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 Σ (set of integrity constraints);
(2) Data $D = \{R_1, \ldots, R_k\}$ (instance of access paths); and
(3) Query φ (a formula)

University of Waterloo

Setup

Setting

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

Definition (Certain Answers)

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

Setup

Setting

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

Definition (Certain Answers)

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

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

We assume that for every access path $R_{\mathrm{AP}}(\vec{x})$ in $D_{\mathcal{A}}$ there is

- a logical predicate $R(\vec{x})$ (with the same arity), and
- a constraint $\forall \vec{x}.R_{\mathrm{AP}}(\vec{x}) \to R(\vec{x})$.

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
   - $\Rightarrow$ conjunction, $\bot$, domain/range, unqualified $\exists$, role inverse, UNA
   - $\Rightarrow$ certain answers in AC_0 for data complexity (maps to SQL)
2. The $\mathcal{EL}$ family
   - $\Rightarrow$ conjunction, qualified $\exists$
   - $\Rightarrow$ certain answers *PTIME-complete* for data complexity
3. The $\mathcal{CFD}$ family
   - $\Rightarrow$ qualified $\forall$ (over total functions), functional dependencies
   - $\Rightarrow$ certain answers *PTIME-complete* for data complexity

DL-Lite Family of DLs

Definition (DL-Lite family: Schemata/TBoxes)

1. *Roles* R and *concepts* C as follows:

$$R ::= P \mid P^- \qquad C ::= \bot \mid A \mid \exists R$$

2. Schemata are represented as TBoxes: a finite set $\mathcal{T}$ of *constraints*

$$C_1 \sqcap \cdots \sqcap C_n \sqsubseteq C \qquad R_1 \sqsubseteq R_2$$

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

University of Waterloo

DL-Lite Family of DLs

Definition (DL-Lite family: Schemata/TBoxes)

1. *Roles R* and *concepts C* as follows:

$$R ::= P \mid P^- \qquad C ::= \bot \mid A \mid \exists R$$

2. Schemata are represented as TBoxes: a finite set $\mathcal{T}$ of *constraints*

$$C_1 \sqcap \cdots \sqcap C_n \sqsubseteq C \qquad R_1 \sqsubseteq R_2$$

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 query.

University of Waterloo

Example

TBox (Schema): $Employee \sqsubseteq \exists Works$

$\exists Works^- \sqsubseteq Project$

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

Example

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

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

Rewriting:

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

University of Waterloo

Example

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

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

Rewriting:

$$Q^\dagger = \begin{array}{l} (\exists y. Works(x, y) \land Project(y)) \lor \\ (\exists y, z. Works(x, y) \land Works(z, y)) \lor \end{array}$$

Example

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

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

Rewriting:

$$
\begin{aligned}
Q^{\dagger} = \quad & (\exists y.Works(x,y) \wedge Project(y)) \vee \\
& (\exists y,z.Works(x,y) \wedge Works(z,y)) \vee \\
& (\exists y.Works(x,y)) \vee
\end{aligned}
$$

University of Waterloo

Example

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

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

Rewriting:

$$
\begin{aligned}
Q^{\dagger} = \quad & (\exists y.Works(x, y) \wedge Project(y)) \vee \\
& (\exists y, z.Works(x, y) \wedge Works(z, y)) \vee \\
& (\exists y.Works(x, y)) \vee \\
& (Employee(x))
\end{aligned}
$$

Example

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

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

Rewriting:

$$
\begin{aligned}
Q^{\dagger} = \quad & (\exists y.Works_{AP}(x,y) \wedge Project_{AP}(y)) \vee \\
& (\exists y,z.Works_{AP}(x,y) \wedge Works_{AP}(z,y)) \vee \\
& (\exists y.Works_{AP}(x,y)) \vee \\
& (Employee_{AP}(x))
\end{aligned}
$$

Query Execution:

$$
Q^{\dagger}\left(\begin{array}{l} \{ Employee(bob), \\ \quad Works(sue, slides) \} \end{array} \right)
$$

Example

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

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

Rewriting:

$$\begin{aligned} Q^{\dagger} = \; & (\exists y. Works_{AP}(x, y) \wedge Project_{AP}(y)) \vee \\ & (\exists y, z. Works_{AP}(x, y) \wedge Works_{AP}(z, y)) \vee \\ & (\exists y. Works_{AP}(x, y)) \vee \\ & (Employee_{AP}(x)) \end{aligned}$$

Query Execution:

$$Q^{\dagger}\left(\begin{array}{l} \{ Employee(bob), \\ \;\; Works(sue, slides) \} \end{array} \right) = \{bob, sue\}$$

QuOnto: Rewriting Approach [Calvanese et al.]

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

QuOnto: Rewriting Approach [Calvanese et al.]

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

Theorem

$\Sigma \cup \mathcal{A} \models Q(\vec{a})$ *if and only if* $D_{\mathcal{A}} \models Q^\dagger(\vec{a})$

University of Waterloo

QuOnto: Rewriting Approach [Calvanese et al.]

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

Theorem

$\Sigma \cup \mathcal{A} \models Q(\vec{a})$ *if and only if* $D_\mathcal{A} \models Q^\dagger(\vec{a})$ $\Leftarrow$ *can be VERY large*

University of Waterloo

$\mathcal{EL}$ 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*
$$C_1 \sqsubseteq C_2 \qquad\qquad R_1 \sqsubseteq R_2$$

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

$\mathcal{EL}$ 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*

$$C_1 \sqsubseteq C_2 \qquad\qquad R_1 \sqsubseteq R_2$$

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 Σ,
2. the conjunctive query Q to a relational query $Q^{\ddagger}$.

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

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 Σ,
2. the conjunctive query Q to a relational query $Q^\ddagger$.

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

Theorem (Lutz, _, Wolter: IJCAI'09)

$$\Sigma \cup \mathcal{A} \models Q(\vec{a}) \textit{ if and only if } D^*_{\mathcal{A}} \models Q^\ddagger(\vec{a})$$

Example (with almost DL-Lite schema)

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

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

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: $\exists y.Works(x, y) \wedge Project(y)$

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

Rewriting:

1. $D^*_{\mathcal{A}} = \{$ $Employee(bob), Works(bob, c_{Works}),$
$Works(sue, slides), Works(sue, c_{Works}), Project(c_{Works}), \}$
2. $Q^{\ddagger} = Q \wedge (x \neq c_{Works})$

Example (with almost DL-Lite schema)

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

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

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

Rewriting:

1. $D^*_{\mathcal{A}} = \{\ Employee(bob), Works(bob, c_{Works}),$
   $Works(sue, slides), Works(sue, c_{Works}), Project(c_{Works}), \}$
2. $Q^{\ddagger} = Q \wedge (x \neq c_{Works})$

Query Execution:

$$Q^{\ddagger}(D^*_{\mathcal{A}}) = \{bob, sue\}$$

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)

$$\Sigma \cup A \models Q(\vec{a}) \text{ if and only if } D^*_{\mathcal{A}} \models Q^{\ddagger}(\vec{a})$$

(… 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)

$$\Sigma \cup A \models Q(\vec{a}) \textit{ if and only if } D^*_{\mathcal{A}} \models Q^{\ddagger}(\vec{a})$$

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

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

$$\Sigma \cup A \models Q(\vec{a}) \textit{ if and only if } D^*_{\mathcal{A}} \models Q^{\text{filter}}(\vec{a})$$

(. . . polynomial in $|\mathcal{H}|$, but uses UDF feature of DB2.)

$\mathcal{CFD}$ family of Logics

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

1. Syntax formed from *path functions* Pf and *concepts* C, D as follows:

$$C ::= A \mid \forall\, \mathrm{Pf}\,.C$$
$$D ::= A \mid \neg C \mid \forall\, \mathrm{Pf}\,.C \mid \mathrm{C} : \mathrm{Pf}_1, \ldots, \mathrm{Pf}_k \to \mathrm{Pf}$$

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* ($\mathrm{Pf}(a) = \mathrm{Pf}'(b)$) assertions.

$\mathcal{CFD}$ family of Logics

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

1. Syntax formed from *path functions* Pf and *concepts* C, D as follows:

$$C ::= A \mid \forall\,\mathsf{Pf}\,.C$$
$$D ::= A \mid \neg C \mid \forall\,\mathsf{Pf}\,.C \mid \mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$$

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* ($\mathsf{Pf}(a) = \mathsf{Pf}'(b)$) assertions.

$\mathcal{CFD}$ family of Logics

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

1. Syntax formed from *path functions* Pf and *concepts* C, D as follows:

$$C ::= A \mid \forall \mathrm{Pf}\,.C$$
$$D ::= A \mid \neg C \mid \forall \mathrm{Pf}\,.C \mid \mathrm{C} : \mathrm{Pf}_1, \ldots, \mathrm{Pf}_k \to \mathrm{Pf}$$

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* ($\mathrm{Pf}(a) = \mathrm{Pf}'(b)$) assertions.

Rewriting Approach: can't work—reachability in ABox (PTIME-c)
Combined Approach: can't work—too many *types* (anon. completion too big)

$\mathcal{CFD}$ family of Logics

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

1. Syntax formed from *path functions* Pf and *concepts* C, D as follows:

$$C ::= A \mid \forall \text{Pf}\,.C$$
$$D ::= A \mid \neg C \mid \forall \text{Pf}\,.C \mid \mathrm{C} : \text{Pf}_1, \ldots, \text{Pf}_k \rightarrow \text{Pf}$$

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.

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.

University of Waterloo

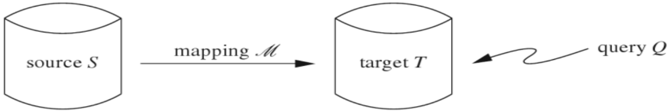
DATA EXCHANGE

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 φ is a CQ over S_P and ψ a CQ over S_L.

The general setting of data exchange is this:



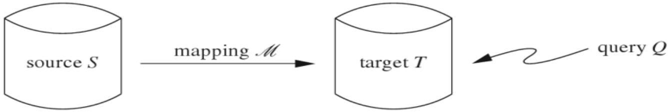
[Arenas et al: Foundations of Data Exchange]

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
  $\forall\vec{x}.\varphi(\vec{x}) \to \exists\vec{y}.\psi(\vec{x},\vec{y})$ where φ is a CQ over S_P and ψ a CQ over S_L.

The general setting of data exchange is this:



[Arenas et al: Foundations of Data Exchange]

Definition

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

... too many solutions (TGDs imply open world @S_L!)

Universal Solutions and Cores

Problem(s):

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

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

Universal Solutions and Cores

Problem(s):

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

$\Rightarrow$ 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.

Universal Solutions and Cores

Problem(s):

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

$\Rightarrow$ 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

$\Rightarrow$ 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

  $\Rightarrow$ variables (marked nulls): *representation system* [Imielinski&Lipski'84]

Universal Solutions and Cores

Problem(s):

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

$\Rightarrow$ 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

  $\Rightarrow$ variables (marked nulls): *representation system* [Imielinski&Lipski'84]

  $\Rightarrow$ 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*.

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);

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)
coNP-hard for *DATA COMPLEXITY*

Example

- Schema&Data:

$$\begin{array}{rcll} \Sigma & = & \{ & \forall x, y.ColNode(x,y) \leftrightarrow Node(x), \\ & & & \forall x, y.ColNode(x,y) \leftrightarrow Colour(y) \quad \} \\ D & = & \{ & Edge = \{(n_i, n_j)\}, Node = \{n_1, \ldots n_m\}, \\ & & & Colour = \{r, g, b\} \quad \} \end{array}$$

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:

$$\Sigma = \{ \forall x, y. ColNode(x,y) \leftrightarrow Node(x), \; \forall x, y. ColNode(x,y) \leftrightarrow Colour(y) \}$$

$$D = \{ Edge = \{(n_i, n_j)\}, Node = \{n_1, \ldots n_m\}, \; Colour = \{r, g, b\} \}$$

- Query: $\exists x, y, c. Edge(x,y) \land ColNode(x,c) \land ColNode(y,c)$

University of Waterloo

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:

$$\Sigma = \{ \forall x, y. ColNode(x, y) \leftrightarrow Node(x), \forall x, y. ColNode(x, y) \leftrightarrow Colour(y) \}$$

$$D = \{ Edge = \{(n_i, n_j)\}, Node = \{n_1, \ldots n_m\}, Colour = \{r, g, b\} \}$$

- Query: $\exists x, y, c. Edge(x, y) \wedge ColNode(x, c) \wedge ColNode(y, c)$

$\Rightarrow$ the graph ($Node$, $Edge$) is NOT 3-colourable.

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:

$$\Sigma = \{ \forall x, y. ColNode(x, y) \leftrightarrow Node(x), \quad \forall x, y. ColNode(x, y) \leftrightarrow Colour(y) \}$$

$$D = \{ Edge = \{(n_i, n_j)\}, Node = \{n_1, \dots n_m\}, \quad Colour = \{r, g, b\} \}$$

- Query: $\exists x, y, c. Edge(x, y) \wedge ColNode(x, c) \wedge ColNode(y, c)$
$\Rightarrow$ 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:

$$\Sigma = \{ \forall x, y. ColNode(x, y) \leftrightarrow Node(x), \forall x, y. ColNode(x, y) \leftrightarrow Colour(y) \}$$

$$D = \{ Edge = \{(n_i, n_j)\}, Node = \{n_1, \ldots n_m\}, Colour = \{r, g, b\} \}$$

- Query: $\exists x, y, c. Edge(x, y) \wedge ColNode(x, c) \wedge ColNode(y, c)$

$\Rightarrow$ the graph $(Node, Edge)$ is NOT 3-colourable.

... coNP-complete for all DLs between $\mathcal{AL}$ and $\mathcal{SHIQ}$.

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

Data Exchange cannot say $\forall x, y. ColNode(x, y) \rightarrow Colour(y)$ (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$^{\mathcal{H}}$ 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.

Certain Answers??

Example (Unintuitive Behaviour of Queries:)

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

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

Certain Answers??

Example (Unintuitive Behaviour of Queries:)

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

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

Embedded SQL-like Example

if "$\exists x.Phone(\texttt{"John"}, x)$" **then**
begin
x := "$Phone(\texttt{"John"}, x)$";
print `"John's phone number is:"` x
end

Certain Answers??

Example (Unintuitive Behaviour of Queries:)

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

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

Embedded SQL-like Example

if "$\exists x.Phone(\texttt{"John"}, x)$" **then**
  begin
    x := "$Phone(\texttt{"John"}, x)$";
    print `"John's phone number is:"` x
  end

Certain Answers??

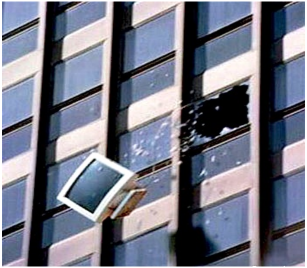
Example (Unintuitive Behaviour of Queries:)

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

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

Embedded SQL-like Example

if "$\exists x.Phone(\texttt{"John"}, x)$" **then**
begin
x := "$Phone(\texttt{"John"}, x)$";
print `"John's phone number is:"` x
end



Summary

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

Summary

- 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