Efficiently querying incomplete data

Cristina Sirangelo
LSV, ENS-Cachan
joint work with Amélie Gheerbrant and Leonid Libkin

February 3rd 2015

Plan

- ➡ Introduction
- ➡ Incomplete data model
- ➡ Querying incomplete data
  - ▸ the key to tractability: naïve evaluation
- ➡ What makes naïve evaluation work?
  - ▸ a general framework
- ➡ Applicability of the framework
- ➡ Moving forward

Incomplete data

Data incompleteness: missing/unknown data values, partially available data, ...

Reasons:

mistakes: wrong/missing entries

restrictions on data access

data heterogeneity: data exchange/integration

Incompleteness and data heterogeneity

Assume data has to be transferred from one data source to another

Emp

name	*dpt*
Green	*sales*
White	*sales*
Brown	*eng*
Black	*eng*

source database

data restructuring

Emploee

name
Green
White
Brown
Black

Manager

mgr	*emp*
?	*Green*
?	*White*
?	*Brown*
?	*Black*

target database

Commonly only some concepts shared by the two data sources

- ✓ e.g. no information about the manager from the source

Querying incomplete data

Emploee

name
Green
White
Brown
Black

Manager

mgr	*emp*
?	*Green*
?	*White*
?	*Brown*
?	*Black*

incomplete database

Q: which employees are managers?

?

Querying incomplete data

Semantics of query answering

- How should the result of a query be defined in the presence of incompleteness?

Query evaluation

- How do we evaluate a query on an incomplete database?
- Can this be done efficiently ?

Emploee

name
Green
White
Brown
Black

Manager

mgr	*emp*
?	*Green*
?	*White*
?	*Brown*
?	*Black*

incomplete database

Q: which employees are managers?

?

Incompleteness in theory and practice

Incompleteness in database systems

- Semantics of query answering: poorly designed
- Query evaluation: very efficient, optimized query engines

eg:

- In SQL, the standard relational database query language, the following are consistent statements for sets X, Y

$$|X| > |Y| \text{ and } X - Y = \varnothing$$

- This may occur if Y contains incomplete information (SQL *nulls*)

Incompleteness in theory and practice

Incompleteness in database systems

- Semantics of query answering: poorly designed
- Query evaluation: very efficient, optimized query engines

Theoretical framework for incompleteness

[Imielinski-Lipski, Abiteboul-Kanellakis-Grahne, etc. 80's]

- Semantics of query answering: clean framework, suitable semantics
- Query evaluation: hard

Incompleteness in theory and practice

Incompleteness in database systems

- Semantics of query answering: poorly designed
- Query evaluation: very efficient, optimized query engines

Theoretical framework for incompleteness

[Imielinski-Lipski, Abiteboul-Kanellakis-Grahne, etc. 80's]

- Semantics of query answering: clean framework, suitable semantics
- Query evaluation: hard

Bridging the gap between theory and systems:

answer queries correctly, use classical query engines

not satisfactorily addressed even in the simplest data model

Plan

- Introduction
- Incomplete data model
- Querying incomplete data
  - the key to tractability: naïve evaluation
- What makes naïve evaluation work?
  - a general framework
- Applicability of the framework
- Moving forward

Incomplete relational data

Database schema (relational signature) σ : a set of relation symbols, with arities

eg σ ={ **Employee, Manager** } *arity(*Employee*)*=1, *arity(*Manager*)*=2

Incomplete database instance (naïve table) of schema σ [Imielinski, Lipski '84]:

associates to each relation symbol R of σ a finite subset of $(Const \cup Var)^{arity(R)}$

I

Employee

Green
x_1
Brown

Manager

Green	x_1
x_1	Brown
Green	x_2

Const : a countably infinite set of constants

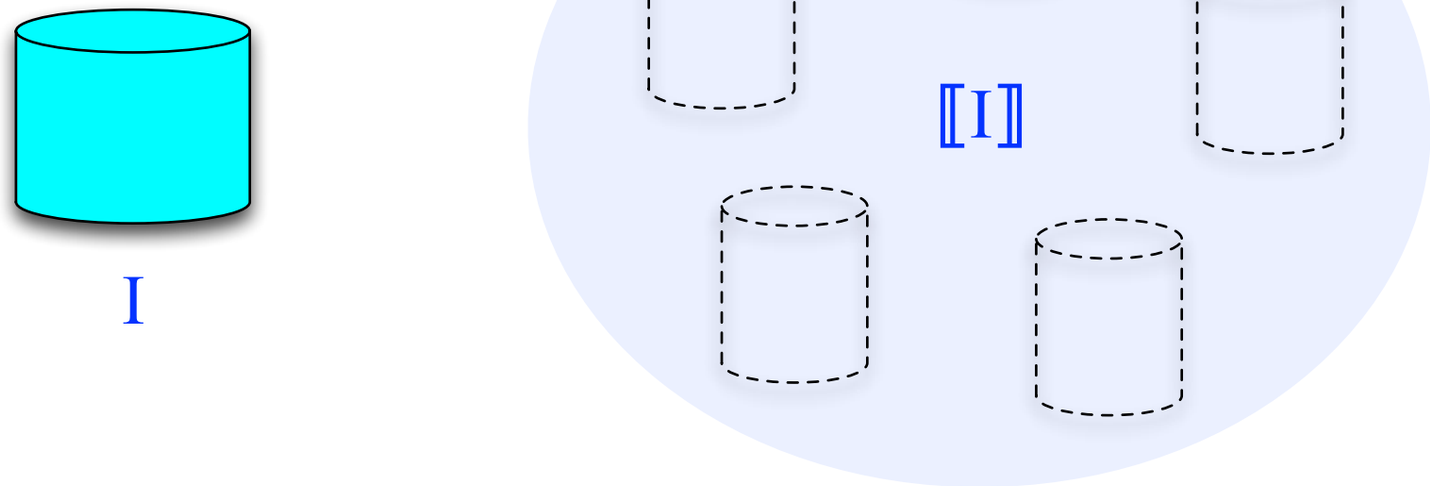
Var : a countably infinite set of variables (nulls)

Complete instance: over *Const*

dom(I) : the subset of *Const* ∪ *Var* occurring in I

Semantics of incompleteness

Any incomplete database represents a set of complete databases (*possible worlds*)



Semantics of incompletness:

a function ⟦ ⟧ associating with each incomplete database a set of complete databases

Semantics of incompleteness

Employee

Green
x_1
Brown

Manager

Green	x_1
x_1	Brown
Green	x_2

I

Three well known relational semantics:

- OWA (*Open World Assumption*) [Imielinski-Lipski '84]
- CWA (*Closed World Assumption*) [Reiter '77, Imielinski-Lipski '84]
- WCWA (*Weak Closed World Assumption*) [Reiter '77]

Semantics of incompleteness

Employee

Green
x_1
Brown

Manager

Green	x_1
x_1	Brown
Green	x_2

I

x_1=White x_2=Black

x_1=x_2=Brown

x_1=White x_2=Black

Green
White
Brown

Green	White
White	Brown
Green	Black

Green
Brown

Green	Brown
Brown	Brown
Black	Brown

Green
White
Brown
Black
Smith

Green	White
White	Brown
Green	Black

⋮

$[\![I]\!]_{OWA}$

OWA:

$[\![I]\!]_{OWA}$ = { D over *Const* | D ⊇ v(I)
for some v: *Var* → *Const* }

v: valuation

Interpretation of incompleteness:

- missing data values, missing tuples

Semantics of incompleteness

Employee

Green
x_1
Brown

Manager

Green	x_1
x_1	Brown
Green	x_2

I

x_1=White x_2=Black

x_1=x_2=Brown

x_1=White x_2=Black

Green	Green	White
White	White	Brown
Brown	Green	Black

Green	Green	Brown
Brown	Brown	Brown

Green	Green	White
White	White	Brown
Brown	Green	Black

⋮

$[\![I]\!]_{CWA}$

CWA:

$[\![I]\!]_{CWA}$ = { D over *Const* | D = v(I)
for some v: *Var* → *Const* }

Interpretation of incompleteness:

- missing data values
- no missing tuples

Semantics of incompleteness

Employee

Green
x_1
Brown

Manager

Green	x_1
x_1	Brown
Green	x_2

I

x_1=White x_2=Black

x_1=x_2=Brown

x_1=White x_2=Black

Green
White
Brown

Green	White
White	Brown
Green	Black

Green
Brown

Green	Brown
Brown	Brown
Green	Green

Green
White
Brown
Black

Green	White
White	Brown
Green	Black
Green	Brown

⋮

$[\![I]\!]_{WCWA}$

WCWA:

$[\![I]\!]_{WCWA}$ ={D over *Const* |

D ⊇ v(I), dom(D)=dom(v(I))

for some v: *Var* → *Const*}

Interpretation of incompleteness:

- missing data values, missing tuples
- no missing domain elements

Plan

- Introduction
- Incomplete data model
- Querying incomplete data
  - the key to tractability: naïve evaluation
- What makes naïve evaluation work?
  - a general framework
- Applicability of the framework
- Moving forward

Queries

Query over σ:

a mapping Q associating to each complete instance I of σ a relation over dom(I)

Q: which employees are managers?

Employee

Green
White
Brown

Manager

Green	White
White	Brown
Green	Black

I

Green
White

Q(I)

usually expressed in fragments of *First Order logic (FO)*

$$\varphi_Q(x) = \text{Employee}(x) \wedge \exists y\ \text{Manager}(x, y)$$

Queries

Boolean query over σ:

a mapping Q associating to each complete instance I of σ a value in {*true*, *false*}

Q: $\exists\, x, y, z\ (\ \mathrm{Manager}(x, y) \wedge\ \mathrm{Manager}(z, x)\ )$ (there is a manager who has a manager)

Employee

Green
White
Brown

Manager

Green	White
White	Brown
Green	Black

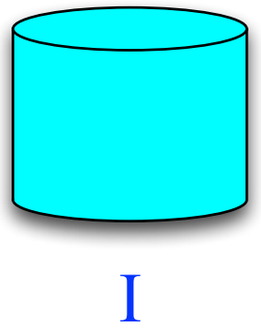
true

I

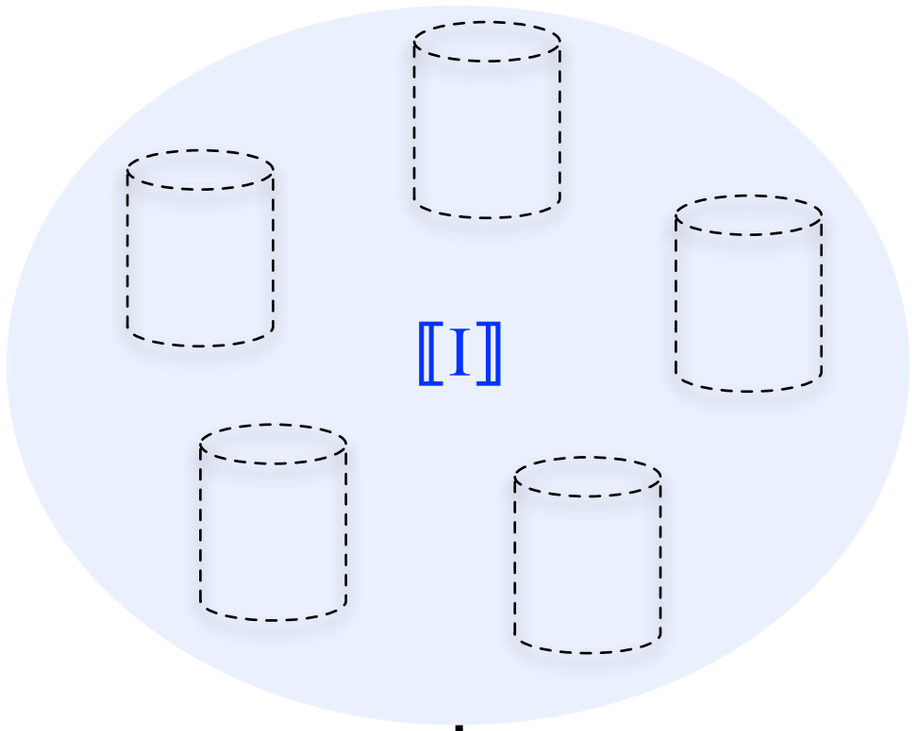
Q(I)

Querying incomplete databases

Semantics of query answering: *certain answers*



I



Q (Boolean)

$$\mathrm{cert}_Q(I) = \bigwedge_{D \in [\![I]\!]} Q(D)$$

Certain answers

Example Q : "There is a manager who has a manager"

$\exists\, x, y, z\ (\ \mathrm{Manager}(x, y) \wedge\ \mathrm{Manager}(z, x)\ )$

Employee

Green
x_1
Brown

Manager

Green	x_1
x_1	Brown
Green	x_2

I

$\mathrm{cert}_Q(I) = \mathit{true}$

under either CWA, OWA and WCWA

⋮

Green
White
Brown

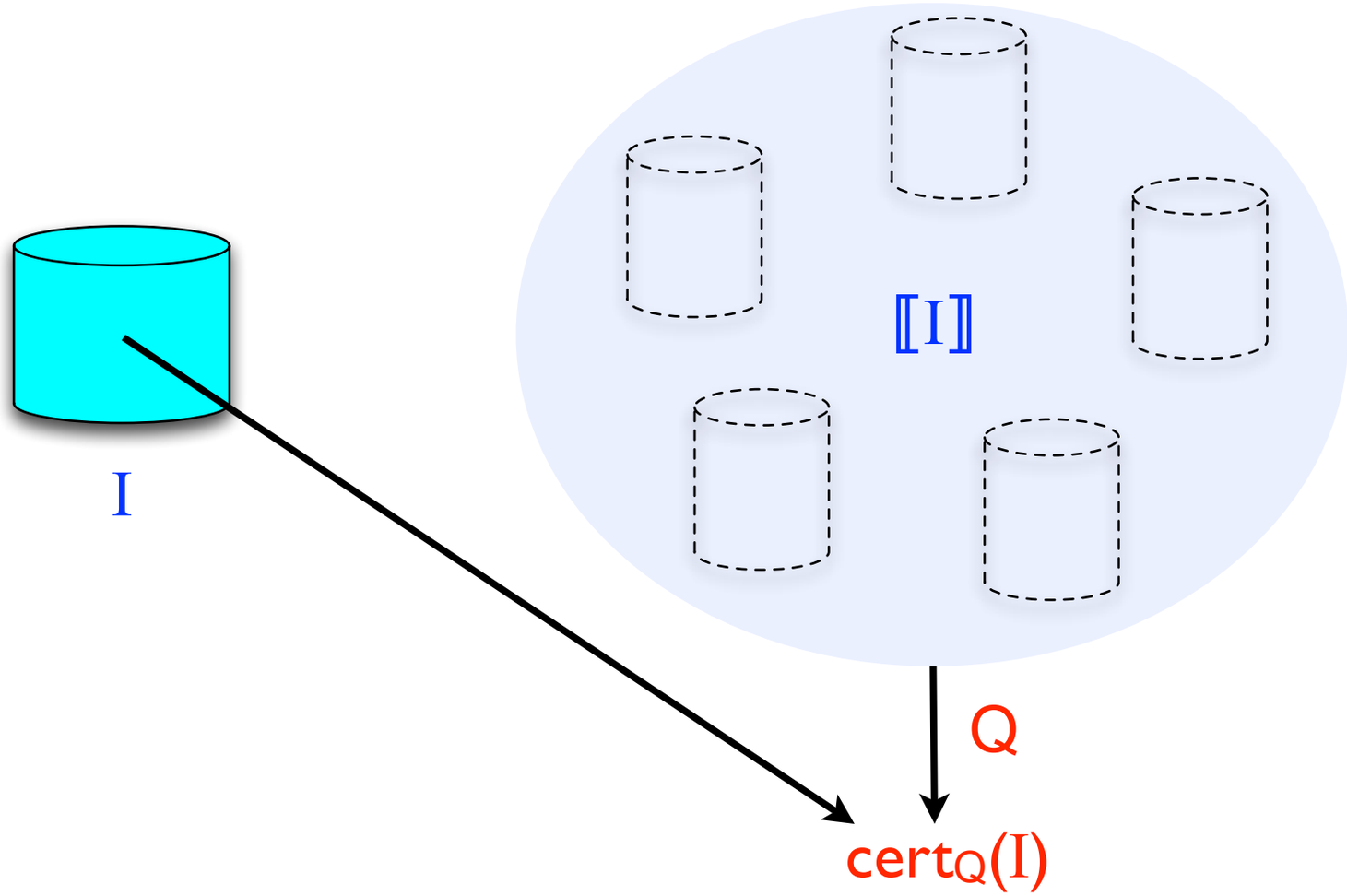
Green	White
White	Brown
Green	Black

⋮

$[\![I]\!]_{\mathrm{CWA}\ /\ \mathrm{OWA}\ /\ \mathrm{WCWA}}$

Computing certain answers

Need to use the available (incomplete) data

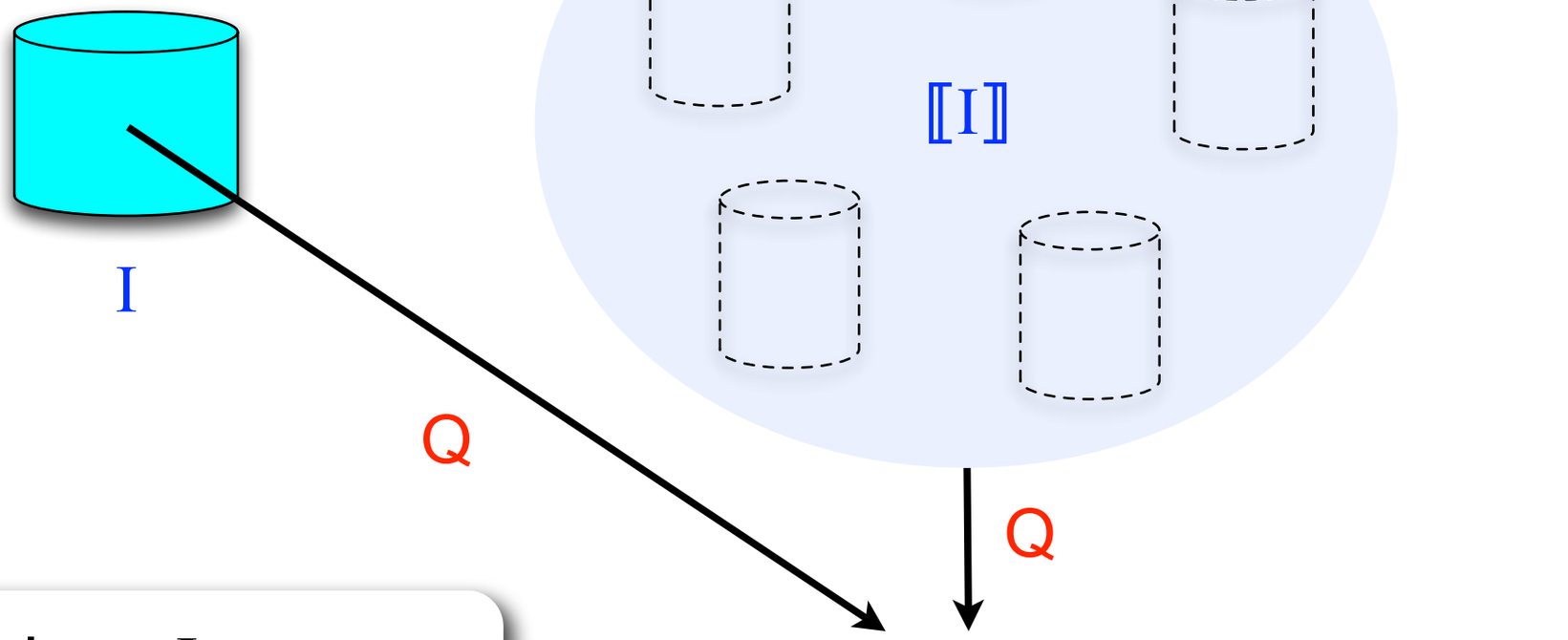


Computing certain answers on I : usually hard

- from coNP-complete to undecidable for FO [Imielinski-Lipski '84, Abiteboul et al '91]

Naïve evaluation

Naïve evaluation works for Q :



Q(I) = $cert_Q(I)$ for all I

Q(I) : Q evaluated directly on I,

as if variables were new distinct constants

($\mathbf{x_i} \neq \mathbf{x_j}$ for $i \neq j$

$\mathbf{x_i} \neq c$ for all $c \in Const$)

Naïve evaluation

Example Q : "There is a manager who has a manager"

$\exists\, x, y, z\ (\ \mathrm{Manager}(x, y) \wedge\ \mathrm{Manager}(z, x)\ )$

Employee

Green
x_1
Brown

Manager

Green	x_1
x_1	Brown
Green	x_2

I

$[\![I]\!]_{CWA\ /\ OWA\ /\ WCWA}$

Green
White
Brown

Green	White
White	Brown
Green	Black

⋮

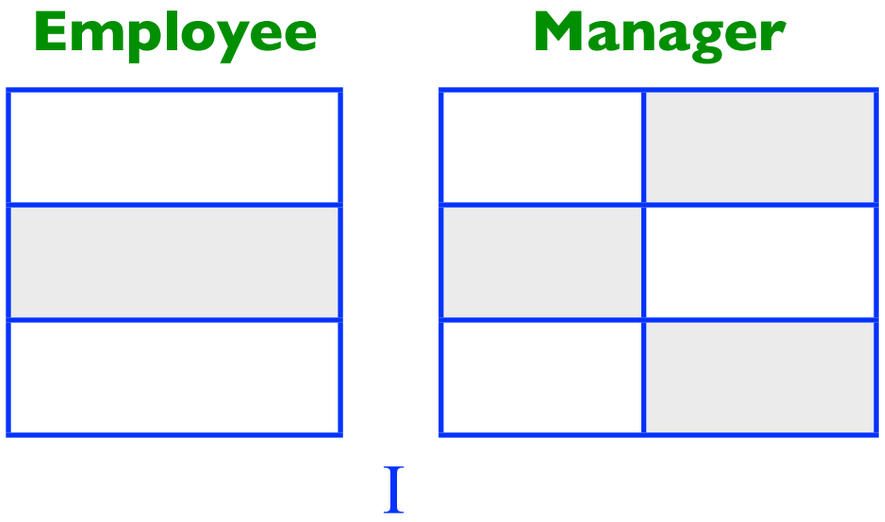
Q(I) = true

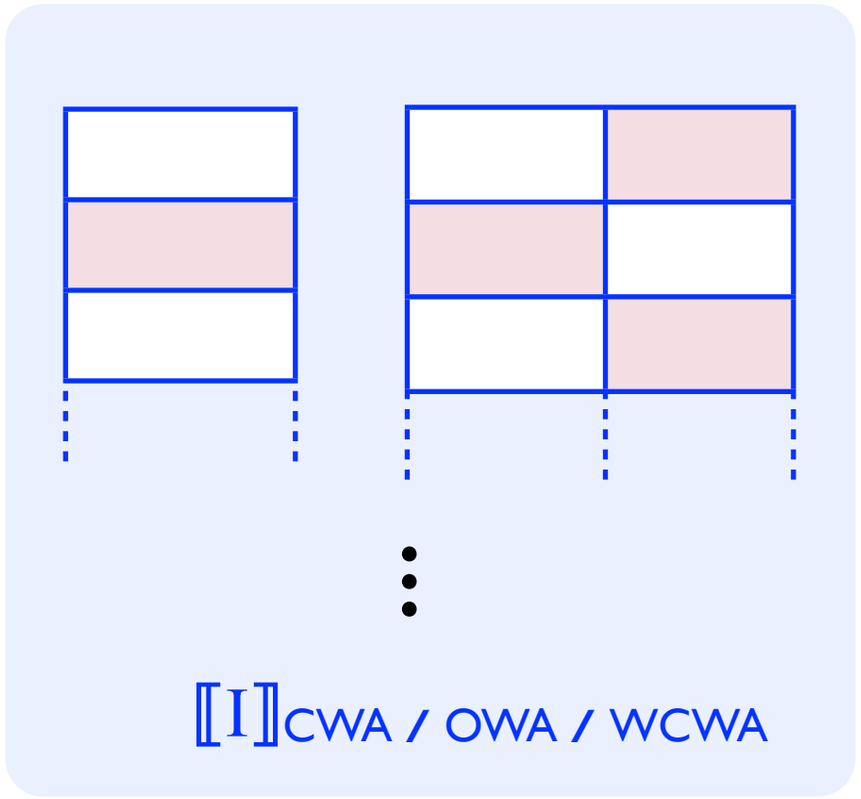
$\mathrm{cert}_Q(I)$ = *true* under CWA, OWA and WCWA

Naïve evaluation

Example Q : "There is a manager who has a manager"

$\exists\, x, y, z\ (\ \mathrm{Manager}(x, y) \wedge\ \mathrm{Manager}(z, x)\ )$





Generalizing:

$Q(I) = \mathrm{cert}_Q(I)$ for all I

⇒ naïve evaluation works for Q

under CWA, OWA and WCWA

Naïve evaluation in theory and practice

Certain answers: an entailment problem (checking that I entails Q) HARD

Naïve evaluation: a model-checking problem (checking $I \models Q$) EFFICIENT

- PTIME in the size of the instance for FO queries
- based on classical query evaluation algorithms of database engines
- can benefit from query optimization techniques

Naïve evaluation *works*

correct query answering semantics, classical query evaluation algorithms /

entailment reduces to (straightforward) model-checking

clearly not always possible ! (undecidable vs. PTIME)

Naïve evaluation does not always work

A concrete example

Q: "All employees are managers"

$\forall x(\ \text{Employee}(x) \rightarrow \exists y\ \text{Manager}(x, y)\ )$

Employee

Green
x_1
Brown

Manager

Green	x_1
x_1	Brown
Brown	x_2

I

$Q(I)$ = *true*

Green
White
Brown
Black

Green	White
White	Brown
Brown	Black

D

$[\![I]\!]_{OWA/WCWA}$

$cert_Q(I)$ = *false* under OWA and WCWA

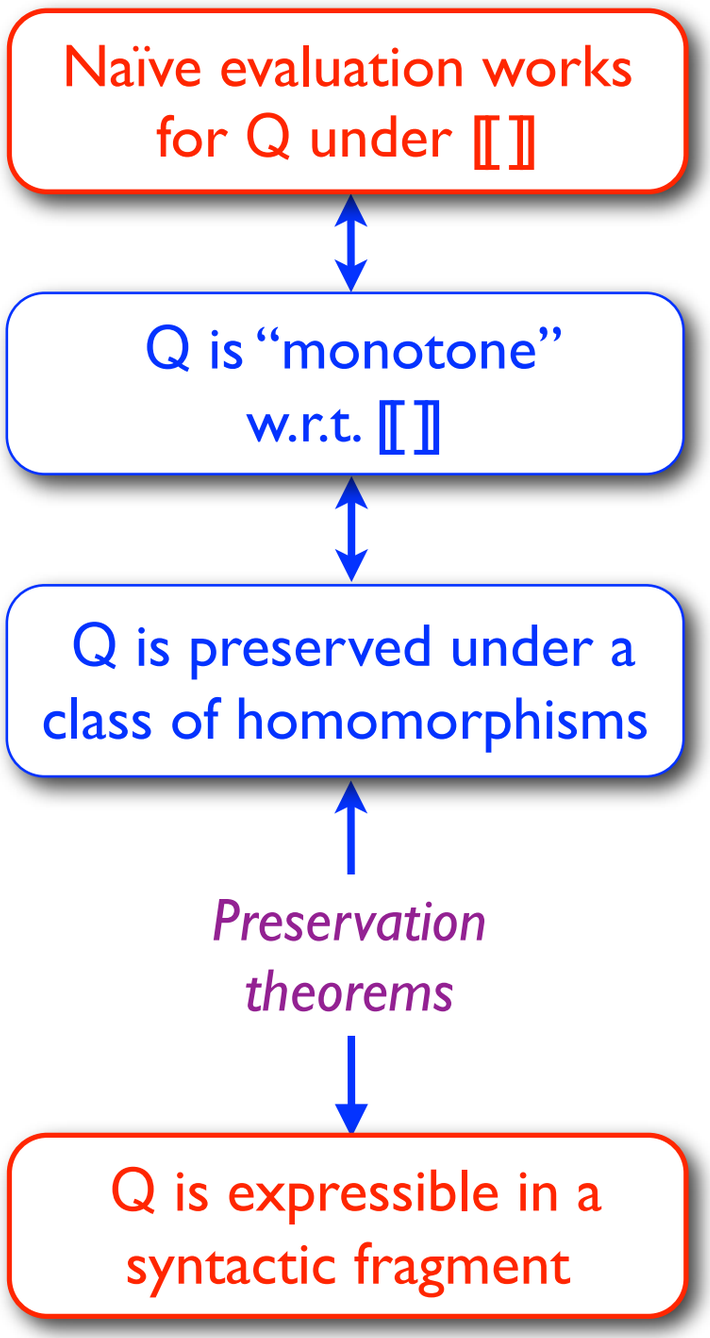
⇒ naïve evaluation does not work for Q under OWA and WCWA

Plan

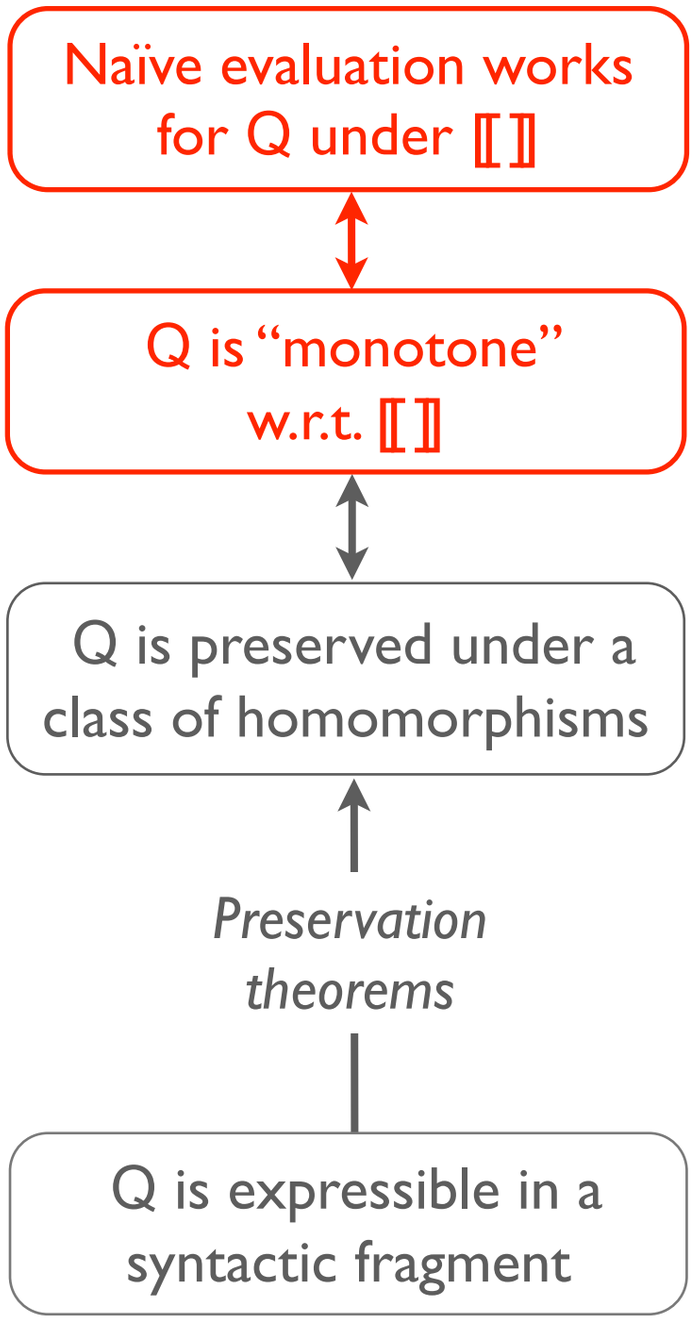
- Introduction
- Incomplete data model
- Querying incomplete data
  - the key to tractability: naïve evaluation
- What makes naïve evaluation work?
  - a general framework
- Applicability of the framework
- Moving forward

Relating naïve evaluation and syntactic fragments

A unified framework for relating naïve evaluation and syntactic fragments for several possible semantics:



Monotonicity and preservation



Shown in a very general setting subsuming every data model / semantics of incompleteness

(even beyond relational databases)

Naïve evaluation and monotonicity

Database domain: a quadruple $\langle \mathcal{D}, C, [\![\ ]\!], \approx \rangle$

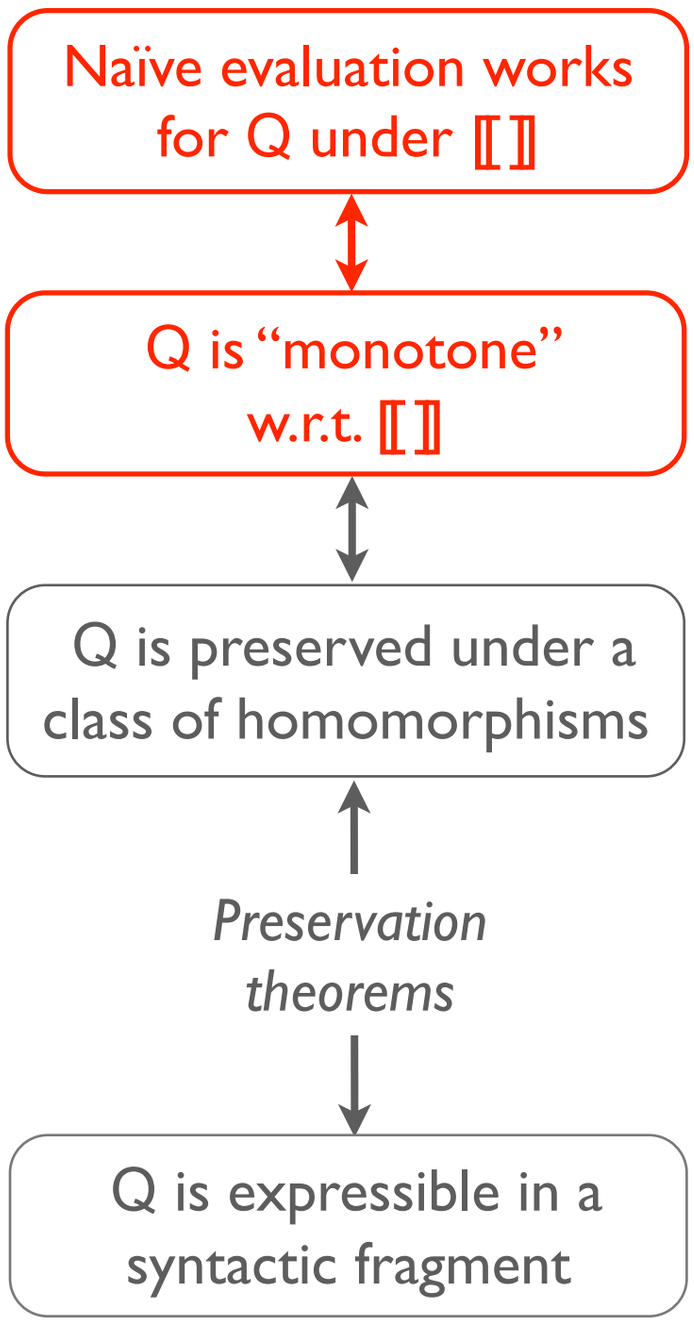
	description	example
$\mathcal{D}$: a set	database objects (complete and incomplete)	all naïve tables over a fixed schema σ
C : a subset of $\mathcal{D}$	complete database objects	all complete instances over σ
$[\![\ ]\!] : \mathcal{D} \to 2^C$	semantics of incompleteness	$[\![\ ]\!]_{OWA}$, $[\![\ ]\!]_{CWA}$, etc.
$\approx$: an equivalence relation on $\mathcal{D}$	equivalence of objects (w.r.t. queries)	isomorphism of relational instances

Boolean query: $Q : \mathcal{D} \to \{true, false\}$

generic : $x \approx y$ implies $Q(x)=Q(y)$

monotone w.r.t. $[\![\ ]\!]$: $y \in [\![x]\!]$ implies $Q(x) \Rightarrow Q(y)$

Naïve evaluation and monotonicity



Over a saturated database domain, if Q is a generic Boolean query:

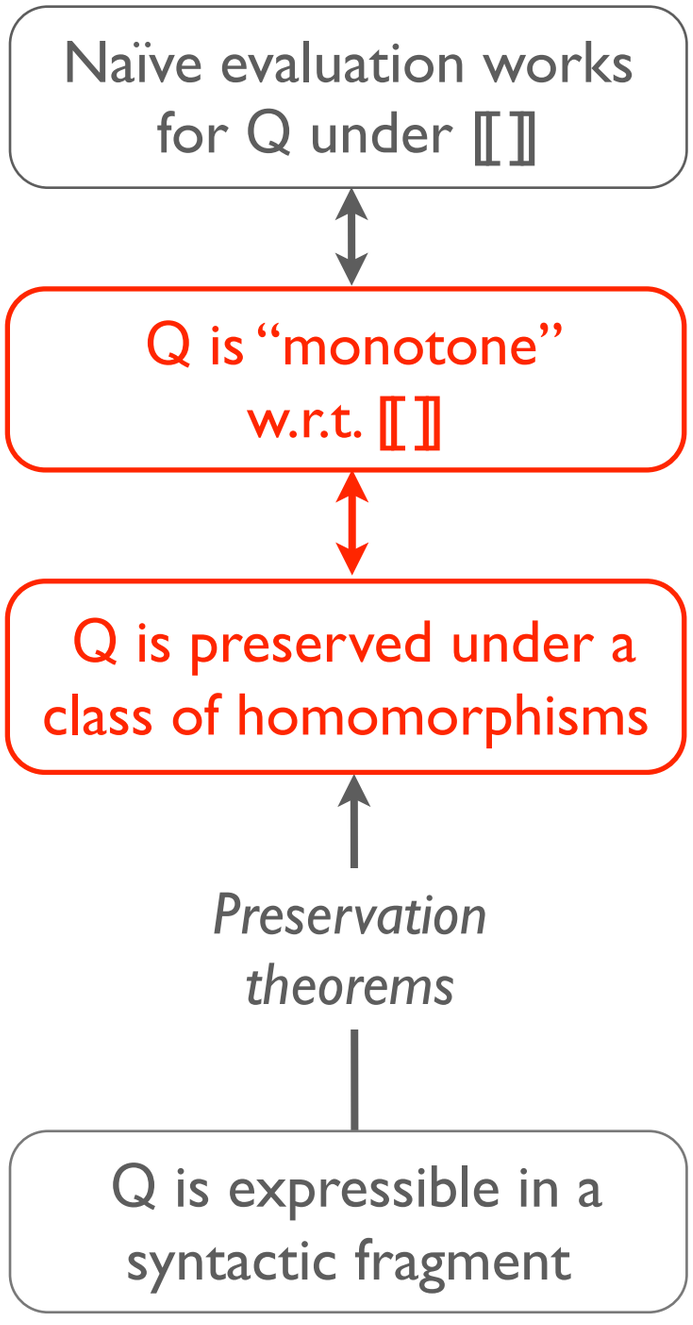
Naïve evaluation works for Q iff Q is monotone w.r.t. ⟦ ⟧

Saturation property for $\langle \mathcal{D}, C, [\![\ ]\!], \approx \rangle$:

For all $x \in \mathcal{D}$ there exists $y \in [\![x]\!]$ $y \approx x$

holds for most common semantics

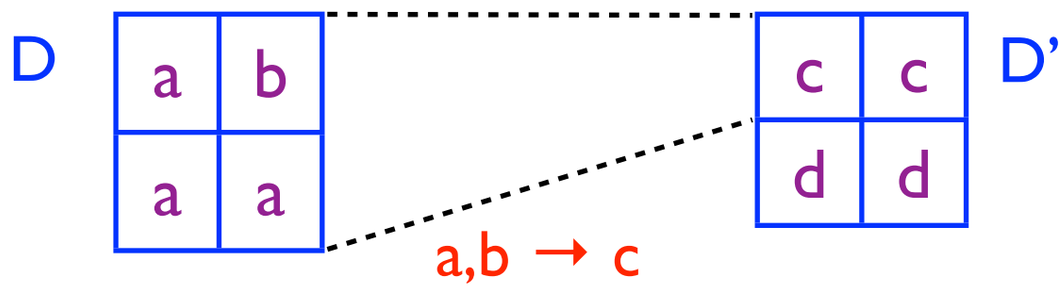
Monotonicity and preservation



- monotonicity = "preservation" under the semantics
- relational semantics: usually homomorphism-based

Monotonicity and preservation

homomorphism $D \to D'$: a mapping $h: dom(D) \to dom(D')$ s.t. $h(D) \subseteq D'$



Q preserved under homomorphism:

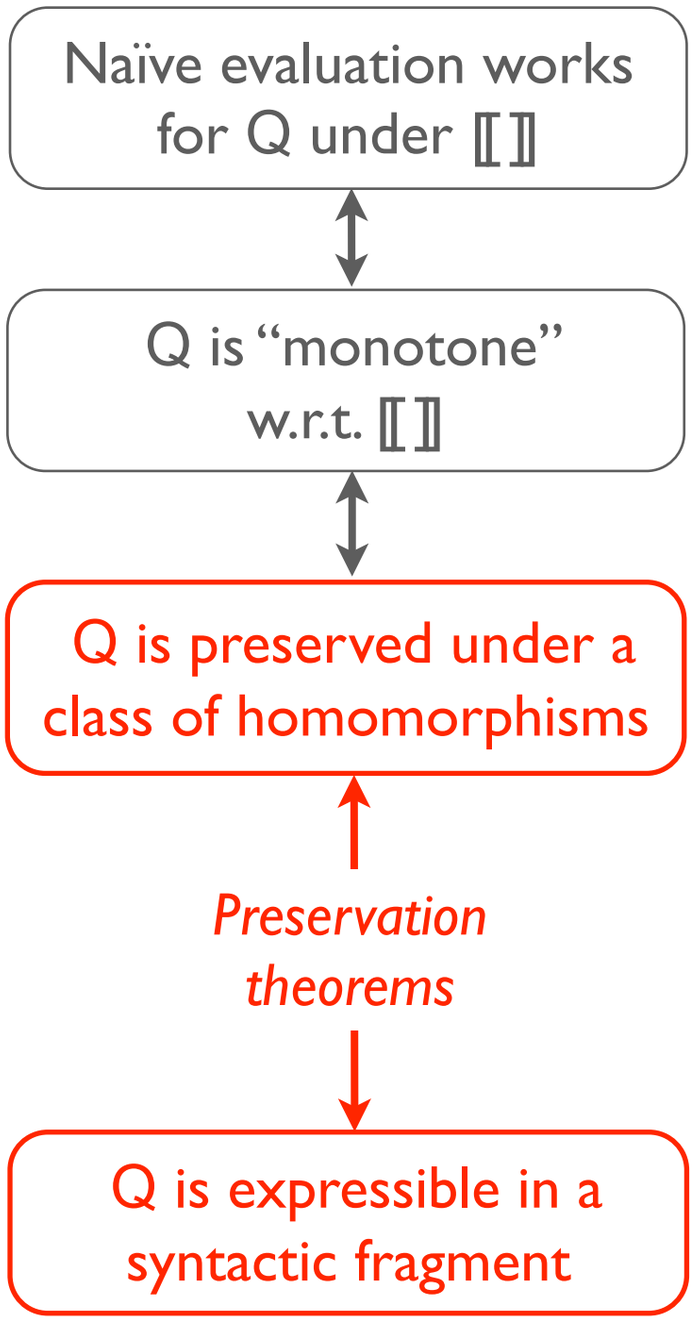
$D \to D'$ implies $Q(D) \Rightarrow Q(D')$ for all D

Many variants: onto homomorphism, strong onto homomorphism, ...

monotonicity w.r.t different semantics ↔

preservation under different notions of homomorphism

Preservation and syntactic fragments



Preservation theorems

- syntactic characterizations of preservation properties of queries in a given logic
- classical results in (finite) model theory

Preservation and syntactic fragments

Homomorphism Preservation Theorem: over arbitrary structures
an FO query Q is preserved under homomorphism iff Q is in ∃Pos

- recently proved over finite structures [Rossman '08]

∃Pos : $\{\exists, \wedge, \vee\}$-FO (Unions of Conjunctive Queries)

Preservation and syntactic fragments

Homomorphism Preservation Theorem: over arbitrary structures
an FO query Q is preserved under homomorphism iff Q is in ∃Pos

- recently proved over finite structures [Rossman '08]

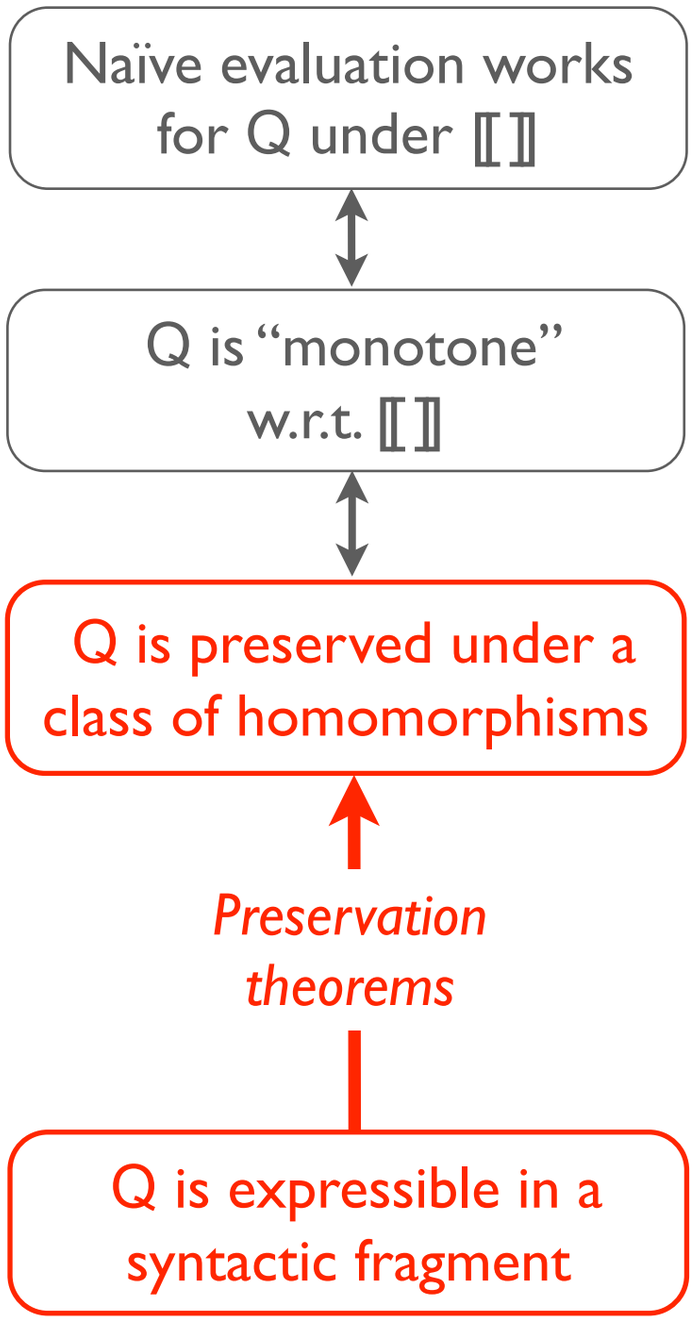
∃Pos : {∃, ∧, ∨}-FO (Unions of Conjunctive Queries)

Lyndon Positivity Theorem [Lyndon '59]: over arbitrary structures
an FO query Q is preserved under onto homomorphism iff Q is in Pos

- fails in the finite [Ajtai-Gurevich 87, Rosen '95, Stolboushkin '95]

Pos: {∃, ∀,∧, ∨}-FO

Preservation and syntactic fragments



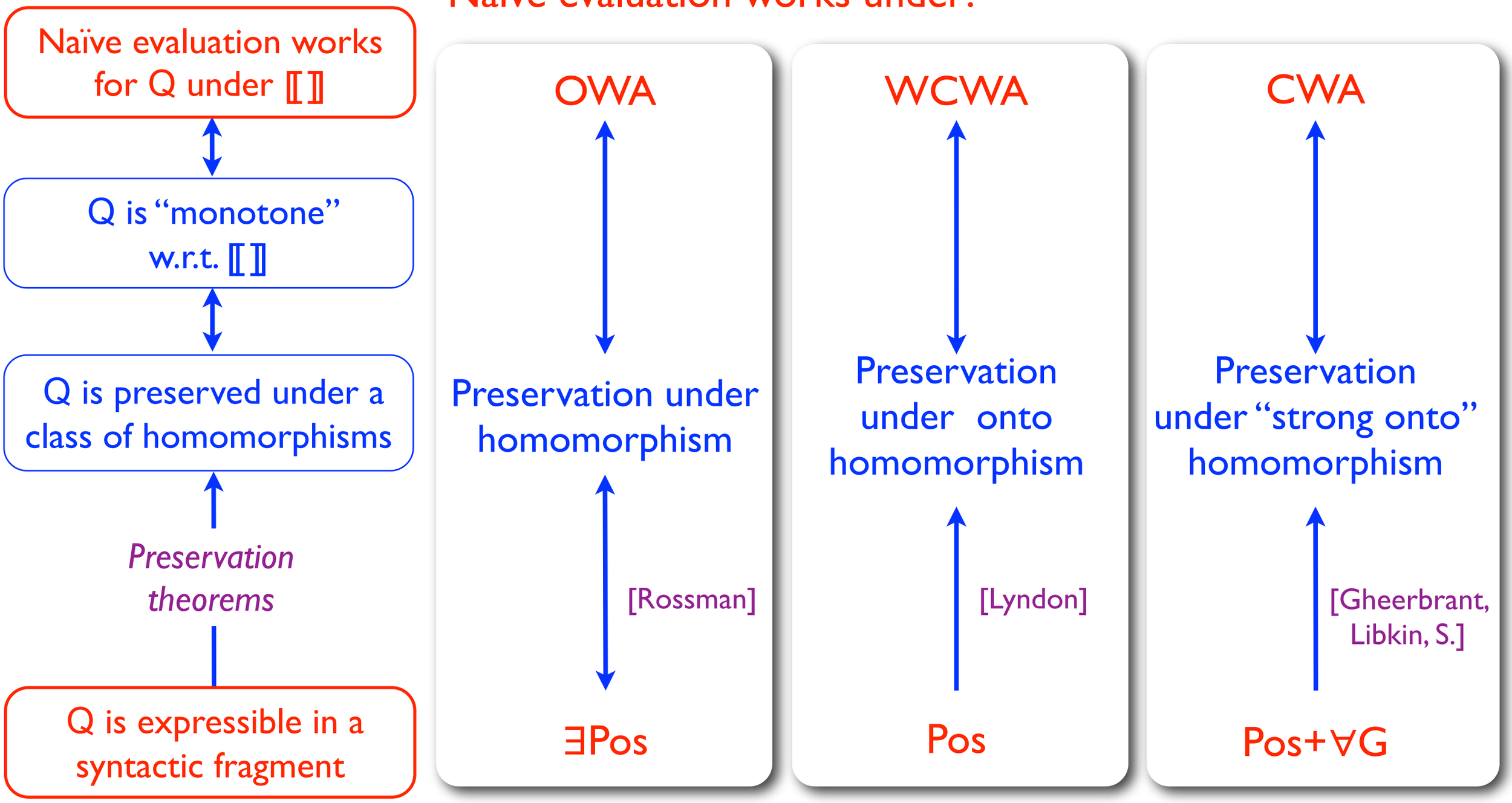
Preservation theorems:

(Syntax ⇒ Preservation) holds in the finite as well

⇒ classes of queries where naïve evaluation works

Naïve evaluation and syntactic fragments

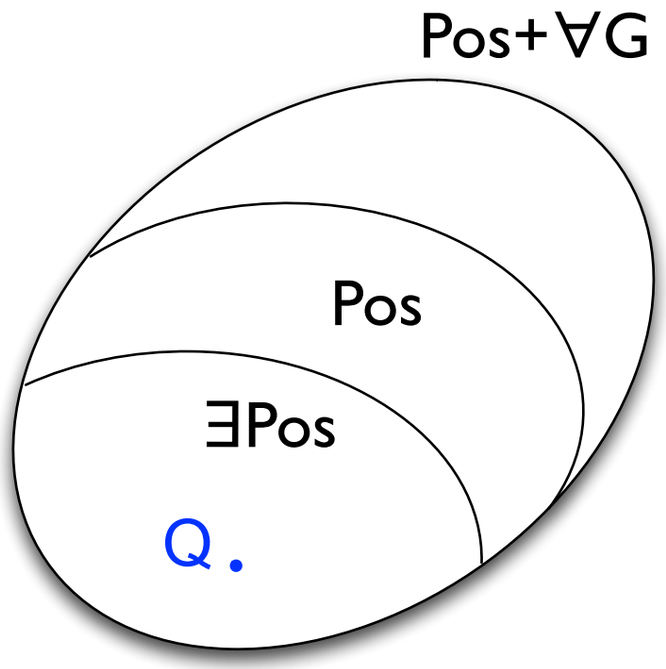
Three well known semantics as instances of our framework



Examples revisited

Q : "There is a manager who has a manager"

$\exists\, x, y, z\ (\ \mathrm{Manager}(x, y) \wedge\ \mathrm{Manager}(z, x)\ )$

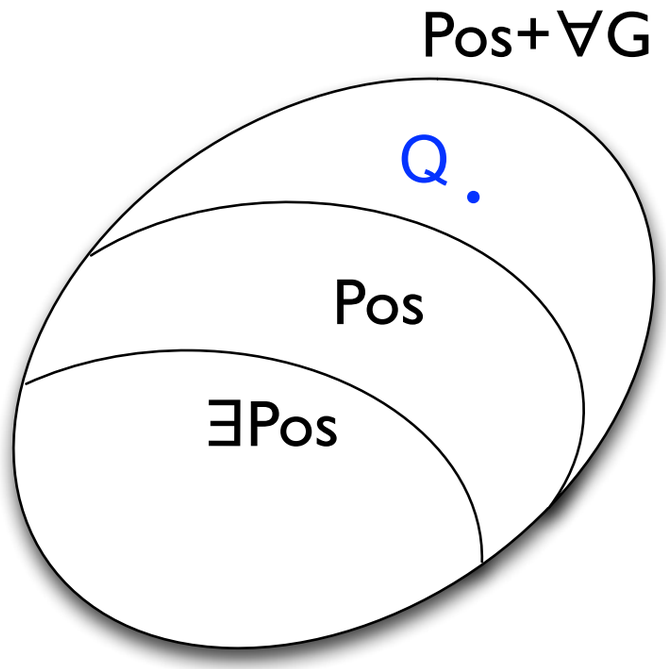


⇒ naïve evaluation works for Q under CWA OWA, WCWA

Examples revisited

Q : "All employees are managers"

$$\forall x(\ \mathrm{Employee}(x) \rightarrow \exists y\ \mathrm{Manager}(x, y)\ )$$



⇒ naïve evaluation works for Q under CWA

(recall: not true under OWA, nor under WCWA)

Pos+∀G extends Pos with a limited form of negation (universal guards)

- a very natural fragment

Naïve evaluation works well beyond ∃Pos under other semantics than OWA

Plan

- Introduction
- Incomplete data model
- Querying incomplete data
  - the key to tractability: naïve evaluation
- What makes naïve evaluation work?
  - a general framework
- Applicability of the framework
- Moving forward

Beyond OWA, CWA and WCWA

Semantics of incompleteness have been considered in several contexts:

- programming semantics, logic programming, data exchange,...

  [Minker'82, Ohori'90, Rounds'91, Libkin'95, Hernich'11]

  ➡ powerset semantics

  ➡ minimal semantics

Beyond OWA, CWA and WCWA

Naïve evaluation works for Q under ⟦ ⟧

↕

Q is "monotone" w.r.t. ⟦ ⟧

↕

Q is preserved under a class of homomorphisms

↑

Preservation theorems

|

Q is expressible in a syntactic fragment

Naïve evaluation works under:

Powerset semantics

↕

Preservation under unions of strong onto homomorphisms

↑

$\exists Pos + \forall G^{bool}$

Minimal semantics

↕ over cores

Preservation under unions of minimal homomorphisms

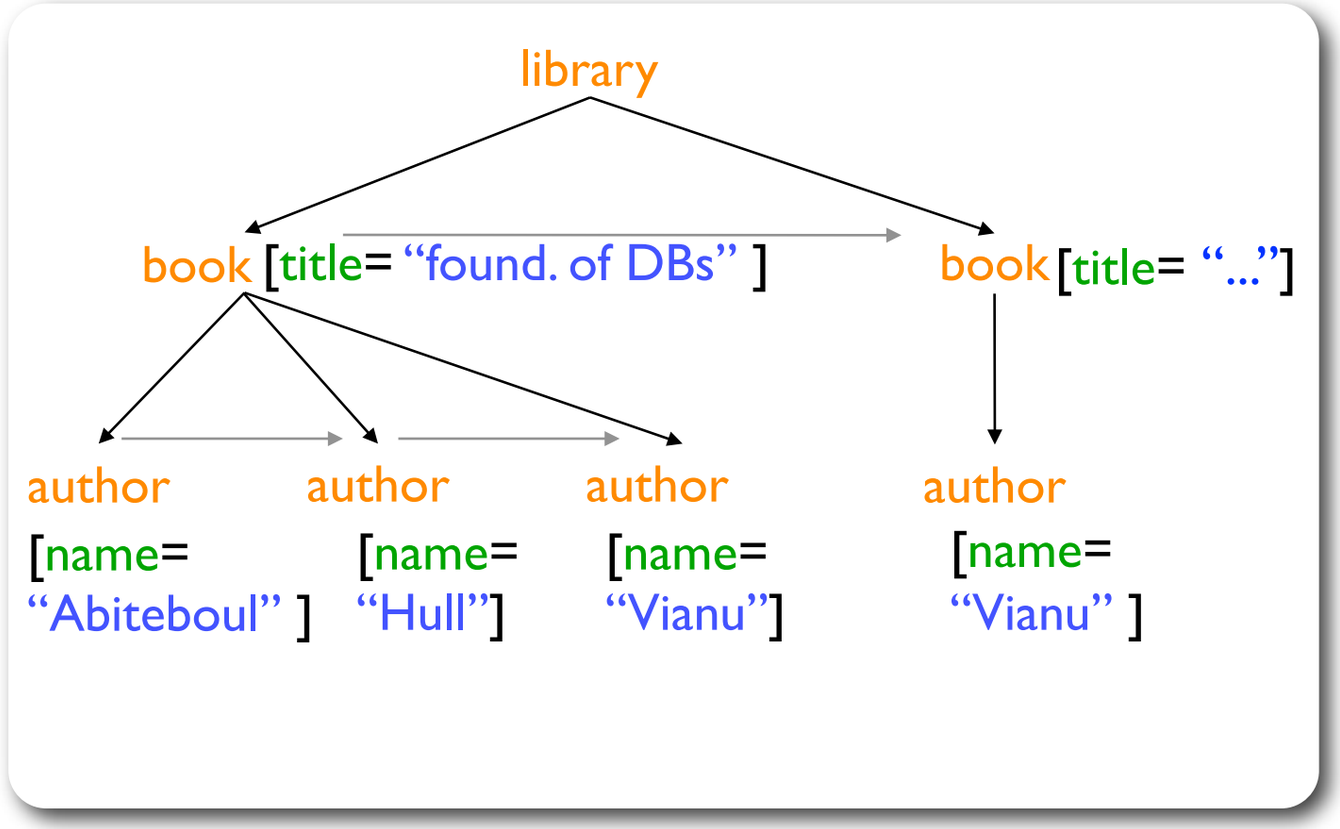
↑

$\exists Pos + \forall G^{bool}$

Beyond the relational data model

XML: hierarchically structured data

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
    <book title= “found. of DBs”>
        <author name= “Abiteboul” > </author>
        <author name= “Hull”>    </author>
        <author name= “Vianu”>  </author>
    </book>
    <book title= “...”>
        <author name= “Vianu” > </author>
        ....
    </book>
</library>
```

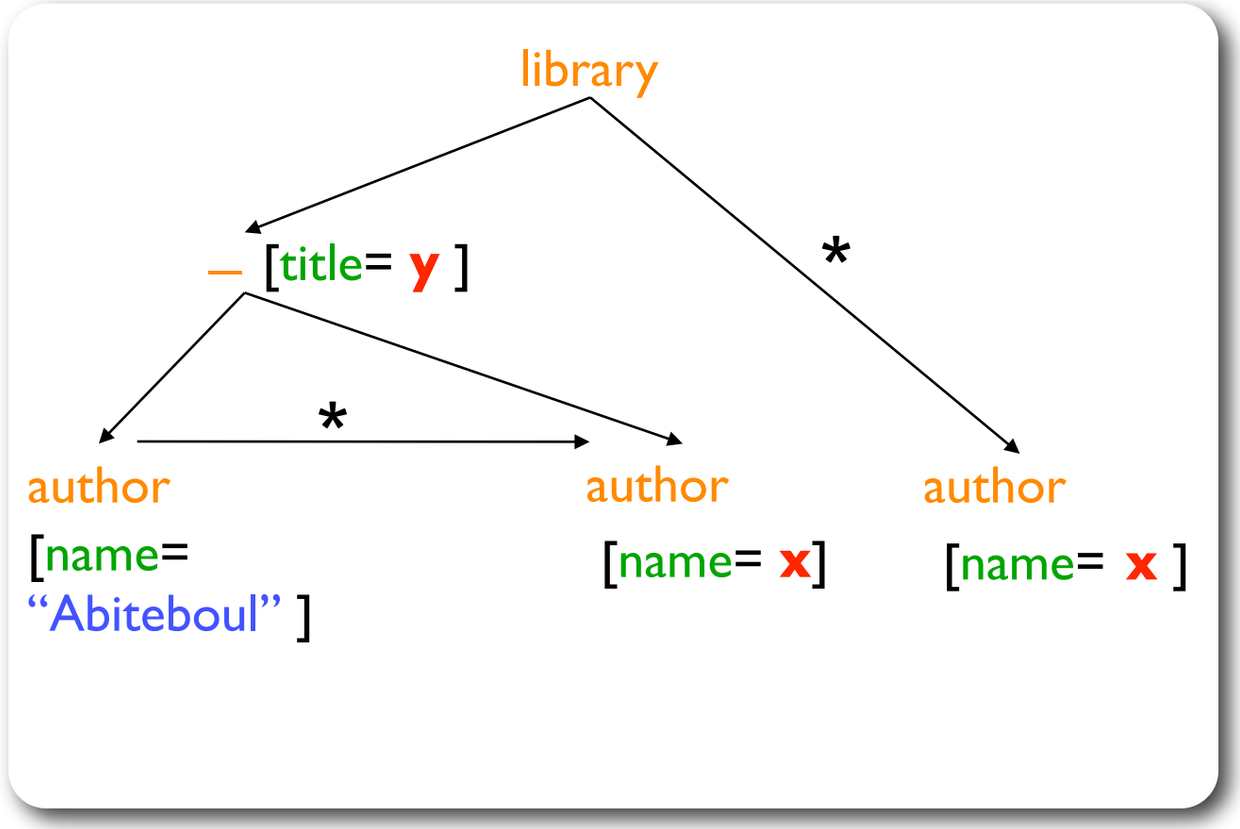


modeled as trees with data values associated to nodes

Beyond the relational data model

Incomplete XML based on a form of "tree patterns" [Barcelo-Libkin-Poggi-S.'10]

- missing data values
- missing nodes
- missing structural information
  - ✓ labels
  - ✓ parent-child, next-sibling relationships
  - ✓ etc.



Tree-pattern-queries: the analog of ∃Pos on trees

Beyond the relational data model

The analog of naïve evaluation works for tree-pattern-queries under OWA on *rigid* tree patterns [Barcelo-Libkin-Poggi-S.'10]

- rigidity: essentially avoids structural incompleteness

Our framework explains this result:

- database domain:
  - ✓ the set of complete/incomplete trees,
  - ✓ OWA semantics: homomorphism-based
- tree pattern queries are preserved under homomorphisms of trees
- rigidity ensures the saturation property

Moving forward

Naïve evaluation on combinations of data models/semantics, e.g

- ➡ XML/CWA
- ➡ graph-structured data

Query languages beyond FO

- ➡ fixed-point logics, fragments of SO, etc.

Naïve evaluation over restricted instances

- ➡ Applications: data integration/exchange

Beyond naïve-evaluation

- ➡ rewriting of the query/instance
  (classical in ontology-based query answering)

Thank you!

Real life paradoxes

- SQL adopts a three-valued logic
  - essentially any comparison involving null values evaluates to *unknown*
- An SQL condition checking $X - Y \neq \varnothing$

```
EXISTS ( SELECT X.A FROM  X
WHERE X.A  NOT IN  ( SELECT  Y.A  FROM  Y ) )
```

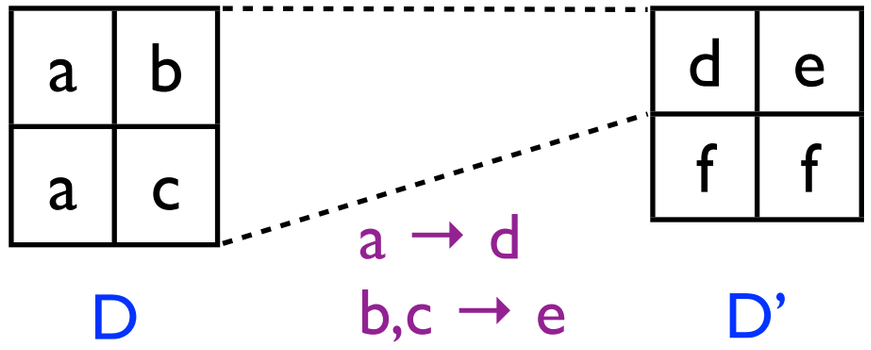
- $X.A = \{1, 2, 3, \ldots, N\}$ and $Y.A = \{null\}$, then $X - Y = \varnothing$ no matter what N is!
- That's how SQL programs work: this is part of the SQL 1999 ANSI Standard

Homomorphisms

Homomorphism $D \rightarrow D'$:

a mapping $h: dom(D) \rightarrow dom(D')$ s.t.

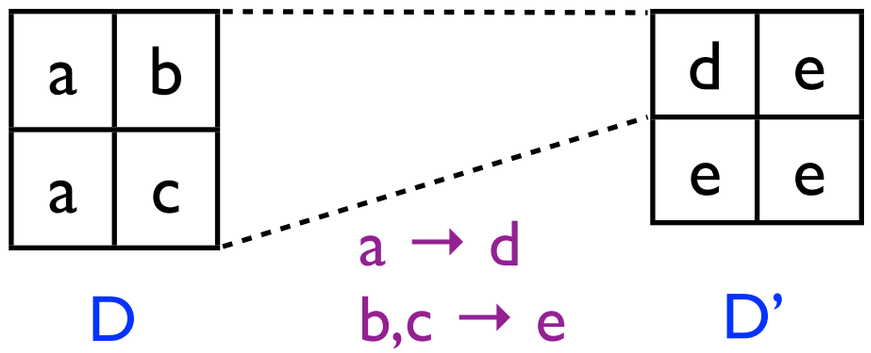
$h(D) \subseteq D'$



Onto homomorphism $D \rightarrow D'$:

a homomorphism $h: D \rightarrow D'$ s.t.

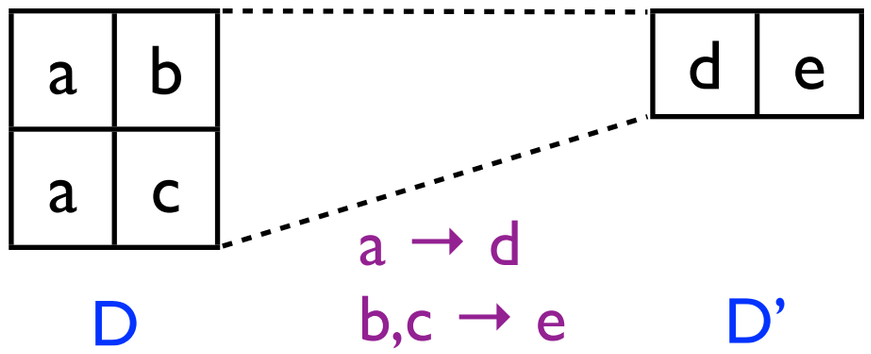
$h(dom(D)) = dom(D')$



Strong onto homomorphism $D \rightarrow D'$:

a homomorphism $h: D \rightarrow D'$ s.t.

$h(D) = D'$



Homomorphisms

- Union of strong onto homomorphisms $D \rightarrow D'$: $\cup_i h_i(D) = D'$

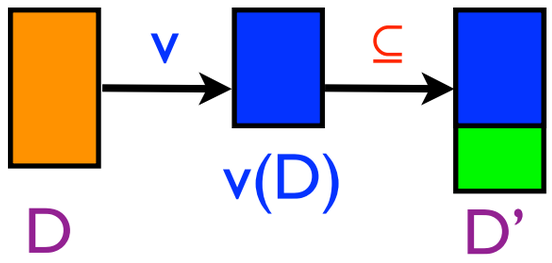
- D-minimal homomorphism h on D :
  there exists no h', preserving all constants preserved by h, s.t. $h'(D) \subsetneq h(D)$

- Union of minimal homomorphisms $D \rightarrow D'$: $\cup_i h_i(D) = D'$

  with $h_1 ... h_n$ D-minimal and preserving the same constants

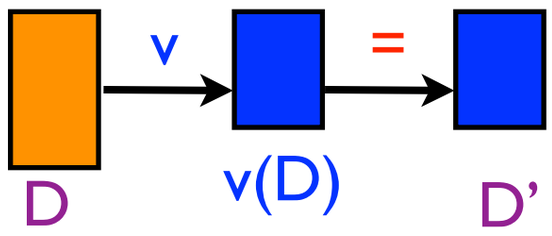
Homomorphism-based relational semantics

OWA



$D' \in [\![D]\!]_{OWA}$ iff $\exists v \;\; v(D) \subseteq D'$

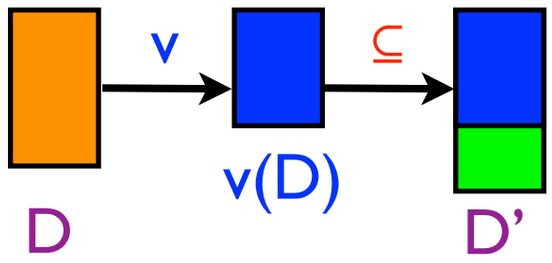
CWA



$D' \in [\![D]\!]_{CWA}$ iff $\exists v \;\; v(D) = D'$

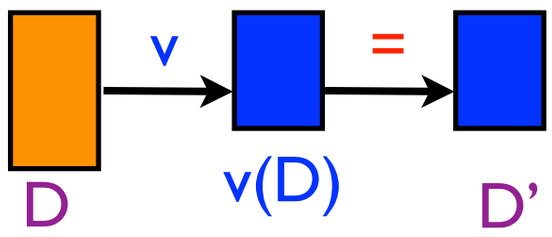
Homomorphism-based relational semantics

OWA



$D' \in [\![D]\!]_{OWA}$ iff $\exists v \;\; v(D) \subseteq D'$

CWA

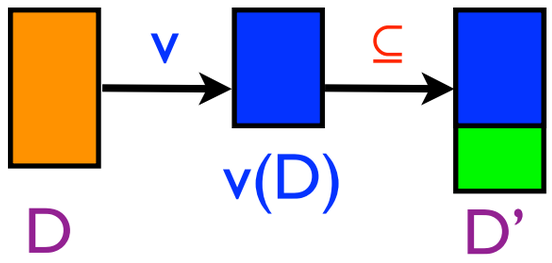


$D' \in [\![D]\!]_{CWA}$ iff $\exists v \;\; v(D) = D'$

- Essentially based on two steps: 1) valuation of nulls 2) extension of the instance
- Other well-known semantics follow the same paradigm:

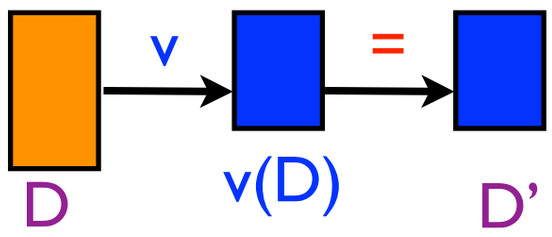
Homomorphism-based relational semantics

OWA



$D' \in [\![D]\!]_{OWA}$ iff $\exists v \;\; v(D) \subseteq D'$

CWA

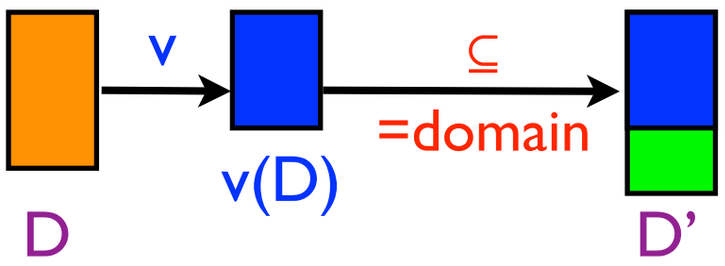


$D' \in [\![D]\!]_{CWA}$ iff $\exists v \;\; v(D) = D'$

- Essentially based on two steps: 1) valuation of nulls 2) extension of the instance
- Other well-known semantics follow the same paradigm:

Weak Closed World Assumption [Reiter 77]

WCWA

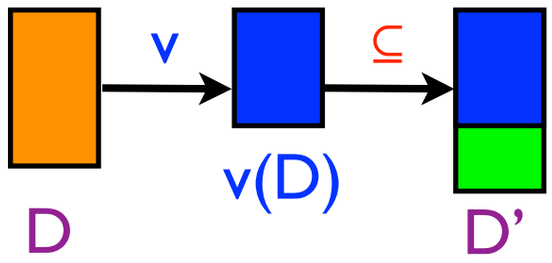


$D' \in [\![D]\!]_{WCWA}$ iff $\exists v \;\; v(D) \subseteq D' \wedge$

$dom(D') = dom(v(D))$

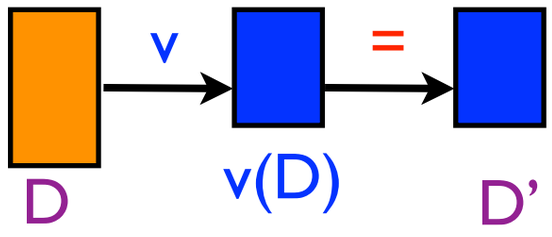
Homomorphism-based relational semantics

OWA



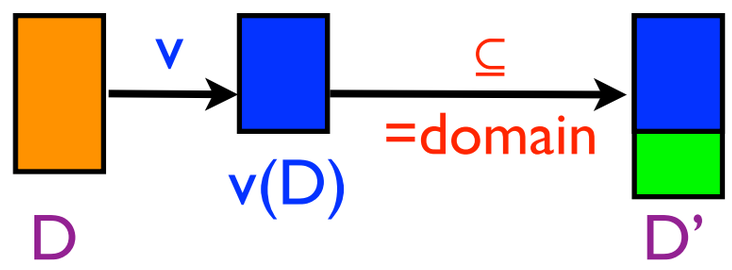
$D' \in [\![D]\!]_{OWA}$ iff $\exists v \;\; v(D) \subseteq D'$

CWA



$D' \in [\![D]\!]_{CWA}$ iff $\exists v \;\; v(D) = D'$

WCWA

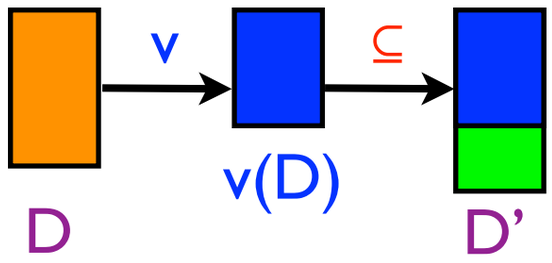


$D' \in [\![D]\!]_{WCWA}$ iff $\exists v \;\; v(D) \subseteq D' \wedge$ $dom(D') = dom(v(D))$

Can be generalized to arbitrary semantic relations...

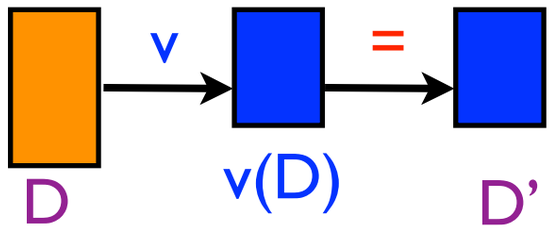
Homomorphism-based relational semantics

OWA



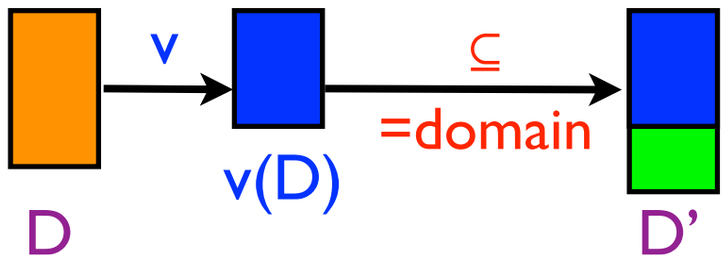
$D' \in [\![D]\!]_{OWA}$ iff $\exists v \;\; v(D) \subseteq D'$

CWA



$D' \in [\![D]\!]_{CWA}$ iff $\exists v \;\; v(D) = D'$

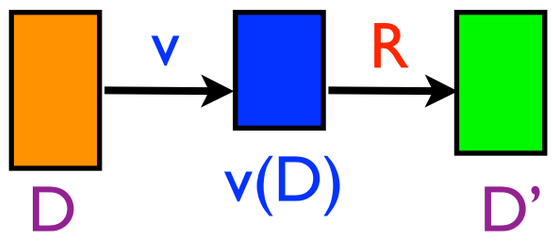
WCWA



$D' \in [\![D]\!]_{WCWA}$ iff $\exists v \;\; v(D) \subseteq D' \wedge$

$dom(D') = dom(v(D))$

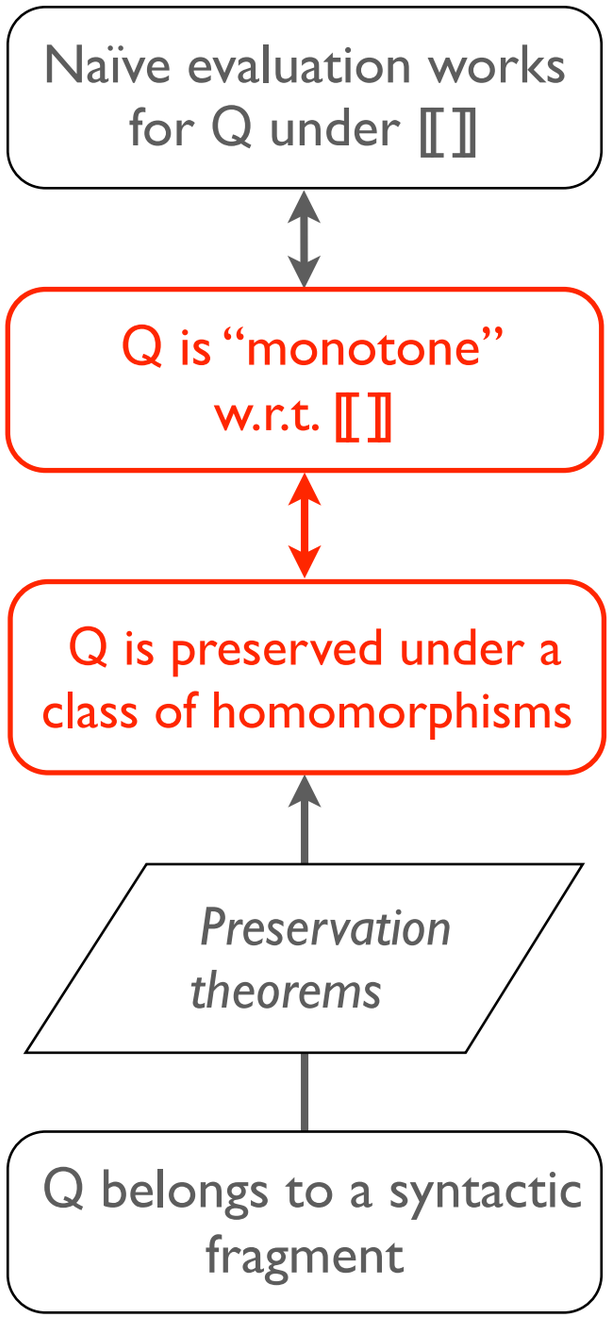
R-based



$D' \in [\![D]\!]_{R}$ iff $\exists v \;\; v(D)\ R\ D'$

R: reflexive binary relation between complete instances

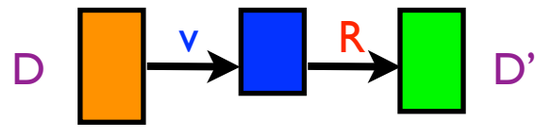
Monotonicity and preservation



- R-homomorphism $D \to D'$ (D and D' complete):

  a mapping h over dom(D) s.t. h(D) R D'

- R-homomorphisms "mimic" the semantic mapping:

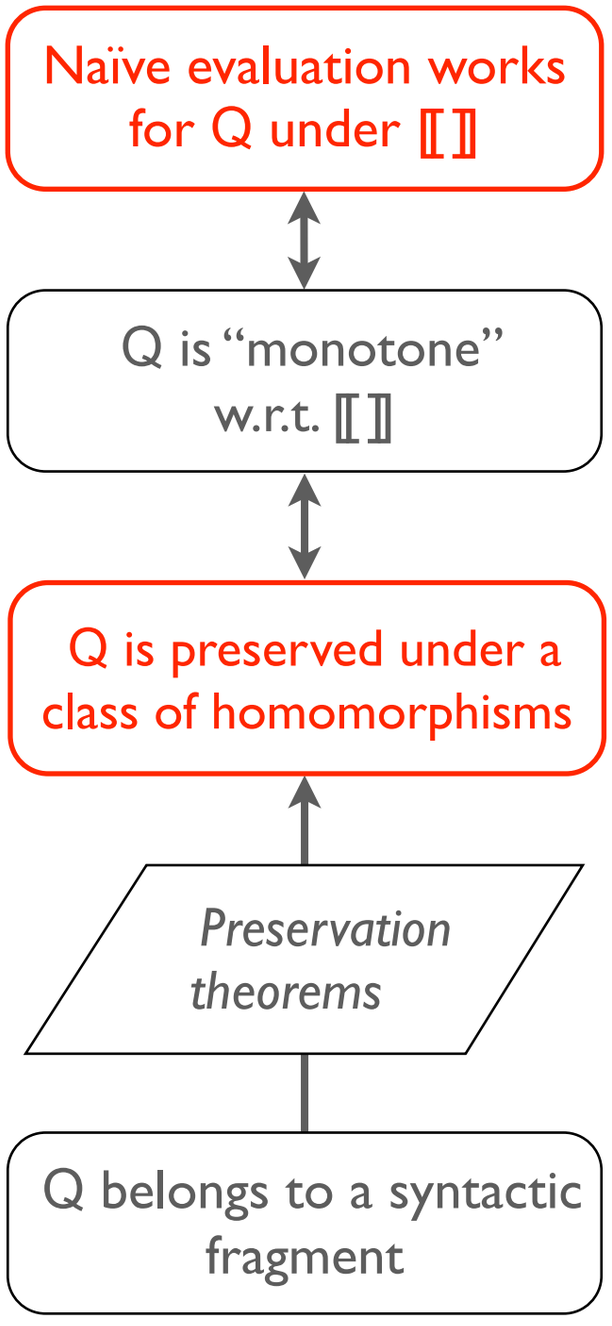


- except that valuations distinguish constants from nulls

- However, using query genericity:

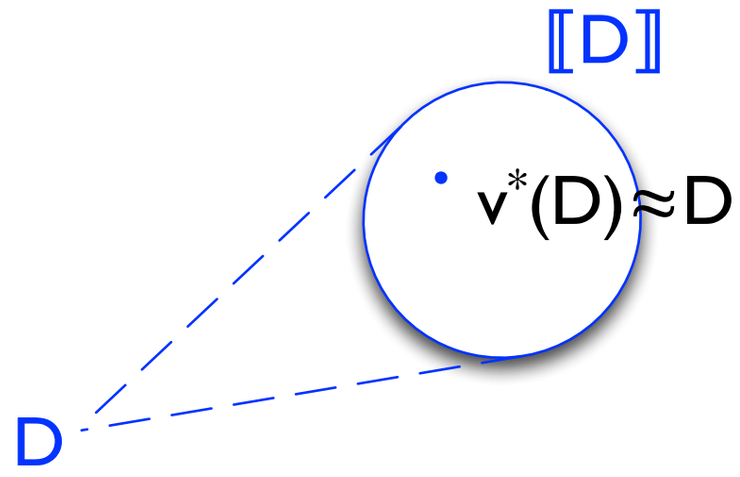
> If a relational semantics [[ ]] is given by R
> and Q is a generic Boolean query
>
> Q is monotone w.r.t. [[ ]] iff
> Q is preserved under R-homomorphisms

Naïve evaluation and preservation



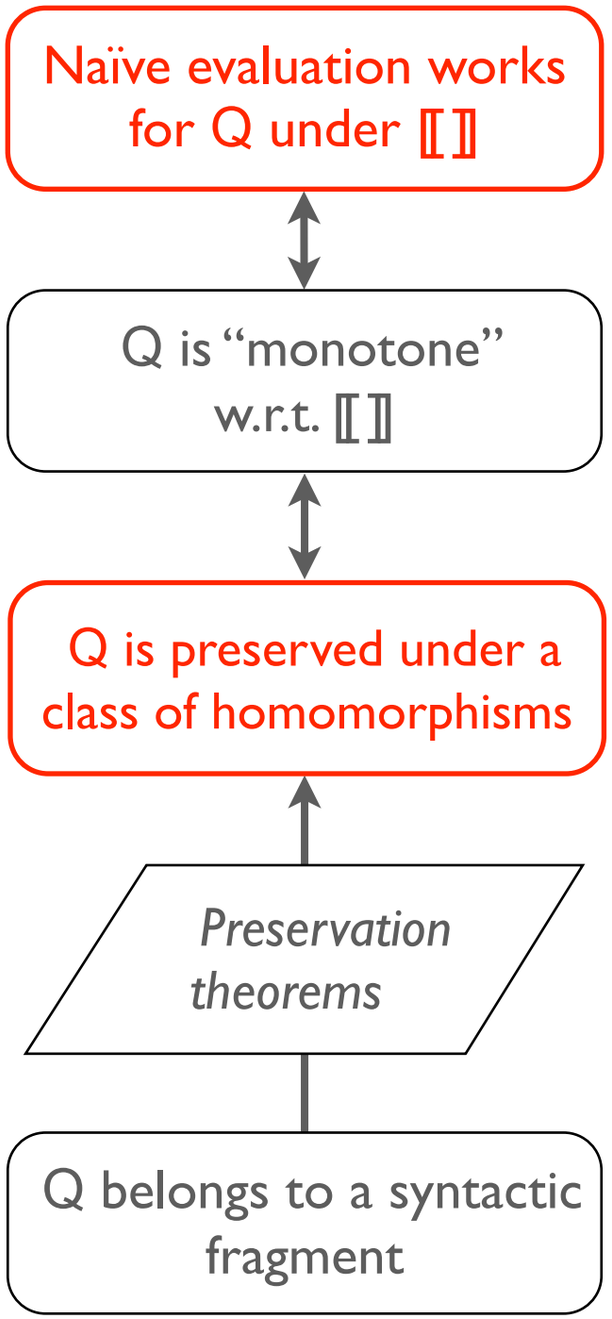
Combining the two steps:

If a relational semantics is given by R, the saturation property holds:



v^*: distinct nulls to distinct constants not occurring in D

Naïve evaluation and preservation



Theorem

If a relational semantics is given by R
and Q is a generic Boolean query

Naïve evaluation works for Q iff
Q is preserved under R-homomorphisms

	R	R-homomorphism
OWA	$\subseteq$	homomorphisms
CWA	$=$	*strong onto* homomorphisms (i.e. homomorphisms D $\rightarrow$h(D))
WCWA	$\subseteq$ =domain	onto homomorphisms

Preservation and syntactic fragments of FO

- What about strong onto homomorphisms ?
  - There is a preservation result in the infinite [Keisler '65]
  - complex syntactic restrictions, one binary relation only, problematic to extend...
- A new sufficient condition for preservation, with a good syntax:

Positive fragment with Universal Guards (Pos+∀G)

$$\varphi := \top \mid \bot \mid R(\bar{x}) \mid x = y \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x \varphi \mid \forall x \varphi \mid$$

$$\left(\begin{array}{ll} \forall \bar{x} \, (\, G(\bar{x}) \rightarrow \varphi \,) & \text{with} \\ & G : \text{a relation or equality symbol} \\ & \bar{x} : \text{a tuple of distinct variables} \end{array} \right)$$

Pos+∀G formulas are preserved under strong onto homomorphisms

Semantics arising from orderings

- Information ordering of Codd tuples :

$t \sqsubseteq t'$ if $t[i]$ constant $\Rightarrow t'[i] = t[i]$ t' is "more informative" than t

- Lifting to sets of tuples [Hoare, Plotkin 70s]
  - $D \sqsubseteq^H D' : \forall\, t \in D \quad \exists\, t' \in D' \quad t \sqsubseteq t'$ (Hoare ordering, open-world)
  - $D \sqsubseteq^P D' :\ D \sqsubseteq^H D'$ and $\forall\, t' \in D' \quad \exists\, t \in D \quad t \sqsubseteq t'$ (Plotkin ordering)

Semantics arising from orderings

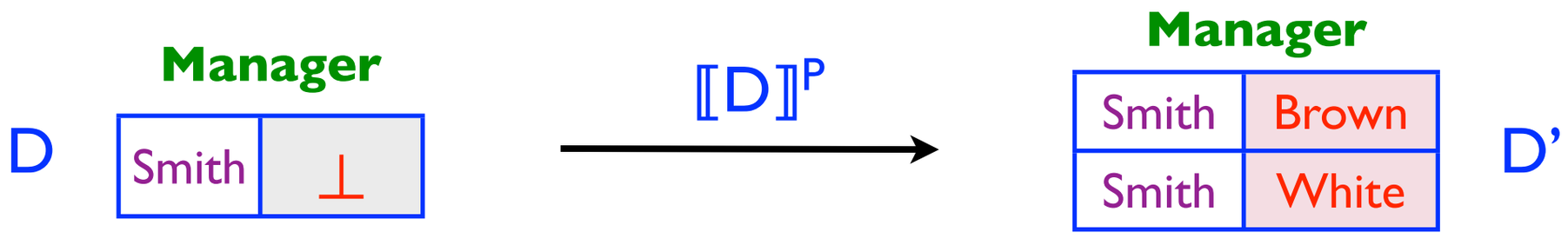
$D \sqsubseteq^H D' : \forall\, t \in D \quad \exists\, t' \in D' \quad t \sqsubseteq t'$ (Hoare ordering, open-world)

$D \sqsubseteq^P D' :\ D \sqsubseteq^H D'$ and $\forall\, t \in D' \quad \exists\, t \in D \quad t \sqsubseteq t'$ (Plotkin ordering)

- Orderings give rise to semantics of Codd databases:

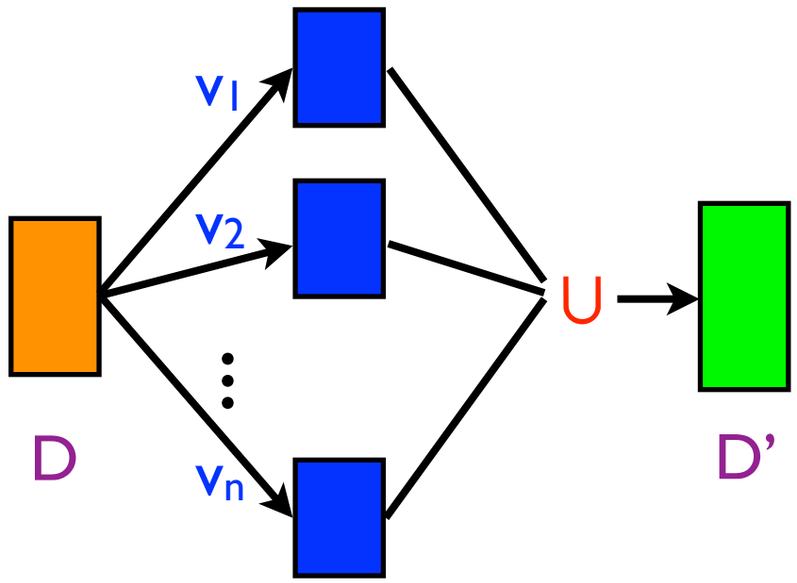
$$[\![D]\!] = \{ \text{complete } D' \mid D \sqsubseteq D' \}$$

- Observe that Plotkin semantics is more "open" than CWA :



Powerset semantics

- Extend (and generalize) Plotkin semantics to naïve databases



Powerset CWA

$D' \in (\!|D|\!)_{CWA}$ iff

$\exists$ valuations $v_1, \ldots v_n$ $D' = \cup_i v_i(D)$

- When restricting to Codd databases Powerset CWA coincides with Plotkin
- Gives rise to a whole new class of semantics: $\cup$ is replaced by any suitable relation

$$\mathbf{R} \subseteq 2^C \times C$$

Naïve evaluation for powerset semantics

- Naïve evaluation ↔ Monotonicity ↔ Preservation continues to hold
- Under the powerset CWA the needed notion is preservation under *unions of strong onto homomorphisms* (i.e. homomorphisms $D \to \bigcup_{i=1}^{n} h_i(D)$)
- We have similar results for powerset semantics based on arbitrary **R**
- An FO fragment preserved under unions of strong onto homomorphisms:

$\exists Pos+\forall G^{bool}$:

$\exists Pos$ extended with universal guarded *sentences* $\forall \bar{x}\ (\ G(\bar{x}) \to \varphi(\bar{x})\ )$

Corollary

If Q is a Boolean query from $\exists Pos+\forall G^{bool}$

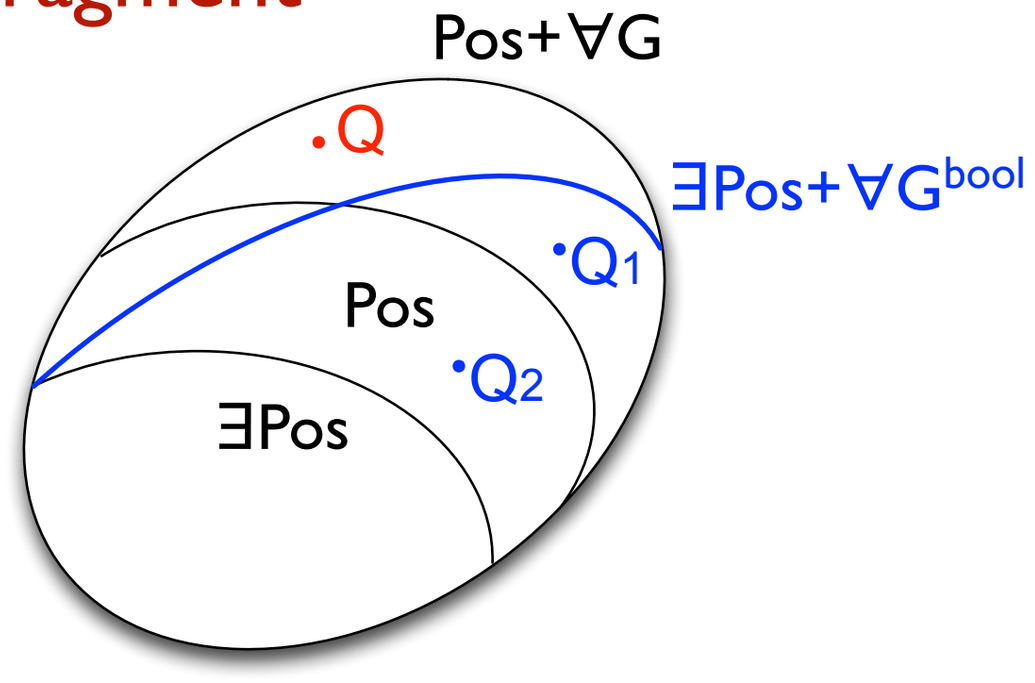
Naïve evaluation works for Q under $(\!|\cdot|\!)_{CWA}$

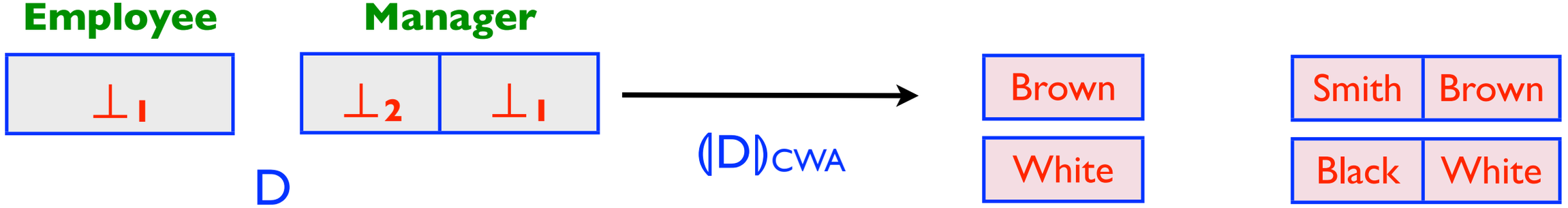
The $\exists$Pos+$\forall$G^{bool} fragment

$Q_1 = \forall x(\ \text{Employee}(x) \rightarrow \exists y\ \text{Manager}(x, y)\ )$

$Q_2 = \forall x(\ \text{Employee}(x))$

$Q = \exists x \forall y(\ \text{Employee}(y) \rightarrow\ \text{Manager}(x, y)\ )$

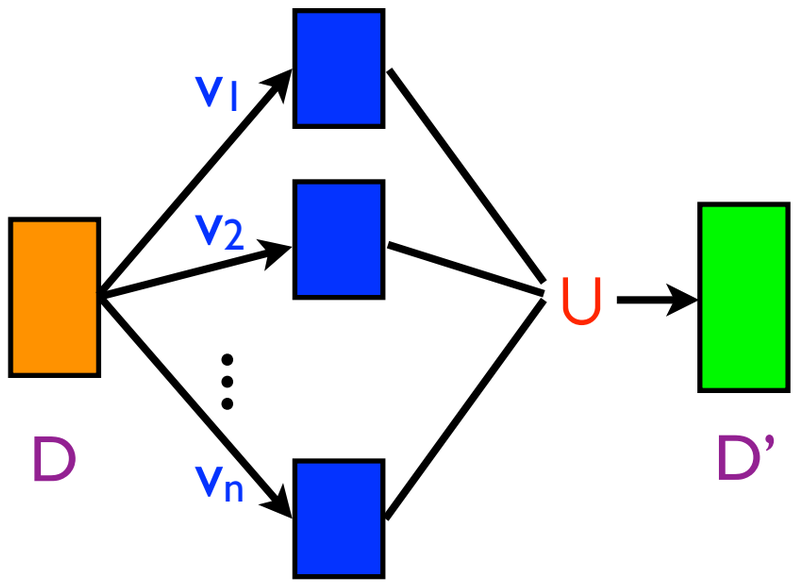




- Naïve evaluation works for Q_1 and Q_2 under $(\!|\cdot|\!)_{CWA}$ $\Leftarrow Q_1, Q_2 \in \exists$Pos+$\forall$G^{bool}
- Naïve evaluation does not work for Q under the $(\!|\cdot|\!)_{CWA}$ $\Rightarrow Q \notin \exists$Pos+$\forall$G^{bool}

Minimal semantics

- A special form of powerset semantics was introduced in the field of deductive databases (GCWA [Minker '82])
- Later modified and adopted as *data exchange* semantics (GCWA* [Hernich'11])
- We define it here for arbitrary incomplete instances:



Minimal Powerset CWA

$D' \in (\!|D|\!)^{min}_{CWA}$ iff

$\exists$ *D-minimal* valuations $v_1, \ldots v_n$

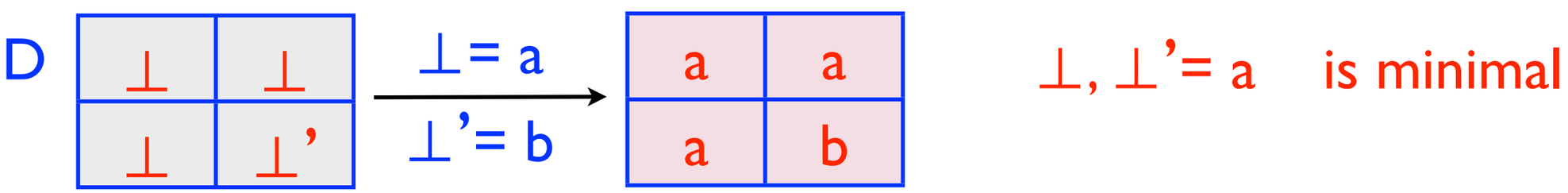
$D' = \bigcup_i v_i(D)$

A valuation v on D is D-minimal if there is no valuation v' s.t. $v'(D) \subsetneq v(D)$

Minimal powerset semantics: U is replaced by any suitable relation $\mathbf{R} \subseteq 2^C \times C$

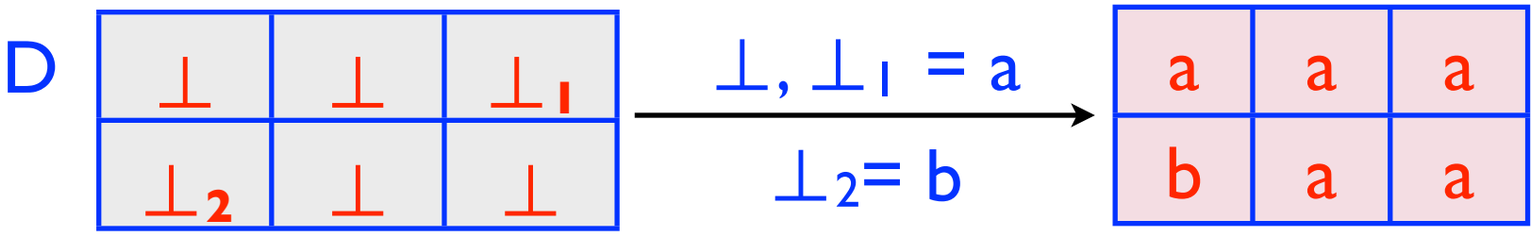
Minimal semantics and the core

- Not all valuations are minimal:



⊥, ⊥'= a is minimal

- true also if D is a *core*



⊥, ⊥1, ⊥2 = a
is minimal

- but if v is a minimal valuation v(D) = v(core(D))

- There are other important connections between minimal semantics and the core (later)

Core of D
substructure D' of D such that D → D'
and there is no D'' ⊊ D' s.t. D → D''
(→: homomorphism preserving constants)

Minimal semantics and the saturation property

Naïve evaluation works for Q under $[\![\ ]\!]$

⇕

Q is "monotone" w.r.t. $[\![\ ]\!]$

saturated domain

Saturation property for $\langle\, \mathcal{D}, C\,, [\![\ ]\!]\,, \approx \,\rangle$:

For all $x \in \mathcal{D}$ there exists $y \in [\![x]\!]$ $\quad y \approx x$

Under the minimal Powerset CWA the saturation property does not hold

D

⊥	⊥
⊥	⊥'

- all D-minimal images are of the form

a	a

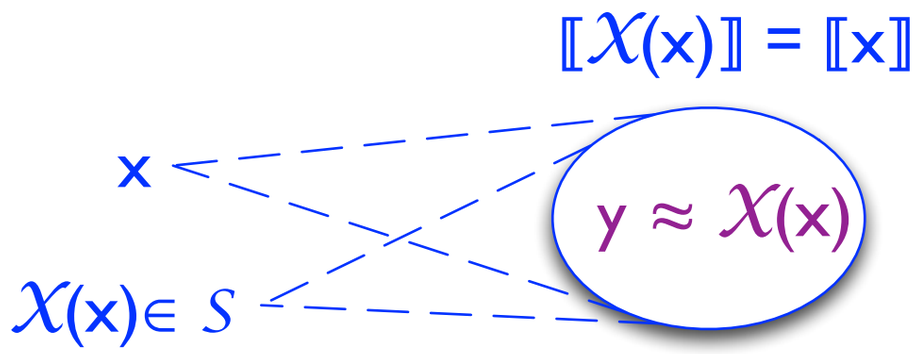
- No union of D-minimal images can be isomorphic to D

The saturation property revisited

$\langle\, \mathcal{D}, C\,, [\![\ ]\!]\,, \approx \,\rangle$ has a saturated subdomain if $\exists\ S$ with $C \subseteq S \subseteq \mathcal{D}$

and a function $\mathcal{X} : \mathcal{D} \to S$ (the *representative* function) s.t.

- $\langle\, S, C\,, [\![]\!]\,, \approx \,\rangle$ is saturated
- $[\![\mathcal{X}(x)]\!] = [\![x]\!]$ for all $x \in \mathcal{D}$



Proposition

If a database domain has a saturated subdomain with representative function $\mathcal{X}$ and Q is a generic Boolean query

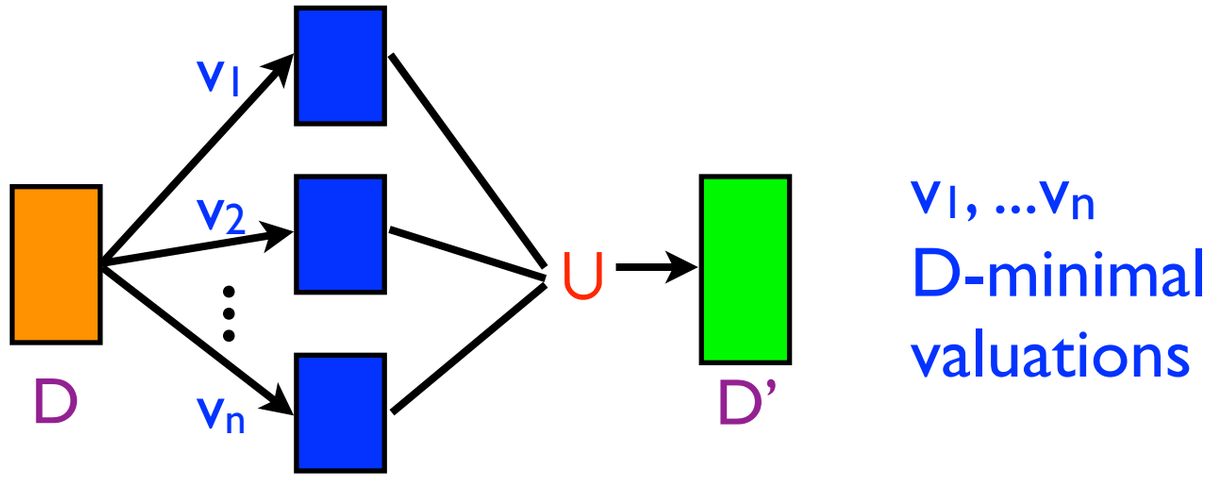
Naïve evaluation works for Q iff
Q is monotone w.r.t. $[\![\ ]\!]$ and $Q(x) = Q(\mathcal{X}(x))$ for all x

Lemma: Under the minimal powerset CWA the set of cores is a saturated subdomain with representative function core(·)

Monotonicity and preservation for minimal semantics

- Monotonicity under minimal powerset CWA is

preservation under the mapping



- Query genericity used with care:
  - valuations are indistinguishable from homomorphisms, however
  - $v_1, \ldots v_n$ are minimal w.r.t all other valuations (not all arbitrary homomorphisms)
  - $v_1, \ldots v_n$ preserve the same elements of D

Naïve evaluation and preservation for minimal semantics

The right notion of preservation:

- D-minimal homomorphism h:
  there exists no h', preserving all constants preserved by h, s.t. $h'(D) \subsetneq h(D)$
- Unions of minimal homomorphisms: homomorphisms $D \rightarrow \bigcup_{i=1}^{n} h_i(D)$

  with $h_1 ... h_n$ D-minimal and preserving the same constants

Theorem
If Q is a generic Boolean query

Naïve evaluation works for Q under the minimal powerset CWA iff
Q is preserved under unions of minimal homomorphisms and
Q(D)=Q(core(D)) for every database D

Similar results hold for arbitrary minimal semantics

Preservation and syntactic fragments for minimal semantics

- Preservation under unions of minimal homomorphisms : no "tight" syntactic fragment known
- *Remark:* unions of minimal homomorphisms are also unions of strong onto homomorphisms

If Q is a Boolean query from $\exists Pos+\forall G^{bool}$, under the minimal powerset CWA:

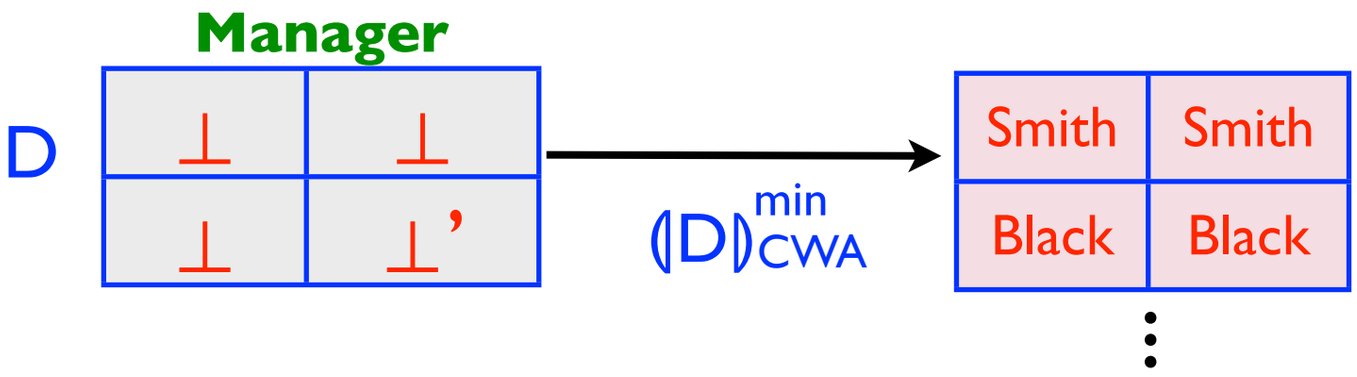
- Naïve evaluation works for Q iff Q(D) = Q(core(D)) for every D
- Naïve evaluation works over cores

Preservation and syntactic fragments for minimal semantics

- Preservation under unions of minimal homomorphisms : no "tight" syntactic fragment known
- *Remark:* unions of minimal homomorphisms are also unions of strong onto homomorphisms

If Q is a Boolean query from $\exists Pos+\forall G^{bool}$, under the minimal powerset CWA:

- Naïve evaluation works for Q iff Q(D) = Q(core(D)) for every D
- Naïve evaluation works over cores



$Q = \forall x, y(\ Manager(x, y) \rightarrow x = y\ )$

$Q(D) = false$ $cert(Q, D) = true$

$Q \in \exists Pos+\forall G^{bool}$ but naïve evaluation does not work

$Q(D) \neq Q(\ core(D)\ )$

Non-Boolean queries

All results can be lifted to non-boolean relational queries. For a k-ary query Q:

- Define a new database domain whose elements are pairs (D, t)
  D: a relational database t : a k-tuple of constants
- Define a boolean query Q^* s.t. $Q^*(D,t)=true$ iff $t \in Q(D)$
- Apply previous results to Q^* and the new database domain ⇒
  derive results for Q over the original relational database domain

For k-ary FO queries , $k \geq 0$

Semantics	Naïve evaluation works for
OWA	∃Pos
WCWA	Pos
CWA	Pos+∀G
Powerset CWA	$\exists Pos+\forall G^{bool}$
Min Powerset CWA	$\exists Pos+\forall G^{bool}$ iff Q(D)=Q(core(D))