Logical Approach to Physical Data Independence and Query Compilation
Classical OBDA and Data Exchange

David Toman

D.R. Cheriton School of Computer Science
University of Waterloo
OBDA AND LITE LOGICS
Setup

Setting

Input:  
1. Schema $\Sigma$ (set of integrity constraints);
2. Data $D = \{R_1, \ldots, R_k\}$ (instance of access paths); and
3. Query $\varphi$ (a formula)
Setup

Setting

Input:  
1. Schema $\Sigma$ (set of integrity constraints);
2. Data $D = \{R_1, \ldots, R_k\}$ (instance of access paths); and
3. Query $\varphi$ (a formula)

Definition (Certain Answers)

$$\text{cert}_{\Sigma, D}(\varphi) = \{\vec{a} \mid \Sigma \cup D \models \varphi(\vec{a})\} = \bigcap_{I \models \Sigma \cup D} \{\vec{a} \mid I \models \varphi(\vec{a})\}$$

Convention: ABox $A$ vs. database $D$

We assume that for every access path $R_{AP}(\vec{x})$ in $D$ there is a logical predicate $R(\vec{x})$ (with the same arity), and a constraint $\forall \vec{x}. R_{AP}(\vec{x}) \rightarrow R(\vec{x})$.
Setup

Setting

Input:  
1. Schema $\Sigma$ (set of integrity constraints);  
2. Data $D = \{R_1, \ldots, R_k\}$ (instance of access paths); and  
3. Query $\varphi$ (a formula)

Definition (Certain Answers)

$$\text{cert}_{\Sigma, D}(\varphi) = \{\overrightarrow{a} | \Sigma \cup D \models \varphi(\overrightarrow{a})\} = \bigcap_{I \models \Sigma \cup D} \{\overrightarrow{a} | I \models \varphi(\overrightarrow{a})\}$$

Convention: ABox $\mathcal{A}$ vs. database $D_\mathcal{A}$

We assume that for every access path $R_{\text{AP}}(\overrightarrow{x})$ in $D_\mathcal{A}$ there is  
1. a logical predicate $R(\overrightarrow{x})$ (with the same arity), and  
2. a constraint $\forall \overrightarrow{x}. R_{\text{AP}}(\overrightarrow{x}) \rightarrow R(\overrightarrow{x})$. 
Can this be Done Efficiently at all?

Question

Can there be a *non-trivial* schema language for which *query answering* (under certain answer semantics) is *tractable* (in data complexity)?
Can this be Done Efficiently at all?

Question

Can there be a non-trivial schema language for which query answering (under certain answer semantics) is tractable (in data complexity)?

YES: Conjunctive queries (or positive) and “lite” Description Logics:

1. The DL-Lite family
   - conjunction, $\bot$, domain/range, unqualified $\exists$, role inverse, UNA
   - certain answers in $AC_0$ for data complexity (maps to SQL)

2. The $\mathcal{EL}$ family
   - conjunction, qualified $\exists$
   - certain answers $PTIME$-complete for data complexity

3. The $\mathcal{CFD}$ family
   - qualified $\forall$ (over total functions), functional dependencies
   - certain answers $PTIME$-complete for data complexity
**Definition (DL-Lite family: Schemata/TBoxes)**

1. **Roles** $R$ and **concepts** $C$ as follows:
   
   $$ R ::= P \mid P^\bot $$
   $$ C ::= \bot \mid A \mid \exists R $$

2. Schemata are represented as TBoxes: a finite set $\mathcal{T}$ of constraints
   
   $$ C_1 \sqcap \cdots \sqcap C_n \sqsubseteq C \quad \quad R_1 \sqsubseteq R_2 $$

Access paths (data) $\Rightarrow$ ABox $\mathcal{A}$ (recall the "convention" about access paths!)
DL-Lite Family of DLs

Definition (DL-Lite family: Schemata/TBoxes)

1. Roles \( R \) and concepts \( C \) as follows:
   \[
   R ::= P \mid P^\perp \quad C ::= \bot \mid A \mid \exists R
   \]

2. Schemata are represented as TBoxes: a finite set \( T \) of constraints
   \[
   C_1 \sqcap \cdots \sqcap C_n \sqsubseteq C \quad R_1 \sqsubseteq R_2
   \]

Access paths (data) \( \Rightarrow \) ABox \( \mathcal{A} \) (recall the “convention” about access paths!)

How to compute answers to CQs?

IDEA: incorporate \textit{schematic knowledge} into the query.
Example

TBox (Schema): $\text{Employee} \sqsubseteq \exists \text{Works}$
$\exists \text{Works}^- \sqsubseteq \text{Project}$

 Conjunctive Query: $\exists y. \text{Works}(x, y) \land \text{Project}(y)$
Example

TBox (Schema):  
\[ \text{Employee} \sqsubseteq \exists \text{Works} \]
\[ \exists \text{Works}^- \sqsubseteq \text{Project} \]

Conjunctive Query:  
\[ \exists y. \text{Works}(x, y) \land \text{Project}(y) \]

Rewriting:

\[ Q^\dagger = (\exists y. \text{Works}(x, y) \land \text{Project}(y)) \lor \]

Query Execution:

\[ Q^\dagger(\{ \text{Employee}(\text{bob}), \text{Works}(\text{sue}, \text{slides}) \}) = \{ \text{bob}, \text{sue} \} \]
Example

TBox (Schema): \( \text{Employee} \sqsubseteq \exists \text{Works} \)
\( \exists \text{Works}^- \sqsubseteq \text{Project} \)

Conjunctive Query: \( \exists y. \text{Works}(x, y) \land \text{Project}(y) \)

Rewriting:

\[
Q^\dagger = (\exists y. \text{Works}(x, y) \land \text{Project}(y)) \lor \\
(\exists y, z. \text{Works}(x, y) \land \text{Works}(z, y)) \lor
\]
Example

TBox (Schema):  
\[ \text{Employee} \sqsubseteq \exists \text{Works} \]
\[ \exists \text{Works}^- \sqsubseteq \text{Project} \]

Conjunctive Query:  
\[ \exists y. \text{Works}(x, y) \land \text{Project}(y) \]

Rewriting:
\[ Q^\dagger = (\exists y. \text{Works}(x, y) \land \text{Project}(y)) \lor \\
(\exists y, z. \text{Works}(x, y) \land \text{Works}(z, y)) \lor \\
(\exists y. \text{Works}(x, y)) \lor \]

Query Execution:
\[ Q^\dagger(\{\text{Employee}(\text{bob}), \text{Works}(\text{sue}, \text{slides})\}) = \{\text{bob}, \text{sue}\} \]
Example

TBox (Schema):

\[ \text{Employee} \sqsubseteq \exists \text{Works} \]
\[ \exists \text{Works}^- \sqsubseteq \text{Project} \]

Conjunctive Query:

\[ \exists y. \text{Works}(x, y) \land \text{Project}(y) \]

Rewriting:

\[ Q^\dagger = (\exists y. \text{Works}(x, y) \land \text{Project}(y)) \lor \]
\[ (\exists y, z. \text{Works}(x, y) \land \text{Works}(z, y)) \lor \]
\[ (\exists y. \text{Works}(x, y)) \lor \]
\[ (\text{Employee}(x)) \]
Example

TBox (Schema):

\[ \text{Employee} \sqsubseteq \exists \text{Works} \]
\[ \exists \text{Works}^\neg \sqsubseteq \text{Project} \]

 Conjunctive Query: \[ \exists y. \text{Works}(x, y) \land \text{Project}(y) \]

Rewriting:

\[ Q^\dagger = (\exists y. \text{Works}_{AP}(x, y) \land \text{Project}_{AP}(y)) \lor \]
\[ (\exists y, z. \text{Works}_{AP}(x, y) \land \text{Works}_{AP}(z, y)) \lor \]
\[ (\exists y. \text{Works}_{AP}(x, y)) \lor \]
\[ (\text{Employee}_{AP}(x)) \]

Query Execution:

\[ Q^\dagger \left( \{ \text{Employee}(\text{bob}), \right. \]
\[ \left. \text{Works}(\text{sue, slides}) \} \right) \]
Example

TBox (Schema):  

\[ \text{Employee} \sqsubseteq \exists \text{Works} \]
\[ \exists \text{Works}^- \sqsubseteq \text{Project} \]

Conjunctive Query:  

\[ \exists y. \text{Works}(x, y) \land \text{Project}(y) \]

Rewriting:

\[ Q^\dagger = (\exists y. \text{Works}_{AP}(x, y) \land \text{Project}_{AP}(y)) \lor \\
(\exists y, z. \text{Works}_{AP}(x, y) \land \text{Works}_{AP}(z, y)) \lor \\
(\exists y. \text{Works}_{AP}(x, y)) \lor \\
(\text{Employee}_{AP}(x)) \]

Query Execution:

\[ Q^\dagger \left( \left\{ \text{Employee}(bob), \text{Works}(sue, slides) \right\} \right) = \{bob, sue\} \]
Input: Conjunctive query $Q$, DL-Lite TBox $\Sigma$

$R = \{Q\}$;

repeat

\hspace{1em} foreach query $Q' \in R$ do

\hspace{2em} foreach axiom $\alpha \in \Sigma$ do

\hspace{3em} if $\alpha$ is applicable to $Q'$ then

\hspace{4em} $R = R \cup \{Q'[\text{lhs}(\alpha)/\text{rhs}(\alpha)]\}$

\hspace{2em} foreach two atoms $D_1, D_2$ in $Q'$ do

\hspace{3em} if $D_1$ and $D_2$ unify then

\hspace{4em} $\sigma = \text{MGU}(D_1, D_2)$; $R = R \cup \{\lambda(Q', \sigma)\}$;

until no query unique up to variable renaming can be added to $R$;

return $Q^{\dagger} := (\bigvee R)$
Input: Conjunctive query \( Q \), DL-Lite TBox \( \Sigma \)

\[ R = \{ Q \}; \]
repeat
  \textbf{foreach} query \( Q' \in R \) do
    \textbf{foreach} axiom \( \alpha \in \Sigma \) do
      \textbf{if} \( \alpha \) is applicable to \( Q' \) \textbf{then}
      \[ R = R \cup \{ Q'[\text{lhs}(\alpha)/\text{rhs}(\alpha)] \} \]
    \textbf{foreach} two atoms \( D_1, D_2 \) in \( Q' \) do
      \textbf{if} \( D_1 \) and \( D_2 \) unify \textbf{then}
      \[ \sigma = \text{MGU}(D_1, D_2); R = R \cup \{ \lambda(Q', \sigma) \}; \]
  until no query unique up to variable renaming can be added to \( R \);
return \( Q^\dagger := (\bigvee R) \)

Theorem

\[ \Sigma \cup \mathcal{A} \models Q(\vec{a}) \text{ if and only if } D_{\mathcal{A}} \models Q^\dagger(\vec{a}) \]
QuOnto: Rewriting Approach [Calvanese et al.]

**Input:** Conjunctive query $Q$, DL-Lite TBox $\Sigma$

$R = \{Q\}$;

repeat

- foreach query $Q' \in R$ do
  - foreach axiom $\alpha \in \Sigma$ do
    - if $\alpha$ is applicable to $Q'$ then
      - $R = R \cup \{Q'[\text{lhs}(\alpha)/\text{rhs}(\alpha)]\}$
  - foreach two atoms $D_1, D_2$ in $Q'$ do
    - if $D_1$ and $D_2$ unify then
      - $\sigma = \text{MGU}(D_1, D_2)$; $R = R \cup \{\lambda(Q', \sigma)\}$

until no query unique up to variable renaming can be added to $R$;

return $Q^\dagger := (\bigvee R)$

**Theorem**

$\Sigma \cup \mathcal{A} \models Q(\bar{a})$ if and only if $D_\mathcal{A} \models Q^\dagger(\bar{a}) \iff$ can be VERY large
**Definition (\(\mathcal{EL}\)-Lite family: Schemata and TBoxes)**

1. **Concepts** \(C\) as follows:
   \[
   C ::= A \mid \top \mid \bot \mid C \sqcap C \mid \exists R.C
   \]

2. Schemata are represented as TBoxes: a finite set \(\mathcal{T}\) of constraints
   \[
   C_1 \sqsubseteq C_2 \quad R_1 \sqsubseteq R_2
   \]

Access paths (data) \(\Rightarrow\) ABox \(\mathcal{A}\) (recall the “convention” about access paths!)
The EL Family of DLs

Definition (\(\mathcal{EL}\)-Lite family: Schemata and TBoxes)

1. **Concepts** \( C \) as follows:
   \[
   C ::= A \mid \top \mid \bot \mid C \sqcap C \mid \exists R.C
   \]

2. Schemata are represented as TBoxes: a finite set \( \mathcal{T} \) of constraints
   \[
   C_1 \sqsubseteq C_2 \quad R_1 \sqsubseteq R_2
   \]

Access paths (data) \( \Rightarrow \) ABox \( \mathcal{A} \) (recall the “convention” about access paths!)

How to compute answers to CQs?

IDEA: incorporate *schematic knowledge* into the data.
Combined Approach

Can an approach based on *rewriting* be used for $\mathcal{EL}$?
Combined Approach

Can an approach based on *rewriting* be used for $\mathcal{EL}$?

NO: $\mathcal{EL}$ is PTIME-complete (data complexity).
Combined Approach

Can an approach based on rewriting be used for $\mathcal{EL}$?

NO: $\mathcal{EL}$ is PTIME-complete (data complexity).

Combined Approach

We effectively transform

1. the ABox (access paths) $\mathcal{A}$ to a canonical structure $D_\mathcal{A}^*$ utilizing $\Sigma$,
2. the conjunctive query $Q$ to a relational query $Q^\dagger$.

...both polynomial in the input(s).
Can an approach based on rewriting be used for $\mathcal{EL}$?

NO: $\mathcal{EL}$ is PTIME-complete (data complexity).

Combined Approach

We effectively transform

1. the ABox (access paths) $\mathcal{A}$ to a canonical structure $D^*_\mathcal{A}$ utilizing $\Sigma$,
2. the conjunctive query $Q$ to a relational query $Q^\dagger$.

...both polynomial in the input(s).

Theorem (Lutz, __, Wolter: IJCAI’09)

$\Sigma \cup \mathcal{A} \models Q(\vec{a}) \text{ if and only if } D^*_\mathcal{A} \models Q^\dagger(\vec{a})$
Example (with almost DL-Lite schema)

TBox (Schema):  
\[ Employee \sqsubseteq \exists Works. Project \]
\[ \exists Works. \top \sqsubseteq \exists Works. Project \]

Conjunctive Query:  
\[ \exists y. Works(x, y) \land Project(y) \]

Data:  
\{ Employee(bob), Works(sue, slides) \}
Example (with almost DL-Lite schema)

TBox (Schema): \( Employee \sqsubseteq \exists Works. Project \)
\( \exists Works. \top \sqsubseteq \exists Works. Project \)

Conjunctive Query: \( \exists y. Works(x, y) \land Project(y) \)

Data: \( \{ Employee(bob), Works(sue, slides) \} \)

Rewriting:

1. \( D_A^* = \{ Employee(bob), Works(bob, c_{Works}), Works(sue, slides), Works(sue, c_{Works}), Project(c_{Works}), \} \)
2. \( Q^\dagger = Q \land (x \neq c_{Works}) \)
Example (with almost DL-Lite schema)

TBox (Schema): $Employee \sqsubseteq \exists Works.\, Project$
               $\exists Works. \top \sqsubseteq \exists Works.\, Project$

Conjunctive Query: $\exists y.\, Works(x, y) \land Project(y)$

Data: $\{\text{Employee(bob), Works(sue, slides)}\}$

Rewriting:

1. $D^*_A = \{\text{Employee(bob), Works(bob, } c_{\text{Works}}),$
               $\text{Works(sue, slides), Works(sue, } c_{\text{Works}}),\, Project(\text{ } c_{\text{Works}}),\} $

2. $Q^\dagger = Q \land (x \neq c_{\text{Works}})$

Query Execution:

$Q^\dagger(D^*_A) = \{\text{bob, sue}\}$
A Combined Approach and DL-Lite

Can the *exponential size* of rewriting be avoided for DL-Lite?
A Combined Approach and DL-Lite

Can the *exponential size* of rewriting be avoided for DL-Lite?

**Yes: using the Combined Approach**

... but query rewriting is much more involved due to *inverse roles*.
A Combined Approach and DL-Lite

Can the *exponential size* of rewriting be avoided for DL-Lite?

Yes: using the Combined Approach

... but query rewriting is much more involved due to *inverse roles*;

Theorem (Konchatov, Lutz, _, Wolter, KR10)

$$\Sigma \cup A \models Q(\vec{a}) \text{ if and only if } D^{\ast}_A \models Q^{\dagger}(\vec{a})$$

(... still exponential for *role hierarchies.*)
A Combined Approach and DL-Lite

Can the *exponential size* of rewriting be avoided for DL-Lite?

**Yes: using the Combined Approach**

...but query rewriting is much more involved due to *inverse roles*;

**Theorem (Konchatov, Lutz, _, Wolter, KR10)**

\[ \Sigma \cup A \models Q(\vec{a}) \text{ if and only if } D^*_A \models Q^\dagger(\vec{a}) \]

(... still exponential for *role hierarchies*.)

**Theorem (Lutz, Seylan,_,Wolter, ISWC13)**

\[ \Sigma \cup A \models Q(\vec{a}) \text{ if and only if } D^*_A \models Q^{\text{filter}}(\vec{a}) \]

(... polynomial in \(|\mathcal{H}|\), but uses UDF feature of DB2.)
**CFD family of Logics**

**Definition (CFD\textsubscript{nc}: Schemata and TBoxes)**

1. Syntax formed from *path functions* \( \text{Pf} \) and *concepts* \( C, D \) as follows:
   \[
   C ::= A \mid \forall \text{Pf}.C \\
   D ::= A \mid \neg C \mid \forall \text{Pf}.C \mid C : \text{Pf}_1, \ldots, \text{Pf}_k \rightarrow \text{Pf}
   \]

2. Schemata are represented as a TBox:
   - finite set \( \mathcal{T} \) of *constraints* \( C \sqsubseteq D \).

3. Data is represented as an ABox (recall again the AP “convention”):
   - finite set \( \mathcal{A} \) of *concept* \( A(a) \) and *equational* \( \text{Pf}(a) = \text{Pf}'(b) \) assertions.
**Definition (CFD\textsubscript{nc}: Schemata and TBoxes)**

1. Syntax formed from *path functions* Pf and *concepts* C, D as follows:
   
   \[C ::= A \mid \forall Pf . C\]
   
   \[D ::= A \mid \neg C \mid \forall Pf . C \mid C : Pf_1, \ldots, Pf_k \rightarrow Pf\]

2. Schemata are represented as a TBox:
   
   finite set \(T\) of constraints \(C \sqsubseteq D\).

3. Data is represented as an ABox (recall again the AP “convention”):
   
   finite set \(A\) of *concept* \(A(a)\) and *equational* \((Pf(a) = Pf'(b))\) assertions.
**Definition (CFD\textsubscript{nc}: Schemata and TBoxes)**

1. Syntax formed from *path functions* \( Pf \) and *concepts* \( C, D \) as follows:
   
   \[
   C ::= A | \forall Pf . C \\
   D ::= A | \neg C | \forall Pf . C | C : Pf_1, \ldots, Pf_k \rightarrow Pf
   \]

2. Schemata are represented as a TBox:
   
   finite set \( T \) of *constraints* \( C \subseteq D \).

3. Data is represented as an ABox (recall again the AP “convention”):
   
   finite set \( A \) of *concept* \( A(a) \) and *equational* \( Pf(a) = Pf'(b) \) assertions.

Rewriting Approach: can’t work—reachability in ABox (PTIME-c)

Combined Approach: can’t work—too many *types* (anon. completion too big)
CFD family of Logics

Definition ($\mathcal{CFD}_{nc}$: Schemata and TBoxes)

1. Syntax formed from path functions $Pf$ and concepts $C, D$ as follows:
   \[
   C ::= A \mid \forall Pf.C \\
   D ::= A \mid \neg C \mid \forall Pf.C \mid C : Pf_1, \ldots, Pf_k \rightarrow Pf
   \]

2. Schemata are represented as a TBox:
   - finite set $T$ of constraints $C \sqsubseteq D$.

3. Data is represented as an ABox (recall again the AP “convention”):
   - finite set $A$ of concept $(A(a))$ and equational $(Pf(a) = Pf'(b))$ assertions.

Query Answering: The Perfect Combined Approach

IDEA: incorporate
- reachability induced by schematic knowledge into the data, and
- types induced by schematic knowledge into the query.
DATA EXCHANGE
Setup

Schema Mapping

- source schema (signature) \( S_P \) and (closed) data;
- target schema (signature) \( S_L \);
- mapping constraints: \( s-t \) TGDs—formulas of the form
  \[ \forall \vec{x}.\varphi(\vec{x}) \rightarrow \exists \vec{y}.\psi(\vec{x}, \vec{y}) \text{ where } \varphi \text{ is a CQ over } S_P \text{ and } \psi \text{ a CQ over } S_L. \]

The general setting of data exchange is this:

[Source \( S \) \( \rightarrow \) Mapping \( M \) \( \rightarrow \) Target \( T \) \( \rightarrow \) Query \( Q \)]

[Arenas et al: Foundations of Data Exchange]
Setup

**Schema Mapping**
- source schema (signature) $S_P$ and (closed) data;
- target schema (signature) $S_L$;
- mapping constraints: *s-t TGDs*—formulas of the form
  \[ \forall \vec{x}. \varphi(\vec{x}) \rightarrow \exists \vec{y}. \psi(\vec{x}, \vec{y}) \]
  where $\varphi$ is a CQ over $S_P$ and $\psi$ a CQ over $S_L$.

[The general setting of data exchange is this:]

[Diagram: source $S$ mapping $M$ to target $T$ with query $Q$]

**Definition**

$J$ (over $S_L$) is a *solution* for $I$ (over $S_P$) w.r.t. $\Sigma$ if $(I, J) \models \Sigma$.

... too many solutions (TGDs imply open world $\forall S_L$!)
Universal Solutions and Cores

Problem(s):

Multiple \textit{solutions} (target instances) for single \textit{closed world} source
\[ \Rightarrow \text{how to answer queries over target? \textit{certain answers} w.r.t. all solutions.} \]
Universal Solutions and Cores

Problem(s):
Multiple solutions (target instances) for single closed world source
⇒ how to answer queries over target? certain answers w.r.t. all solutions.

IDEA:
Find the best solution: one that can be used instead of every other solution.
Problem(s):

Multiple *solutions* (target instances) for single *closed world* source

⇒ how to answer queries over target? *certain answers* w.r.t. all solutions.

IDEA:

Find the *best* solution: one that can be used instead of every other solution.

- an *universal solution*: homomorphism to all other solutions
## Universal Solutions and Cores

### Problem(s):

Multiple *solutions* (target instances) for single *closed world* source

⇒ how to answer queries over target? *certain answers* w.r.t. all solutions.

### IDEA:

Find the *best* solution: one that can be used instead of every other solution.

- an *universal solution*: homomorphism to all other solutions

⇒ variables (marked nulls): *representation system* [Imielinski&Lipski’84]
Universal Solutions and Cores

Problem(s):
Multiple solutions (target instances) for single closed world source
⇒ how to answer queries over target? certain answers w.r.t. all solutions.

IDEA:
Find the best solution: one that can be used instead of every other solution.

- an universal solution: homomorphism to all other solutions
  ⇒ variables (marked nulls): representation system [Imielinski&Lipski’84]
  ⇒ can be used to answer CQ/UCQ (how and why?)
Universal Solutions and Cores

Problem(s):
Multiple solutions (target instances) for single closed world source
⇒ how to answer queries over target? certain answers w.r.t. all solutions.

IDEA:
Find the best solution: one that can be used instead of every other solution.
  • an universal solution: homomorphism to all other solutions
    ⇒ variables (marked nulls): representation system [Imielinski&Lipski’84]
    ⇒ can be used to answer CQ/UCQ (how and why?)
  • a smallest universal solution—the core.
Problem(s):
Multiple solutions (target instances) for single closed world source
⇒ how to answer queries over target? certain answers w.r.t. all solutions.

IDEA:
Find the best solution: one that can be used instead of every other solution.
- an universal solution: homomorphism to all other solutions
  ⇒ variables (marked nulls): representation system [Imielinski&Lipski’84]
  ⇒ can be used to answer CQ/UCQ (how and why?)
- a smallest universal solution—the core.

core can be constructed using the chase (in PTIME);
Universal Solutions and Cores

Problem(s):

Multiple *solutions* (target instances) for single *closed world* source
⇒ how to answer queries over target? *certain answers* w.r.t. all solutions.

IDEA:

Find the *best* solution: one that can be used instead of every other solution.
- an *universal solution*: homomorphism to all other solutions
  ⇒ variables (marked nulls): *representation system* [Imielinski&Lipski’84]
  ⇒ can be used to answer CQ/UCQ (how and why?)
- a smallest universal solution—the *core*.

- core can be constructed using the *chase* (in PTIME);
- what happens if we have additional constraints on the target (S_L)?
LIMITS AND ISSUES WITH POSSIBLE WORLDS
Certain Answers: What is the Price?

High Computational Cost even for mild deviation from Lite Logics (and CQ) 
\[ \text{coNP-hard for DATA COMPLEXITY} \]

Example

- **Schema & Data:**
  \[
  \Sigma = \{ \forall x, y. \text{ColNode}(x, y) \iff \text{Node}(x), \\
  \forall x, y. \text{ColNode}(x, y) \iff \text{Colour}(y) \}
  \]
  \[
  D = \{ \text{Edge} = \{(n_i, n_j)\}, \text{Node} = \{n_1, \ldots n_m\}, \\
  \text{Colour} = \{r, g, b\} \}
  \]
High Computational Cost even for mild deviation from *Lite* Logics (and CQ)

\textit{coNP-hard} for \textit{DATA COMPLEXITY}

**Example**

- **Schema&Data:**
  \[
  \Sigma = \{ \forall x, y. \text{ColNode}(x, y) \leftrightarrow \text{Node}(x), \\
  \forall x, y. \text{ColNode}(x, y) \leftrightarrow \text{Colour}(y) \} \\
  D = \{ \text{Edge} = \{(n_i, n_j)\}, \text{Node} = \{n_1, \ldots n_m\}, \\
  \text{Colour} = \{r, g, b\} \}
  \]

- **Query:** \( \exists x, y, c. \text{Edge}(x, y) \land \text{ColNode}(x, c) \land \text{ColNode}(y, c) \)
High Computational Cost even for mild deviation from *Lite Logics* (and CQ)

coNP-hard for *DATA COMPLEXITY*

Example

- **Schema & Data:**
  \[
  \Sigma = \{ \forall x, y. \text{ColNode}(x, y) \leftrightarrow \text{Node}(x), \\
  \forall x, y. \text{ColNode}(x, y) \leftrightarrow \text{Colour}(y) \}
  \]
  \[
  D = \{ \text{Edge} = \{(n_i, n_j)\}, \text{Node} = \{n_1, \ldots, n_m\}, \\
  \text{Colour} = \{r, g, b\} \}
  \]

- **Query:**
  \[
  \exists x, y, c. \text{Edge}(x, y) \land \text{ColNode}(x, c) \land \text{ColNode}(y, c) \\
  \Rightarrow \text{the graph } (\text{Node}, \text{Edge}) \text{ is NOT 3-colourable.}
  \]
Certain Answers: What is the Price?

High Computational Cost even for mild deviation from Lite Logics (and CQ) 

\[ \text{coNP-hard for DATA COMPLEXITY} \]

Example

- **Schema&Data:**
  \[ \Sigma = \{ \forall x, y. \text{ColNode}(x, y) \leftrightarrow \text{Node}(x), \forall x, y. \text{ColNode}(x, y) \leftrightarrow \text{Colour}(y) \} \]
  \[ D = \{ \text{Edge} = \{(n_i, n_j)\}, \text{Node} = \{n_1, \ldots n_m\}, \text{Colour} = \{r, g, b\} \} \]

- **Query:** \( \exists x, y, c. \text{Edge}(x, y) \land \text{ColNode}(x, c) \land \text{ColNode}(y, c) \)

\[ \Rightarrow \text{the graph (Node, Edge) is NOT 3-colourable.} \]

... coNP-complete for all DLs between \( \mathcal{AL} \) and \( \mathcal{SHIQ} \).
Certain Answers: What is the Price?

High Computational Cost even for mild deviation from Lite Logics (and CQ)

*coNP-hard* for DATA COMPLEXITY

### Example

**Schema & Data:**

\[
\Sigma = \{ \forall x, y. \text{ColNode}(x, y) \leftrightarrow \text{Node}(x), \\
\forall x, y. \text{ColNode}(x, y) \leftrightarrow \text{Colour}(y) \} \\
D = \{ \text{Edge} = \{(n_i, n_j)\}, \text{Node} = \{n_1, \ldots, n_m\}, \\
\text{Colour} = \{r, g, b\} \} \\
\]

**Query:** \( \exists x, y, c. \text{Edge}(x, y) \wedge \text{ColNode}(x, c) \wedge \text{ColNode}(y, c) \)

\( \Rightarrow \) the graph \((\text{Node, Edge})\) is NOT 3-colourable.

\( \ldots \) coNP-complete for all DLs between \( \mathcal{AL} \) and \( \mathcal{SHIQ} \).

OBDA-Lite can only say \( \text{Colour} \supseteq \{r, g, b\} \) (due to OWA)

Data Exchange cannot say \( \forall x, y. \text{ColNode}(x, y) \rightarrow \text{Colour}(y) \) (not an s-t TGD)
Certain Answers: What about more complex Queries?

(safe) Negation, Inequality

Theorem (Gutíerrez-Basulto et al., RR13)

OBDA for CQ with single inequality or with safe negated atoms over DL-Lite$^H$ is undecidable.

Aggregation

⇒ **count/sum** aggregate functions do not play nicely with *certain answers*
  - epistemic operators (count the number of *known* answers) [Calvanese et al., ONISW08]
  - range/lower bounds semantics (at least so many) [Kostylev and Reutter, AAAI13]

... and it is (data complexity-wise) hard in all cases.
Example (Unintuitive Behaviour of Queries:)

1. $\exists x. \text{Phone}("John", x)$?
2. $\text{Phone}("John", x)$?

under $\Sigma = \{ \forall x. \text{Person}(x) \rightarrow \exists y. \text{Phone}(x, y) \}$
and $D = \{ \text{Person}("John") \}$. 

Limits and Issues with Possible Worlds
Certain Answers??

Example (Unintuitive Behaviour of Queries:)

1. \( \exists x. \text{Phone}("John", x) \)?
2. \( \text{Phone}("John", x) \)\

under \( \Sigma = \{ \forall x. \text{Person}(x) \rightarrow \exists y. \text{Phone}(x, y) \} \)
and \( D = \{ \text{Person}("John") \} \).

Embedded SQL-like Example

if "\( \exists x. \text{Phone}("John", x) \)" then
    begin
        x := "\text{Phone}("John", x)";
        print "John’s phone number is:" x
    end
Example (Unintuitive Behaviour of Queries:)

1. $\exists x. \text{Phone}(\text{"John"}, x)$? $\Rightarrow$ YES
2. $\text{Phone}(\text{"John"}, x)$? $\Rightarrow \{\}$

under $\Sigma = \{\forall x. \text{Person}(x) \rightarrow \exists y. \text{Phone}(x, y)\}$ and $D = \{\text{Person}(\text{"John"})\}$.

Embedded SQL-like Example

if $\exists x. \text{Phone}(\text{"John"}, x)$ then
begin
  x := "\text{Phone}(\text{"John"}, x)";
  print "John’s phone number is:"
end
Example (Unintuitive Behaviour of Queries:)

1. $\exists x. \text{Phone}(\text{"John"}, x) \Rightarrow \text{YES}$
2. $\text{Phone}(\text{"John"}, x) \Rightarrow \emptyset$

under $\Sigma = \{ \forall x. \text{Person}(x) \rightarrow \exists y. \text{Phone}(x, y) \}$
and $D = \{ \text{Person}(\text{"John"}) \}$.

Embedded SQL-like Example

if $\exists x. \text{Phone}(\text{"John"}, x)$ then
begin
\hspace{1em} x := "\text{Phone}(\text{"John"}, x)";
\hspace{1em} \text{print } \text{"John’s phone number is:" } x$
end
certain answers are tractable only for \textit{Lite schemata} and \textit{Conjunctive/UC Queries}

pretty much any extension leads to complexity (decidability) issues
certain answers are tractable only for *Lite schemata* and *Conjunctive/UC Queries*

pretty much any extension leads to complexity (decidability) issues

Next time: **THE DATABASE EMPIRE STRIKES BACK**