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

Pre-class conversations

- Recapitulation of Datalog & Query Equivalence
- Suggestion: Scribes with 2 iterations

- today:
  - Query equivalence of CQs & homomorphisms

Topic 1: Data Models and Query Languages

- **Lecture 1 (Tue 1/18):** Course introduction / 1 SQL / PostgreSQL setup / SQL Activities
- **Lecture 2 (Fri 1/21):** 1 SQL
- **Lecture 3 (Tue 1/25):** 1 SQL
- **Lecture 4 (Fri 1/28):** 1 SQL, 2 Logic & Relational Calculus
- **Lecture 5 (Tue 2/1):** 2 Logic & Relational Calculus
- **Lecture 6 (Fri 2/4):** 3 Relational Algebra & Codd's Theorem
- **Lecture 7 (Tue 2/8):** 3 Relational Algebra & Codd's Theorem
- **Lecture 8 (Fri 2/11):** 3 Relational Algebra & Codd's Theorem / 4 Datalog & Recursion
- **Lecture 9 (Tue 2/15):** 4 Datalog & Recursion
- **Lecture 10 (Tue 2/18):** 4 Datalog & Recursion
- **Lecture 11 (Tue 2/22):** 4 Datalog & Recursion

  Pointers to relevant concepts & supplementary material:

  - **Unit 1. SQL**: [SAMS'12], [CS 3200], [Cow'03] Ch3 & Ch5, [Complete'08] Ch6, [Silberschatz+'20] Ch3.8
  - **Unit 2. Logic & Relational Calculus**: First-Order Logic (FOL), relational calculus (RC): [Barland+'08] 4.1.2 & 4.2.1 & 4.4, [Genesereth+] Ch6, [Halpern+'01], [Cow'03] Ch4.3 & 4.4, [Elmasri, Navathe'15] Ch8.6 & Ch8.7, [Silberschatz+'20] Ch27.1 & Ch27.2, [Alice'95] Ch3.1-3.3 & Ch4.2 & Ch4.4 & Ch5.3-5.4, [Barker-Plummer+'11] Ch11
  - **Unit 3. Relational Algebra & Codd's Theorem**: Relational Algebra (RA), Codd's theorem: [Cow'03] Ch4.2, [Complete'08] Ch2.4 & Ch5.1-5.2, [Elmasri, Navathe'15] Ch8, [Silberschatz+'20] Ch2.6, [Alice'95] Ch4.4 & Ch5.4
  - **Unit 4. Datalog & Recursion**: Datalog, recursion, Stratified Datalog with negation, Datalog evaluation strategies, Stable Model semantics, Answer Set Programming (ASP): [Complete'08] Ch5.3, [Cow'03] Ch 24, [Koutris'19] L9 & L10, [G., Suciu'10]
  - **Unit 5. Alternative Data Models**: NoSQL: [Hellerstein, Stonebraker'05], [Sadalage, Fowler'12], [Harrison'16]

Topic 2: Complexity of Query Evaluation & Reverse Data Management

- **CONTINUED Lecture 11 (Tue 2/22):** 1 Conjunctive Queries
- **Lecture 12 (Fri 2/25):** Conjunctive Queries
- **Lecture 13 (Tue 3/1):** Beyond Conjunctive Queries
- **Lecture 14 (Fri 3/4):** Provenance
- **Lecture 15 (Tue 3/8):** Provenance, Reverse Data Management

  Pointers to relevant concepts & supplementary material:

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

Topic 3: Efficient Query Evaluation & Factorized Representations

- **Lecture 16 (Fri 3/11):** Acyclic Queries
- Spring break
- **Lecture 17 (Tue 3/22):** Acyclic Queries
- **Lecture 18 (Fri 3/25):** Cyclic Queries
- **Lecture 19 (Tue 3/29):** Cyclic Queries
- **Lecture 20 (Fri 4/1):** Factorized Representations
- **Lecture 21 (Tue 4/5):** Factorized Representations
- **Lecture 22 (Fri 4/8):** Top-k & Optimization Problems
- **Lecture 23 (Tue 4/12):** Top-k & Optimization Problems

  Pointers to relevant concepts & supplementary material:

  - **Unit 1. Acyclic Queries**: query hypergraph, Yannakakis algorithm, GYO reduction, dynamic programming, algebraic semirings, [Alice] Ch6.4, [Koutris'19] L4, enumeration, ranked enumeration:[Tziavelis+'20]
  - **Unit 2. Cyclic Queries**: tree & hypertree decomposition, query widths, fractional hypertree width, AGM bound, worst-case optimal join algorithms, optimal algorithms, submodular width and multiple decompositions: [AGM'13], [NPRR'18], [KNR'17], [KNS'17]
  - **Unit 3. Factorized Representations**: normalization, factorized databases [Olteanu, Schleich'16]
  - **Unit 4. Top-k & Optimization Problems**: shortest paths, dynamic programming (DP), Yannakakis, semirings, rankings, top-k: [Roughgarden'10], [Ilyas+08], [Rahul, Tao'19], ranked enumeration [Tziavelis+'19]

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

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

Complexity of the Query Evaluation Problem

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

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

  ... PSPACE-complete.

  - PSPACE: decision problems, can be solved using an amount of memory that is polynomial in the input length (~ in polynomial amount of space).
  - PSPACE-complete: PSPACE + every other PSPACE problem can be transformed to it in polynomial time (PSPACE-hard)
- Proof: We need to show both
  - This problem is in PSPACE.
  - This problem is PSPACE-hard. (We only focus on this task for Boolean RC queries)

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

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/

Conjunctive Queries (CQs)

- 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)

Examples of Conjunctive Queries

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

- Return paths of Length 2: (binary output)

  RC: ?

  RA: ?

  Datalog: ?

Examples of Conjunctive Queries

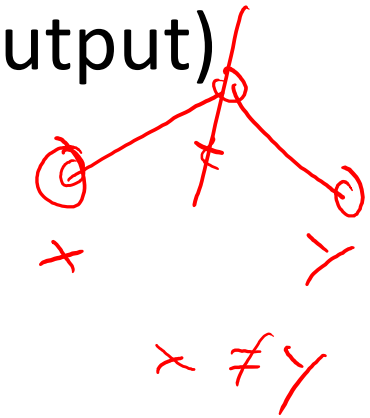
$E(\text{from}, \text{to})$

- Return paths of Length 2: (binary output)

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

RA: ?

Datalog: ?



E

1	2
2	1

Is there a path of length 2 ?

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

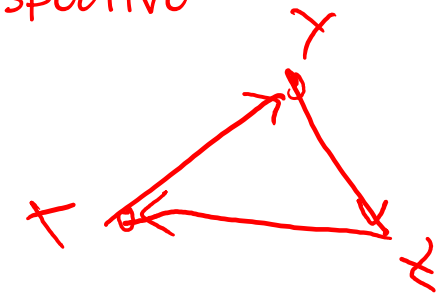
$E(\text{from}, \text{to})$

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



- Is there a 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: $\{(x, y) \mid \exists z[E(x, z) \wedge E(z, y)]\}$

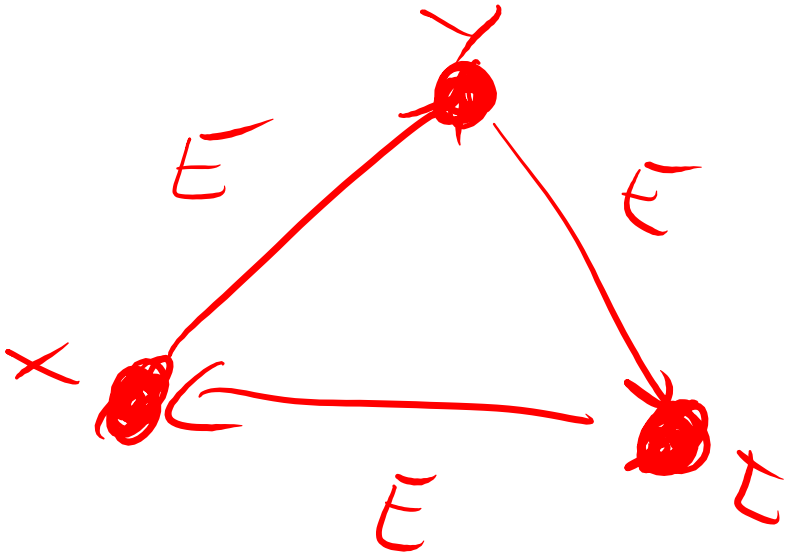
  RA: $\pi_{\$1,\$4}(\sigma_{\$2=\$3}(E \times E))$ unnamed perspective

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

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

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

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

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

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

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

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

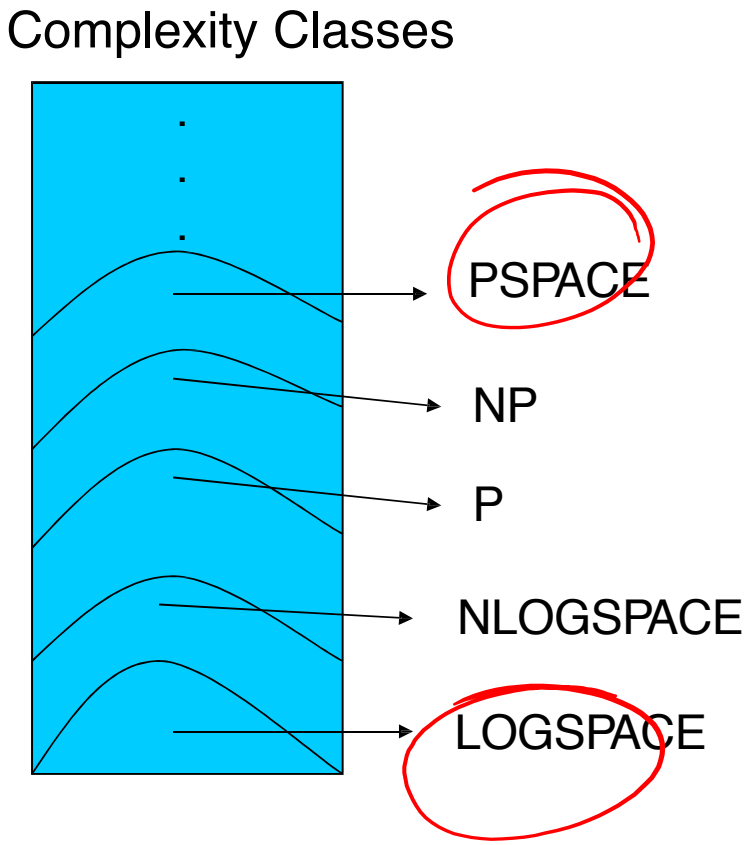
Vardi's Taxonomy of the Query Evaluation Problem

Vardi's "empirical" discovery:

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

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

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

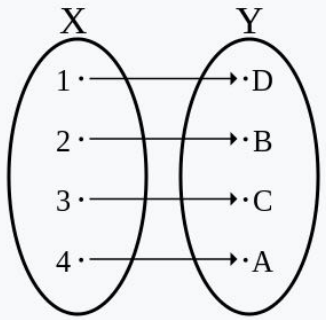
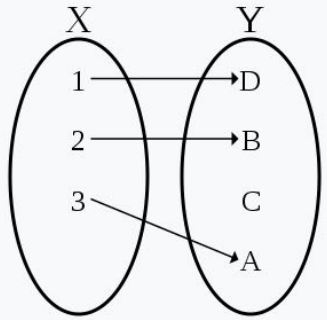
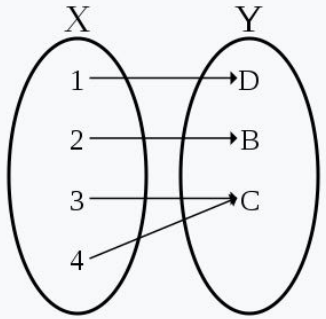
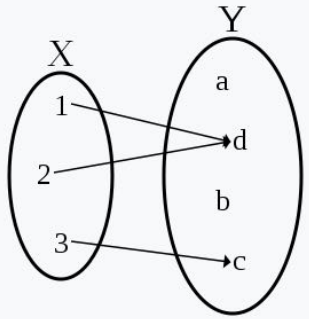
- The Query Evaluation Problem for Relational Calculus:
  - Data Complexity is in LOGSPACE
  - Combined Complexity is PSPACE-complete
  - Query Complexity is PSPACE-complete.

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
  - Tree pattern queries
  - Nested queries

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

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

Function ?

Injective function ?

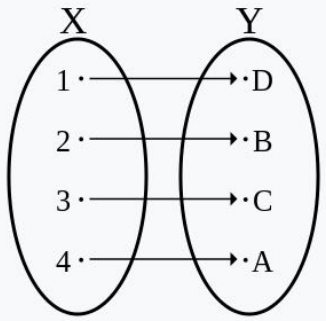
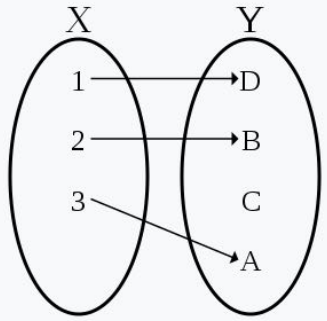
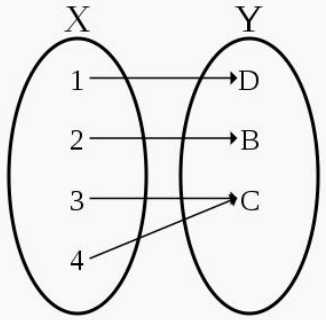
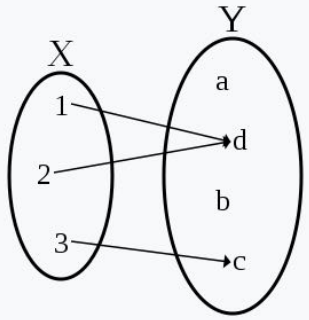
Surjective function ?

Bijective function ?

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

Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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)

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

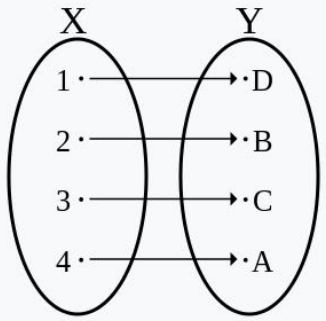
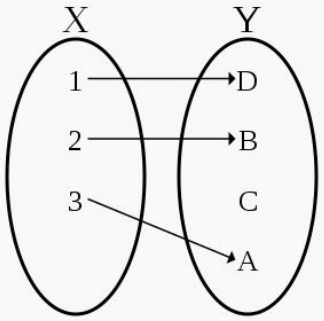
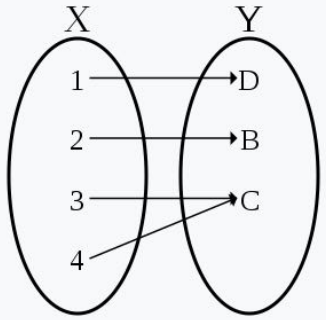
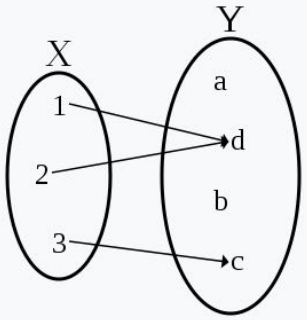
Injective function ?

Surjective function ?

Bijective function ?

Source: 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)

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

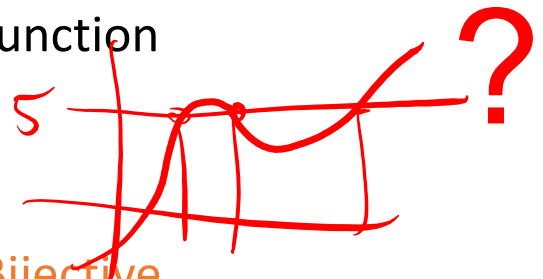
Injective function ("one-to-one"): each element of the codomain is mapped to by at most one element of the domain (i.e. distinct elements of the domain map to distinct elements in the codomain)

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

logical transpose without inequality:

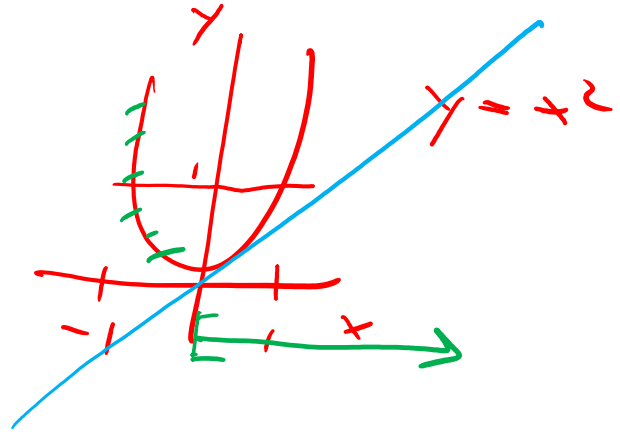
$$\forall x, x' \in X. [f(x) = f(x') \Rightarrow x = x']$$

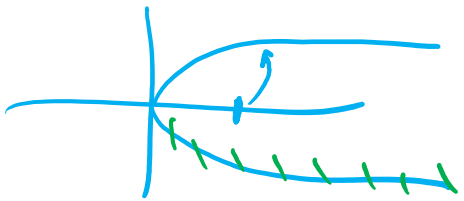
Surjective function



Bijective function

?

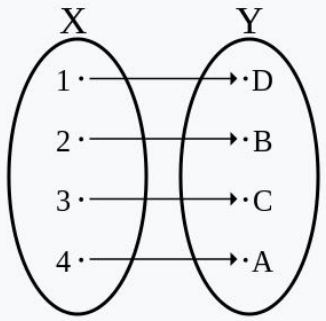
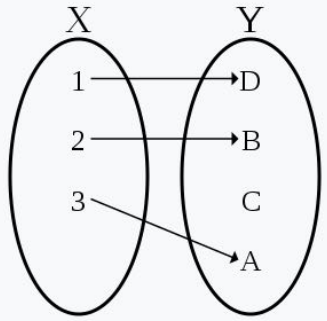
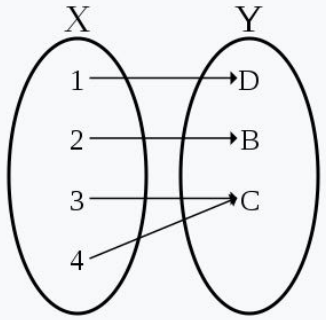
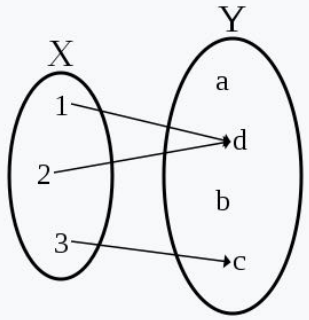




Source: https://en.wikipedia.org/wiki/Bijection,_injection_and_surjection

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)

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

logical transpose without inequality:

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

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)

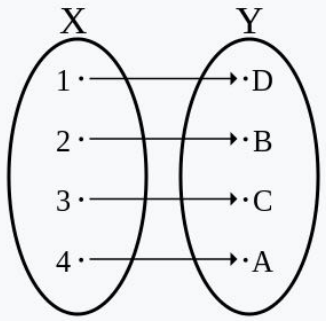
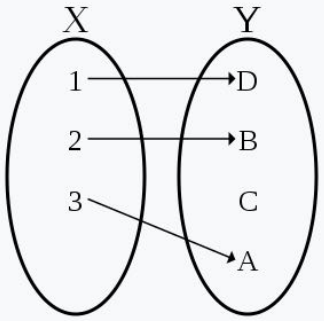
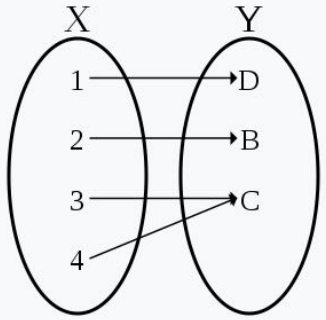
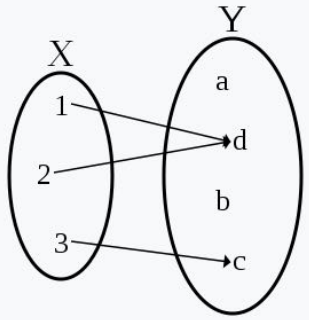
$$\forall y \in Y, \exists x \in X[y = f(x)]$$

Bijective function

?

Source: 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)

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

logical transpose without inequality:

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

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)

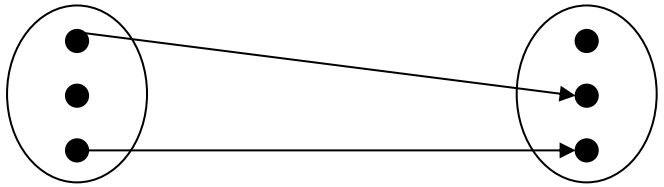
$$\forall y \in Y, \exists x \in X[y = f(x)]$$

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

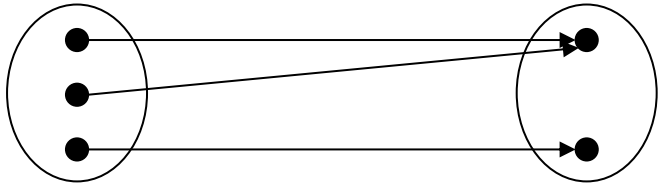
$$\forall y \in Y, \exists! \, x \in X[y = f(x)]\}$$

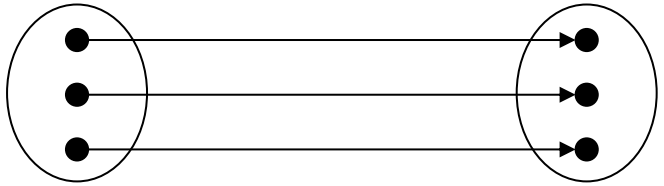
Source: https://en.wikipedia.org/wiki/Bijection,_injection_and_surjection

Mappings: Injection, Surjection, and Bijection

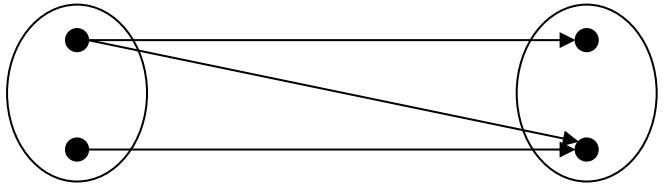

?


?

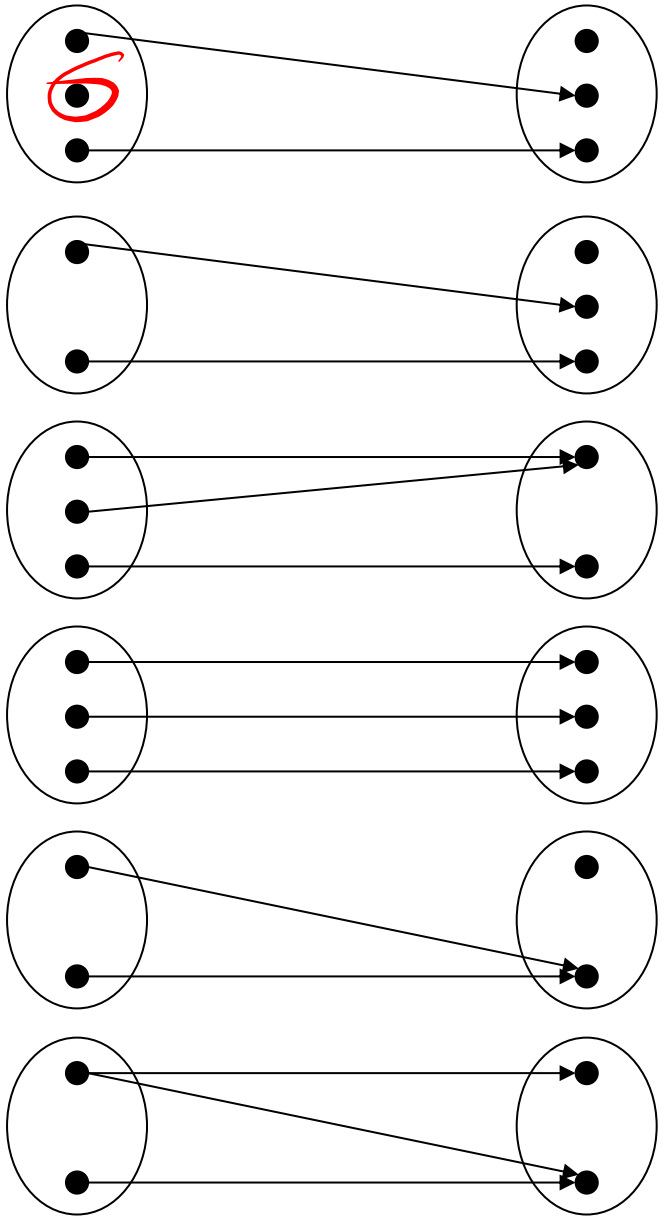

?


?


?


?

Mappings: Injection, Surjection, and Bijection



not a mapping (or function)!

?

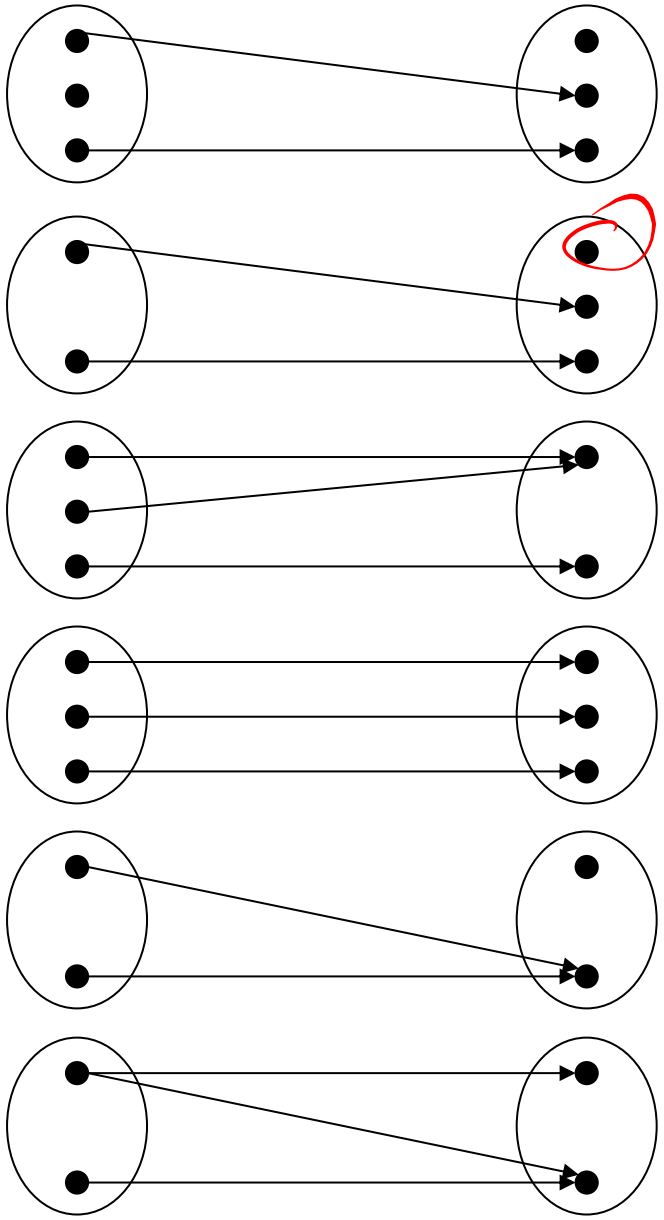
?

?

?

?

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

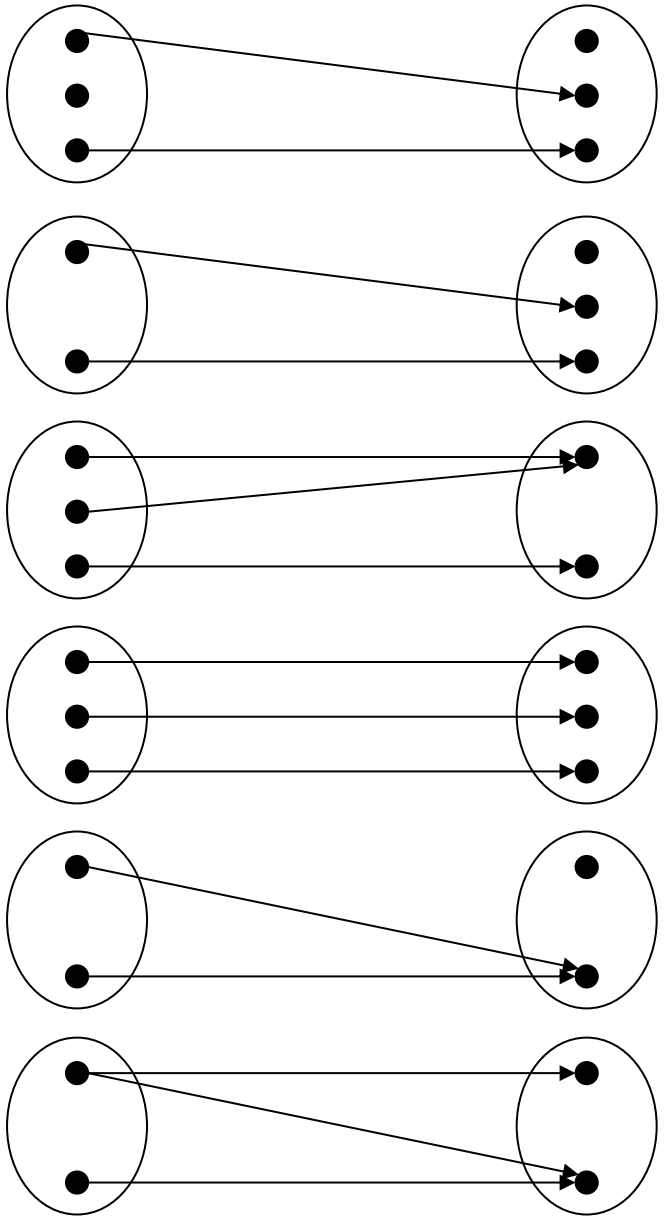
?

?

?

?

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

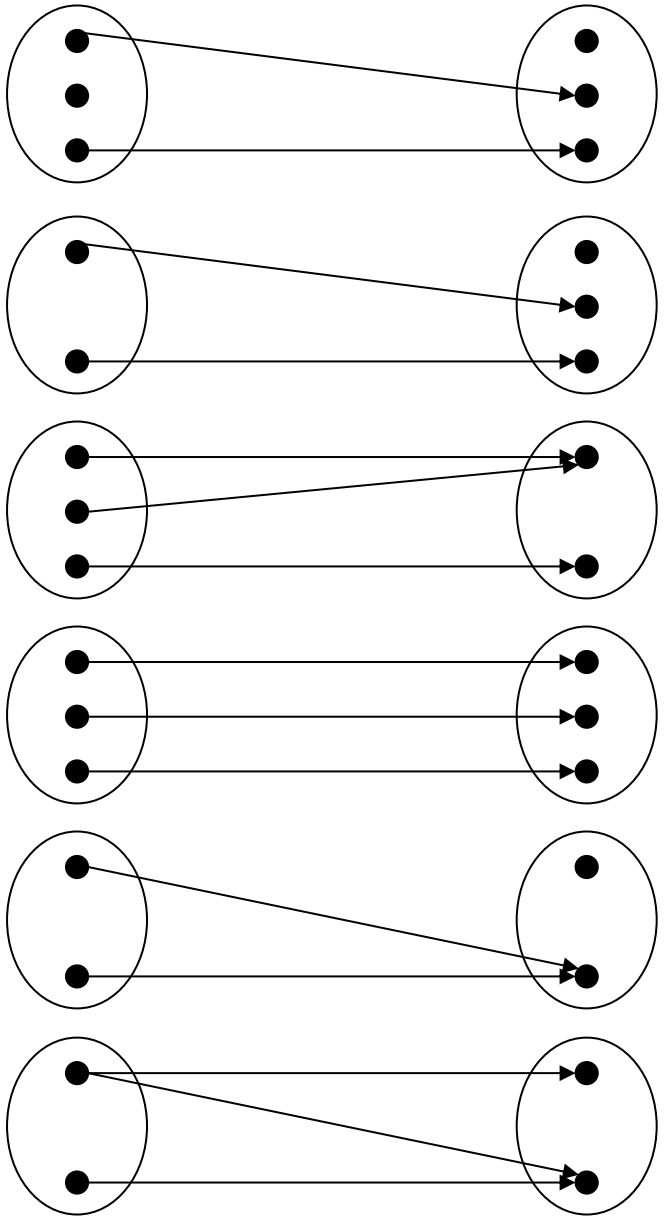
surjective (or onto): every element y in the codomain Y of f has at least one element x in the domain that maps to it

?

?

?

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

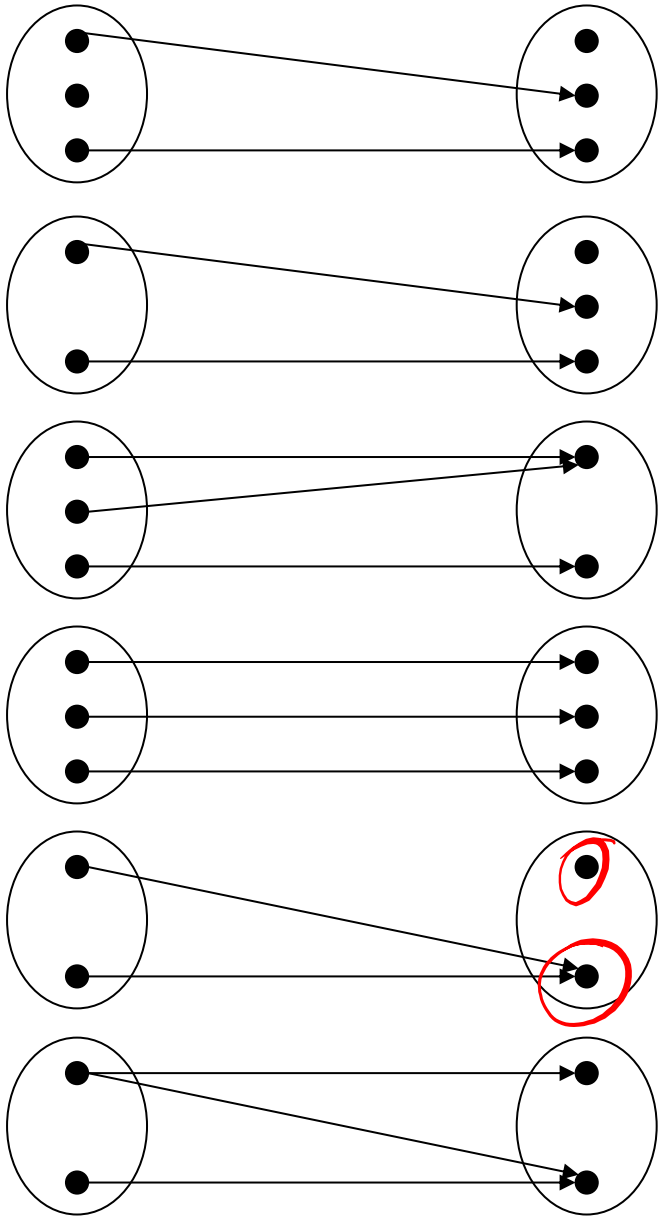
surjective (or onto): every element y in the codomain Y of f has at least one element x in the domain that maps to it

injective & surjective = bijection

?

?

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

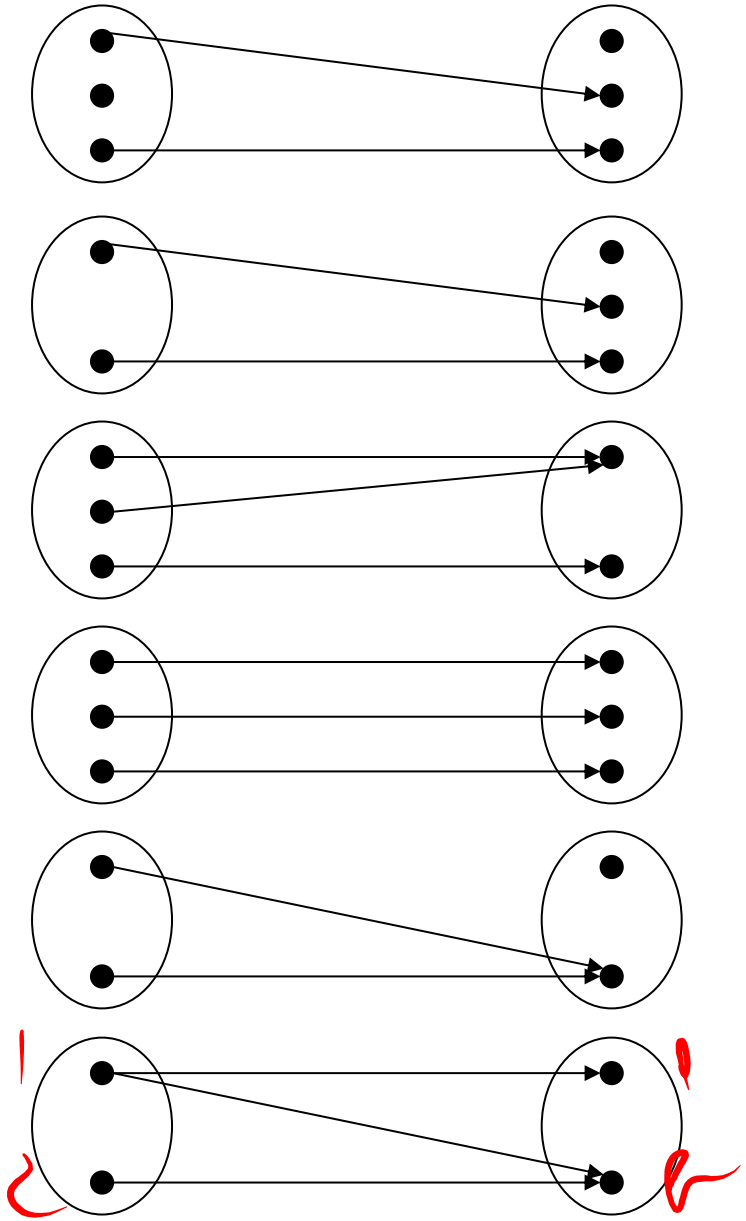
surjective (or onto): every element y in the codomain Y of f has at least one element x in the domain that maps to it

injective & surjective = bijection

neighter

?

Mappings: Injection, Surjection, and Bijection



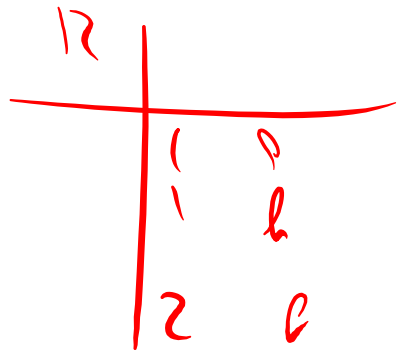
not a mapping (or function)!

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

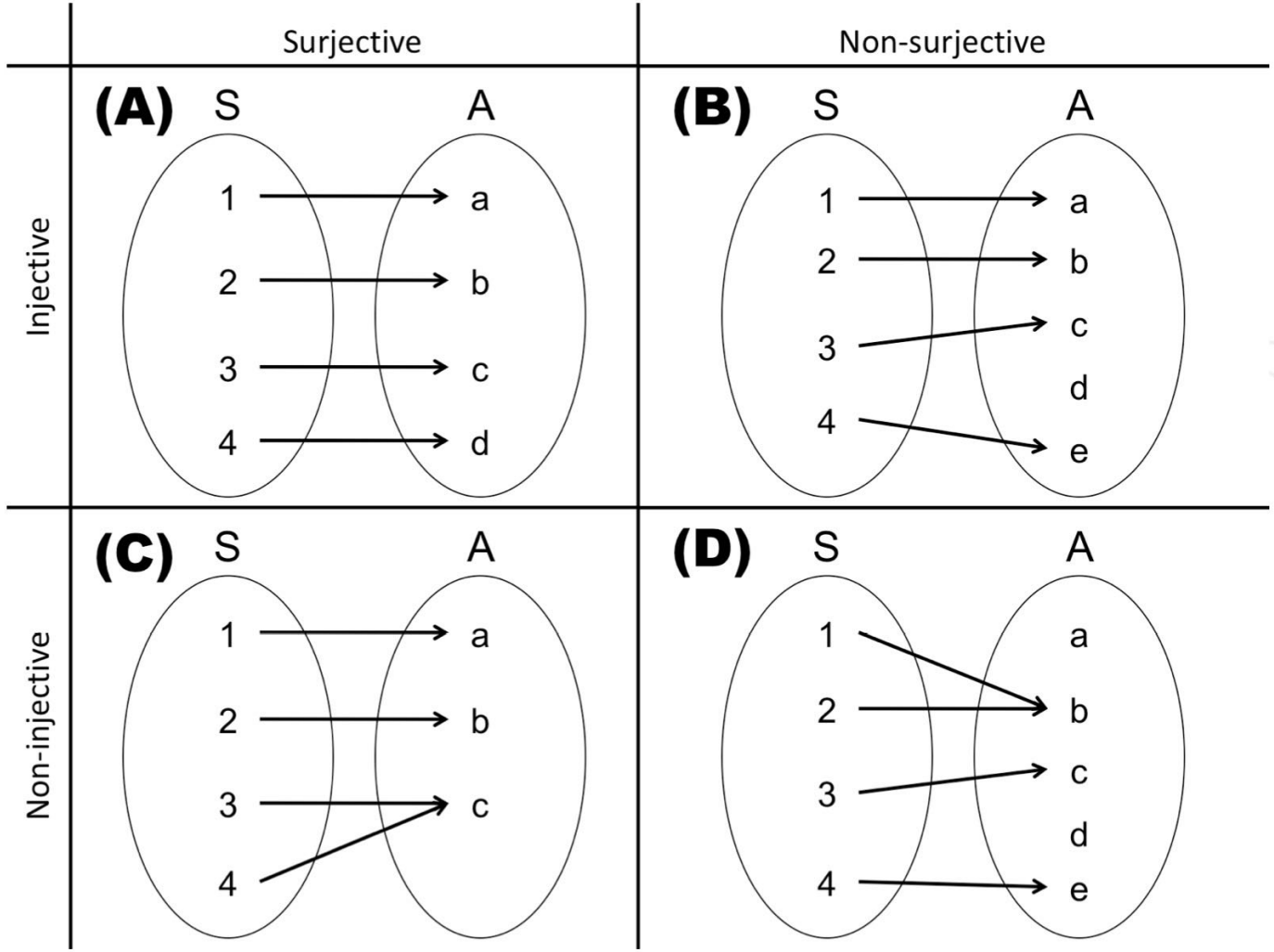
injective & surjective = bijection

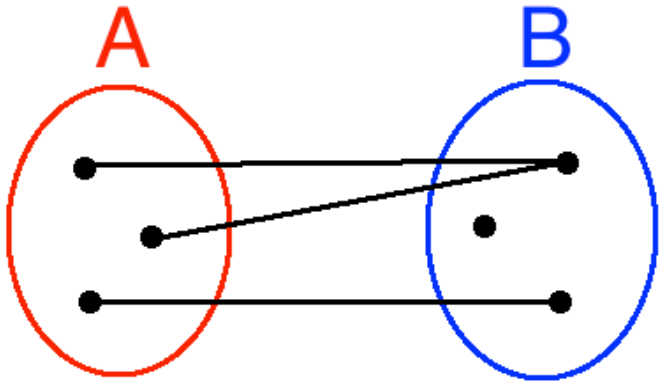
neighter



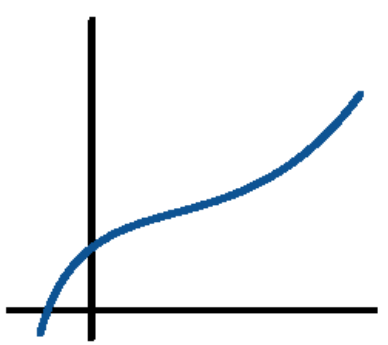
not even a mapping!

Bijection, Injection, and Surjection

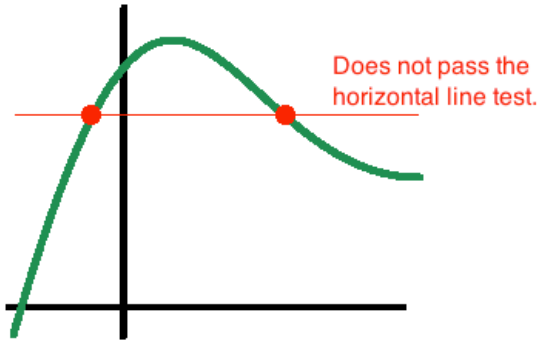




Neither Injective or Surjective
Two elements in set A maps to the same element in set B (not injective), and one element in set B is not in the image or range of the function that maps set A to B (not surjective).



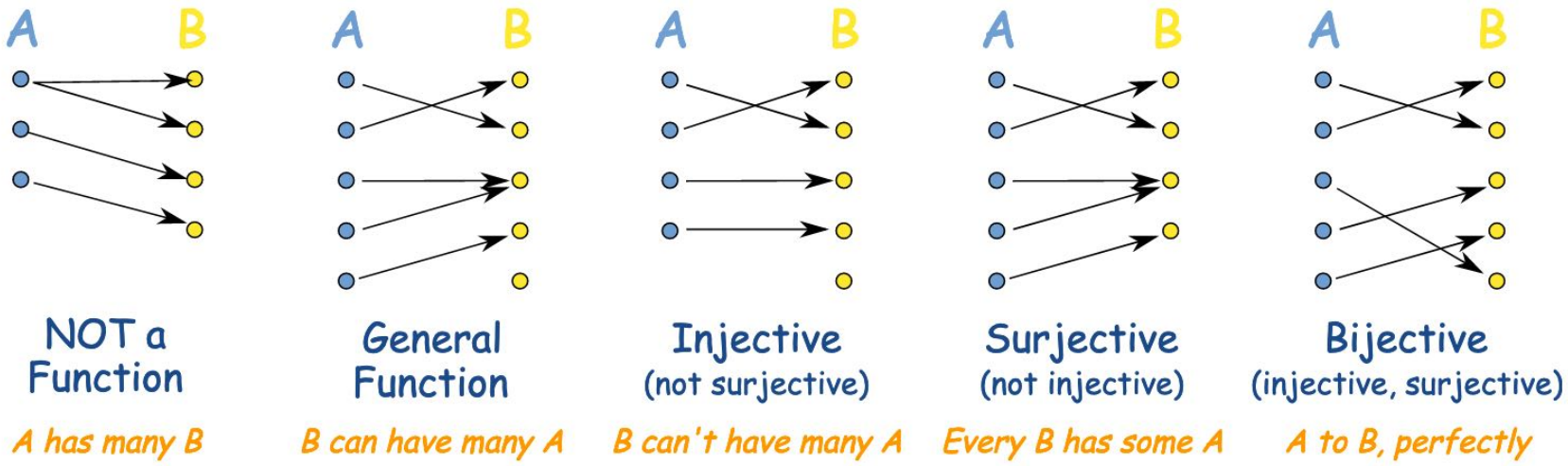
Injective (One-to-one)

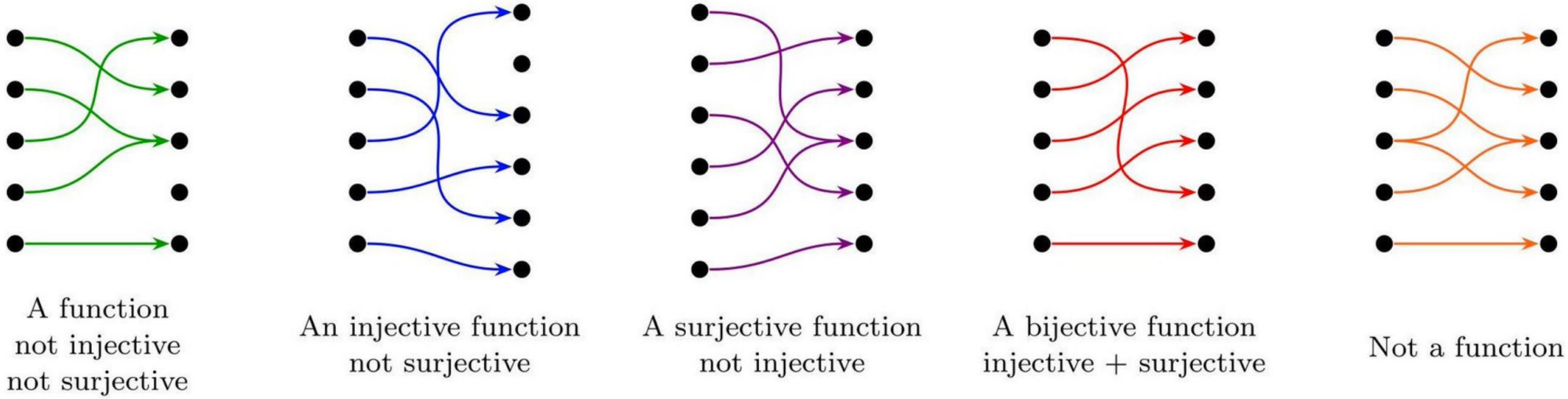


Not Injective

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

Bijection, Injection, and Surjection





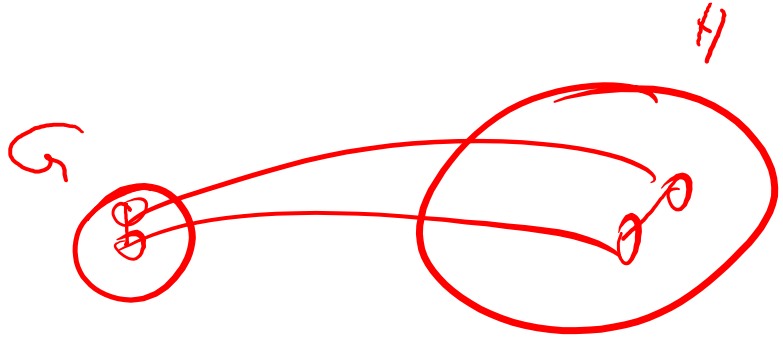
Sources: https://www.mathsisfun.com/sets/injective-surjective-bijective.html, https://twitter.com/jdhamkins/status/841318019397779456, Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

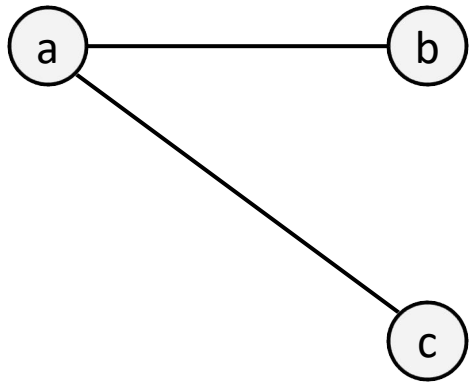
We make a detour to Graph matching

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

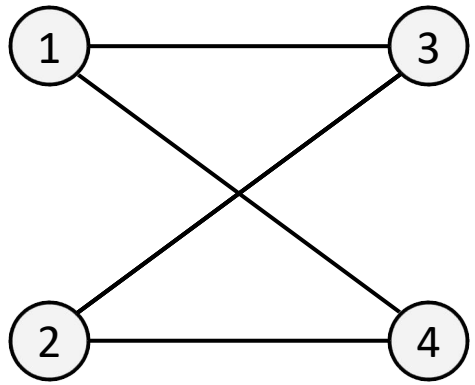
Homomorphism

- 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

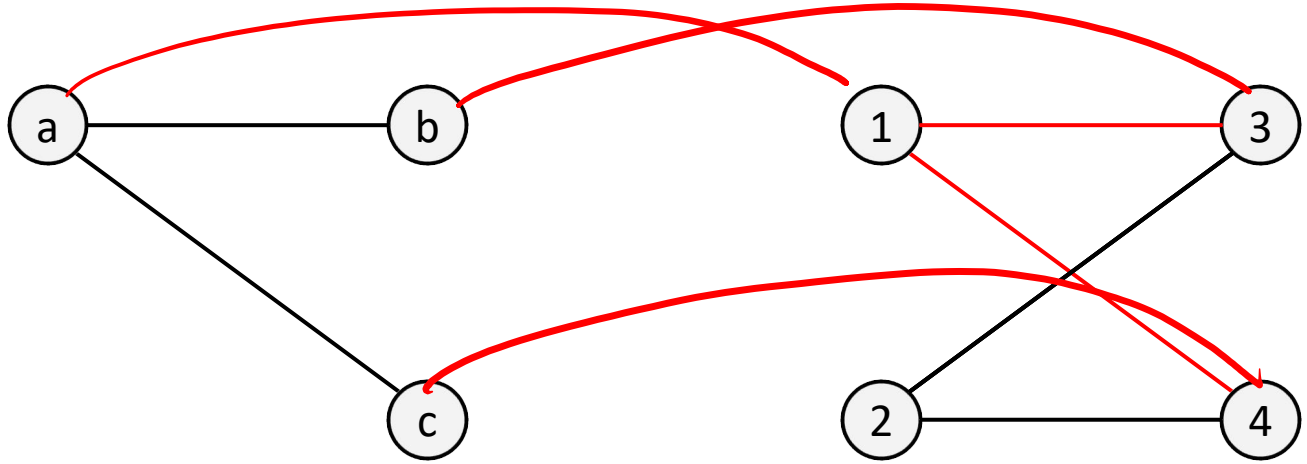


H

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



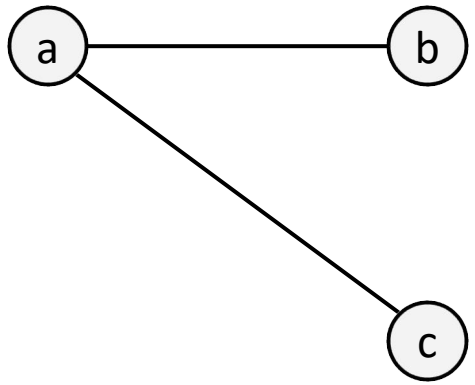
G H

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

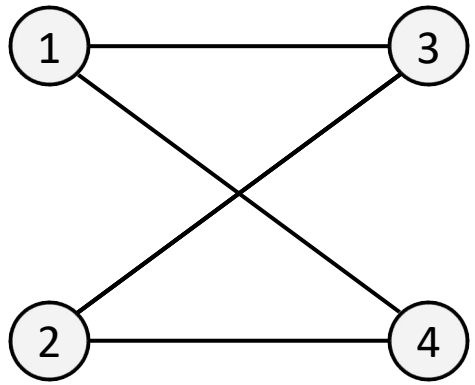
does not need to be surjective!

Homomorphism

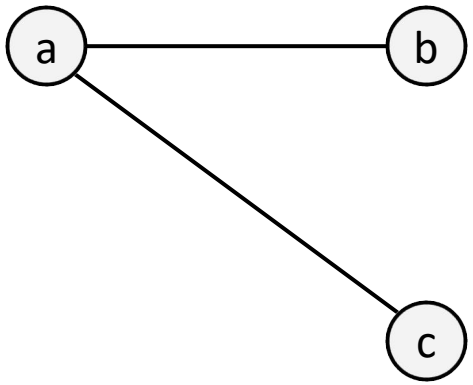
- A graph homomorphism h from graph $G(V_G,E_G)$ to $H(V_H,E_H)$, is a mapping from V_G to V_H such that $\{x,y\} \in E_G$ implies $\{h(x),h(y)\} \in E_H$
  - "edge-preserving": if two nodes in G are linked by an edge, then they are mapped to two nodes in H that are also linked



G



H



G

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

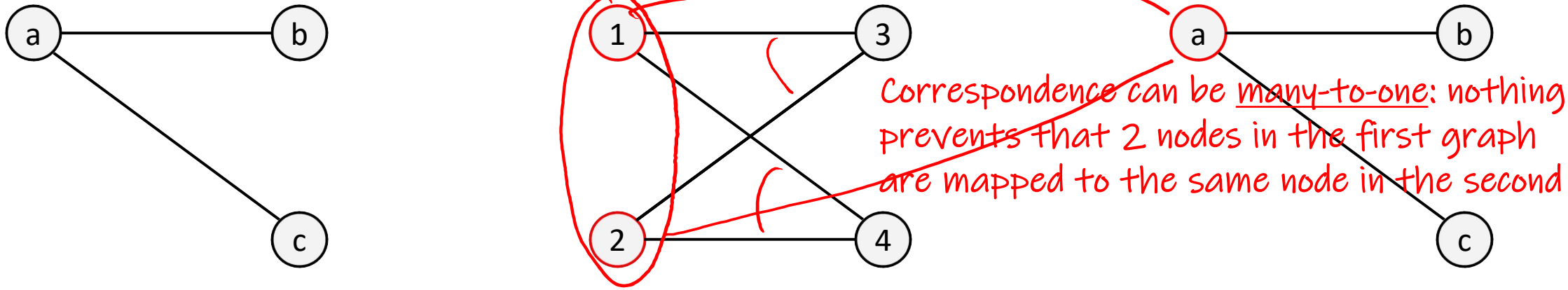
does not need to be surjective!

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)}

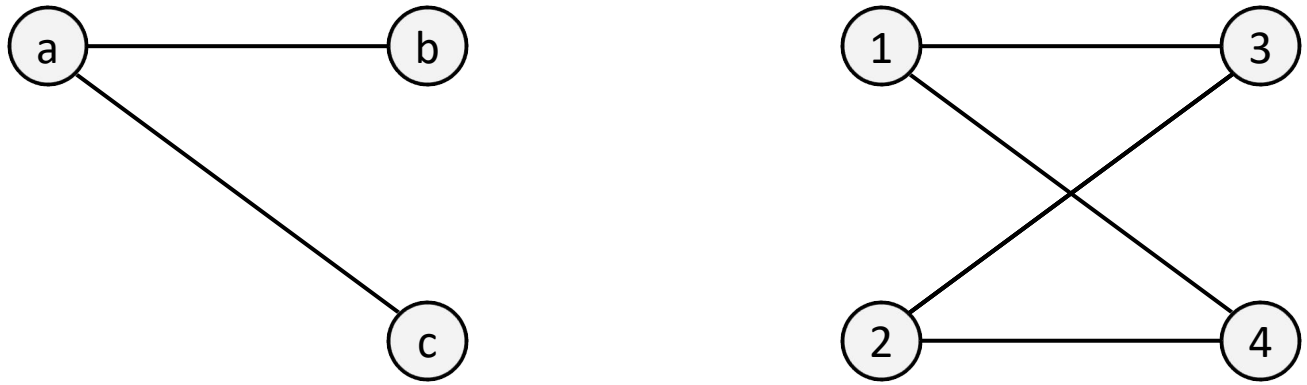
does not need to be surjective!

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

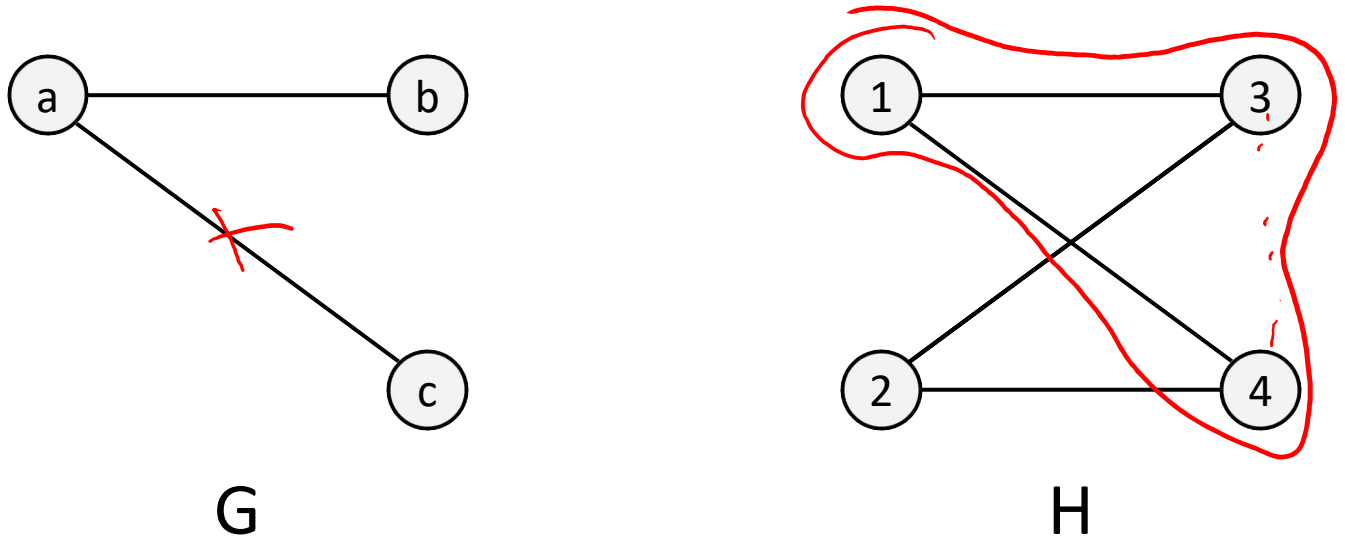


G H

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 $\{h(u),h(v)\} \in E_H$
  - We need to find a one-to-one correspondence



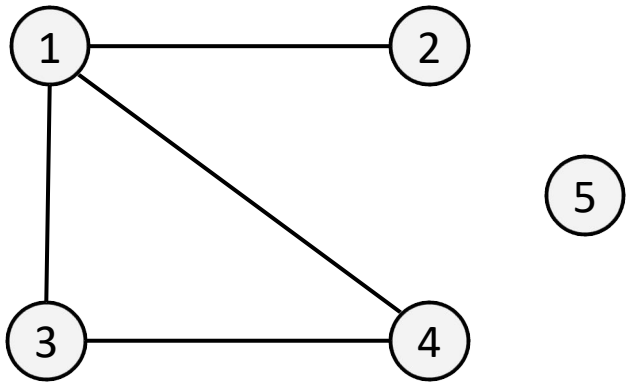
G H

Is there an isomorphism from G to H?

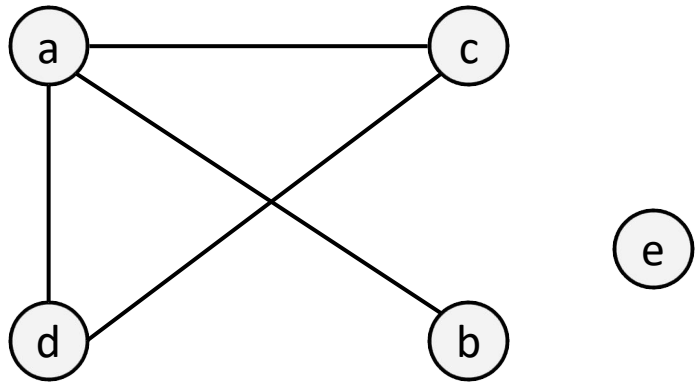
They are homomorphically equivalent, but not isomorphic!

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 $\{h(u),h(v)\} \in E_H$
  - We need to find a one-to-one correspondence



G

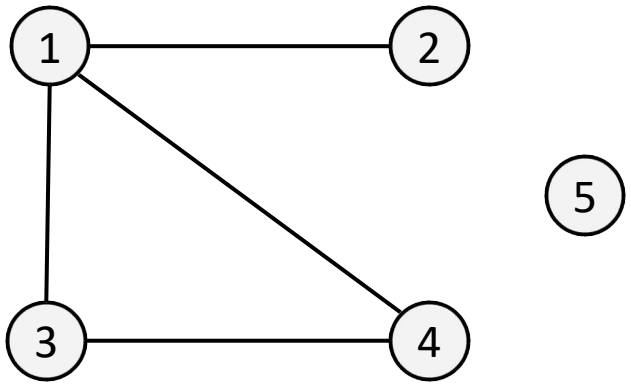


H

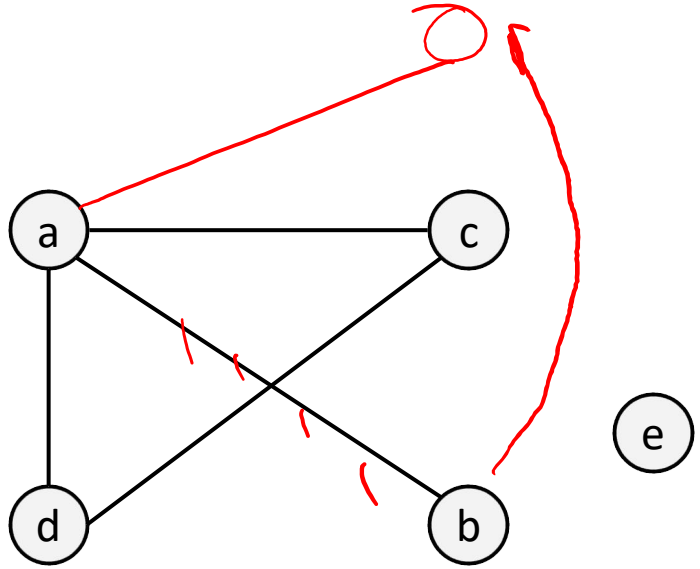
Is there an isomorphism from G to H?

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 $\{h(u),h(v)\} \in E_H$
  - We need to find a one-to-one correspondence



G



H

Is there an isomorphism from G to H?

Yes:

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

bijection = surjective and injective mapping

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

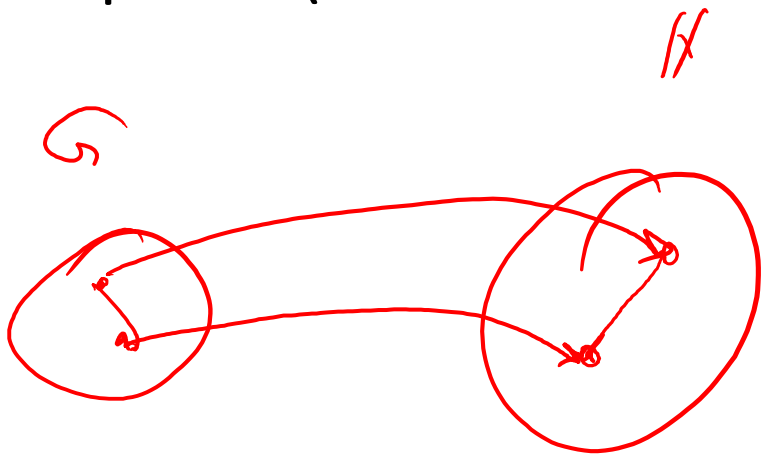
- T2-1: Conjunctive Queries (CQs)
  - CQ equivalence and containment
  - Graph homomorphisms
  - Homomorphism beyond graphs
  - CQ containment
  - CQ minimization
- T2-2: Equivalence Beyond CQs
  - Union of CQs, and inequalities
  - Union of CQs equivalence under bag semantics
  - Tree pattern queries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

Graph Homomorphism beyond graphs

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

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

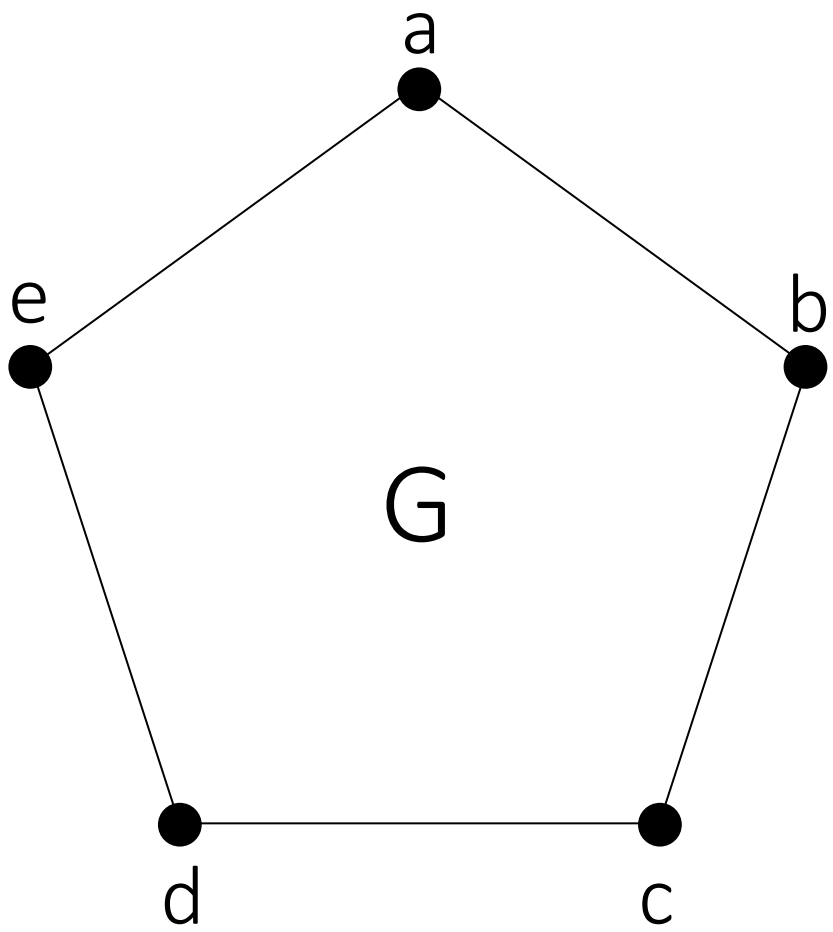
We sometimes write G → H (G ↛ H) if there is a homomorphism (no homomorphism) of G to H



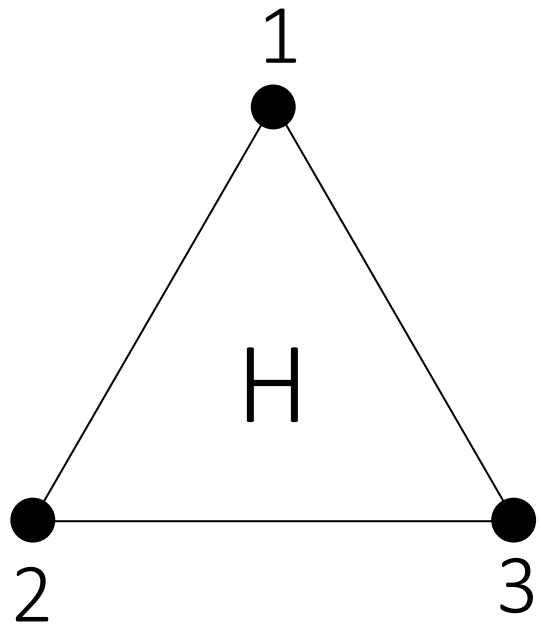
Definition of a homomorphism naturally extends to:

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

An example

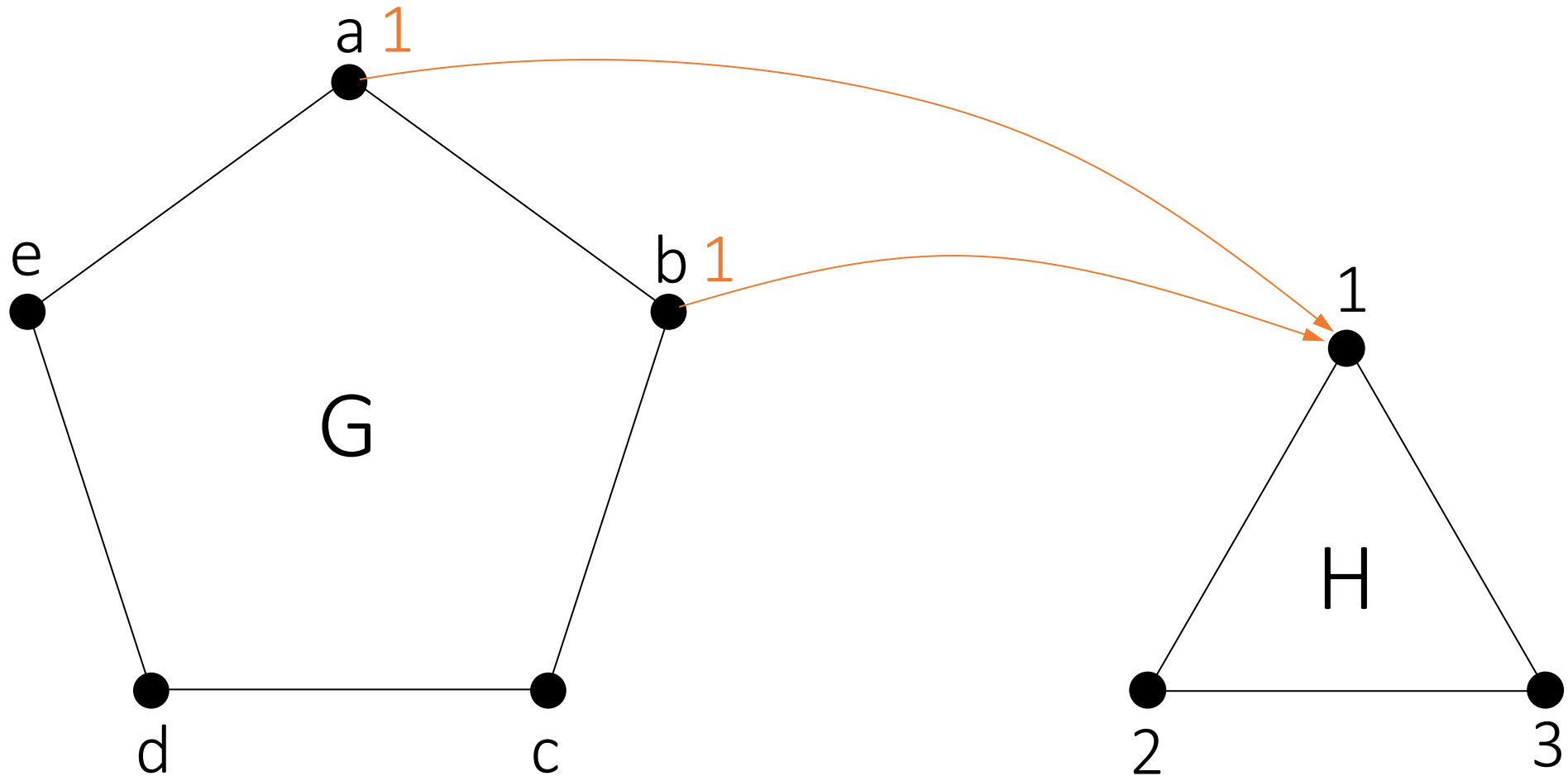


3 "colors" of the vertices



Based upon an example from Rick Brewster's Graph homomorphism tutorial, 2006

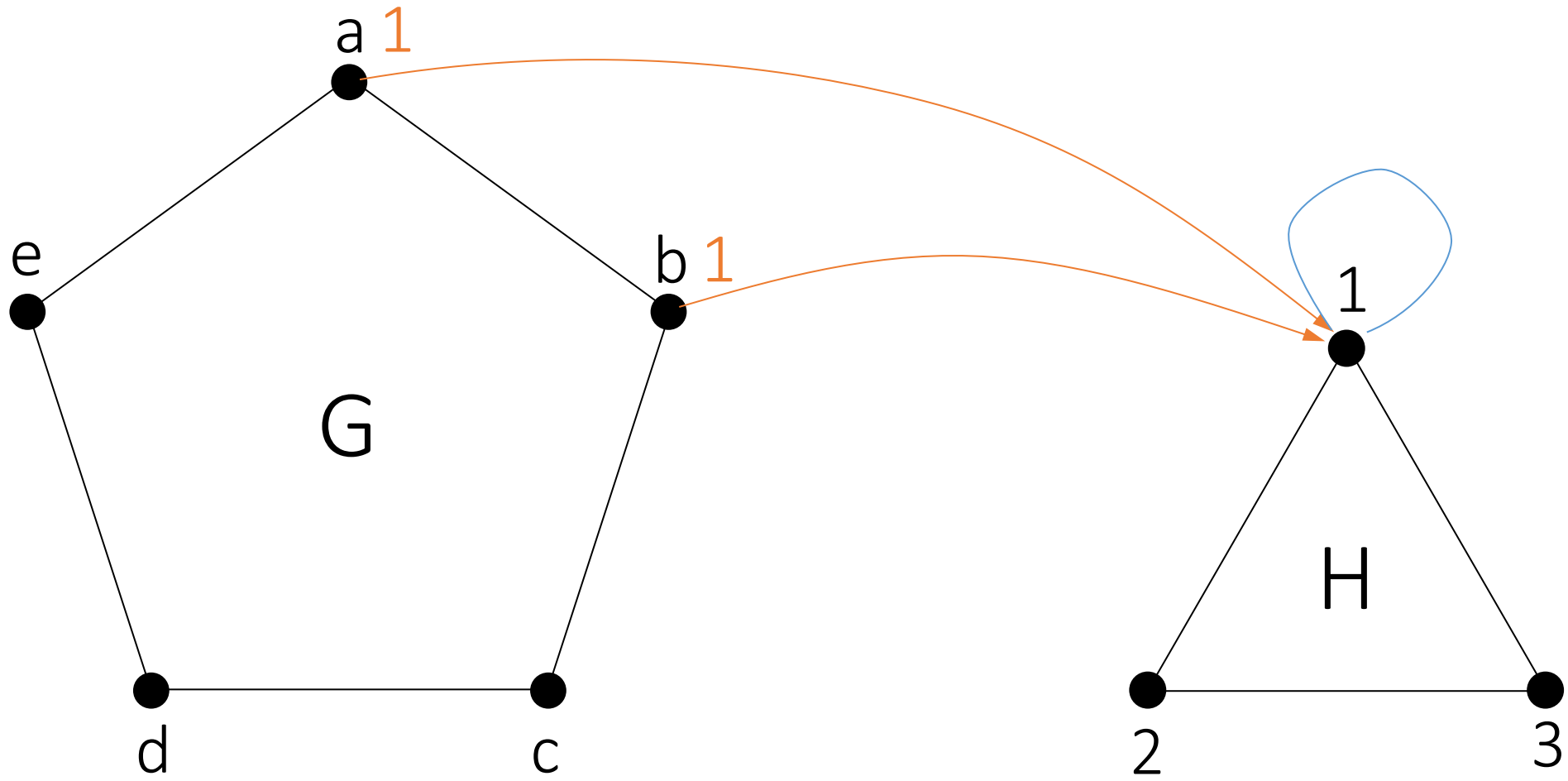
An example



Can this assignment be extended to a homomorphism? ?

Based upon an example from Rick Brewster's Graph homomorphism tutorial, 2006

An example

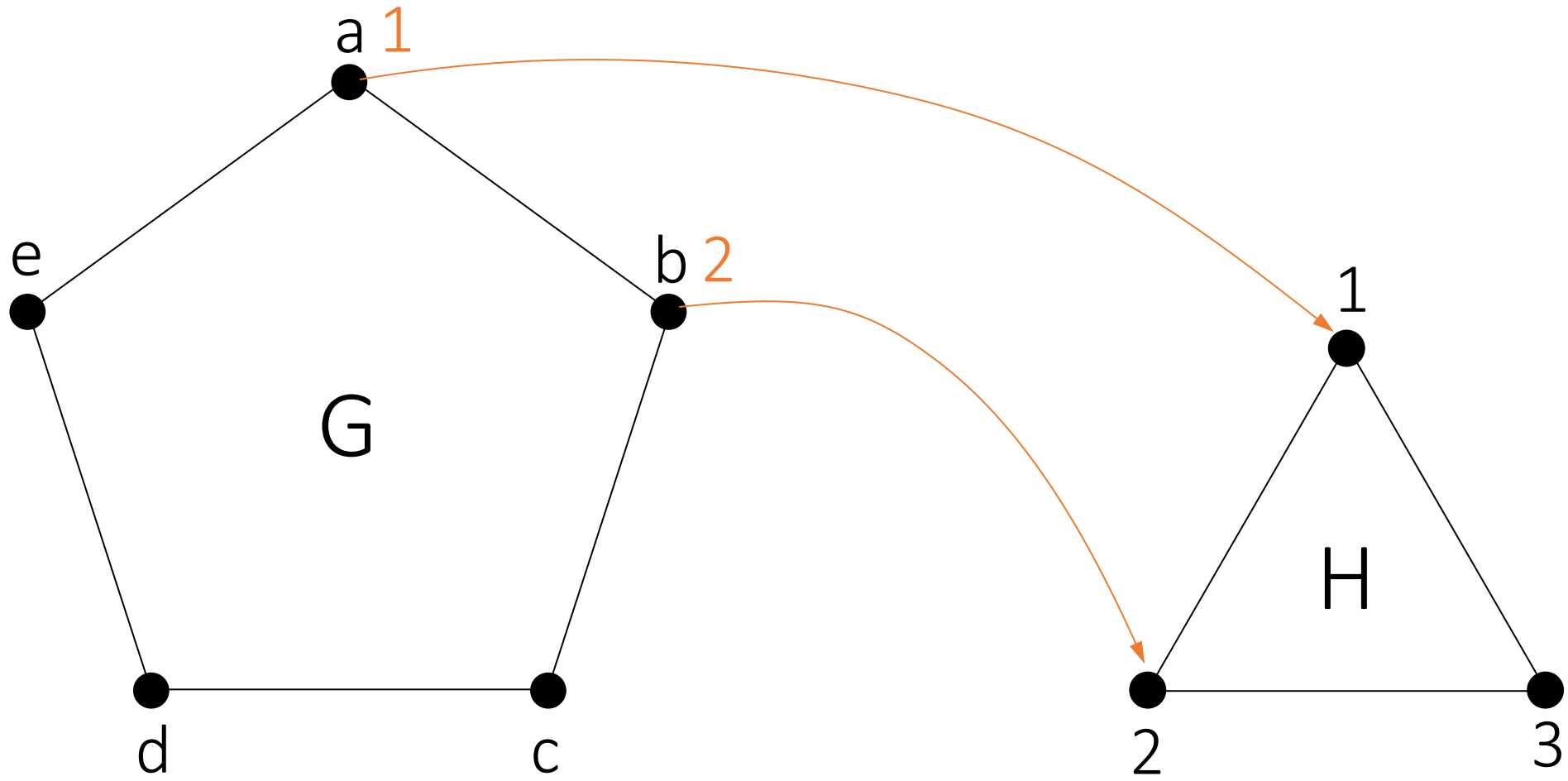


Can this assignment be extended to a homomorphism?

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

Based upon an example from Rick Brewster's Graph homomorphism tutorial, 2006
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

An example



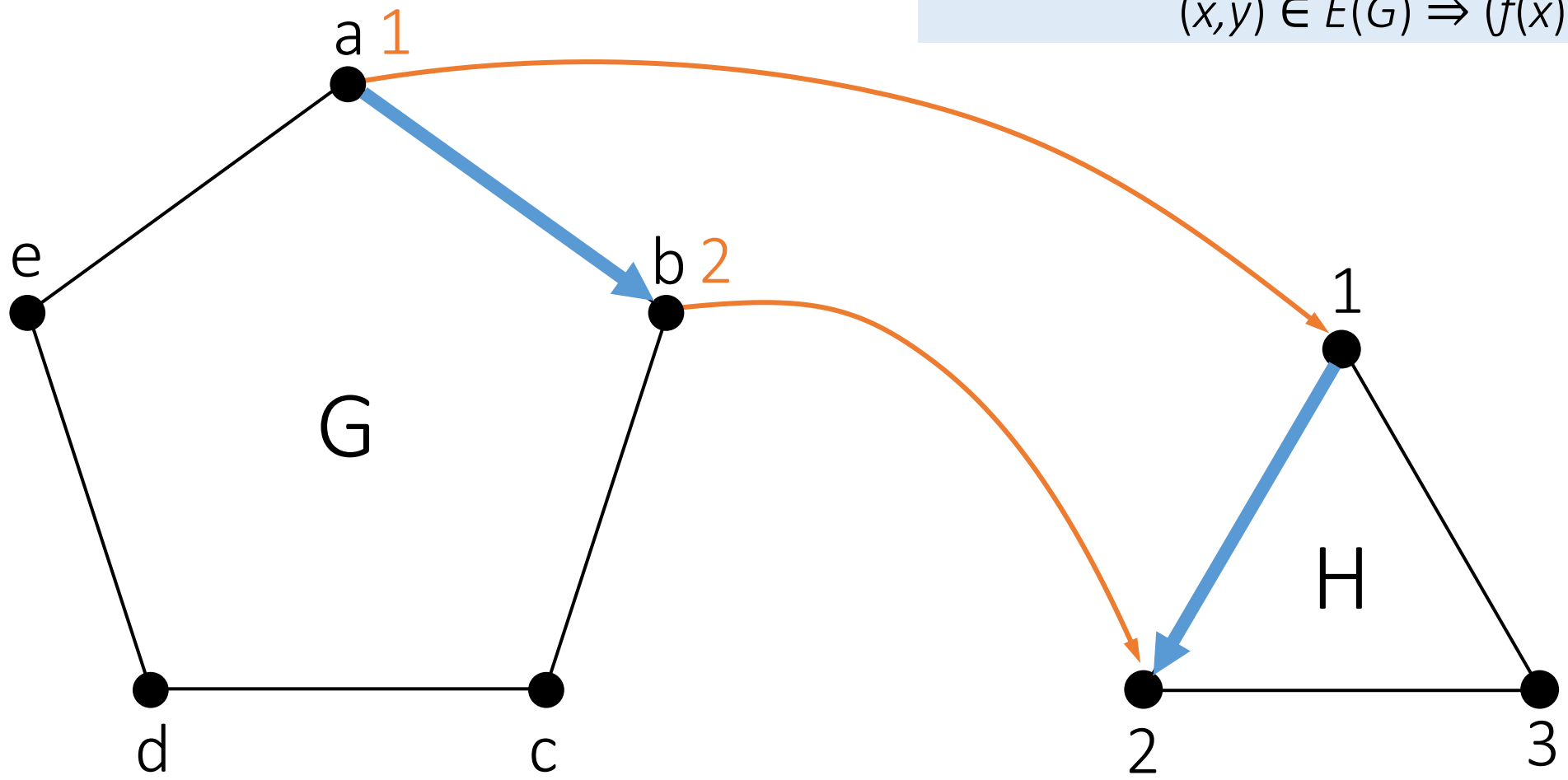
Can this assignment be extended to a homomorphism? ?

Based upon an example from Rick Brewster's Graph homomorphism tutorial, 2006
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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*

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



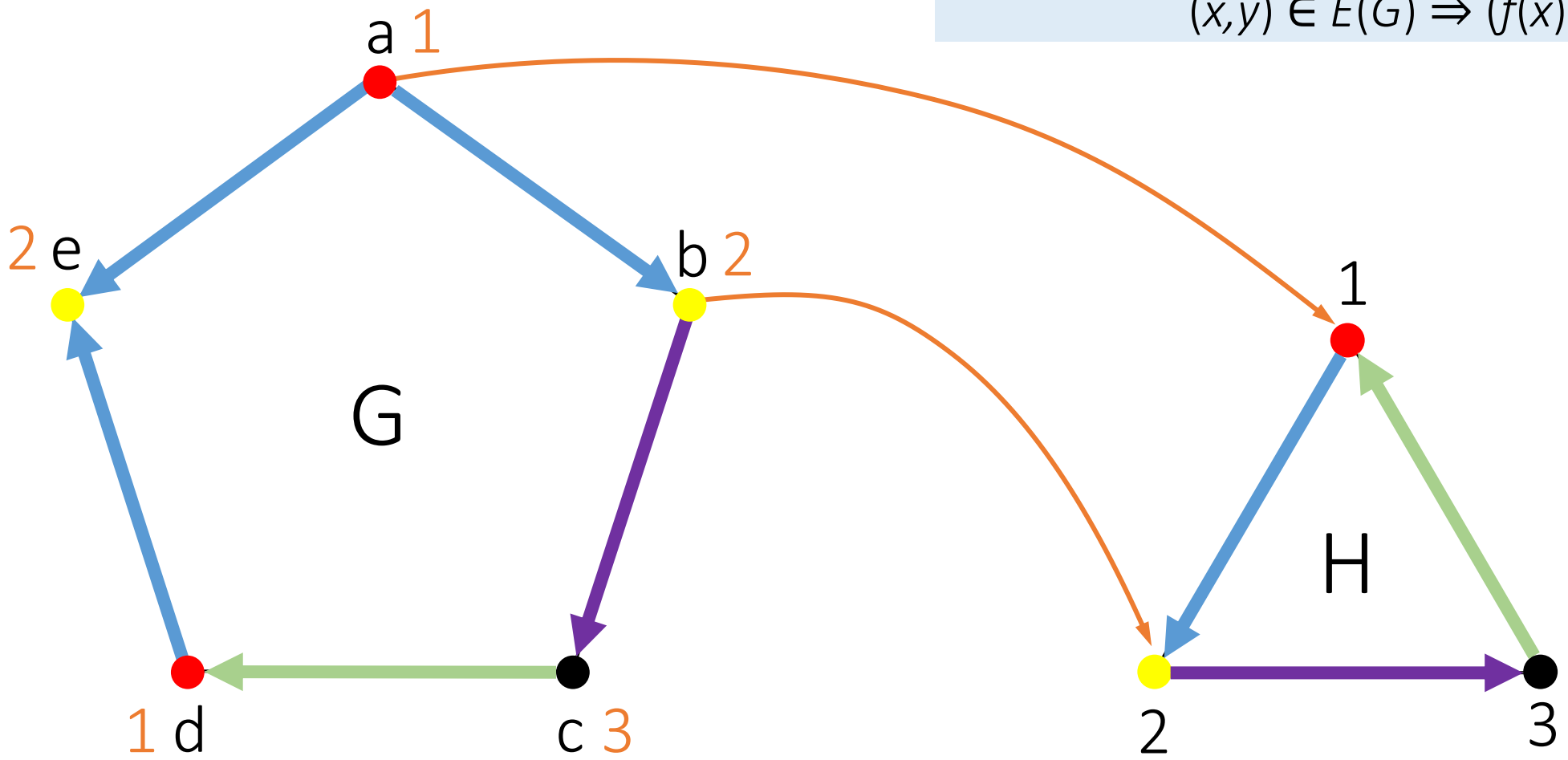
Can this assignment be extended to a homomorphism? ?

Based upon an example from Rick Brewster's Graph homomorphism tutorial, 2006

An example

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

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

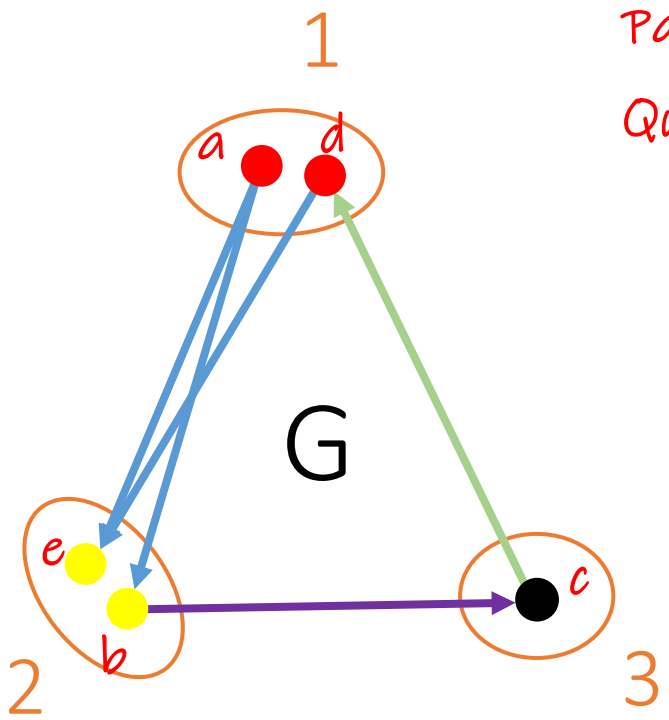


Based upon an example from Rick Brewster's Graph homomorphism tutorial, 2006

An example

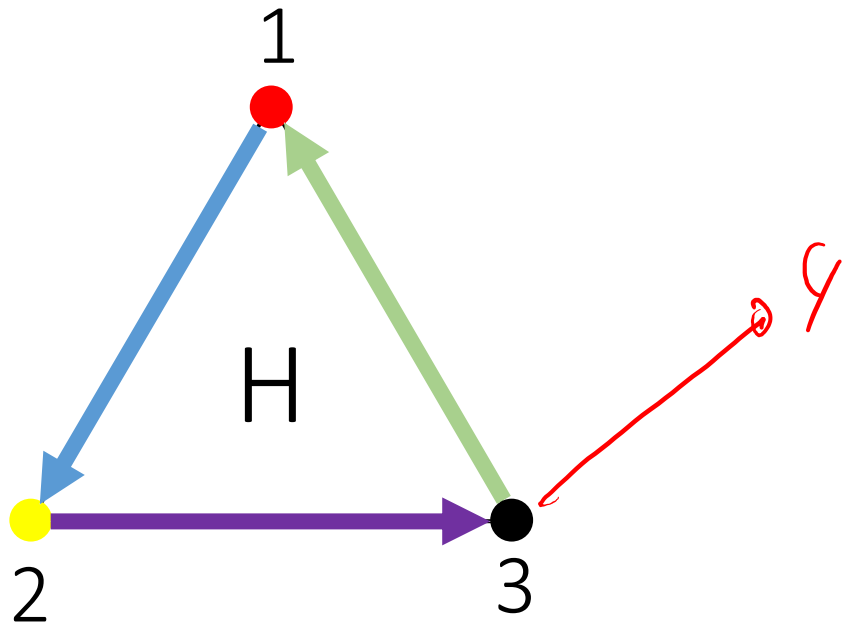
Basically a partitioning problem!

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



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

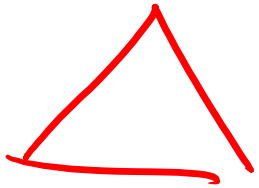
Quotient set: {[a], [b], [c]}



Based upon an example from Rick Brewster's Graph homomorphism tutorial, 2006

Some observations

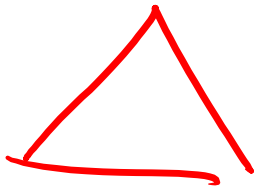
When does G → K_3 hold? (K_3 = 3-clique = triangle)



?

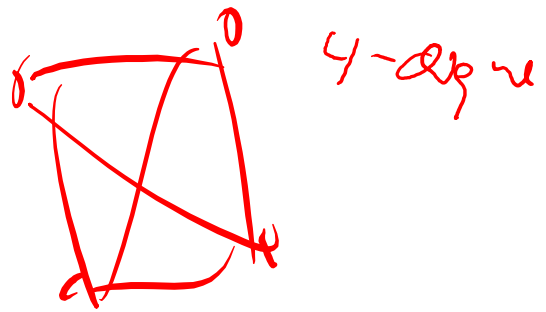
Some observations

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



iff G is 3-colorable

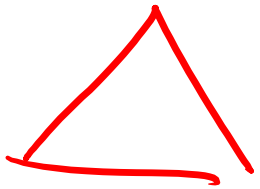
When does G → K_d hold? (K_d = d-clique)



?

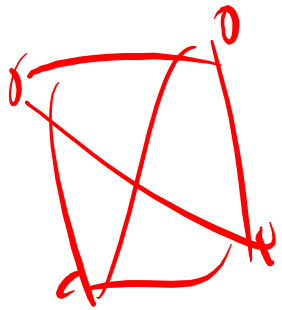
Some observations

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



iff G is 3-colorable

When does G → K_d hold? (K_d = d-clique)



iff G is d-colorable

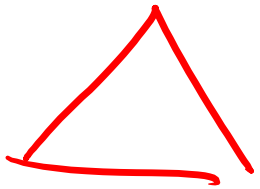
Thus homomorphisms generalize colorings:
Notation: G → H is an H-coloring of G.

What is the complexity of testing for the existence of a homomorphism (in the size of G)?

?

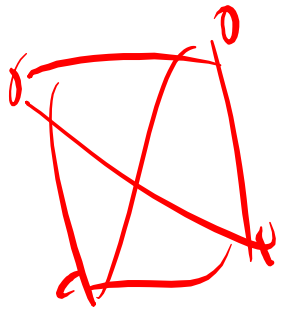
Some observations

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



iff G is 3-colorable

When does G → K_d hold? (K_d = d-clique)



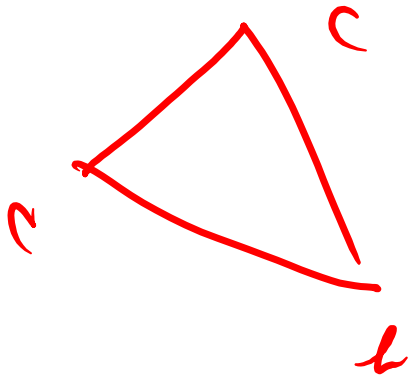
iff G is d-colorable

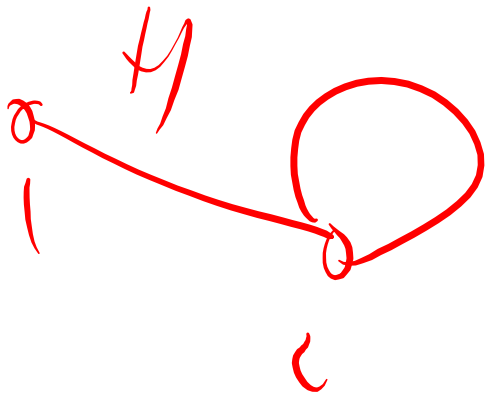
Thus homomorphisms generalize colorings:
Notation: G → H is an H-coloring of G.

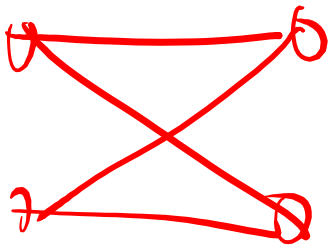
What is the complexity of testing for the existence of a homomorphism (in the size of G)?

NP-complete

The complexity of H-coloring

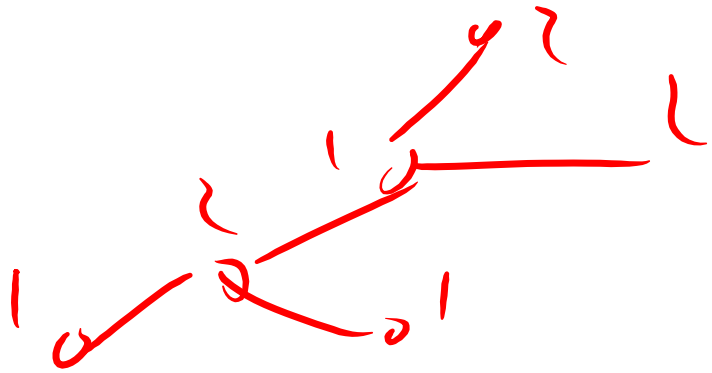






H-coloring:
Let H be a fixed graph.
Instance: A graph G.
Question: Does G admit an H-coloring?

Theorem [Hell, Nesetril'90]:
If H is bipartite or contains a self-loop, then H-coloring 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

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

$$\exists x.\exists y.\ E(x,y) \quad \Rightarrow \not\Leftarrow \quad \exists x.\ E(x,x)$$

?

Which of formulas implies the other?

Repeated variable names

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

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

E

s	t
1	2

E

s	t
1	1

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

?

Example: from addition to multiplication

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

?

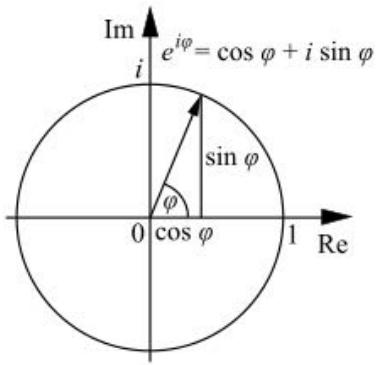
Paragraph screenshot from p.37 in 2004 - Dummit, Foote - Abstract algebra (book, 3rd ed). https://www.wiley.com/en-us/Abstract+Algebra%2C+3rd+Edition-p-9780471433347

Example: from addition to multiplication

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



Paragraph screenshot from p.37 in 2004 - Dummit, Foote - Abstract algebra (book, 3rd ed). https://www.wiley.com/en-us/Abstract+Algebra%2C+3rd+Edition-p-9780471433347

Example: from addition to multiplication

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

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

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

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

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

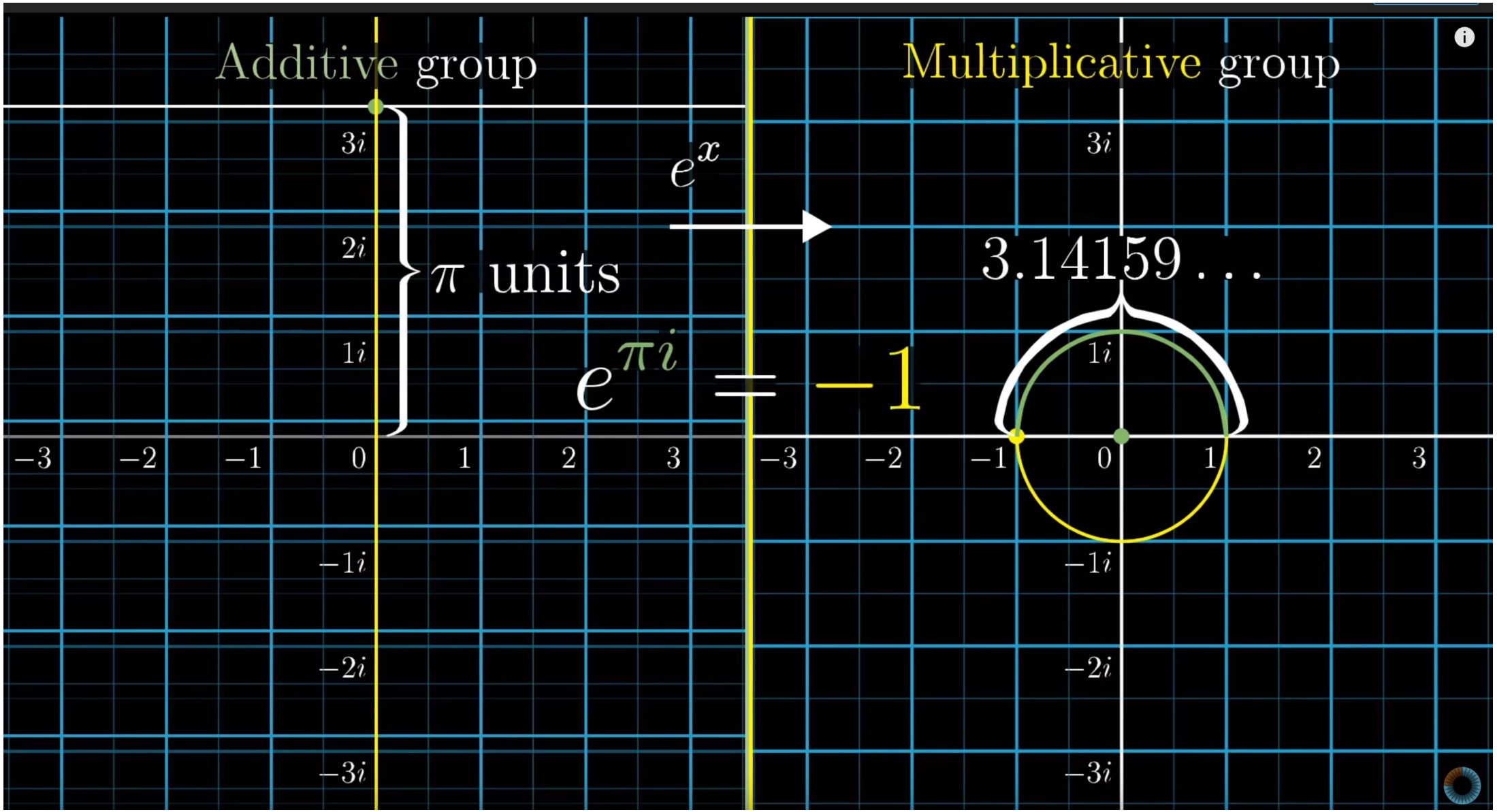
$$e^{ix} \times e^{iy} = e^{ix} \times e^{iy}$$

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

f is not 1-1

Source: Socratica. Homomorphisms, 2014: https://www.youtube.com/watch?v=cYzp5IWqCsg
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Example: from addition to multiplication

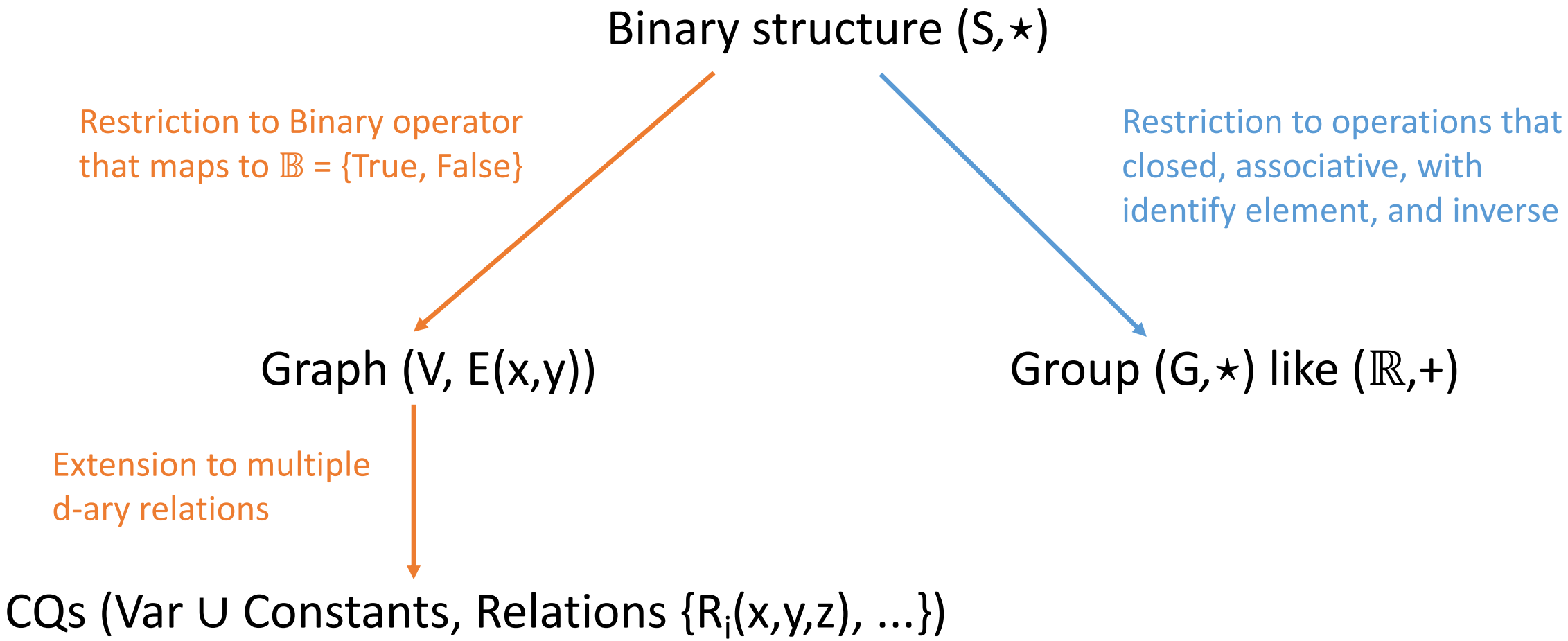


Source: 3blue1brown. Euler's formula with introductory group theory, 2017: https://www.youtube.com/watch?v=mvmuCPvRoWQ
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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



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

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

Query Containment

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

the answer (set of tuples) returned by one is guaranteed to be identical to the other answer

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 Containment

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

the answer (set of tuples) returned by one is guaranteed to be identical to the other answer

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 $q_1 \Leftarrow q_2$)

Query homomorphisms (and containment)

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:

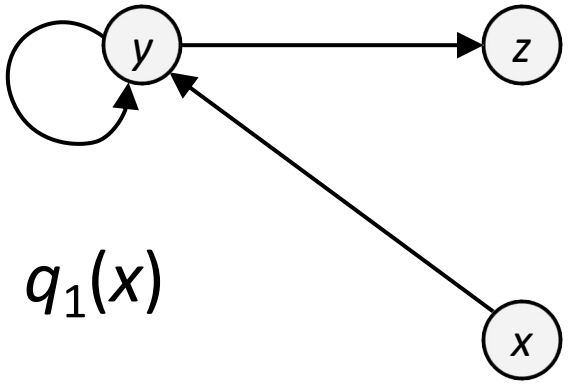
for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

need to be same relation!

Example

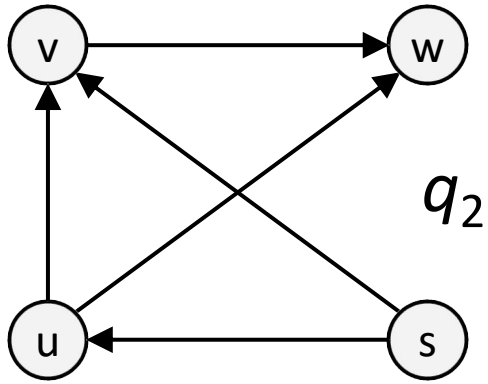
$q_1(x)$:- $R(x,y)$, $R(y,y)$, $R(y,z)$

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



$q_1(x)$

$h_{2\to1}$: ?



$q_2(x)$

Query homomorphisms (and containment)

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:

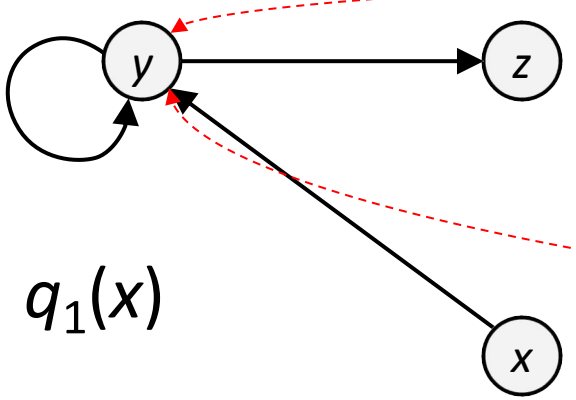
for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

need to be same relation!

Example

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

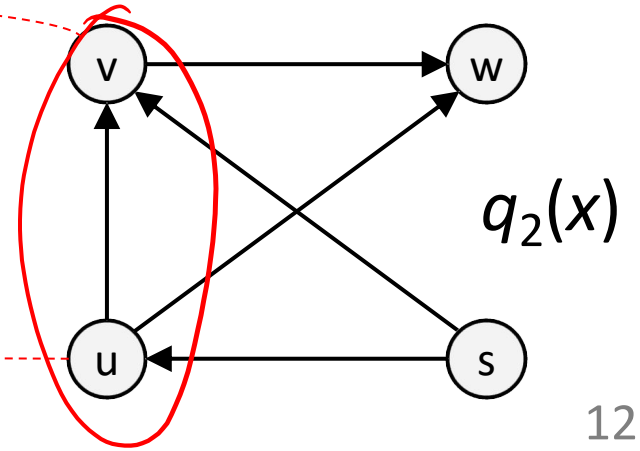
$q_2(s)$:- $R(s,u)$, $R(u,w)$, $R(s,v)$, $R(v,w)$, $R(u,v)$



$q_1(x)$

Also: $h_{2\to1}$': {s,u,v,w} → {y}; recall [Hell, Nesetril'90]

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



$q_2(x)$

Query homomorphisms (and containment)

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:

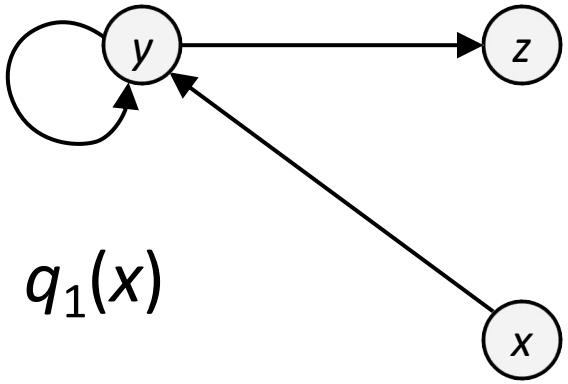
for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

need to be same relation!

Example

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

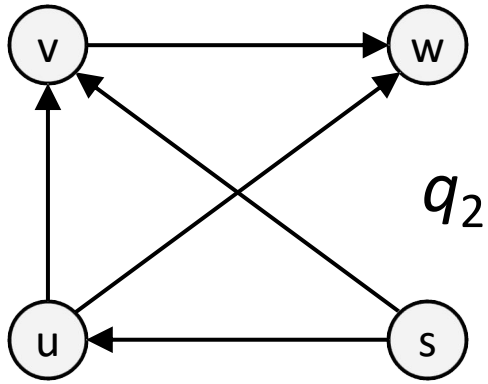
$q_2(s)$:- $R(s,u)$, $R(u,w)$, $R(s,v)$, $R(v,w)$, $R(u,v)$



$q_1(x)$

$h_{1\to2}$: ?

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



$q_2(x)$

Query homomorphisms (and containment)

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:

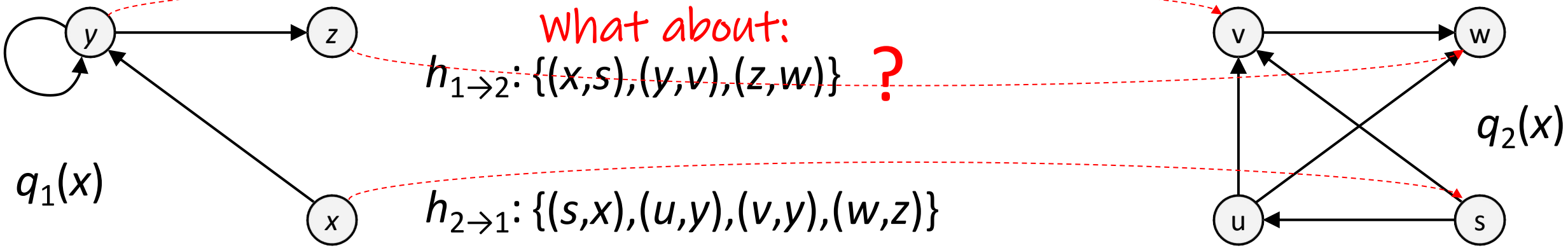
for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

need to be same relation!

Example

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

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



Query homomorphisms (and containment)

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:

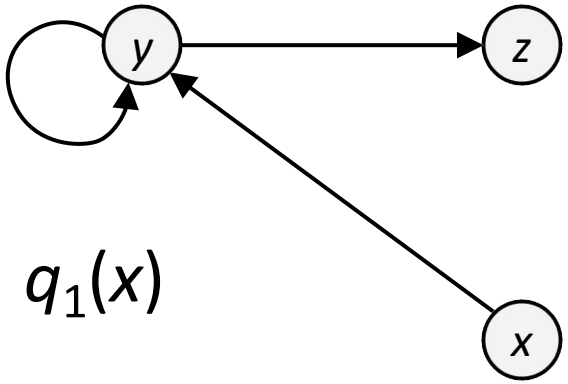
for every atom $R(x_1,x_2,...)$ in q_2, there is an atom $R(h(x_1), h(x_2), ...)$ in q_1

need to be same relation!

Example

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

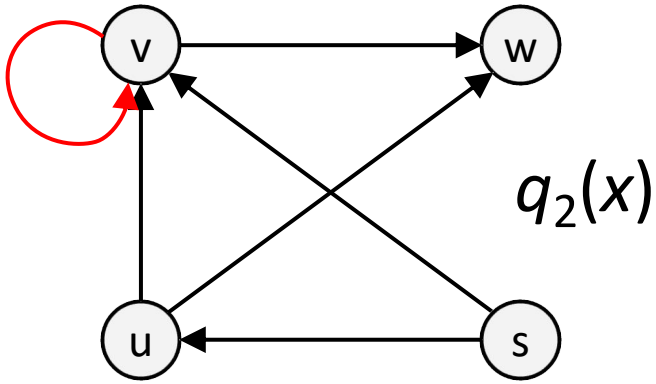
$q_2(s)$:- $R(s,u)$, $R(u,w)$, $R(s,v)$, $R(v,w)$, $R(u,v)$, $R(v,v)$



$q_1(x)$

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

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



$q_2(x)$

Query homomorphisms and containment

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(1,1)

E(1,2)

Compare to our earlier example:

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

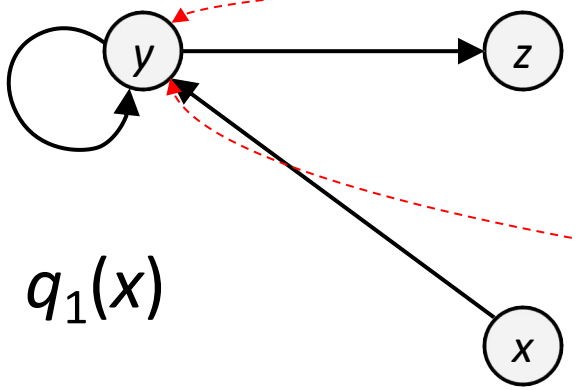
?

Example

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

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

We will use homomorphisms to reason about query containment. We try to understand the direction



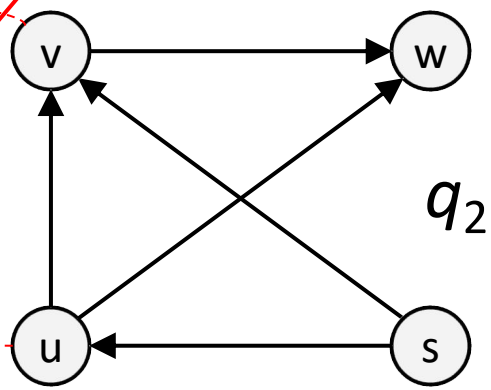
$q_1(x)$

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

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

$q_1 \not\supseteq q_2$

$q_1 \subseteq q_2$



$q_2(x)$

Query homomorphisms and containment

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

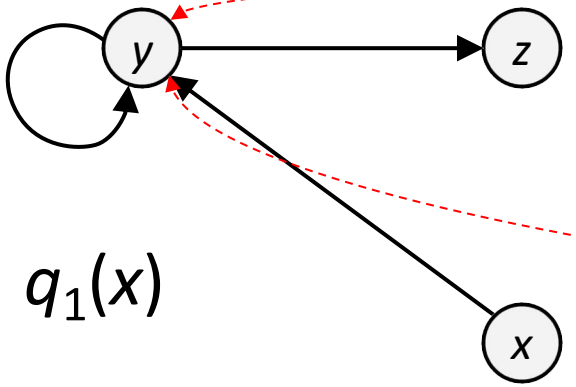
E(1,2)

Example

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

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

We will use homomorphisms to reason about query containment. We try to understand the direction



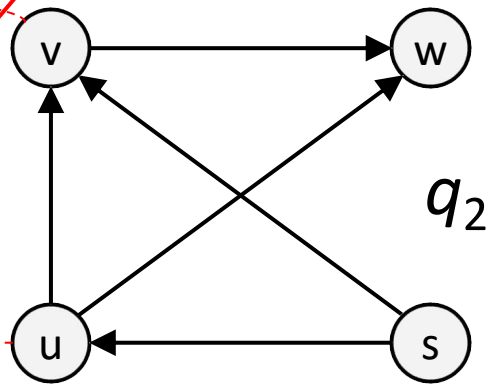
$q_1(x)$

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

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

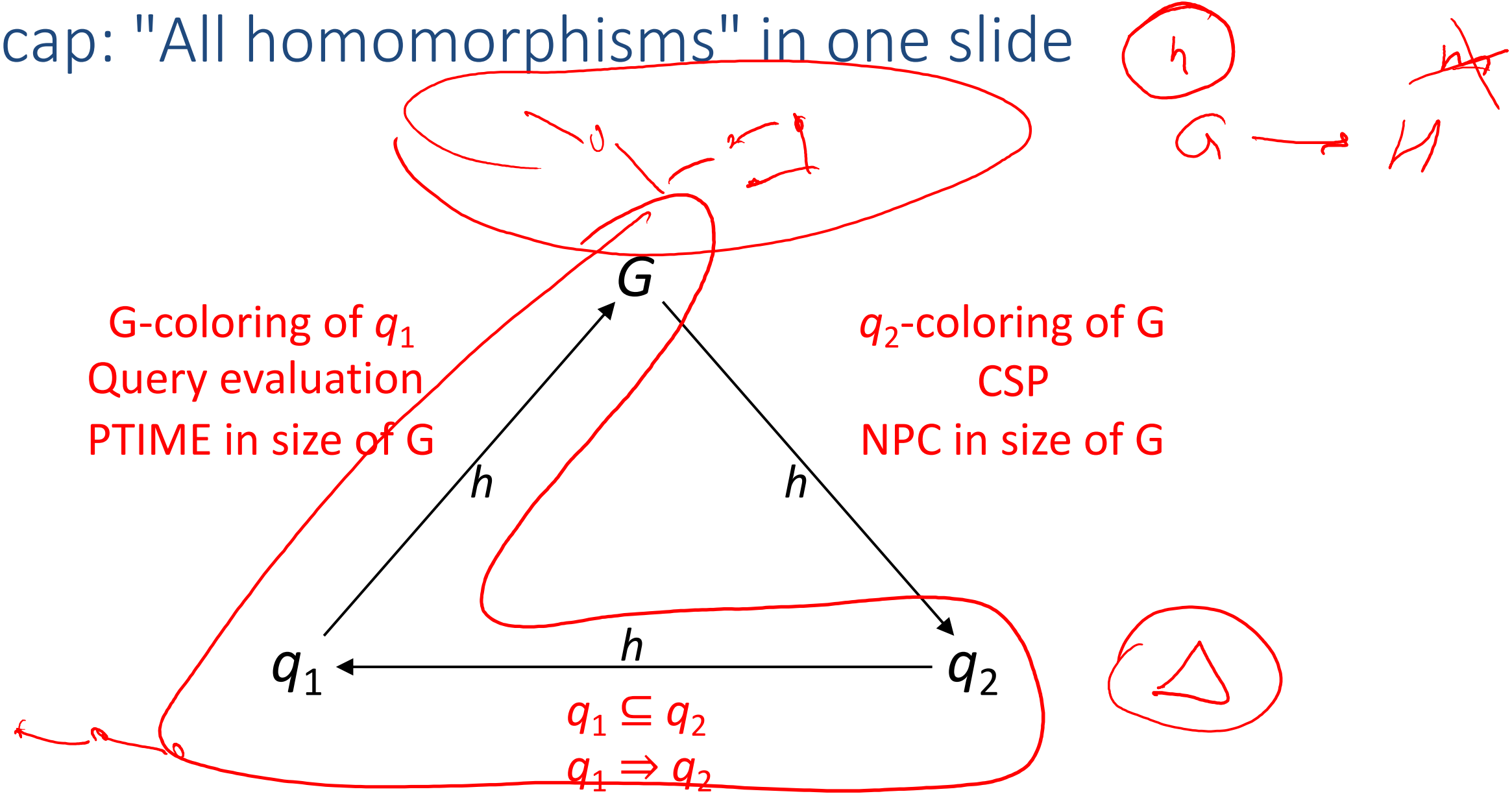
$q_1 \not\supseteq q_2$

$q_1 \subseteq q_2$



$q_2(x)$

Recap: "All homomorphisms" in one slide



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

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

$D_c[q_1]$ = ?

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

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

$D_c[q_1]$ = {R('x','y'), R('y','y'), R('y','z')}

≡ {R(a,b), R(b,b), R(b,c)}

≡ {R(1,2), R(2,2), R(2,3)}

Just treat each variable as different constant ☺

VAR CONST.

x → 1

y → 2

z → 3

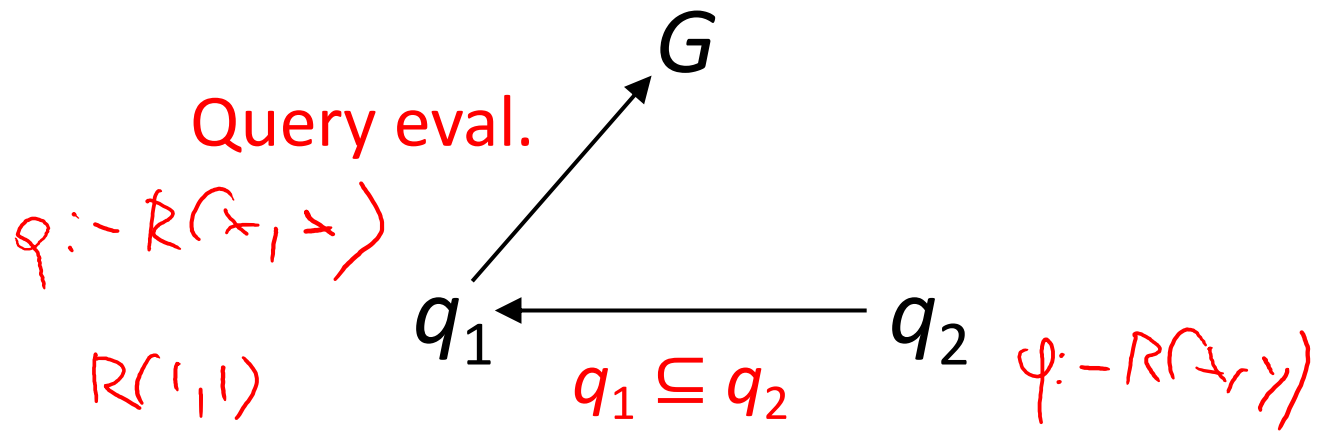
[Chandra and Merlin 1977]

THEOREM (Query Containment)

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

1) $q_1 \subseteq q_2$ $\quad (q_1 \Rightarrow q_2)$
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, Merlin. "Optimal implementation of conjunctive queries in relational data bases." STOC 1977. https://doi.org/10.1145/800105.803397

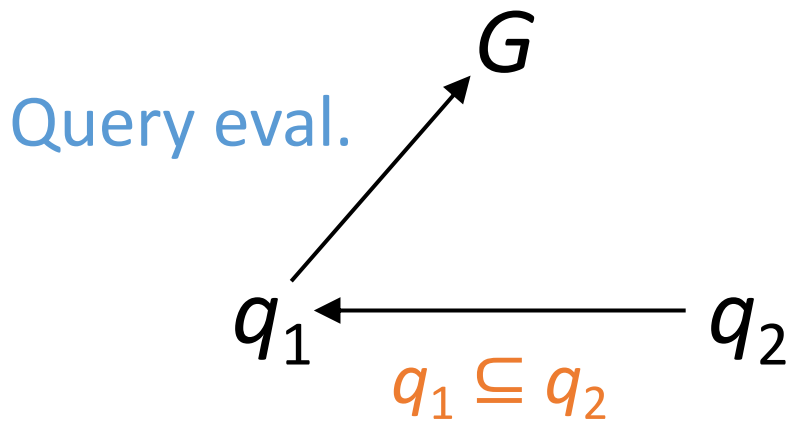
[Chandra and Merlin 1977]

We show: If there is a homomorphism $h_{2\to1}$, then for any D: $q_1(D) \Rightarrow q_2(D)$

1. For $q_1(D)$ to hold, there is a valuation v s.t. $v(q_1) \in D$
2. We will show that the composition $g = v \circ h$ is a valuation for q_2
   - 2a. By definition of h, for every $R(x_1,x_2,...)$ in q_2, $R(h(x_1),h(x_2),...)$ in q_1
   - 2b. 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]

We show: If there is a homomorphism $h_{2\to1}$, then for any D: $q_1(D) \Rightarrow q_2(D)$

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

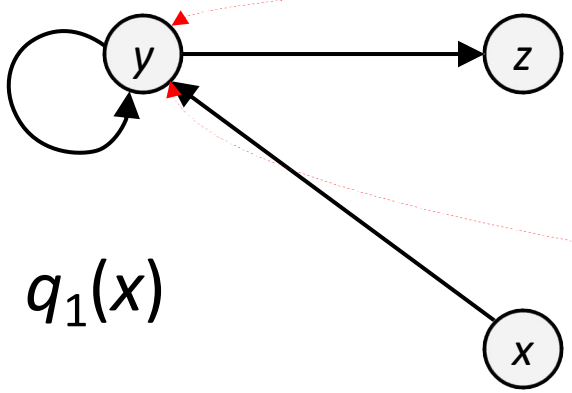
$g=v \circ h$

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

Example

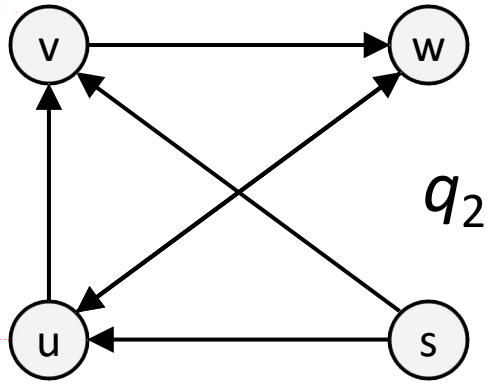
$q_1() :- R(x,y), R(y,y), R(y,z)$

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



$q_1(x)$

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



$q_2(x)$

[Chandra and Merlin 1977]

We show: If there is a homomorphism $h_{2\to1}$, then for any D: $q_1(D) \Rightarrow q_2(D)$

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

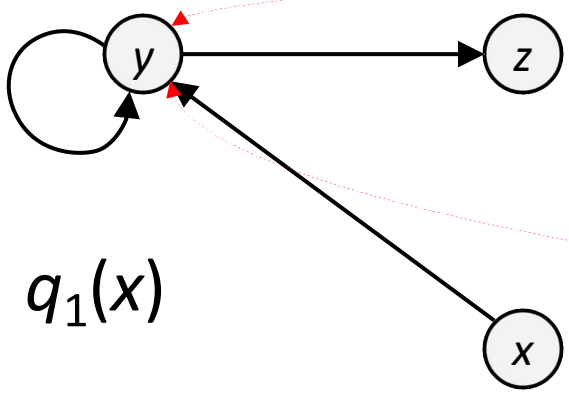
$g=v \circ h$

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

Example

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

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

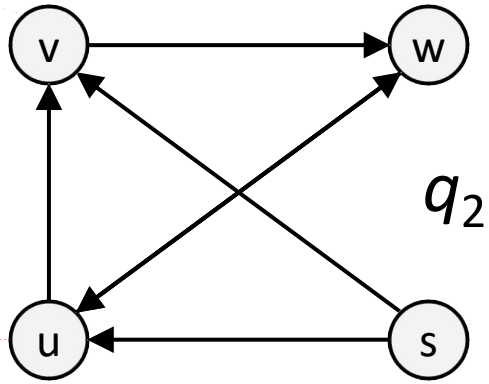


$q_1(x)$

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

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]

We show: If there is a homomorphism $h_{2\to1}$, then for any D: $q_1(D) \Rightarrow q_2(D)$

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

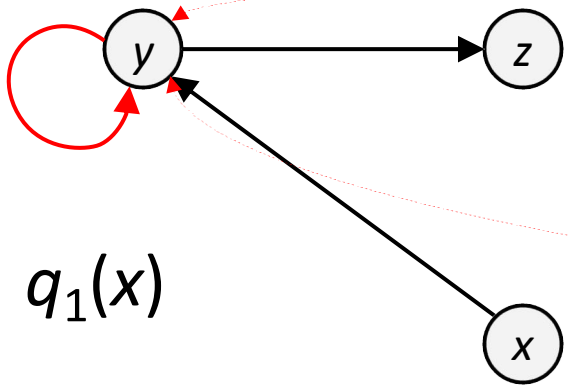
$g=v \circ h$

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

Example

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

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



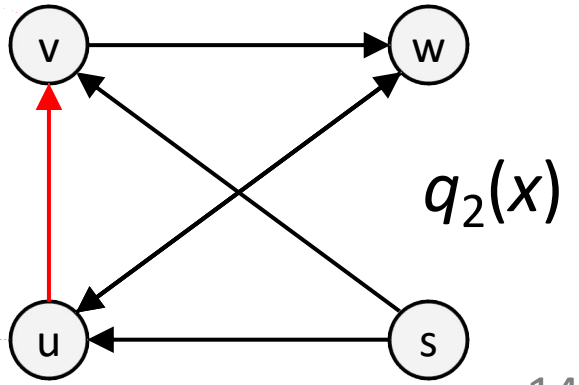
$q_1(x)$

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

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

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

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 (in the size of Q or Q'):

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 Homomophism 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

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/

Exercise: Find Homomorphisms

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

Order of subgoals in the query does not matter (thus written here as 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 ?

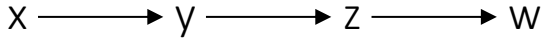
q_4: {E(x,y),E(y,x),E(y,y)}

q_5: {E(x,x)}

Example by Andreas Pieris: https://homepages.inf.ed.ac.uk/apieris/courses/atfd2020/index.html

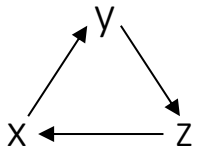
Exercise: Find the Homomorphisms

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

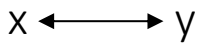


Order of subgoals in the query does not matter (thus written here as sets)

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

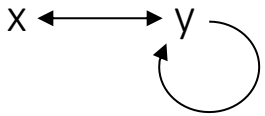


q_3: {E(x,y),E(y,x)}

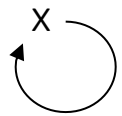


What is the containment relation between these queries ?

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



q_5: {E(x,x)}

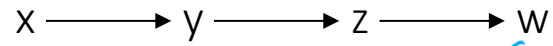


Example by Andreas Pieris: https://homepages.inf.ed.ac.uk/apieris/courses/atfd2020/index.html
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Exercise: Find the Homomorphisms

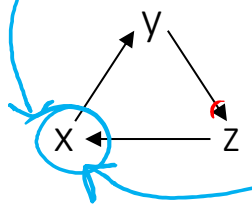
Q

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

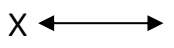


q_2: {E(x,y),E(y,z),E(z,x)}

D

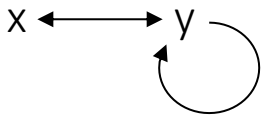


q_3: {E(x,y),E(y,x)}

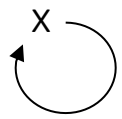


What is the containment relation between these queries ?

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

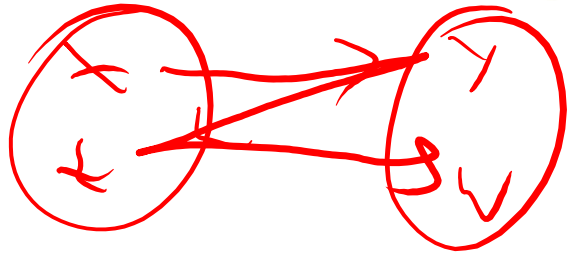


q_5: {E(x,x)}

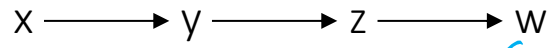


Example by Andreas Pieris: https://homepages.inf.ed.ac.uk/apieris/courses/atfd2020/index.html
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

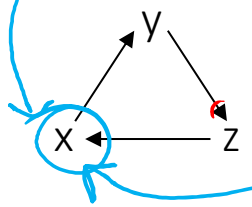
Exercise: Find the Homomorphisms



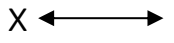
q_1: {E(x,y),E(y,z),E(z,w)}



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

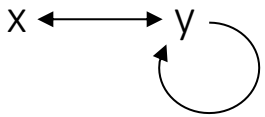


q_3: {E(x,y),E(y,x)}

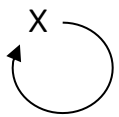


What is the containment relation between these queries ?

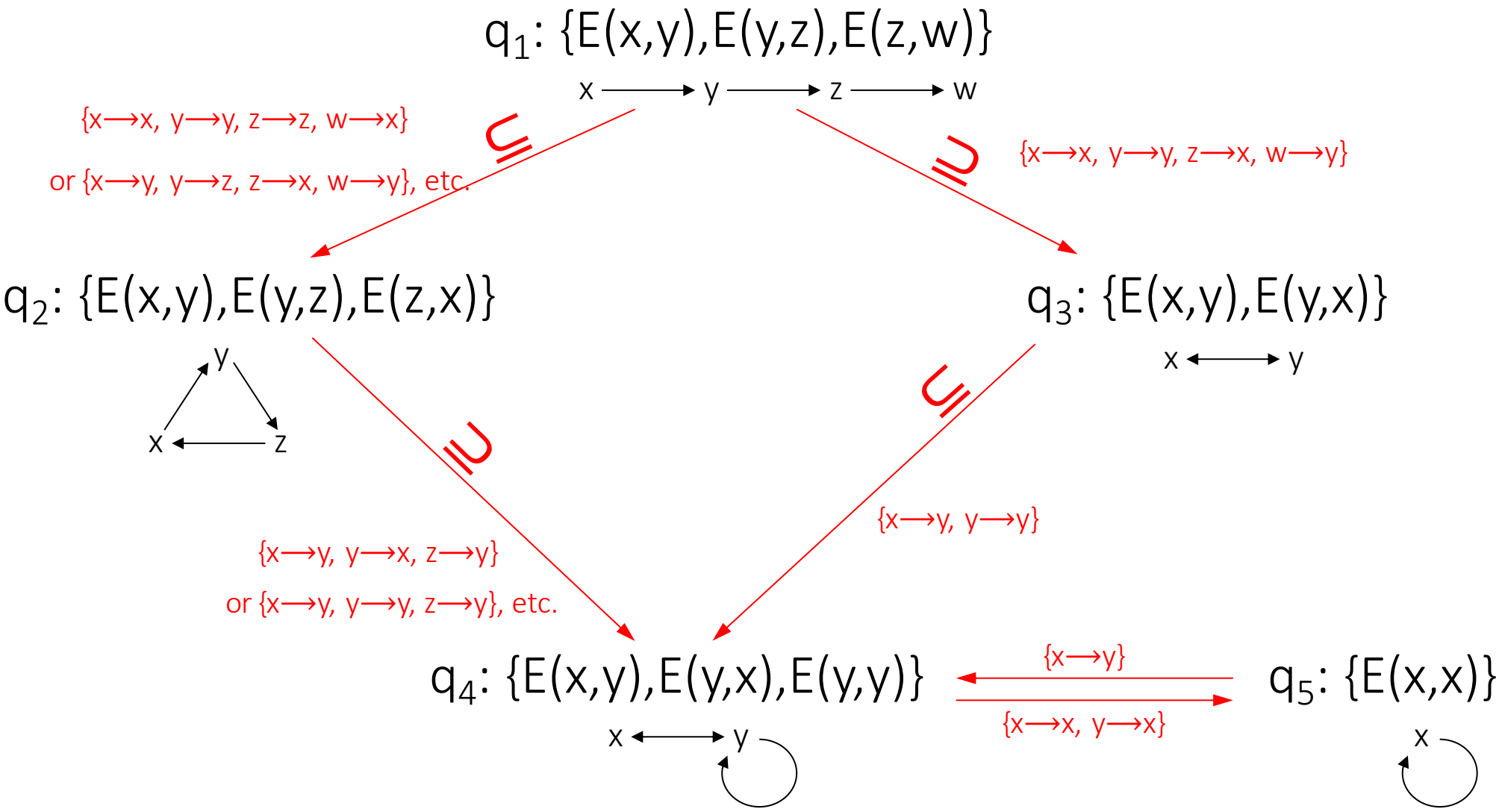
q_4: {E(x,y),E(y,x),E(y,y)}



q_5: {E(x,x)}



Exercise: Find the Homomorphisms



Example by Andreas Pieris: https://homepages.inf.ed.ac.uk/apieris/courses/atfd2020/index.html
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

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

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 $q_1 \subseteq q_2$?

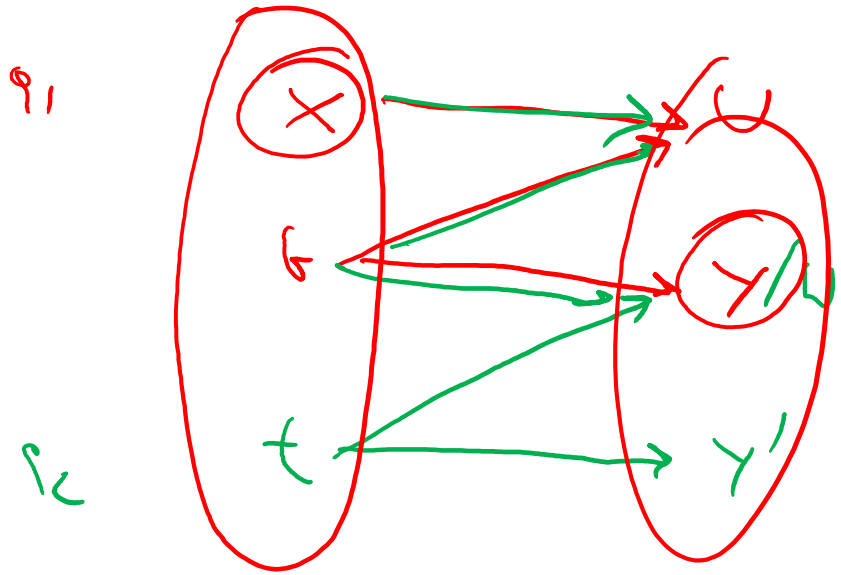
Query Homeomorphism Practice

$q_1(x,y)$:- 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 $q_1 \subseteq q_2$?

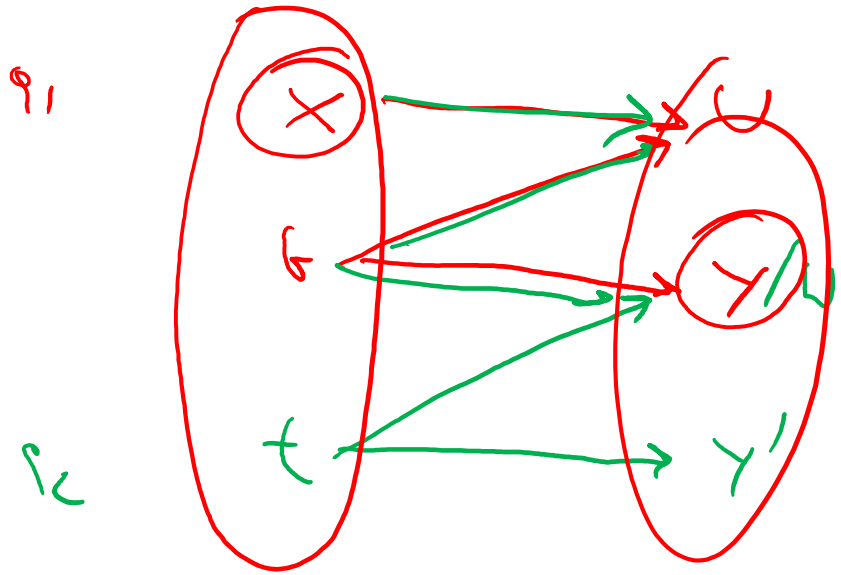
Query Homeomorphism Practice

$q_1(x,y)$:- 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 $q_1 \subseteq q_2$