Using Deductive Databases Technology for Approximate Query Answering in Partially Complete Databases

Alvaro Cortés⋄ Marc Denecker⋄
Ofer Arieli* Maurice Bruynooghe⋄

⋄: Katholieke Universiteit Leuven, Belgium
*
*: The Academic College of Tel-Aviv/Yaffo, Israel
Reiter’s Closed World Assumption (1978)

In an arbitrary relational database, the Closed World Assumption (CWA) rules as *false* all those tuples that are not in the database.

**Train Time Table**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brussels Centraal</td>
<td>8:03</td>
</tr>
<tr>
<td>Antwerpen Centraal</td>
<td>8:05</td>
</tr>
<tr>
<td>Ghent St. Pieters</td>
<td>8:13</td>
</tr>
<tr>
<td>Brugge</td>
<td>8:22</td>
</tr>
</tbody>
</table>

Reiter’s CWA allows us to conclude that there is no train to Hasselt at 8:04, for instance.

The CWA applies to stand-alone databases storing complete (and correct) information about the world.
In complete databases

What happens when the database is not complete?

<table>
<thead>
<tr>
<th>Name</th>
<th>Telephone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leen Desmet</td>
<td>6531421</td>
</tr>
<tr>
<td>Leen Desmet</td>
<td>09-23314</td>
</tr>
<tr>
<td>Bart Delvaux</td>
<td>5985625</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bart Delvaux</td>
<td>Computer Sci.</td>
</tr>
<tr>
<td>Leen Desmet</td>
<td>Philosophy</td>
</tr>
<tr>
<td>David Finner</td>
<td>Computer Sci.</td>
</tr>
</tbody>
</table>

This database does not store complete knowledge about collaborators from the Philosophy department.

⇒ The $CWA$ is not the correct approach in this context
An alternative view: Open-World Assumption

The other extreme: The Open-World Assumption (OWA)

- The world can be in any state in which all database atoms are true
- A common approach in data integration systems
- The OWA is often too incomplete and underestimates the knowledge in a database.

How to identify those parts of the database that are complete?

Different approaches were presented to specify that the database is partially complete.

- First approach: [Motro 88]
- We follow the approach of Local Closed-World Assumption of [Levy96] and [CDAB05]

A specification of the “areas” in the real world in which the tables of the database contain all (true) tuples.
Expressing Local Closed-World Assumptions

A *Local Closed-World Assumption* (*LCWA*) is an expression [CDAB05]:

$$\text{LCWA}(P(\bar{x}), \Psi[\bar{x}])$$

Where:

- $\Psi[\bar{x}]$: The window of expertise of the *LCWA*
- $P(\bar{x})$: A database predicate, called the object of the *LCWA*
- $\Psi[\bar{x}]$: The free variables in $\Psi$ are a subset of $\bar{x}$

“For all $\bar{x}$ such that $\Psi$ holds in the *real world*, if $P(\bar{x})$ is true in the real world, then $P(\bar{x})$ appears in the database”

This extends Levy’s formalism of local completeness constraints, which allows only positive conjunctive queries as windows of expertise.
Example of the LCWA

Database knows about all the Telephone numbers of people in the CS (computer Science) department:

\[ \text{LCWA} (\text{Telephone}(x, y), \text{Dept}(x, \text{CS})) \]

- Windows of expertise: \( \text{Dept}(x, \text{CS}) \)
- Object of the \( \text{LCWA} \): \( \text{Telephone}(x, y) \)

"For all persons \( x \) of the department of computer science, all true facts of the form Telephone \( (x, y) \) appear in the database."
Let $D$ a set of ground atoms (a database) and the LCWA expression

$$\theta = \text{LCWA}(P(\overline{x}), \Psi[\overline{x}]),$$

the meaning of $\theta$ given $D$ is

$$\mathcal{M}_D(\theta) = \forall \overline{x} \left( \Psi[\overline{x}] \supset (P(\overline{x}) \equiv (P(\overline{x}) \in P^D)) \right)$$

Where “$P(\overline{x}) \in P^D$” is a shorthand for

$$\bigvee_{\overline{a} \in P^D} (\overline{x} = \overline{a})$$
A locally closed database $\mathcal{D}$ over $\Sigma$ is pair $(D, \mathcal{L})$, where $D$ is a database instance and $\mathcal{L} = \{\theta_1, \ldots, \theta_m\}$ is a finite set of LCWAs.

The semantics of $\mathcal{D}$ is:

$$\mathcal{M}(\mathcal{D}) = A \land \bigwedge_{j=1}^{m} \mathcal{M}_D(\theta_j) \land \text{UNA}(\Sigma) \land \text{DCA}(\Sigma).$$

Where

- $A$: The conjunction of atoms in $D$.
- $\text{UNA}(\Sigma)$: Unique Names Axioms.
- $\text{DCA}(\Sigma)$: Domain Closure Axioms.
Query Answering

We are interested in evaluating queries $Q$ with respect to $\mathcal{M}(\mathfrak{D})$:

- $\bar{t}$ is a certain answer for $Q[\bar{x}]$ in $\mathcal{M}(\mathfrak{D})$, if
  \[ \mathcal{M}(\mathfrak{D}) \models Q[\bar{t}/\bar{x}] . \]

  Set of certain answers: $\text{Cert}_\mathfrak{D}(Q[\bar{x}])$.

- $\bar{t}$ is a possible answer for $Q[\bar{x}]$ in $\mathcal{M}(\mathfrak{D})$, if
  \[ \mathcal{M}(\mathfrak{D}) \cup Q[\bar{t}/\bar{x}] \text{ is satisfiable.} \]

  Set of possible answers: $\text{Poss}_\mathfrak{D}(Q[\bar{x}])$.

Complexity result [CDAB05]:

- Computing $\text{Cert}_\mathfrak{D}(Q[\bar{x}])$ is in co-NP-complete.
- Computing $\text{Poss}_\mathfrak{D}(Q[\bar{x}])$ is in NP-complete.

$\Rightarrow$ Checking whether $\mathcal{M}(\mathfrak{D}) \models Q[\bar{d}]$ is an expensive task.
We present a tractable method for *approximate query answering*

- Approximate answers to queries:
  - Under approximation of *certain answers*.
  - Over approximation of *possible answers*.

What kind of technology? Deductive databases

This strategy allows:

- An *efficient* and *sound* method to compute certain and possible answers.
  - To generalize previous approaches that imposed restrictions on the structure of the database [CDAB-AAAI07].

The compromise:

- The answers are incomplete (but in important cases, complete)
The Approach Using Using Deductive Databases

- Translate $\mathcal{D}$ into a deductive database under the Well-Founded semantics
- Certain or possible queries are translated into the deductive database query
- Translated queries are then evaluated using standard deductive databases inference systems
Constructing the Deductive Database

Given:

- Locally closed database $\mathfrak{D} = (D, \mathcal{L})$ based on vocabulary $\Sigma$.

Define the program vocabulary $\Sigma_{\Pi}$ of a logic program $\Pi$ based on $\Sigma$ as follows:

- Variables and constant symbols are copied from $\Sigma$ to $\Sigma_{\Pi}$;
- for every predicate $P$ in $\Sigma$ introduce symbols in $\Sigma_{\Pi}$.
  - $P^c_+$, $P^c_-:$ Represent what is certainly true/false
  - $P^p_+$, $P^p_-:$ Represent what is possibly true/false
  - $P_{db}$: Represents the atoms of the database
The Deductive Database for Certain Answers

Let $\mathcal{D} = (D, \mathcal{L})$ be a locally closed database based on vocabulary $\Sigma$. The \textit{approximation program} $\Pi_{\mathcal{D}}$ based on $\Sigma_{\Pi}$ is defined as follows:

$$\Pi_{\mathcal{D}} : \begin{cases} 
P_{db}(\bar{a}) \text{ for every atom } P(\bar{a}) \text{ in } D. \\
P^c_+(\bar{x}) \leftarrow P_{db}(\bar{x}). \\
P^c_-(\bar{x}) \leftarrow \neg P_{db}(\bar{x}), \Psi^c_+(\bar{x}). 
\end{cases}$$

$\Psi^c_+(\bar{x})$ is derived from $\Psi_P(\bar{x})$, the window of expertise of predicate $P$ in $\mathcal{L}$, by:

- replacing positive literals $Q(\bar{x})$ by $Q^c_+(\bar{x})$ and
- negative literals $\neg Q(\bar{x})$ by $Q^c_-(\bar{x})$. 


The Deductive Database for Possible Answers

For possible answers, extend the approximation database $\Pi_\Omega$ with the rules

\[
\begin{align*}
P_+^p(x) &\leftarrow \text{not } P_-^c(x). \\
P_-^p(x) &\leftarrow \text{not } P_+^c(x).
\end{align*}
\]

and perform a similar transformation on the query where positive occurrences of an atom $P(x)$ are replace by $P^p_+(x)$ and negative occurrences by $P^p_-(x)$. 
Consider: $\mathcal{L} = \{\text{LCWA}(P(x), Q(x)), \text{LCWA}(Q(x), x = c)\}$, $D = \{P(a), Q(c)\}$, and the query $Q[x] = P(x)$.

$$\Pi_\mathcal{D} : \begin{cases} 
P_{db}(a). & Q_{db}(c). \\
P_c^+(x) \leftarrow P_{db}(x). & Q_c^+(x) \leftarrow Q_{db}(x). \\
P_c^-(x) \leftarrow \neg P_{db}(x), Q_c^+(x). & Q_c^-(x) \leftarrow \neg Q_{db}(x), x = c. \\
P_p^+(x) \leftarrow \neg P_c^-(x). & P_p^-(x) \leftarrow \neg P_c^+(x). \\
Q_p^+(x) \leftarrow \neg Q_c^-(x). & Q_p^-(x) \leftarrow \neg Q_c^+(x). \end{cases}$$

$Cert_{\Pi_\mathcal{D}}(Q[x]) = \{a\}$, $Poss_{\Pi_\mathcal{D}}(Q[x]) = \text{Dom}(\mathcal{D}) - \{c\}$.

$Poss_{\Pi_\mathcal{D}}(Q[x])$ is not domain independent!
Let $\mathcal{D} = (D, \mathcal{L})$ be a locally closed database.

(Soundness) For a query $Q$:

$$\text{Cert}_{\Pi_D}(Q[\overline{x}]) \subseteq \text{Cert}_D(Q[\overline{x}]) \subseteq \text{Poss}_D(Q[\overline{x}]) \subseteq \text{Poss}_{\Pi_D}(Q[\overline{x}])$$

(Completeness) If every window of expertise in $\mathcal{L}$ is a conjunction of literals:

If $Q[\overline{x}]$ is a conjunction of literals, then

$$\text{Cert}_{\Pi_D}(Q[\overline{x}]) = \text{Cert}_D(Q[\overline{x}]).$$

If $Q[\overline{x}]$ is a disjunction of literals, then

$$\text{Poss}_{\Pi_D}(Q[\overline{x}]) = \text{Poss}_D(Q[\overline{x}]).$$
The computation time of $Cert_{\Pi_{\mathcal{D}}}(Q[x])$ and $Poss_{\Pi_{\mathcal{D}}}(Q[x])$ by $\Pi_{\mathcal{D}}$ is polynomial in $|D|$. 
Domain Independent Queries

⇒ Queries that can be answered with the information in the database regardless the underlying domain.

- $Poss_{\Pi_{\mathcal{D}}}(Q[x])$ was not domain independent.
  - $Poss_{\Pi_{\mathcal{D}}}(Q[x]) = Dom(\mathcal{D}) - \{c\}$

- Translation of queries may introduce or delete domain independence. Translation may turn a safe query into an unsafe one or vice versa.

- In locally closed databases, domain independence of queries depend on the form of queries and the form of the LCWAs.
  - Deciding whether a query is domain independent is undecidable.
  - Syntactic restrictions in queries and LCWAS can be imposed to ensure domain independence.
Conclusions and Ongoing Work

Tractable methods for approximate querying in locally closed databases, based on standard deductive database techniques

Deductive databases approach generalizes algorithm based on rewriting techniques

Extensions for future work:

- Integrity constraints.
- Null values.
- Views