

Reasoning with Constraints

Andreas Pieris

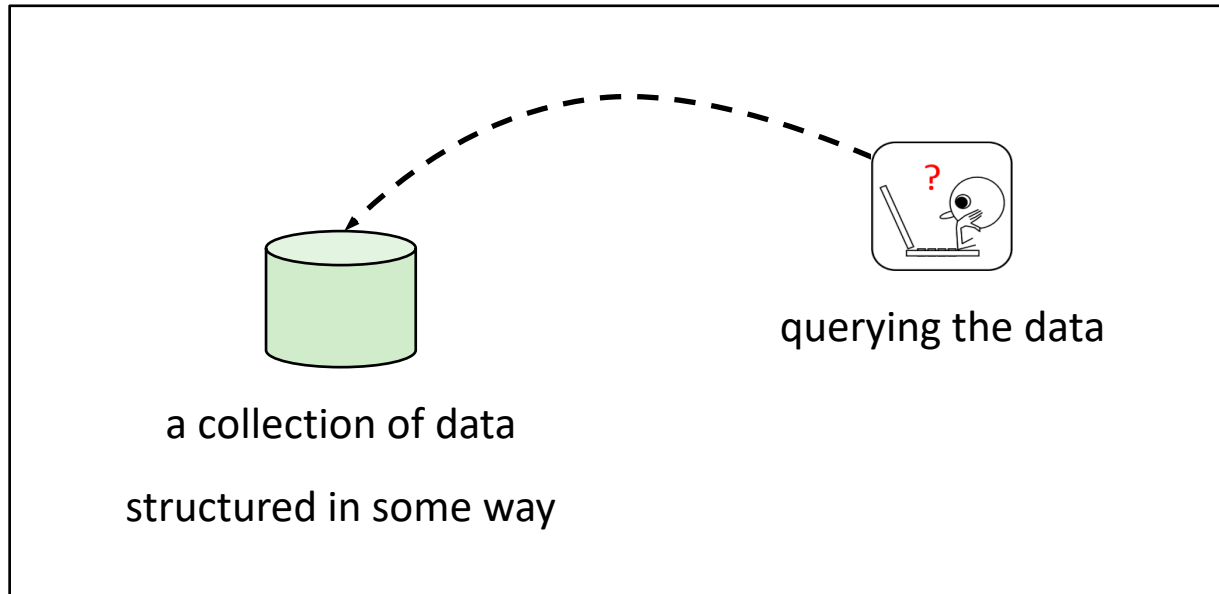
University of Edinburgh & University of Cyprus

EDBT-Intented Summer School, Bordeaux, France, July 4 - 9, 2022

Foundations of Databases by Abiteboul, Hull and Vianu - accessible at <http://webdam.inria.fr/Alice/>



Principles of Databases by Arenas, Barcelo, Libkin, Martens and P. - currently under development, a preliminary version is accessible at <https://github.com/pdm-book/community>

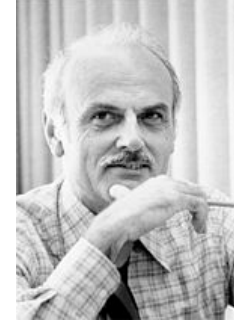


Data Model

mathematical abstraction for structuring the data
independent from the physical implementation

Relational Model

- Many ad hoc data models before 1970
 - Hard to work with
 - Hard to reason about
- **1970: Relational Model by Edgar Frank Codd**
 - Data are stored in **relations** (or tables)
 - Queried using a **declarative language**
 - DBMS converts declarative queries into **procedural queries** that are optimized and executed
- Key Advantages
 - Simple and clean mathematical model (based on **logic**)
 - Separation of declarative and procedural



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

Relational Databases

Database Schema: a finite set of **relation names** together with their **attributes names**

Flight	origin:string	destination:string	airline:string
---------------	---------------	--------------------	----------------

Airport	code:string	city:string
----------------	-------------	-------------

+

Database Instance: data conforming to the schema

VIE	LHR	BA
LHR	EDI	BA
LGW	GLA	U2
LCA	VIE	OS

VIE	Vienna
LHR	London
LGW	London
LGW	Larnaca
GLA	Glasgow
EDI	Edinburgh

Relational Databases

Flight	origin:string	destination:string	airline:string
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

Airport	code:string	city:string
	VIE	Vienna
	LHR	London
	LGW	London
	LGW	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

- **Ignore attribute types** - data elements are coming from a countably infinite set **Const** (constant values)
- A relational database is a *finite* set of **relational atoms**

Relational Databases

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(LGW,Larnaca),
	Airport(GLA,Glasgow),
	Airport(EDI,Edinburgh)

...we will keep using the table representation without the attribute types

Querying Relational Databases

List the airlines that fly directly from London to Glasgow

Flight	origin	destination	airline
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

Airport	code	city
	VIE	Vienna
	LHR	London
	LGW	London
	LGW	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

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

Querying Relational Databases

List the airlines that fly directly from London to Glasgow

Flight	origin	destination	airline
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

???

Airport	code	city
	VIE	Vienna
	LHR	London
	LGW	London
	LGW	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

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

Querying Relational Databases

List the airlines that fly directly from London to Glasgow

Flight	origin	destination	airline
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

???

Airport	code	city
	VIE	Vienna
	LHR	London
	LGW	London
	LGW	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

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

Querying Relational Databases

List the airlines that fly directly from London to Glasgow

Flight	origin	destination	airline
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

???

Airport	code	city
	VIE	Vienna
	LHR	London
	LGW	London
	LGW	Larnaca
	GLA	Glasgow
	EDI	Edinburgh

we should specify that the code of an airport uniquely determines the city

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

Integrity Constraints

*specify semantic properties that should be satisfied by
every database instance of a certain schema*

- Development of transparent and usable database schemas
- Play a crucial role in query optimization

Integrity Constraints

*specify semantic properties that should be satisfied by
every database instance of a given schema*

Functional Dependencies & Inclusion Dependencies

- Development of transparent and flexible database schemas
- Optimizing the evaluation of queries

Functional & Inclusion Dependencies

Person	p_id	name
---------------	------	------

Profession	p_id	pr_name
-------------------	------	---------

the id of a person uniquely determines that person

$$\text{Person} : \{1\} \rightarrow \{2\}$$

the first attribute of **Profession** stores person ids

$$\text{Profession}[1] \subseteq \text{Person}[1]$$

Syntax of Functional Dependencies

A **functional dependency (FD)** σ over a schema **S** is an expression of the form

$$R : X \rightarrow Y$$

where

- R is a relation name of **S**
- $X, Y \subseteq \{1, \dots, \text{arity}(R)\}$ - $\text{arity}(R) = \#\text{attributes of } R$

If $X \cup Y = \{1, \dots, \text{arity}(R)\}$, then σ is called a **key dependency**, written as $\text{Key}(R) = X$

If σ is the only key of R , then it is called **primary key**

Syntax of Functional Dependencies

Person	p_id	name
---------------	------	------

Profession	p_id	pr_name
-------------------	------	---------

the id of a person uniquely determines that person

Person : {1} → {2}

or simply

Key(Person) = {1}

Semantics of Functional Dependencies

A database instance D of a schema S **satisfies an FD** σ of the form $R : X \rightarrow Y$ over S ,

denoted $D \models \sigma$, if, for each pair of atoms $R(a_1, \dots, a_n)$ and $R(b_1, \dots, b_n)$ in D ,

$$\pi_X(a_1, \dots, a_n) = \pi_X(b_1, \dots, b_n) \Rightarrow \pi_Y(a_1, \dots, a_n) = \pi_Y(b_1, \dots, b_n)$$



projection - $\pi_{\{1,2,4\}}(a,b,c,d,e) = (a,b,d)$

We say that D **satisfies a set** Σ of FDs, denoted $D \models \Sigma$, if $D \models \sigma$ for each $\sigma \in \Sigma$

Semantics of Functional Dependencies

Person	p_id	name
---------------	------	------

Profession	p_id	pr_name
-------------------	------	---------

the id of a person uniquely determines that person

$$\text{Person} : \{1\} \rightarrow \{2\}$$

or

$$\forall x \forall y (\text{Person}(x,y) \wedge \text{Person}(x,z) \rightarrow y = z)$$

special case of an **equality-generating dependency** - $\forall x \forall y (\phi(\mathbf{x}) \rightarrow x_1 = x_2)$

Syntax of Inclusion Dependencies

An **inclusion dependency (IND)** σ over a schema S is an expression of the form

$$R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$$

where

- R and P are relation names of S
- (i_1, \dots, i_k) is a non-empty list of distinct integers from $\{1, \dots, \text{arity}(R)\}$
- (j_1, \dots, j_k) is a non-empty list of distinct integers from $\{1, \dots, \text{arity}(P)\}$

Semantics of Inclusion Dependencies


R	att ₁	att ₂	att ₃
	a	b	c
	c	d	e

P	att ₁	att ₂	att ₃

$$R[2,1] \subseteq P[1,3]$$

Semantics of Inclusion Dependencies

R	att ₁	att ₂	att ₃
	a	b	c
	c	d	e




P	att ₁	att ₂	att ₃
	b	*	a

$$R[2,1] \subseteq P[1,3]$$

Semantics of Inclusion Dependencies

R	att ₁	att ₂	att ₃
	a	b	c
	c	d	e



P	att ₁	att ₂	att ₃
	b	*	a
	d	*	c

$$R[2,1] \subseteq P[1,3]$$

Semantics of Inclusion Dependencies

A database instance D of a schema S **satisfies an IND** σ of the form $R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$ over S , denoted $D \models \sigma$, if, for every $R(a_1, \dots, a_n)$ in D , there exists $P(b_1, \dots, b_m)$ in D such that

$$\pi_{(i_1, \dots, i_k)}(a_1, \dots, a_n) = \pi_{(j_1, \dots, j_k)}(b_1, \dots, b_m)$$

We say that D **satisfies a set** Σ of INDs, denoted $D \models \Sigma$, if $D \models \sigma$ for each $\sigma \in \Sigma$

Semantics of Inclusion Dependencies

Person	p_id	name
---------------	------	------

Profession	p_id	pr_name
-------------------	------	---------

the first attribute of **Profession** stores person ids

$$\text{Profession}[1] \subseteq \text{Person}[1]$$

or

$$\forall x \forall y (\text{Profession}(x,y) \rightarrow \exists z \text{Person}(x,z))$$

special case of a **tuple-generating dependency** - $\forall x \forall y (\phi(x) \rightarrow \exists z \psi(x))$

Semantics of Inclusion Dependencies

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

P	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$R[2,1] \subseteq P[1,3]$$

or

$$\forall x \forall y \forall z (R(x,y,z) \rightarrow \exists w P(y,w,x))$$

Integrity Constraints - Basic Algorithmic Tasks

- Check whether a database is valid w.r.t. a set of constraints
- Discover new constraints from existing ones

Integrity Constraints - Basic Algorithmic Tasks

- Check whether a database is valid w.r.t. a set of constraints
 - Discover new constraints from existing ones
- Satisfaction**
&
Logical Implication

Satisfaction of Constraints

FD-Satisfaction

Input: a database D of a schema S , and a set Σ of FDs over S

Output: **true** if $D \models \Sigma$, and **false** otherwise

IND-Satisfaction

Input: a database D of a schema S , and a set Σ of INDs over S

Output: **true** if $D \models \Sigma$, and **false** otherwise

Satisfaction of FDs

Theorem: FD-Satisfaction is feasible in **polynomial time**

Consider a database instance **D** of a schema **S**, and a set Σ of FDs over **S**. The following is an algorithm for **FD-Satisfaction**

for every $R : X \rightarrow Y$ in Σ do

for each pair of atoms $R(a_1, \dots, a_n)$ and $R(b_1, \dots, b_n)$ in **D** do

if $\pi_X(a_1, \dots, a_n) = \pi_X(b_1, \dots, b_n)$ and $\pi_Y(a_1, \dots, a_n) \neq \pi_Y(b_1, \dots, b_n)$ then

return **false**

return **true**

which clearly runs in polynomial time

Satisfaction of INDs

Theorem: IND-Satisfaction is feasible in **polynomial time**

Consider a database instance **D** of a schema **S**, and a set Σ of INDs over **S**. The following is an algorithm for **IND-Satisfaction**

for every $R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$ in Σ do

 for each atom $R(a_1, \dots, a_n)$ in **D** do

 if there is no $P(b_1, \dots, b_n)$ in **D** with $\pi_{(i_1, \dots, i_k)}(a_1, \dots, a_n) = \pi_{(j_1, \dots, j_k)}(b_1, \dots, b_n)$ then

 return **false**

return **true**

which clearly runs in polynomial time

Logical Implication of Constraints

A set Σ of constraints over a schema \mathbf{S} **implies** a constraint σ over \mathbf{S} , denoted $\Sigma \models \sigma$, if, for every database instance \mathbf{D} of \mathbf{S} , it holds that $\mathbf{D} \models \Sigma \Rightarrow \mathbf{D} \models \sigma$

FD-Implication

Input: a set Σ of FDs over a schema \mathbf{S} , and an FD σ over \mathbf{S}

Output: true if $\Sigma \models \sigma$, and false otherwise

IND-Implication

Input: a set Σ of INDs over a schema \mathbf{S} , and an IND σ over \mathbf{S}

Output: true if $\Sigma \models \sigma$, and false otherwise

Characterizing Implication for FDs

For an FD σ of the form $R : X \rightarrow Y$, we define the set of relational atoms

$$\text{Violate}[\sigma] = \{R(x_1, \dots, x_n), R(y_1, \dots, y_n)\}$$

where

- $x_1, \dots, x_n, y_1, \dots, y_n$ are variables from a countably infinite set **Var** (disjoint from **Const**)
- for each distinct $i, j \in \{1, \dots, n\}$, $x_i \neq x_j$ and $y_i \neq y_j$
- for each $i \in \{1, \dots, n\}$, $x_i = y_i$ iff $i \in X$

the prototypical set of
atoms that violates σ

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$\text{Violate}[\sigma] = \{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

$$\sigma = R : \{2\} \rightarrow \{3\}$$

Characterizing Implication for FDs

Given a set Σ of FDs over a schema \mathbf{S} , and an FD σ over \mathbf{S} ,

transform $\text{Violate}[\sigma]$ into a *finite* set of relational atoms $\text{Violate}[\sigma]_{\Sigma}$ that satisfies Σ such that

$$\Sigma \models \sigma \iff \text{Violate}[\sigma]_{\Sigma} \models \sigma$$



acts as a “representative” of all the database instances of \mathbf{S} that satisfy Σ



We get an algorithm for **FD-Implication**:

if $\text{Violate}[\sigma]_{\Sigma} \models \sigma$ then

return **true**

else

return **false**

Homomorphism

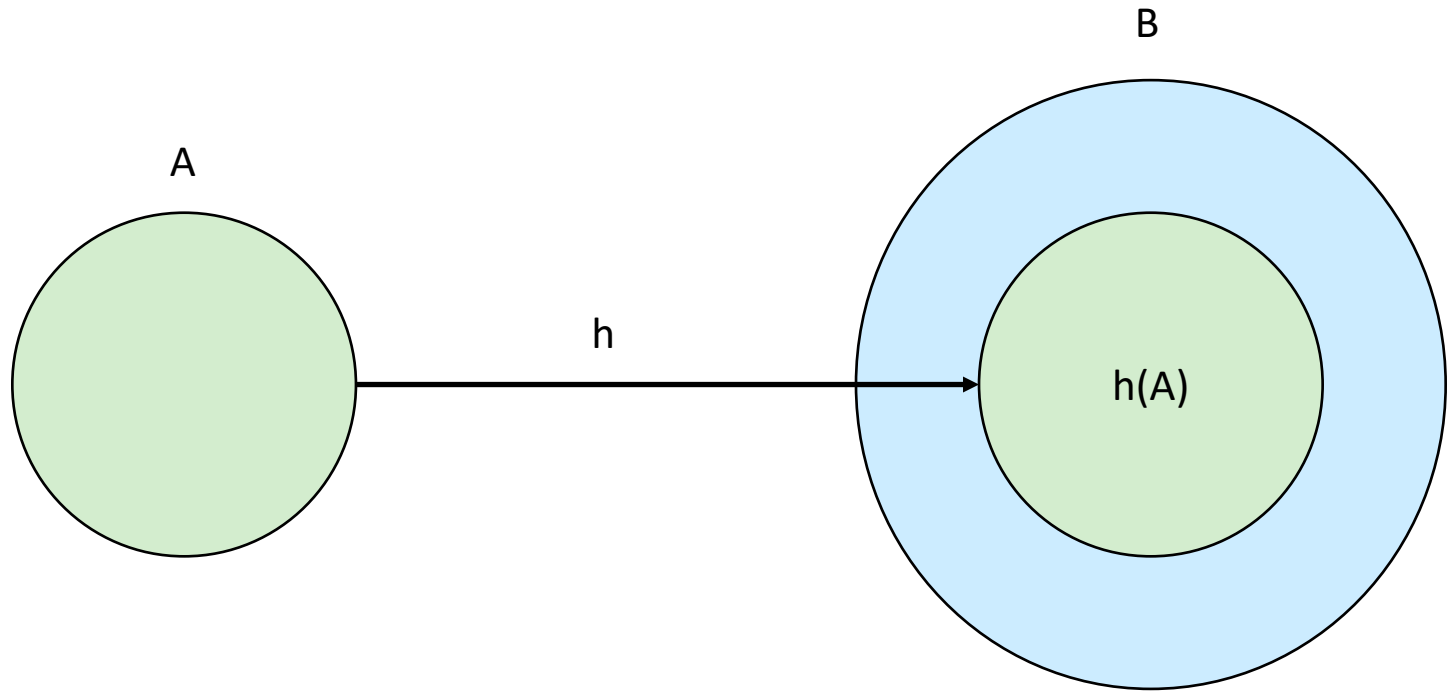
- Structure-preserving functions between two objects of the same type

set of variables and constants in A



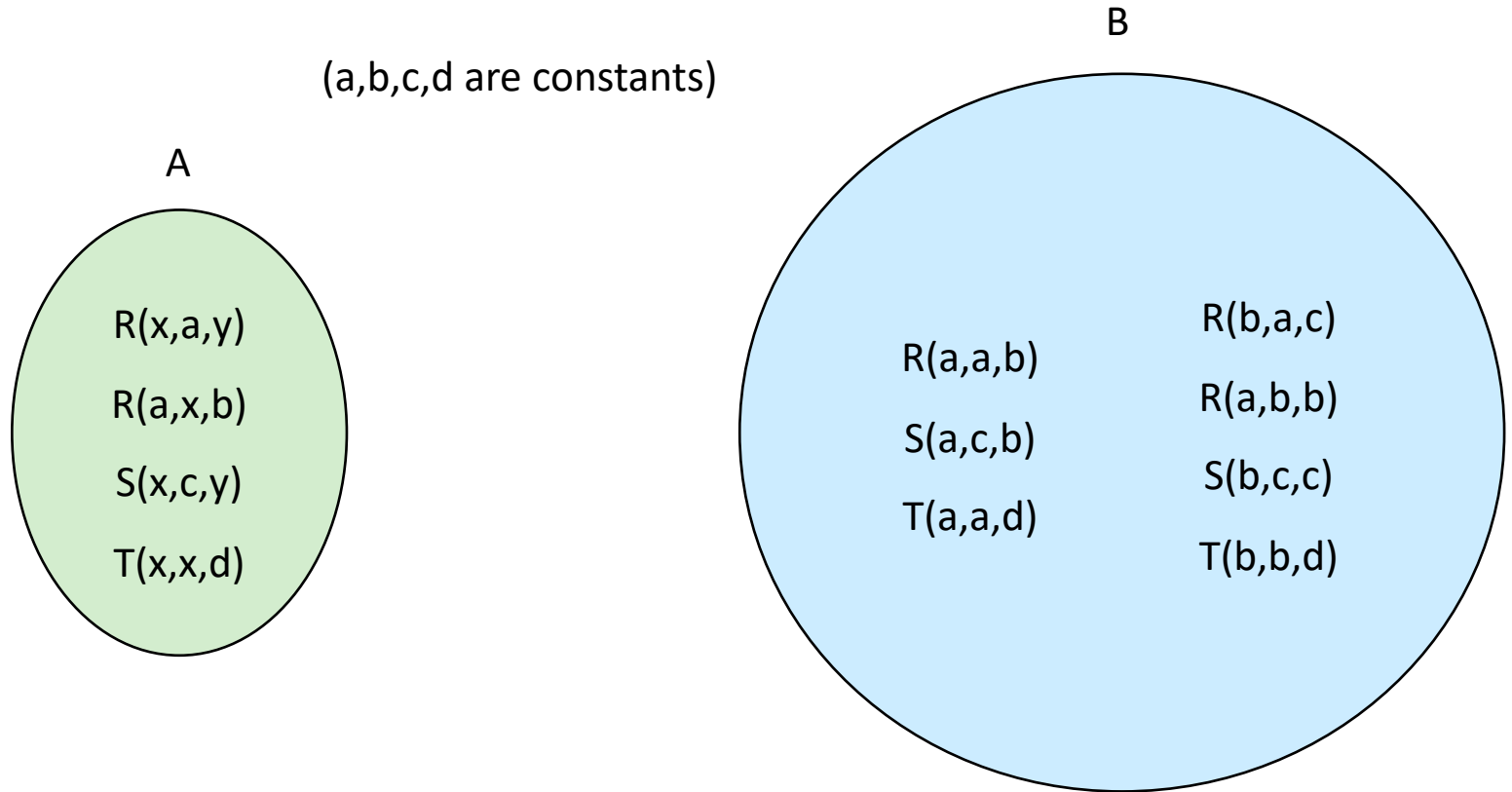
- A **homomorphism** from a set of atoms A to a set of atoms B is a function $h : \text{Terms}(A) \rightarrow \text{Terms}(B)$ such that:
 1. t is a constant of **Const** $\Rightarrow h(t) = t$
 2. $R(t_1, \dots, t_k) \in A \Rightarrow h(R(t_1, \dots, t_k)) = R(h(t_1), \dots, h(t_k)) \in B$
- If $h(\mathbf{u}) = \mathbf{v}$, where \mathbf{u} and \mathbf{v} are tuples of the same length over $\text{Terms}(A)$ and $\text{Terms}(B)$, respectively, then h is a homomorphism from (A, \mathbf{u}) to (B, \mathbf{v})
- We write $A \rightarrow B$ if there exists a homomorphism from A to B, and $(A, \mathbf{u}) \rightarrow (B, \mathbf{v})$ if there exists a homomorphism from (A, \mathbf{u}) to (B, \mathbf{v})

Homomorphism

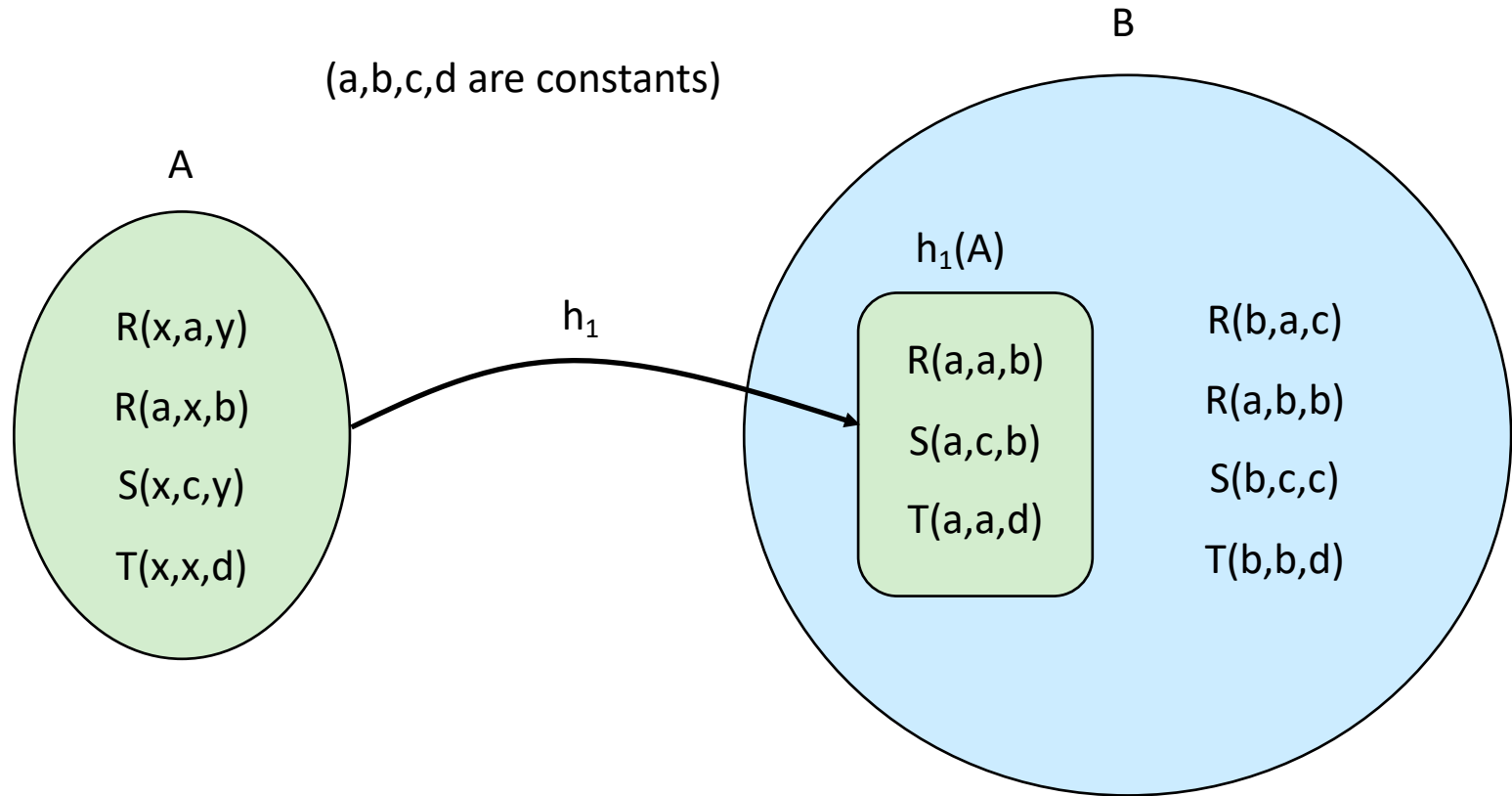


$h : \text{Terms}(A) \rightarrow \text{Terms}(B)$ that is the identity on constants

Homomorphism

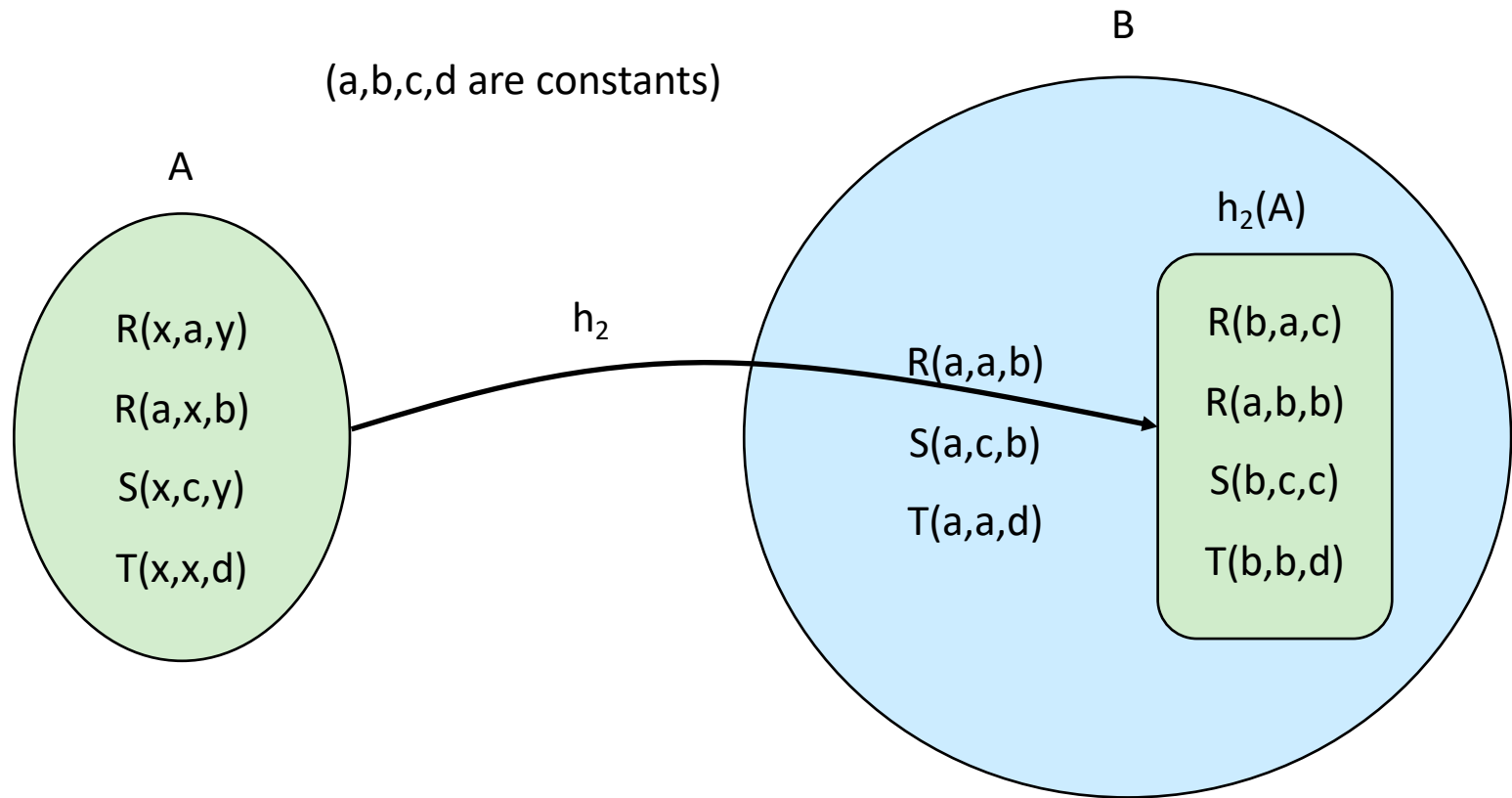


Homomorphism



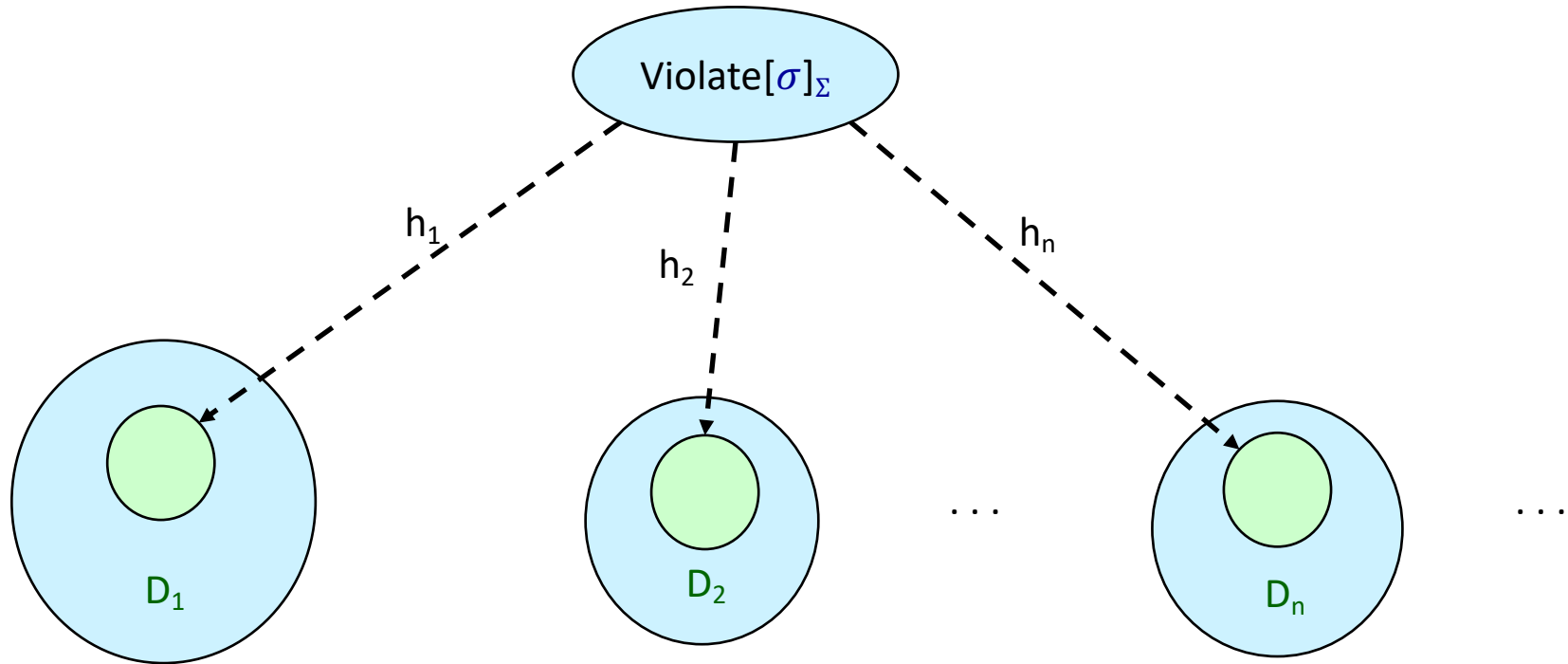
$$h_1 = \{a \mapsto a, b \mapsto b, c \mapsto c, d \mapsto d, x \mapsto a, y \mapsto b\}$$

Homomorphism



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

“Representative” Set of Atoms



1. $\text{Violate}[\sigma]_{\Sigma}$ satisfies Σ
2. For each database D such that $\text{Violate}[\sigma] \rightarrow D$ and $D \models \Sigma$, it holds that $\text{Violate}[\sigma]_{\Sigma} \rightarrow D$

Characterizing Implication for FDs

Given a set Σ of FDs over a schema \mathbf{S} , and an FD σ over \mathbf{S} ,

transform $\text{Violate}[\sigma]$ into a *finite* set of relational atoms $\text{Violate}[\sigma]_{\Sigma}$ that satisfies Σ such that

$$\Sigma \models \sigma \iff \text{Violate}[\sigma]_{\Sigma} \models \sigma$$



acts as a “representative” of all the
database instances of \mathbf{S} that satisfy Σ



We get an algorithm for **FD-Implication**:

if $\text{Violate}[\sigma]_{\Sigma} \models \sigma$ then

return **true**

else

return **false**

Characterizing Implication for FDs

Given a set Σ of FDs over a schema S , and an FD σ over S ,

transform $\text{Violate}[\sigma]$ into a set of relational atoms $\text{Violate}[\sigma]_{\Sigma}$ that satisfies Σ such that

$$\Sigma \models \sigma \iff \text{Violate}[\sigma]_{\Sigma} \models \sigma$$

acts as a “representative” of all the
database instances of S that satisfy Σ

**Chase Procedure
for FDs**



We get an algorithm for **FD-Implication**:

if $\text{Violate}[\sigma]_{\Sigma} \models \sigma$ then

 return **true**

else

 return **false**

FD-Chase by Example

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R : \{2\} \rightarrow \{1\} \\ R : \{1\} \rightarrow \{3\} \end{array} \right\}$$

FD-Chase by Example

R	att ₁	att ₂	att ₃
---	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R : \{2\} \rightarrow \{1\} \\ R : \{1\} \rightarrow \{3\} \end{array} \right\}$$

$$\{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

FD-Chase by Example

R	att ₁	att ₂	att ₃
---	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R : \{2\} \rightarrow \{1\} \\ R : \{1\} \rightarrow \{3\} \end{array} \right\}$$

$$\{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

$$\{R(x_1, x_2, x_3), R(x_1, x_2, y_3)\}$$

FD-Chase by Example

R	att ₁	att ₂	att ₃
---	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R : \{2\} \rightarrow \{1\} \\ R : \{1\} \rightarrow \{3\} \end{array} \right\}$$

$$\{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

$$\{R(x_1, x_2, x_3), R(x_1, x_2, y_3)\}$$

$$\{R(x_1, x_2, x_3), R(x_1, x_2, x_3)\}$$

FD-Chase by Example

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R : \{2\} \rightarrow \{1\} \\ R : \{1\} \rightarrow \{3\} \end{array} \right\}$$

$$\{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

$$\{R(x_1, x_2, x_3), R(x_1, x_2, y_3)\}$$

$$\{R(x_1, x_2, x_3)\}$$

FD-Chase: Applicability

Consider a finite (constant-free) set of atoms A , and an FD σ of the form $R : X \rightarrow Y$

- σ is **applicable** to A with $((u_1, \dots, u_n), (v_1, \dots, v_n))$, where $R(u_1, \dots, u_n)$ and $R(v_1, \dots, v_n)$ are atoms of A , if $\pi_X(u_1, \dots, u_n) = \pi_X(v_1, \dots, v_n)$ and $\pi_Y(u_1, \dots, u_n) \neq \pi_Y(v_1, \dots, v_n)$

- Let $h_{(u,v)} : \text{Terms}(A) \rightarrow \text{Terms}(A)$ such that, for each $w \in \text{Terms}(A)$,

$$h_{(u,v)}(w) = \begin{cases} u_i & \text{if } w = v_i \text{ and } u_i < v_i \text{ for } i \in \{1, \dots, k\} \\ v_i & \text{if } w = u_i \text{ and } v_i < u_i \text{ for } i \in \{1, \dots, k\} \\ w & \text{otherwise} \end{cases}$$

($<$ - lexicographic order over \mathbf{Var})

- The **result** of applying σ to A with (u,v) is the set of atoms $A' = h_{(u,v)}(A) - \mathbf{A}[\sigma, (u,v)]\mathbf{A}'$

FD-Chase by Example

R	att ₁	att ₂	att ₃
---	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\} \quad \Sigma = \left\{ \begin{array}{l} R : \{2\} \rightarrow \{1\} \\ R : \{1\} \rightarrow \{3\} \end{array} \right\}$$

$$\{R(x_1, x_2, x_3), R(y_1, x_2, y_3)\}$$

$$[R : \{2\} \rightarrow \{1\}, ((x_1, x_2, x_3), (y_1, x_2, y_3))]$$

$$\{R(x_1, x_2, x_3), R(x_1, x_2, y_3)\}$$

$$[R : \{1\} \rightarrow \{3\}, ((x_1, x_2, x_3), (x_1, x_2, y_3))]$$

$$\{R(x_1, x_2, x_3)\}$$

FD-Chase: Chase Sequence

Consider a finite (constant-free) set of atoms A , and a set Σ of FDs

A **finite chase sequence** of A under Σ is a finite sequence of sets of atoms A_0, \dots, A_n , where

- $A = A_0$
- for each $i \in \{0, \dots, n-1\}$, there exists an FD $\sigma = R : X \rightarrow Y$ in Σ , and atoms $R(\mathbf{u})$ and $R(\mathbf{v})$ in A_i such that $A_i[\sigma, (\mathbf{u}, \mathbf{v})]A_{i+1}$
- for every FD $\sigma = R : X \rightarrow Y$ in Σ , and atoms $R(\mathbf{u})$ and $R(\mathbf{v})$ in A , σ is **not** applicable to A_n with (\mathbf{u}, \mathbf{v})

FD-Chase: Chase Sequence

Consider a finite (constant-free) set of atoms A , and a set Σ of FDs

An **infinite chase sequence** of A under Σ is an infinite sequence A_0, A_1, \dots where

- $A = A_0$
- for each $i \geq 0$, there exists an FD $\sigma = R : X \rightarrow Y$ in Σ , and atoms $R(\mathbf{u})$ and $R(\mathbf{v})$ in A_i
such that $A_i[\sigma, (\mathbf{u}, \mathbf{v})]A_{i+1}$

FD-Chase: Chase Homomorphism

Consider a finite (constant-free) set of atoms A , and a set Σ of FDs

For a finite chase sequence $s = A_0, \dots, A_n$ of A under Σ such that

$$A_0[\sigma_0, (\mathbf{u}_0, \mathbf{v}_0)]A_1[\sigma_1, (\mathbf{u}_1, \mathbf{v}_1)]A_2 \cdots A_{n-1}[\sigma_{n-1}, (\mathbf{u}_{n-1}, \mathbf{v}_{n-1})]A_n$$

we define its **chase homomorphism**, denoted h_s , as the composition of functions

$$h_{(\mathbf{u}_0, \mathbf{v}_0)} \circ h_{(\mathbf{u}_1, \mathbf{v}_1)} \circ \cdots \circ h_{(\mathbf{u}_n, \mathbf{v}_n)}$$

$$h_s(A_0) = h_s(A) = A_n$$

FD-Chase: Key Properties

Consider a finite (constant-free) set of atoms A , and a set Σ of FDs

- There is no infinite chase sequence of A under Σ
- Let A_0, \dots, A_n and B_0, \dots, B_m be finite chase sequences of A under Σ . Then $A_n = B_m$
 \Rightarrow we can refer to **the** result of the chase of A under Σ , denoted $\text{Chase}(A, \Sigma)$
 \Rightarrow we can refer to **the** chase homomorphism of A under Σ , denoted $h_{A, \Sigma}$
- $\text{Chase}(A, \Sigma) \models \Sigma$
- $\text{Chase}(A, \Sigma)$ can be computed in polynomial time
- Let A' be a set of atom such that $(A, \mathbf{u}) \rightarrow (A', \mathbf{v})$ and $A' \models \Sigma$. Then $(\text{Chase}(A, \Sigma), h_{A, \Sigma}(\mathbf{u})) \rightarrow (A', \mathbf{v})$

Characterizing Implication for FDs

Proposition: Consider a set Σ of FDs over a schema S , and an FD σ over S . It holds that

$$\Sigma \models \sigma \iff \text{Chase}(\text{Violate}[\sigma], \Sigma) \models \sigma$$

↓

We get an algorithm for **FD-Implication**:

```
if Chase(Violate[ $\sigma$ ],  $\Sigma$ )  $\models$   $\sigma$  then
    return true
else
    return false
```

↓

Theorem: FD-Implication is feasible in **polynomial time**

Recap

- Integrity constraints - specify semantic properties
- Syntax and semantics of FDs and INDs
- Basic algorithmic tasks: satisfaction and logical implication
- Both **FD-** and **IND-Satisfaction** are feasible in polynomial time
- **FD-Implication** is feasible in polynomial time - FD-Chase our main tool

Characterizing Implication for INDs

For an IND σ of the form $R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$, we define the singleton set

$$\text{Violate}[\sigma] = \{R(x_1, \dots, x_n)\}$$

where x_1, \dots, x_n are distinct variables

the prototypical set of
atoms that violates σ

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

P	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$\text{Violate}[\sigma] = \{R(x_1, x_2, x_3)\}$$

$$\sigma = R[2,1] \subseteq P[1,3]$$

Characterizing Implication for INDs

Given a set Σ of INDs over a schema \mathbf{S} , and an IND σ over \mathbf{S} ,

transform $\text{Violate}[\sigma]$ into a *finite* set of relational atoms $\text{Violate}[\sigma]_{\Sigma}$ that satisfies Σ such that

$$\Sigma \models \sigma \iff \text{Violate}[\sigma]_{\Sigma} \models \sigma$$



acts as a “representative” of all the
database instances of \mathbf{S} that satisfy Σ



We get an algorithm for **IND-Implication**:

```
if  $\text{Violate}[\sigma]_{\Sigma} \models \sigma$  then  
    return true  
else  
    return false
```


Characterizing Implication for INDs

Given a set Σ of INDs over a schema S , and an IND σ over S ,

transform $\text{Violate}[\sigma]$ into a *finite* set of relational atoms $\text{Violate}[\sigma]_\Sigma$ that satisfies Σ such that

$$\Sigma \models \sigma \iff \text{Violate}[\sigma]_\Sigma \models \sigma$$

acts as a “representative” of all the
database instances of S that satisfy Σ

**Chase Procedure
for INDs**



We get an algorithm for **IND-Implication**:

```
if  $\text{Violate}[\sigma]_\Sigma \models \sigma$  then  
    return true  
else  
    return false
```

IND-Chase by Example

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

P	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R[2] \subseteq P[1] \\ P[1,2] \subseteq P[2,1] \end{array} \right\}$$

IND-Chase by Example

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

P	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R[2] \subseteq P[1] \\ P[1,2] \subseteq P[2,1] \end{array} \right\}$$

$$\{R(x_1, x_2, x_3)\}$$

IND-Chase by Example

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

P	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R[2] \subseteq P[1] \\ P[1,2] \subseteq P[2,1] \end{array} \right\}$$

$$\{R(x_1, x_2, x_3)\}$$

$$\{R(x_1, x_2, x_3), P(x_2, y_1, y_2)\}$$

IND-Chase by Example

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

P	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R[2] \subseteq P[1] \\ P[1,2] \subseteq P[2,1] \end{array} \right\}$$

$$\{R(x_1, x_2, x_3)\}$$

$$\{R(x_1, x_2, x_3), P(x_2, y_1, y_2)\}$$

$$\{R(x_1, x_2, x_3), P(x_2, y_1, y_2), P(y_1, x_2, y_3)\}$$

IND-Chase by Example

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

P	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R[2] \subseteq P[1] \\ P[1,2] \subseteq P[2,1] \end{array} \right\}$$

$$\{R(x_1, x_2, x_3)\}$$

$$\{R(x_1, x_2, x_3), P(x_2, y_1, y_2)\}$$

$$\{R(x_1, x_2, x_3), P(x_2, y_1, y_2), P(y_1, x_2, y_3)\}$$

IND-Chase by Example

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

P	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3)\} \quad \Sigma = \left\{ R[2,3] \subseteq R[1,3] \right\}$$

$\{R(x_1, x_2, x_3)\}$

$\{R(x_1, x_2, x_3), R(x_2, y_1, x_3)\}$

$\{R(x_1, x_2, x_3), R(x_2, y_1, x_3), R(y_1, y_2, x_3)\}$

$\{R(x_1, x_2, x_3), R(x_2, y_1, x_3), R(y_1, y_2, x_3), R(y_2, y_3, x_3)\}$

\vdots

IND-Chase: Applicability

Consider a finite set of atoms A , and an IND σ of the form $R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$

- σ is **applicable** to A with $\mathbf{u} = (u_1, \dots, u_n)$ if $R(u_1, \dots, u_n)$ belongs to A
- Let $\text{New}(\sigma, \mathbf{u}) = P(v_1, \dots, v_m)$, where, for each $\ell \in \{1, \dots, m\}$,

$$v_\ell = \begin{cases} u_{i_r} & \text{if } \ell = j_r \text{ for } r \in \{1, \dots, k\} \\ x[\ell; \sigma, \mathbf{u}] \in \mathbf{Var} \setminus \text{Terms}(A) & \text{otherwise} \end{cases}$$

- The **result** of applying σ to A with \mathbf{u} is the set of atoms $A' = A \cup \{\text{New}(\sigma, \mathbf{u})\}$ - $\mathbf{A}[\sigma, \mathbf{u}]\mathbf{A}'$

IND-Chase by Example

R	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

P	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$A = \{R(x_1, x_2, x_3)\}$$

$$\Sigma = \left\{ \begin{array}{l} R[2] \subseteq P[1] \\ P[1,2] \subseteq P[2,1] \end{array} \right\}$$

$$\{R(x_1, x_2, x_3)\}$$

$$[R[2] \subseteq P[1], (x_1, x_2, x_3)]$$

$$\{R(x_1, x_2, x_3), P(x_2, y_1, y_2)\}$$

$$[P[1,2] \subseteq P[2,1], (x_2, y_1, y_2)]$$

$$\{R(x_1, x_2, x_3), P(x_2, y_1, y_2), P(y_1, x_2, y_3)\}$$

IND-Chase: Chase Sequence

Consider a set of atoms A , and a set Σ of INDs

A **finite chase sequence** of A under Σ is a finite sequence of sets of atoms A_0, \dots, A_n , where

- $A = A_0$
- for each $i \in \{0, \dots, n-1\}$, there exists an IND $\sigma = R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$ in Σ , and an atom $R(\mathbf{u}) \in A_i$ such that $\text{New}(\sigma, \mathbf{u}) \notin A_i$ and $A_i[\sigma, \mathbf{u}]A_{i+1}$
- for every IND $\sigma = R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$ in Σ , and atom $R(\mathbf{u})$ in A , $\text{New}(\sigma, \mathbf{u}) \in A_n$

The **result** of the chase sequence A_0, \dots, A_n is the set of atoms A_n

IND-Chase: Chase Sequence

Consider a set of atoms A , and a set Σ of INDs

An **infinite chase sequence** of A under Σ is a finite sequence of sets of atoms A_0, A_1, \dots where

- $A = A_0$
- for each $i \geq 0$, there exists an IND $\sigma = R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$ in Σ , and an atom $R(\mathbf{u}) \in A_i$ such that $\text{New}(\sigma, \mathbf{u}) \notin A_i$ and $A_i[\sigma, \mathbf{u}] \subseteq A_{i+1}$
- for each $i \geq 0$, and for each IND $\sigma = R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$ in Σ and atom $R(\mathbf{u}) \in A_i$ such that σ is applicable to A_i with \mathbf{u} , there exists $j > i$ such that $\text{New}(\sigma, \mathbf{u}) \in A_j$ - **fairness**

IND-Chase: Fairness

R	att ₁	att ₂
----------	------------------	------------------

P	att ₁	att ₂
----------	------------------	------------------

$$A = \{R(x_1, x_2)\} \quad \Sigma = \left\{ \begin{array}{l} R[2] \subseteq R[1] \\ R[1] \subseteq P[1] \end{array} \right\}$$

$\{R(x_1, x_2)\}$

$\{R(x_1, x_2), R(x_2, x_3)\}$

$\{R(x_1, x_2), R(x_2, x_3), R(x_3, x_4)\}$

$\{R(x_1, x_2), R(x_2, x_3), R(x_3, x_4), R(x_4, x_5)\}$

\vdots

IND-Chase: Fairness

R	att ₁	att ₂
---	------------------	------------------

P	att ₁	att ₂
---	------------------	------------------

$$A = \{R(x_1, x_2)\}$$

$$\Sigma = \left\{ \begin{array}{l} R[2] \subseteq R[1] \\ R[1] \subseteq P[1] \end{array} \right\}$$

$\{R(x_1, x_2)\}$

$\{R(x_1, x_2), R(x_2, x_3)\}$

$\{R(x_1, x_2), R(x_2, x_3), R(x_3, x_4)\}$

$\{R(x_1, x_2), R(x_2, x_3), R(x_3, x_4), R(x_4, x_5)\}$

\vdots

does not satisfy Σ

IND-Chase: Chase Sequence

Consider a set A , and a set Σ of INDs

An **infinite chase sequence** of A under Σ is a finite sequence of sets of atoms A_0, A_1, \dots where

- $A = A_0$
- for each $i \geq 0$, there exists an IND $\sigma = R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$ in Σ , and an atom $R(\mathbf{u}) \in A_i$ such that $\text{New}(\sigma, \mathbf{u}) \notin A_i$ and $A_i[\sigma, \mathbf{u}] = A_{i+1}$
- for each $i \geq 0$, and for each IND $\sigma = R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$ in Σ and atom $R(\mathbf{u}) \in A_i$ such that σ is applicable to A_i with \mathbf{u} , there exists $j > i$ such that $\text{New}(\sigma, \mathbf{u}) \in A_j$ - **fairness**

The **result** of the chase sequence A_0, A_1, \dots is the set of atoms $A_0 \cup A_1 \cup \dots$

IND-Chase: Key Properties

Consider a finite set of atoms A , and a set Σ of INDs

- All chase sequences of A under Σ are either finite or infinite
- Let A_0, \dots, A_n and B_0, \dots, B_m be finite chase sequences of A under Σ . Then $A_n = B_m$
- Let A_0, A_1, \dots and B_0, B_1, \dots be infinite chase sequences of A under Σ . Then $\bigcup A_i = \bigcup B_i$
 \Rightarrow we can refer to **the** result of the chase of A under Σ , denoted $\text{Chase}(A, \Sigma)$
- $\text{Chase}(A, \Sigma) \models \Sigma$
- Let A' be a set of atom such that $(A, \mathbf{u}) \rightarrow (A', \mathbf{v})$ and $A' \models \Sigma$. Then $(\text{Chase}(A, \Sigma), \mathbf{u}) \rightarrow (A', \mathbf{v})$

Characterizing *Unrestricted* Implication for INDs

A set Σ of INDs over a schema \mathbf{S} **implies without restriction** an IND σ over \mathbf{S} , denoted $\Sigma \models_{\text{unr}} \sigma$, if, for every (**possibly infinite**) database instance \mathbf{D} of \mathbf{S} , $\mathbf{D} \models \Sigma \Rightarrow \mathbf{D} \models \sigma$

Proposition: Consider a set Σ of INDs over a schema \mathbf{S} , and an IND σ over \mathbf{S} . It holds that $\Sigma \models_{\text{unr}} \sigma \iff \text{Chase}(\text{Violate}[\sigma], \Sigma) \models \sigma$

Theorem: Consider a set Σ of INDs over a schema \mathbf{S} , and an IND σ over \mathbf{S} . It holds that $\Sigma \models \sigma \iff \Sigma \models_{\text{unr}} \sigma$ - **finite controllability of logical implication**

Characterizing “*Restricted*” Implication for INDs

A set Σ of INDs over a schema \mathbf{S} **implies without restriction** an IND σ over \mathbf{S} , denoted $\Sigma \models_{\text{unr}} \sigma$, if, for every (**possibly infinite**) database instance \mathbf{D} of \mathbf{S} , $\mathbf{D} \models \Sigma \Rightarrow \mathbf{D} \models \sigma$

Proposition: Consider a set Σ of INDs over a schema \mathbf{S} , and an IND σ over \mathbf{S} . It holds that $\Sigma \models_{\text{unr}} \sigma \iff \text{Chase}(\text{Violate}[\sigma], \Sigma) \models \sigma$

Theorem: Consider a set Σ of INDs over a schema \mathbf{S} , and an IND σ over \mathbf{S} . It holds that $\Sigma \models \sigma \iff \Sigma \models_{\text{unr}} \sigma$ - **finite controllability of logical implication**



Corollary: Consider a set Σ of INDs over a schema \mathbf{S} , and an IND σ over \mathbf{S} . It holds that $\Sigma \models \sigma \iff \text{Chase}(\text{Violate}[\sigma], \Sigma) \models \sigma$

An Algorithm for IND-Implication

Corollary: Consider a set Σ of INDs over a schema \mathbf{S} , and an IND σ over \mathbf{S} . It holds that

$$\Sigma \models \sigma \iff \text{Chase}(\text{Violate}[\sigma], \Sigma) \models \sigma$$

⇓

We get an algorithm for **IND-Implication**



if $\text{Chase}(\text{Violate}[\sigma], \Sigma) \models \sigma$ then

return **true**

else

return **false**

the result of the chase might be infinite

An Algorithm for IND-Implication

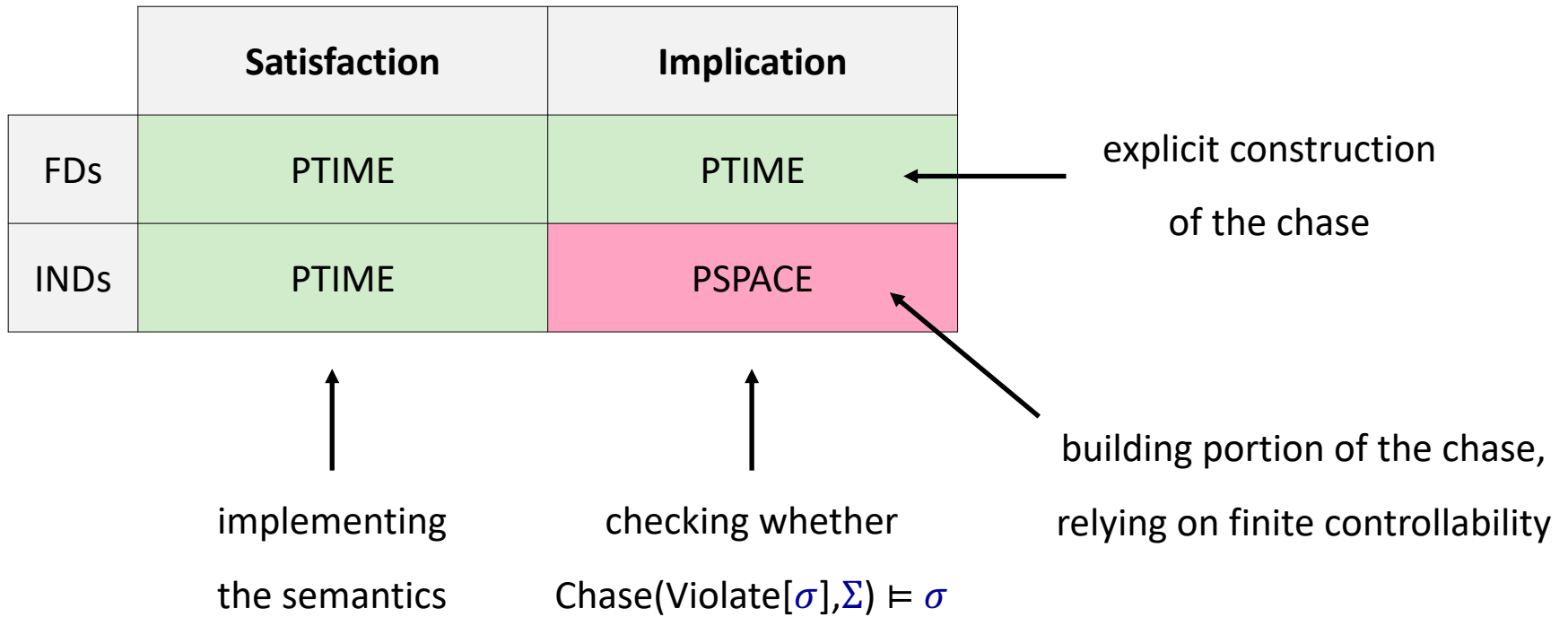
Consider a set Σ of INDs over a schema \mathbf{S} , and an IND $\sigma = R[i_1, \dots, i_k] \subseteq P[j_1, \dots, j_k]$ over \mathbf{S}

- We need to check whether $\text{Chase}(\text{Violate}[\sigma], \Sigma) \models \sigma$
- Recall that $\text{Violate}[\sigma] = \{R(x_1, \dots, x_n)\} \Rightarrow$ our task boils down to checking whether $\text{Chase}(\text{Violate}[\sigma], \Sigma)$ contains an atom $P(y_1, \dots, y_n)$ with $\pi_{(i_1, \dots, i_k)}(x_1, \dots, x_n) = \pi_{(j_1, \dots, j_k)}(y_1, \dots, y_n)$
- Construct non-deterministically, starting from $R(x_1, \dots, x_n)$, a sequence of atoms via chase steps without storing more than two consecutive atoms



Theorem: IND-Implication is feasible in **polynomial space**

Recap



Integrity Constraints
&
Static Analysis of Queries

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

List the airlines that fly directly from London to Glasgow

Flight	origin	destination	airline
	VIE	LHR	BA
	LHR	EDI	BA
	LGW	GLA	U2
	LCA	VIE	OS

Airport	code	city
	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

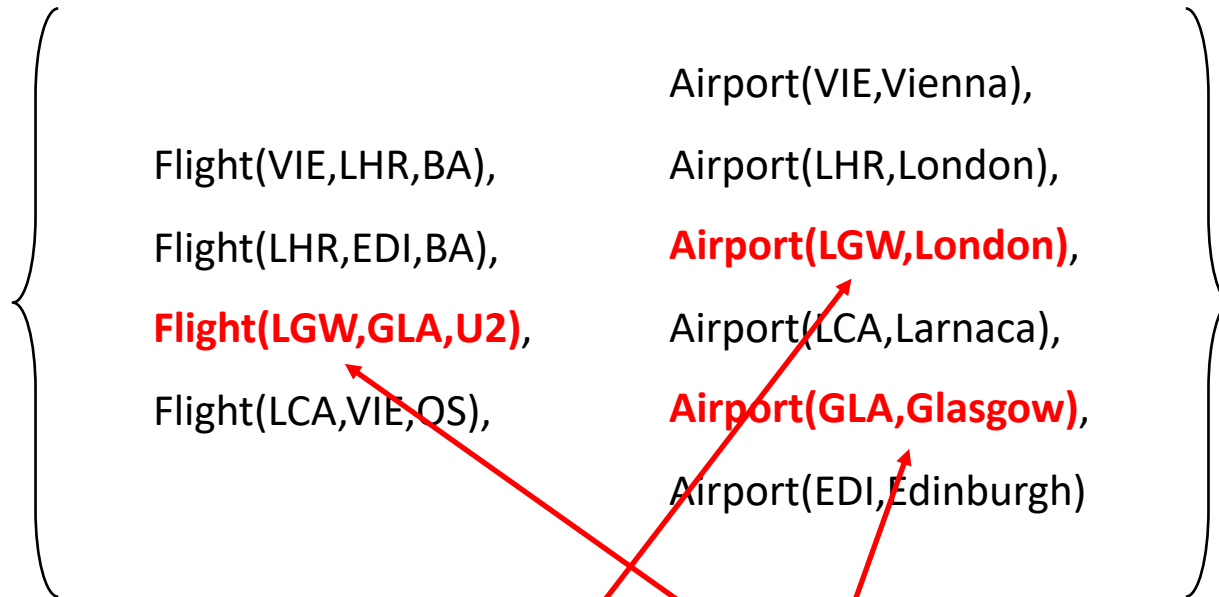
List the airlines that fly directly from London to Glasgow

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



$\{x \mapsto \text{LGW}, y \mapsto \text{GLA}, z \mapsto \text{U2},$
 $\text{London} \mapsto \text{London}, \text{Glasgow} \mapsto \text{Glasgow}\}$

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

Containment of CQs

Given two CQs Q_1 and Q_2 over a schema S , Q_1 is **contained** in Q_2 , denoted $Q_1 \subseteq Q_2$, if $Q_1(D) \subseteq Q_2(D)$ for every database instance D of S

CQ-Containment

Input: two conjunctive queries Q_1 and Q_2

Output: **true** if $Q_1 \subseteq Q_2$, and **false** otherwise

*Crucial task in
query optimization*

- Replace a query Q_1 with a query Q_2 that is easier to evaluate
- But, we have to guarantee that $Q_1(D) = Q_2(D)$ for every database D
- This boils down to two containment checks: $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$

Homomorphism Theorem

A **query homomorphism** from $Q_1(x_1, \dots, x_k) :- \text{body}_1$ to $Q_2(y_1, \dots, y_k) :- \text{body}_2$

is a function $h : \text{Terms}(\text{body}_1) \rightarrow \text{Terms}(\text{body}_2)$ such that:

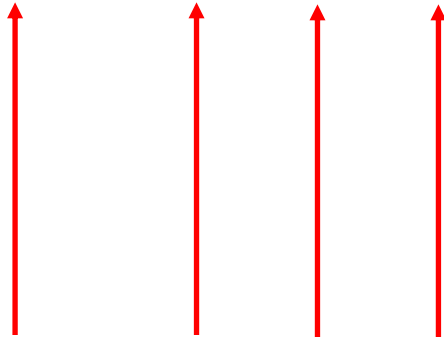
1. h is a homomorphism from body_1 to body_2
2. $(h(x_1), \dots, h(x_k)) = (y_1, \dots, y_k)$

Homomorphism Theorem: Let Q_1 and Q_2 be conjunctive queries. It holds that

$$Q_1 \subseteq Q_2 \iff \text{there exists a query homomorphism from } Q_2 \text{ to } Q_1$$

Homomorphism Theorem: Example

$Q_1(x,y) :- R(x,z), S(z,z), R(z,y)$



$h = \{t \mapsto x, u \mapsto y, v \mapsto z, w \mapsto z\}$

$Q_2(t,u) :- R(t,v), S(v,w), R(w,u)$

- h is a query homomorphism from Q_2 to $Q_1 \Rightarrow Q_1 \subseteq Q_2$
- But, there is no homomorphism from Q_1 to $Q_2 \Rightarrow Q_1 \not\subseteq Q_2$

Existence of a Query Homomorphism

Theorem: Let Q_1 and Q_2 be conjunctive queries. The problem of deciding whether there exists a query homomorphism from Q_2 to Q_1 is NP-complete

(NP-membership) Guess a function, and verify that it is a query homomorphism

(NP-hardness) Reduction from **3-Colorability**

By applying the homomorphism theorem we get that:

Corollary: CQ-Containment is NP-complete

Containment of CQs Under FDs

R	att ₁	att ₂
---	------------------	------------------

Q_1 :- R(x,y), R(y,z), R(z,x), R(x,z)

Q_2 :- R(x,y), R(y,y), R(y,x)

there is no query homomorphism from Q_2 to Q_1 \Rightarrow Q_1 is not contained in Q_2

but, what if we focus on database instances that satisfy $R : \{1\} \rightarrow \{2\}$?

Q_1 is now contained in Q_2

Containment of CQs Under INDs

R	att ₁	att ₂
----------	------------------	------------------

P	att ₁	att ₂	att ₃
----------	------------------	------------------	------------------

$$Q_1 \text{ :- } R(x,y), R(y,z), R(z,x)$$

$$Q_2 \text{ :- } R(x,y), R(y,z), R(z,x), P(x,y,z)$$

there is no query homomorphism from Q_2 to $Q_1 \Rightarrow Q_1$ is not contained in Q_2

but, what about if we focus on database instances that satisfy

$$R[1,2] \subseteq P[1,2] \quad P[2,3] \subseteq R[1,2] \quad P[3,1] \subseteq R[1,2]?$$

Q_1 is now contained in Q_2

Containment of CQs Revisited

Given two CQs Q_1 and Q_2 over a schema S , and a set of constraints Σ over S ,

Q_1 is **contained** in Q_2 **under** Σ , denoted $Q_1 \subseteq_{\Sigma} Q_2$, if

$Q_1(D) \subseteq Q_2(D)$ for every database instance D of S that satisfies Σ

CQ-Containment-FD

Input: two conjunctive queries Q_1 and Q_2 , and a set of Σ FDs

Output: true if $Q_1 \subseteq_{\Sigma} Q_2$, and false otherwise

CQ-Containment-IND

Input: two conjunctive queries Q_1 and Q_2 , and a set of Σ INDs

Output: true if $Q_1 \subseteq_{\Sigma} Q_2$, and false otherwise


Containment of CQs Revisited

Given two CQs Q and Q' over a schema S , and a set Σ of FDs/INDs over S ,

transform Q into a *finite* CQ Q_Σ that satisfies Σ such that

$Q \subseteq_\Sigma Q'$ \Leftrightarrow **there exists a query homomorphism from Q' to Q_Σ**

acts as a “representative” of all
the CQs over S that satisfy Σ



Containment of CQs Under FDs

(we assume constant-free CQs)

Theorem: Let $Q_1(\mathbf{x})$ and $Q_2(\mathbf{y})$ be conjunctive queries, and Σ a set of FDs. It holds that

$Q_1 \subseteq_{\Sigma} Q_2 \iff$ there exists a query homomorphism from Q_2 to $\text{Chase}(Q_1, \Sigma)$

its output tuple is $h_{Q_1, \Sigma}(\mathbf{x})$



\Downarrow

Theorem: CQ-Containment-FD is NP-complete

Unrestricted Containment of CQs Under Constraints

Given two CQs Q_1 and Q_2 over a schema S , and a set of constraints Σ over S ,

Q_1 is **contained** in Q_2 **under Σ without restriction**, denoted $Q_1 \subseteq_{\Sigma, \text{unr}} Q_2$, if

$Q_1(D) \subseteq Q_2(D)$ for every (**possibly infinite**) database instance D of S that satisfies Σ

Unrestricted Containment of CQs Under INDs

Proposition: Let Q_1 and Q_2 be conjunctive queries, and Σ a set of INDs. It holds that

$Q_1 \subseteq_{\Sigma, \text{unr}} Q_2 \iff$ there exists a query homomorphism from Q_2 to $\text{Chase}(Q_1, \Sigma)$

Theorem: Let Q_1 and Q_2 be conjunctive queries, and Σ a set of INDs. It holds that

$Q_1 \subseteq_{\Sigma} Q_2 \iff Q_1 \subseteq_{\Sigma, \text{unr}} Q_2$ - **finite controllability of CQ-Containment-IND**

“Restricted” Containment of CQs Under INDs

Proposition: Let Q_1 and Q_2 be conjunctive queries, and Σ a set of INDs. It holds that

$Q_1 \subseteq_{\Sigma, \text{unr}} Q_2 \iff$ there exists a query homomorphism from Q_2 to $\text{Chase}(Q_1, \Sigma)$

Theorem: Let Q_1 and Q_2 be conjunctive queries, and Σ a set of INDs. It holds that

$Q_1 \subseteq_{\Sigma} Q_2 \iff Q_1 \subseteq_{\Sigma, \text{unr}} Q_2$ - **finite controllability of CQ-Containment-IND**



Corollary: Let Q_1 and Q_2 be conjunctive queries, and Σ a set of INDs. It holds that

$Q_1 \subseteq_{\Sigma} Q_2 \iff$ there exists a query homomorphism from Q_2 to $\text{Chase}(Q_1, \Sigma)$

An Algorithm for CQ-Containment-IND

Let Q_1 and Q_2 be conjunctive queries, and Σ a set of INDs

- We need to check whether there exists a query homomorphism from Q_2 to $\text{Chase}(Q_1, \Sigma)$
- Non-deterministically construct a subquery Q' of $\text{Chase}(Q_1, \Sigma)$ with at most $|Q_2|$ atoms, which can be done without storing more than $2 \cdot |Q_2|$ atoms at each step
- Guess a function h , and verify that is a query homomorphism from Q_2 to Q'



Theorem: CQ-Containment-IND is feasible in **polynomial space**

Recap

	Satisfaction	Implication	CQ-Containment
FDs	PTIME	PTIME*	NP*
INDs	PTIME	PSPACE**	PSPACE**

↗
implementing
the semantics

↑
checking whether
 $\text{Chase}(\text{Violate}[\sigma], \Sigma) \models \sigma$

↖
checking for query hom.
from Q_2 to $\text{Chase}(Q_1, \Sigma)$

*explicit construction of the chase

**building portion of the chase, relying on finite controllability

Concluding Remark

**the chase procedure is a fundamental algorithmic tool
for reasoning with constraints**

- Computing data exchange solutions
- Computing certain answers in data integration settings
- Ontology-mediated query answering
- Data cleaning
- ...