Efficiently querying incomplete data

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

February 3rd 2015
Plan

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

Incompleteness and data heterogeneity

Commonly only some concepts shared by the two data sources

✓ e.g. no information about the manager from the source
Querying incomplete data

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?

Q: which employees are managers?

<table>
<thead>
<tr>
<th>Employee</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>mgr</td>
</tr>
<tr>
<td>Green</td>
<td>?</td>
</tr>
<tr>
<td>White</td>
<td>?</td>
</tr>
<tr>
<td>Brown</td>
<td>?</td>
</tr>
<tr>
<td>Black</td>
<td>?</td>
</tr>
</tbody>
</table>

incomplete database
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 \(X, Y\)
  
  \(|X| > |Y| \text{ and } X - Y = \emptyset\)

- 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) $\sigma$: a set of relation symbols, with arities

$$\sigma = \{ \text{Employee, Manager} \} \quad \text{arity(Employee)} = 1, \quad \text{arity(Manager)} = 2$$

Incomplete database instance (naïve table) of schema $\sigma$ [Imielinski, Lipski ’84]:

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

<table>
<thead>
<tr>
<th>Employee</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_1$</td>
</tr>
<tr>
<td>Brown</td>
<td>Brown</td>
</tr>
<tr>
<td>Green</td>
<td>$x_2$</td>
</tr>
</tbody>
</table>

$\text{Const}$: a countably infinite set of constants

$\text{Var}$: a countably infinite set of variables (nulls)

Complete instance: over $\text{Const}$

$\text{dom}(I)$: the subset of $\text{Const} \cup \text{Var}$ occurring in $I$
Any incomplete database represents a set of complete databases (possible worlds)

Semantics of incompleteness:
a function $\llbracket I \rrbracket$ associating with each incomplete database a set of complete databases
Semantics of incompleteness

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

**OWA:**

\[[I]_{OWA} = \{ D \text{ over } Const \mid D \supseteq v(I) \text{ for some } v: Var \rightarrow Const \}\]

\(v: valuation\)

**Interpretation of incompleteness:**

- missing data values, missing tuples
Semantics of incompleteness

CWA:
\[
\llbracket I \rrbracket_{CWA} = \{ D \mid D = v(I) \text{ for some } v : \text{Var} \to \text{Const} \}
\]

Interpretation of incompleteness:
• missing data values
• no missing tuples
Semantics of incompleteness

\[ [I]_{WCWA} = \{ D \mid D \supseteq v(I), \text{dom}(D) = \text{dom}(v(I)) \text{ for some } v : \text{Var} \rightarrow \text{Const} \} \]

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
Query over $\sigma$: a mapping $Q$ associating to each complete instance $I$ of $\sigma$ a relation over $\text{dom}(I)$.

**Q:** which employees are managers?

<table>
<thead>
<tr>
<th>Employee</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>White</td>
<td>White</td>
</tr>
<tr>
<td>Brown</td>
<td>Brown</td>
</tr>
</tbody>
</table>

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

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

Boolean query over $\sigma$:
a mapping $Q$ associating to each complete instance $I$ of $\sigma$ a value in $\{true, false\}$

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

<table>
<thead>
<tr>
<th>Employee</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>White</td>
<td>White</td>
</tr>
<tr>
<td>Brown</td>
<td>Brown</td>
</tr>
<tr>
<td></td>
<td>Green</td>
</tr>
<tr>
<td></td>
<td>Black</td>
</tr>
</tbody>
</table>

$I$ $\rightarrow$ $true$ $Q(I)$
Querying incomplete databases

Semantics of query answering: *certain answers*

\[ \text{cert}_Q(I) = \bigwedge_{D \in [I]} Q(D) \]
Certain answers

Example

Q: “There is a manager who has a manager”

\[ \exists x, y, z \ ( \text{Manager}(x, y) \land \text{Manager}(z, x) ) \]

\[
\begin{array}{c|c|c}
\text{Employee} & \text{Manager} \\
\hline
\text{Green} & \text{Green} & \mathbf{x_1} \\
\text{Brown} & \mathbf{x_1} & \text{Brown} \\
\text{Green} & \text{Green} & \mathbf{x_2} \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{Manager} & \text{Manager} \\
\hline
\text{Green} & \text{Green} & \text{White} \\
\text{White} & \text{White} & \text{Brown} \\
\text{Brown} & \text{Green} & \text{Black} \\
\end{array}
\]

\[
\left[\text{I}\right]_{\text{CWA} / \text{OWA} / \text{WCWA}}
\]

\[
\text{cert}_\text{Q}(\text{I}) = \text{true}
\]

under either CWA, OWA and 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 works for $Q$:

$Q(I)$: $Q$ evaluated directly on $I$, as if variables were new distinct constants

$x_i \neq x_j$ for $i \neq j$

$x_i \neq c$ for all $c \in \text{Const}$

$Q(I) = \text{cert}_Q(I)$ for all $I$
Naïve evaluation

Example

Q: “There is a manager who has a manager”

\[ \exists x, y, z \ ( \text{Manager}(x, y) \land \text{Manager}(z, x) ) \]

<table>
<thead>
<tr>
<th>Employee</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Green</td>
</tr>
<tr>
<td>(x_1)</td>
<td>(x_1)</td>
</tr>
<tr>
<td>Brown</td>
<td>Brown</td>
</tr>
<tr>
<td></td>
<td>(x_2)</td>
</tr>
</tbody>
</table>

\[ Q(I) = \text{true} \]

\[ \text{cert}_Q(I) = \text{true} \text{ under CWA, OWA and WCWA} \]
Naïve evaluation

Example  Q: “There is a manager who has a manager”
\[ \exists x, y, z \ ( \text{Manager}(x, y) \land \text{Manager}(z, x) ) \]

Generalizing:
\[ Q(I) = \text{cert}_Q(I) \text{ for all } I \]
\[ \Rightarrow \text{naïve evaluation works for } Q \text{ under CWA, OWA and WCWA} \]
Naïve evaluation in theory and practice

**Certain answers**: an entailment problem (checking that $I \models Q$) HARD

**Naïve evaluation**: a model-checking problem (checking $I \vDash Q$) 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”

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

\[
\begin{array}{|c|c|}
\hline
\text{Employee} & \text{Manager} \\
\hline
\text{Green} & \text{Green} \\
\hline
x_1 & x_1 \\
\hline
\text{Brown} & \text{Brown} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{Green} & \text{White} \\
\hline
\text{White} & \text{Brown} \\
\hline
\text{Brown} & \text{Black} \\
\hline
\end{array}
\]

\[ Q(I) = \text{true} \]

\[ \text{cert}_Q(I) = \text{false} \text{ under OWA and WCWA} \]

\[ \Rightarrow \text{naïve evaluation does not work for Q under OWA and WCWA} \]
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:

- 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
Monotonicity and preservation

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

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 \( \langle D, C, \llbracket \rrbracket, \approx \rangle \)

<table>
<thead>
<tr>
<th>(D) : a set</th>
<th>database objects (complete and incomplete)</th>
<th>all naïve tables over a fixed schema (\sigma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C) : a subset of (D)</td>
<td>complete database objects</td>
<td>all complete instances over (\sigma)</td>
</tr>
<tr>
<td>(\llbracket \rrbracket : D \rightarrow 2^C)</td>
<td>semantics of incompleteness</td>
<td>(\llbracket \rrbracket_{OVA}, \llbracket \rrbracket_{CWA}, \text{etc.})</td>
</tr>
<tr>
<td>(\approx) : an equivalence relation on (D)</td>
<td>equivalence of objects (w.r.t. queries)</td>
<td>isomorphism of relational instances</td>
</tr>
</tbody>
</table>

Boolean query: \(Q : D \rightarrow \{\text{true, false}\}\)

- generic: \(x \approx y\) implies \(Q(x) = Q(y)\)
- monotone w.r.t. \(\llbracket \rrbracket\): \(y \in \llbracket x \rrbracket\) implies \(Q(x) \Rightarrow Q(y)\)
Naïve evaluation and monotonicity

Naïve evaluation works for \( Q \) under \( \llbracket \cdot \rrbracket \)

\( Q \) is “monotone” w.r.t. \( \llbracket \cdot \rrbracket \)

\( Q \) is preserved under a class of homomorphisms

Preservation theorems

\( Q \) is expressible in a syntactic fragment

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

Naïve evaluation works for \( Q \) iff \( Q \) is monotone w.r.t. \( \llbracket \cdot \rrbracket \)

Saturation property for \( \langle \mathcal{D}, C, \llbracket \cdot \rrbracket, \approx \rangle \):

For all \( x \in \mathcal{D} \) there exists \( y \in \llbracket x \rrbracket \) \( y \approx x \)

holds for most common semantics
Monotonicity and preservation

Naïve evaluation works for $Q$ under $\llbracket \llbracket$.

$Q$ is “monotone” w.r.t. $\llbracket \llbracket$.

$Q$ is preserved under a class of homomorphisms.

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

Preservation theorems

$Q$ is expressible in a syntactic fragment.
Monotonicity and preservation

**Monotonicity and preservation**

**Homomorphism** $D \rightarrow D'$: a mapping $h: \text{dom}(D) \rightarrow \text{dom}(D')$ s.t. $h(D) \subseteq D'$

![Diagram](image)

$D$ and $D'$ with elements $a, b, c, d$

- $a, b \rightarrow c$
- $a \rightarrow d$

$Q$ preserved under homomorphism:

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

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

Monotonicity w.r.t different semantics $\leftrightarrow$ preservation under different notions of homomorphism
Preservation and syntactic fragments

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

$Q$ is “monotone” w.r.t. $[\cdot]$

$Q$ is preserved under a class of homomorphisms

$Q$ is expressible in a syntactic fragment

Preservation theorems

- syntactic characterizations of preservation properties of queries in a given logic
- classical results in (finite) model theory
Homomorphism Preservation Theorem: over arbitrary structures
an FO query $Q$ is preserved under homomorphism iff $Q$ is in $\exists\text{Pos}$

- recently proved over finite structures [Rossman ’08]

$\exists\text{Pos} : \{\exists, \land, \lor\}$-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 $\exists \text{Pos}$

- recently proved over finite structures [Rossman ’08]

\[
\exists \text{Pos} : \{\exists, \land, \lor\}-\text{FO} \quad (\text{Unions of Conjunctive Queries})
\]

**Lyndon Positivity Theorem** [Lyndon ’59]: over arbitrary structures an FO query $Q$ is preserved under onto homomorphism iff $Q$ is in $\text{Pos}$

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

\[
\text{Pos} : \{\exists, \forall, \land, \lor\}-\text{FO}
\]
Preservation and syntactic fragments

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

$Q$ is “monotone” w.r.t. $[[ ]]$

$Q$ is preserved under a class of homomorphisms

Preservation theorems:

$(\text{Syntax} \Rightarrow \text{Preservation})$ holds in the finite as well

$\Rightarrow$ classes of queries where naïve evaluation works

$Q$ is expressible in a syntactic fragment
Naïve evaluation and syntactic fragments

Three well known semantics as instances of our framework

Naïve evaluation works under:

- **OWA**
  - Preservation under homomorphism
  - Preserves under $\exists\text{Pos}$
  - Preserves under monotone w.r.t. $\exists\text{Pos}$

- **WCWA**
  - Preservation under onto homomorphism
  - Preserves under $\text{Pos}$

- **CWA**
  - Preservation under “strong onto” homomorphism
  - Preserves under $\text{Pos} + \forall G$
Examples revisited

Q: “There is a manager who has a manager”

\[ \exists x, y, z \ ( \text{Manager}(x, y) \land \text{Manager}(z, x) ) \]

⇒ naïve evaluation works for Q under CWA OWA, WCWA
Examples revisited

Q: “All employees are managers”
\[ \forall x \left( \text{Employee}(x) \rightarrow \exists y \ \text{Manager}(x, y) \right) \]

⇒ naïve evaluation works for Q under CWA
(recall: not true under OWA, nor under WCWA)

\[ \text{Pos+}\forall G \] extends Pos with a limited form of negation (universal guards)

- a very natural fragment

Naïve evaluation works well beyond \( \exists \text{Pos} \) 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 under:

Powerset semantics

Preservation under unions of strong onto homomorphisms

Minimal semantics

Preservation under unions of minimal homomorphisms

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

\( \exists \mathsf{Pos} \land \forall G^{\mathsf{bool}} \)
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 name="Hull" />
    <author name="Vianu" />
  </book>
  <book title="...">
    <author name="Vianu" />
    ....
  </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 $X - Y \neq \emptyset$
  
  ```sql
  EXISTS ( SELECT X.A FROM X
  WHERE X.A NOT IN ( SELECT Y.A FROM Y ) )
  ```
- $X.A = \{1, 2, 3, \ldots, N\}$ and $Y.A = \{\text{null}\}$, then $X - Y = \emptyset$ 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 $h: \text{dom}(D) \rightarrow \text{dom}(D')$ s.t.

$h(D) \subseteq D'$

Onto homomorphism $D \rightarrow D'$:

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

$h(\text{dom}(D)) = \text{dom}(D')$

Strong onto homomorphism $D \rightarrow D'$:

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

$h(D) = D'$
Homomorphisms

- **Union of strong onto homomorphisms** \( D \rightarrow D' : \bigcup_i h_i (D) = D' \)

- **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** \( D \rightarrow D' : \bigcup_i h_i (D) = D' \)

  with \( h_1 \ldots h_n \) \( D \)-minimal and preserving the same constants
Homomorphism-based relational semantics

**OWA**

\[
\begin{align*}
D & \xrightarrow{v} v(D) \supseteq D' \\
\forall v(D) & \subseteq D' \\
D' & \in \llbracket D \rrbracket_{OWA} \iff \exists v v(D) \subseteq D' \\
\end{align*}
\]

**CWA**

\[
\begin{align*}
D & \xrightarrow{v} v(D) = D' \\
\forall v(D) & = D' \\
D' & \in \llbracket D \rrbracket_{CWA} \iff \exists v v(D) = D' \\
\end{align*}
\]
Homomorphism-based relational semantics

- 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

- 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]**

\[ D' \in \llbracket D \rrbracket_{OWA} \iff \exists v \ v(D) \subseteq D' \]

\[ D' \in \llbracket D \rrbracket_{CWA} \iff \exists v \ v(D) = D' \]

\[ D' \in \llbracket D \rrbracket_{WCWA} \iff \exists v \ v(D) \subseteq D' \land \text{dom}(D') = \text{dom}(v(D)) \]
### Homomorphism-based relational semantics

<table>
<thead>
<tr>
<th>OWA</th>
<th>$D \xrightarrow{v} v(D) \subseteq D'$</th>
<th>$D' \in \llbracket D \rrbracket_{OWA}$ iff $\exists v \ v(D) \subseteq D'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWA</td>
<td>$D \xrightarrow{v} v(D) = D'$</td>
<td>$D' \in \llbracket D \rrbracket_{CWA}$ iff $\exists v \ v(D) = D'$</td>
</tr>
<tr>
<td>WCWA</td>
<td>$D \xrightarrow{v} v(D) \subseteq \text{domain}(D')$</td>
<td>$D' \in \llbracket D \rrbracket_{WCWA}$ iff $\exists v \ v(D) \subseteq D' \land \text{dom}(D') = \text{dom}(v(D))$</td>
</tr>
</tbody>
</table>

Can be generalized to arbitrary semantic relations...
### Homomorphism-based Relational Semantics

<table>
<thead>
<tr>
<th>OWA</th>
<th>$\mathcal{D} \xrightarrow{v} \mathcal{v}(\mathcal{D}) \subseteq \mathcal{D}'$</th>
<th>$\mathcal{D}' \in \llbracket \mathcal{D} \rrbracket_{\text{OWA}}$ iff $\exists v \ \mathcal{v}(\mathcal{D}) \subseteq \mathcal{D}'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWA</td>
<td>$\mathcal{D} \xrightarrow{v} \mathcal{v}(\mathcal{D}) = \mathcal{D}'$</td>
<td>$\mathcal{D}' \in \llbracket \mathcal{D} \rrbracket_{\text{CWA}}$ iff $\exists v \ \mathcal{v}(\mathcal{D}) = \mathcal{D}'$</td>
</tr>
<tr>
<td>WCWA</td>
<td>$\mathcal{D} \xrightarrow{v} \mathcal{v}(\mathcal{D}) \subseteq \text{domain} \mathcal{D}'$</td>
<td>$\mathcal{D}' \in \llbracket \mathcal{D} \rrbracket_{\text{WCWA}}$ iff $\exists v \ \mathcal{v}(\mathcal{D}) \subseteq \mathcal{D}' \land \text{dom}(\mathcal{D}') = \text{dom}(\mathcal{v}(\mathcal{D}))$</td>
</tr>
<tr>
<td>R-based</td>
<td>$\mathcal{D} \xrightarrow{v} \mathcal{v}(\mathcal{D}) \xrightarrow{R} \mathcal{D}'$</td>
<td>$\mathcal{D}' \in \llbracket \mathcal{D} \rrbracket_{R}$ iff $\exists v \ \mathcal{v}(\mathcal{D}) R \mathcal{D}'$</td>
</tr>
</tbody>
</table>

$R$: reflexive binary relation between complete instances
Monotonicity and preservation

- **R-homomorphism** $D \rightarrow D'$ (D and D' complete):
  
a mapping $h$ over $\text{dom}(D)$ s.t. $h(D) \text{ R } D'$

- 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

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

- Naïve evaluation works for \( Q \) under \( \llbracket \cdot \rrbracket \)
- \( Q \) is “monotone” w.r.t. \( \llbracket \cdot \rrbracket \)
- \( Q \) is preserved under a class of homomorphisms
- \( Q \) belongs to a syntactic fragment

Preservation theorems

Combining the two steps:

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

\[ \llbracket D \rrbracket \]

\[ \nu^*(D) \cong D \]

\( \nu^* \): 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

<table>
<thead>
<tr>
<th>$R$</th>
<th>R-homomorphism</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWA</td>
<td>$\subseteq$ homomorphisms</td>
</tr>
<tr>
<td>CWA</td>
<td>= strong onto homomorphisms (i.e. homomorphisms $D \rightarrow h(D)$)</td>
</tr>
<tr>
<td>WCWA</td>
<td>$\subseteq$ onto homomorphisms</td>
</tr>
</tbody>
</table>

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

$Q$ is preserved under a class of homomorphisms

$Q$ belongs to a syntactic fragment

Preservation theorems
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)**

\[
\varphi := T \mid \bot \mid R(\overline{x}) \mid x = y \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \exists x \varphi \mid \forall x \varphi
\]

\[
\left( \forall \overline{x} \left( G(\overline{x}) \rightarrow \varphi \right) \right)
\]

with

- \( G \) : a relation or equality symbol
- \( \overline{x} \) : a tuple of distinct variables

**Pos+∀G formulas are preserved under strong onto homomorphisms**
Semantics arising from orderings

- Information ordering of Codd tuples:
  \[ \text{if } t[i] \text{ constant } \Rightarrow t'[i] = t[i] \quad \text{then } t' \text{ is “more informative” than } t \]

  \[
  t \begin{array}{c|c|c|c|}
  a & \bot_1 & b & \bot_2 \\
  \end{array} \sqsubseteq \begin{array}{c|c|c|c|}
  a & a & b & \bot_3 \\
  \end{array}
  \]

- 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)
  - \( D \sqsubseteq^P D' : D \sqsubseteq^H D' \quad \text{and} \quad \forall t' \in D' \quad \exists t \in D \quad t \sqsubseteq t' \) (Plotkin ordering)
Semantics arising from orderings

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

\[ D \sqsubseteq^P D' : D \sqsubseteq^H D' \text{ and } \forall t \in D' \ \exists t \in D \ \ t \sqsubseteq t' \] (Plotkin ordering)

- Orderings give rise to semantics of Codd databases:
  \[ \llbracket D \rrbracket = \{ \text{complete } D' \mid D \sqsubseteq D' \} \]

- Observe that Plotkin semantics is more “open” than CWA:

\( \square \)
• Extend (and generalize) Plotkin semantics to naïve databases

\[
D' \in (D)_{\text{CWA}} \iff \exists \text{ valuations } v_1, \ldots, v_n \ D' = \bigcup_i v_i(D)
\]

• When restricting to Codd databases Powerset CWA coincides with Plotkin

• Gives rise to a whole new class of semantics: \( \bigcup \) is replaced by any suitable relation

\[
R \subseteq 2^C \times C
\]
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:
  \[
  \exists \text{Pos}^+ \forall \text{G}^{\text{bool}} : \\
  \exists \text{Pos} \text{ extended with universal guarded sentences } \forall \vec{x} \left( \text{G}(\vec{x}) \to \varphi(\vec{x}) \right)
  \]

**Corollary**
If \( Q \) is a Boolean query from \( \exists \text{Pos}^+ \forall \text{G}^{\text{bool}} \)
Naïve evaluation works for \( Q \) under \((\cdot)^{\text{CWA}}\)
The $\exists$Pos+$\forall$G\text{bool} fragment

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

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

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

- Naïve evaluation works for $Q_1$ and $Q_2$ under $\langle \cdot \rangle_{CWA} \iff Q_1, Q_2 \in \exists$Pos+$\forall$G\text{bool}$

- Naïve evaluation does not work for $Q$ under the $\langle \cdot \rangle_{CWA} \Rightarrow Q \not\in \exists$Pos+$\forall$G\text{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:

\[ D' = \bigcup_i v_i(D) \]

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 \( R \subseteq 2^C \times C \)
Minimal semantics and the core

- Not all valuations are minimal:

\[ \bot = a \]
\[ \bot' = b \]

\[ \bot, \bot' = a \] is minimal

- true also if \( D \) is a core

\[ \bot, \bot_1 = a \]
\[ \bot_2 = b \]

\[ \bot, \bot_1, \bot_2 = a \] is minimal

- but if \( v \) is a minimal valuation \( v(D) = v(\text{core}(D)) \)

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

Core of \( D \)

substructure \( D' \) of \( D \) such that \( D \rightarrow D' \)
and there is no \( D'' \) s.t. \( D \rightarrow D'' \) (\( \rightarrow \): homomorphism preserving constants)
Minimal semantics and the saturation property

Saturation property for $\langle D, C, \llbracket \cdot \rrbracket, \approx \rangle$:

For all $x \in D$ there exists $y \in \llbracket x \rrbracket$ such that $y \approx x$.

Under the minimal Powerset CWA, the saturation property does not hold.

- All $D$-minimal images are of the form $a$ and $a$.
- 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 \( X : \mathcal{D} \rightarrow S \) (the representative function) s.t.

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

**Proposition**

If a database domain has a saturated subdomain with representative function \( X \) and \( Q \) is a generic Boolean query

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

**Lemma:** Under the minimal powerset CWA the set of cores is a saturated subdomain with representative function \( \text{core}(\cdot) \)
Monotonicity and preservation for minimal semantics

- Monotonicity under minimal powerset CWA is preservation under the mapping:

\[ D \rightarrow U \rightarrow D' \]

- 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 $D \to \bigcup_{i=1}^{n} h_i(D)$
  
  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 $\exists Pos^+ \forall G^\text{bool}$, under the minimal powerset CWA:
- Naïve evaluation works for Q iff $Q(D) = Q(\text{core}(D))$ 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 $\exists \text{Pos}^+ \forall G^\text{bool}$, under the minimal powerset CWA:

- Naïve evaluation works for Q iff $Q(D) = Q(\text{core}(D))$ for every $D$
- Naïve evaluation works over cores

$Q = \forall x, y (\text{Manager}(x, y) \rightarrow x = y)$
$Q(D) = \text{false}$ $\text{cert}(Q, D) = \text{true}$

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

$Q(D) \neq Q(\text{core}(D))$
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. \(Q^*(D, t) = \text{true} \iff t \in Q(D)\)

- 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, \(k \geq 0\)

<table>
<thead>
<tr>
<th>Semantics</th>
<th>Naïve evaluation works for</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWA</td>
<td>(\exists Pos)</td>
</tr>
<tr>
<td>WCWA</td>
<td>(Pos)</td>
</tr>
<tr>
<td>CWA</td>
<td>(Pos + \forall G)</td>
</tr>
<tr>
<td>Powerset CWA</td>
<td>(\exists Pos + \forall G^{\text{bool}})</td>
</tr>
<tr>
<td>Min Powerset CWA</td>
<td>(\exists Pos + \forall G^{\text{bool}} \iff Q(D) = Q(\text{core}(D)))</td>
</tr>
</tbody>
</table>