Semantics of Query Answering in Data Exchange

André Hernich

Department of Computer Science
Humboldt University Berlin

DEIS 2010, Dagstuhl
Outline

1. Goals of Query Answering in Data Exchange
2. The Basic Query Answering Semantics
3. Alternative Semantics
Goal: Answer queries posed against target data
(Fagin, Kolaitis, Miller, Popa ’03)
Example

Source instance:

<table>
<thead>
<tr>
<th>Book</th>
<th>title</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

Solution:

<table>
<thead>
<tr>
<th>Author</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publ</th>
<th>title</th>
<th>a_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>2</td>
</tr>
</tbody>
</table>

Schema mapping:

\[ \forall t \forall a \left( \text{Book}(t, a) \rightarrow \exists id \text{ Author}(id, a) \land \text{Publ}(t, id) \right) \]
Example

Source instance:

<table>
<thead>
<tr>
<th>Book</th>
<th>title</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

Solution:

<table>
<thead>
<tr>
<th>Author</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publ</th>
<th>title</th>
<th>a_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>2</td>
</tr>
</tbody>
</table>

Schema mapping:

- $\forall t \forall a \left( \text{Book}(t, a) \rightarrow \exists id \ \text{Author}(id, a) \land \text{Publ}(t, id) \right)$

Example query over target schema

Who are the authors of "Algebra"?

$$Q(a) := \exists id \left( \text{Publ}(\text{"Algebra"}, id) \land \text{Author}(id, a) \right)$$
1. What is the “right” answer to/semantics of a query?
Example

Source instance:

<table>
<thead>
<tr>
<th>Book</th>
<th>title</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

Solution:

<table>
<thead>
<tr>
<th>Author</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publ</th>
<th>title</th>
<th>a_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>2</td>
</tr>
</tbody>
</table>

Schema mapping:

\[ \forall t \forall a ( \text{Book}(t, a) \rightarrow \exists id \text{Author}(id, a) \land \text{Publ}(t, id) ) \]

Example query over target schema

Who are the authors of “Algebra”?

\[ Q(a) \equiv \exists id ( \text{Publ} (“Algebra”, id) \land \text{Author}(id, a) ) \]
1. What is the “right” answer to/semantics of a query?

Problem: many solutions with different sets of answers
Fundamental Issues

1. What is the “right” answer to/semantics of a query?
   **Problem:** many solutions with *different* sets of answers

2. Which solutions are appropriate for query answering?
   **Problem:** queries have to be answered *without* source instance
1. What is the “right” answer to/semantics of a query?
   **Problem**: many solutions with *different* sets of answers

2. Which solutions are appropriate for query answering?
   **Problem**: queries have to be answered *without* source instance

3. What is the complexity of query answering?
   (computing the solution & evaluating the query)
Outline

1. Goals of Query Answering in Data Exchange
2. The Basic Query Answering Semantics
3. Alternative Semantics
Idea: return “safe” answers
The Certain Answers Semantics

Idea: return “safe” answers

\[Q(T_1) = \{a_1, a_2, \ldots\}\]
\[Q(T_2) = \{b_1, b_2, \ldots\}\]
\[Q(T_3) = \{c_1, c_2, \ldots\}\]

Definition (Fagin, Kolaitis, Miller, Popa '03)
a is a certain answer to Q on M and S if and only if a ∈ Q(T) for all solutions T for S under M.
The Certain Answers Semantics

Idea: return “safe” answers

\[ Q(T_1) = \{ a_1, a_2, \ldots \} \]
The Certain Answers Semantics

Idea: return “safe” answers

\[ Q(T_1) = \{a_1, a_2, \ldots\} \]
\[ Q(T_2) = \{b_1, b_2, \ldots\} \]

Definition (Fagin, Kolaitis, Miller, Popa '03)

\( a \) is a certain answer to \( Q \) on \( M \) and \( S \) \( \iff \) \( a \in Q(T) \) for all solutions \( T \) for \( S \) under \( M \).
The Certain Answers Semantics

Idea: return “safe” answers

\[ Q(T_1) = \{a_1, a_2, \ldots\} \]
\[ Q(T_2) = \{b_1, b_2, \ldots\} \]
\[ Q(T_3) = \{c_1, c_2, \ldots\} \]

\[ \text{Definition (Fagin, Kolaitis, Miller, Popa '03)} \]

\[ a \text{ is a certain answer to } Q \text{ on } M \text{ and } S \iff a \in Q(T) \text{ for all solutions } T \text{ for } S \text{ under } M \]
The Certain Answers Semantics

Idea: return “safe” answers

\[ Q(T_1) = \{a_1, a_2, \ldots\} \]
\[ Q(T_2) = \{b_1, b_2, \ldots\} \]
\[ Q(T_3) = \{c_1, c_2, \ldots\} \]

Definition (Fagin, Kolaitis, Miller, Popa ’03)

\( a \) is a certain answer to \( Q \) on \( M \) and \( S \)
\[ \iff a \in Q(T) \text{ for all solutions } T \text{ for } S \text{ under } M \]
Example

**Source instance:**

<table>
<thead>
<tr>
<th>Book</th>
<th>title</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

**Solution:**

<table>
<thead>
<tr>
<th>Author</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publ</th>
<th>title</th>
<th>a_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>2</td>
</tr>
</tbody>
</table>

**Schema mapping:**

\[
\forall t \forall a \left( \text{Book}(t, a) \rightarrow \exists id \ \text{Author}(id, a) \land \text{Publ}(t, id) \right)
\]
Source instance:

<table>
<thead>
<tr>
<th>Book</th>
<th>title</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

Solution:

<table>
<thead>
<tr>
<th>Author</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publ</th>
<th>title</th>
<th>a_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>2</td>
</tr>
</tbody>
</table>

Schema mapping:

\[ \forall t \forall a \ ( \text{Book}(t, a) \rightarrow \exists id \ \text{Author}(id, a) \land \text{Publ}(t, id) ) \]

Query: Who are the authors of “Algebra”?

\[ Q(a) := \exists id ( \text{Publ}(“Algebra”, id) \land \text{Author}(id, a) ) \]
Example

Source instance:

<table>
<thead>
<tr>
<th>Book</th>
<th>title</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebra</td>
<td>Lang</td>
<td></td>
</tr>
<tr>
<td>Logic</td>
<td>Hodges</td>
<td></td>
</tr>
</tbody>
</table>

Solution:

<table>
<thead>
<tr>
<th>Author</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publ</th>
<th>title</th>
<th>a_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebra</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Logic</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Schema mapping:

- $\forall t \forall a ( \text{Book}(t, a) \rightarrow \exists id \ \text{Author}(id, a) \land \text{Publ}(t, id) )$

Query: Who are the authors of “Algebra”?

$Q(a) := \exists id ( \text{Publ}(“Algebra”, id) \land \text{Author}(id, a) )$

Certain answers: \{“Lang”\}
Example

Source instance:

<table>
<thead>
<tr>
<th>Book</th>
<th>title</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

Solution:

<table>
<thead>
<tr>
<th>Author</th>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Lang</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Hodges</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publ</th>
<th>title</th>
<th>a_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algebra</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Logic</td>
<td>2</td>
</tr>
</tbody>
</table>

Schema mapping:

- $\forall t \forall a \left( \text{Book}(t, a) \rightarrow \exists id \ \text{Author}(id, a) \land \text{Publ}(t, id) \right)$

Query: Who are the authors of “Algebra”?

$$Q(a) := \exists id \left( \text{Publ} (“Algebra”, id) \land \text{Author}(id, a) \right)$$

Certain answers: \{“Lang”\}
The Certain Answers and UCQs

**Consensus:** suitable for unions of conjunctive queries (UCQs)
**The Certain Answers and UCQs**

**Consensus:** suitable for unions of conjunctive queries (UCQs)

**Theorem (Fagin, Kolaitis, Miller, Popa ’03)**

For every schema mapping $M$, source instance $S$ for $M$, universal solution $T$ for $S$, and UCQ $Q$

$$\text{certain answers to } Q = \{ a \in Q(T) \mid a \text{ is null-free} \}$$
The Certain Answers and UCQs

**Consensus:** suitable for unions of conjunctive queries (UCQs)

**Theorem (Fagin, Kolaitis, Miller, Popa ’03)**

For every schema mapping $M$, source instance $S$ for $M$, universal solution $T$ for $S$, and UCQ $Q$

$$\text{certain answers to } Q = \{ a \in Q(T) \mid a \text{ is null-free} \}$$

“Ingredients” for the proof:

- Solutions for $S$

$T' \overset{h}{\rightarrow} T$

$\bar{a} \in Q(T) \implies h(\bar{a}) \in Q(T') \equiv \bar{a}$
Consensus: suitable for unions of conjunctive queries (UCQs)

Theorem (Fagin, Kolaitis, Miller, Popa '03)

For every schema mapping $M$, source instance $S$ for $M$, universal solution $T$ for $S$, and UCQ $Q$

certain answers to $Q = \{a \in Q(T) \mid a \text{ is null-free}\}$

“Ingredients” for the proof:

$\bar{a} \in Q(T) \implies h(\bar{a}) \in Q(T')$

More general: for queries preserved under homomorphisms
Widely agreed: the certain answers semantics is suitable

issue of appropriate solutions and query answering
less well understood
+ Widely agreed: the certain answers semantics is suitable
- issue of appropriate solutions and query answering less well understood

(Data) complexity results:

- evaluation of UCQs with $\leq 1$ inequality per disjunct in PTIME on universal solutions (Fagin, Kolaitis, Miller, and Popa ’03)
- co-NP-complete for CQs with $\geq 2$ inequalities (Mądry ’05)
- fragments of UCQs with $\leq 2$ inequalities per disjunct in PTIME on universal solutions (Arenas, Barceló, Reutter ’09)
and Monotonic Queries in General

+ Widely agreed: the certain answers semantics is suitable
- issue of appropriate solutions and query answering less well understood

(Data) complexity results:

- evaluation of UCQs with $\leq 1$ inequality per disjunct in PTIME on universal solutions (Fagin, Kolaitis, Miller, and Popa ’03)
- co-NP-complete for CQs with $\geq 2$ inequalities (Mądry ’05)
- fragments of UCQs with $\leq 2$ inequalities per disjunct in PTIME on universal solutions (Arenas, Barceló, Reutter ’09)

“Generic” approach: based on extension of universal solutions (Deutsch, Nash, Remmel ’08)
Counter-intuitive answers possible on non-monotonic queries
(Fagin, Arenas, Barceló, Libkin ’04; Libkin ’06)
Counter-intuitive answers possible on non-monotonic queries  
(Fagin, Arenas, Barceló, Libkin ’04; Libkin ’06)

Example (copy relation \( E \) to \( E' \))

**Schema mapping:** \( \forall x \forall y ( E(x, y) \rightarrow E'(x, y) ) \)

**Source instance:**

**Solution:**

\[
\begin{align*}
E(\text{a}, \text{b}) & \quad E'(\text{a}, \text{b})
\end{align*}
\]
Counter-intuitive answers possible on non-monotonic queries
(Fagin, Arenas, Barceló, Libkin ’04; Libkin ’06)

Example (copy relation $E$ to $E'$)

<table>
<thead>
<tr>
<th>Schema mapping:</th>
<th>$\forall x \forall y ( E(x, y) \rightarrow E'(x, y) )$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source instance:</td>
<td>Solution:</td>
</tr>
</tbody>
</table>

Query: $Q(x) :=$ Is there exactly one $y$ with $E'(x, y)$?

- Expected answers: $\{a\}$
Counter-intuitive answers possible on non-monotonic queries
(Fagin, Arenas, Barceló, Libkin ’04; Libkin ’06)

Example (copy relation $E$ to $E'$)

Schema mapping: $\forall x \forall y ( E(x, y) \rightarrow E'(x, y) )$

Source instance: Solution:

```
Source instance:
```

Query: $Q(x) := \text{Is there exactly one } y \text{ with } E'(x, y)?$

- Expected answers: $\{a\}$
- The certain answers: $\emptyset$
Counter-intuitive answers possible on non-monotonic queries
(Fagin, Arenas, Barceló, Libkin ’04; Libkin ’06)

Example (copy relation $E$ to $E'$)

Schema mapping: $\forall x \forall y ( E(x, y) \rightarrow E'(x, y) )$

Source instance: Solution: Another solution:

Query: $Q(x) :=$ Is there exactly one $y$ with $E'(x, y)$?

- Expected answers: $\{a\}$
- The certain answers: $\emptyset$
Outline

1. Goals of Query Answering in Data Exchange
2. The Basic Query Answering Semantics
3. Alternative Semantics
Dealing with Non-Monotonic Queries

1. Use the certain answers semantics

2. Use alternative semantics
Dealing with Non-Monotonic Queries

1 Use the certain answers semantics
   - manually rule out undesired solutions via suitable constraints

2 Use alternative semantics
Motivating Example Revisited

Example (copy relation $E$ to $E'$)

Schema mapping: $\forall x \forall y \ (E(x, y) \rightarrow E'(x, y))$

Source instance:  

Solution:  

Another solution:  

Query: $Q(x) := \text{Is there exactly one } y \text{ with } E'(x, y)$?

- Expected answers: $\{a\}$
- The certain answers: $\emptyset$
Motivating Example Revisited

Example (copy relation $E$ to $E'$)

Schema mapping:
\[ \forall x \forall y ( E(x, y) \rightarrow E'(x, y) ) \]
\[ \forall x \forall y ( \neg E(x, y) \rightarrow \neg E'(x, y) ) \]

Source instance:
\[ a \xrightarrow{E} b \]

Solution:
\[ a \xrightarrow{E'} b \]

Another solution:
\[ a \xrightarrow{E'} b \]

Query:
\[ Q(x) := \text{Is there exactly one } y \text{ with } E'(x, y)? \]

- Expected answers: \{a\}
- The certain answers: \emptyset
1. Use the certain answers semantics
   - manually rule out undesired solutions via suitable constraints

2. Use alternative semantics
Dealing with Non-Monotonic Queries

1. Use the certain answers semantics
   - manually rule out undesired solutions via suitable constraints
   - requires richer constraint language
   - almost no research in this direction

2. Use alternative semantics

"If something is not mentioned, take it to be false."
Dealing with Non-Monotonic Queries

1. Use the certain answers semantics
   - manually rule out undesired solutions via *suitable constraints*
   - requires richer constraint language
   - almost no research in this direction

2. Use alternative semantics (this talk)
Dealing with Non-Monotonic Queries

1. **Use the certain answers semantics**
   - *manually* rule out undesired solutions via *suitable constraints*
   - requires richer constraint language
   - almost no research in this direction

2. **Use alternative semantics (this talk)**
   - *automatically* rule out undesired solutions *via heuristics*
   - no richer constraint language
   - can build on research from non-monotonic reasoning
Dealing with Non-Monotonic Queries

1. Use the certain answers semantics
   - manually rule out undesired solutions via suitable constraints
   - requires richer constraint language
   - almost no research in this direction

2. Use alternative semantics (this talk)
   - automatically rule out undesired solutions via heuristics
   - no richer constraint language
   - can build on research from non-monotonic reasoning

Basis: variants of Closed World Assumption (CWA) (Reiter ’78)

“If something is not mentioned, take it to be false.”
Motivating Example Revisited

Example (copy relation $E$ to $E'$)

Schema mapping:
$$\forall x \forall y \left( E(x, y) \rightarrow E'(x, y) \right)$$
$$\forall x \forall y \left( \neg E(x, y) \rightarrow \neg E'(x, y) \right)$$

Source instance:  
Solution:  
Another solution:

Query: $Q(x) :=$ Is there exactly one $y$ with $E'(x, y)$?

- Expected answers: $\{a\}$
- The certain answers: $\emptyset$
CWA-Semantics

- for schema mappings defined by *s-t tgds, t-tgds, and egds* (Libkin '06; H., Schweikardt '07)
- family of semantics, based on CWA-solutions (= solutions valid under the CWA-semantics)
CWA-Semantics

• for schema mappings defined by \textit{s-t tgds, t-tgds, and egds} (Libkin '06; H., Schweikardt '07)

• family of semantics, based on \textbf{CWA-solutions} (= solutions valid under the CWA-semantics)

• \textbf{CWA-certain answers semantics}:

\[ S \xrightarrow{M} T_1 \xrightarrow{Q} T_2 \ldots \]

\ldots like the certain answers semantics, \textit{except}: 
CWA-Semantics

- for schema mappings defined by *s-t tgds, t-tgds, and egds* (Libkin '06; H., Schweikardt '07)
- family of semantics, based on **CWA-solutions** (= solutions valid under the CWA-semantics)
- **CWA-certain answers semantics:**

\[ S \xrightarrow{M} T_1 \xrightarrow{Q} T_2 \ldots \]

...like the certain answers semantics, *except*:
- the \( T_i \) are CWA-solutions
CWA-Semantics

- for schema mappings defined by s-t tgds, t-tgds, and egds
  (Libkin '06; H. Schweikardt '07)
- family of semantics, based on CWA-solutions
  (= solutions valid under the CWA-semantics)
- CWA-certain answers semantics:

...like the certain answers semantics, except:
- the $T_i$ are CWA-solutions
- $Q$ is evaluated under a special semantics for instances with nulls
CWA-Solutions

**Rule:** all atoms and facts in CWA-solutions must be justified by the source instance and the schema mapping

<table>
<thead>
<tr>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S = { P(a) } \quad \forall x ( P(x) \rightarrow \exists y \ E(x, y) ) )</td>
</tr>
</tbody>
</table>
Rule: all atoms and facts in CWA-solutions must be justified by the source instance and the schema mapping

Criteria

1. Derivability

Example

\[ S = \{ P(a) \} \quad \forall x \left( P(x) \rightarrow \exists y \ E(x, y) \right) \]
CWA-Solutions

**Rule:** all atoms and facts in CWA-solutions must be **justified** by the source instance and the schema mapping

**Criteria**

1. **Derivability**

**Example**

\[ S = \{ P(a) \} \quad \forall x( P(x) \rightarrow \exists y \ E(x, y) ) \]

**Solution:**

![Diagram](image)
CWA-Solutions

**Rule:** all atoms and facts in CWA-solutions must be **justified** by the source instance and the schema mapping

**Criteria**

1. **Derivability**

**Example**

\[ S = \{ P(a) \} \quad \forall x ( P(x) \rightarrow \exists y E(x, y) ) \]

**Solution:**

![Diagram showing derivability example](attachment:diagram.png)

\( a \) is not derivable from the source instance and schema mapping.
Rule: all atoms and facts in CWA-solutions must be justified by the source instance and the schema mapping

Criteria

1. Derivability

Example

\[ S = \{ P(a) \} \land \forall x ( P(x) \rightarrow \exists y E(x, y) ) \]

Solution:

\[ a \rightarrow c \]
\[ a \rightarrow d \]

Characterization (Libkin ’06; H., Schweikardt ’07)

CWA-solutions = universal solutions derivable from the source instance using a certain variant of the chase

E.g., core solution = minimal CWA-solution
**CWA-Solutions**

**Rule:** all atoms and facts in CWA-solutions must be **justified** by the source instance and the schema mapping

**Criteria**

1. Derivability
2. Parsimony

**Example**

\[ S = \{ P(a) \} \quad \forall x \left( P(x) \rightarrow \exists y \ E(x, y) \right) \]

**Solution:**

- [Diagram showing a, c, and d nodes with connections]

Characterization (Libkin '06; H., Schweikardt '07)

CWA-solutions = universal solutions derivable from the source instance using a certain variant of the chase

E.g., core solution = minimal CWA-solution
Rule: all atoms and facts in CWA-solutions must be justified by the source instance and the schema mapping

Criteria
1. Derivability
2. Parsimony

Example

\[ S = \{ P(a) \} \quad \forall x ( P(x) \rightarrow \exists y E(x, y) ) \]

Solution:

- Same justification used twice

Characterization (Libkin ’06; H., Schweikardt ’07)
CWA-solutions = universal solutions derivable from the source instance using a certain variant of the chase. E.g., core solution = minimal CWA-solution.
**CWA-Solutions**

**Rule:** all atoms and facts in CWA-solutions must be **justified** by the source instance and the schema mapping

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Derivability</td>
<td>( S = { P(a) } \quad \forall x ( P(x) \rightarrow \exists y E(x, y) ) )</td>
</tr>
<tr>
<td>2 Parsimony</td>
<td>Solution: ( a ) ( \rightarrow ) ( c )</td>
</tr>
</tbody>
</table>

Characterization (Libkin '06; H., Schweikardt '07)

CWA-solutions = universal solutions derivable from the source instance using a certain variant of the chase.

E.g., core solution = minimal CWA-solution.
**Rule:** all atoms and facts in CWA-solutions must be justified by the source instance and the schema mapping.

**Criteria**

1. Derivability
2. Parsimony
3. No invented facts

**Example**

Let $S = \{ P(a) \}$, where $\forall x \ (P(x) \rightarrow \exists y \ E(x, y))$.

**Solution:**

```
S = \{ P(a) \} \quad \forall x \ (P(x) \rightarrow \exists y \ E(x, y))
```

![Diagram](attachment:diagram.png)
CWA-Solutions

**Rule:** all atoms and facts in CWA-solutions must be **justified**
by the source instance and the schema mapping

<table>
<thead>
<tr>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Derivability</td>
</tr>
<tr>
<td>2 Parsimony</td>
</tr>
<tr>
<td>3 No invented facts</td>
</tr>
</tbody>
</table>

**Example**

\[ S = \{ P(a) \} \quad \forall x ( P(x) \rightarrow \exists y E(x, y) ) \]

**Solution:**

- Contant \( c \) is invented
**CWA-Solutions**

**Rule:** all atoms and facts in CWA-solutions must be **justified** by the source instance and the schema mapping.

<table>
<thead>
<tr>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>Derivability</strong></td>
</tr>
<tr>
<td>2. <strong>Parsimony</strong></td>
</tr>
<tr>
<td>3. <strong>No invented facts</strong></td>
</tr>
</tbody>
</table>

**Example**

$$S = \{ P(a) \} \quad \forall x \left( P(x) \rightarrow \exists y \ E(x, y) \right)$$

**Solution:**

![Diagram showing the relationship between a and ⊥]
CWA-Solutions

**Rule:** all atoms and facts in CWA-solutions must be **justified** by the source instance and the schema mapping

<table>
<thead>
<tr>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Derivability</td>
</tr>
<tr>
<td>2. Parsimony</td>
</tr>
<tr>
<td>3. No invented facts</td>
</tr>
</tbody>
</table>

**Example**

\[ S = \{ P(a) \} \quad \forall x \left( P(x) \rightarrow \exists y \ E(x, y) \right) \]

unique CWA-solution:

![Diagram](image)
CWA-Solutions

**Rule:** all atoms and facts in CWA-solutions must be justified by the source instance and the schema mapping

<table>
<thead>
<tr>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Derivability</td>
</tr>
<tr>
<td>2. Parsimony</td>
</tr>
<tr>
<td>3. No invented facts</td>
</tr>
</tbody>
</table>

**Example**

\[ S = \{ P(a) \} \quad \forall x \big( P(x) \rightarrow \exists y \ E(x, y) \big) \]

unique CWA-solution:

![Diagram](attachment:image.png)

**Characterization** (Libkin ’06; H., Schweikardt ’07)

CWA-solutions = **universal solutions** derivable from the source instance using a certain variant of the chase

E.g., core solution = minimal CWA-solution
Query Evaluation under the CWA-Semantics

**Theorem (Libkin ’06)**

For every schema mapping $M$ defined by $s$-$t$ tgds, every source instance $S$, and every query $Q$,

$$CWA$$-certain answers to $Q$ on $M$ and $S = \Box Q(T),$$

where $T = \textit{canonical solution}$ for $S$ under $M$. 
Query Evaluation under the CWA-Semantics

**Theorem (Libkin ’06)**

For every schema mapping $M$ defined by $s$-$t$ tgds, every source instance $S$, and every query $Q$,

$$CWA$$-certain answers to $Q$ on $M$ and $S = \Box Q(T),$$

where $T =$ canonical solution for $S$ under $M$.

What is $\Box Q(T)$?

- $T$ may contain incomplete information in the form of nulls

---

Example

\[ a \rightarrow \perp \]
Query Evaluation under the CWA-Semantics

**Theorem (Libkin ’06)**

For every schema mapping \( M \) defined by s-t tgds, every source instance \( S \), and every query \( Q \),

\[
\text{CWA-certain answers to } Q \text{ on } M \text{ and } S = \Box Q(T),
\]

where \( T = \text{canonical solution for } S \text{ under } M \).

**What is \( \Box Q(T) \)?**

- \( T \) may contain incomplete information in the form of nulls
- Possible worlds of \( T \): instances arising from \( T \) by assigning constants to nulls

**Example**

\[\text{a} \quad \rightarrow \quad \bot\]
### Theorem (Libkin ’06)

For every schema mapping $M$ defined by $s$-$t$ tgds, every source instance $S$, and every query $Q$,

$$CWA\text{-}certain \text{ answers to } Q \text{ on } M \text{ and } S = \Box Q(T),$$

where $T =$ canonical solution for $S$ under $M$.

### What is $\Box Q(T)$?

- $T$ may contain incomplete information in the form of nulls.
- Possible worlds of $T$: instances arising from $T$ by assigning constants to nulls.

### Example

Possible worlds:

1. $a$
Query Evaluation under the CWA-Semantics

**Theorem (Libkin ’06)**

For every schema mapping $M$ defined by $s$-$t$ tgds, every source instance $S$, and every query $Q$,

$$CWA$$-certain answers to $Q$ on $M$ and $S = \Box Q(T),$$

where $T =$ canonical solution for $S$ under $M$.

### What is $\Box Q(T)$?

- $T$ may contain incomplete information in the form of nulls
- Possible worlds of $T$: instances arising from $T$ by assigning constants to nulls

### Example

possible worlds:

- $a$,$\perp$
- $a$,$a$,$b$
Query Evaluation under the CWA-Semantics

**Theorem (Libkin '06)**

For every schema mapping $M$ defined by s-t tgds, every source instance $S$, and every query $Q$,

$$\text{CWA-certain answers to } Q \text{ on } M \text{ and } S = □Q(T),$$

where $T = \text{canonical solution for } S \text{ under } M$.

**What is □Q(T)?**

- $T$ may contain incomplete information in the form of nulls
- Possible worlds of $T$: instances arising from $T$ by assigning constants to nulls

**Example**

![Diagram](image_url)

Possible worlds:

- $a$
- $a \rightarrow b$
- $a \rightarrow c$
- $\ldots$
Query Evaluation under the CWA-Semantics

**Theorem (Libkin ’06)**

For every schema mapping $M$ defined by $s$-$t$ tgds, every source instance $S$, and every query $Q$,

\[ CWA\text{-certain answers to } Q \text{ on } M \text{ and } S = \Box Q(T), \]

where $T = \text{canonical solution for } S \text{ under } M$.

What is $\Box Q(T)$?

- $T$ may contain **incomplete information** in the form of nulls
- **Possible worlds of** $T$: instances arising from $T$ by assigning constants to nulls
- $\Box Q(T)$: the certain answers to $Q$ over the possible worlds of $T$

**Example**

![Diagram](https://example.com/diagram.png)

**Possible worlds:**

- $a$, $a \xrightarrow{} b$, $a \xrightarrow{} c$,
- $\ldots$
Generalization and Restriction of the CWA-Semantics

Modifications of the CWA-semantics
(both for schema mappings defined by s-t tgds only):

• “Mixed world” semantics (Libkin, Sirangelo ’08)

• Endomorphic images semantics (Afrati, Kolaitis ’08)
Generalization and Restriction of the CWA-Semantics

Modifications of the CWA-semantics (both for schema mappings defined by s-t tgds only):

• **“Mixed world” semantics** (Libkin, Sirangelo ’08)
  • based on generalized notion of possible worlds of an instance
  • generalized constraint language (annotated s-t tgds)

• **Endomorphic images semantics** (Afrati, Kolaitis ’08)
Generalization and Restriction of the CWA-Semantics

Modifications of the CWA-semantics (both for schema mappings defined by s-t tgds only):

• "Mixed world" semantics (Libkin, Sirangelo ’08)
  - based on generalized notion of possible worlds of an instance
  - generalized constraint language (annotated s-t tgds)

• Endomorphic images semantics (Afrati, Kolaitis ’08)
  - based on restricted notion of possible worlds of an instance
  - shown to be suitable for special aggregate queries
Two natural properties are “missing”:

1. Invariance under logically equivalent schema mappings
2. Reflection of “standard semantics” of constraints
Two natural properties are “missing”:

1. Invariance under logically equivalent schema mappings
2. Reflection of “standard semantics” of constraints
Reflection of “Standard Semantics” of Constraints

Example

Schema mapping:

$$\forall x \left( P(x) \rightarrow \exists y \, E(x, y) \right)$$

Source instance: $$S = \{P(a)\}$$

Unique CWA-solution:

![Diagram showing a directed graph with a node labeled 'a' and another labeled '⊥' connected by an arrow.]
Reflection of “Standard Semantics” of Constraints

Example

Schema mapping:
\[ \forall x ( P(x) \rightarrow \exists y E(x, y) ) \]

Source instance: \( S = \{ P(a) \} \)

Unique CWA-solution: \( a \rightarrow \bot \)

Example query: \( Q := \) Is there exactly one \( y \) with \( E(a, y) \)?

CWA-answers: yes
**Example**

**Schema mapping:**

\[ \forall x ( P(x) \rightarrow \exists y E(x, y) ) \equiv \forall x ( P(x) \rightarrow \bigvee_{y \in \text{Const}} E(x, y) ) \]

**Source instance:** \( S = \{ P(a) \} \)

**Unique CWA-solution:** ![Diagram](image.png)

**Example query:** \( Q := \text{Is there exactly one } y \text{ with } E(a, y)? \)

**CWA-answers:** yes
Reflection of “Standard Semantics” of Constraints

Example

Schema mapping:

\[ \forall x ( P(x) \rightarrow \exists y \ E(x, y) ) \equiv \forall x ( P(x) \rightarrow \bigvee_{y \in \text{Const}} E(x, y) ) \]

Source instance: \( S = \{ P(a) \} \)

Unique CWA-solution: \[ a \rightarrow \bot \]

Example query: \( Q := \text{Is there exactly one } y \text{ with } E(a, y)? \)

CWA-answers: yes

Desired answer: no
<table>
<thead>
<tr>
<th>Definition (H. ’10, restricted version)</th>
</tr>
</thead>
</table>

1. **GCWA*-solutions:**
   - ground solutions that are *unions of minimal solutions*

2. **GCWA*-answers:**
   - the certain answers over GCWA*-solutions
The GCWA*-Semantics

Definition (H. ’10, restricted version)

1. GCWA*-solutions:
   ground solutions that are unions of minimal solutions
2. GCWA*-answers:
   the certain answers over GCWA*-solutions

- inspired by semantics for deductive databases:
  GCWA (Minker ’82) and EGCWA (Yahya, Henschen ’85)
- invariant under logically equivalent schema mappings
- intuitively: reflects “standard semantics” of constraints
**Example**

**Schema mapping:** \( \forall x ( P(x) \rightarrow \exists y E(x, y) ) \)

**Source instance:** \( S = \{P(a)\} \)

**GCWA* solutions:**

- Union of one minimal solution
- Union of two minimal solutions
- Union of three minimal solutions

**Query:** \( Q := \text{Is there exactly one } y \text{ with } E(a, y)? \)

**GCWA* answers:** no (as desired)
Motivating Example Revisited

**Example**

**Schema mapping:** \( \forall x \ ( P(x) \rightarrow \exists y \ E(x, y) ) \)

**Source instance:** \( S = \{ P(a) \} \)

**GCWA* solutions:** \( \{ \text{union of one minimal solution} \)
Motivating Example Revisited

Example

Schema mapping: \( \forall x ( P(x) \rightarrow \exists y E(x, y) ) \)

Source instance: \( S = \{ P(a) \} \)

GCWA* solutions: \( \{ a \rightarrow b, a \rightarrow c \} \) union of two minimal solutions

Query: \( Q: \) Is there exactly one \( y \) with \( E(a, y) \)?

GCWA* answers: no (as desired)
Motivating Example Revisited

Example

Schema mapping: \( \forall x ( P(x) \rightarrow \exists y E(x, y) ) \)

Source instance: \( S = \{ P(a) \} \)

GCWA* solutions: \{ union of three minimal solutions \}

Query: \( Q := \text{Is there exactly one } y \text{ with } E(a, y) ? \)

GCWA* answers: no (as desired)
Motivating Example Revisited

Example

Schema mapping: $\forall x ( P(x) \rightarrow \exists y E(x, y) )$

Source instance: $S = \{ P(a) \}$

GCWA* solutions:

- Union of one minimal solution
- Union of two minimal solutions
- Union of three minimal solutions

Query: $Q := \text{Is there exactly one } y \text{ with } E(a, y)\text{?}$

GCWA*-answers: no (as desired)
Basic Results

- for monotonic queries: GCWA*-answers = certain answers
  (actually true for almost all of the preceding semantics)
Basic Results

- for monotonic queries: GCWA*-answers = certain answers (actually true for almost all of the preceding semantics)
- There is a simple schema mapping $M$ defined by s-t tgds, and a Boolean CQ $Q$ with one negated atom for which

\[
\text{EVAL}(M, Q)
\]

**Input:** source instance $S$

**Question:** Are the GCWA*-answers to $Q$ on $M$ and $S$ non-empty?

is co-NP-hard

(simple reduction from clique problem)
Basic Results

- for monotonic queries: GCWA*-answers = certain answers (actually true for almost all of the preceding semantics)

- There is a simple schema mapping $M$ defined by s-t tgds, and a Boolean CQ $Q$ with one negated atom for which

  \[
  \text{EVAL}(M, Q)
  \]

  \[
  \text{Input:} \quad \text{source instance } S
  \]

  \[
  \text{Question:} \quad \text{Are the GCWA*-answers to } Q \text{ on } M \text{ and } S \text{ non-empty?}
  \]

  is co-NP-hard

  (simple reduction from clique problem)

- There is a simple schema mapping $M$ defined by s-t tgds, and a Boolean FO query $Q$ for which $\text{EVAL}(M, Q)$ is undecidable.
universal query: FO query of the form $\forall \vec{x} \varphi$, $\varphi$ quantifier-free

**Theorem (H. ’10)**

For every properly restricted schema mapping $M$ and for each universal query $Q$ there is a polynomial time algorithm for:

*Input:* the core solution for some source instance $S$ for $M$

*Output:* the GCWA*-answers to $Q$ on $M$ and $S$
universal query: FO query of the form $\forall \bar{x} \varphi$, $\varphi$ quantifier-free

**Theorem (H. ’10)**

For every properly restricted schema mapping $M$ and for each universal query $Q$ there is a polynomial time algorithm for:

**Input:** the core solution for some source instance $S$ for $M$

**Output:** the GCWA*-answers to $Q$ on $M$ and $S$

**Restriction:** $M$ specified by packed s-t tgds

$$\forall \bar{x} \forall \bar{y} \left( \varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \cdots R(\cdots z \cdots) \land \cdots \land R'(\cdots z \cdots) \cdots \right)$$
universal query: FO query of the form $\forall \vec{x} \varphi$, $\varphi$ quantifier-free

**Theorem (H. '10)**

For every properly restricted schema mapping $M$ and for each universal query $Q$ there is a polynomial time algorithm for:

**Input:** the core solution for some source instance $S$ for $M$

**Output:** the GCWA*-answers to $Q$ on $M$ and $S$

**Restriction:** $M$ specified by packed s-t tgds

$$\forall \vec{x} \forall \vec{y} \left( \varphi(\vec{x}, \vec{y}) \rightarrow \exists \vec{z} \; \cdots R(\cdots z \cdots) \land \cdots \land R'(\cdots z \cdots) \cdots \right)$$

**Recall:** Here the core solution can be computed in polynomial time
Step 1/4: Reduction to Satisfiability Problem

\( M \): schema mapping, defined by packed s-t tgds
\( Q \): universal query (Boolean)

**Input:** source instance \( S \) (for the moment)

**Question:** Are the GCWA*-answers to \( Q \) non-empty?
Step 1/4: Reduction to Satisfiability Problem

\( M \): schema mapping, defined by packed s-t tgds
\( Q \): universal query (Boolean)

Input: source instance \( S \) (for the moment)
Question: Are the GCWA*-answers to \( Q \) non-empty?

- Idea: test whether there is a GCWA*-solution \( T \) with \( T \models \neg Q \)
Step 1/4: Reduction to Satisfiability Problem

\( M \): schema mapping, defined by packed s-t tgds
\( Q \): universal query (Boolean)

**Input:** source instance \( S \) (for the moment)

**Question:** Are the GCWA*-answers to \( Q \) non-empty?

- **Idea:** test whether there is a GCWA*-solution \( T \) with \( T \models \neg Q \)
- **Observation:**

\[
\neg Q \equiv \neg \forall \bar{x} \varphi(\bar{x}) \quad \varphi: \text{ quantifier-free}
\]
Step 1/4: Reduction to Satisfiability Problem

M: schema mapping, defined by packed s-t tgds
Q: universal query (Boolean)

Input: source instance S (for the moment)

Question: Are the GCWA*-answers to Q non-empty?

- Idea: test whether there is a GCWA*-solution T with \( T \models \neg Q \)
- Observation:

\[
\neg Q \equiv \exists \bar{x} \neg \varphi(\bar{x}) \quad \varphi: \text{quantifier-free}
\]
Step 1/4: Reduction to Satisfiability Problem

$M$: schema mapping, defined by packed s-t tgds
$Q$: universal query (Boolean)

Input: source instance $S$ (for the moment)
Question: Are the GCWA*-answers to $Q$ non-empty?

- Idea: test whether there is a GCWA*-solution $T$ with $T \models \neg Q$
- Observation:

$$
\neg Q \equiv \exists \overline{x} \bigvee_{i=1}^{n} \varphi_i(\overline{x}_i) \\
\varphi_i: \text{conjunction of atoms or negated atoms}
$$
Step 1/4: Reduction to Satisfiability Problem

\( M \): schema mapping, defined by packed s-t tgds  
\( Q \): universal query (Boolean)

**Input:** source instance \( S \) (for the moment)  
**Question:** Are the GCWA*-answers to \( Q \) non-empty?

- **Idea:** test whether there is a GCWA*-solution \( T \) with \( T \models \neg Q \)
- **Observation:**

\[
\neg Q \equiv \bigvee_{i=1}^{n} \exists \bar{x}_i \varphi_i(\bar{x}_i) \quad \varphi_i: \text{conjunction of atoms or negated atoms}
\]
**Step 1/4: Reduction to Satisfiability Problem**

\(M\): schema mapping, defined by packed s-t tgds

\(Q\): universal query (Boolean)

**Input:** source instance \(S\) (for the moment)

**Question:** Are the GCWA*-answers to \(Q\) non-empty?

- **Idea:** test whether there is a GCWA*-solution \(T\) with \(T \models \neg Q\)
- **Observation:**
  \[
  \neg Q \equiv \bigvee_{i=1}^{n} \exists \bar{x}_i \, \varphi_i(\bar{x}_i)
  \]
  \(\varphi_i\): conjunction of atoms or negated atoms

- **Remains:** test whether for some \(i\) there is a GCWA*-solution \(T\) for \(S\) with
  \[
  T \models \exists \bar{x}_i \, \varphi_i(\bar{x}_i)
  \]
Step 1/4: Reduction to Satisfiability Problem

\( M \): schema mapping, defined by packed s-t tgd\( s \)
\( Q \): universal query (Boolean)

Input: source instance \( S \) (for the moment)

Question: Are the GCWA*-answers to \( Q \) non-empty?

- Idea: test whether there is a GCWA*-solution \( T \) with \( T \models \neg Q \)

- Observation:
  \[
  \neg Q \equiv \bigvee_{i=1}^{n} \exists \bar{x}_i \ \varphi_i(\bar{x}_i) \quad \varphi_i: \text{conjunction of atoms or negated atoms}
  \]

- Remains: test whether for some \( i \) there is a GCWA*-solution \( T \) for \( S \) with
  \[
  T \models \exists \bar{x}_i \ \varphi_i(\bar{x}_i)
  \]
M: schema mapping, defined by packed s-t tgds
Q: universal query (Boolean)

Input: source instance S (for the moment)

Question: Are the GCWA*-answers to Q non-empty?

• Idea: test whether there is a GCWA*-solution T with $T \models \neg Q$

• Observation:

$$\neg Q \equiv \bigvee_{i=1}^{n} \exists \bar{x}_i \varphi_i(\bar{x}_i) \quad \varphi_i: \text{conjunction of atoms or negated atoms}$$

• Remains: test whether for some i there is a set $\mathcal{T}$ of ground minimal solutions for S with $1 \leq |\mathcal{T}|$ and

$$\bigcup \mathcal{T} \models \exists \bar{x}_i \varphi_i(\bar{x}_i)$$
Step 1/4: Reduction to Satisfiability Problem

\( M \): schema mapping, defined by packed s-t tgds

\( Q \): universal query (Boolean)

**Input:** source instance \( S \) (for the moment)

**Question:** Are the GCWA*-answers to \( Q \) non-empty?

- **Idea:** test whether there is a GCWA*-solution \( T \) with \( T \models \neg Q \)

- **Observation:**
  \[
  \neg Q \equiv \bigvee_{i=1}^{n} \exists \bar{x}_i \varphi_i(\bar{x}_i)
  \]
  \( \varphi_i \): conjunction of atoms or negated atoms

- **Remains:** test whether for some \( i \) there is a set \( T \) of ground minimal solutions for \( S \) with \( 1 \leq |T| \leq |\varphi_i| \) and

  \[
  \bigcup T \models \exists \bar{x}_i \varphi_i(\bar{x}_i)
  \]
Step 2/4: Reformulation in Terms of the Core

Query: $\exists \bar{x} \varphi(\bar{x})$, $\varphi$ conjunction of atoms and neg. atoms

Question: Are there ground minimal solutions $T_1, \ldots, T_{|\varphi|}$ for $S$ with

$$\bigcup_i T_i \models \exists \bar{x} \varphi(\bar{x})?$$
Query: \( \exists \bar{x} \varphi(\bar{x}) \), \( \varphi \) conjunction of atoms and neg. atoms

Question: Are there ground minimal solutions \( T_1, \ldots, T_{|\varphi|} \) for \( S \) with
\[
\bigcup_{i} T_i \models \exists \bar{x} \varphi(\bar{x})
\]?

Lemma

*ground minimal solutions for \( S \) = minimal possible worlds of the core solution for \( S \)*
Step 2/4: Reformulation in Terms of the Core

Query: \( \exists \bar{x} \, \varphi(\bar{x}) \), \( \varphi \) conjunction of atoms and neg. atoms

Question: Are there ground minimal solutions \( T_1, \ldots, T_{|\varphi|} \) for \( S \) with
\[
\bigcup_i T_i \models \exists \bar{x} \, \varphi(\bar{x})
\]?

Lemma

**ground minimal solutions for** \( S \)
- \( = \) **minimal possible worlds of the core solution for** \( S \)

New question: Are there minimal possible worlds \( T_1, \ldots, T_{|\varphi|} \) of the core solution for \( S \) with \( \bigcup_i T_i \models \exists \bar{x} \, \varphi(\bar{x}) \)?
Step 3/4: Find Appropriate Minimal Instances

**Lemma**

*M: schema mapping defined by packed s-t tgds*

*Q: query* $\exists \overline{x} \varphi(\overline{x})$, $\varphi$ conjunction of atoms and negated atoms

There is a polynomial time algorithm for

**Input:** core solution $C$ for some source instance $S$ for $M$

**Question:** Are there minimal possible worlds $T_1, \ldots, T_{|\varphi|}$ of $C$ with $\bigcup_i T_i \models Q$
### Step 3/4: Find Appropriate Minimal Instances

**Lemma**

*M: schema mapping defined by packed s-t tgds
Q: query \( \exists \bar{x} \varphi(\bar{x}) \), \( \varphi \) conjunction of atoms and negated atoms

There is a polynomial time algorithm for

**Input:** core solution \( C \) for some source instance \( S \) for \( M \)

**Question:** Are there minimal possible worlds \( T_1, \ldots, T_{|\varphi|} \) of \( C \) with \( \bigcup_i T_i \models Q \)

### Problems to overcome:

- In general, infinitely many minimal possible worlds of \( C \)

**Solution:** canonical representation
Step 3/4: Find Appropriate Minimal Instances

**Lemma**

*M: schema mapping defined by packed s-t tgds*

*Q: query* \( \exists \bar{x} \varphi(\bar{x}), \varphi \text{ conjunction of atoms and negated atoms} \)

*There is a polynomial time algorithm for*

**Input:** core solution \( C \) for some source instance \( S \) for \( M \)

**Question:** Are there minimal possible worlds \( T_1, \ldots, T_{|\varphi|} \) of \( C \) with \( \bigcup_i T_i \models Q \)

**Problems to overcome:**

- **In general,** infinitely many minimal possible worlds of \( C \)
  - **Solution:** canonical representation
- **Still exponentially many instances**
  - **Solution:** reduce set of instances that need to be considered to polynomial size
Step 4/4: A Special Case

Reduction for special case: given atom $R(\bar{a})$, test whether $R(\bar{a})$ belongs to some minimal instance in $\text{poss}(C)$

1. Key property: number of nulls in atom blocks of $C$ bounded by a constant (Fagin, Kolaitis, Popa ’03)
Step 4/4: A Special Case

Reduction for special case: given atom $R(\bar{a})$, test whether $R(\bar{a})$ belongs to some minimal instance in $\text{poss}(C)$

1. Key property: number of nulls in atom blocks of $C$ bounded by a constant (Fagin, Kolaitis, Popa ’03)

$$C = \{E(a, \perp),
E(b, a)
R(a, \perp, \perp')\}$$

Gaifman graph:

$$\begin{align*}
E(a, \perp) & \quad E(b, a) \\
\mid & \\
R(a, \perp, \perp') & 
\end{align*}$$
Step 4/4: A Special Case

Reduction for special case: given atom $R(\bar{a})$, test whether $R(\bar{a})$ belongs to some minimal instance in $\text{poss}(C)$

1 Key property: number of nulls in atom blocks of $C$ bounded by a constant (Fagin, Kolaitis, Popa '03)

$C = \{E(a, \bot), E(b, a), R(a, \bot, \bot', \bot')\}$

Gaifman graph:

```
   E(a, \bot)       E(b, a)
   \|               \|
R(a, \bot, \bot') R(a, \bot, \bot')
```

atom block 1

atom block 2
Step 4/4: A Special Case

Reduction for special case: given atom \( R(\bar{a}) \), test whether \( R(\bar{a}) \)
belongs to some minimal instance in \( \text{poss}(C) \)

1. **Key property:** number of nulls in atom blocks of \( C \) bounded by a constant (Fagin, Kolaitis, Popa '03)

   \[
   C = \{ E(a, \bot), \\
   E(b, a) \\
   R(a, \bot, \bot') \}
   \]

   **Gaifman graph:**

   - Atom block 1
     - \( E(a, \bot) \)
     - \( R(a, \bot, \bot') \)
   - Atom block 2
     - \( E(b, a) \)

2. **First idea:** use minimal instances arising from atom blocks of \( C \)
   by replacing nulls with constants . . .
Step 4/4: A Special Case

Reduction for special case: given atom $R(\bar{a})$, test whether $R(\bar{a})$ belongs to some minimal instance in $\text{poss}(C)$

1. **Key property:** number of nulls in atom blocks of $C$ bounded by a constant (Fagin, Kolaitis, Popa '03)

   $$C = \{ E(a, \bot), E(b, a), R(a, \bot, \bot') \}$$

   Gaifman graph:
   
   **atom block 1**
   
<table>
<thead>
<tr>
<th>$E(a, \bot)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R(a, \bot, \bot')$</td>
</tr>
</tbody>
</table>
   
   **atom block 2**
   
   | $E(b, a)$ |

2. **First idea:** use minimal instances arising from atom blocks of $C$ by replacing nulls with constants . . . fails
Step 4/4: A Special Case

Reduction for special case: given atom $R(\bar{a})$, test whether $R(\bar{a})$ belongs to some minimal instance in $\text{poss}(C)$

1. **Key property:** number of nulls in **atom blocks** of $C$ bounded by a constant (Fagin, Kolaitis, Popa '03)

   $$C = \{E(a, \bot), \quad \begin{array}{c} E(b, a) \quad E(b, a) \\ R(a, \bot, \bot') \quad \text{atom block 1} \end{array} \}$$

   **Gaifman graph:**

   \[
   \begin{array}{c}
   E(a, \bot) \\
   \quad \mid
   \quad \quad R(a, \bot, \bot')
   \end{array}
   \]

   **atom block 2**

2. **First idea:** use minimal instances arising from atom blocks of $C$ by replacing nulls with constants . . . **fails**

3. **Instead:** consider the cores of images of $C$ under special mappings
Step 4/4: A Special Case

**Reduction for special case:** given atom $R(\bar{a})$, test whether $R(\bar{a})$ belongs to some minimal instance in $\text{poss}(C)$

1. **Key property:** number of nulls in **atom blocks** of $C$ bounded by a constant (Fagin, Kolaitis, Popa '03)

   \[ C = \{E(a, \bot), E(b, a), R(a, \bot, \bot')\} \]

   **Gaifman graph:**

   \[
   \begin{array}{c}
   E(a, \bot) \\
   \downarrow \\
   R(a, \bot, \bot') \\
   \end{array}
   \quad \begin{array}{c}
   E(b, a) \\
   \end{array}
   \]

   atom block 1

   atom block 2

2. **First idea:** use minimal instances arising from atom blocks of $C$ by replacing nulls with constants ... **fails**

3. **Instead:** consider the cores of images of $C$ under special mappings ... here packed s-t tgds come into play
• Widely agreed: for monotonic queries use the certain answers
Summary

- Widely agreed: for monotonic queries use the certain answers
  - answering queries preserved under homomorphisms well understood
Summary

• Widely agreed: for monotonic queries use the certain answers
  • answering queries preserved under homomorphisms well understood
  • few results for more general monotonic queries
Summary

- Widely agreed: for monotonic queries use the certain answers
  - answering queries preserved under homomorphisms well understood
  - few results for more general monotonic queries

- Several semantics for non-monotonic queries
• Widely agreed: for monotonic queries use the certain answers
  • answering queries preserved under homomorphisms well understood
  • few results for more general monotonic queries

• Several semantics for non-monotonic queries
  • based on rules for excluding undesired solutions
Summary

• Widely agreed: for monotonic queries use the certain answers
  • answering queries preserved under homomorphisms well understood
  • few results for more general monotonic queries

• Several semantics for non-monotonic queries
  • based on rules for excluding undesired solutions
  • each reflects a certain intuition about what “not mentioned” by a source instance and schema mapping means
• Widely agreed: for monotonic queries use the certain answers
  • answering queries preserved under homomorphisms well understood
  • few results for more general monotonic queries

• Several semantics for non-monotonic queries
  • based on rules for excluding undesired solutions
  • each reflects a certain intuition about what “not mentioned”
    by a source instance and schema mapping means
  • query evaluation may be hard, is not really understood
Open Problems

Lots of open problems, e.g.:

- When is (non-monotonic) query answering tractable?
Lots of open problems, e.g.:

- When is (non-monotonic) query answering tractable?
- For which queries and schema mappings?
Lots of open problems, e.g.:

- When is (non-monotonic) query answering tractable?
  - For which queries and schema mappings?
  - ...and under which semantics?
Lots of open problems, e.g.:

- When is (non-monotonic) query answering tractable?
  - For which queries and schema mappings?
  - ...and under which semantics?
  - Data complexity? Combined complexity?
Lots of open problems, e.g.:

- When is (non-monotonic) query answering tractable?
  - For which queries and schema mappings?
  - ...and under which semantics?
  - Data complexity? Combined complexity?

- Alternative approaches, e.g., stick with the certain answers, but use richer constraint language
Fagin, Kolaitis, Miller, and Popa. Data exchange: Semantics and query answering. ICDT 2003

Libkin. Data exchange and incomplete information. PODS 2006

H. and Schweikardt. CWA-solutions for data exchange settings with target dependencies. PODS 2007

Libkin and Sirangelo. Data exchange and schema mappings in open and closed worlds. PODS 2008

Afrati and Kolaitis. Answering aggregate queries in data exchange. PODS 2008

H. and Schweikardt. Logic and data exchange: Which solutions are good solutions? In Logic and the Foundations of Game and Decision Theory (LOFT 8), 2008

H. Answering non-monotonic queries in relational data exchange. ICDT 2010