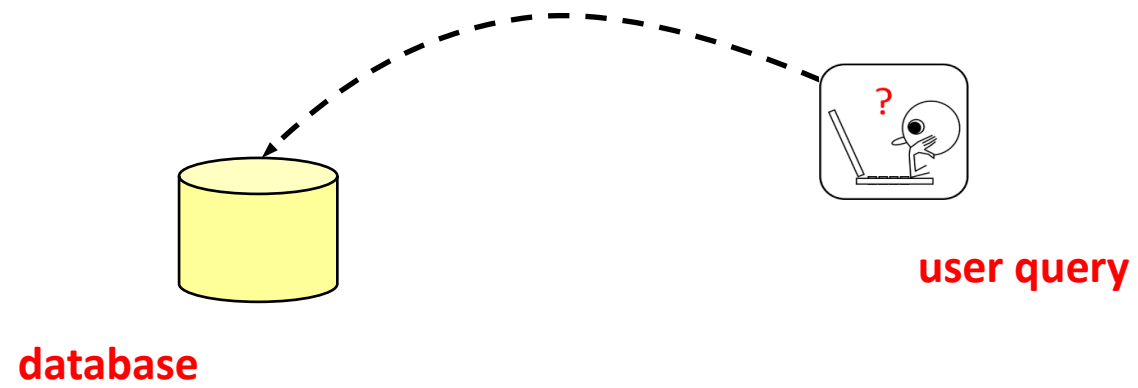




A Quick Recap

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
--------	---------------	--------------------	----------------

Airport	<i>code</i>	<i>city</i>
---------	-------------	-------------

A Quick Recap

List all the airlines

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

A Quick Recap

List all the airlines

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS



{BA, U2, OS}

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

π_{airline} Flight

A Quick Recap

List the codes of the airports in London

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

A Quick Recap

List the codes of the airports in London

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh



{LHR, LGW}

$\pi_{\text{code}} (\sigma_{\text{city}='London'} \text{ Airport})$

A Quick Recap

List the airlines that fly directly from London to Glasgow

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

$\pi_{\text{airline}} ((\text{Flight} \bowtie_{\text{origin=code}} (\sigma_{\text{city='London'}} \text{Airport})) \bowtie_{\text{destination=code}} (\sigma_{\text{city='Glasgow'}} \text{Airport}))$

A Quick Recap

$\pi_{\text{airline}} ((\text{Flight} \bowtie_{\text{origin=code}} (\underbrace{\sigma_{\text{city}='London'} \text{ Airport}})) \bowtie_{\text{destination=code}} (\underbrace{\sigma_{\text{city}='Glasgow'} \text{ Airport}}))$

<i>code</i>	<i>city</i>
LHR	London
LGW	London

<i>code</i>	<i>city</i>
GLA	Glasgow

A Quick Recap

$\pi_{\text{airline}} ((\text{Flight} \bowtie_{\text{origin=code}} (\sigma_{\text{city='London'}} \text{Airport})) \bowtie_{\text{destination=code}} (\sigma_{\text{city='Glasgow'}} \text{Airport}))$

<i>code</i>	<i>city</i>
LHR	London
LGW	London

<i>code</i>	<i>city</i>
GLA	Glasgow

<i>origin</i>	<i>destination</i>	<i>airline</i>	<i>code</i>	<i>city</i>
LHR	EDI	BA	LHR	London
LGW	GLA	U2	LGW	London

A Quick Recap

$\pi_{\text{airline}} ((\text{Flight} \bowtie_{\text{origin}=\text{code}} (\sigma_{\text{city}='London'} \text{Airport})) \bowtie_{\text{destination}=\text{code}} (\sigma_{\text{city}='Glasgow'} \text{Airport}))$

<i>code</i>	<i>city</i>
LHR	London
LGW	London

<i>code</i>	<i>city</i>
GLA	Glasgow

<i>origin</i>	<i>destination</i>	<i>airline</i>	<i>code</i>	<i>city</i>
LHR	EDI	BA	LHR	London
LGW	GLA	U2	LGW	London

<i>origin</i>	<i>destination</i>	<i>airline</i>	<i>code</i>	<i>city</i>	<i>code</i>	<i>city</i>
LGW	GLA	U2	LGW	London	GLA	Glasgow

A Quick Recap

$\pi_{\text{airline}} ((\text{Flight} \bowtie_{\text{origin}=\text{code}} (\sigma_{\text{city}='London'} \text{Airport})) \bowtie_{\text{destination}=\text{code}} (\sigma_{\text{city}='Glasgow'} \text{Airport}))$

<i>code</i>	<i>city</i>
LHR	London
LGW	London

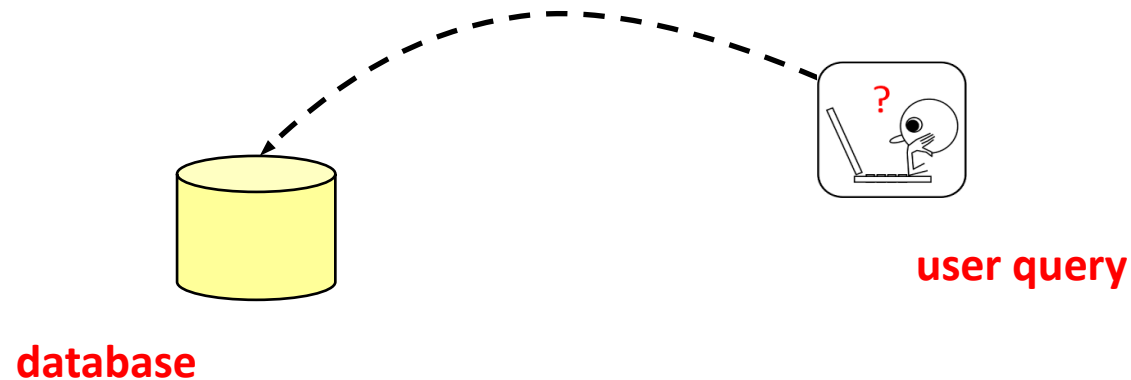
<i>code</i>	<i>city</i>
GLA	Glasgow

<i>origin</i>	<i>destination</i>	<i>airline</i>	<i>code</i>	<i>city</i>
LHR	EDI	BA	LHR	London
LGW	GLA	U2	LGW	London

<i>origin</i>	<i>destination</i>	<i>airline</i>	<i>code</i>	<i>city</i>	<i>code</i>	<i>city</i>
LGW	GLA	U2	LGW	London	GLA	Glasgow

<i>airline</i>
U2

understand how a DBMS efficiently evaluates a query over a database
(the main focus of this course)



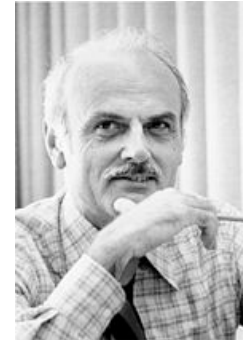
a glimpse on the mathematical foundations of query evaluation
(next two weeks)

Conjunctive Queries: Syntax and Semantics

(Chapters 12 and 13 of DBT)

Codd's Theorem

Relational Algebra = Relational Calculus



Edgar F. Codd
(1923 - 2003)
Turing Award 1981

- Queries are written using a **declarative language**
- DBMS converts declarative queries into **procedural queries** that are optimized and executed

Relational Calculus

List all the airlines

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS



{BA, U2, OS}

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

$\{z \mid \exists x \exists y (\text{Flight}(x,y,z))\}$

Relational Calculus

List the codes of the airports in London

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh



{LHR, LGW}

$\{x \mid \text{Airport}(x, \text{London})\}$

Relational Calculus

List the airlines that fly directly from London to Glasgow

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS



{U2}

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

$\{z \mid \exists x \exists y (\text{Airport}(x, \text{London}) \wedge \text{Airport}(y, \text{Glasgow}) \wedge \text{Flight}(x, y, z))\}$

A Core Relational Query Language

Conjunctive Queries (CQ)

= $\{\sigma, \pi, \bowtie\}$ -fragment of relational algebra

= relational calculus without $\neg$, $\forall$, $\exists$, $=$

= simple SELECT-FROM-WHERE SQL queries

= (only AND and equality in the WHERE clause)

Syntax of Conjunctive Queries

$$Q(\mathbf{x}) := \exists \mathbf{y} (R_1(\mathbf{v}_1) \wedge \cdots \wedge R_m(\mathbf{v}_m))$$

- $R_1, \dots, R_m$ are relation names
- $\mathbf{x}, \mathbf{y}, \mathbf{v}_1, \dots, \mathbf{v}_m$ are tuples of variables
- each variable mentioned in $\mathbf{v}_i$ appears either in $\mathbf{x}$ or $\mathbf{y}$
- the variables in $\mathbf{x}$ are free called **distinguished** or **output variables**

It is very convenient to see conjunctive queries as rule-based queries of the form

$$Q(\mathbf{x}) \text{ :- } \underbrace{R_1(\mathbf{v}_1), \dots, R_m(\mathbf{v}_m)}$$

this is called the **body** of Q that can be seen as a set of atoms

Conjunctive Queries: Example 1

List all the airlines

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS



{BA, U2, OS}

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

π_{airline} Flight

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

$\{z \mid \exists x \exists y \text{ Flight}(x,y,z)\}$

Conjunctive Queries: Example 2

List the codes of the airports in London

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh



{LHR, LGW}

$\pi_{\text{code}} (\sigma_{\text{city}='London'} \text{ Airport})$

{x | Airport(x,London)}

$Q(x) \text{ :- Airport}(x, \text{London})$

Conjunctive Queries: Example 3

List the airlines that fly directly from London to Glasgow

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS



{U2}

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

$\pi_{\text{airline}} ((\text{Flight} \bowtie_{\text{origin=code}} (\sigma_{\text{city='London'}} \text{Airport})) \bowtie_{\text{destination=code}} (\sigma_{\text{city='Glasgow'}} \text{Airport}))$

$\{z \mid \exists x \exists y (\text{Airport}(x, \text{London}) \wedge \text{Airport}(y, \text{Glasgow}) \wedge \text{Flight}(x, y, z))\}$

Conjunctive Queries: Example 3

List the airlines that fly directly from London to Glasgow

Flight	<i>origin</i>	<i>destination</i>	<i>airline</i>
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS



{U2}

Airport	<i>code</i>	<i>city</i>
	VIE	Vienna
	LHR	London
	LGW	London
	LCA	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

$Q(z) \text{ :- Airport}(x, \text{London}), \text{Airport}(y, \text{Glasgow}), \text{Flight}(x, y, z)$

Pattern Matching Problem

List the airlines that fly directly from London to Glasgow

Flight(VIE,LHR,BA),	Airport(VIE,Vienna),
Flight(LHR,EDI,BA),	Airport(LHR,London),
Flight(LGW,GLA,U2),	Airport(LGW,London),
Flight(LCA,VIE,OS),	Airport(LCA,Larnaca),
	Airport(GLA,Glasgow),
	Airport(EDI,Edinburgh)

$Q(z) \text{ :- Airport}(x,\text{London}), \text{Airport}(y,\text{Glasgow}), \text{Flight}(x,y,z)$

Pattern Matching Problem

List the airlines that fly directly from London to Glasgow

Flight(VIE,LHR,BA),	Airport(VIE,Vienna),
Flight(LHR,EDI,BA),	Airport(LHR,London),
Flight(LGW,GLA,U2),	Airport(LGW,London),
Flight(LCA,VIE,OS),	Airport(LCA,Larnaca),
	Airport(GLA,Glasgow),
	Airport(EDI,Edinburgh)

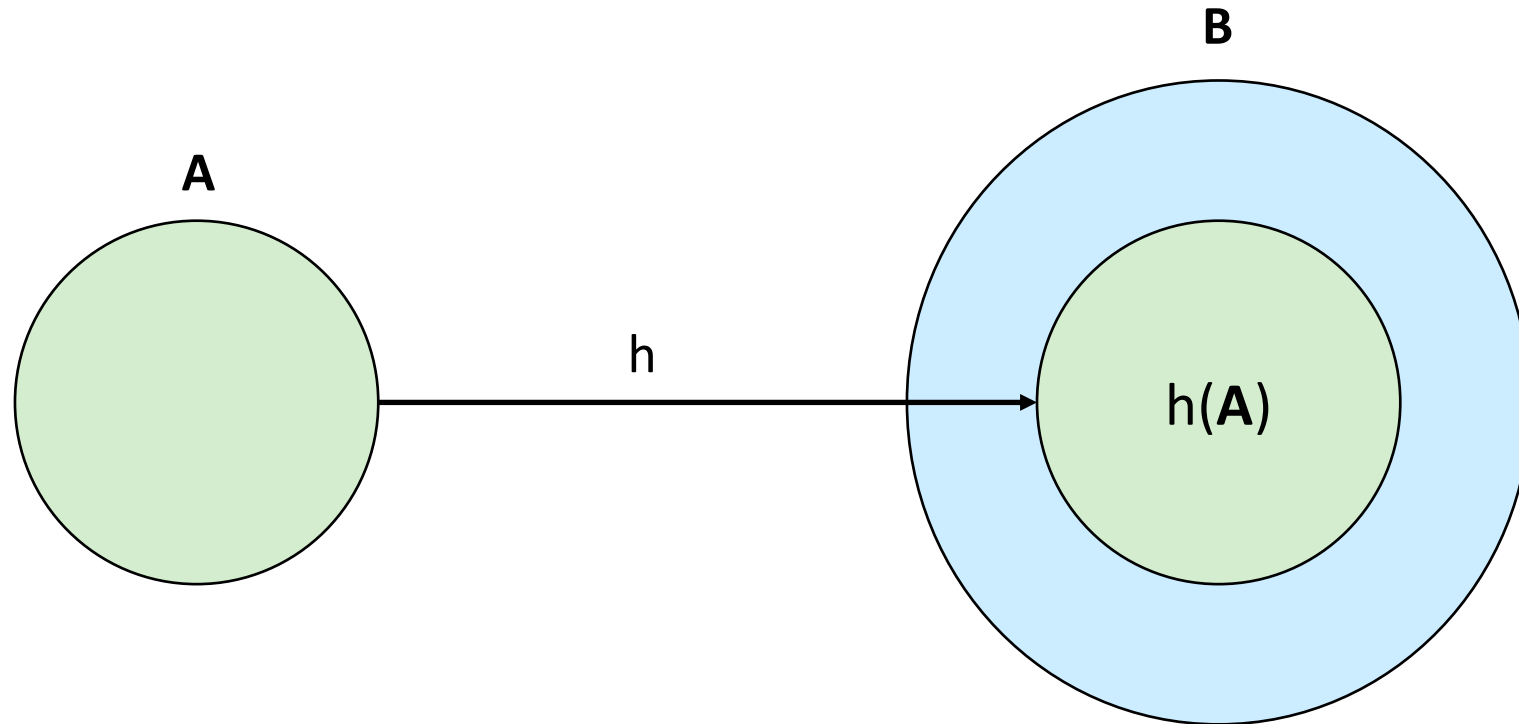
Q(z) :- Airport(x,London), Airport(y,Glasgow), Flight(x,y,z)

Homomorphism

- Pattern matching - properly formalized via the key notion of **homomorphism**
- A **substitution** from a set of terms **S** to a set of terms **T** is a function $h : \mathbf{S} \rightarrow \mathbf{T}$, i.e., h is a set of **mappings** of the form $s \mapsto t$, where $s \in \mathbf{S}$ and $t \in \mathbf{T}$
- A **homomorphism** from a set of atoms **A** to a set of atoms **B** is a substitution $h : \text{terms}(\mathbf{A}) \rightarrow \text{terms}(\mathbf{B})$ such that:
 1. t is a constant value $\Rightarrow h(t) = t$
 2. $R(t_1, \dots, t_k) \in \mathbf{A} \Rightarrow h(R(t_1, \dots, t_k)) = R(h(t_1), \dots, h(t_k)) \in \mathbf{B}$

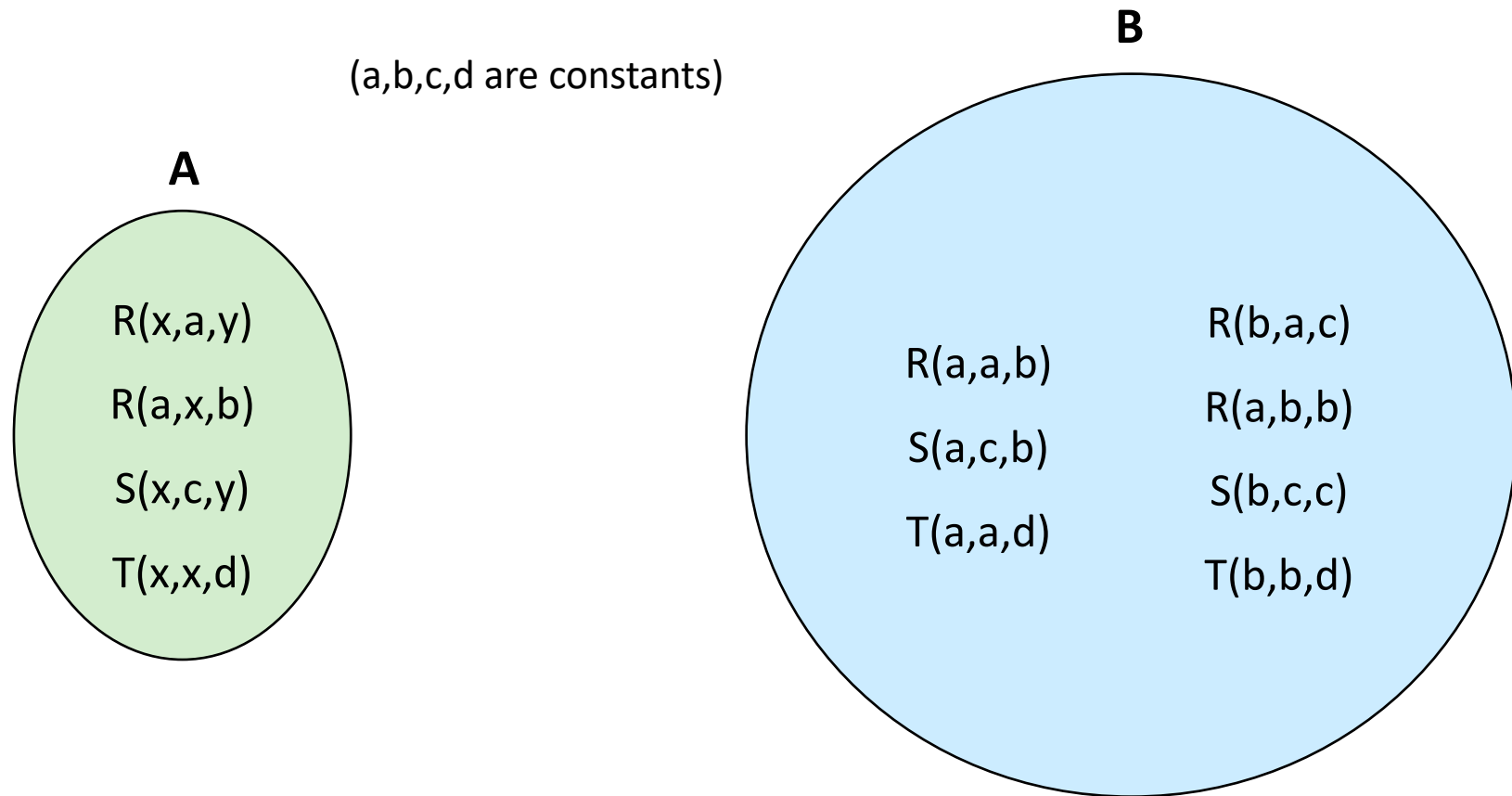
$(\text{terms}(\mathbf{A}) = \{t \mid t \text{ is a variable or a constant value that occurs in } \mathbf{A}\})$

Homomorphism

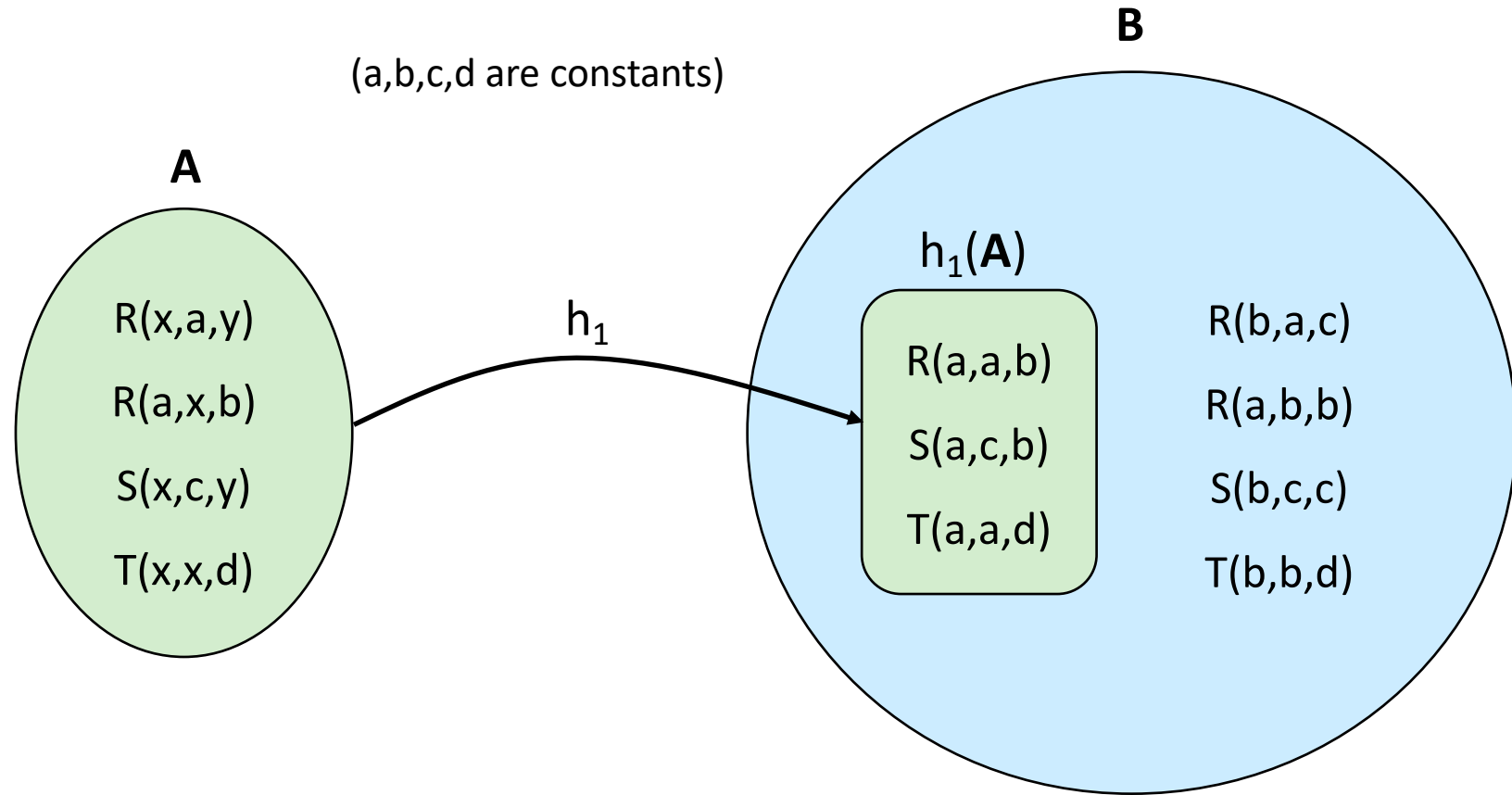


$h : \text{terms}(\mathbf{A}) \rightarrow \text{terms}(\mathbf{B})$ that is the identity on constants

Homomorphism

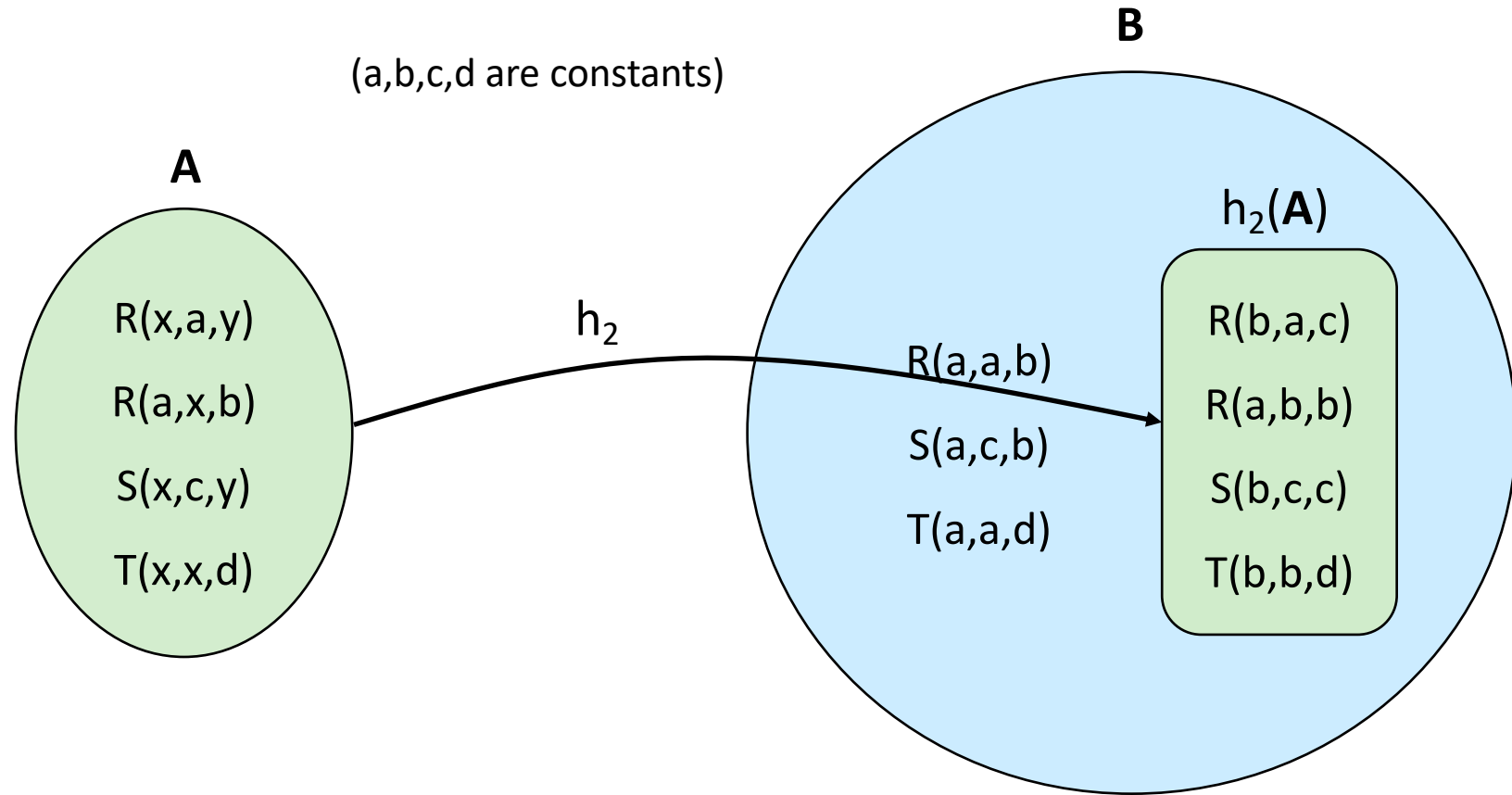


Homomorphism



$$h_1 = \{a \mapsto a, b \mapsto b, c \mapsto c, d \mapsto d, x \mapsto a, y \mapsto b\}$$

Homomorphism



$$h_2 = \{a \mapsto a, b \mapsto b, c \mapsto c, d \mapsto d, x \mapsto b, y \mapsto c\}$$

Find the Homomorphisms

$$S_1 = \{P(x_1, y_1), P(y_1, z_1), P(z_1, w_1)\}$$

$$S_2 = \{P(x_2, y_2), P(y_2, z_2), P(z_2, x_2)\}$$

$$S_3 = \{P(x_3, y_3), P(y_3, x_3)\}$$

$$S_4 = \{P(x_4, y_4), P(y_4, x_4), P(y_4, y_4)\}$$

$$S_5 = \{P(x_5, x_5)\}$$

Find the Homomorphisms

$$S_1 = \{P(x_1, y_1), P(y_1, z_1), P(z_1, w_1)\}$$

$$\{x_1 \mapsto x_2, y_1 \mapsto y_2, z_1 \mapsto z_2, w_1 \mapsto x_2\}$$

$$\{x_1 \mapsto x_3, y_1 \mapsto y_3, z_1 \mapsto x_3, w_1 \mapsto y_3\}$$

$$S_2 = \{P(x_2, y_2), P(y_2, z_2), P(z_2, x_2)\}$$

$$S_3 = \{P(x_3, y_3), P(y_3, x_3)\}$$

$$S_4 = \{P(x_4, y_4), P(y_4, x_4), P(y_4, y_4)\}$$

$$S_5 = \{P(x_5, x_5)\}$$

Find the Homomorphisms

$$S_1 = \{P(x_1, y_1), P(y_1, z_1), P(z_1, w_1)\}$$

$$\{x_1 \mapsto x_2, y_1 \mapsto y_2, z_1 \mapsto z_2, w_1 \mapsto x_2\}$$

$$\{x_1 \mapsto x_3, y_1 \mapsto y_3, z_1 \mapsto x_3, w_1 \mapsto y_3\}$$

$$S_2 = \{P(x_2, y_2), P(y_2, z_2), P(z_2, x_2)\}$$

$$S_3 = \{P(x_3, y_3), P(y_3, x_3)\}$$

$$\{x_2 \mapsto y_4, y_2 \mapsto x_4, z_2 \mapsto y_4\}$$

$$\{x_3 \mapsto x_4, y_3 \mapsto y_4\}$$

$$S_4 = \{P(x_4, y_4), P(y_4, x_4), P(y_4, y_4)\}$$

$$S_5 = \{P(x_5, x_5)\}$$

Find the Homomorphisms

$$S_1 = \{P(x_1, y_1), P(y_1, z_1), P(z_1, w_1)\}$$

$$\{x_1 \mapsto x_2, y_1 \mapsto y_2, z_1 \mapsto z_2, w_1 \mapsto x_2\}$$

$$\{x_1 \mapsto x_3, y_1 \mapsto y_3, z_1 \mapsto x_3, w_1 \mapsto y_3\}$$

$$S_2 = \{P(x_2, y_2), P(y_2, z_2), P(z_2, x_2)\}$$

$$S_3 = \{P(x_3, y_3), P(y_3, x_3)\}$$

$$\{x_2 \mapsto y_4, y_2 \mapsto x_4, z_2 \mapsto y_4\}$$

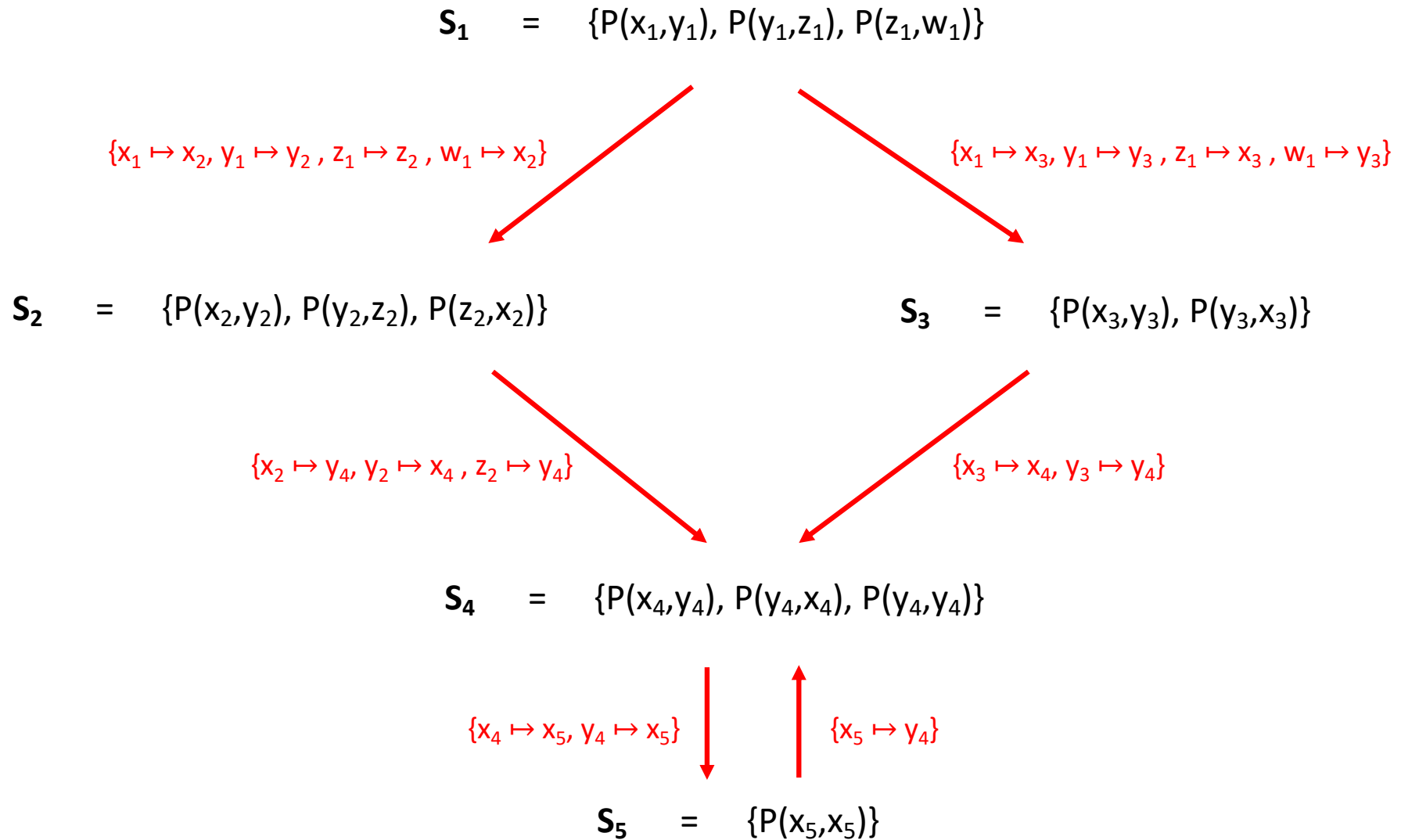
$$\{x_3 \mapsto x_4, y_3 \mapsto y_4\}$$

$$S_4 = \{P(x_4, y_4), P(y_4, x_4), P(y_4, y_4)\}$$

$$\{x_4 \mapsto x_5, y_4 \mapsto x_5\}$$

$$S_5 = \{P(x_5, x_5)\}$$

Find the Homomorphisms



Homomorphisms Compose

$$S_1 = \{P(x_1, y_1), P(y_1, z_1), P(z_1, w_1)\}$$

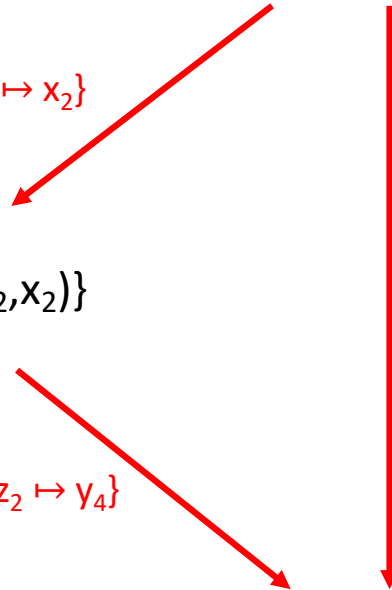
$$\{x_1 \mapsto x_2, y_1 \mapsto y_2, z_1 \mapsto z_2, w_1 \mapsto x_2\}$$

$$S_2 = \{P(x_2, y_2), P(y_2, z_2), P(z_2, x_2)\}$$

$$\{x_1 \mapsto y_4, y_1 \mapsto x_4, z_1 \mapsto y_4, w_1 \mapsto y_4\}$$

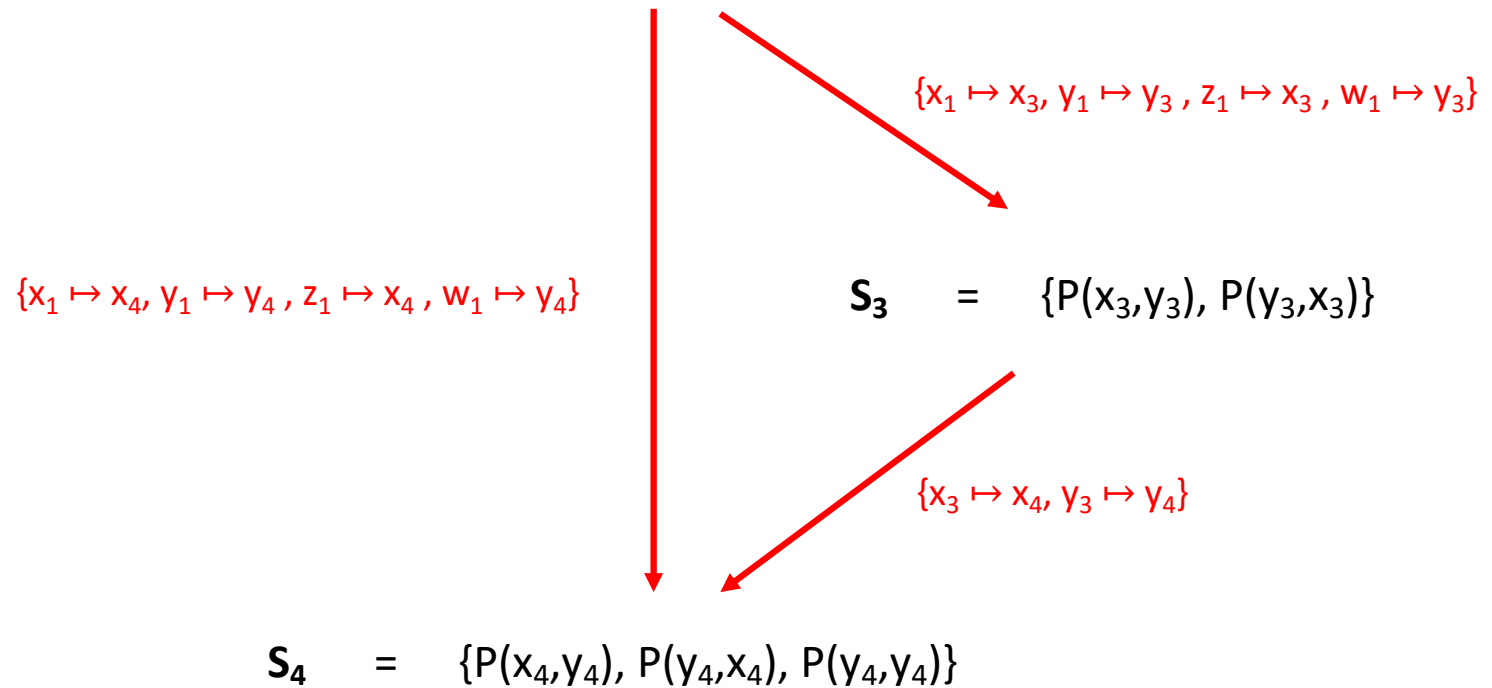
$$\{x_2 \mapsto y_4, y_2 \mapsto x_4, z_2 \mapsto y_4\}$$

$$S_4 = \{P(x_4, y_4), P(y_4, x_4), P(y_4, y_4)\}$$



Homomorphisms Compose

$$S_1 = \{P(x_1, y_1), P(y_1, z_1), P(z_1, w_1)\}$$



Semantics of Conjunctive Queries

- A **match** of a conjunctive query $Q(x_1, \dots, x_k) :- \text{body}$ in a database D is a homomorphism h from the set of atoms **body** to the set of atoms D

- The **answer** to $Q(x_1, \dots, x_k) :- \text{body}$ over D is the set of k -tuples

$$Q(D) := \{(h(x_1), \dots, h(x_k)) \mid h \text{ is a match of } Q \text{ in } D\}$$

- The answer consists of the witnesses for the **distinguished variables** of Q

Pattern Matching Problem

List the airlines that fly directly from London to Glasgow

Flight(VIE,LHR,BA),	Airport(VIE,Vienna),
Flight(LHR,EDI,BA),	Airport(LHR,London),
Flight(LGW,GLA,U2),	Airport(LGW,London),
Flight(LCA,VIE,OS),	Airport(LCA,Larnaca),
	Airport(GLA,Glasgow),
	Airport(EDI,Edinburgh)

Q(z) :- Airport(x,London), Airport(y,Glasgow), Flight(x,y,z)

Pattern Matching Problem

List the airlines that fly directly from London to Glasgow

