Ontology-based Data Access
a.k.a. Queries and the Open World Assumption

David Toman

D. R. Cheriton School of Computer Science
University of Waterloo

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*?

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of *some* predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*? OPTION 1:

Definition (Implicit Definability)

A query Q is *implicitly definable in Ds* if $Q(M_1) = Q(M_2)$ for all pairs of databases $M_1 \models \mathcal{T}$ and $M_2 \models \mathcal{T}$ s. t. $A_i(M_1) = A_i(M_2)$ for all $A_i \in D$.

1. Chase/Craig Interpolation provides *rewriting* $\psi(D)$
2. In some cases φ is not implicitly definable

$\Rightarrow$ in particular when *OWA* plays a role (e.g., NULLs)

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*? OPTION 1:

Definition (Implicit Definability)

A query Q is *implicitly definable in Ds* if $Q(M_1) = Q(M_2)$ for all pairs of databases $M_1 \models \mathcal{T}$ and $M_2 \models \mathcal{T}$ s. t. $A_i(M_1) = A_i(M_2)$ for all $A_i \in D$.

1. Chase/Craig Interpolation provides *rewriting* $\psi(D)$
2. In some cases φ is not implicitly definable

$\Rightarrow$ in particular when *OWA* plays a role (e.g., NULLs)

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*? OPTION 2:

Definition (Certain Answers)

Answer to $\varphi(D)$ under $\mathcal{T}$:= $\text{cert}_{\mathcal{T},D}(\varphi) = \bigcap_{M \models \mathcal{T} \cup D} \{\vec{a} \mid M, \vec{a} \models \varphi\}$

1. Essentially a variant of [Imielinski&Lipski] approach
2. Answer to φ is *always* defined (unlike in OPTION 1)

... any drawbacks?

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*? OPTION 2:

Definition (Certain Answers)

$$\text{Answer to } \varphi(D) \text{ under } \mathcal{T} := \text{cert}_{\mathcal{T},D}(\varphi) = \bigcap_{M \models \mathcal{T} \cup D} \{\vec{a} \mid M, \vec{a} \models \varphi\}$$

1. Essentially a variant of [Imielinski&Lipski] approach
2. Answer to φ is *always* defined (unlike in OPTION 1)

. . . any drawbacks?

Open World Assumption and Possible Worlds

Setting

Input: (1) Schema $\mathcal{T}$ (set of integrity constraints);
(2) Data $D = \{A_1, \ldots, A_k\}$ (instance of some predicates); and
(3) Query φ (a formula)

How do we *answer φ over D w.r.t. $\mathcal{T}$*? OPTION 2:

Definition (Certain Answers)

$$\text{Answer to } \varphi(D) \text{ under } \mathcal{T} := \text{cert}_{\mathcal{T},D}(\varphi) = \bigcap_{M \models \mathcal{T} \cup D} \{\vec{a} \mid M, \vec{a} \models \varphi\}$$

1. Essentially a variant of [Imielinski&Lipski] approach
2. Answer to φ is *always* defined (unlike in OPTION 1)

... any drawbacks?

Certain Answers: Impact on Queries

Example (Unintuitive Behaviour of Queries:)

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

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

Certain Answers: Impact on Queries

Example (Unintuitive Behaviour of Queries:)

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

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

What is the Price?

High Computational Cost:

coNP-hard for *DATA COMPLEXITY*

Example

- Schema&Data:

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

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

- Query:

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

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

What is the Price?

High Computational Cost:

coNP-hard for *DATA COMPLEXITY*

Example

- Schema&Data:

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

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

- Query:

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

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

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

What is the Price?

High Computational Cost:

coNP-hard for *DATA COMPLEXITY*

Example

- Schema&Data:

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

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

- 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 $\mathcal{SHIQ}$.

What is the Price?

High Computational Cost:

coNP-hard for *DATA COMPLEXITY*

Example

- Schema&Data:

$$\mathcal{T} = \{ \begin{array}{l} \forall x, y. ColNode(x, y) \leftrightarrow Node(x), \\ \forall x, y. ColNode(x, y) \leftrightarrow Colour(y) \end{array} \}$$

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

- 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 $\mathcal{SHIQ}$.

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*?

YES: *Conjunctive queries* (or positive) and
certain *(dialects of) Description Logics* (or OWL profiles):

1. The DL-Lite family
   - ⇒ conjunction, $\perp$, domain/range, unqualified $\exists$, role inverse, UNA
   - ⇒ certain answers in AC_0 for data complexity (i.e., maps to SQL)
2. The $\mathcal{EL}$ family
   - ⇒ conjunction, qualified $\exists$
   - ⇒ certain answers *PTIME-complete* for data complexity

…schemas are *weak on purpose*: queries *must not* be definable.

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*?

YES: *Conjunctive queries* (or positive) and
certain *(dialects of) Description Logics* (or OWL profiles):

1. The DL-Lite family

   $\Rightarrow$ conjunction, $\perp$, domain/range, unqualified $\exists$, role inverse, UNA

   $\Rightarrow$ certain answers in AC_0 for data complexity (i.e., maps to SQL)

2. The $\mathcal{EL}$ family

   $\Rightarrow$ conjunction, qualified $\exists$

   $\Rightarrow$ certain answers *PTIME-complete* for data complexity

. . . schemas are *weak on purpose*: queries *must not* be definable.

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*?

YES: *Conjunctive queries* (or positive) and certain *(dialects of) Description Logics* (or OWL profiles):

1. The DL-Lite family
   $\Rightarrow$ conjunction, $\bot$, domain/range, unqualified $\exists$, role inverse, UNA
   $\Rightarrow$ certain answers in AC_0 for data complexity (i.e., maps to SQL)
2. The $\mathcal{EL}$ family
   $\Rightarrow$ conjunction, qualified $\exists$
   $\Rightarrow$ certain answers *PTIME-complete* for data complexity

...schemas are *weak on purpose*: queries *must not* be definable.

DL-Lite Family of DLs

Definition (DL-Lite family: Schemata and TBoxes)

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

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

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

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

Definition (DL-Lite family: Data and ABoxes)

ABox $\mathcal{A}$ is a finite set of *concept* $A(a)$ and *role* assertions $P(a, b)$.

$\Rightarrow$ OWA here: ABox does NOT say “these are all the tuples”!

How to compute answers to CQs?

IDEA: incorporate *schematic knowledge* into the query.

DL-Lite Family of DLs

Definition (DL-Lite family: Schemata and TBoxes)

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

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

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

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

Definition (DL-Lite family: Data and ABoxes)

ABox $\mathcal{A}$ is a finite set of *concept* $A(a)$ and *role* assertions $P(a, b)$.

$\Rightarrow$ OWA here: ABox does NOT say “these are all the tuples”!

How to compute answers to CQs?

IDEA: incorporate *schematic knowledge* into the query.

Example

TBox (Schema):

$$\begin{aligned} \mathit{Employee} &\sqsubseteq \exists \mathit{Works} \\ \exists \mathit{Works}^{-} &\sqsubseteq \mathit{Project} \end{aligned}$$

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

Rewriting:

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

Query Execution:

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

Example

TBox (Schema):

$$\begin{aligned} Employee &\sqsubseteq \exists Works \\ \exists Works^- &\sqsubseteq Project \end{aligned}$$

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

Rewriting:

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

Query Execution:

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

Example

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

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

Rewriting:

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

Query Execution:

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

Example

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

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

Rewriting:

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

Query Execution:

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

QuOnto: Rewriting Approach [Calvanese et al.]

Input: Conjunctive query Q, DL-Lite TBox $\mathcal{T}$
$R = \{Q\}$;
repeat
    foreach *query* $Q' \in R$ **do**
        foreach *axiom* $\alpha \in \mathcal{T}$ **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

$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q$ *if and only if* $\mathcal{A}, \vec{a} \models Q^\dagger$ $\Leftarrow$ *can be VERY large*

QuOnto: Rewriting Approach [Calvanese et al.]

Input: Conjunctive query Q, DL-Lite TBox $\mathcal{T}$
$R = \{Q\}$;
repeat
  foreach *query* $Q' \in R$ **do**
    foreach *axiom* $\alpha \in \mathcal{T}$ **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

$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q$ *if and only if* $\mathcal{A}, \vec{a} \models Q^\dagger$ $\Leftarrow$ *can be VERY large*

QuOnto: Rewriting Approach [Calvanese et al.]

Input: Conjunctive query Q, DL-Lite TBox $\mathcal{T}$
$R = \{Q\}$;
repeat
  foreach *query* $Q' \in R$ **do**
    foreach *axiom* $\alpha \in \mathcal{T}$ **do**
      if α *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**
        $\sigma = MGU(D_1, D_2)$; $R = R \cup \{\lambda(Q', \sigma)\}$;
until *no query unique up to variable renaming can be added to* R ;
return $Q^\dagger := (\bigvee R)$

Theorem

$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q$ *if and only if* $\mathcal{A}, \vec{a} \models Q^\dagger$ $\Leftarrow$ *can be VERY large*

$\mathcal{EL}$ 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. Schemas are represented as TBoxes: a finite set $\mathcal{T}$ of *constraints*
$$C_1 \sqsubseteq C_2 \qquad\qquad R_1 \sqsubseteq R_2$$

Definition ($\mathcal{EL}$-Lite family: Data and ABoxes)

ABox $\mathcal{A}$ is a finite set of *concept* $A(a)$ and *role* assertions $P(a, b)$.

$\Rightarrow$ OWA again: ABox does NOT say “these are all the tuples”!

How to compute answers to CQs?

IDEA: incorporate *schematic knowledge* into the data.

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

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

Definition ($\mathcal{EL}$-Lite family: Data and ABoxes)

ABox $\mathcal{A}$ is a finite set of *concept* $A(a)$ and *role* assertions $P(a, b)$.

$\Rightarrow$ OWA again: ABox does NOT say "these are all the tuples"!

How to compute answers to CQs?

IDEA: incorporate *schematic knowledge* into the data.

Combined Approach

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

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

Combined Approach

We effectively transform

1. the ABox $\mathcal{A}$ to a relational database $D_{\mathcal{A}}$ using constraints in $\mathcal{T}$,
2. the conjunctive query Q to a relational query $Q^{\ddagger}$.

…both *polynomial* in the input(s)

Theorem (Lutz, T., Wolter: IJCAI'09)

$$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q \textit{ if and only if } D_{\mathcal{A}}, \vec{a} \models Q^{\ddagger}$$

Combined Approach

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

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

Combined Approach

We effectively transform

1. the ABox $\mathcal{A}$ to a relational database $D_{\mathcal{A}}$ using constraints in $\mathcal{T}$,
2. the conjunctive query Q to a relational query $Q^{\ddagger}$.

…both *polynomial* in the input(s)

Theorem (Lutz, T., Wolter: IJCAI'09)

$$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q \textit{ if and only if } D_{\mathcal{A}}, \vec{a} \models Q^{\ddagger}$$

Combined Approach

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

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

Combined Approach

We effectively transform

1. the ABox $\mathcal{A}$ to a relational database $D_{\mathcal{A}}$ using constraints in $\mathcal{T}$,
2. the conjunctive query Q to a relational query $Q^{\ddagger}$.

…both *polynomial* in the input(s)

Theorem (Lutz, T., Wolter: IJCAI'09)

$$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q \text{ if and only if } D_{\mathcal{A}}, \vec{a} \models Q^{\ddagger}$$

Combined Approach

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

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

Combined Approach

We effectively transform

1. the ABox $\mathcal{A}$ to a relational database $D_{\mathcal{A}}$ using constraints in $\mathcal{T}$,
2. the conjunctive query Q to a relational query $Q^{\ddagger}$.

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

Theorem (Lutz, T., Wolter: IJCAI'09)

$$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q \text{ if and only if } D_{\mathcal{A}}, \vec{a} \models Q^{\ddagger}$$

Example (with DL-Lite schema)

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

$\exists Works.\top \sqsubseteq \exists Works.Project$

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

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

Rewriting:

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

Query Execution:

$$Q^{\ddagger}(D_{\mathcal{A}}) = \{bob, sue\}$$

Example (with DL-Lite schema)

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

$\exists Works.\top \sqsubseteq \exists Works.Project$

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

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

Rewriting:

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

Query Execution:

$$Q^{\ddagger}(D_{\mathcal{A}}) = \{bob, sue\}$$

Example (with DL-Lite schema)

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

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

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

Rewriting:

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

Query Execution:

$$Q^{\ddagger}(D_{\mathcal{A}}) = \{bob, sue\}$$

Experiments

Ontology **NCI** (70k axioms, 65k classes, 70 roles)

Size of the original $\mathcal{A}$									
Concept	100K	100K	100K	200K	200K	200K	400K	800K	1.6M
Role	25K	50K	75K	40K	65K	90K	360K	1.5M	5.8M
Size of the completed $D_{\mathcal{A}}$									
Concept	440K	440K	441K	683K	683K	684K	1.3M	2.6M	5.1M
Role	197K	237K	273K	323K	371K	414K	986K	2.7M	8.2M
Query execution time in seconds									
Q1 (2c1r)	0.19	0.19	0.20	0.23	0.25	0.24	0.27	0.46	0.59
Q2 (3c2r)	0.23	0.22	0.23	0.52	0.25	0.56	0.33	0.42	0.69
Q3 (3c2r)	0.25	0.27	0.26	0.31	0.31	0.31	0.42	0.86	1.13
Q4 (4c3r)	0.24	0.23	0.23	0.25	0.26	0.25	0.31	0.42	1.44
Q5 (5c5r)	0.36	0.36	0.30	0.60	0.34	0.45	2.24	7.93	128

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*;
(…and still exponential for *role hierarchies*.)

Theorem (Konchatov et al., KR10)

$$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q \textit{ if and only if } D_{\mathcal{A}}, \vec{a} \models Q^{\ddagger}$$

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*;

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

Theorem (Konchatov et al., KR10)

$$\mathcal{T} \cup \mathcal{A}, \vec{a} \models Q \text{ if and only if } D_{\mathcal{A}}, \vec{a} \models Q^{\ddagger}$$

Experiments: a Comparison

Ontology **Core** (381 axioms, 81 concepts, 58 roles)

Size of the original $\mathcal{A}$									
Individuals	100k			200k			300k		
Concept	5.0M			10.0M			20.0M		
Role	5.0M			10.0M			20.0M		
Size of the completed $D_{\mathcal{A}}$									
Concept	11.8M			23.7M			54.5M		
Role	5.7M			11.4M			27.8M		
Query execution time (base, combined, rewritten)									
Q1	0.46	3.97	25"32	0.80	5.73	38"33	1.28	7.32	23"04
Q2	0.53	5.97	97"47	0.86	6.65	67"13	1.34	8.03	71"49
Q3	0.50	1.10	2"15	0.81	1.78	3"28	1.87	3.12	5"31
Q4	0.20	1.00	12"02	0.78	2.57	13"38	1.70	3.86	14"55

Summary

1. Answering queries over databases *with respect to schema constraints/ontologies* is hard.

2. Choice between:

   Query Definability:

   $\Rightarrow$ expressive schema languages and queries
   $\Rightarrow$ rewritten queries in AC_0 ($\sim$ efficient)
   $\Rightarrow$ but rewriting *is hard to find* and may *not exist*

   Certain Answers:

   $\Rightarrow$ weak schema languages and positive queries only
   $\Rightarrow$ rewritten queries still complex (data complexity)
   $\Rightarrow$ but certain answers are *always defined*