Updated 2/24/2023

Topic 2: Complexity of Query Evaluation
Unit 1: Conjunctive Queries
Lecture 14

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp23)

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

2/24/2023

Pre-class conversations

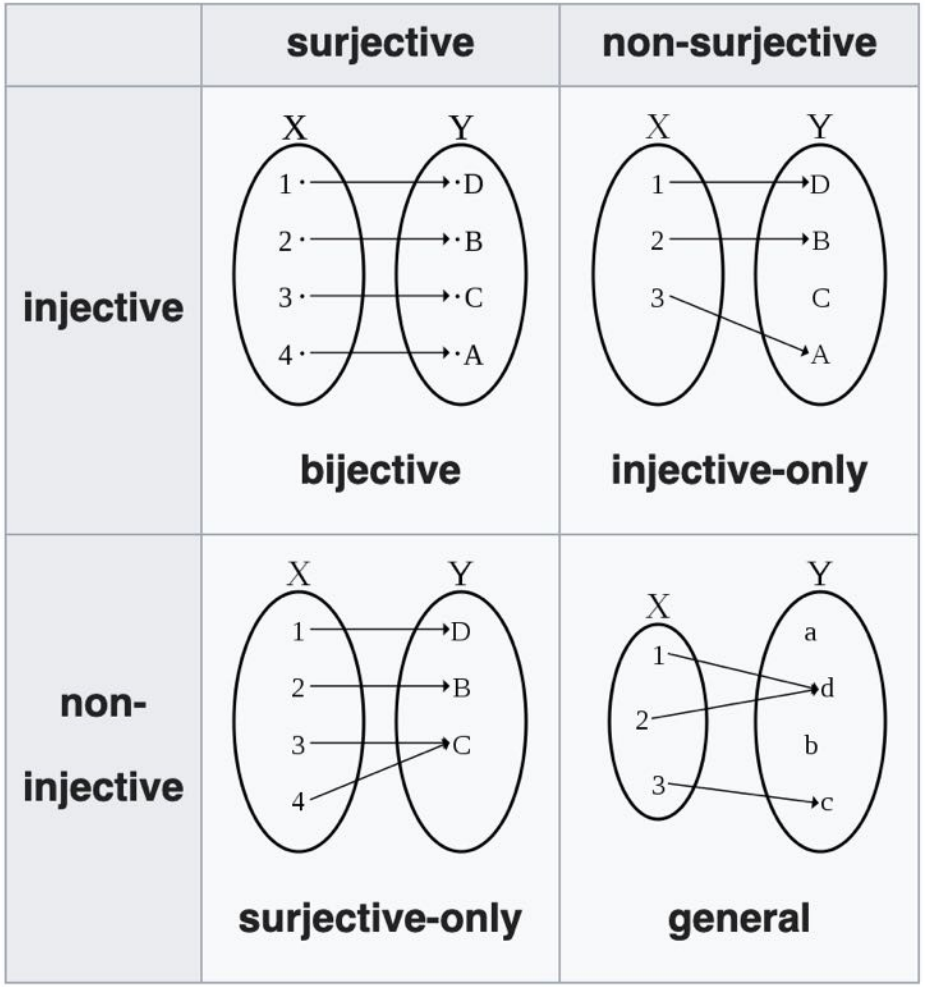
- Last class summary
- Project ideas

- Today:
  - Homomorphisms and the connections to:
    - Query containment
    - Query minimization
    - Query evaluation

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 ?

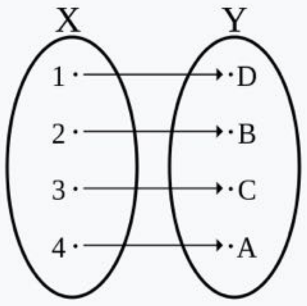
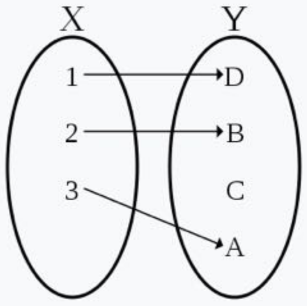
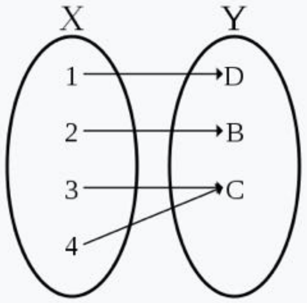
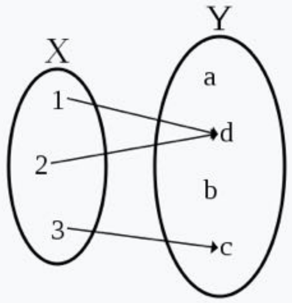
Surjective function ?

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 $f\colon X \to Y$

	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

?

$\exists!\, y \in Y[P(y)]$

$\exists y \in Y[P(y) \wedge \forall y' \in Y[P(y') \Rightarrow y = y']]$

$\exists y \in Y[P(y) \wedge \neg\exists y' \in Y[P(y') \wedge y \neq y']]$

Surjective function

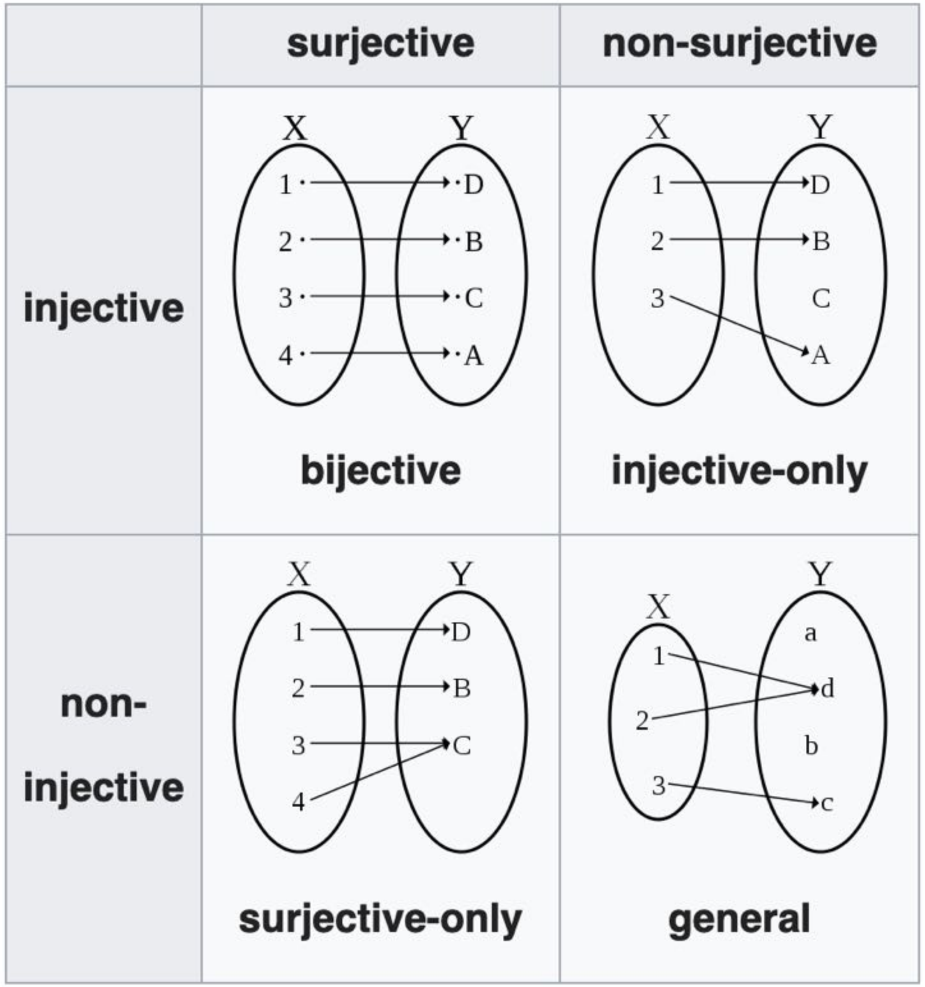
?

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

$\exists!\, y \in Y[P(y)]$

$\exists y \in Y[P(y) \wedge \forall y' \in Y[P(y') \Rightarrow y = y']]$

$\exists y \in Y[P(y) \wedge \neg\exists y' \in Y[P(y') \wedge y \neq y']]$

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

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

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

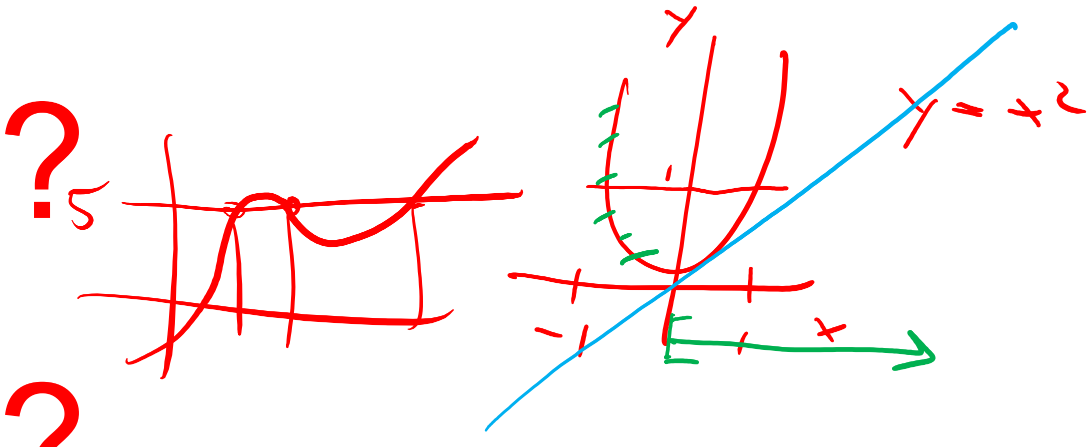
$\forall x \in X, \exists!\, y \in Y[y = f(x)]\}$

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)

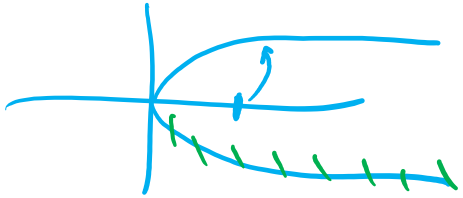
$\ldots \wedge \forall x, x' \in X.[x \neq x' \Rightarrow f(x) \neq f(x')]$

logical transpose without inequality: $\ldots \wedge \forall x, x' \in X.[f(x) = f(x') \Rightarrow x = x']$

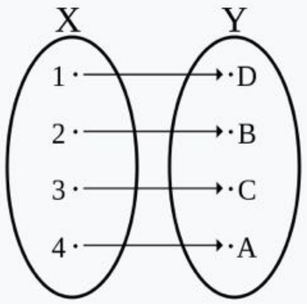
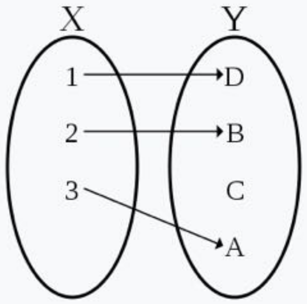
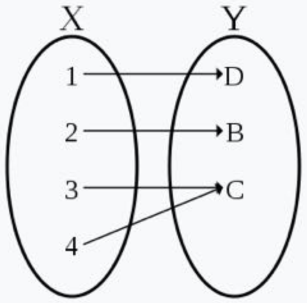
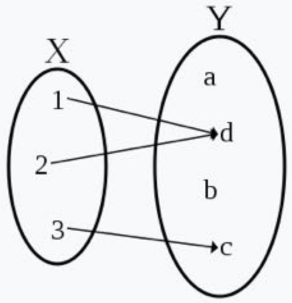
Surjective function ?



Bijective function ?



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

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

$\exists! \, y \in Y[P(y)]$

$\exists y \in Y[P(y) \land \forall y' \in Y[P(y') \Rightarrow y = y']]$

$\exists y \in Y[P(y) \land \neg \exists y' \in Y[P(y') \land y \neq y']]$

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

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

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

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

logical transpose without inequality: $\ldots \land \forall x, x' \in X. [f(x) = f(x') \Rightarrow x = x']$

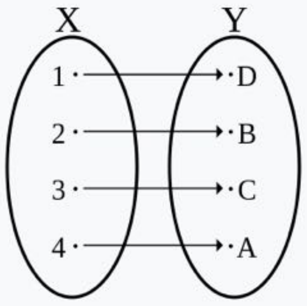
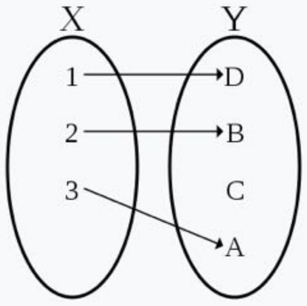
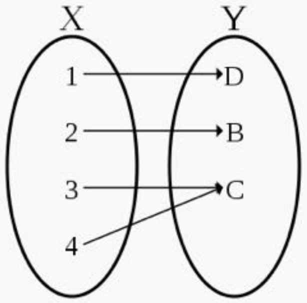
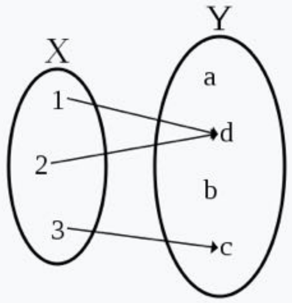
Surjective function ("onto"): each element of the codomain is mapped to by <u>at least one</u> element of the domain (i.e. the image and the codomain of the function are equal)

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

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

$\exists! y \in Y[P(y)]$

$\exists y \in Y[P(y) \wedge \forall y' \in Y[P(y') \Rightarrow y = y']]$

$\exists y \in Y[P(y) \wedge \neg \exists y' \in Y[P(y') \wedge y \neq y']]$

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

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)

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

logical transpose without inequality: $\ldots \wedge \forall x, x' \in X.[f(x) = f(x') \Rightarrow x = x']$

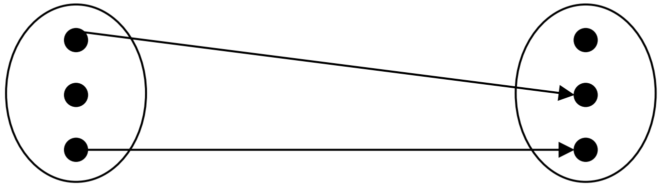
Surjective function ("onto"): each element of the codomain is mapped to by at least one element of the domain (i.e. the image and the codomain of the function are equal)

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

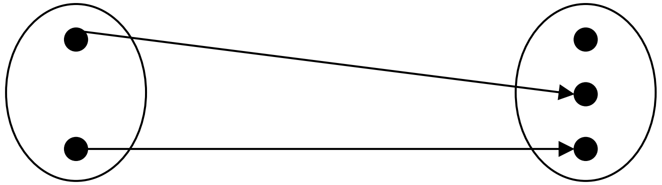
Bijective function ("invertible"): each element of the codomain is mapped to by exactly one element of the domain (both injective and surjective)

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

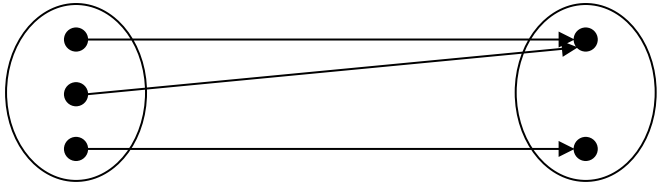
Mappings: Injection, Surjection, and Bijection



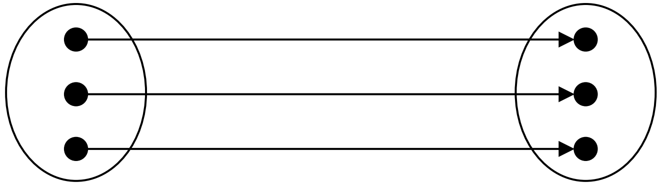
?



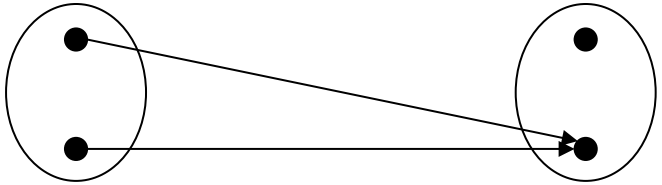
?



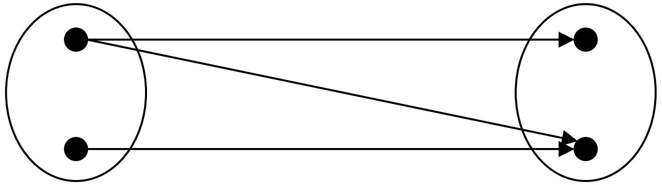
?



?

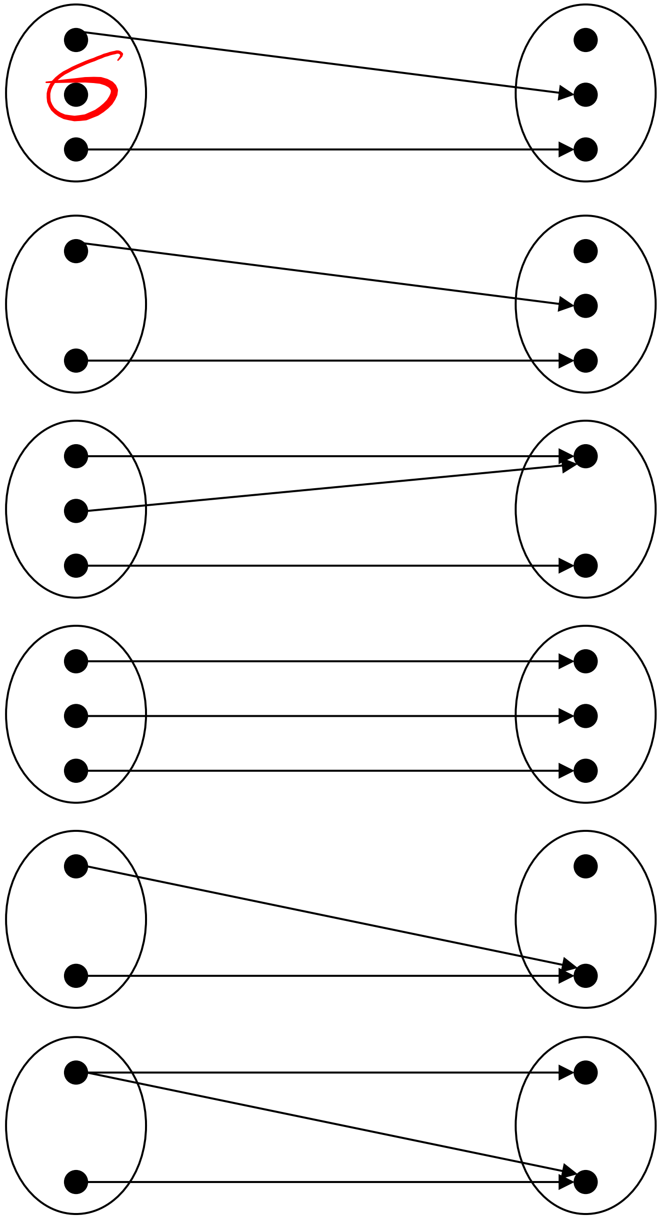


?



?

Mappings: Injection, Surjection, and Bijection



not a mapping (or function)!

?

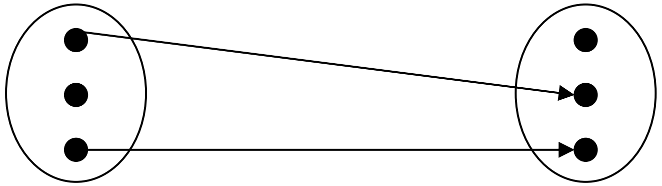
?

?

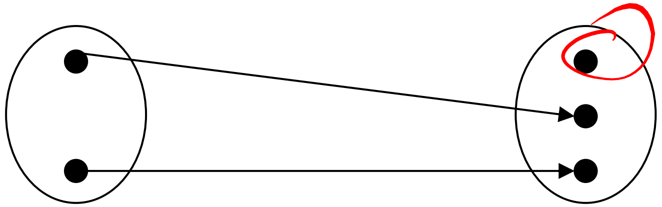
?

?

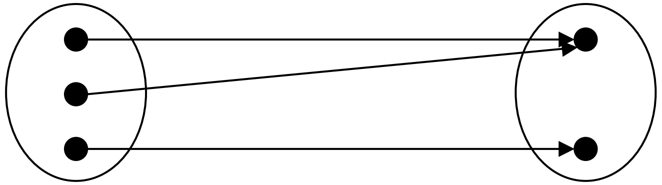
Mappings: Injection, Surjection, and Bijection



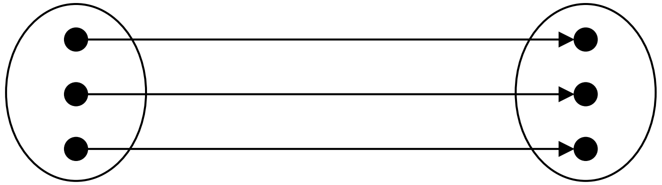
not a mapping (or function)!



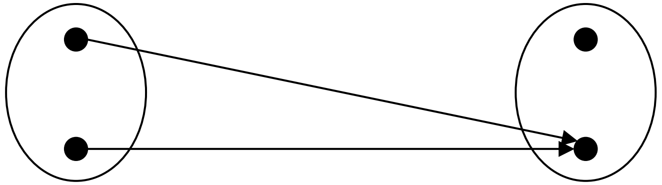
injective function (or one-to-one): maps distinct elements of its domain to distinct elements of its codomain



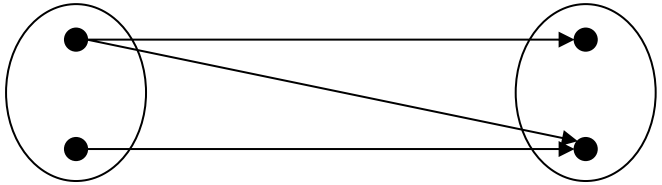
?



?

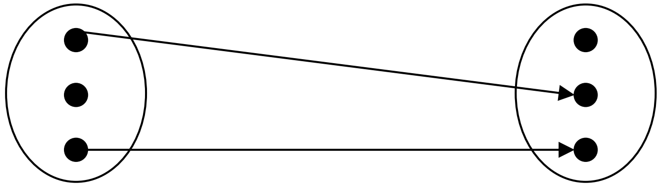


?

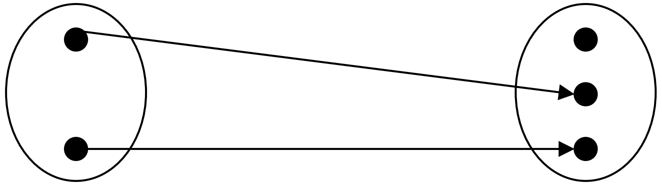


?

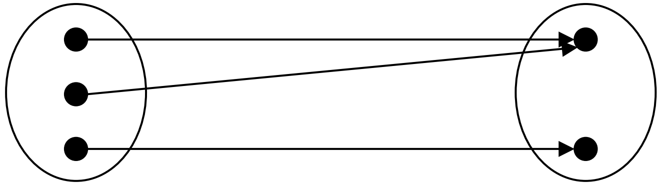
Mappings: Injection, Surjection, and Bijection



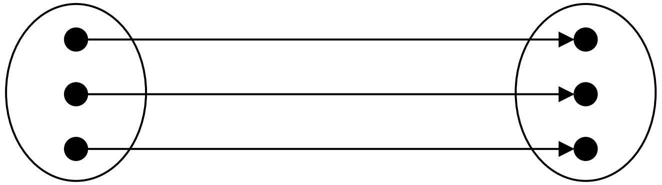
not a mapping (or function)!



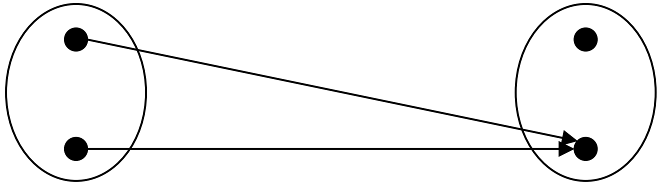
injective function (or one-to-one): maps distinct elements of its domain to distinct elements of its codomain



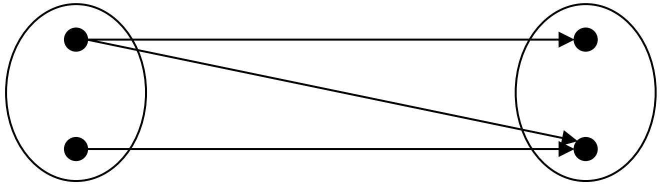
surjective (or onto): every element y in the codomain Y of f has at least one element x in the domain that maps to it



?

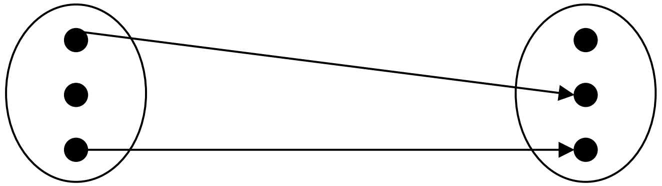


?

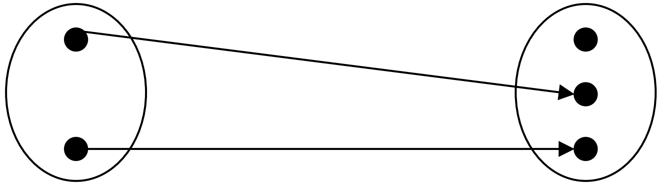


?

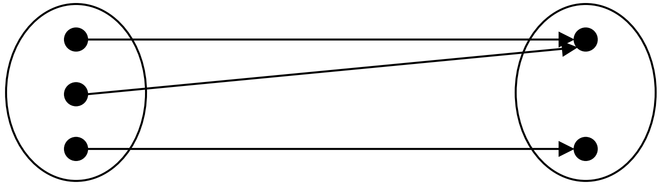
Mappings: Injection, Surjection, and Bijection



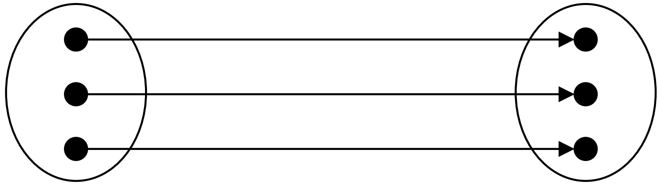
not a mapping (or function)!



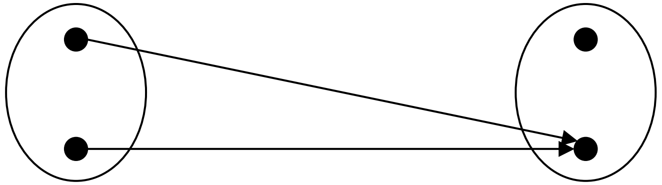
injective function (or one-to-one): maps distinct elements of its domain to distinct elements of its codomain



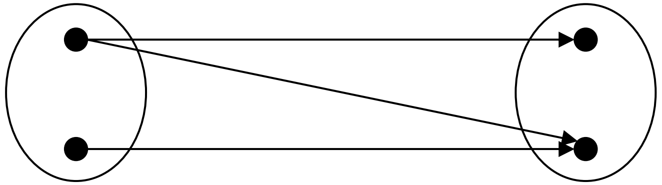
surjective (or onto): every element y in the codomain Y of f has at least one element x in the domain that maps to it



injective & surjective = bijection

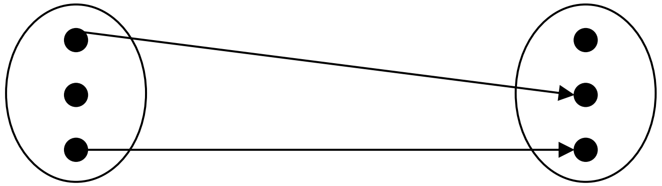


?

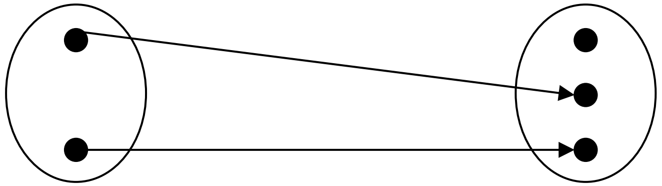


?

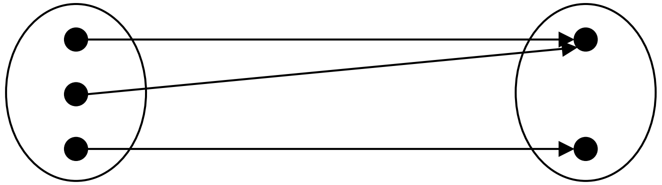
Mappings: Injection, Surjection, and Bijection



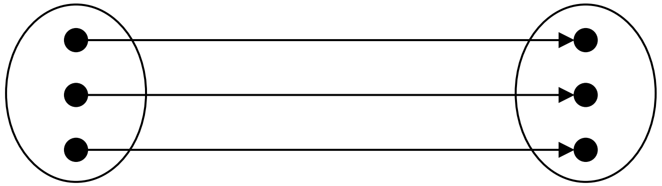
not a mapping (or function)!



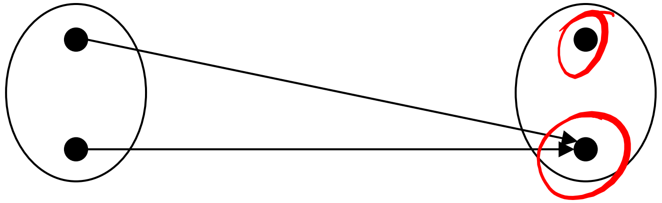
injective function (or one-to-one): maps distinct elements of its domain to distinct elements of its codomain



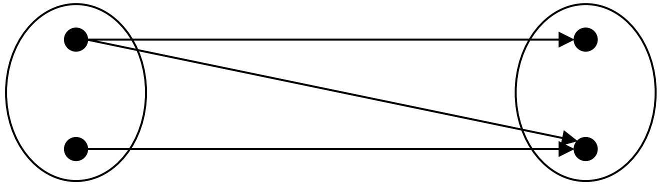
surjective (or onto): every element y in the codomain Y of f has at least one element x in the domain that maps to it



injective & surjective = bijection

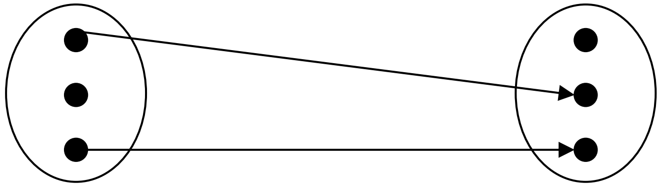


neighter

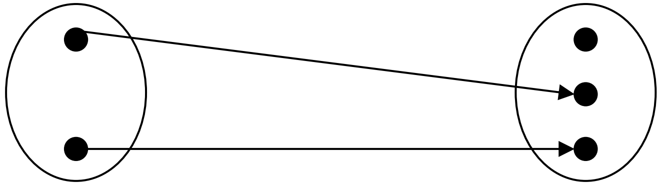


?

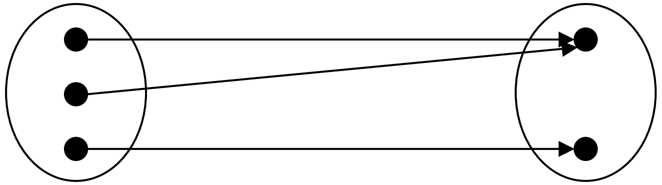
Mappings: Injection, Surjection, and Bijection



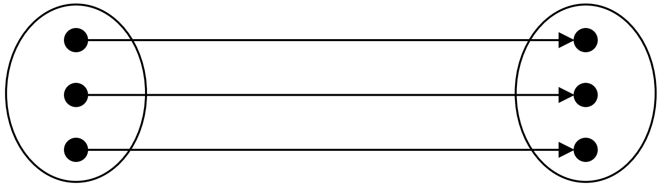
not a mapping (or function)!



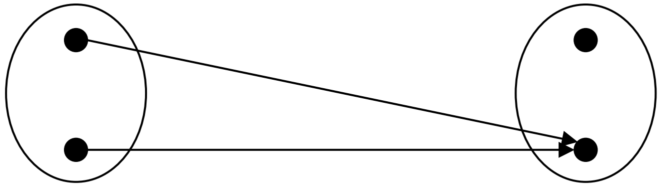
injective function (or one-to-one): maps distinct elements of its domain to <u>distinct elements of its codomain</u>



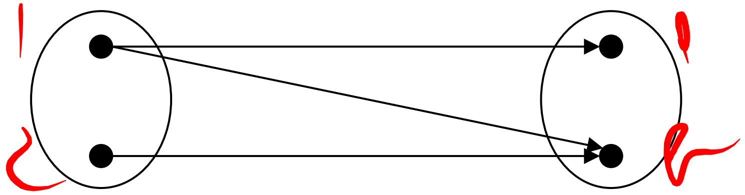
surjective (or onto): <u>every element y in the codomain Y</u> of f has at least one element x in the domain that maps to it



injective & surjective = bijection



neighter

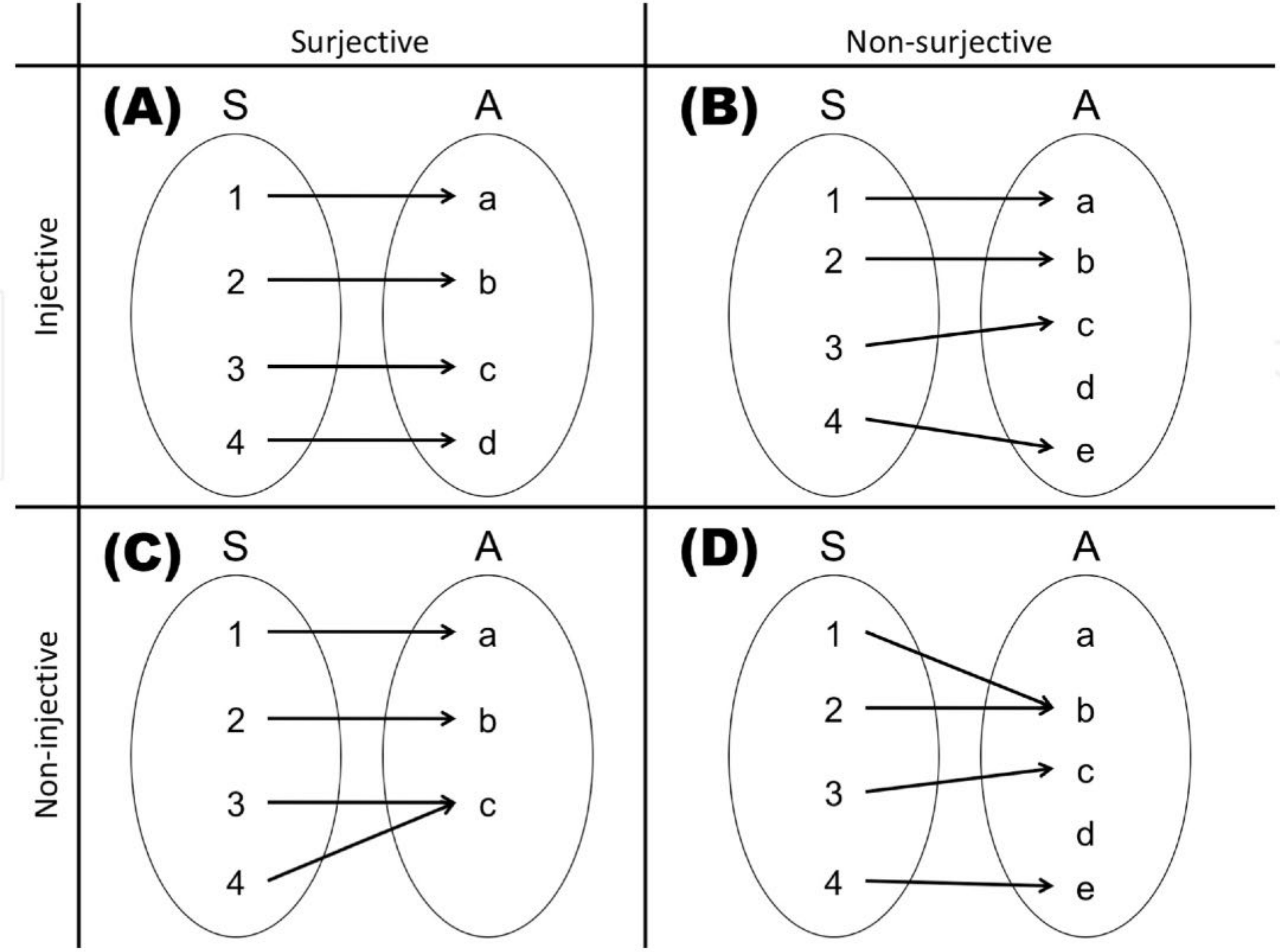


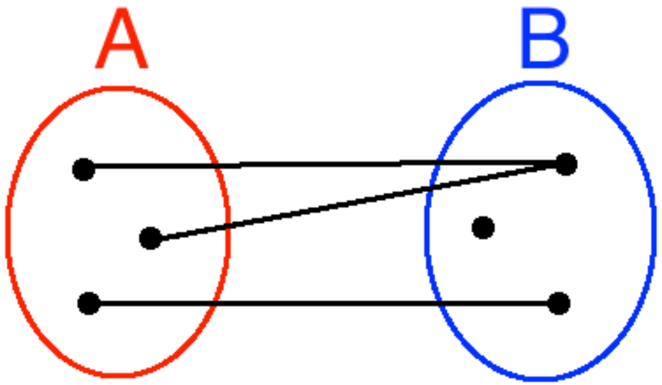
not even a mapping!

R

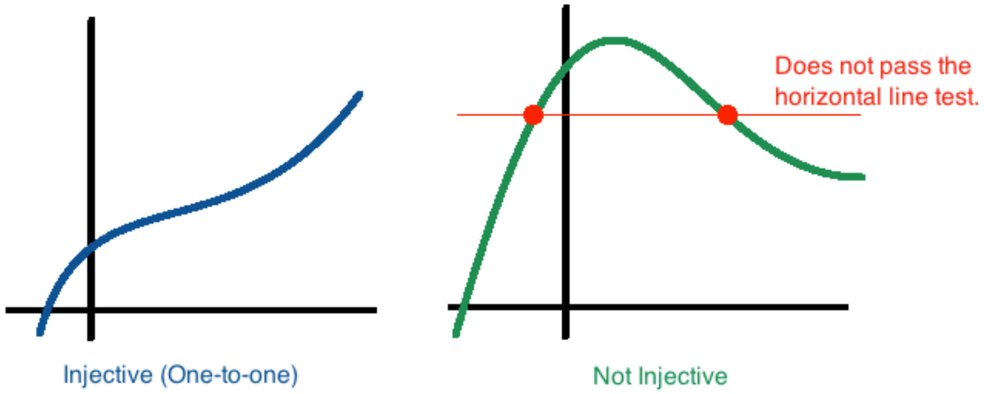
1	a
1	b
2	b

Bijection, Injection, and Surjection



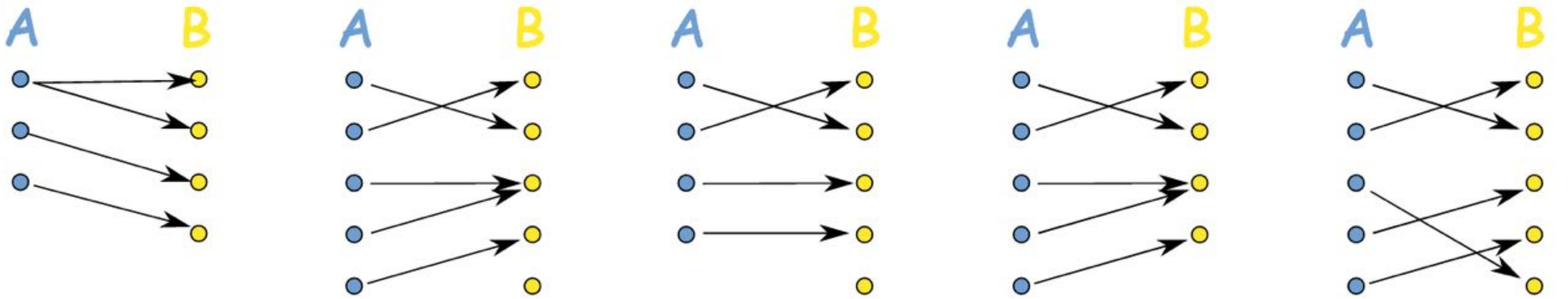


Neither Injective or Surjective
Two elements in set A maps to the same element in set B (not injective), and one element in set B is not in the image or range of the function that maps set A to B (not surjective).



Sources: 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



NOT a Function

A has many B

General Function

B can have many A

Injective (not surjective)

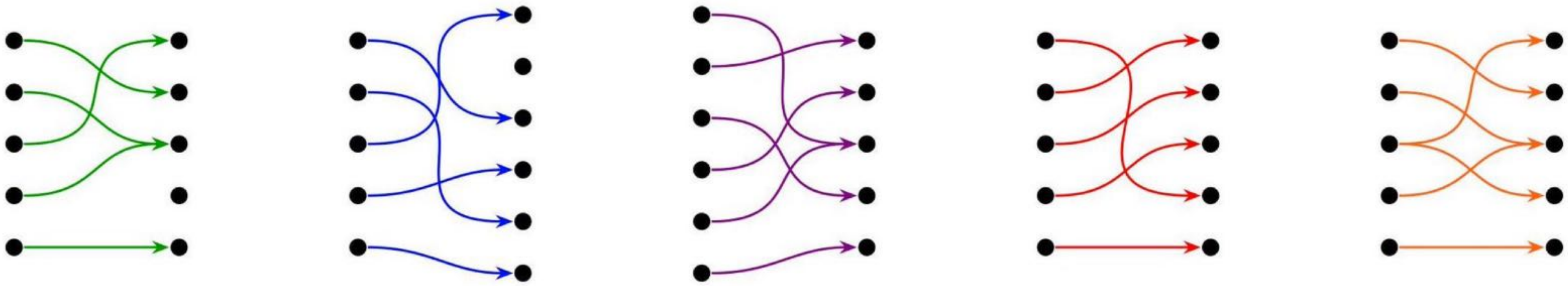
B can't have many A

Surjective (not injective)

Every B has some A

Bijective (injective, surjective)

A to B, perfectly



A function not injective not surjective

An injective function not surjective

A surjective function not injective

A bijective function injective + surjective

Not a function

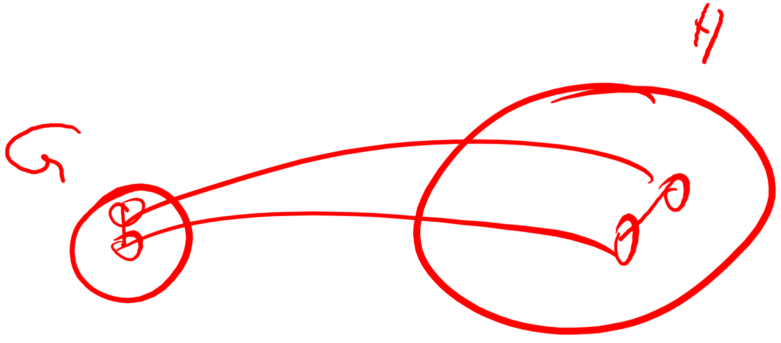
Sources: https://www.mathsisfun.com/sets/injective-surjective-bijective.html, https://twitter.com/jdhamkins/status/841318019397779456, Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

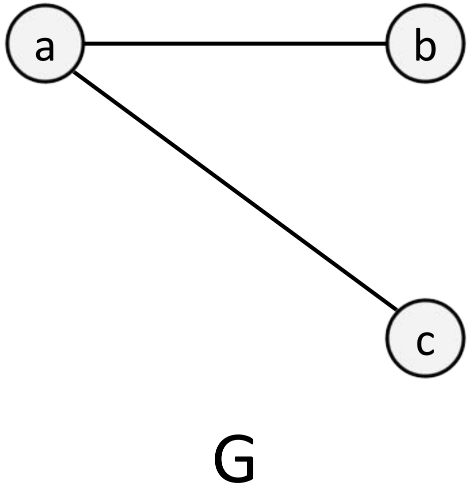
We make a detour to Graph matching

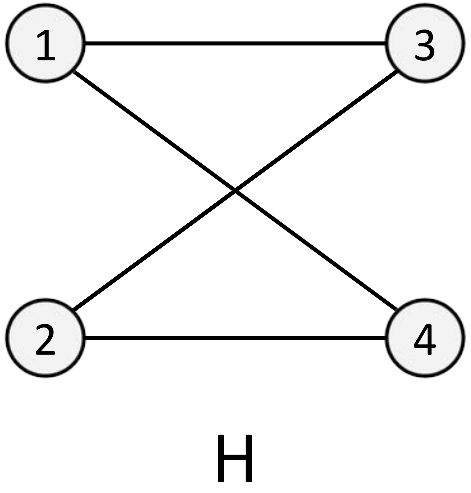
- Finding a correspondence between the nodes and the edges of two graphs that satisfies some (more or less stringent) constraints

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



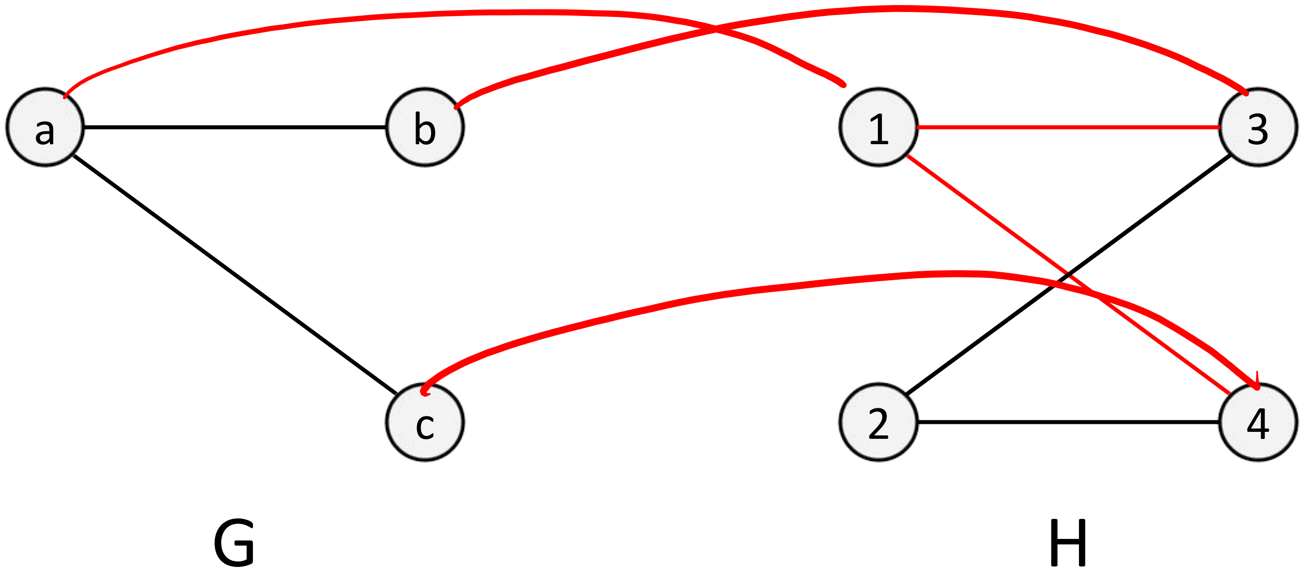




Is there a homomorphism from G to H ?

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

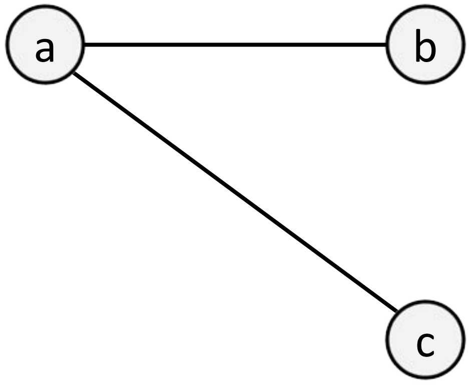


h: {(a,1), (b,3), (c,4)}

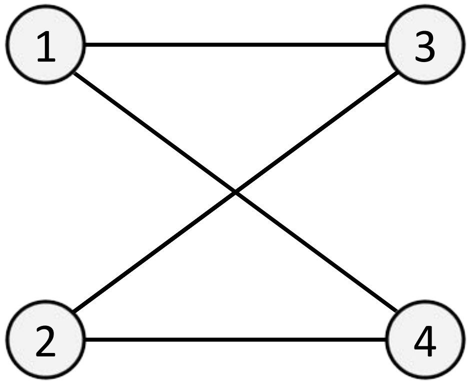
does not need to be surjective!

Homomorphism

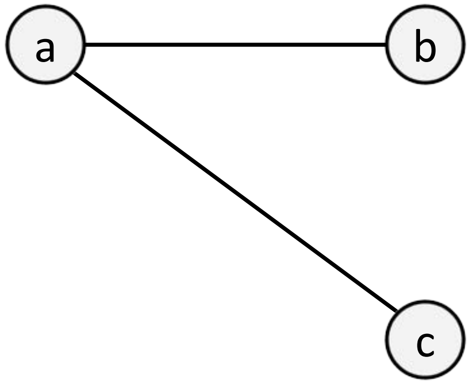
- A graph homomorphism *h* from graph $G(V_G,E_G)$ to $H(V_H,E_H)$, is a mapping from V_G to V_H such that $\{x,y\} \in E_G$ implies $\{h(x),h(y)\} \in E_H$
  - "edge-preserving": if two nodes in *G* are linked by an edge, then they are mapped to two nodes in *H* that are also linked



G



H



G

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

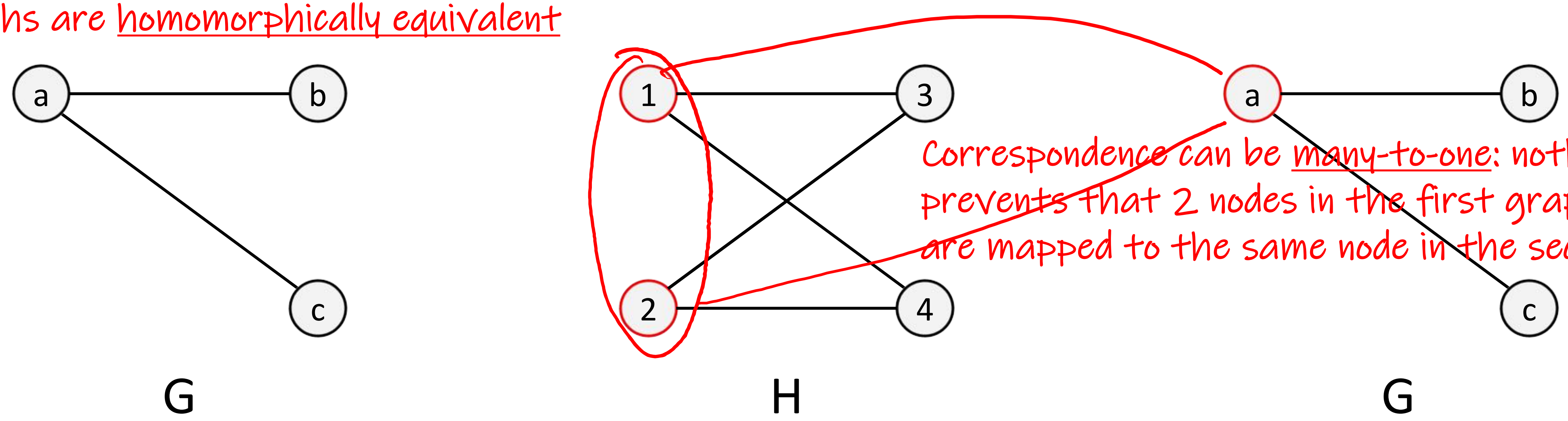
does not need to be surjective!

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



Correspondence can be many-to-one: nothing prevents that 2 nodes in the first graph are mapped to the same node in the second

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

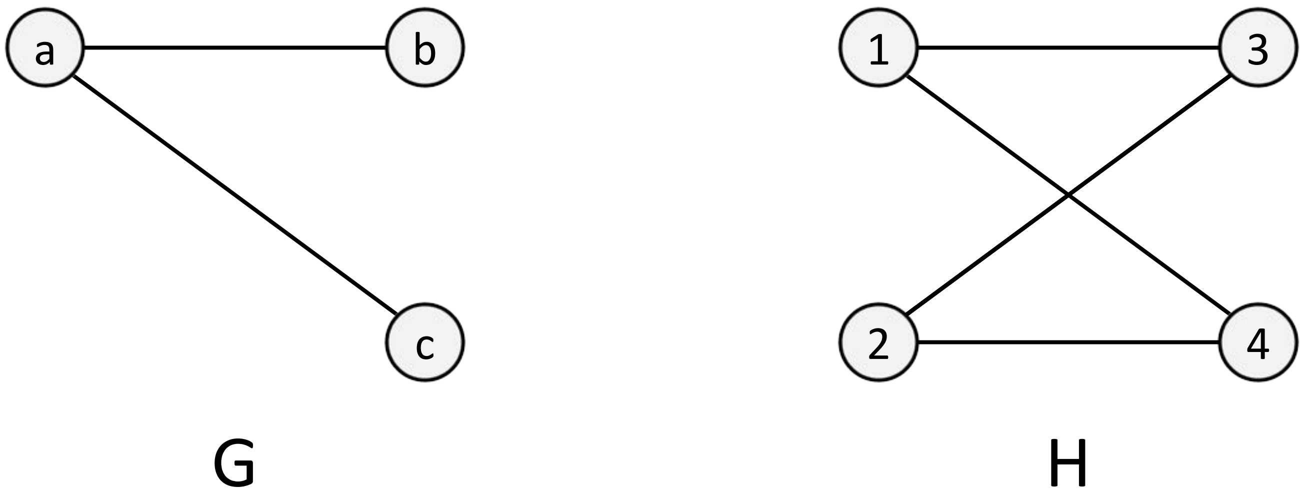
does not need to be surjective!

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

does not need to be injective!

Graph Isomorphism

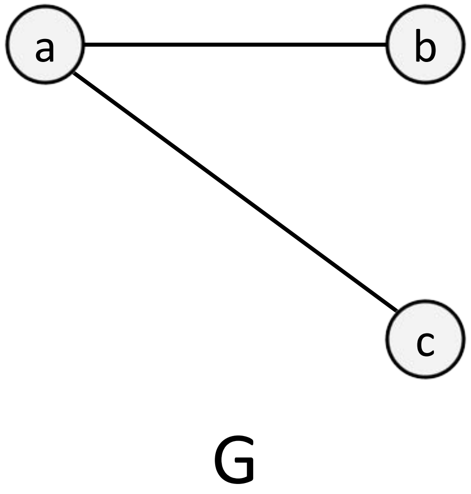
- Graphs $G(V_G,E_G)$ and $H(V_H,E_H)$ are isomorphic iff there is an invertible h from V_G to V_H s.t. $\{x,y\} \in E_G$ iff $\{h(u),h(v)\} \in E_H$
  - We need to find a one-to-one correspondence

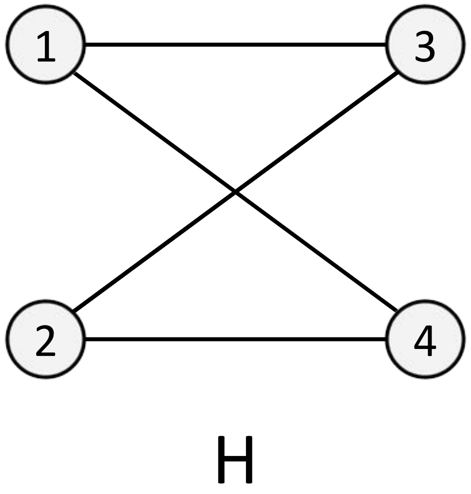


Is there an isomorphism
from G to H ?

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



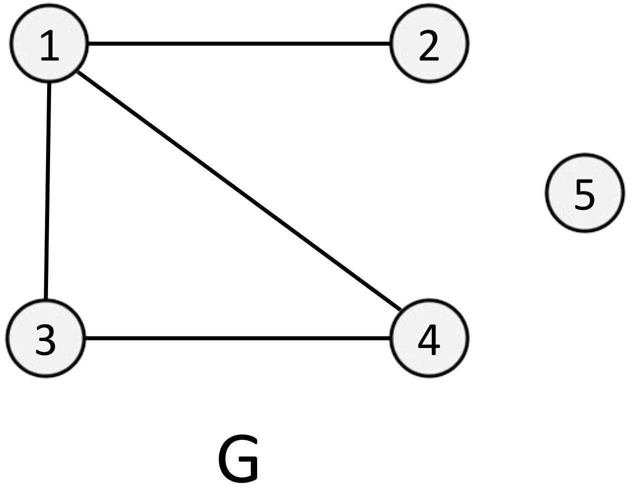


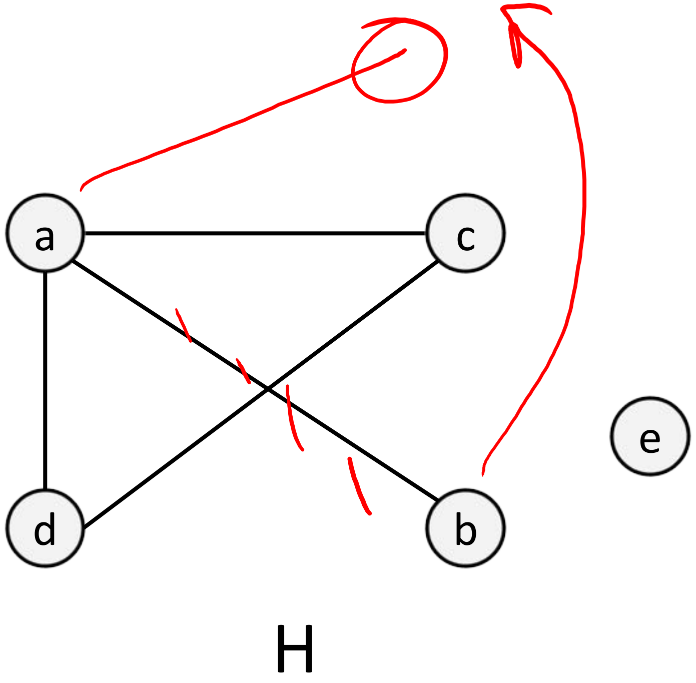
Is there an isomorphism from G to H?

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

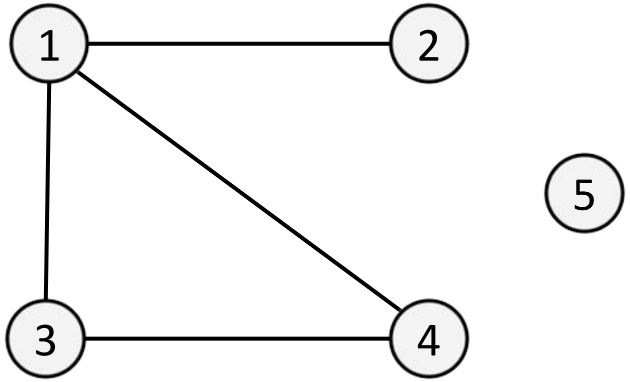




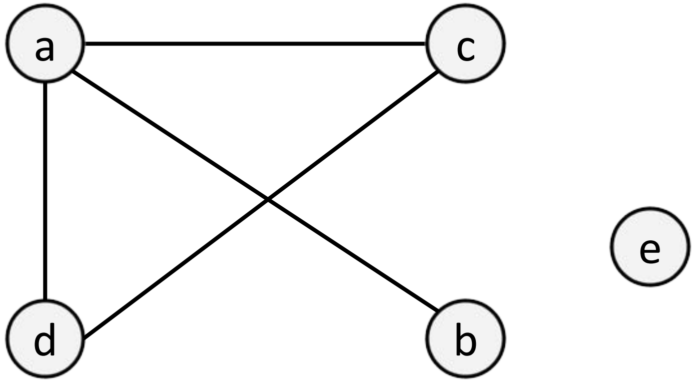
Is there an isomorphism
from G to H?

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

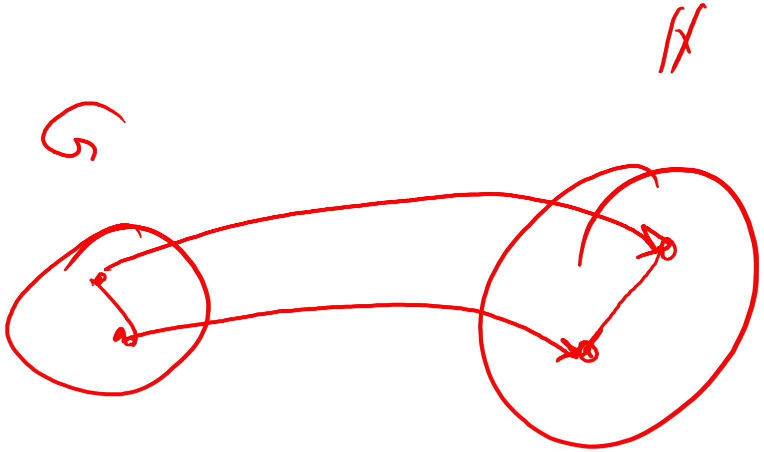
- T2-1: Conjunctive Queries (CQs)
  - CQ equivalence and containment
  - Graph homomorphisms
  - Homomorphism beyond graphs
  - CQ containment
  - CQ minimization
- T2-2: Equivalence Beyond CQs
  - Union of CQs, and inequalities
  - Union of CQs equivalence under bag semantics
  - Tree pattern queries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

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

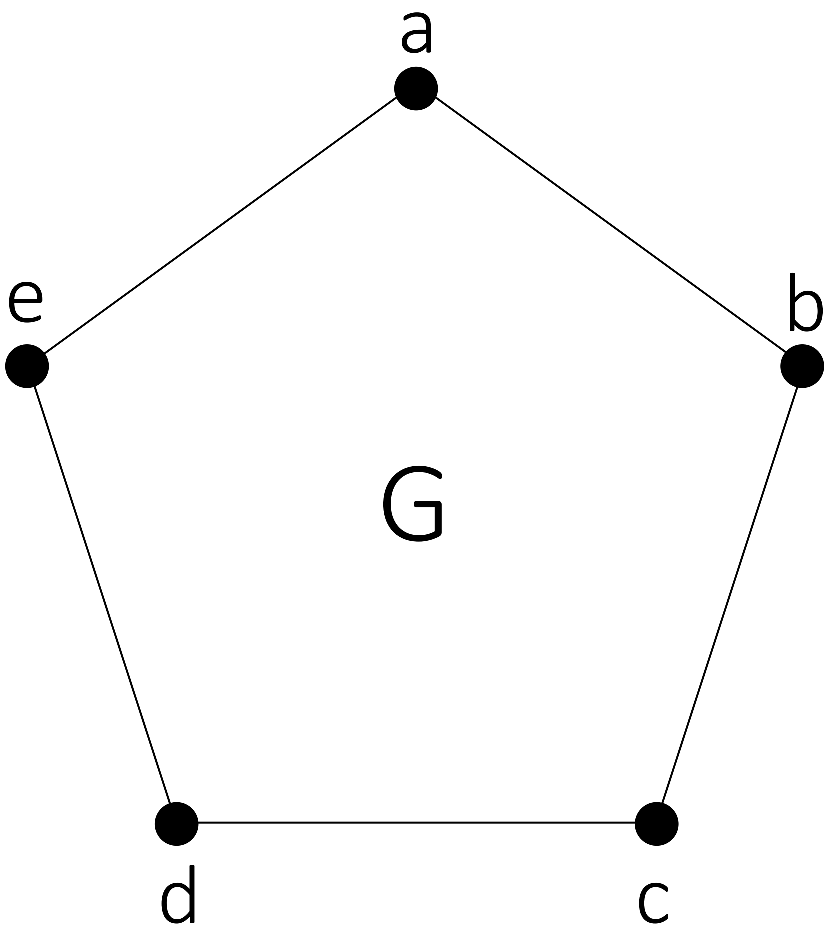
We sometimes write G → H (G ↛ H) if there is a homomorphism (no homomorphism) of G to H



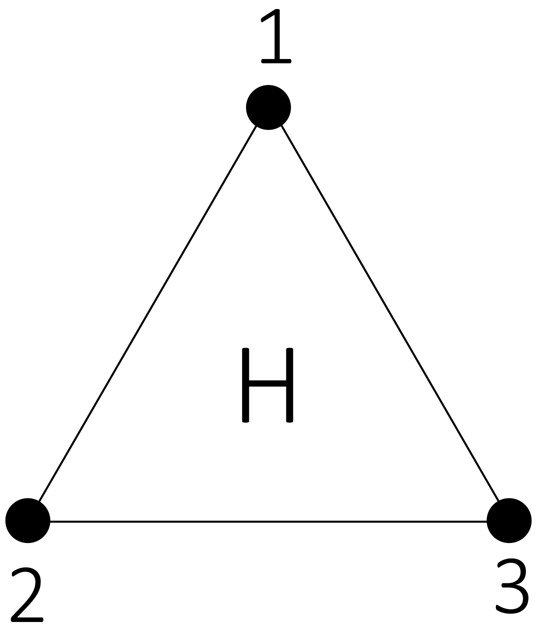
Definition of a homomorphism naturally extends to:

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

An example

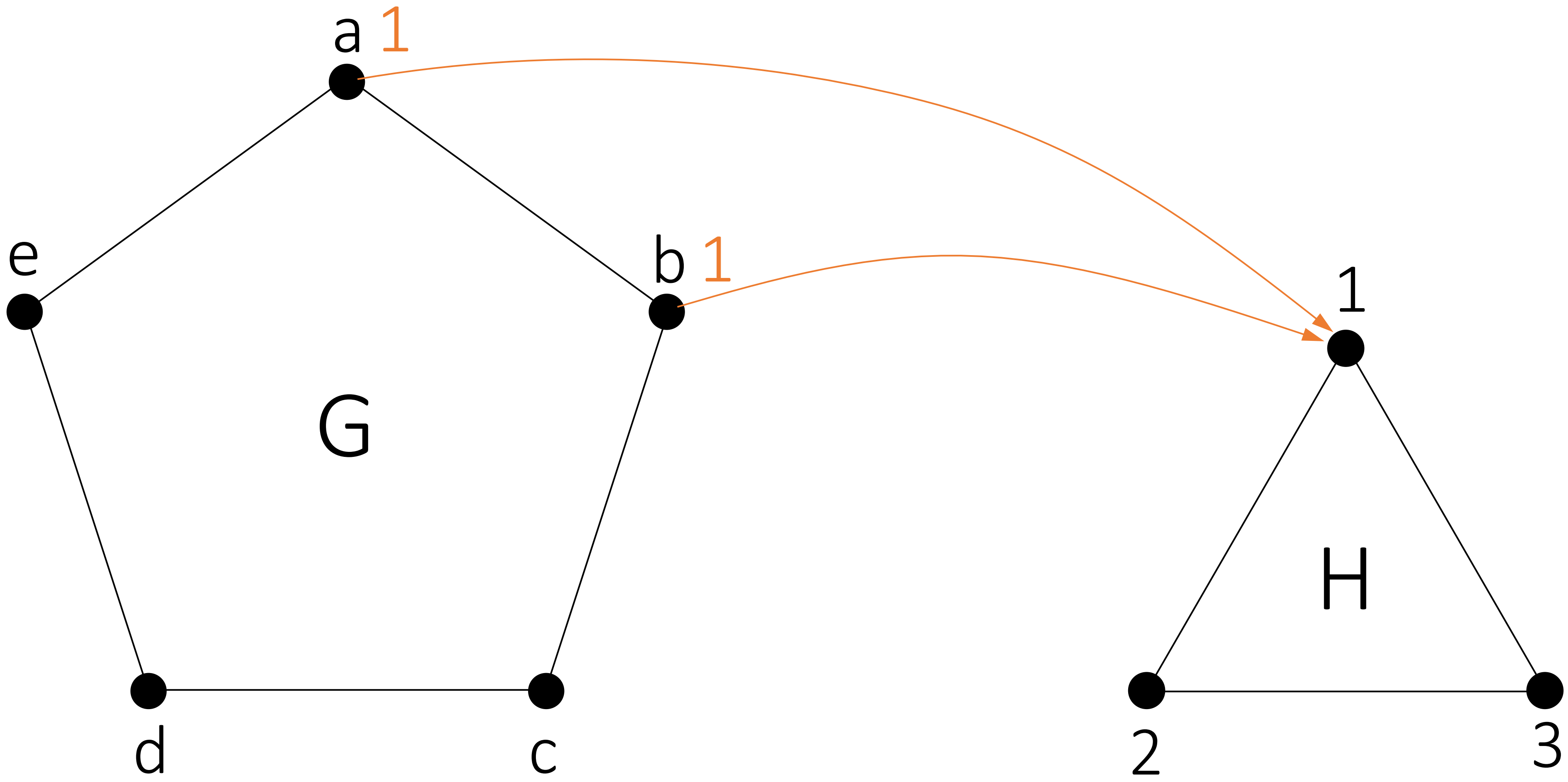


3 "colors" of the vertices



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

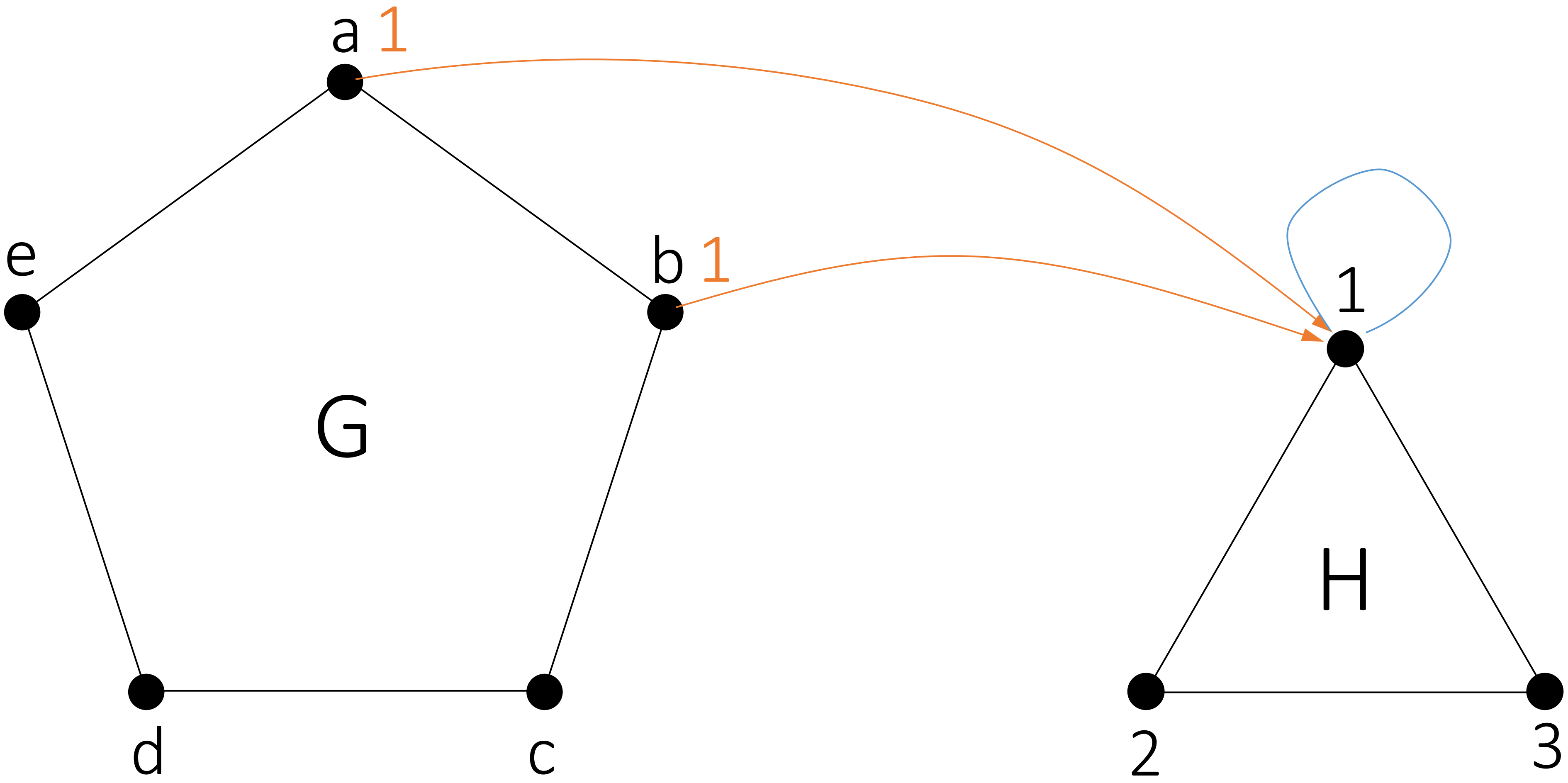
An example



Can this assignment be extended to a homomorphism? ?

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

An example

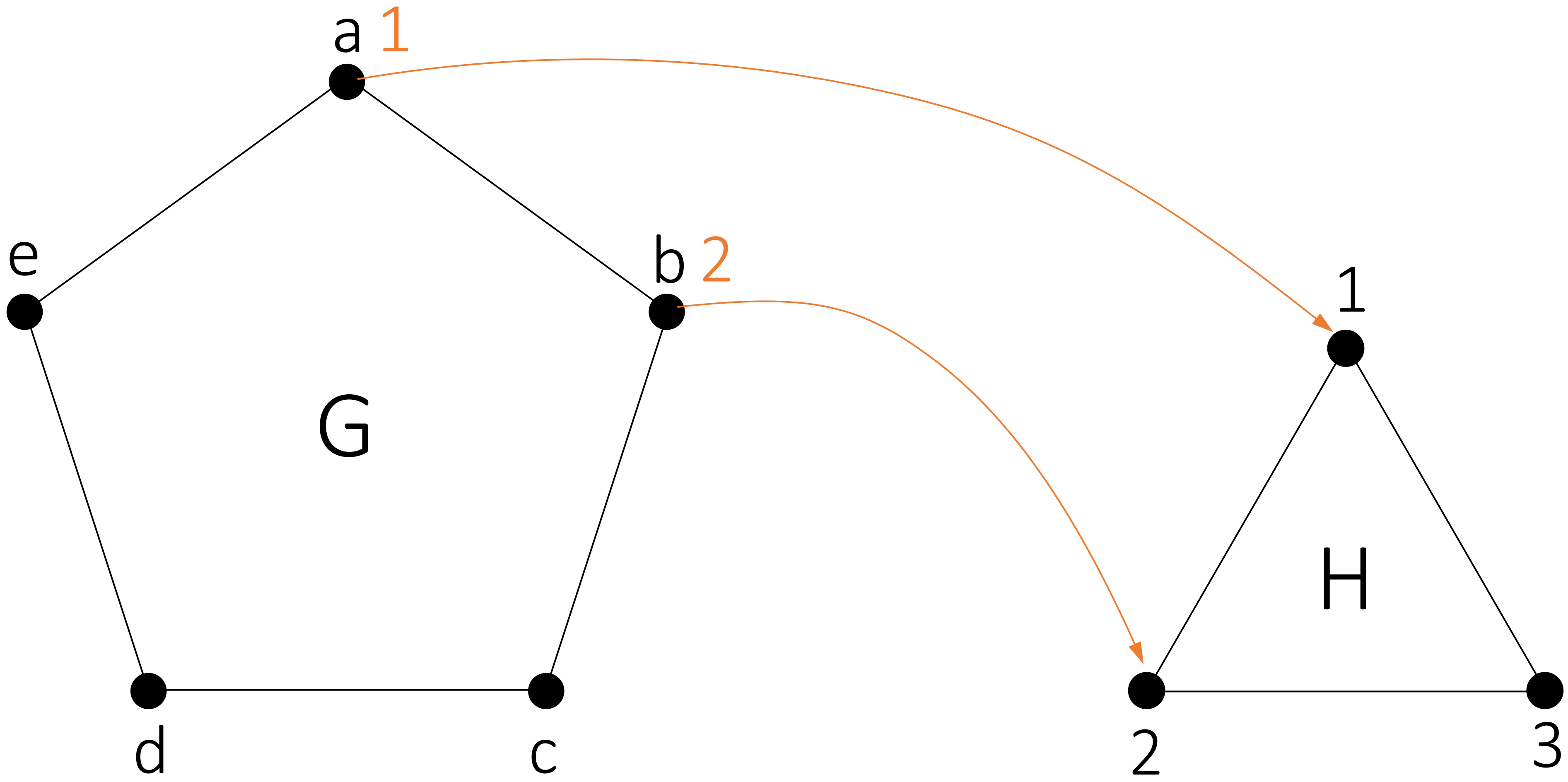


Can this assignment be extended to a homomorphism?

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

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

An example



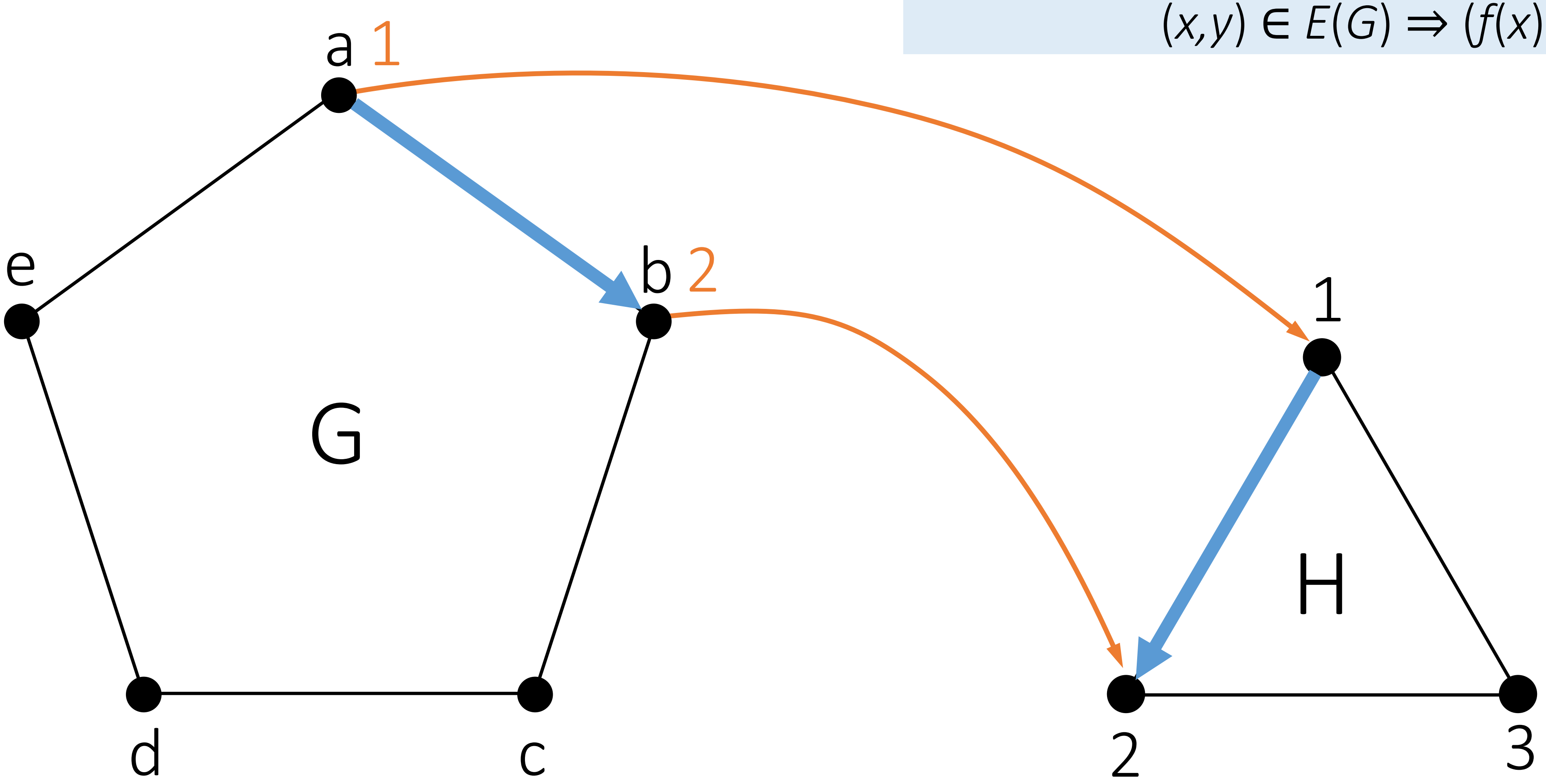
Can this assignment be extended to a homomorphism? ?

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*) → *V*(*H*) s.t. that

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



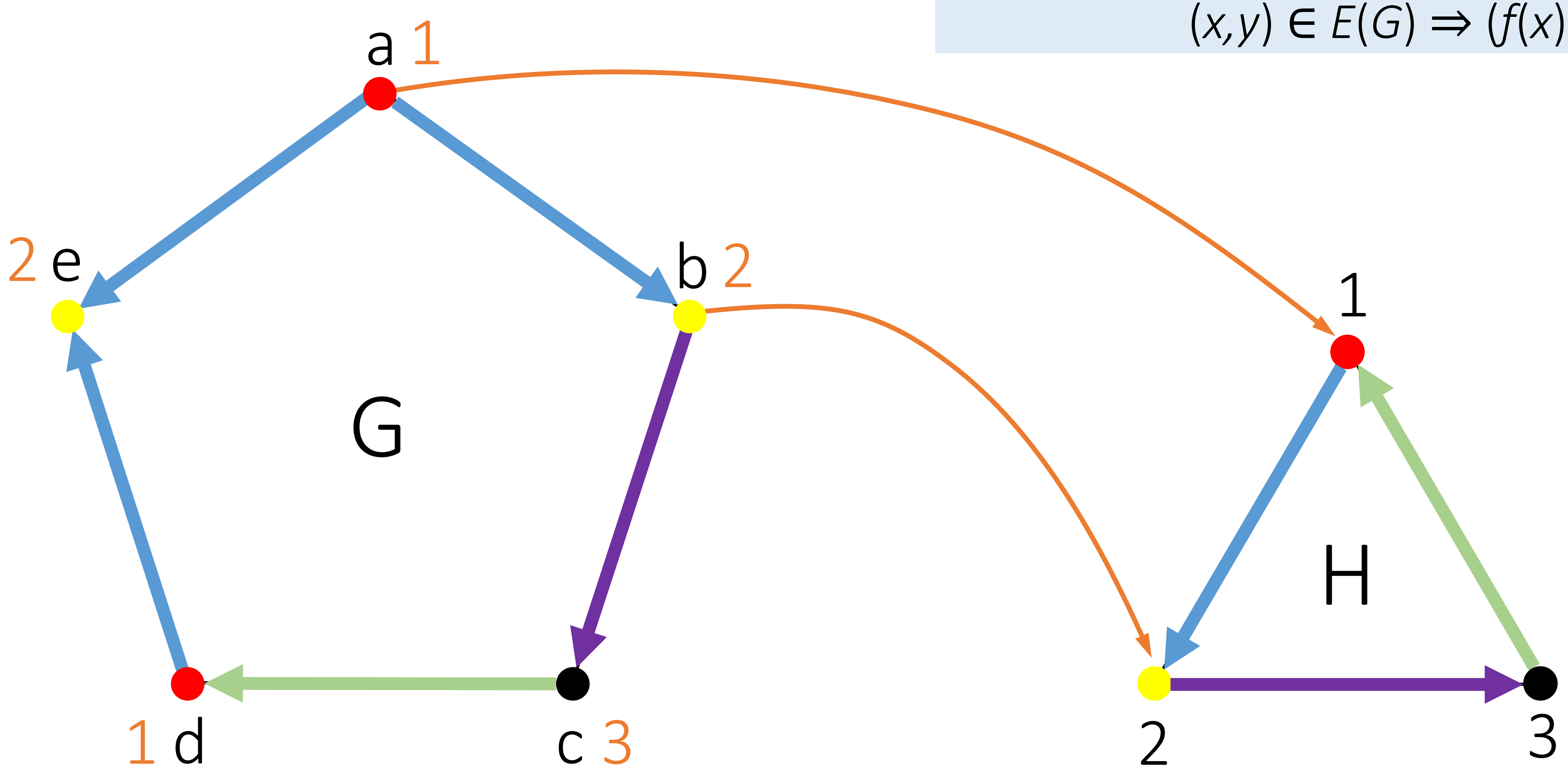
Can this assignment be extended to a homomorphism? ?

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*) → *V*(*H*) s.t. that

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

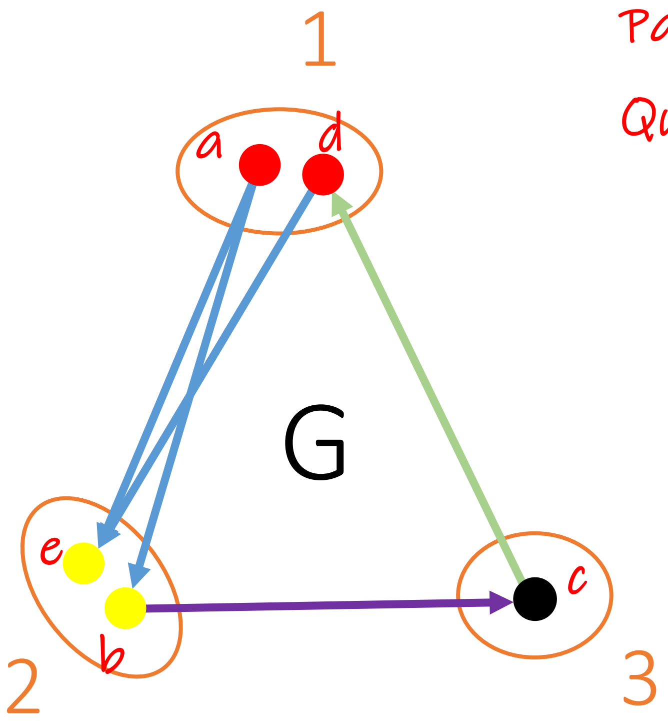


Based upon an example from Rick Brewster's Graph homomorphism tutorial, 2006

An example

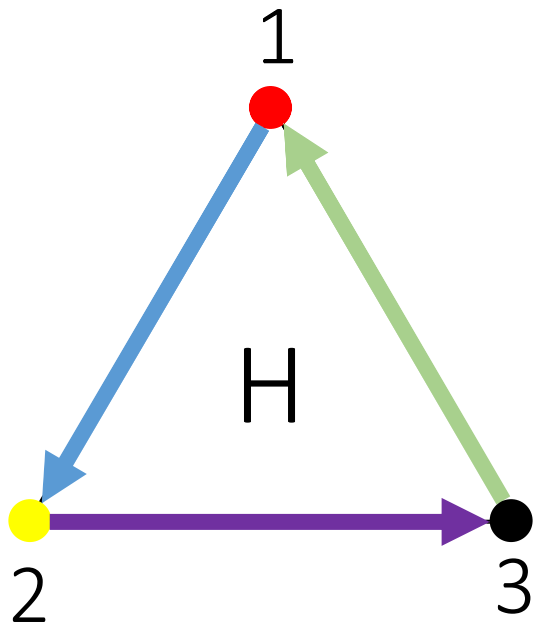
Basically a partitioning problem!

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



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

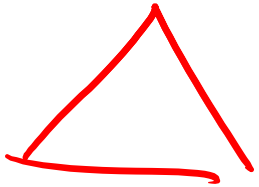
Quotient set: {[a], [b], [c]}



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)



?

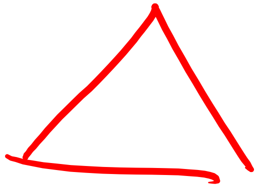
Some observations

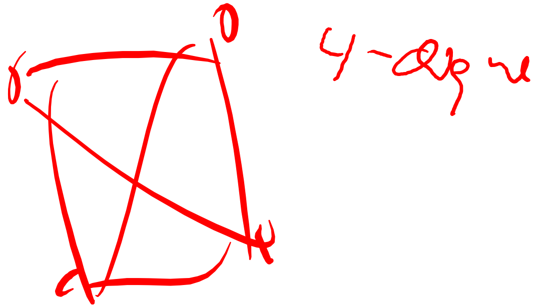
When does G → K_3 hold? (K_3 = 3-clique = triangle)

iff G is 3-colorable

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

?

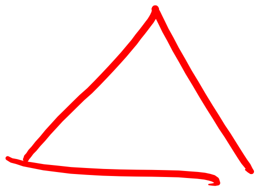




More on 3-coloring: https://en.wikipedia.org/wiki/Graph_coloring#Computational_complexity

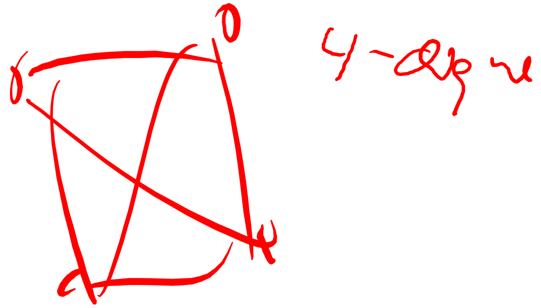
Some observations

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.

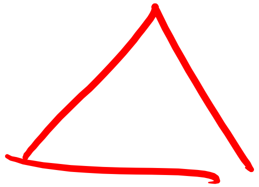
What is the complexity of testing for the existence of a homomorphism (in the size of G)?

?

More on 3-coloring: https://en.wikipedia.org/wiki/Graph_coloring#Computational_complexity

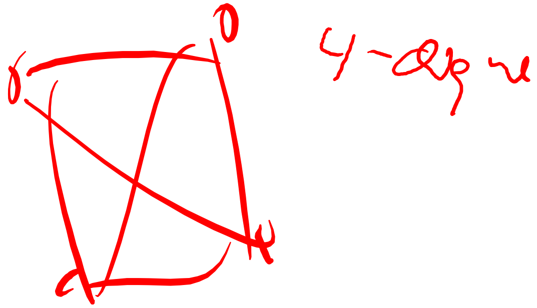
Some observations

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

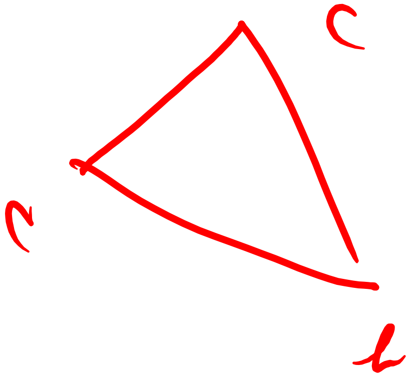
Thus homomorphisms generalize colorings:
Notation: G → H is an H-coloring of G.

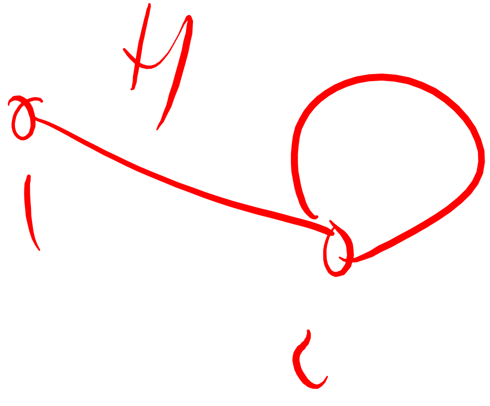
What is the complexity of testing for the existence of a homomorphism (in the size of G)?

NP-complete

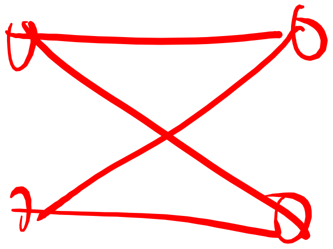
More on 3-coloring: https://en.wikipedia.org/wiki/Graph_coloring#Computational_complexity

The complexity of H-coloring

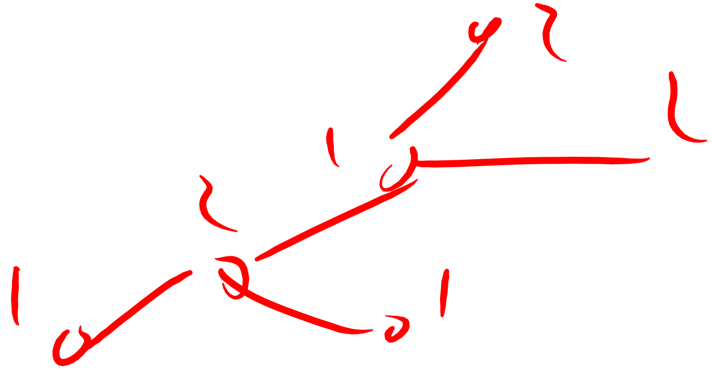




H-coloring:
Let H be a fixed graph.
Instance: A graph G.
Question: Does G admit an H-coloring?



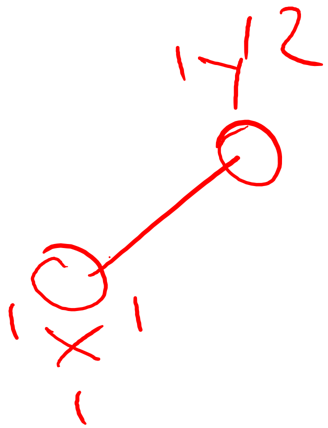
Theorem [Hell, Nesetril'90]:
If H is bipartite or contains a self-loop, then H-coloring is polynomial time solvable; otherwise, H is NP-complete.

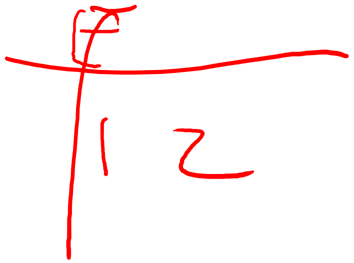


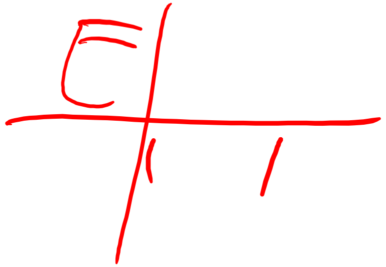
[Hell, Nesetril'90]: Hell, Nešetřil. On the complexity of H-coloring. Journal of Combinatorial Theory, 1990. https://doi.org/10.1016/0095-8956(90)90132-J

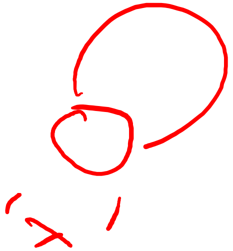
Repeated variable names

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









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

? 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)

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

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?

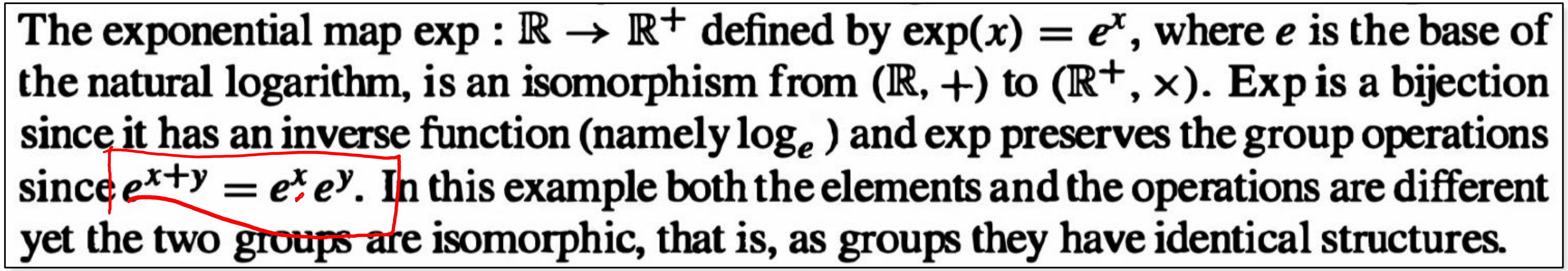
?

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
  - the positive real numbers with multiplication $(\mathbb{R}^+,\cdot)$
  - It is even an isomorphism!

$h(x+y) = h(x) \cdot h(y)$

$h: (\mathbb{R},+) \longrightarrow (\mathbb{R}^+,\cdot)$



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 $\boxed{e^{x+y} = e^x \cdot e^y}$. 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?

?

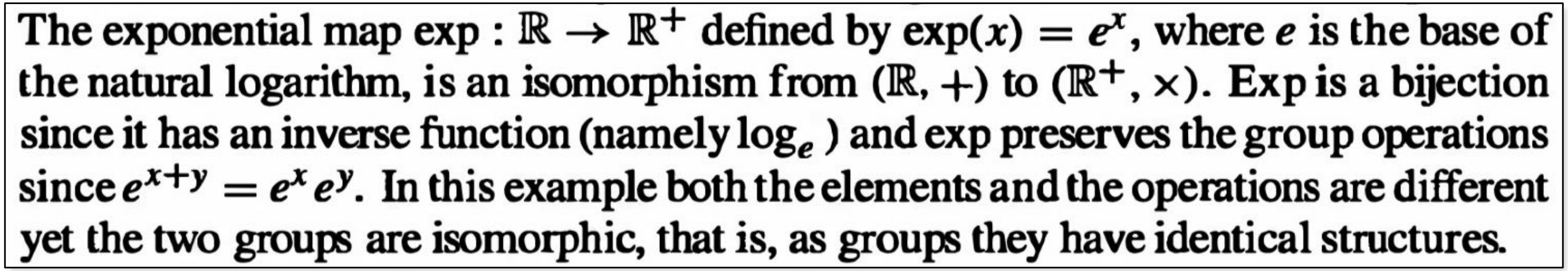
Paragraph screenshot from p.37 in 2004 - Dummit, Foote - Abstract algebra (book, 3rd ed). https://www.wiley.com/en-us/Abstract+Algebra%2C+3rd+Edition-p-9780471433347

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
  - the positive real numbers with multiplication $(\mathbb{R}^+,\cdot)$
  - It is even an isomorphism!

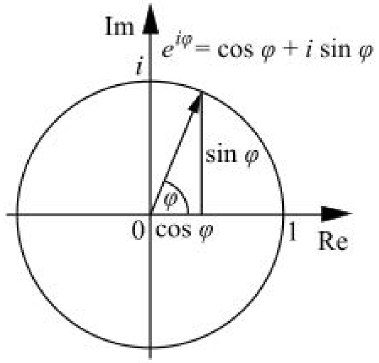
$h(x+y) = h(x) \cdot h(y)$

$h : (\mathbb{R},+) \longrightarrow (\mathbb{R}^+,\cdot)$



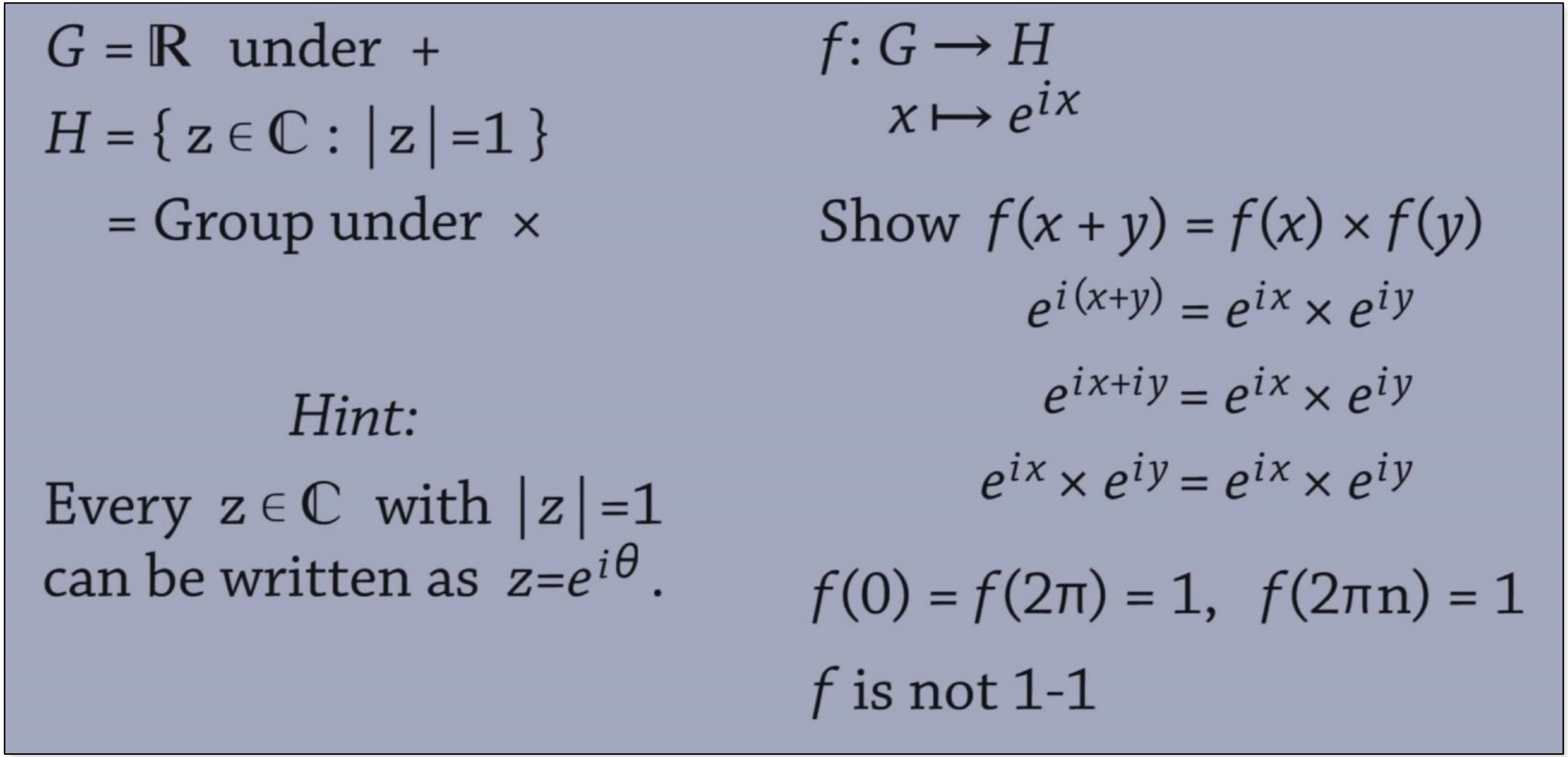
The exponential map $\exp : \mathbb{R} \to \mathbb{R}^+$ defined by $\exp(x) = e^x$, where e is the base of the natural logarithm, is an isomorphism from $(\mathbb{R}, +)$ to $(\mathbb{R}^+, \times)$. Exp is a bijection since it has an inverse function (namely $\log_e$) and exp preserves the group operations 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 $f(x+y) = f(x) \times f(y)$

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

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

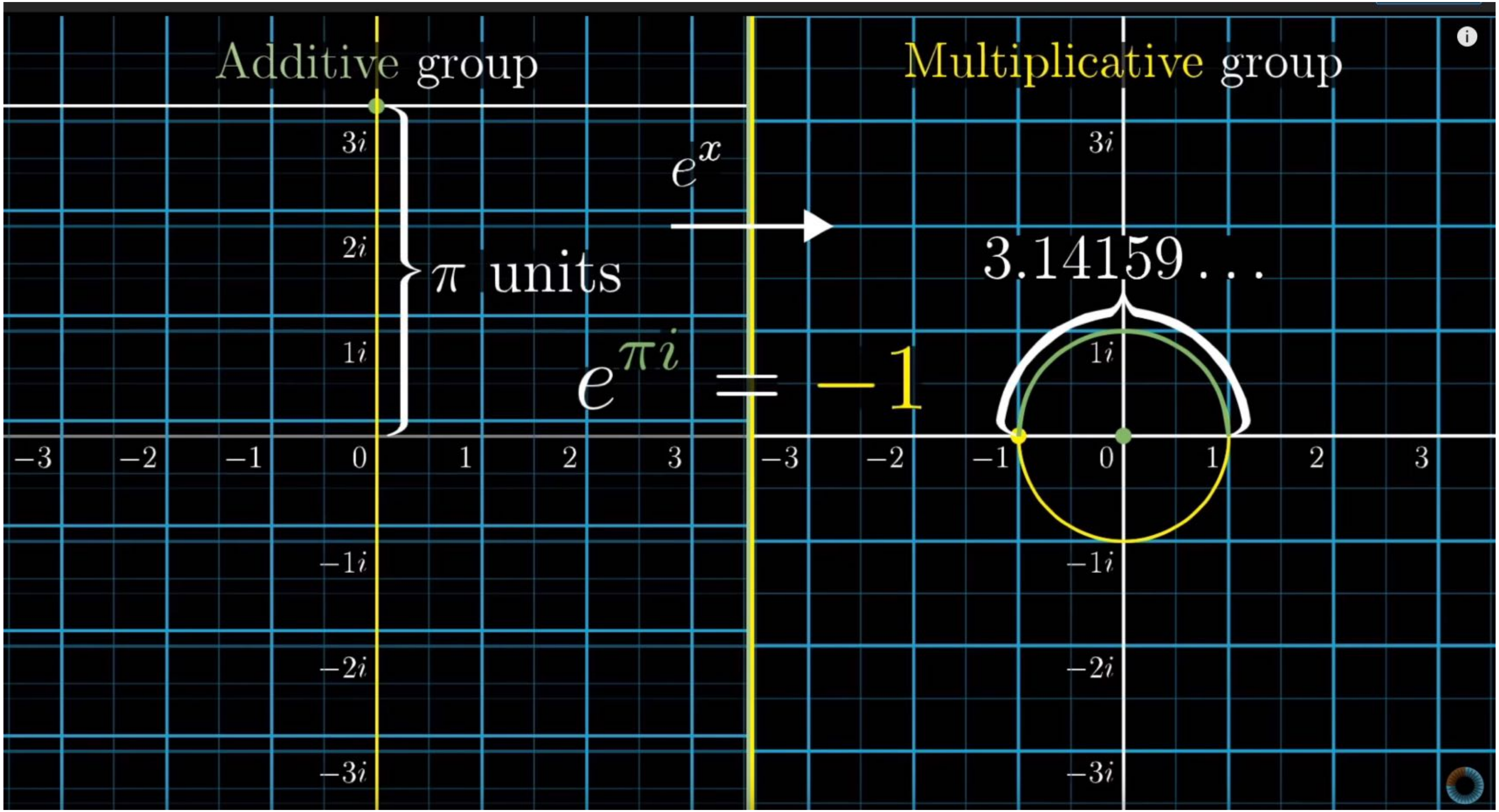
$$e^{ix} \times e^{iy} = e^{ix} \times e^{iy}$$

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

f is not 1-1

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

Example: from addition to multiplication



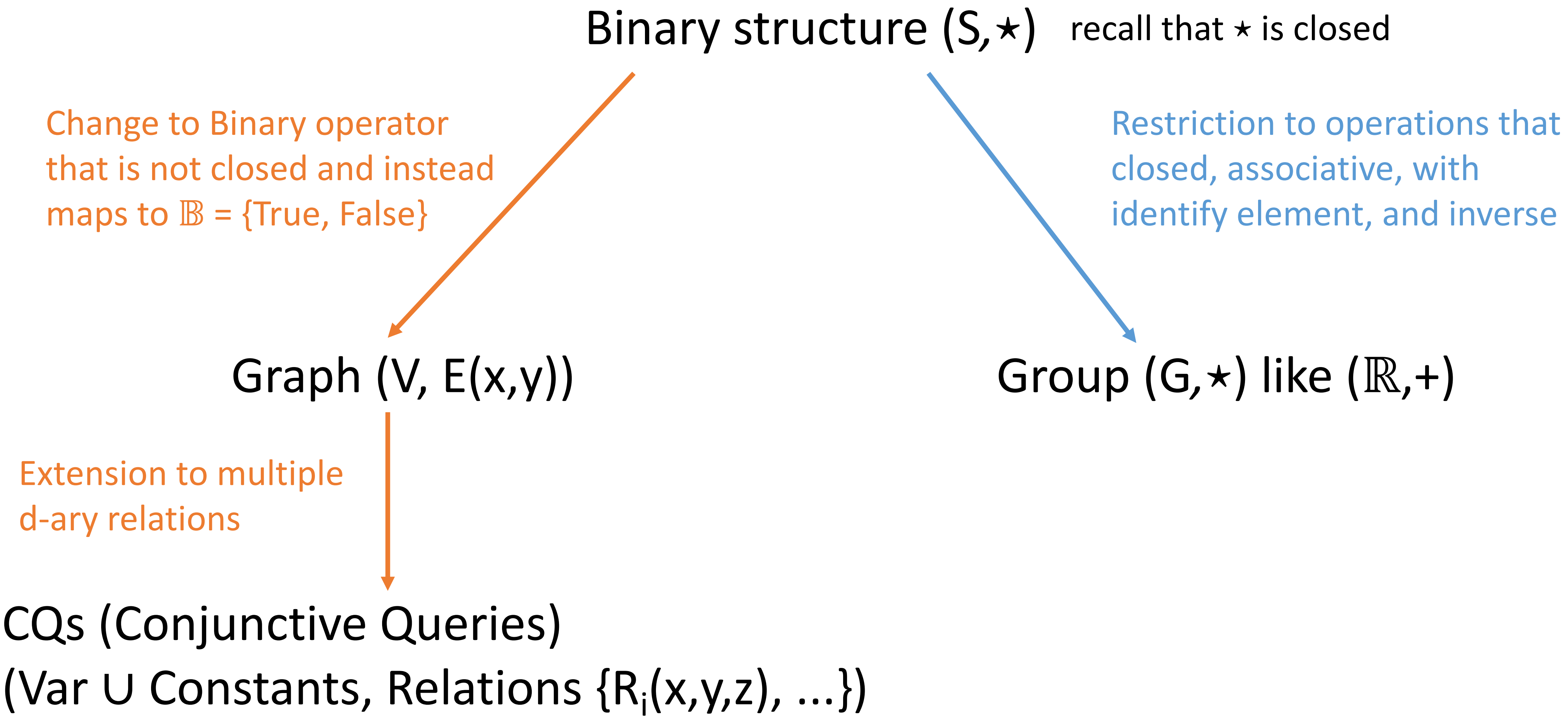
Source: 3blue1brown. Euler's formula with introductory group theory, 2017: https://www.youtube.com/watch?v=mvmuCPvRoWQ

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



Binary structure $(S,\star)$ recall that $\star$ is closed

Change to Binary operator that is not closed and instead maps to $\mathbb{B}$ = {True, False}

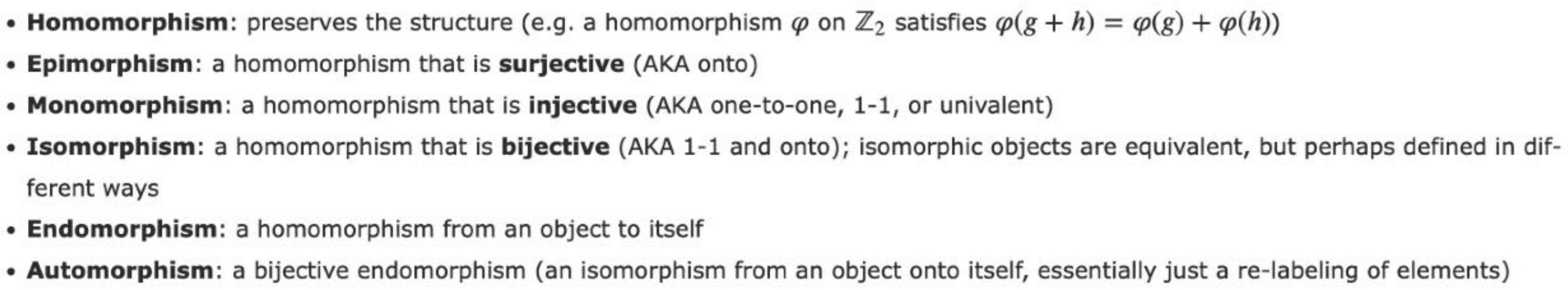
Restriction to operations that closed, associative, with identify element, and inverse

Graph (V, E(x,y))

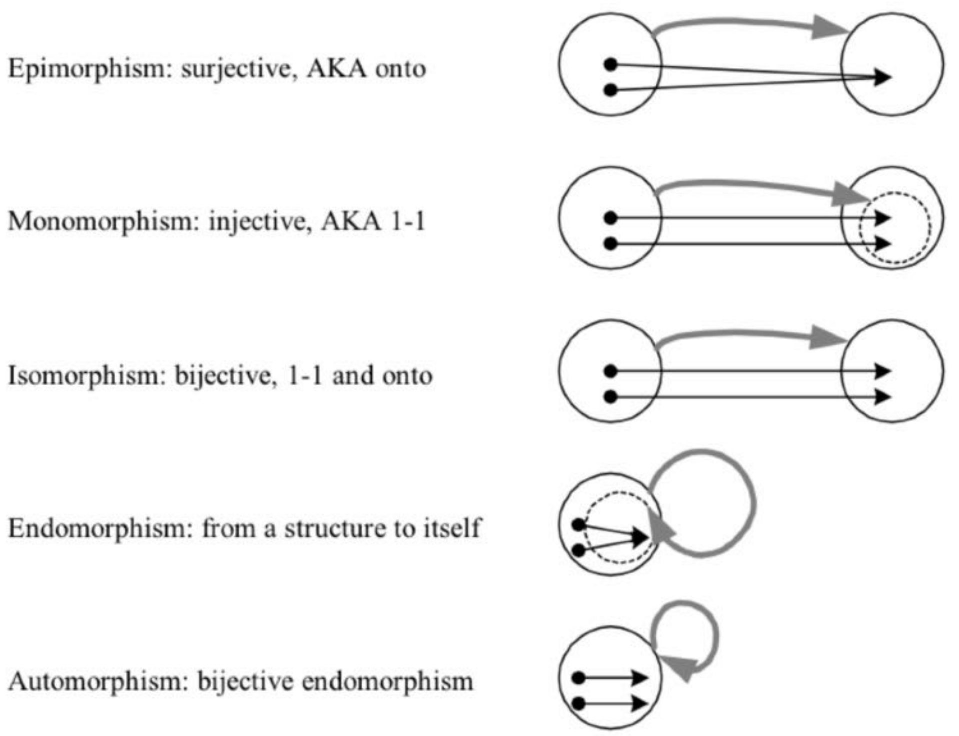
Group $(G,\star)$ like $(\mathbb{R},+)$

Extension to multiple d-ary relations

CQs (Conjunctive Queries)
(Var U Constants, Relations $\{R_i(x,y,z), \ldots\}$)



- **Homomorphism**: preserves the structure (e.g. a homomorphism φ on $\mathbb{Z}_2$ satisfies $\varphi(g+h)=\varphi(g)+\varphi(h)$)
- **Epimorphism**: a homomorphism that is **surjective** (AKA onto)
- **Monomorphism**: a homomorphism that is **injective** (AKA one-to-one, 1-1, or univalent)
- **Isomorphism**: a homomorphism that is **bijective** (AKA 1-1 and onto); isomorphic objects are equivalent, but perhaps defined in different ways
- **Endomorphism**: a homomorphism from an object to itself
- **Automorphism**: a bijective endomorphism (an isomorphism from an object onto itself, essentially just a re-labeling of elements)


Epimorphism: surjective, AKA onto
Monomorphism: injective, AKA 1-1
Isomorphism: bijective, 1-1 and onto
Endomorphism: from a structure to itself
Automorphism: bijective endomorphism

Source: https://www.mathphysicsbook.com/mathematics/mathematical-structures/defining-mathematical-structures-and-mappings/

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

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

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

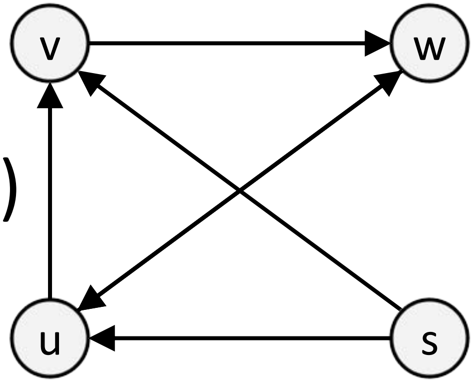
need to be same relation!

Example

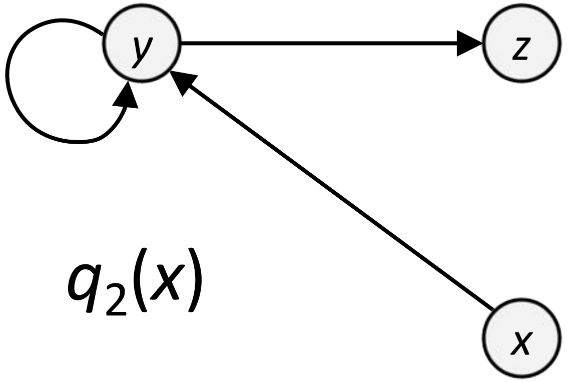
q_1 :- $R(s,u), R(u,w), R(s,v), R(v,w), R(u,v)$

q_2 :- $R(x,y), R(y,y), R(y,z)$

$q_1(x)$



$h_{1\rightarrow 2}$= ?



$q_2(x)$

Query homomorphisms

A homomorphism h from Boolean q_1 to q_2 is a function

h: var(q_1) → var(q_2) ∪ const(q_2) such that:

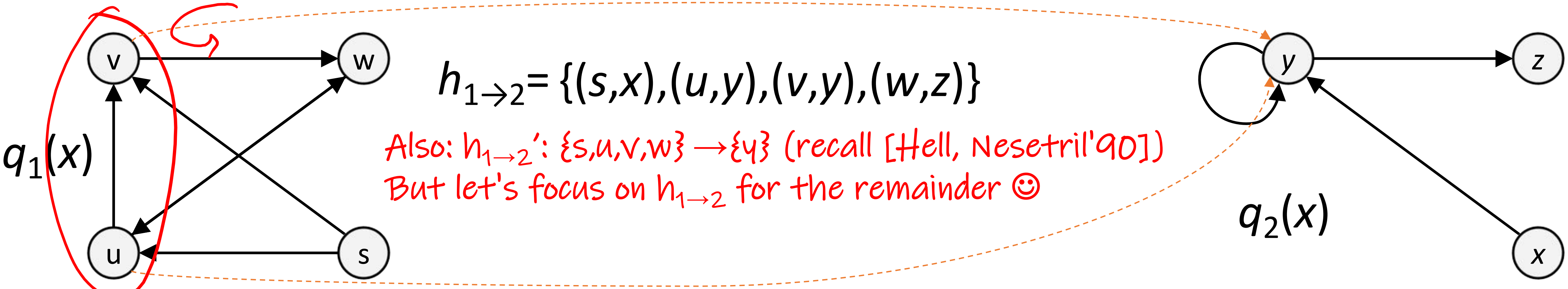
for every atom $R(x_1,x_2,\ldots)$ in q_1, there is an atom $R(h(x_1), h(x_2), \ldots)$ in q_2

need to be same relation!

<u>Example</u>

q_1 :- $R(s,u), R(u,w), R(s,v), R(v,w), R(u,v)$

q_2 :- $R(x,y), R(y,y), R(y,z)$



$h_{1\to 2}$= {(s,x),(u,y),(v,y),(w,z)}

Also: $h_{1\to 2}'$: {s,u,v,w} → {y} (recall [Hell, Nesetril'90])

But let's focus on $h_{1\to 2}$ for the remainder ☺

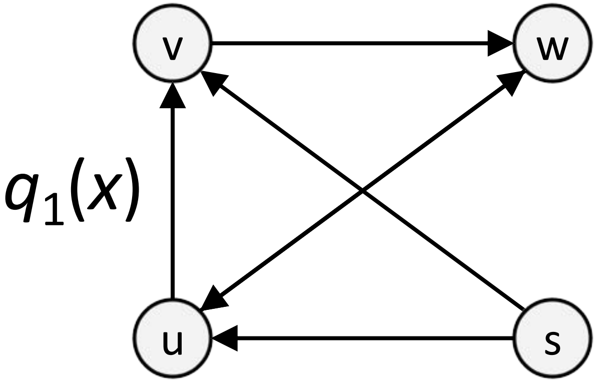
Query homomorphisms

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

Example

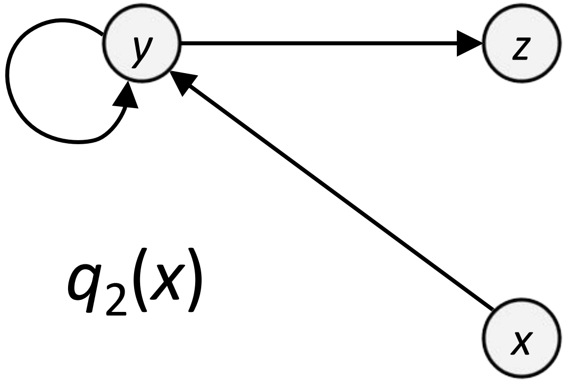
q_1 :- $R(s,u), R(u,w), R(s,v), R(v,w), R(u,v)$

q_2 :- $R(x,y), R(y,y), R(y,z)$



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

$h_{2\to1}$: ?



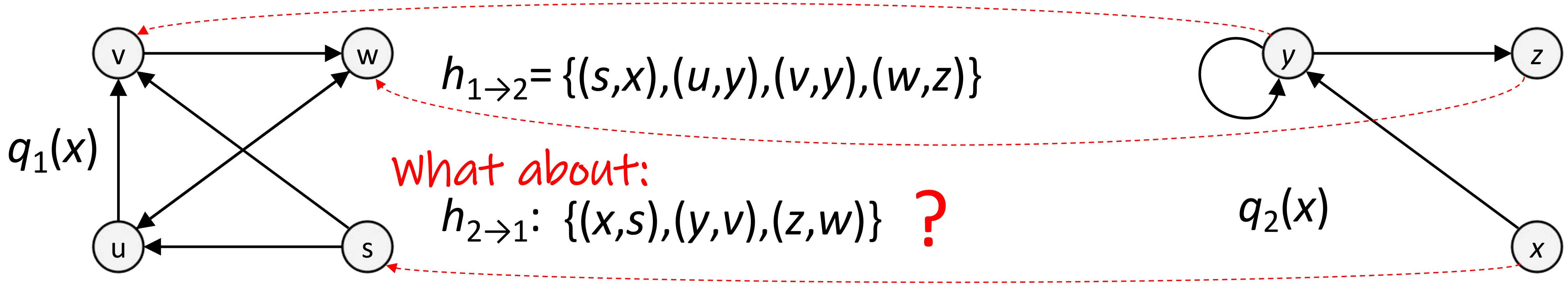
Query homomorphisms

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

Example

q_1 :- $R(s,u), R(u,w), R(s,v), R(v,w), R(u,v)$

q_2 :- $R(x,y), R(y,y), R(y,z)$



$q_1(x)$

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

What about:

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

$q_2(x)$

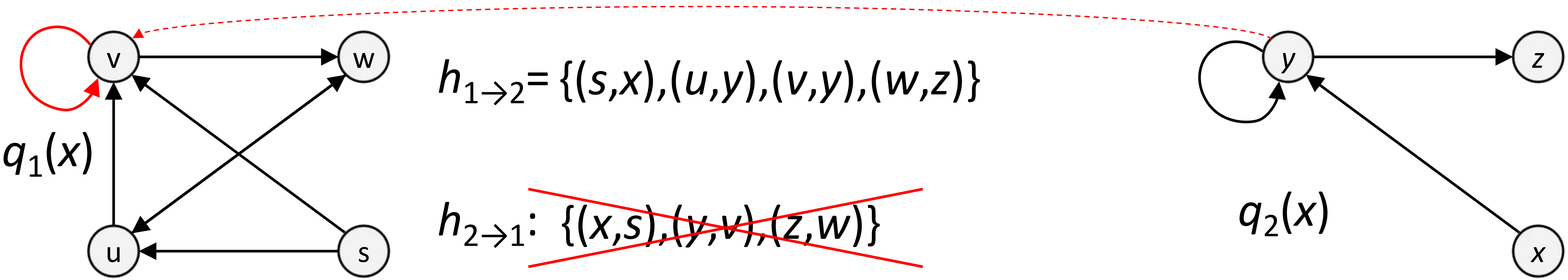
Query homomorphisms

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

Example

q_1 :- $R(s,u)$, $R(u,w)$, $R(s,v)$, $R(v,w)$, $R(u,v)$, R(v,v)

q_2 :- $R(x,y)$, $R(y,y)$, $R(y,z)$



$h_{1\rightarrow 2}= \{(s,x),(u,y),(v,y),(w,z)\}$

$h_{2\rightarrow 1}$: ~~$\{(x,s),(y,v),(z,w)\}$~~

Query homomorphisms and containment

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

E(1,2)

Compare to our earlier example:

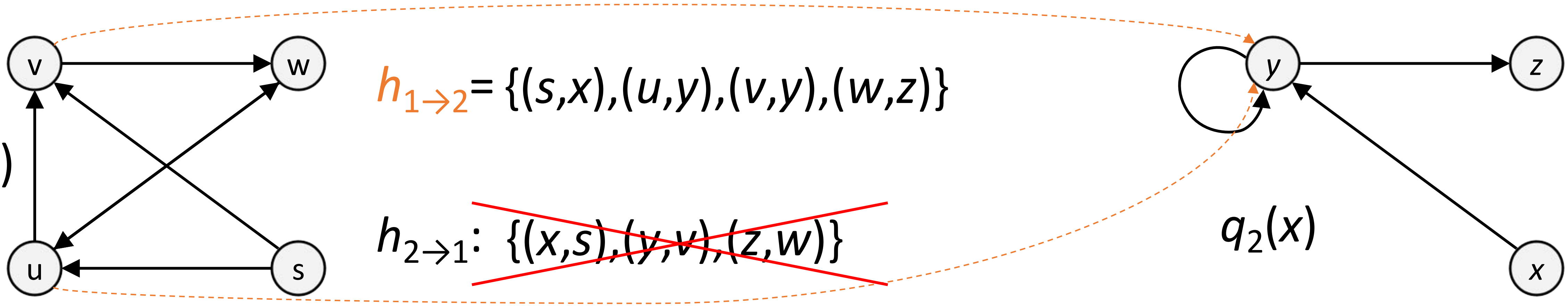
$\exists x.\exists y.\ E(x,y) \;\underset{\Leftarrow}{\Rightarrow}\; \exists x.\ E(x,x)$?

E(1,1)

Example

q_1 :- $R(s,u), R(u,w), R(s,v), R(v,w), R(u,v)$

q_2 :- $R(x,y), R(y,y), R(y,z)$



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

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

Query homomorphisms and containment

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

E(1,2)
True

Compare to our earlier example:

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

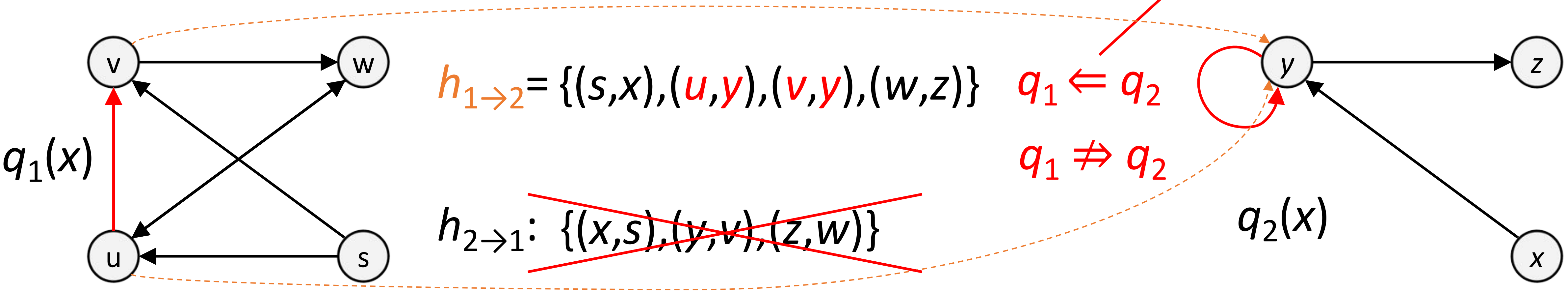
E(1,1)
False

Example

q_1 :- $R(s,u), R(u,w), R(s,v), R(v,w), R(u,v)$

q_2 :- $R(x,y), R(y,y), R(y,z)$

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



$q_1(x)$

$h_{1\rightarrow 2}$= {(s,x),(u,y),(v,y),(w,z)}

$q_1 \Leftarrow q_2$

$q_1 \nRightarrow q_2$

$h_{2\rightarrow 1}$: ~~{(x,s),(y,v),(z,w)}~~

$q_2(x)$

Overview: "All homomorphisms" in one slide

"q_1-coloring of G "

Constraint Satisfaction Problems (CSP)

NP-C in size of G

"G-coloring of q_1 "

Query evaluation

$G \vDash q_2$

PTIME in size of G

$$\begin{tikzcd}
 & G \arrow[dl, "h"'] & \\
q_1 \arrow[rr, "h"] & & q_2 \arrow[ul, "h"']
\end{tikzcd}$$

Query containment

$q_1 \supseteq q_2$

$q_1 \Leftarrow q_2$

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

$q_2(x)$:- $R(x,y)$, $R(y,y)$, $R(y,z)$

$D_c[q_2]$ = ?

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

$q_2(x)$:- $R(x,y)$, $R(y,y)$, $R(y,z)$

$D_c[q_2]$ = {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)}

Var		Const
x	→	1
y	→	2
z	→	3

Just treat each variable as different constant ☺

[Chandra and Merlin 1977]

THEOREM (Query Containment)

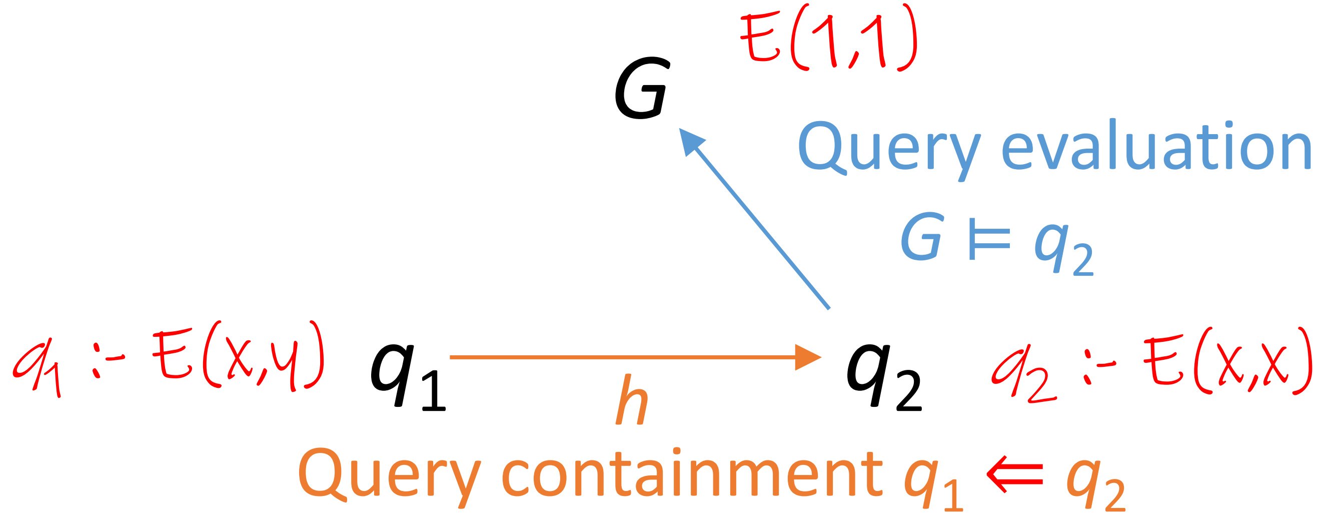
Given two Boolean CQs q_1, q_2, the following statements are equivalent:

1) $q_1 \Leftarrow q_2$ $\quad$ $(q_1 \supseteq q_2)$

2) There is a homomorphism $h_{1\rightarrow 2}$ from q_1 to q_2

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

We will look at 2) ⇒ 1),
and it is similar to 2) ⇒ 3)



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 $h_{1\to 2}$, then for any D: $q_1(\mathrm{D}) \Leftarrow q_2(\mathrm{D})$

1. For $q_2(\mathrm{D})$ to hold, there is a valuation v s.t. $v(q_2) \in \mathrm{D}$
2. We will show that the composition $g = v \circ h$ is a valuation for q_1

$g = v \circ h$

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

E(1,1)

$$\begin{tikzcd} & G & \\ q_1 \arrow[ur, "g"] \arrow[rr, "h"'] & & q_2 \arrow[ul, "v"'] \end{tikzcd}$$

Query evaluation

$G \models q_2$

q_1 :- E(x,y)

q_2 :- E(x,x)

Query containment $q_1 \Leftarrow q_2$

Chandra, Merlin. "Optimal implementation of conjunctive queries in relational data bases." STOC 1977. https://doi.org/10.1145/800105.803397

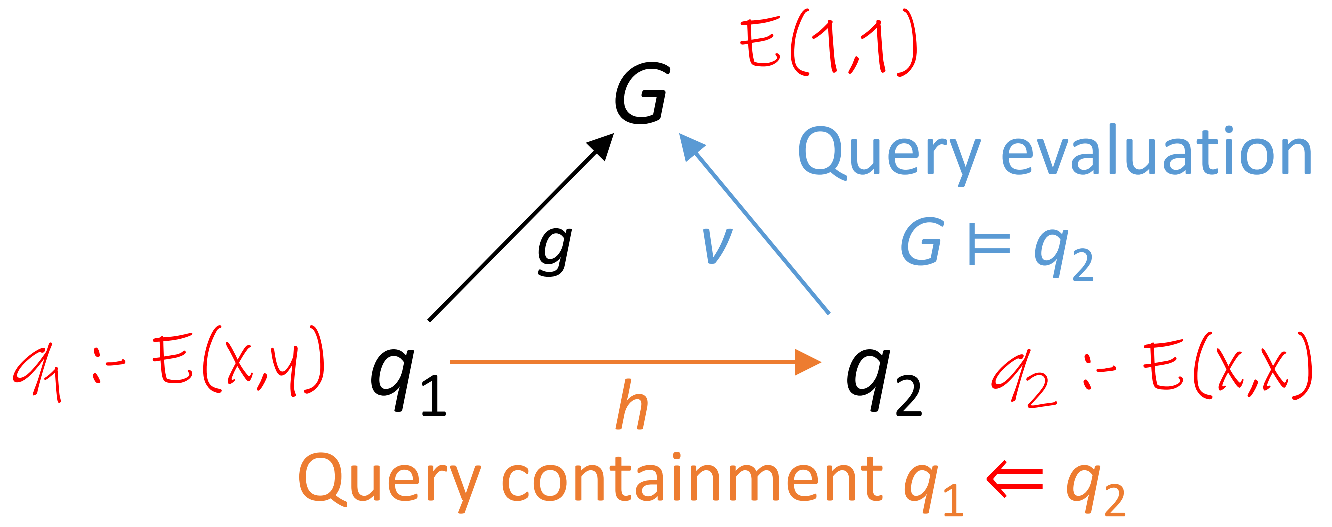
[Chandra and Merlin 1977]

We show: If there is a homomorphism $h_{1\to2}$, then for any D: $q_1(\mathrm{D}) \Leftarrow q_2(\mathrm{D})$

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

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

QED ☺



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 $h_{1\rightarrow 2}$, then for any D: q_1(D) $\Leftarrow$ q_2(D)

1. For q_2(D) to hold, there is a valuation v s.t. $v(q_2) \in$ D

2. We will show that the composition $g = v \circ h$ is a valuation for q_1

    2a. By definition of h, for every $R(x_1,x_2,\ldots)$ in q_1, $R(h(x_1),h(x_2),\ldots)$ in q_2

    2b. By definition of v, for every $R(x_1,x_2,\ldots)$ in q_1, $R(v(h(x_1)),v(h(x_2)),\ldots)$ in D

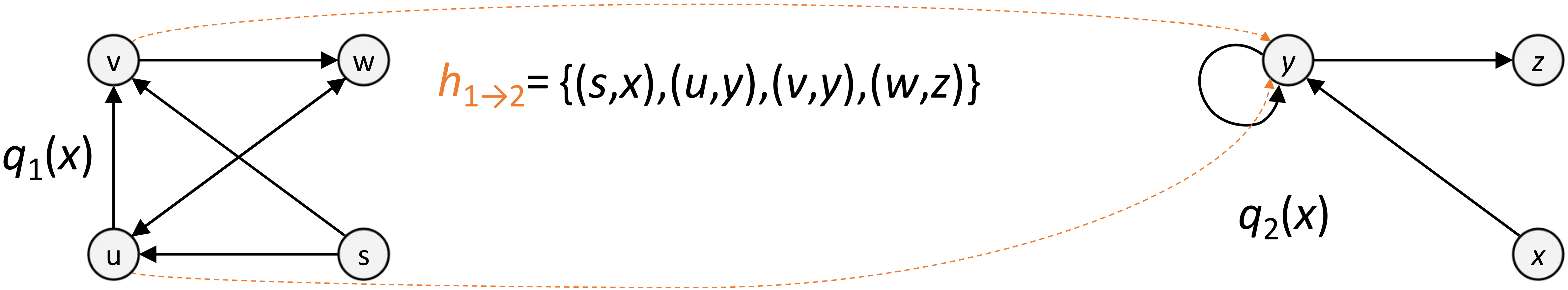
g=$v \circ h$

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

<u>Example</u>

q_1 :- $R(s,u)$, $R(u,w)$, $R(s,v)$, $R(v,w)$, $R(u,v)$

q_2 :- $R(x,y)$, $R(y,y)$, $R(y,z)$



$h_{1\rightarrow 2}$= {(s,x),(u,y),(v,y),(w,z)}

[Chandra and Merlin 1977]

We show: If there is a homomorphism $h_{1\rightarrow2}$, then for any D: q_1(D) ⇐ q_2(D)

1. For q_2(D) to hold, there is a valuation v s.t. $v(q_2) \in$ D

2. We will show that the composition $g = v \circ h$ is a valuation for q_1

   2a. By definition of h, for every $R(x_1,x_2,\ldots)$ in q_1, $R(h(x_1),h(x_2),\ldots)$ in q_2

   2b. By definition of v, for every $R(x_1,x_2,\ldots)$ in q_1, $R(v(h(x_1)),v(h(x_2)),\ldots)$ in D

g=$v \circ h$

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

Example

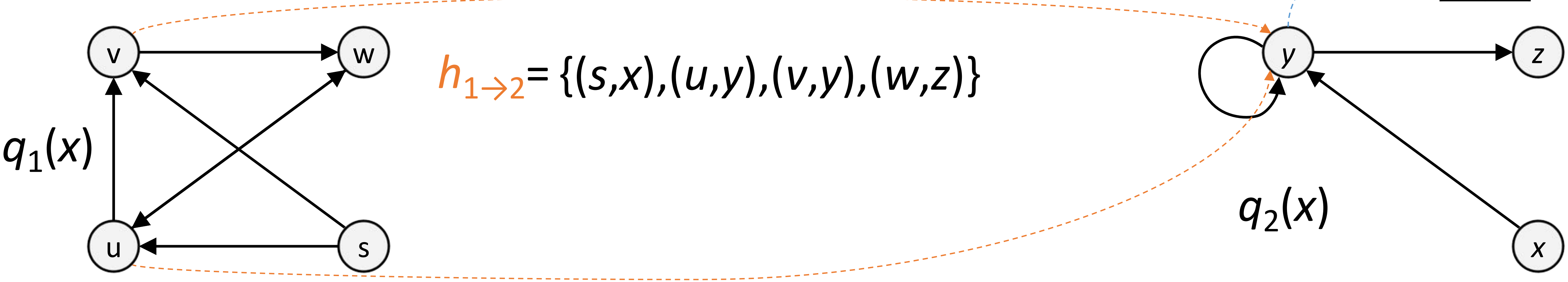
q_1 :- $R(s,u)$, $R(u,w)$, $R(s,v)$, $R(v,w)$, $R(u,v)$

q_2 :- $R(x,y)$, $R(y,y)$, $R(y,z)$

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

R

A	B
a	b
b	b
b	c



$h_{1\rightarrow2}$= {(s,x),(u,y),(v,y),(w,z)}

$q_1(x)$

$q_2(x)$

[Chandra and Merlin 1977]

We show: If there is a homomorphism $h_{1\rightarrow 2}$, then for any D: $q_1(D) \Leftarrow q_2(D)$

1. For $q_2(D)$ to hold, there is a valuation v s.t. $v(q_2) \in D$

2. We will show that the composition $g = v \circ h$ is a valuation for q_1

    2a. By definition of h, for every $R(x_1,x_2,\ldots)$ in q_1, $R(h(x_1),h(x_2),\ldots)$ in q_2

    2b. By definition of v, for every $R(x_1,x_2,\ldots)$ in q_1, $R(v(h(x_1)),v(h(x_2)),\ldots)$ in D

$g=v \circ h$

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

<u>Example</u>

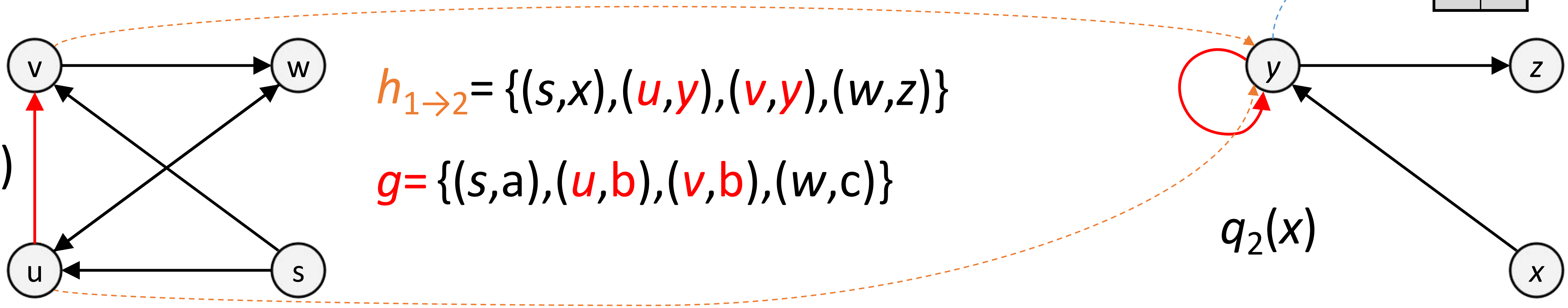
q_1 :- $R(s,u), R(u,w), R(s,v), R(v,w), R(u,v)$

q_2 :- $R(x,y), R(y,y), R(y,z)$

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

R	A	B
	a	b
	b	b
	b	c

$q_1(x)$



$h_{1\rightarrow 2}=\{(s,x),(u,y),(v,y),(w,z)\}$

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

$q_2(x)$

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 $h: Q' \to Q$

(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