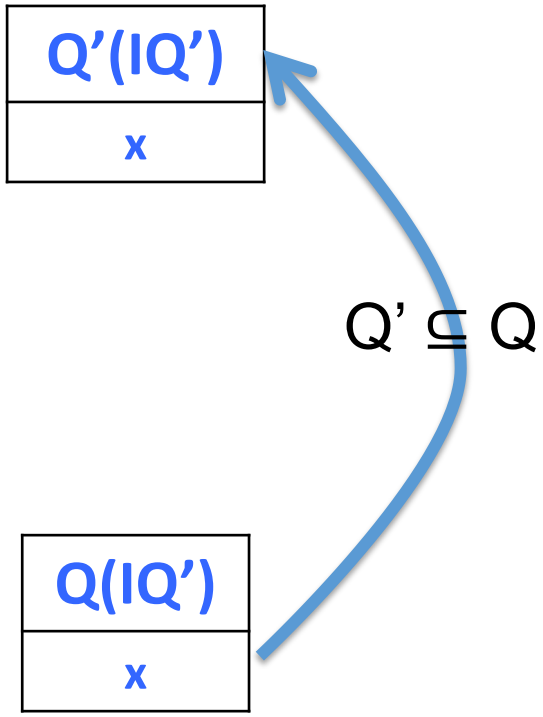
Central issue for query optimization

If there is a homomorphism from Q to Q', Q' ⊆ Q



If Q' ⊆ Q, there is a homomorphism from Q to Q'





Q’ ⊆ Q iff there is a homomorphism from Q to Q’

The problem is NP-complete

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
  - Nested queries
  - Tree pattern queries

Islands of Tractability of CQ Evaluation

- Major Research Program: Identify tractable cases of the combined complexity of conjunctive query evaluation.
- Note: Over the years, this program has been pursued by two different research communities:
  - The Database Theory community
  - The Constraint Satisfaction community
- Explanation:

  Constraint Satisfaction Problem

  ≡ (Feder-Vardi, 1993)

  Homomorphism Problem

  ≡ (Chandra-Merlin, 1977)

  Conjunctive Query Evaluation

Beyond Conjunctive Queries

- What can we say about query languages of intermediate expressive power between conjunctive queries and the full relational calculus?

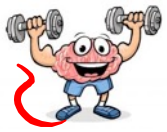
- Conjunctive queries form the sublanguage of relational algebra obtained by using only cartesian product, projection, and selection with equality conditions.

- The next step would be to consider relational algebra expressions that also involve union.

Beyond Conjunctive Queries

- Definition:
  - A union of conjunctive queries (UCQ) is a query expressible by an expression of the form $q_1 \cup q_2 \cup \ldots \cup q_m$, where each q_i is a conjunctive query.
  - A monotone query is a query expressible by a relational algebra expression which uses only union, cartesian product, projection, and selection with equality condition.
- Fact:
  - Every union of conjunctive queries is a monotone query.
  - Every monotone query is equivalent to a union of conjunctive queries, but
    - the union may have exponentially many disjuncts.
- (normal form for monotone queries).
  - Monotone queries are precisely the queries expressible by relational calculus expressions using $\wedge$, $\vee$, and $\exists$ only.

Unions of CQs and Monotone Queries



Union of Conjunctive Queries (UCQ)

Given edge relation $E(A,B)$, find paths of length 1 or 2

RA ?

RC ?

Unions of CQs and Monotone Queries

Union of Conjunctive Queries (UCQ)

Given edge relation $E(A,B)$, find paths of length 1 or 2

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

RC ?

Unions of CQs and Monotone Queries

Union of Conjunctive Queries (UCQ)

Given edge relation $E(A,B)$, find paths of length 1 or 2

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

RC $E(x_1, x_2) \vee \exists z[E(z, x_2) \wedge E(z, x_2)]$

Unions of CQs and Monotone Queries

Union of Conjunctive Queries (UCQ)

Given edge relation $E(A,B)$, find paths of length 1 or 2

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

RC $E(x_1, x_2) \vee \exists z[E(z, x_2) \wedge E(z, x_2)]$

Monotone Query

Assume schema R(A,B), S(A,B), T(B,C), V(B,C)

Is following query monotone ? $(R \cup S) \bowtie (T \cup V)$

Unions of CQs and Monotone Queries

Union of Conjunctive Queries (UCQ)

Given edge relation $E(A,B)$, find paths of length 1 or 2

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

RC $E(x_1, x_2) \vee \exists z[E(z, x_2) \wedge E(z, x_2)]$

Monotone Query

Assume schema R(A,B), S(A,B), T(B,C), V(B,C)

Is following query monotone? $(R \cup S) \bowtie (T \cup V)$

Equal to a UCQ? ?

Unions of CQs and Monotone Queries

Union of Conjunctive Queries (UCQ)

Given edge relation *E*(*A*,*B*), find paths of length 1 or 2

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

RC $E(x_1, x_2) \vee \exists z[E(z, x_2) \wedge E(z, x_2)]$

Monotone Query

Assume schema R(A,B), S(A,B), T(B,C), V(B,C)

Is following query monotone? $(R \cup S) \bowtie (T \cup V)$

Equal to a UCQ? $(R \bowtie T) \cup (R \bowtie V) \cup (S \bowtie T) \cup (S \bowtie V)$

The Containment Problem for Unions of CQs

THEOREM [Sagiv and Yannakakis 1981]
Let $q_1 \cup q_2 \cup \cdots \cup q_m$ *and* $q'_1 \cup q'_2 \cup \cdots \cup q'_n$ *be two UCQs. Then the following are equivalent:*

1) $q_1 \cup q_2 \cup \cdots \cup q_m \subseteq q'_1 \cup q'_2 \cup \cdots \cup q'_n$
2) For every $i \leq m$, there is $j \leq n$ such that $q_i \subseteq q'_j$

Proof: Use the Homomorphism Theorem

1. ⇒ 2. Since $D_C[q_i] \models q_i$, we have that $D_C[q_i] \models q_1 \cup q_2 \cup \ldots \cup q_m$ hence $D_C[q_i] \models q'_1 \cup q'_2 \cup \ldots \cup q'_n$, hence there is some $j \leq n$ such that $D_C[q_i] \models q'_j$, hence (by the Homomorphism Theorem) $q_i \subseteq q'_j$.

2. ⇒ 1. This direction is obvious.

The Complexity of Database Query Languages

	Relational Calculus	CQs	UCQs
Query Eval.: Data Complexity	In LOGSPACE (hence, in P)	In LOGSPACE (hence, in P)	In LOGSPACE (hence, in P)
Query Eval.: Combined Compl.	PSPACE-complete	NP-complete	NP-complete
Query Equivalence & Containment	Undecidable	NP-complete	NP-complete

Source: Phokion Kolaitis

Monotone Queries

- Even though monotone queries have the same expressive power as unions of conjunctive queries, the containment problem for monotone queries has higher complexity than the containment problem for unions of conjunctive queries (syntax/complexity tradeoff)

- Theorem: Sagiv and Yannakakis – 1982

  The containment problem for monotone queries is Π_2^p-complete.

- Note: The prototypical Π_2^p-complete problem is $\forall\exists$-SAT, i.e., the restriction of QBF to formulas of the form

$$\forall x_1 \ldots \forall x_m \exists y_1 \ldots \exists y_n \, \phi.$$

Source: Phokion Kolaitis

The Complexity of Database Query Languages

	Relational Calculus	CQs	UCQs	Monotone queries
Query Eval.: Data Complexity	In LOGSPACE (hence, in P)	In LOGSPACE (hence, in P)	In LOGSPACE (hence, in P)	In LOGSPACE (hence, in P)
Query Eval.: Combined Compl.	PSPACE-complete	NP-complete	NP-complete	NP-complete
Query Equivalence & Containment	Undecidable	NP-complete	NP-complete	Π_2^p-complete

Source: Phokion Kolaitis

Conjunctive Queries with Inequalities

- Definition: Conjunctive queries with inequalities form the sublanguage of relational algebra obtained by using only cartesian product, projection, and selection with equality and inequality (≠, <, $\leq$) conditions.

- Example: Q(x,y):-- E(x,z), E(z,w),E(w,y), z ≠ w, z < y.

- Theorem: (Klug – 1988, van der Meyden – 1992)
  - The query containment problem for conjunctive queries with inequalities is Π_2^p-complete.
  - The query evaluation problem for conjunctive queries with inequalities in NP-complete.

Source: Phokion Kolaitis

The Complexity of Database Query Languages

	Relational Calculus	CQs	UCQs	Monotone queries / CQs with inequalities
Query Eval.: Data Complexity	In LOGSPACE (hence, in P)	In LOGSPACE (hence, in P)	In LOGSPACE (hence, in P)	In LOGSPACE (hence, in P)
Query Eval.: Combined Compl.	PSPACE-complete	NP-complete	NP-complete	NP-complete
Query Equivalence & Containment	Undecidable	NP-complete	NP-complete	Π_2^p-complete

Source: Phokion Kolaitis

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
  - Nested queries
  - Tree pattern queries

Following slides are from Phokion Kolaitis's talk on "Logic and databases" at "Logical structures in Computation Boot Camp", Berkeley 2016:
https://simons.berkeley.edu/talks/logic-and-databases

Logic and Databases

Phokion G. Kolaitis

UC Santa Cruz & IBM Research – Almaden

Lecture 4 – Part 1

Credit: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Thematic Roadmap

- ✓ Logic and Database Query Languages
  - Relational Algebra and Relational Calculus
  - Conjunctive queries and their variants
  - Datalog
- ✓ Query Evaluation, Query Containment, Query Equivalence
  - Decidability and Complexity
- ✓ Other Aspects of Conjunctive Query Evaluation
- • Alternative Semantics of Queries
  - Bag Databases: Semantics and Conjunctive Query Containment
  - Probabilistic Databases: Semantics and Dichotomy Theorems for Conjunctive Query Evaluation
  - Inconsistent Databases: Semantics and Dichotomy Theorems

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Alternative Semantics

- So far, we have examined logic and databases under classical semantics:
  - The database relations are sets.
  - Tarskian semantics are used to interpret queries definable be first-order formulas.
- Over the years, several different alternative semantics of queries have been investigated. We will discuss three such scenarios:
  - The database relations can be bags (multisets).
  - The databases may be probabilistic.
  - The databases may be inconsistent.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Sets vs. Multisets

Relation EMPLOYEE(name, dept, salary)

- Relational Algebra Expression:

  $\pi_{\text{salary}}\ (\sigma_{\text{dept = CS}}\ (\text{EMPLOYEE}))$

- SQL query:

```
SELECT   salary
FROM     EMPLOYEE
WHERE    dpt = 'CS'
```

- SQL returns a bag (multiset) of numbers in which a number may appear several times, provided different faculty had the same salary.
- SQL does not eliminate duplicates, in general, because:
  - Duplicates are important for aggregate queries (e.g., average)
  - Duplicate elimination takes nlogn time.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Relational Algebra Under Bag Semantics

Operation	**Multiplicity**
Union $R_1 \cup R_2$	$m_1 + m_2$
Intersection $R_1 \cap R_2$	$\min(m_1, m_2)$
Product $R_1 \times R_2$	$m_1 \times m_2$
Projection and Selection	Duplicates are not eliminated

- R_1

A	B
1	2
1	2
2	3

- R_2

B	C
2	4
2	5

- $(R_1 \bowtie R_2)$

A	B	C
1	2	4
1	2	4
1	2	5
1	2	5

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Conjunctive Queries Under Bag Semantics

Chaudhuri & Vardi – 1993

Optimization of ***Real*** Conjunctive Queries

- Called for a re-examination of conjunctive-query optimization under bag semantics.
- In particular, they initiated the study of the

  containment problem for conjunctive queries

  under bag semantics.
- This problem has turned out to be *much more challenging* than originally perceived.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

PROBLEMS

Problems worthy
of attack
prove their worth
by hitting back.

in: *Grooks* by Piet Hein (1905-1996)

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Query Containment Under Set Semantics

Class of Queries	**Complexity of Query Containment**
Conjunctive Queries	NP-complete Chandra & Merlin – 1977
Unions of Conjunctive Queries	NP-complete Sagiv & Yannakakis - 1980
Conjunctive Queries with $\neq$, $\leq$, $\geq$	Π_2^p-complete Klug 1988, van der Meyden -1992
First-Order (SQL) queries	Undecidable Trakhtenbrot - 1949

8

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Bag Semantics vs. Set Semantics

- For bags R_1, R_2:
  $R_1 \subseteq_{BAG} R_2$ if $m(\mathbf{a},R_1) \leq m(\mathbf{a},R_2)$, for every tuple $\mathbf{a}$.
- $Q^{BAG}(D)$: Result of evaluating Q on (bag) database D.
- $Q_1 \subseteq_{BAG} Q_2$ if for every (bag) database D, we have that

$$Q_1{}^{BAG}(D) \subseteq_{BAG} Q_2{}^{BAG}(D).$$

Fact:

- $Q_1 \subseteq_{BAG} Q_2$ implies $Q_1 \subseteq Q_2$.
- The converse does **not** always hold.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Bag Semantics vs. Set Semantics

Fact: $Q_1 \subseteq Q_2$ does not imply that $Q_1 \subseteq_{BAG} Q_2$.

Example:

- $Q_1(x)$:- P(x), T(x)
- $Q_2(x)$:- P(x)

- $Q_1 \subseteq Q_2$ (obvious from the definitions)
- $Q_1 \not\subseteq_{BAG} Q_2$
- Consider the (bag) instance D = {P(a), T(a), T(a)}. Then:
  - $Q_1(D)$ = {a,a}
  - $Q_2(D)$ = {a}, so $Q_1(D) \not\subseteq Q_2(D)$.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Query Containment under Bag Semantics

- Chaudhuri & Vardi - 1993 stated that:

  Under bag semantics, the containment problem for conjunctive queries is Π_2^p-hard.

- Problem:
  - What is the exact complexity of the containment problem for conjunctive queries under bag semantics?
  - Is this problem decidable?

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Query Containment Under Bag Semantics

- 23 years have passed since the containment problem for conjunctive queries under bag semantics was raised.
- Several attacks to solve this problem have failed.
- At least two technically flawed PhD theses on this problem have been produced.
- Chaudhuri and Vardi have withdrawn the claimed Π_2^p-hardness of this problem; no one has provided a proof.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Query Containment Under Bag Semantics

- The containment problem for conjunctive queries under bag semantics remains **open** to date.

- However, progress has been made towards the containment problem under bag semantics for the two main extensions of conjunctive queries:
  - Unions of conjunctive queries
  - Conjunctive queries with ≠

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Unions of Conjunctive Queries

Theorem (Ioannidis & Ramakrishnan – 1995):
Under bag semantics, the containment problem for unions of conjunctive queries is **undecidable**.

Hint of Proof:
Reduction from Hilbert's 10th Problem.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Hilbert's 10th Problem

- Hilbert's 10th Problem – 1900
  (10th in Hilbert's list of 23 problems)

Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.

In effect, Hilbert's 10th Problem is:
Find an algorithm for the following problem:
Given a polynomial $P(x_1,...,x_n)$ with integer coefficients, does it have an all-integer solution?

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Hilbert's 10th Problem

- Hilbert's 10th Problem – 1900

  (10th in Hilbert's list of 23 problems)

  Find an algorithm for the following problem:

  Given a polynomial $P(x_1,...,x_n)$ with integer coefficients, does it have an all-integer solution?

- Y. Matiyasevich – 1971

  (building on M. Davis, H. Putnam, and J. Robinson)

  – Hilbert's 10th Problem is **undecidable**, hence **no** such algorithm exists.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Hilbert's 10th Problem

- Fact: The following variant of Hilbert's 10th Problem is undecidable:
  - Given two polynomials $p_1(x_1,\ldots x_n)$ and $p_2(x_1,\ldots x_n)$ with positive integer coefficients and no constant terms, is it true that $p_1 \leq p_2$?

    In other words, is it true that $p_1(a_1,\ldots,a_n) \leq p_2(a_1,\ldots a_n)$, for all positive integers $a_1,\ldots,a_n$?

- Thus, there is no algorithm for deciding questions like:
  - Is $3x_1^4x_2x_3 + 2x_2x_3 \leq x_1^6 + 5x_2x_3$?

17

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Unions of Conjunctive Queries

Theorem (Ioannidis & Ramakrishnan – 1995):

Under bag semantics, the containment problem for unions of conjunctive queries is **undecidable**.

Hint of Proof:

- Reduction from the previous variant of Hilbert's 10th Problem:
  - Use joins of unary relations to encode monomials (products of variables).
  - Use unions to encode sums of monomials.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Unions of Conjunctive Queries

Example: Consider the polynomial $3x_1^4x_2x_3 + 2x_2x_3$

- The monomial $x_1^4x_2x_3$ is encoded by the conjunctive query $P_1(w),P_1(w),P_1(w), P_1(w), P_2(w),P_3(w)$.

- The monomial x_2x_3 is encoded by the conjunctive query $P_2(w),P_3(w)$.

- The polynomial $3x_1^4x_2x_3 + 2x_2x_3$ is encoded by the union having:
  - three copies of $P_1(w),P_1(w),P_1(w), P_1(w), P_2(w),P_3(w)$ and
  - two copies of $P_2(w),P_3(w)$.

19

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Complexity of Query Containment

Class of Queries	Complexity – Set Semantics	Complexity – Bag Semantics
Conjunctive queries	NP-complete CM – 1977	
Unions of conj. queries	NP-complete SY - 1980	Undecidable IR - 1995
Conj. queries with $\neq$, $\leq$, $\geq$	Π_2^p-complete vdM - 1992	
First-order (SQL) queries	Undecidable Trakhtenbrot - 1949	Undecidable

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Conjunctive Queries with ≠

Theorem (Jayram, K …, Vee – 2006):

Under bag semantics, the containment problem for conjunctive queries with ≠ is **undecidable**.

In fact, this problem is **undecidable** even if

- the queries use only a single relation of arity 2;
- the number of inequalities in the queries is at most some fixed (albeit huge) constant.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Complexity of Query Containment

Class of Queries	Complexity – Set Semantics	Complexity – Bag Semantics
Conjunctive queries	NP-complete CM – 1977	**Open**
Unions of conj. queries	NP-complete SY - 1980	Undecidable IR - 1995
Conj. queries with $\neq$, $\leq$, $\geq$	Π_2^p-complete vdM - 1992	Undecidable JKV - 2006
First-order (SQL) queries	Undecidable Trakhtenbrot - 1949	Undecidable

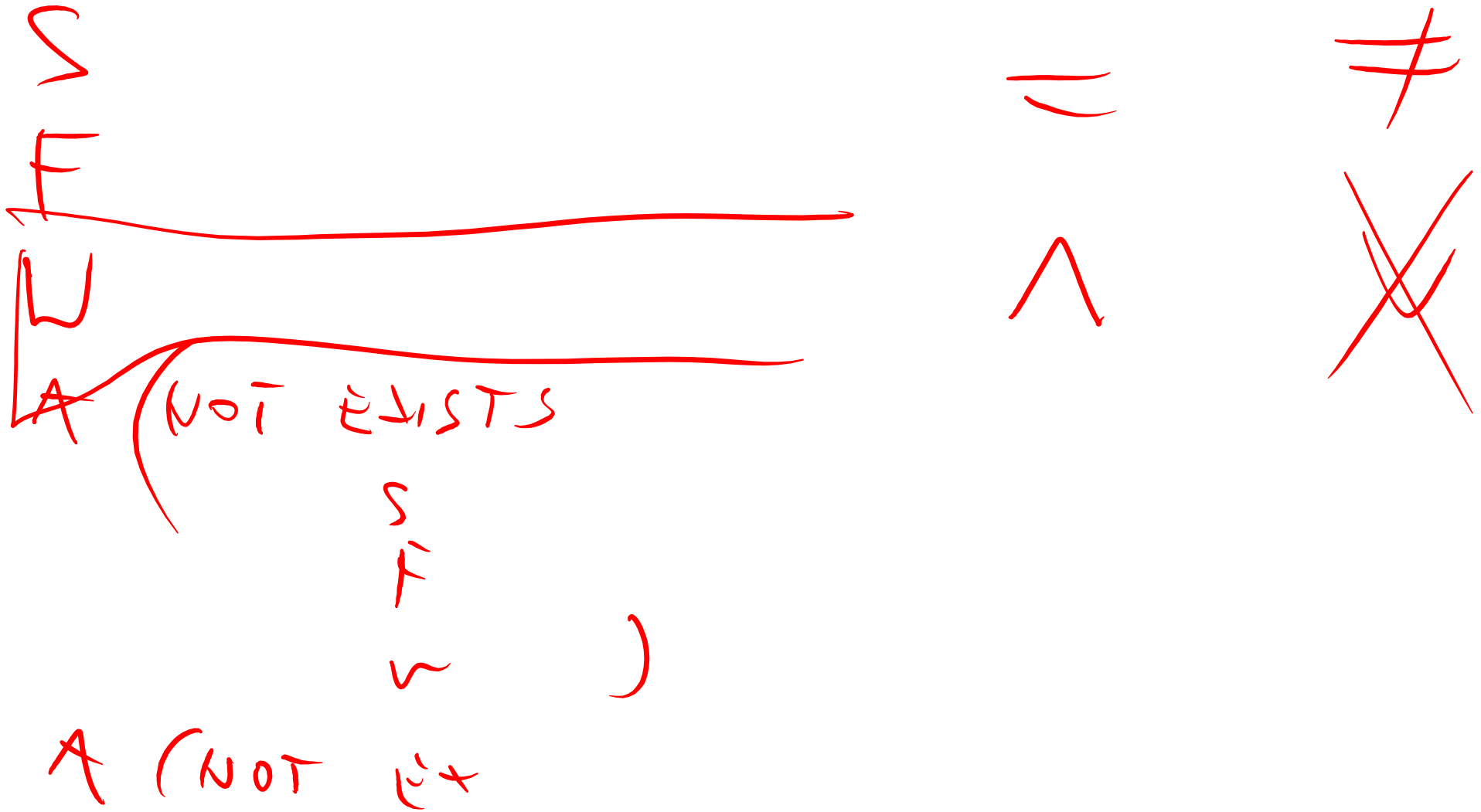
26

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01

Subsequent Developments

- Some progress has been made towards identifying special classes of conjunctive queries for which the containment problem under bag semantics is decidable.
  - Afrati, Damigos, Gergatsoulis – 2010
    - Projection-free conjunctive queries.
  - Kopparty and Rossman – 2011
    - A large class of boolean conjunctive queries on graphs.

Source: Phokion Kolaitis: https://simons.berkeley.edu/talks/phokion-kolaitis-2016-09-01


S
F
U
A (NOT EXISTS
S
F
~)
A (NOT it
=
≠
∧

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

Updated 3/5/2021

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

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp21)

https://northeastern-datalab.github.io/cs7240/sp21/

3/5/2021

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
  - Nested queries
  - Tree pattern queries

Exercise: Find the Homomorphisms

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

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

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

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

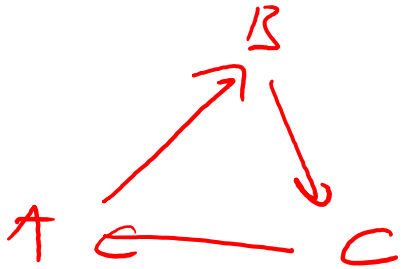
What is the containment relation between these queries ?

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

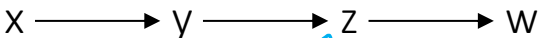
q_5: {E(x,x)}

Example by Andreas Pieris

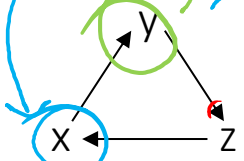
Exercise: Find the Homomorphisms



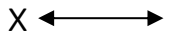
q_1: {E(x,y),E(y,z),E(z,w)}



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

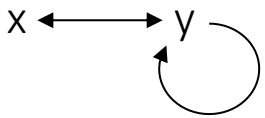


q_3: {E(x,y),E(y,x)}

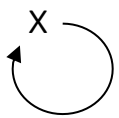


What is the containment relation between these queries ?

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

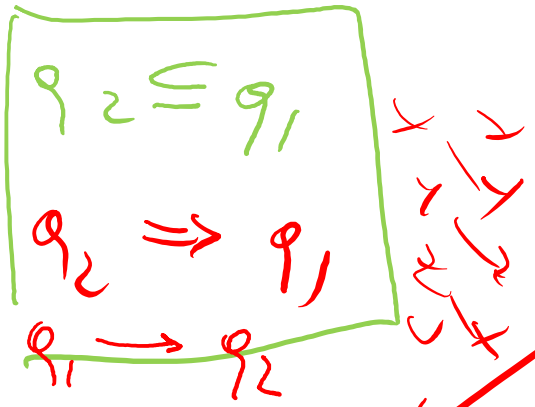


q_5: {E(x,x)}

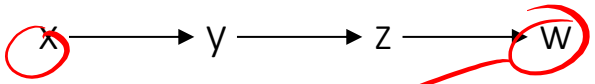


Example by Andreas Pieris

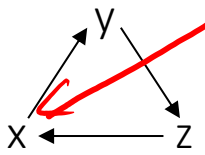
Exercise: Find the Homomorphisms



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



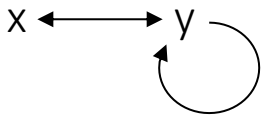
q_2: {E(x,y),E(y,z),E(z,x)}



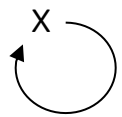
q_3: {E(x,y),E(y,x)}

What is the containment relation between these queries ?

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



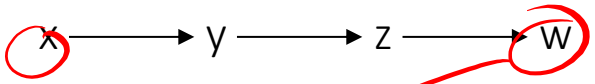
q_5: {E(x,x)}

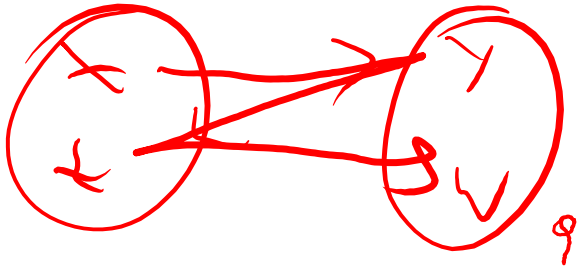


Example by Andreas Pieris

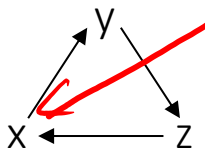
Exercise: Find the Homomorphisms

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

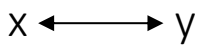


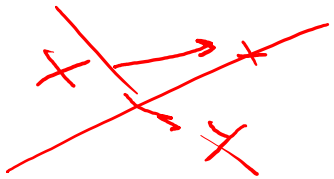


q_2: {E(x,y),E(y,z),E(z,x)}



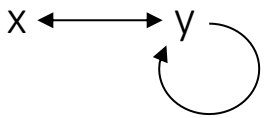
q_3: {E(x,y),E(y,x)}



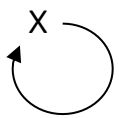


What is the containment relation between these queries ?

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_1: {E(x,y),E(y,z),E(z,w)}

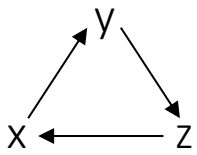
x ⟶ y ⟶ z ⟶ w

{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}

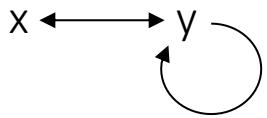
q_2: {E(x,y),E(y,z),E(z,x)}



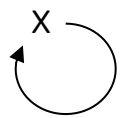
q_3: {E(x,y),E(y,x)}

x ⟷ y

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

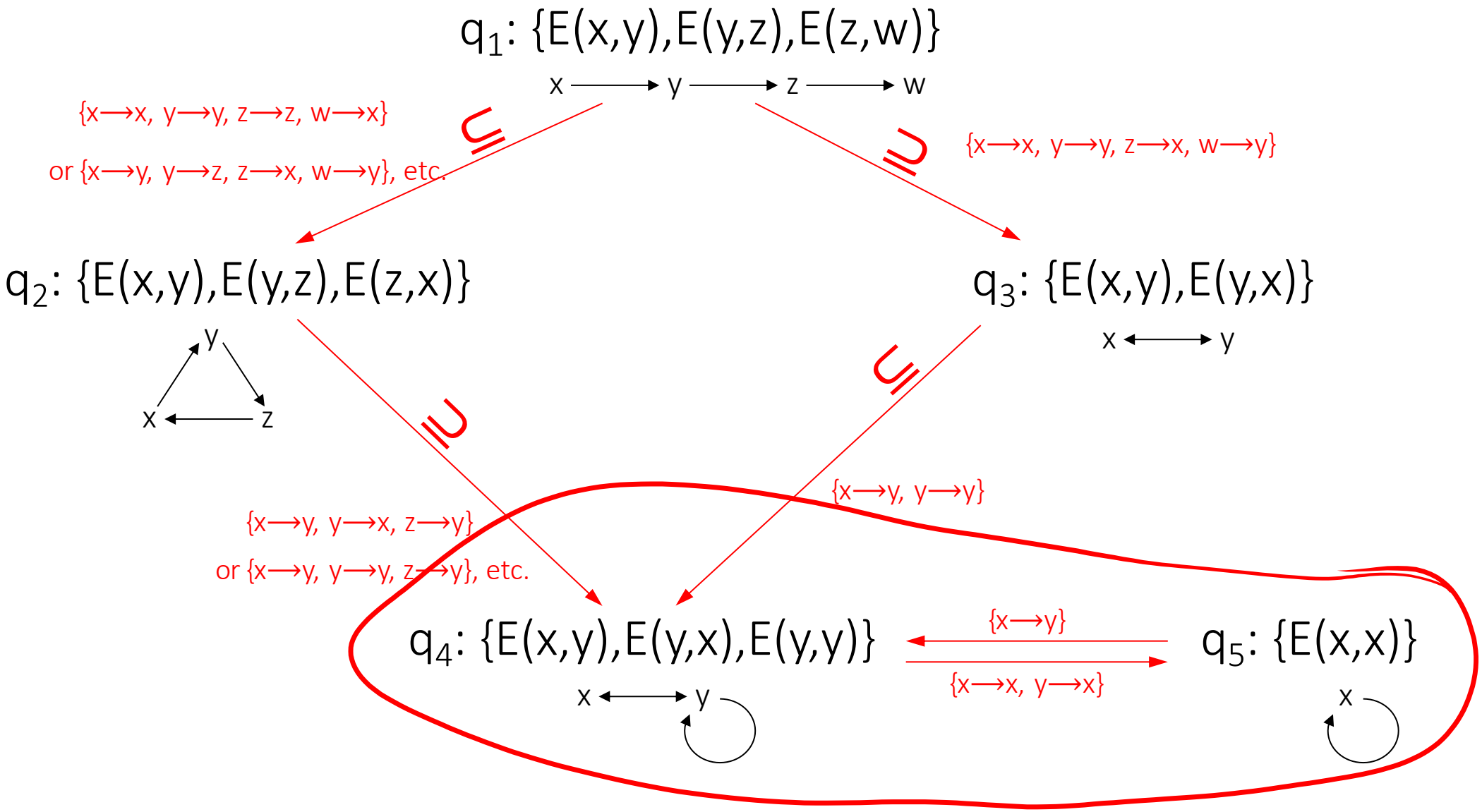


q_5: {E(x,x)}



Example by Andreas Pieris

Exercise: Find the Homomorphisms



Example by Andreas Pieris

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


$q_1(x,y)$:- 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)

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

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

$q_1 \longrightarrow q_2$

Thus ?

Which query contains the other?

Query Homeomorphism Practice


$q_1(x,y)$:- 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)

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

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

$q_1 \longrightarrow q_2$

Thus $q_2 \subseteq q_1$!

Query Homeomorphism Practice

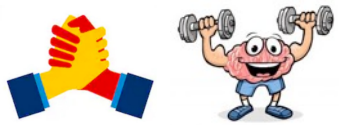

$q_1(x,y)$:- R(x,u),R(v,u),R(v,y) $var(q_1) = \{x, u, v, y\}$

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

Is there any homomorphism $q_2 \longrightarrow q_1$

and then $q_1 \subseteq q_2$?

Query Homeomorphism Practice

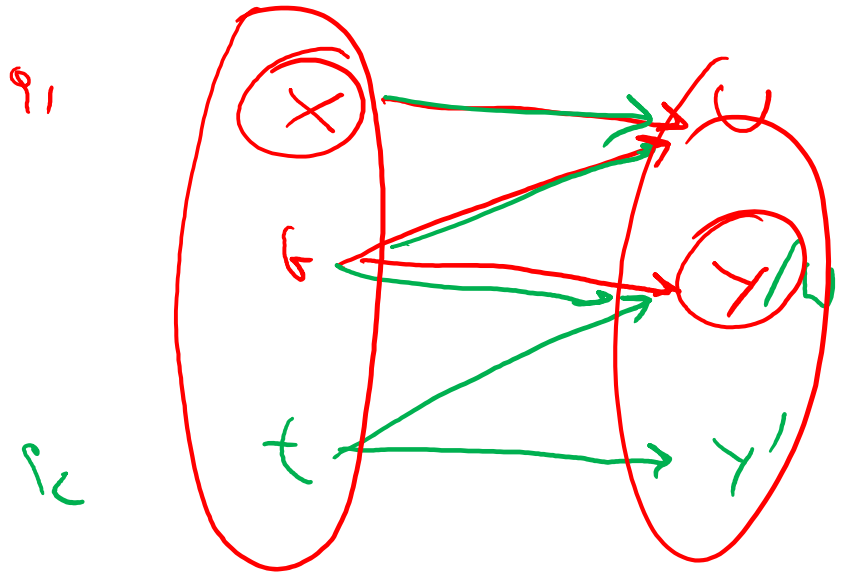


$q_1(x,y)$:- R(x,u),R(v,u),R(v,y)

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

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

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



$q_2 \longrightarrow q_1$

Thus $q_1 \subseteq q_2$?

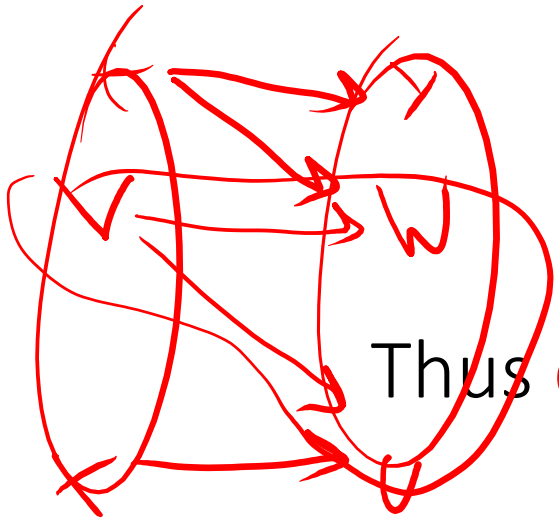
Query Homeomorphism Practice

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

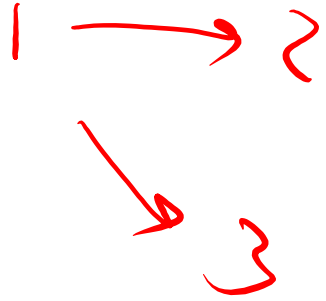
$q_2(x,y)$:- R(x,u),R(v,u),R(v,w),R(t,w),R(t,y)

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

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



Thus $q_1 \subseteq q_2$



Minimizing Conjunctive Queries

- Goal: minimize the number of joins in a query
- Definition: A conjunctive query Q is minimal if...

?

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. $Q \equiv Q'$
    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 be 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 :-E(x,y), E(y,z)

Q':-E(x,y)

We know Q' ⟶ Q

Thus: Q ⊆ Q'

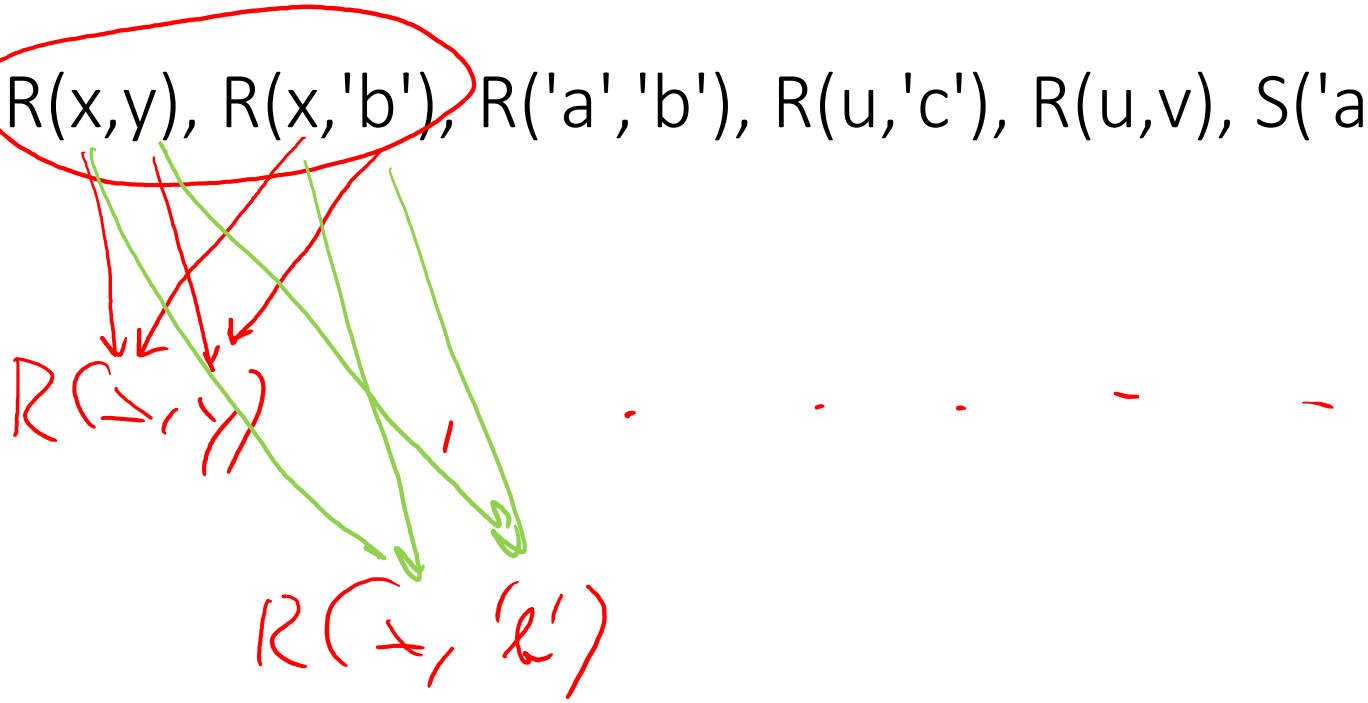
Q' ⊆ Q

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

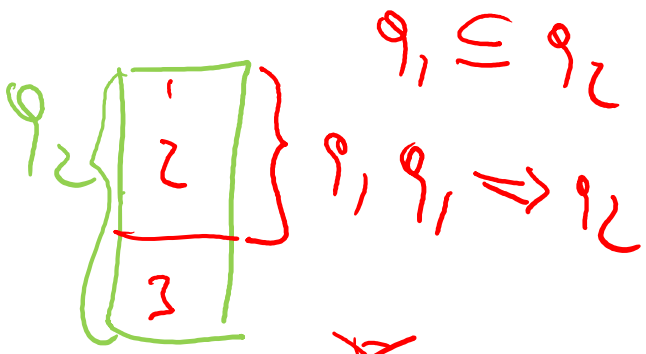
a,b,c,d are constants

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

{y⟶b}

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

q_2 TRUE q_1 FALSE



$q_1 \subseteq q_2$

$q_1, q_1 \Rightarrow q_2$

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

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)

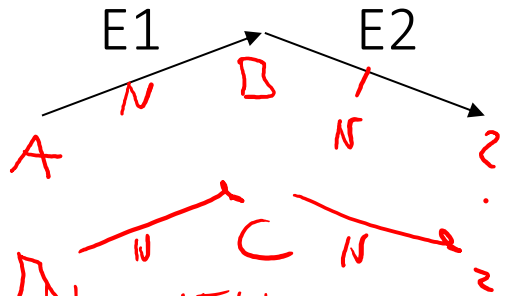
611

name	university	manager
Alice	Northeastern	Bob
Bob	Northeastern	Cecile
Cecile	Northeastern	
...	...	...

NEU employees managed by NEU emp.:

```
CREATE VIEW NeuMentors AS
SELECT DISTINCT E1.name,E1.manager
FROM Employee E1, Employee E2
WHERE E1.manager = E2.name
AND E1.university = 'Northeastern'
AND E2.university= 'Northeastern'
```

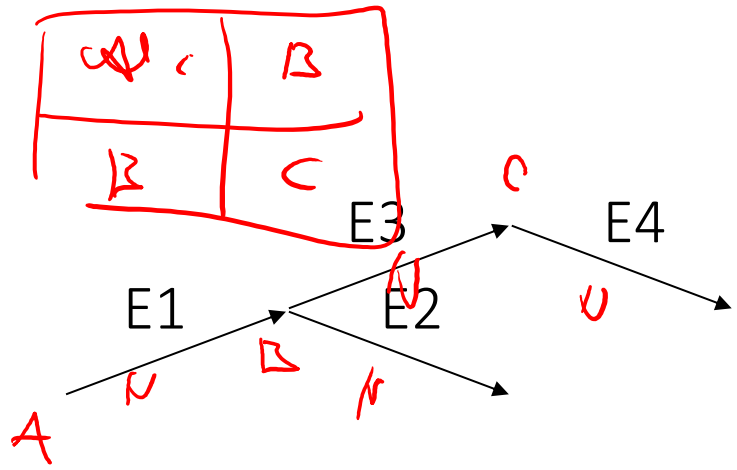
←This query / view is minimal



NEU emp. managed by NEU emp. managed by NEU emp.:

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

←This query is minimal



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.manager = E2.name AND E1.manager = E3.name AND E3.manager = E4.name
AND E1.university = 'Northeastern' AND E2.university = 'Northeastern'
AND E3.university = 'Northeastern' AND E4.university = 'Northeastern'
```

Is this query still minimal? ?

Example adopted from Dan Suciu

Query Minimization for Views

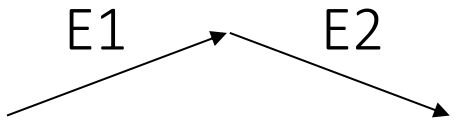
Employee(name, university, manager)

name	university	manager
Alice	Northeastern	Bob
Bob	Northeastern	Cecile
Cecile	Northeastern	
...	...	...

NEU employees managed by NEU emp.:

```
CREATE VIEW NeuMentors AS
SELECT DISTINCT E1.name,E1.manager
FROM Employee E1, Employee E2
WHERE E1.manager = E2.name
AND E1.university = 'Northeastern'
AND E2.university= 'Northeastern'
```

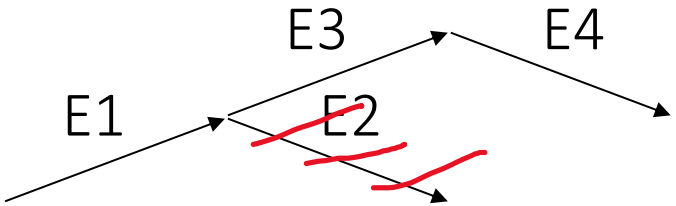
←This query / view is minimal



NEU emp. managed by NEU emp. managed by NEU emp.:

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

←This query is minimal



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.manager = E2.name AND E1.manager = E3.name AND E3.manager = E4.name
AND E1.university = 'Northeastern' AND E2.university = 'Northeastern'
AND E3.university = 'Northeastern' AND E4.university = 'Northeastern'
```

E2 is redundant!

Example adopted from Dan Suciu

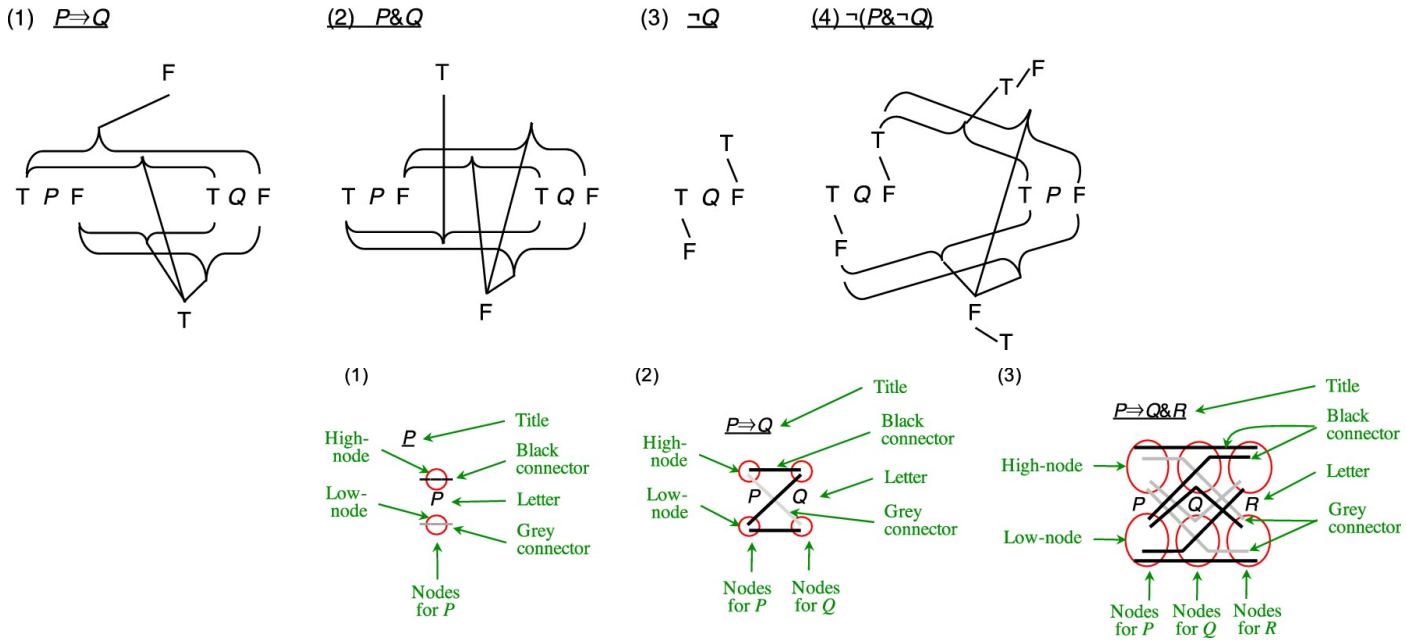
Outline: T2-1/2: Query Evaluation & Query Equivalence

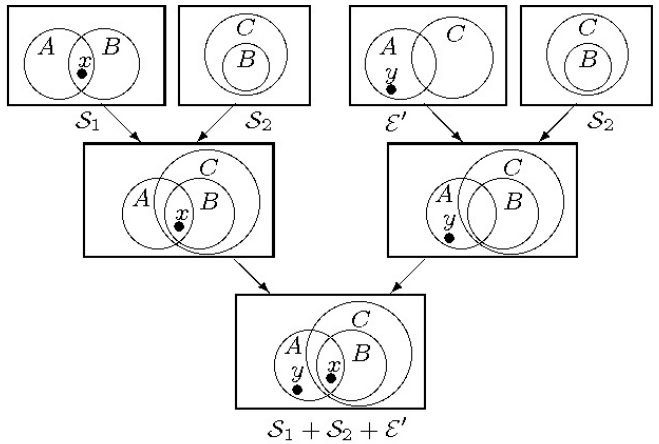
- T2-1: Conjunctive Queries (CQs)
  - CQ equivalence and containment
  - Graph homomorphisms
  - Homomorphism beyond graphs
  - CQ containment
  - CQ minimization
- T2-2: Equivalence Beyond CQs
  - Union of CQs, and inequalities
  - Union of CQs equivalence under bag semantics
  - Nested queries
  -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

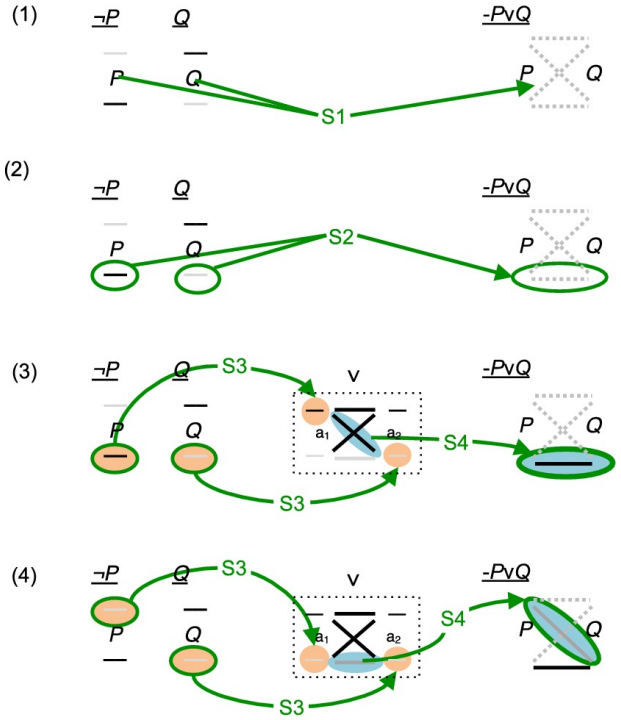
Equivalence of nested queries

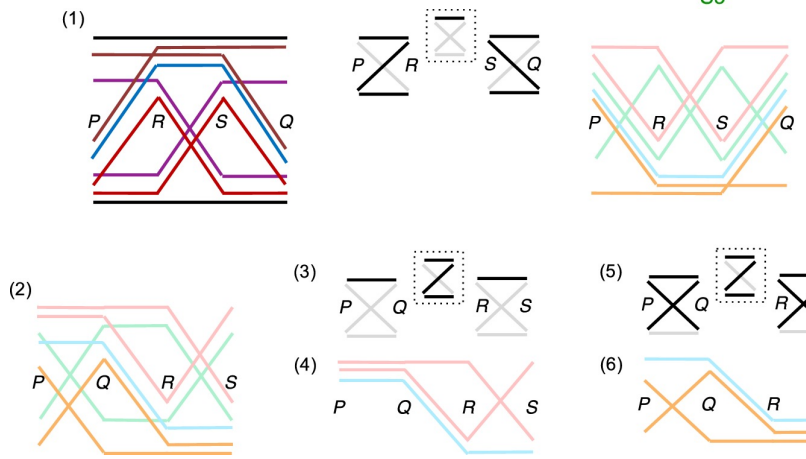
- Query equivalence is one of the foundational questions in database theory (and practice?)
  - touches on logics and decidability
  - what modifications allow tractability
- Lots of work (and open questions) on query equivalence
  - But not so much on nested queries!
- Related to QueryViz project (http://queryviz.com) 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

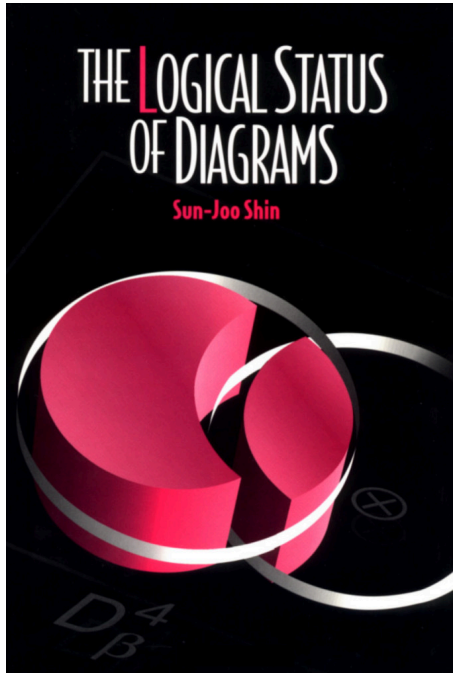








Diagrammatic reasoning systems and their expressiveness



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? ?

```
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)))
```

Nesting Depth

"Query blocks"

Likes
drinker
beer

Scopes of tuple variables

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

```
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)))
```

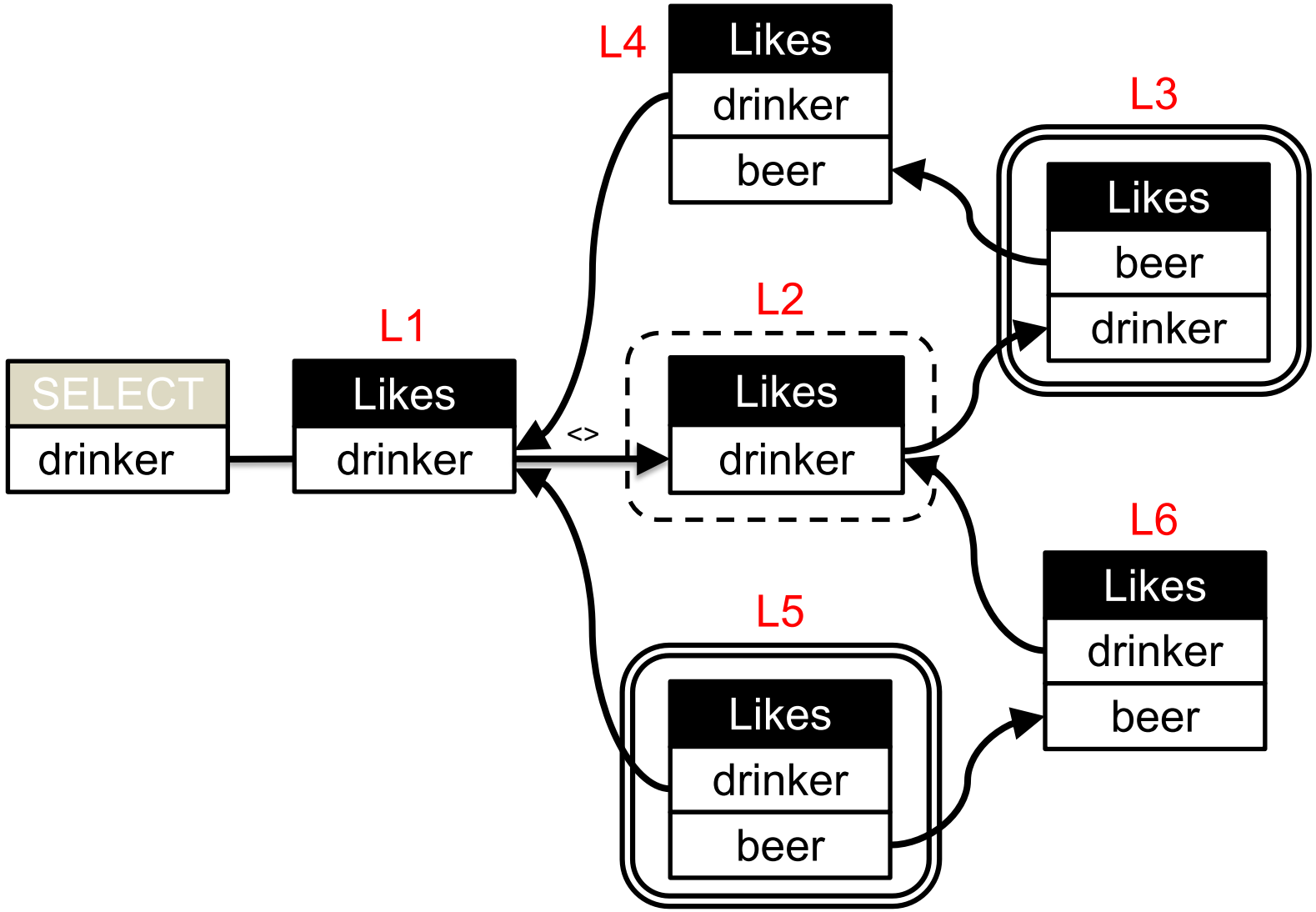
"Query blocks"

Nesting Depth

Likes
drinker
beer

Scopes of tuple variables

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



Leventidis+ QueryVis: Logic-based Diagrams help Users Understand Complicated SQL Queries Faster, SIGMOD 2020. https://doi.org/10.1145/3318464.3389767

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."

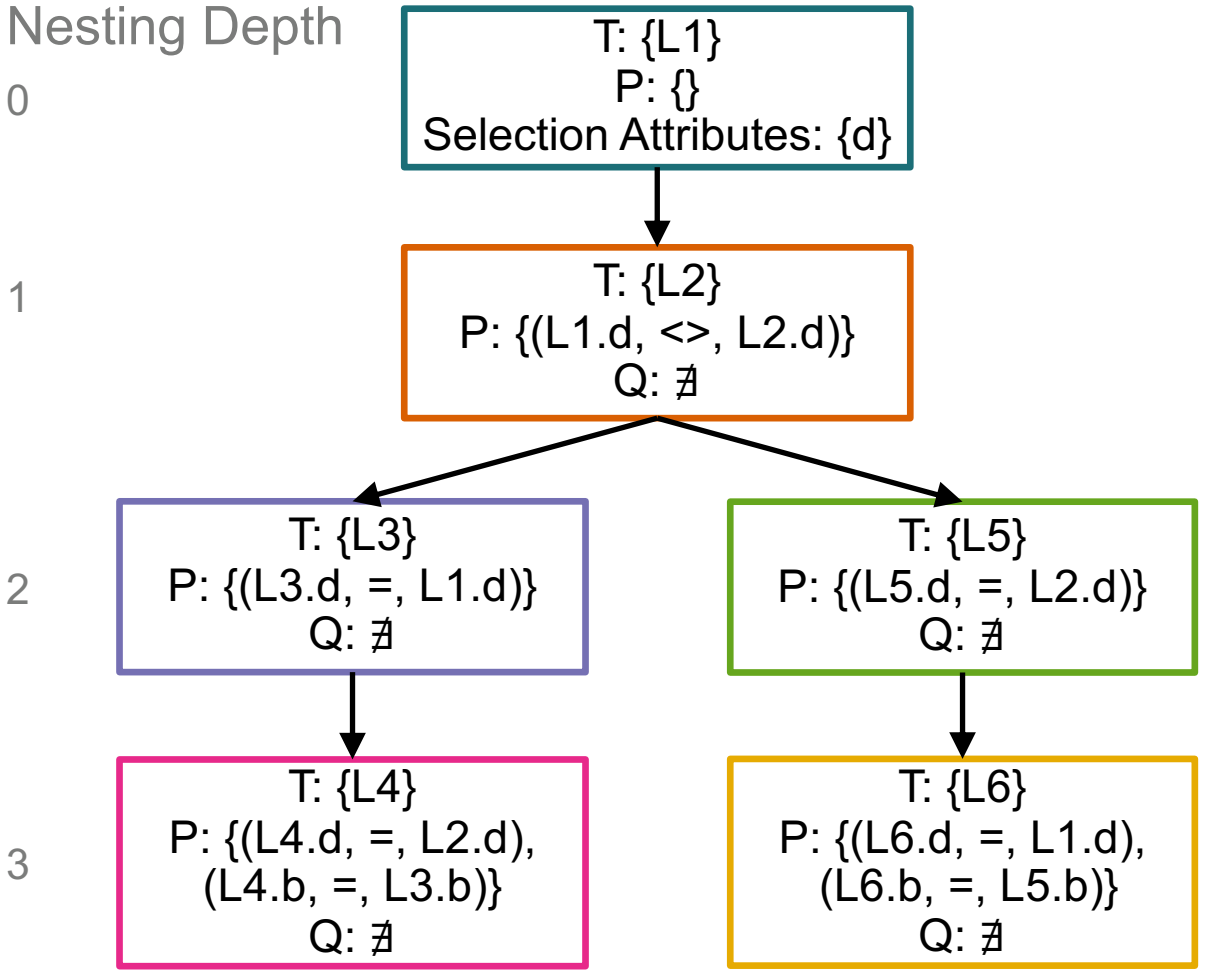
Let x be a drinker, and S(x) be the set of liked beers by drinker x.
Find any drinker x, s.t. there does not exist another drinker x′, x for which:
$S(x') \subseteq S(x)$ *and* $S(x') \supseteq S(x)$

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

Likes
drinker
beer

Likes
d
b

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



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

scope \ type	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 selection 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?

Simplifying notation

Schema: R(A,B)

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

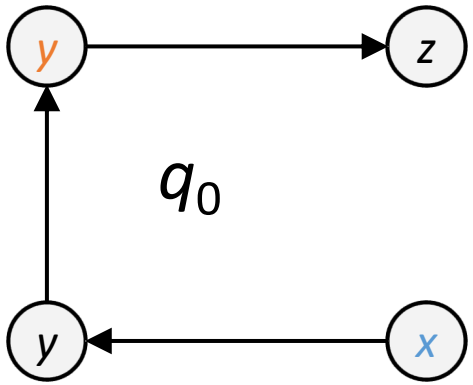
```
SELECT  TRUE
FROM    R R1, R R2, R R3
WHERE   R1.B = R2.A
AND     R2.B = R3.A
NOT EXISTS
        (SELECT *
        FROM    R R4, R R5, R R6
        WHERE   R4.B = R5.A
        AND     R5.B = R6.A
        AND     R4.A = R1.A
        AND     R6.A = R2.B)
```

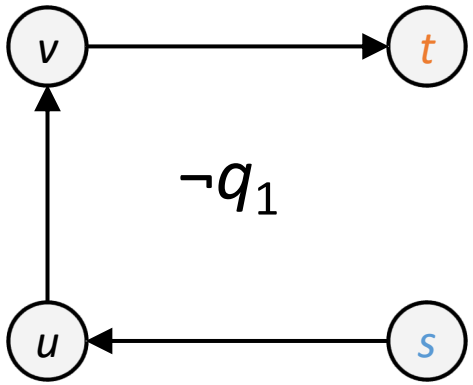
$$q_0 :- R(x,y), R(y,z), R(z,w)$$

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

$$q_0 :- R(x,y), R(y,z), R(z,w), \neg q_1(x,z)$$

$$\exists R1, R2, R3 \in R \; (R1.B=R2.A \wedge R2.B=R3.A \wedge \nexists R4, R5, R6 \in R \; (R4.B=R5.A \wedge R5.B=R6.A \wedge R4.A=R1.A \wedge R6.A = R2.B))$$





s=x, t=y

Simplifying notation

Schema: R(A,B)

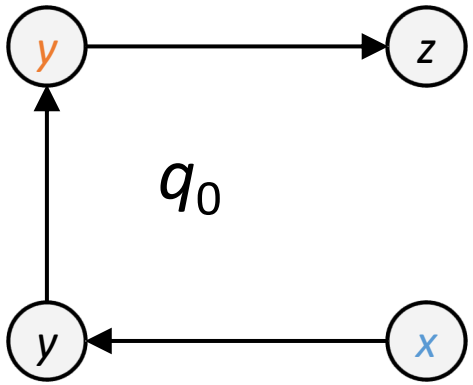
What will become handy, is a short convenient notation for queries

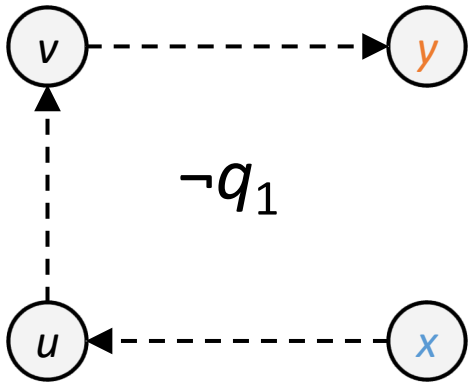
```
SELECT  TRUE
FROM    R R1, R R2, R R3
WHERE   R1.B = R2.A
AND     R2.B = R3.A
NOT EXISTS
        (SELECT *
        FROM    R R4, R R5, R R6
        WHERE   R4.B = R5.A
        AND     R5.B = R6.A
        AND     R4.A = R1.A
        AND     R6.A = R2.B)
```

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

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

$\exists$ R1, R2, R3 $\in$ R
(R1.B=R2.A $\wedge$ R2.B=R3.A $\wedge$
$\nexists$ R4, R5, R6 $\in$ R
(R4.B=R5.A $\wedge$ R5.B=R6.A $\wedge$
R4.A=R1.A $\wedge$ R6.A = R2.B)
)





Simplifying notation

Schema: R(A,B)

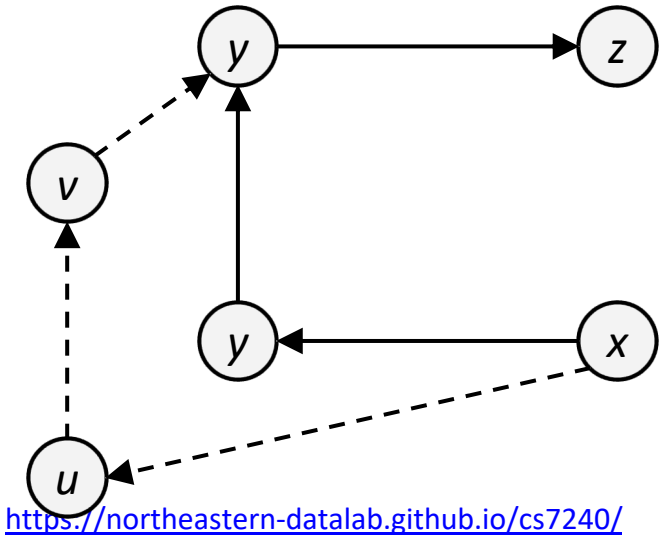
What will become handy, is a short convenient notation for queries

```
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)
```

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

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

$$\exists\, R1, R2, R3 \in R$$
$$(R1.B=R2.A \wedge R2.B=R3.A \wedge$$
$$\nexists\, R4, R5, R6 \in R$$
$$(R4.B=R5.A \wedge R5.B=R6.A \wedge$$
$$R4.A=R1.A \wedge R6.A = R2.B)$$
$$)$$



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

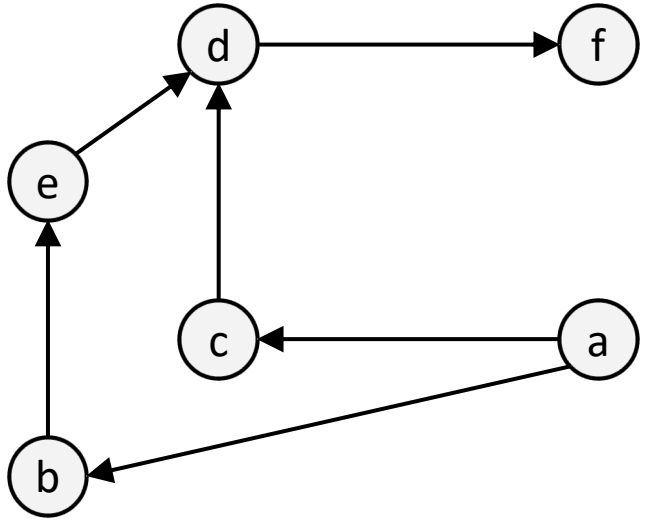
See Barany, Cate, Segoufin, "Guarded negatation ", JACM 2015

guardedness

Exercise

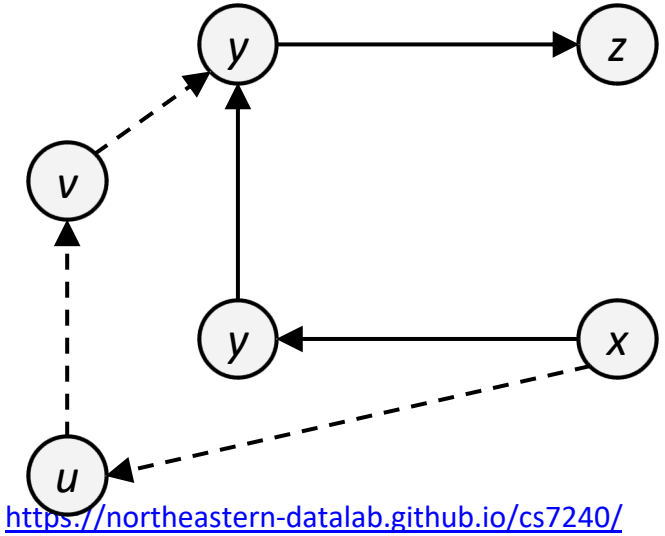
Schema: R(A,B)

Database *D*



Does the query below evaluate to true on above database?

Query *q*

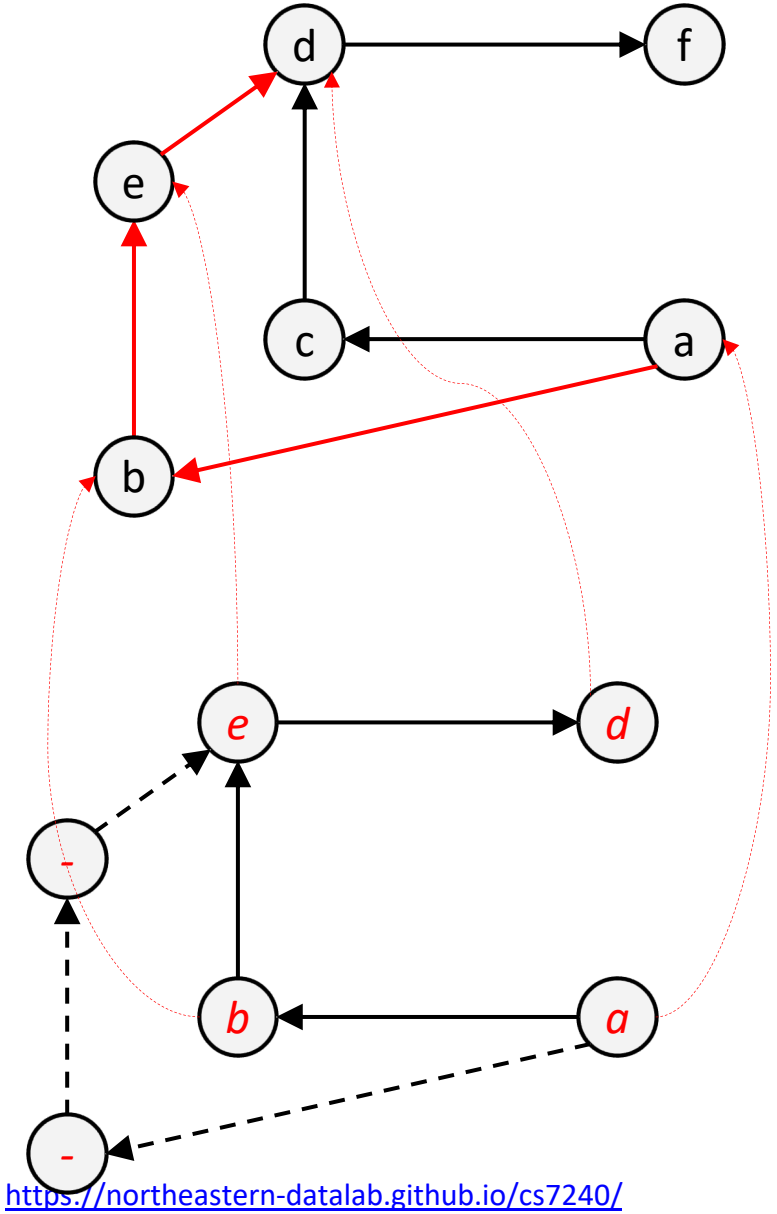


Exercise

Schema: R(A,B)

Database *D*

Query *q*



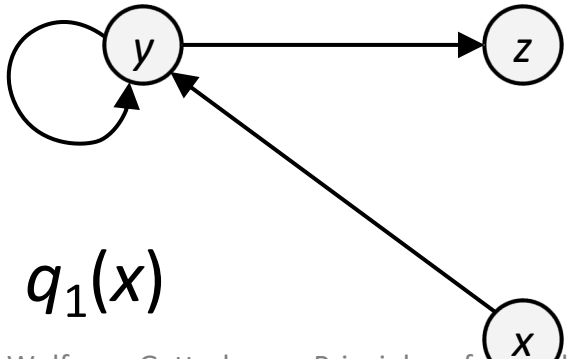
Question

- 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?

Example

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

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



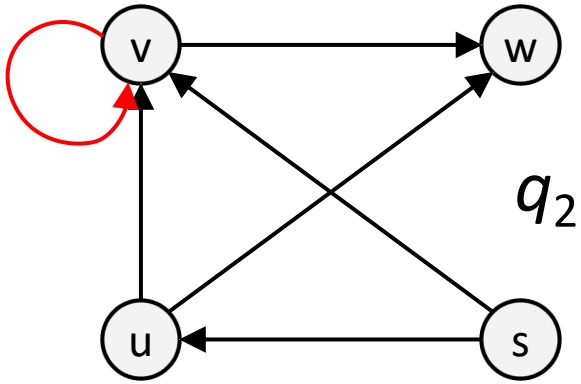
$q_1(x)$

$h_{1\to2}$: {(x,s),(y,v),(z,w)}

$q_1 \not\subseteq q_2$

$h_{2\to1}$: {(s,x),(u,y),(v,y),(w,z)}

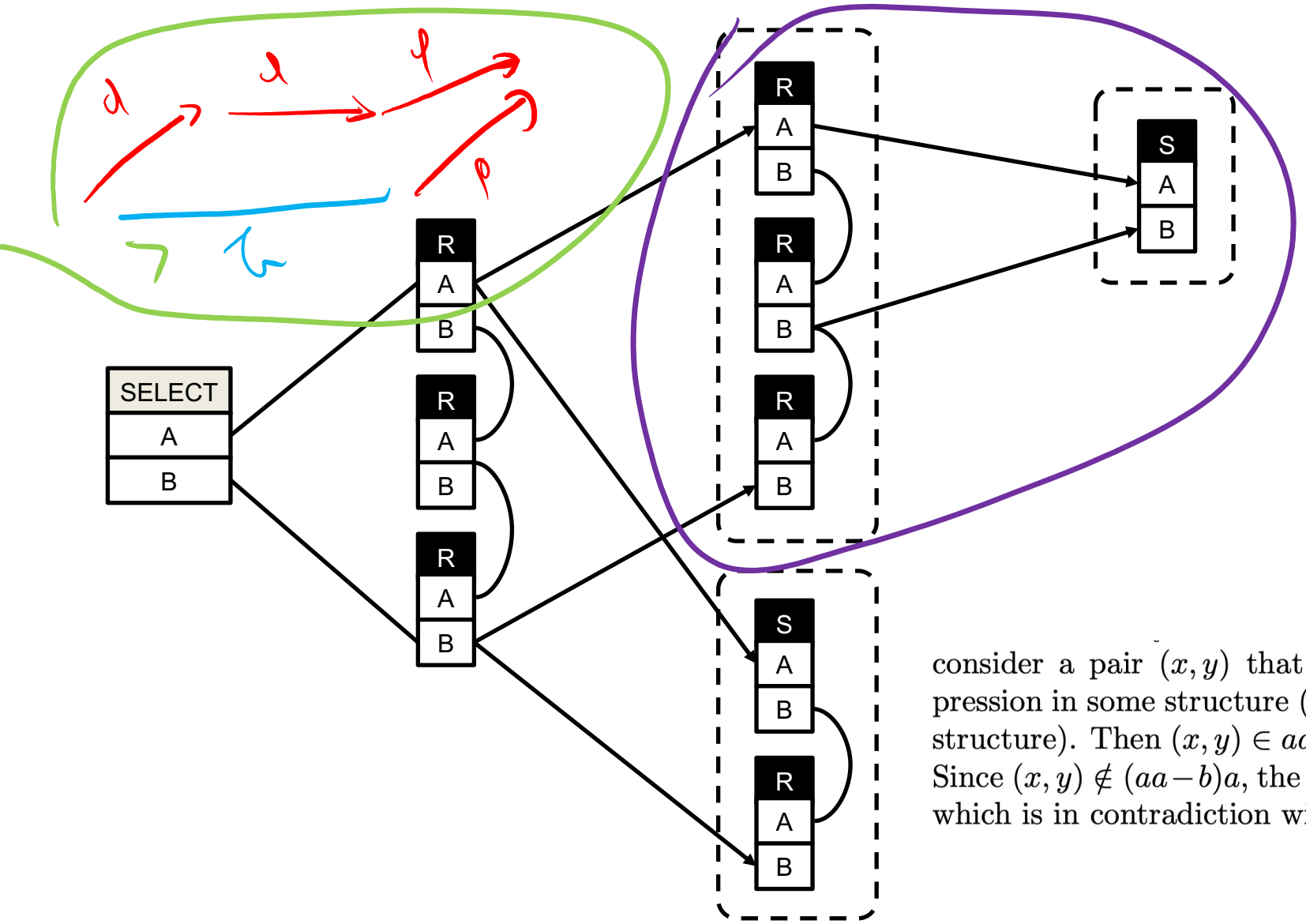
$q_1 \subseteq q_2$



$q_2(x)$

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



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

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

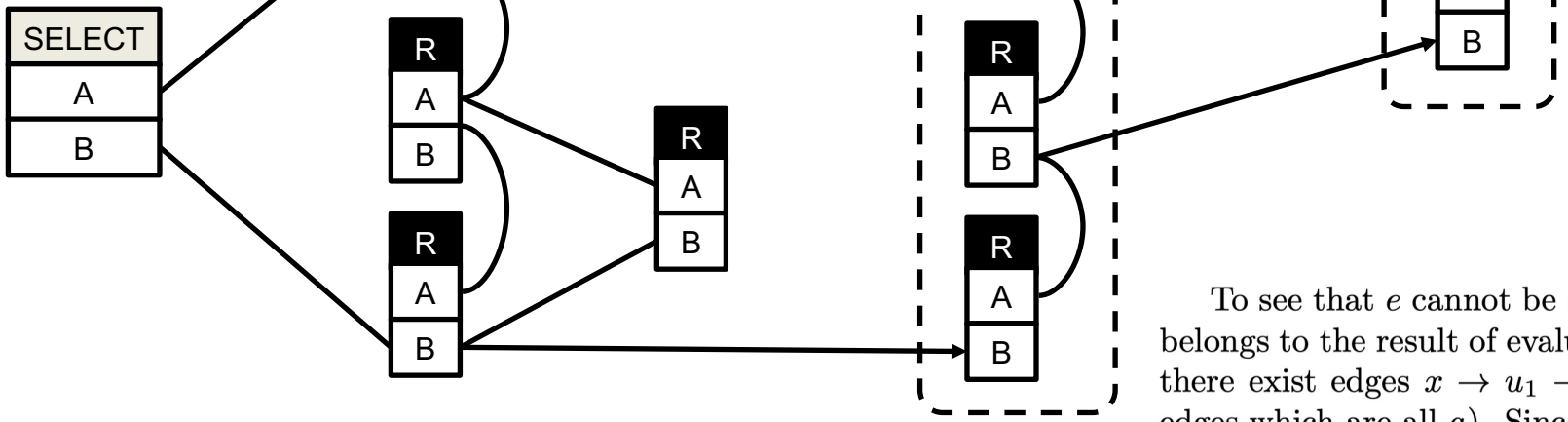
$= aef - aef \cup bf - bf$

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. □

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

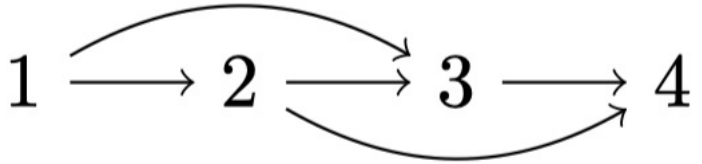
$X - (Y \cup Z) = X - Y - Z$

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



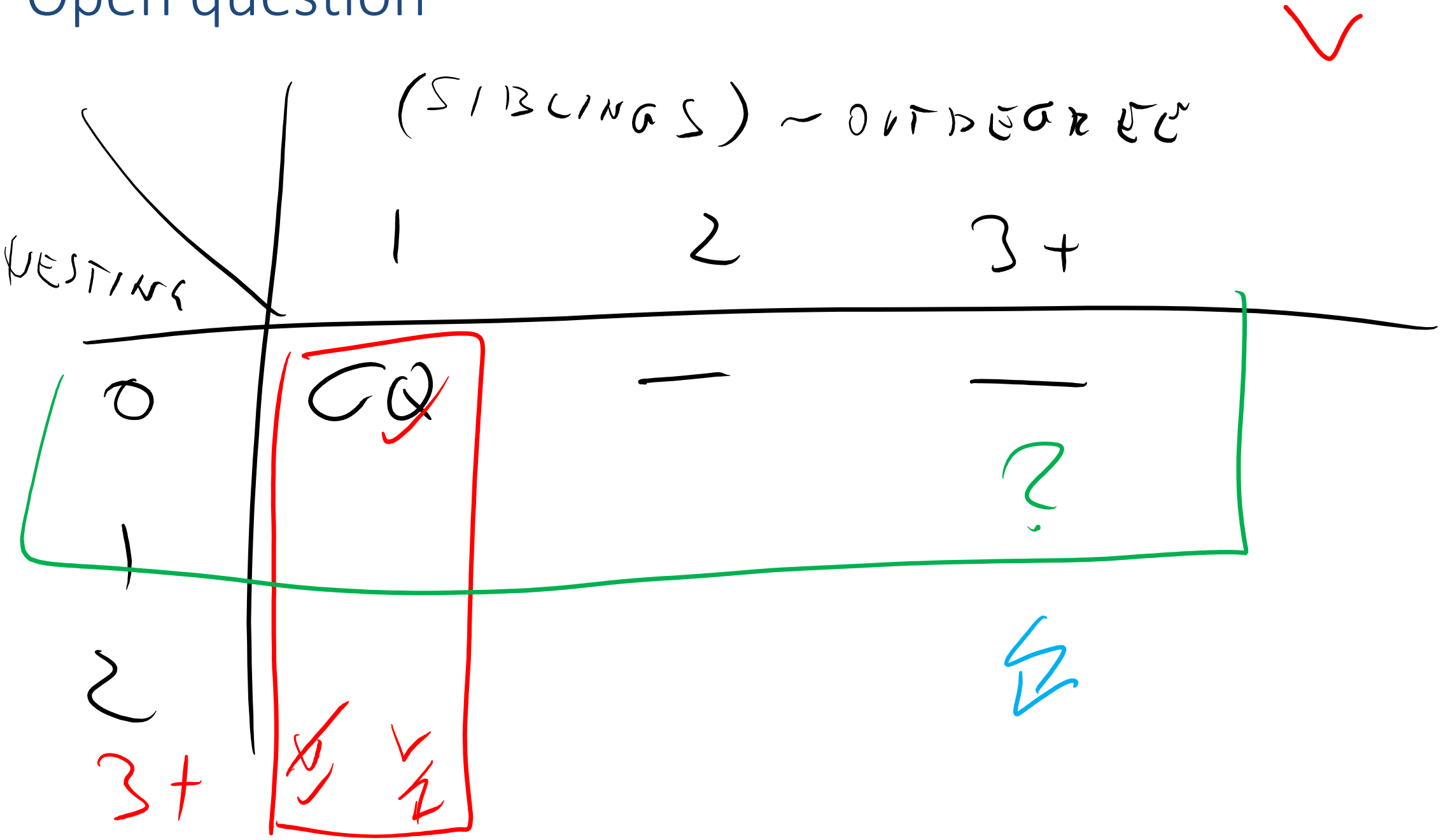
To see that e cannot be satisfied by any series-parallel graph, suppose (x, y) belongs to the result of evaluating e on some structure. Since $(x, y) \in a(a \cap aa)$, there exist edges $x \to u_1 \to u_2 \to y$ and $u_1 \to y$ (we omit the labels on the edges which are all a). Since $(x, y) \notin (aa - a)a$, 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



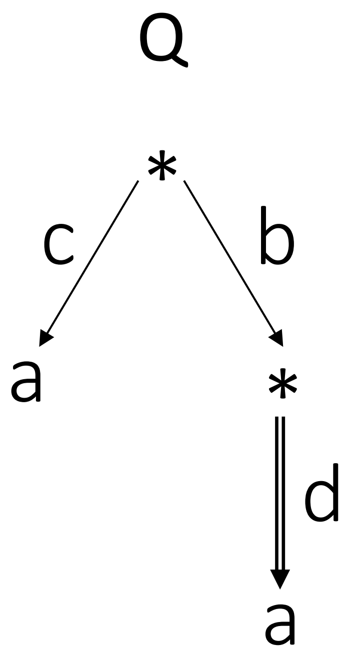
Pointers to related work

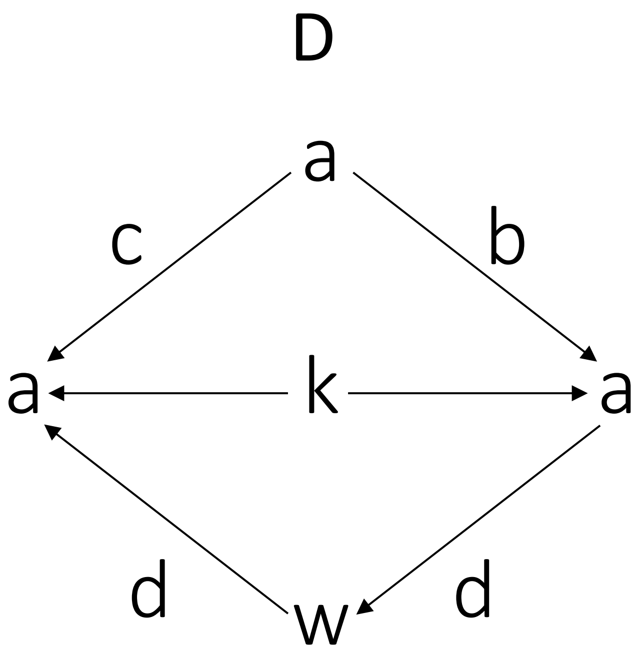
- Kolaitis. *Logic and Databases*. Logical Structures in Computation Boot Camp, Berkeley 2016. https://simons.berkeley.edu/talks/logic-and-databases
- Abiteboul, Hull, Vianu. *Foundations of Databases*. Addison Wesley, 1995. http://webdam.inria.fr/Alice/, Ch 2.1: Theoretical background, Ch 6.2: Conjunctive 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
- Gatterbauer. *Databases will visualize queries too*. PVLDB 2011. http://www.vldb.org/pvldb/vol4/p1498-gatterbauer.pdf

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
  - Nested queries
  - Tree pattern queries

Tree pattern queries

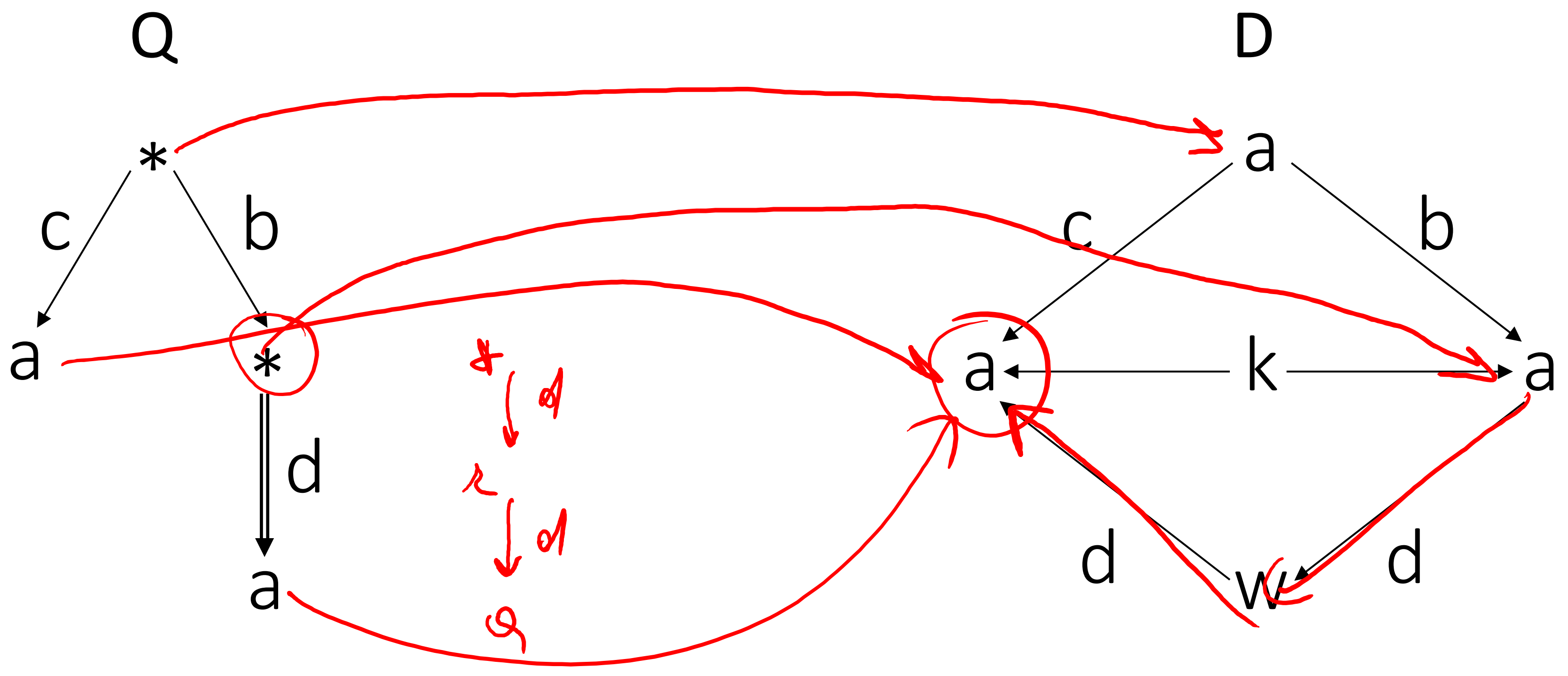




Does the query on the left find a match on in the data on the right (i.e. is there a homomorphism from left to right)? ?

Notice that "a", "b", "c" are labels (not node ids), thus like constants in a query

Tree pattern queries



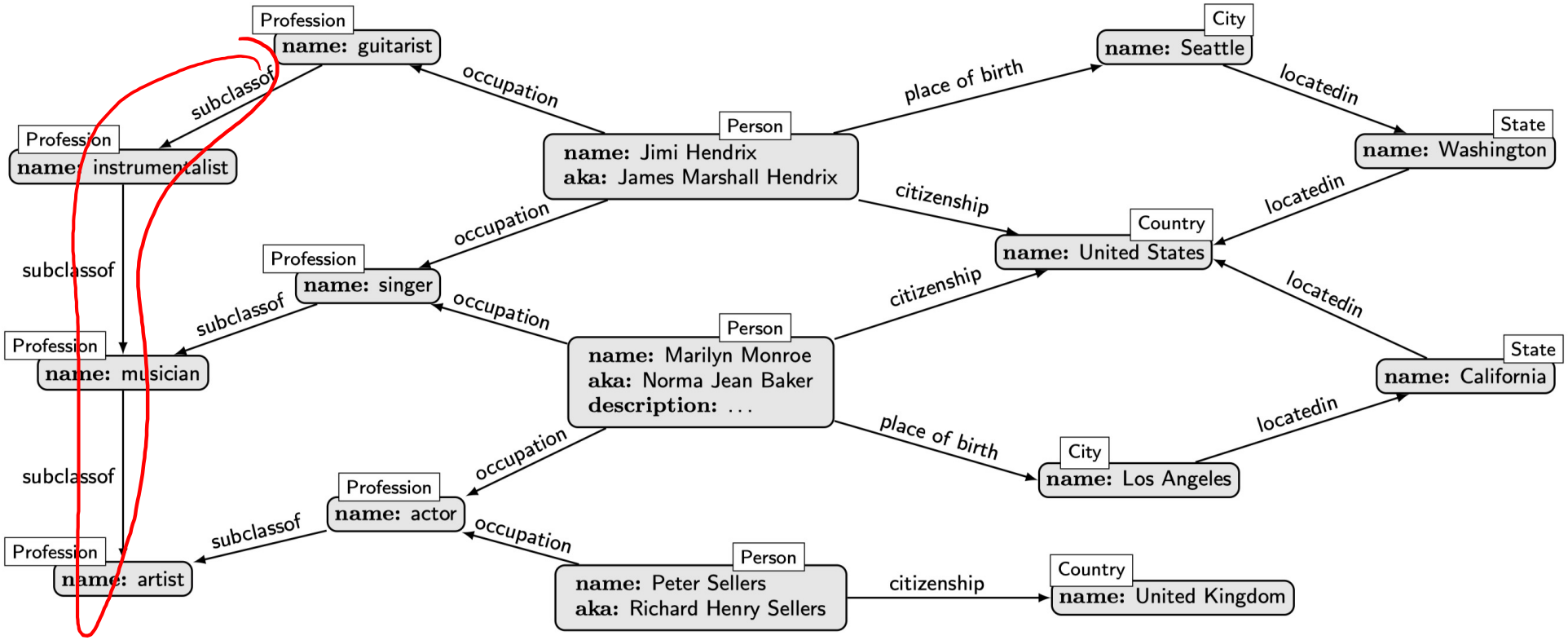


Figure 1: A graph database (as a *property graph*), inspired on a fragment of WikiData

?

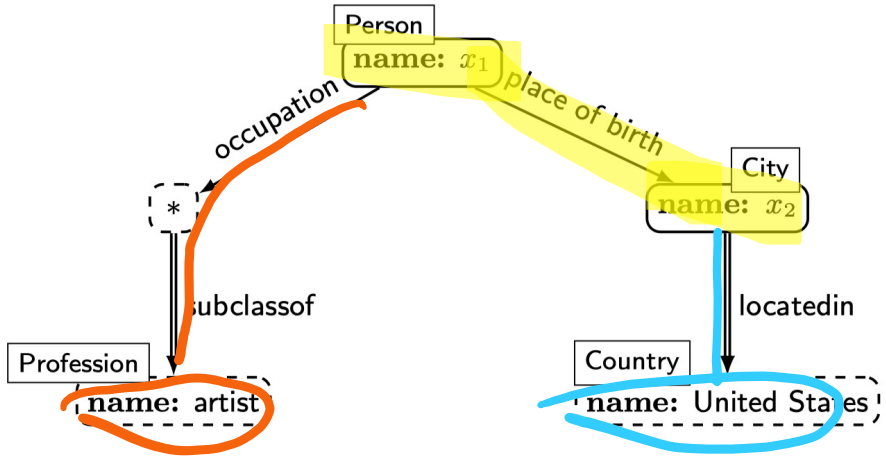


Figure 2: A tree pattern finding the artists who were born in the United States. The query returns the person names and the cities where they were born. (Fully circled nodes are return nodes.)

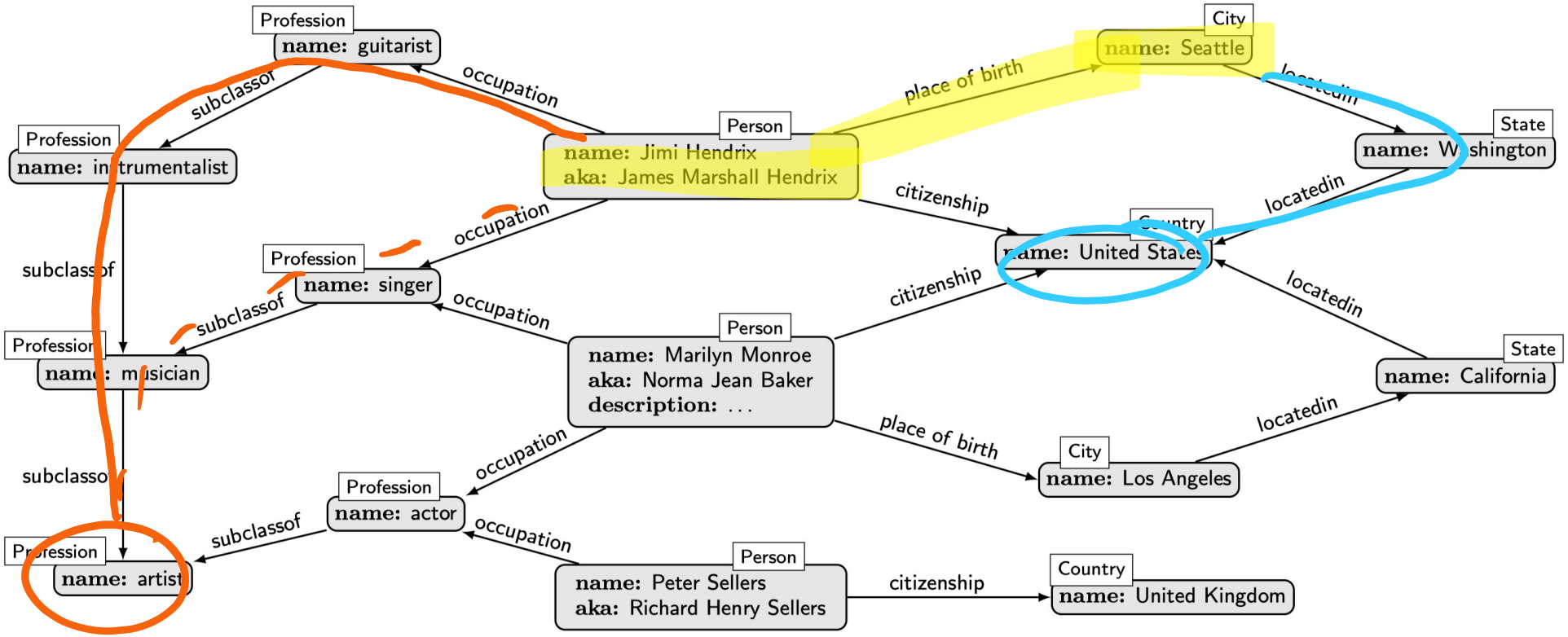


Figure 1: A graph database (as a *property graph*), inspired on a fragment of WikiData

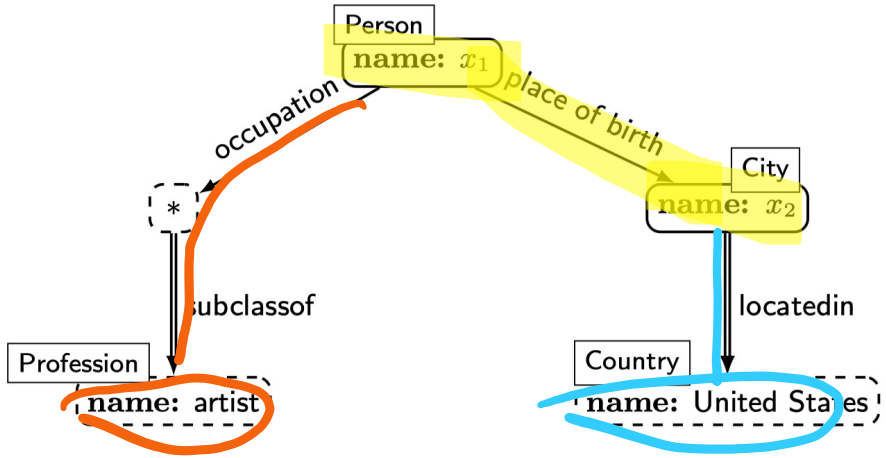


Figure 2: A tree pattern finding the artists who were born in the United States. The query returns the person names and the cities where they were born. (Fully circled nodes are return nodes.)

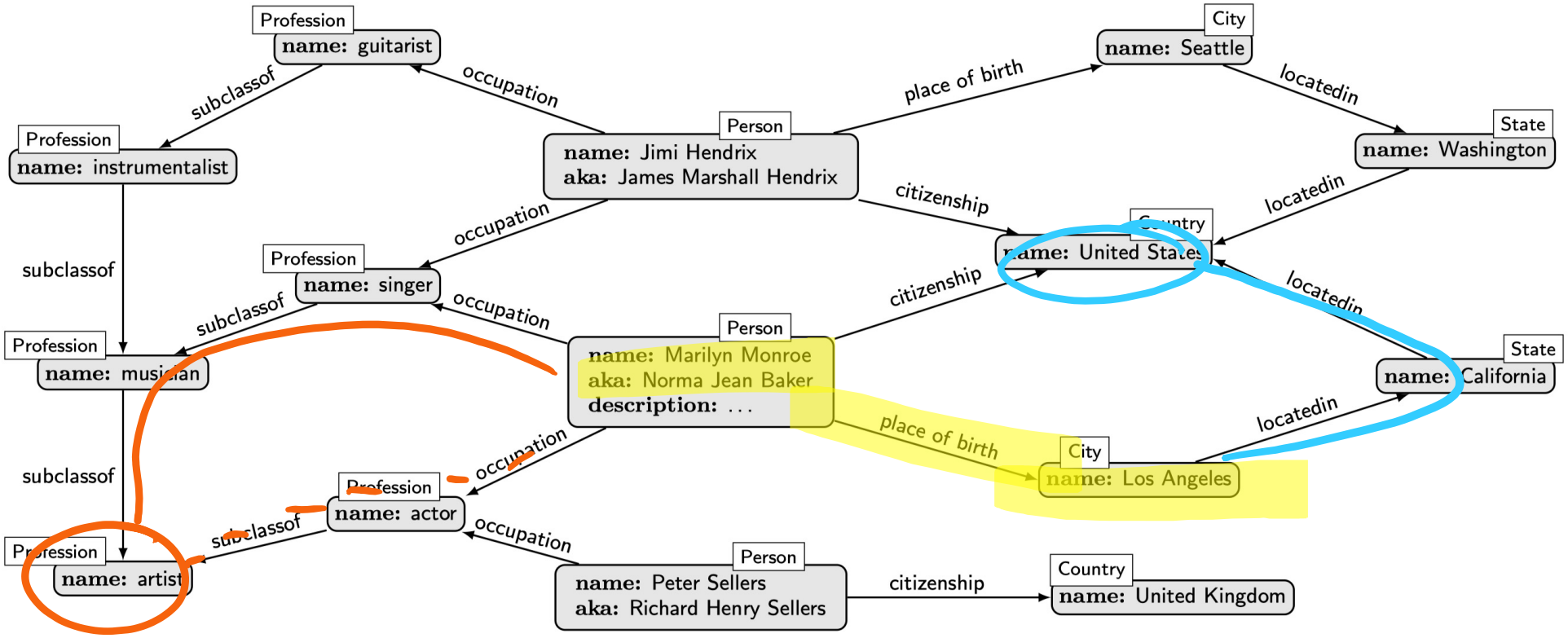


Figure 1: A graph database (as a *property graph*), inspired on a fragment of WikiData

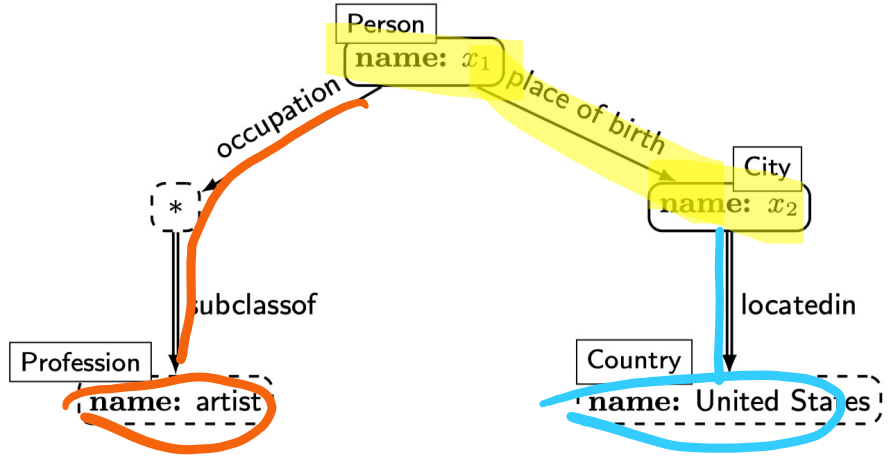
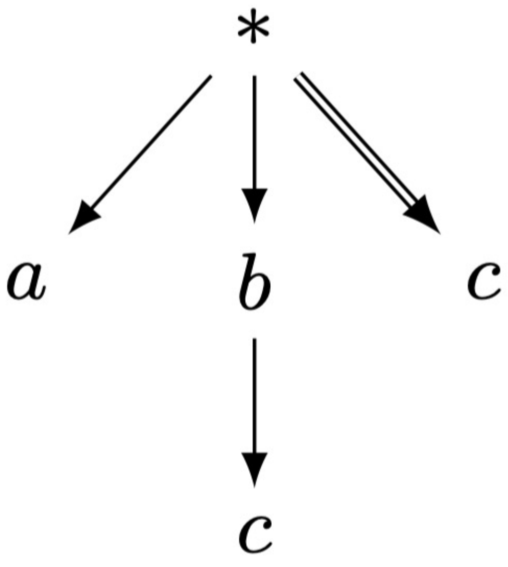
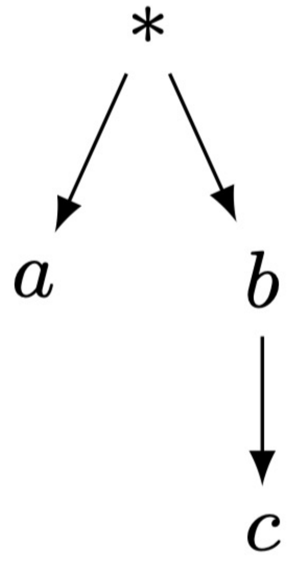


Figure 2: A tree pattern finding the artists who were born in the United States. The query returns the person names and the cities where they were born. (Fully circled nodes are return nodes.)

Optimizing tree patterns



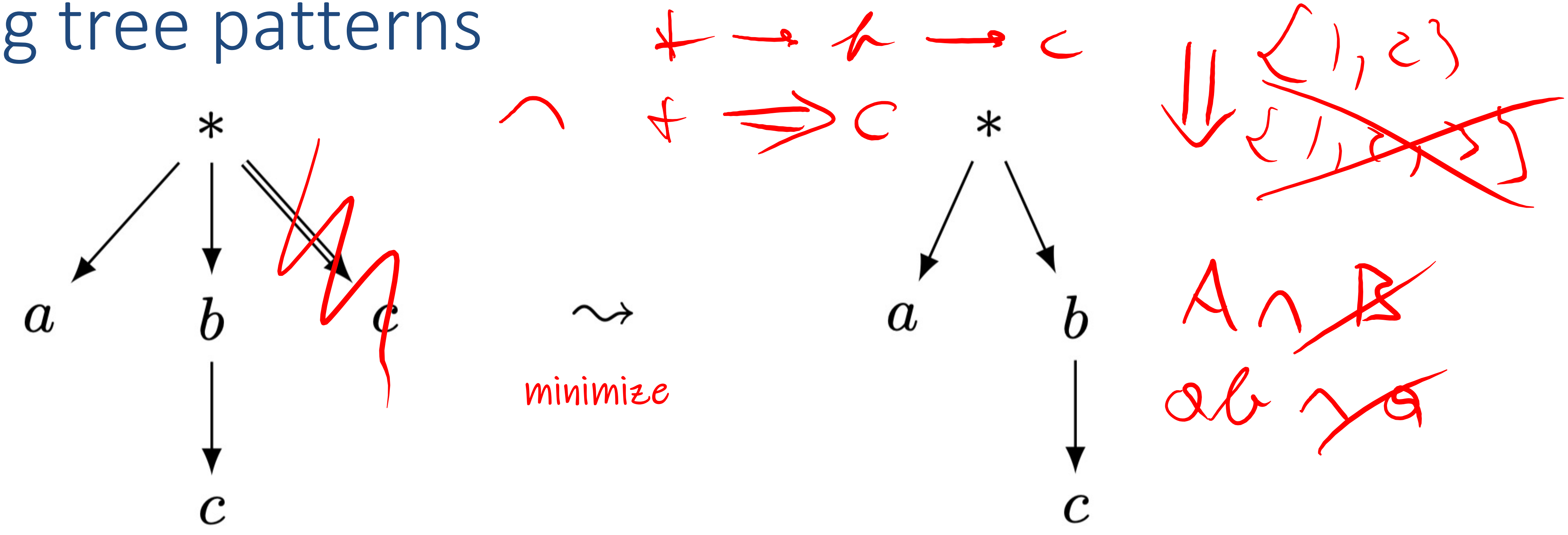


How are those two tree patterns related to each other?

?

Example from: "Optimizing Tree Patterns for Querying Graph- and Tree-Structured Data" by Czerwinski, Martens, Niewerth, Parys. SIGMOD record 2017. https://doi.org/10.1145/3093754.3093759

Optimizing tree patterns



TREE PATTERN MINIMIZATION	
Given:	A tree pattern p and $k \in \mathbb{N}$
Question:	Is there a tree pattern q, equivalent to p, such that its size is at most k?

Minimality =? Nonredundancy

1.4 History of the Problem

Although the patterns we consider here have been widely studied [14, 24, 36, 15, 22, 1, 9, 4, 32], their minimization problem remained elusive for a long time. The most important previous work for their minimization was done by Kimelfeld and Sagiv [22] and by Flesca, Furfaro, and Masciari [14, 15].

The key challenge was understanding the relationship between *minimality* (M) and *nonredundancy* (NR). Here, a tree pattern is minimal if it has the smallest number of nodes among all equivalent tree patterns. It is nonredundant if none of its leaves (or branches[2]) can be deleted while remaining equivalent. The question was if minimality and nonredundancy are the same ([22, Section 7] and [15, p. 35]):

> M $\stackrel{?}{=}$ NR PROBLEM:
>
> Is a tree pattern minimal
> if and only if it is nonredundant?

Notice that a part of the M $\stackrel{?}{=}$ NR problem is easy to see: a minimal pattern is trivially also nonredundant (that is, M $\subseteq$ NR). The opposite direction is much less clear.

If the problem would have a positive answer, it would mean that the simple algorithmic idea summarised in Algorithm 1 correctly minimizes tree patterns. Therefore, the M $\stackrel{?}{=}$ NR problem is a natural question about the design of minimization algorithms for tree patterns.

Algorithm 1 Computing a nonredundant subpattern

Input: A tree pattern p
Output: A nonredundant tree pattern q, equivalent to p

```
while a leaf of p can be removed
                    (remaining equivalent to p) do
    Remove the leaf
end while
return the resulting pattern
```

The M $\stackrel{?}{=}$ NR problem is also a question about complexity. The main source of complexity of the nonredundancy algorithm lies in testing equivalence between a pattern p and a pattern p', which is generally coNP-complete [24]. If M $\stackrel{?}{=}$ NR has a positive answer, then TREE PATTERN MINIMIZATION would also be coNP-complete.

In fact, the problem was claimed to be coNP-complete in 2003 [14, Theorem 2], but the status of the minimization- and the M $\stackrel{?}{=}$ NR problems were re-opened by Kimelfeld and Sagiv [22], who found errors in the proofs. Flesca et al.'s journal paper then proved that M = NR for a limited class of tree patterns, namely those where *every wildcard node has at most one child* [15]. Nevertheless, for tree patterns,

(a) the status of the M $\stackrel{?}{=}$ NR problem and

(b) the complexity of the minimization problem

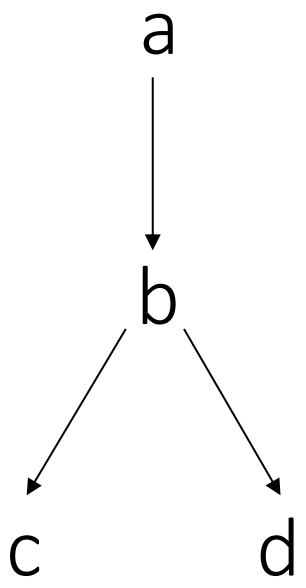
remained open.

Czerwinski, Martens, Niewerth, Parys [PODS 2016}

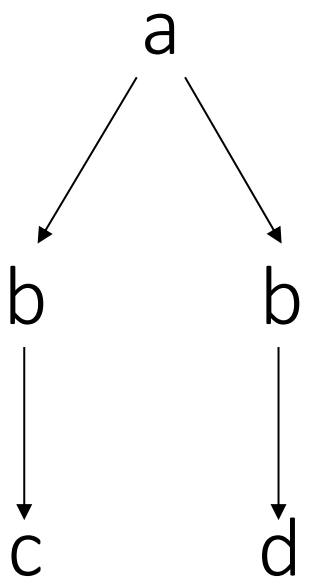
(a) There exists a tree pattern that is nonredundant but not minimal. Therefore, M $\neq$ NR.

(b) TREE PATTERN MINIMIZATION is Σ_2^P-complete. This implies that even the main idea in Algorithm 1 cannot work unless coNP = Σ_2^P.

Tree pattern containment



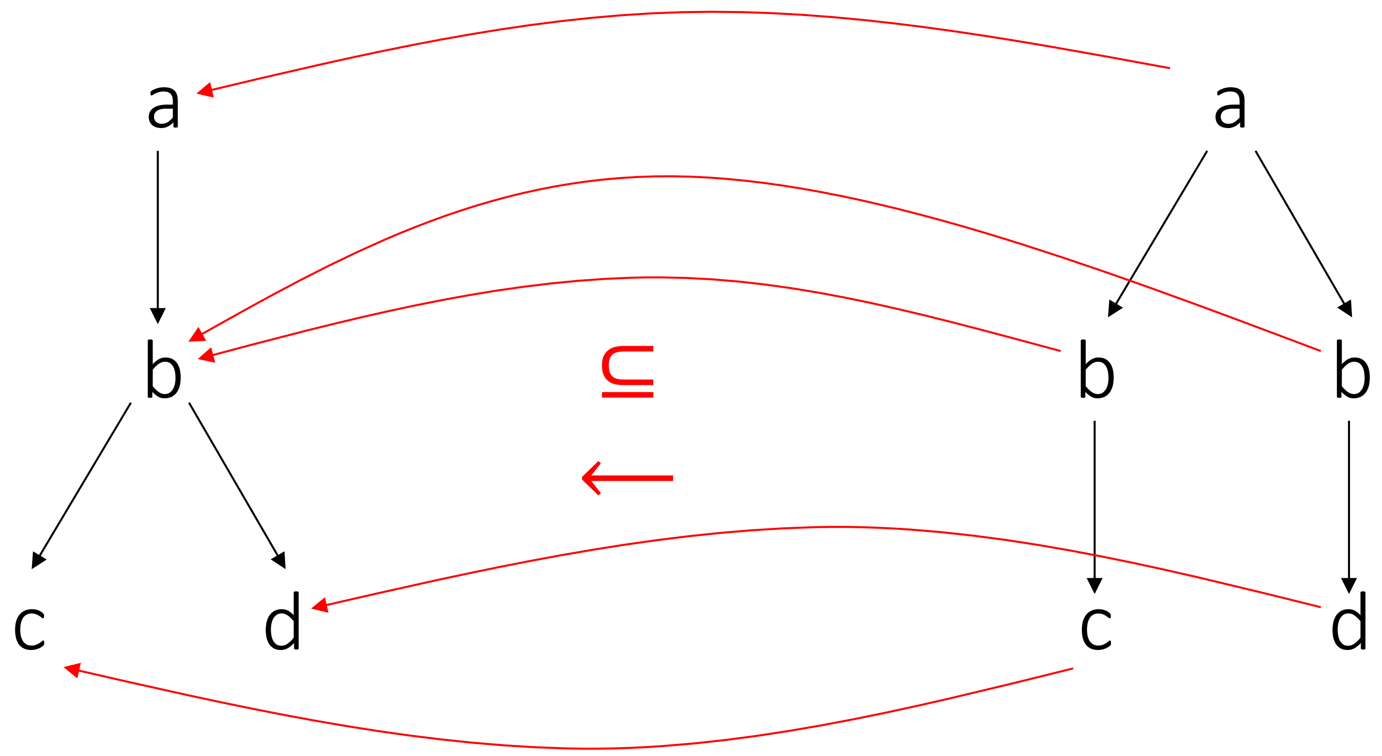
$\subseteq$
or
$\supseteq$



?

Example from G. "Optimizing Tree Patterns for Subgraph Embedding". In Graph Data Management: Fundamental Issues and Recent Developments. SIGMOD record 2017. https://doi.org/10.1145/3093754.3093759

Tree pattern containment



but ⊉!

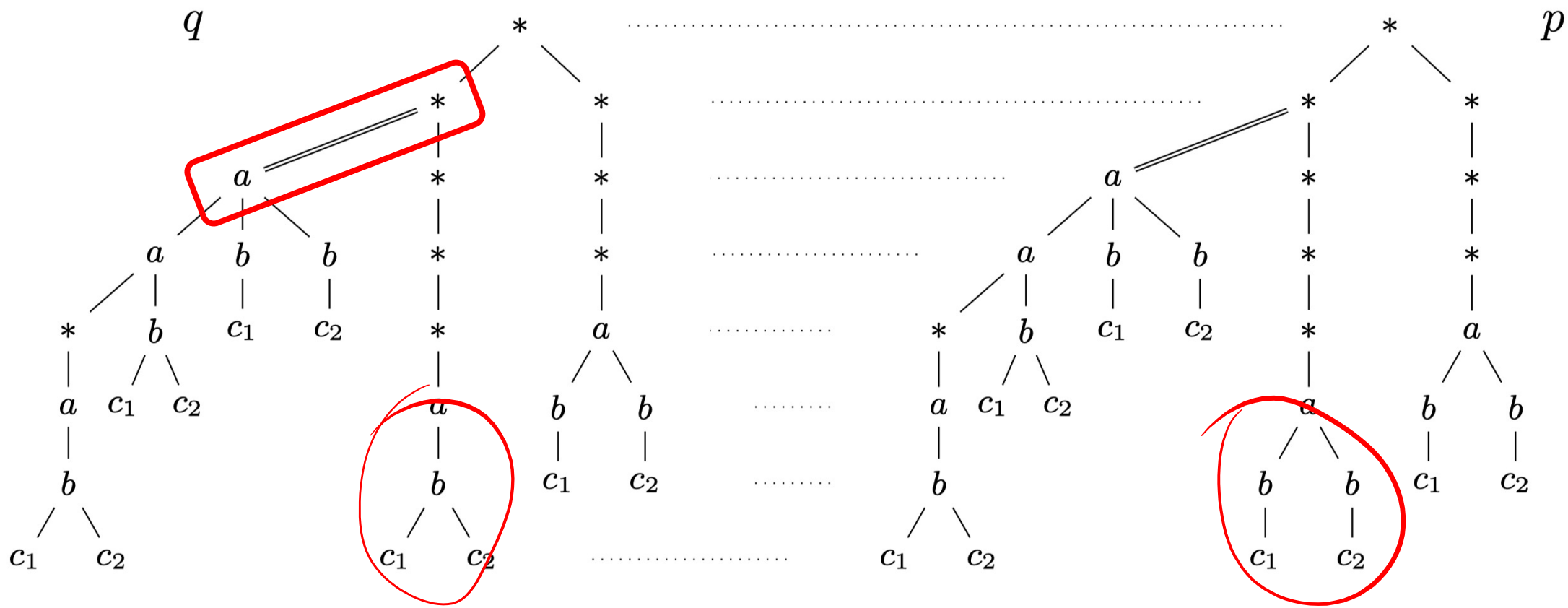


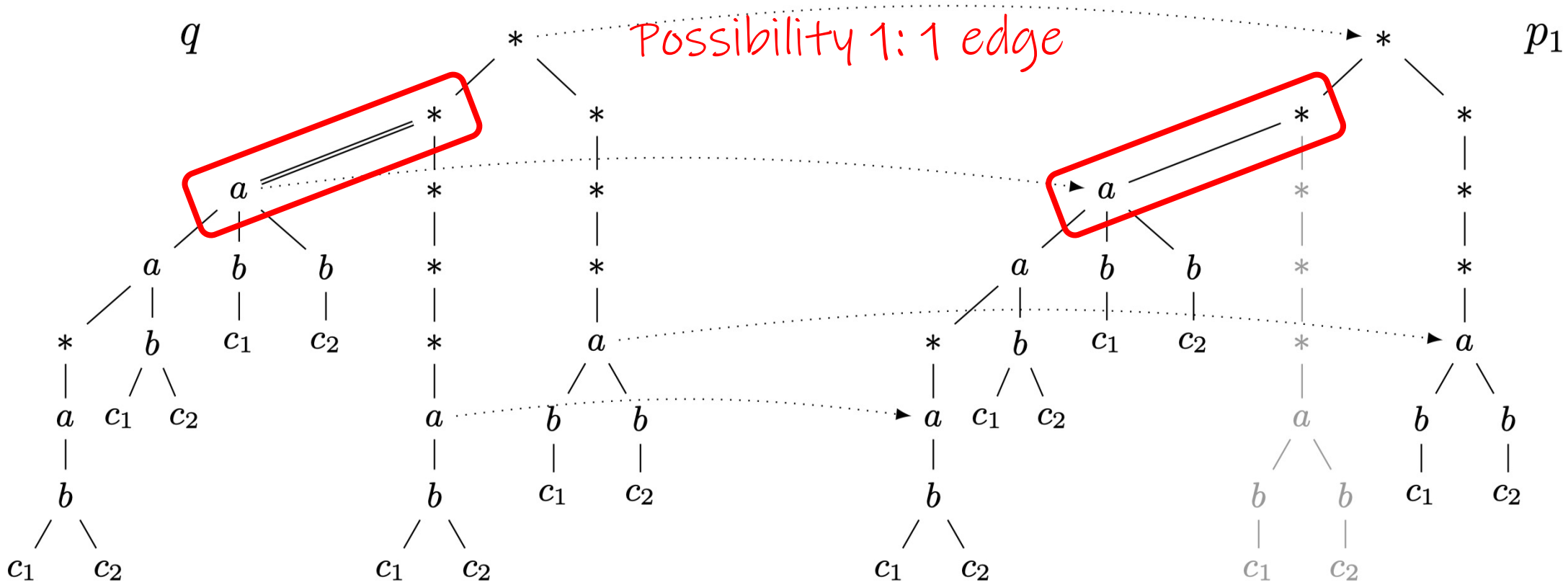
Figure 7: A non-redundant tree pattern p (right) and an equivalent tree pattern q that is smaller (left)

$q \subseteq p$ from previous argument. To be shown: $q \supseteq p$

To be shown $q \supseteq p$: (idea: whenever p matches, then also q)

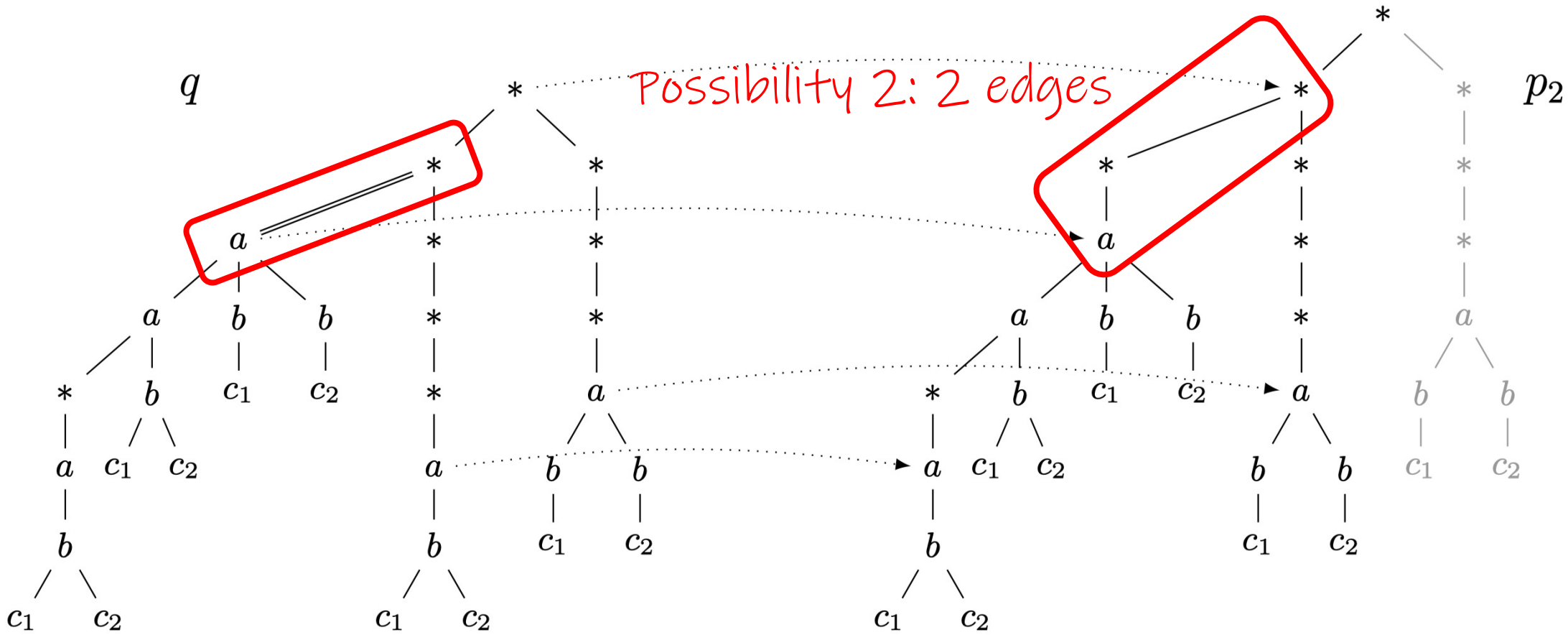
Idea: $a = \star$ can be matched in 3 ways in a graph

Example from: "Optimizing Tree Patterns for Querying Graph- and Tree-Structured Data" by Czerwinski, Martens, Niewerth, Parys. SIGMOD record 2017. https://doi.org/10.1145/3093754.3093759



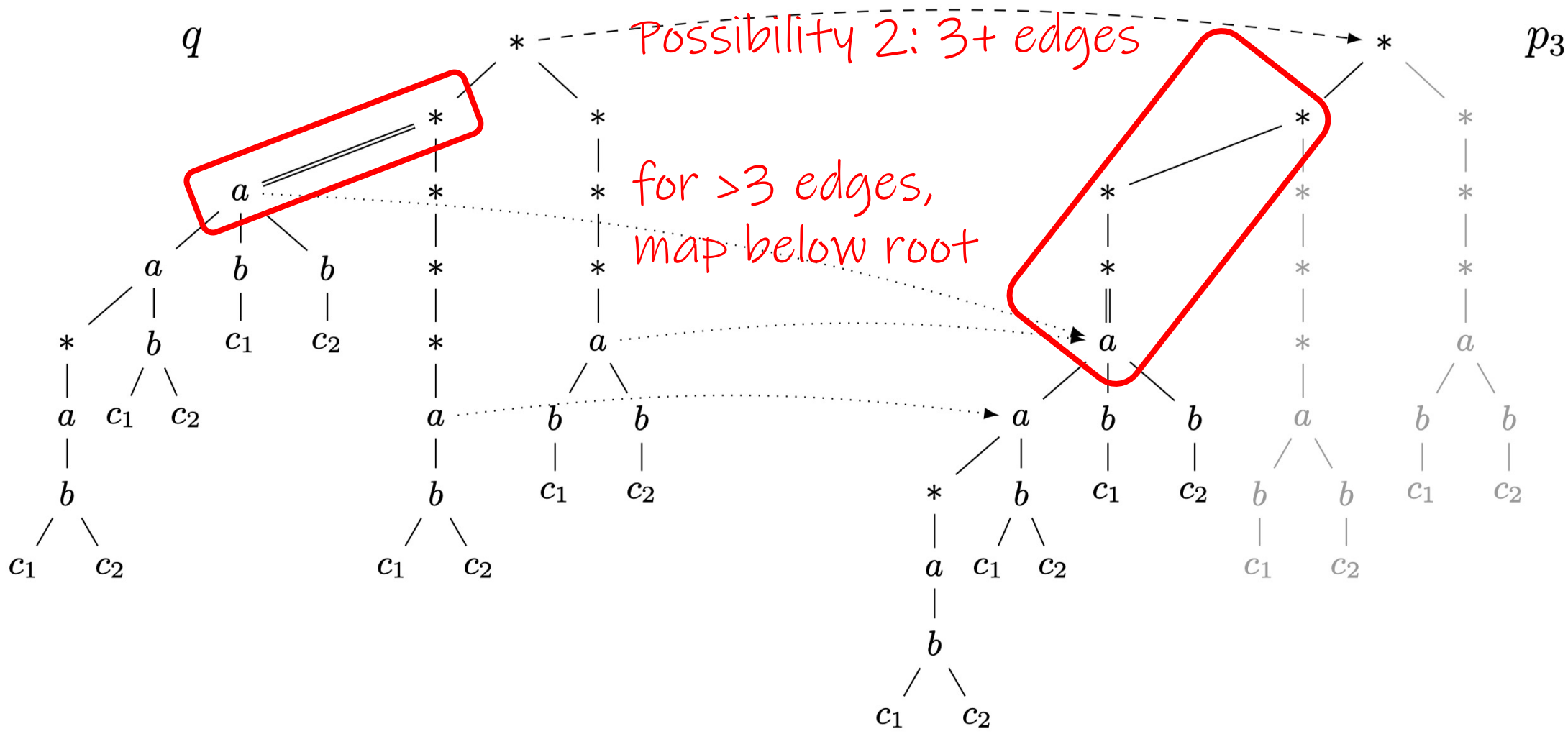
(a) How q can be matched if p_1 can be matched

Example from: "Optimizing Tree Patterns for Querying Graph- and Tree-Structured Data" by Czerwinski, Martens, Niewerth, Parys. SIGMOD record 2017. https://doi.org/10.1145/3093754.3093759



(b) How q can be matched if p_2 can be matched

Example from: "Optimizing Tree Patterns for Querying Graph- and Tree-Structured Data" by Czerwinski, Martens, Niewerth, Parys. SIGMOD record 2017. https://doi.org/10.1145/3093754.3093759



(c) How q can be matched if p_3 can be matched

Example from: "Optimizing Tree Patterns for Querying Graph- and Tree-Structured Data" by Czerwinski, Martens, Niewerth, Parys. SIGMOD record 2017. https://doi.org/10.1145/3093754.3093759