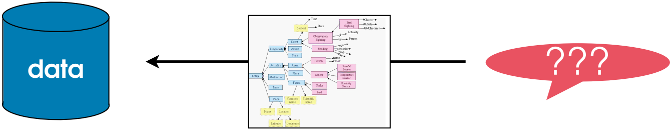
ONTOLOGY-MEDIATED QUERY ANSWERING

Harnessing knowledge to get more from data

Meghyn Bienvenu (*LaBRI - CNRS & University of Bordeaux*)

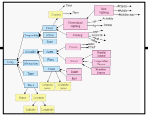


incomplete database

ontology

domain knowledge

user query

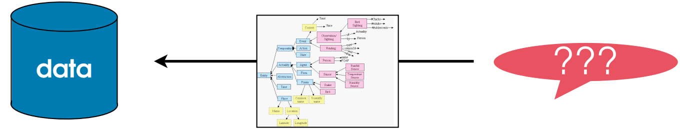


patient data
“Melanie has listeriosis”
“Paul has Lyme disease”

medical knowledge
“Listeriosis & Lyme disease are bacterial infections”

user query
“Find all patients with bacterial infections”

***expected answers*: Melanie, Paul**



patient data
"Melanie has listeriosis"
"Paul has Lyme disease"

medical knowledge
"Listeriosis & Lyme disease are bacterial infections"

user query
"Find all patients with bacterial infections"

***expected answers*: Melanie, Paul**

Why use an ontology?

- extend the vocabulary (making queries easier to formulate)
- provide a unified view of multiple data sources
- obtain more answers to queries (by exploiting domain knowledge)

Two main objectives:

- give a brief introduction to OMQA
- show connections between OMQA and theoretical CS

Structure of the talk:

- Introductory material
  - description logic (DL) ontologies, OMQA problem, **query rewriting**
- Understanding query rewriting
  - natural questions related to size and existence of rewritings
  - links to circuit complexity, automata, CSP

INTRODUCTION TO OMQA & QUERY REWRITING

OUR FOCUS: DESCRIPTION LOGIC ONTOLOGIES

Ontologies typically described using logic-based formalisms

In this talk: ontologies formulated in description logics (DLs)

- family of decidable fragments of first-order logic (FO)
- range from fairly simple to highly expressive
- complexity of query answering well understood
- lots of practical work on algorithms and implementations
- basis for OWL web ontology language (W3C standard)

Today, we'll mainly focus on three particular DLs:

- $\mathcal{ALC}$, $\mathcal{EL}$, DL-Lite$_R$

Building blocks of DLs:

- **concept names** (unary predicates, classes) — Prof, Course
- **role names** (binary predicates, properties) — teaches
- **individual names** (constants) — marie, inf100

Building blocks of DLs:

- **concept names** (unary predicates, classes) Prof, Course
- **role names** (binary predicates, properties) teaches
- **individual names** (constants) marie, inf100

Build **complex concepts and roles** using constructors. For example:

- Non-professors: $\neg\mathsf{Prof}$
- Profs who teach a Master's course: $\mathsf{Prof} \sqcap \exists\mathsf{teaches}.\mathsf{MCourse}$
- Taught by: $\mathsf{teaches}^{-}$

Note: set of available constructors **depends on the particular DL**!

Knowledge base (KB) = ABox (data) + TBox (ontology)

ABox contains **facts about specific individuals**

- finite set of **concept assertions** $A(a)$ and **role assertions** $r(a, b)$
- example assertions: $\mathsf{Prof}(\mathit{marie})$, $\mathsf{teaches}(\mathit{marie}, \mathit{inf}100)$

Knowledge base (KB) = ABox (data) + TBox (ontology)

ABox contains **facts about specific individuals**

- finite set of **concept assertions** $A(a)$ and **role assertions** $r(a, b)$
- example assertions: $\mathsf{Prof}(\mathit{marie}), \mathsf{teaches}(\mathit{marie}, \mathit{inf}100)$

TBox contains **general knowledge about the domain of interest**

- finite set of **axioms** (types of axioms depends on the DL)
- **concept inclusions** most common form of axiom
  - $C \sqsubseteq D$, with C, D complex concepts
  - intuitive meaning: "**everything that is a C is also a D**"
- examples on later slides

Interpretation $\mathcal{I}$ ("possible world") (like FO logic semantics)

- **domain of objects** $\Delta^{\mathcal{I}}$ (possibly infinite set)
- **interpretation function** $\cdot^{\mathcal{I}}$ that maps
  - **concept name** A $\rightsquigarrow$ set of objects $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - **role name** r $\rightsquigarrow$ set of pairs of objects $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
  - **individual name** a $\rightsquigarrow$ object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

Interpretation $\mathcal{I}$ ("possible world") (like FO logic semantics)

- **domain of objects** $\Delta^{\mathcal{I}}$ (possibly infinite set)
- **interpretation function** $\cdot^{\mathcal{I}}$ that maps
  - **concept name** $A \rightsquigarrow$ set of objects $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - **role name** $r \rightsquigarrow$ set of pairs of objects $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
  - **individual name** $a \rightsquigarrow$ object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- extend $\cdot^{\mathcal{I}}$ to complex concepts and roles, for example:
  - $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ $\quad (\exists r.C)^{\mathcal{I}} = \{d_1 \mid \text{ exists } (d_1, d_2) \in r^{\mathcal{I}} \text{ with } d_2 \in C^{\mathcal{I}}\}$

Interpretation $\mathcal{I}$ ("possible world") (like FO logic semantics)

- **domain of objects** $\Delta^\mathcal{I}$ (possibly infinite set)
- **interpretation function** $\cdot^\mathcal{I}$ that maps
  - **concept name** $A \rightsquigarrow$ set of objects $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$
  - **role name** $r \rightsquigarrow$ set of pairs of objects $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
  - **individual name** $a \rightsquigarrow$ object $a^\mathcal{I} \in \Delta^\mathcal{I}$
- extend $\cdot^\mathcal{I}$ to complex concepts and roles, for example:
  - $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$ $\quad (\exists r.C)^\mathcal{I} = \{d_1 \mid \text{ exists } (d_1, d_2) \in r^\mathcal{I} \text{ with } d_2 \in C^\mathcal{I}\}$

Satisfaction in an interpretation

$\mathcal{I}$ satisfies $B(a) \Leftrightarrow a^\mathcal{I} \in B^\mathcal{I}$ $\qquad$ $\mathcal{I}$ satisfies $C \sqsubseteq D \Leftrightarrow C^\mathcal{I} \subseteq D^\mathcal{I}$

Model of a KB $\mathcal{K}$ = interpretation that satisfies all statements in $\mathcal{K}$

$\mathcal{K}$ **entails** α (written $\mathcal{K} \models \alpha$) = every model $\mathcal{I}$ of $\mathcal{K}$ satisfies α

DESCRIPTION LOGIC ALC

In $\mathcal{ALC}$, we have the following concept constructors:

- top concept $\top$ (acts as a "wildcard", denotes set of all things)
- bottom concept $\bot$ (denotes empty set)
- conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$)
- restricted forms of existential and universal quantification ($\exists$, $\forall$)

Complex concepts are formed as follows:

$$C, D := \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \forall r.C$$

where A is a concept name, r a role name.

In $\mathcal{ALC}$, we have the following concept constructors:

- top concept $\top$ (acts as a "wildcard", denotes set of all things)
- bottom concept $\bot$ (denotes empty set)
- conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$)
- restricted forms of existential and universal quantification ($\exists$, $\forall$)

Complex concepts are formed as follows:

$$C, D := \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \forall r.C$$

where A is a concept name, r a role name.

$\mathcal{ALC}$ TBox: set of concept inclusions $C \sqsubseteq D$

Professors and MCFs are disjoint classes of faculty

$$\mathsf{Prof} \sqsubseteq \mathsf{Faculty} \quad \mathsf{Mcf} \sqsubseteq \mathsf{Faculty} \quad \mathsf{Prof} \sqsubseteq \neg\mathsf{Mcf}$$

Every Master's student is supervised by some faculty member

$$\mathsf{MStudent} \sqsubseteq \exists \mathsf{supervisedBy}.\mathsf{Faculty}$$

Master's students are students who only take Master-level courses

$$\mathsf{MStudent} \sqsubseteq \mathsf{Student} \sqcap \forall \mathsf{takesCourse}.\mathsf{MCourse}$$

FO translation:

$\forall x\ (\mathsf{MStudent}(x) \rightarrow (\mathsf{Student}(x) \wedge \forall y\ \mathsf{takesCourse}(x, y) \rightarrow \mathsf{MCourse}(y))$

In $\mathcal{EL}$, complex concepts are constructed as follows:

$$C, D := \top \mid A \mid C \sqcap D \mid \exists r.C$$

$\mathcal{EL}$ TBox = set of inclusions $C \sqsubseteq D$, with C, D as above

In $\mathcal{EL}$, complex concepts are constructed as follows:

$$C, D := \top \mid A \mid C \sqcap D \mid \exists r.C$$

$\mathcal{EL}$ TBox = set of inclusions $C \sqsubseteq D$, with C, D as above

Advantage w.r.t. $\mathcal{ALC}$: reasoning much simpler (PTIME vs. EXPTIME)

Despite lower expressivity, $\mathcal{EL}$ very useful in practice

- used for large-scale biomedical ontologies (example: SNOMED)
- importance witnessed by OWL 2 EL profile

We present the dialect DL-Lite$_R$ (which underlies OWL 2 QL profile).

DL-Lite$_R$ TBoxes contain two types of axioms:

- concept inclusions $B_1 \sqsubseteq B_2$, $B_1 \sqsubseteq \neg B_2$
- role inclusions $S_1 \sqsubseteq S_2$, $S_1 \sqsubseteq \neg S_2$

where $B := A \mid \exists S \qquad S := r \mid r^-$

We present the **dialect DL-Lite$_R$** (which underlies **OWL 2 QL profile**).

DL-Lite$_R$ TBoxes contain two types of axioms:

- **concept inclusions** $B_1 \sqsubseteq B_2$, $B_1 \sqsubseteq \neg B_2$
- **role inclusions** $S_1 \sqsubseteq S_2$, $S_1 \sqsubseteq \neg S_2$

where $B := A \mid \exists S$ $\quad S := r \mid r^-$

Some DL-Lite$_R$ axioms:

- Every professor teaches something: $\mathsf{Prof} \sqsubseteq \exists \mathsf{teaches}$
- Everything that is taught is a course: $\exists \mathsf{teaches}^- \sqsubseteq \mathsf{Course}$
- Teaches inverse of taughtBy:
  $\mathsf{teaches} \sqsubseteq \mathsf{taughtBy}^-$, $\mathsf{teaches}^- \sqsubseteq \mathsf{taughtBy}$

Instance queries (IQs): find instances of a given concept or role

$\mathrm{Faculty}(x)$ $\mathrm{teaches}(x, y)$

Instance queries (IQs): find instances of a given concept or role

$\text{Faculty}(x)$ $\quad\quad$ $\text{teaches}(x, y)$

Conjunctive queries (CQs) $\sim$ select-project-join queries in SQL

conjunctions of atoms, some variables can be existentially quantified

$$\exists y.\, \text{Faculty}(x) \wedge \text{teaches}(x, y)$$

(find all faculty members that teach something)

Instance queries (IQs): find instances of a given concept or role

$\mathsf{Faculty}(x)$ $\mathsf{teaches}(x, y)$

Conjunctive queries (CQs) $\sim$ select-project-join queries in SQL

conjunctions of atoms, some variables can be existentially quantified

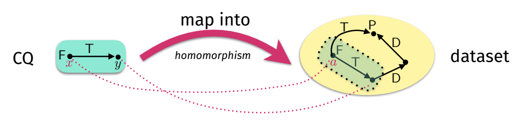
$$\exists y.\, \mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$$

(find all faculty members that teach something)

Ontology-mediated query (OMQ):

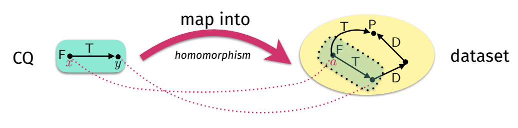
pair $(\mathcal{T}, q)$ with $\mathcal{T}$ a TBox and q a query (IQ / CQ)

Answering CQs in database setting



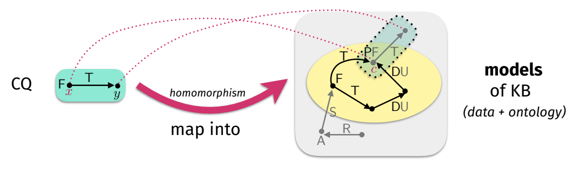
database D + query q $\rightsquigarrow$ set of answers $ans(q, D)$

Answering CQs in database setting



database D + query q $\rightsquigarrow$ set of answers $ans(q, D)$

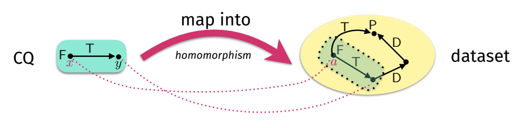
Answering CQs in the presence of a TBox (ontology)



model $\mathcal{I}$ of KB $(\mathcal{T}, \mathcal{A})$ + query q $\rightsquigarrow$ set of answers $ans(q, \mathcal{I})$

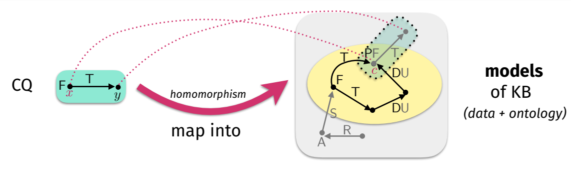
QUERY ANSWERING: DATABASE VS ONTOLOGY SETTINGS

Answering CQs in database setting



database D + query q $\rightsquigarrow$ set of answers $ans(q, D)$

Answering CQs in the presence of a TBox (ontology)



model $\mathcal{I}$ of KB $(\mathcal{T}, \mathcal{A})$ + query q $\rightsquigarrow$ set of answers $ans(q, \mathcal{I})$

Question: how to combine the answers from different models?

Certain answers:

- tuples of inds $\vec{a}$ such that $\vec{a} \in ans(q, \mathcal{I})$ for **every model** $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$
- corresponds to a form of **entailment**, we'll write $\mathcal{T}, \mathcal{A} \models q(\vec{a})$

Certain answers:

- tuples of inds $\vec{a}$ such that $\vec{a} \in ans(q, \mathcal{I})$ for **every model** $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$
- corresponds to a form of **entailment**, we'll write $\mathcal{T}, \mathcal{A} \models q(\vec{a})$

Ontology-mediated query answering: computing certain answers

Certain answers:

- tuples of inds $\vec{a}$ such that $\vec{a} \in ans(q, \mathcal{I})$ for every model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$
- corresponds to a form of entailment, we'll write $\mathcal{T}, \mathcal{A} \models q(\vec{a})$

Ontology-mediated query answering: computing certain answers

For Horn DLs (no form of disjunction) like $\mathcal{EL}$ and DL-Lite$_R$:
enough to consider a single canonical model

- idea: exhaustively apply TBox axioms to ABox
- possibly infinite ($A \sqsubseteq \exists r.A$)
- forest-shaped (ABox + new tree structures)
- give correct answer to all CQs

OMQA viewed as a **decision problem** (yes-or-no question):

PROBLEM: $\mathcal{Q}$ **answering in** $\mathcal{L}$ ($\mathcal{Q}$ a query language, $\mathcal{L}$ a DL)

INPUT: An n**-ary query** $q \in \mathcal{Q}$, an **ABox** $\mathcal{A}$, a $\mathcal{L}$**-TBox** $\mathcal{T}$, and a **tuple** $\vec{a} \in \mathsf{Ind}(\mathcal{A})^n$

QUESTION: **Does** $\mathcal{T}, \mathcal{A} \models q(\vec{a})$**?**

OMQA viewed as a **decision problem** (yes-or-no question):

PROBLEM: $\mathcal{Q}$ **answering in** $\mathcal{L}$ ($\mathcal{Q}$ a query language, $\mathcal{L}$ a DL)

INPUT: An n**-ary query** $q \in \mathcal{Q}$, an **ABox** $\mathcal{A}$, a $\mathcal{L}$**-TBox** $\mathcal{T}$, and a **tuple** $\vec{a} \in \mathsf{Ind}(\mathcal{A})^n$

QUESTION: **Does** $\mathcal{T}, \mathcal{A} \models q(\vec{a})$**?**

Combined complexity: in terms of size of whole input

Data complexity: in terms of **size of $\mathcal{A}$ only**

- view rest of input as fixed (of constant size)
- motivation: ABox (data) typically much larger than rest of input

data complexity $\leq$ combined complexity

Note: use $|\mathcal{A}|$ to denote **size of** $\mathcal{A}$ (similarly for $|\mathcal{T}|$, $|q|$, etc.)

Idea: reduce OMQA to database query evaluation

- rewriting step: OMQ $(\mathcal{T}, q) \rightsquigarrow$ first-order (SQL) query q'
- evaluation step: evaluate query q' using relational DB system

Advantage: harness efficiency of relational database systems

Idea: reduce OMQA to database query evaluation

- rewriting step: OMQ $(\mathcal{T}, q) \rightsquigarrow$ first-order (SQL) query q'
- evaluation step: evaluate query q' using relational DB system

Advantage: harness efficiency of relational database systems

Key notion: first-order (FO) rewriting

- FO query q' is an FO-rewriting of $(\mathcal{T}, q)$ iff for every ABox $\mathcal{A}$:

$$\mathcal{T}, \mathcal{A} \models q(\vec{a}) \quad \Leftrightarrow \quad DB_{\mathcal{A}} \models q'(\vec{a})$$

Informally: evaluating q' over $\mathcal{A}$ (viewed as DB) gives correct result

Idea: reduce OMQA to database query evaluation

- rewriting step: OMQ $(\mathcal{T}, q) \rightsquigarrow$ first-order (SQL) query q'
- evaluation step: evaluate query q' using relational DB system

Advantage: harness efficiency of relational database systems

Key notion: first-order (FO) rewriting

- FO query q' is an FO-rewriting of $(\mathcal{T}, q)$ iff for every ABox $\mathcal{A}$:

$$\mathcal{T}, \mathcal{A} \models q(\vec{a}) \quad \Leftrightarrow \quad DB_{\mathcal{A}} \models q'(\vec{a})$$

Informally: evaluating q' over $\mathcal{A}$ (viewed as DB) gives correct result

Can also consider Datalog rewritings, defined analogously

Good news: every CQ and DL-Lite$_R$ ontology has an FO-rewriting

Good news: every CQ and DL-Lite$_R$ ontology has an FO-rewriting

Example:

TBox $\mathcal{T} = \{ \text{Prof} \sqsubseteq \text{Faculty} \ \text{Mcf} \sqsubseteq \text{Faculty} \ \text{CR} \sqsubseteq \text{Faculty} \ \text{DR} \sqsubseteq \text{Faculty}$
$\text{Prof} \sqsubseteq \exists\text{teaches} \ \text{Mcf} \sqsubseteq \exists\text{teaches} \}$

Query $q_0 = \exists y \, \text{Faculty}(x) \land \text{teaches}(x, y)$

Good news: **every CQ and DL-Lite$_R$ ontology has an FO-rewriting**

Example:

TBox $\mathcal{T} = \{\, \text{Prof} \sqsubseteq \text{Faculty} \;\; \text{Mcf} \sqsubseteq \text{Faculty} \;\; \text{CR} \sqsubseteq \text{Faculty} \;\; \text{DR} \sqsubseteq \text{Faculty}$
$\text{Prof} \sqsubseteq \exists\text{teaches} \;\; \text{Mcf} \sqsubseteq \exists\text{teaches} \,\}$

Query $q_0 = \exists y\, \text{Faculty}(x) \wedge \text{teaches}(x, y)$

The following query is an **FO-rewriting of** $(\mathcal{T}, q_0)$:

$$q_0 \quad \vee \quad \text{Prof}(x) \quad \vee \quad \text{Mcf}(x)$$
$$\vee \quad \exists y\, \text{CR}(x) \wedge \text{teaches}(x, y) \quad \vee \quad \exists y\, \text{DR}(x) \wedge \text{teaches}(x, y)$$

Good news: **every CQ and DL-Lite$_R$ ontology has an FO-rewriting**

Example:

TBox $\mathcal{T} = \{$ Prof $\sqsubseteq$ Faculty Mcf $\sqsubseteq$ Faculty CR $\sqsubseteq$ Faculty DR $\sqsubseteq$ Faculty
Prof $\sqsubseteq \exists$teaches Mcf $\sqsubseteq \exists$teaches $\}$

Query $q_0 = \exists y\, \mathsf{Faculty}(x) \land \mathsf{teaches}(x, y)$

The following query is an **FO-rewriting of** $(\mathcal{T}, q_0)$:

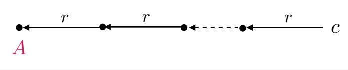
$$q_0 \quad \lor \quad \mathsf{Prof}(x) \quad \lor \quad \mathsf{Mcf}(x)$$
$$\lor \quad \exists y\, \mathsf{CR}(x) \land \mathsf{teaches}(x, y) \quad \lor \quad \exists y\, \mathsf{DR}(x) \land \mathsf{teaches}(x, y)$$

Existence of FO-rewritings $\Rightarrow$ **low data complexity** ($\mathsf{AC}_0 \subsetneq \mathsf{PTIME}$)

$\mathcal{EL} : \sqcap, \exists r.C$

In $\mathcal{EL}$, FO-rewritings need not exist:

- no FO-rewriting of $A(x)$ w.r.t. $\{\exists r.A \sqsubseteq A\}$



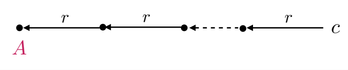
unbounded propagation of A along r

WHAT ABOUT EL?

$\mathcal{EL} : \sqcap, \exists r.C$

In $\mathcal{EL}$, FO-rewritings need not exist:

- no FO-rewriting of $A(x)$ w.r.t. $\{\exists r.A \sqsubseteq A\}$



unbounded propagation of A along r

Datalog rewritings always exist: Datalog $\sim$ function-free Horn clauses

- Datalog program Π: $r(x,y) \wedge A(x) \rightarrow A(y) \quad A(x) \rightarrow \text{goal}(x)$
- $\mathcal{T}, \mathcal{A} \models A(c)$ iff can derive $\text{goal}(c)$ from $\mathcal{A}$ using Π

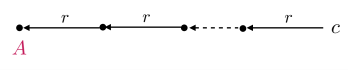
Can pass on rewriting to Datalog engine

WHAT ABOUT EL?

$\mathcal{EL}$: $\sqcap, \exists r.C$

In $\mathcal{EL}$, FO-rewritings need not exist:

- no FO-rewriting of $A(x)$ w.r.t. $\{\exists r.A \sqsubseteq A\}$



unbounded propagation of A along r

Datalog rewritings always exist: Datalog $\sim$ function-free Horn clauses

- Datalog program Π: $r(x,y) \wedge A(x) \rightarrow A(y)$ $\quad A(x) \rightarrow \text{goal}(x)$
- $\mathcal{T}, \mathcal{A} \models A(c)$ iff can derive $\text{goal}(c)$ from $\mathcal{A}$ using Π

Can pass on rewriting to Datalog engine

Datalog rewriting $\Rightarrow$ PTIME data complexity for CQ answering

WHAT ABOUT ALC?

$\mathcal{ALC}$: $\sqcap, \sqcup, \sqcap, \exists r.C, \forall r.C$

Neither FO nor Datalog rewritings need exist

Encoding of non-3-colourability:

TBox axioms:

- $\top \sqsubseteq R \sqcup G \sqcup B$
- $B \sqcap \exists \mathrm{edge}.B \sqsubseteq \mathrm{clash}$ (same for R, G)

Graph is 3-colourable $\Leftrightarrow$ Boolean query $\exists x.\mathrm{clash}(x)$ not entailed

CQ answering has coNP data complexity

Query rewriting:
data-independent reduction of OMQA to DB query evaluation

Query rewriting:
data-independent reduction of OMQA to DB query evaluation

To gain better understanding of query rewriting,
we consider the following natural questions:

1. Size of rewritings — DL-Lite
   - How large are the rewritten queries?
2. Existence of rewritings — beyond DL-Lite
   - When is query rewriting applicable?

SIZE OF REWRITINGS

Lots of rewriting algorithms for DL-Lite designed and tested

Most produce unions of conjunctive queries (UCQs = $\vee$ of CQs)

Lots of rewriting algorithms for DL-Lite designed and tested

Most produce unions of conjunctive queries (UCQs = $\vee$ of CQs)

Experiments showed that such rewritings can be huge!

- can be difficult / impossible to generate and evaluate

Not hard to see smallest UCQ-rewriting may be exponentially large:

Lots of rewriting algorithms for DL-Lite designed and tested

Most produce unions of conjunctive queries (UCQs = $\vee$ of CQs)

Experiments showed that such rewritings can be huge!

- can be difficult / impossible to generate and evaluate

Not hard to see smallest UCQ-rewriting may be exponentially large:

- Query: $A_1^0(x) \wedge \ldots \wedge A_n^0(x)$
- Ontology: $A_1^1 \sqsubseteq A_1^0 \quad A_2^1 \sqsubseteq A_2^0 \quad \ldots \quad A_n^1 \sqsubseteq A_n^0$
- Rewriting: $\bigvee_{(i_1,\ldots,i_n)\in\{0,1\}} A_1^{i_1}(x) \wedge A_1^{i_1}(x) \wedge \ldots \wedge A_1^{i_1}(x)$

Lots of rewriting algorithms for DL-Lite designed and tested

Most produce unions of conjunctive queries (UCQs = $\vee$ of CQs)

Experiments showed that such rewritings can be huge!

- can be difficult / impossible to generate and evaluate

Not hard to see smallest UCQ-rewriting may be exponentially large:

- Query: $A_1^0(x) \wedge \ldots \wedge A_n^0(x)$
- Ontology: $A_1^1 \sqsubseteq A_1^0 \quad A_2^1 \sqsubseteq A_2^0 \quad \ldots \quad A_n^1 \sqsubseteq A_n^0$
- Rewriting: $\bigvee_{(i_1,\ldots,i_n)\in\{0,1\}} A_1^{i_1}(x) \wedge A_1^{i_1}(x) \wedge \ldots \wedge A_1^{i_1}(x)$

But: simple polysize FO-rewriting does exist! $\bigwedge_{i=1}^{n}(A_i^0(x) \vee A_i^1(x))$

DIFFERENT FORMS OF REWRITINGS

PE-rewritings: positive existential queries (only $\exists$, $\wedge$, $\vee$)

$$(r(x,y) \vee s(y,x)) \wedge (A(x) \vee (B(x) \wedge \exists z\, p(x,z))) \wedge (A(y) \vee (B(y) \wedge \exists z\, p(y,z)))$$

DIFFERENT FORMS OF REWRITINGS

PE-rewritings: positive existential queries (only $\exists$, $\wedge$, $\vee$)

$$(r(x,y) \vee s(y,x)) \wedge (A(x) \vee (B(x) \wedge \exists z\, p(x,z))) \wedge (A(y) \vee (B(y) \wedge \exists z\, p(y,z)))$$

NDL-rewritings: non-recursive Datalog queries

$$q_1(x,y), q_2(x), q_2(y) \to \mathrm{goal}(x,y)$$

$$r(x,y) \to q_1(x,y) \qquad A(x) \to q_2(x)$$

$$s(y,x) \to q_1(x,y) \qquad B(x), p(x,z) \to q_2(x)$$

PE-rewritings: positive existential queries (only $\exists$, $\wedge$, $\vee$)

$$(r(x,y) \vee s(y,x)) \wedge (A(x) \vee (B(x) \wedge \exists z\, p(x,z))) \wedge (A(y) \vee (B(y) \wedge \exists z\, p(y,z)))$$

NDL-rewritings: non-recursive Datalog queries

$$q_1(x,y), q_2(x), q_2(y) \to \mathrm{goal}(x,y)$$

$$r(x,y) \to q_1(x,y) \qquad A(x) \to q_2(x)$$

$$s(y,x) \to q_1(x,y) \qquad B(x), p(x,z) \to q_2(x)$$

FO-rewritings: first-order queries (can also use $\forall$, $\neg$)

PE-rewritings: positive existential queries (only $\exists$, $\wedge$, $\vee$)

$$(r(x,y) \vee s(y,x)) \wedge (A(x) \vee (B(x) \wedge \exists z\, p(x,z))) \wedge (A(y) \vee (B(y) \wedge \exists z\, p(y,z)))$$

NDL-rewritings: non-recursive Datalog queries

$$q_1(x,y), q_2(x), q_2(y) \rightarrow \mathrm{goal}(x,y)$$

$$r(x,y) \rightarrow q_1(x,y) \qquad A(x) \rightarrow q_2(x)$$

$$s(y,x) \rightarrow q_1(x,y) \qquad B(x), p(x,z) \rightarrow q_2(x)$$

FO-rewritings: first-order queries (can also use $\forall$, $\neg$)

What if we replace UCQs by PE / NDL / FO?
Do we get polysize rewritings?

Exponential blowup unavoidable for PE / NDL-rewritings

Formally: sequence of CQs q_n and DL-Lite$_R$ TBoxes $\mathcal{T}_n$ such that

- PE- and NDL-rewritings of $(\mathcal{T}_n, q_n)$ exponential in $|q_n| + |\mathcal{T}_n|$
- FO-rewritings of $(\mathcal{T}_n, q_n)$ superpolynomial unless $\mathsf{NP}/\mathrm{poly} \subseteq \mathsf{NC}^1$

Exponential blowup unavoidable for PE / NDL-rewritings

Formally: sequence of CQs q_n and DL-Lite$_R$ TBoxes $\mathcal{T}_n$ such that

- PE- and NDL-rewritings of $(\mathcal{T}_n, q_n)$ exponential in $|q_n| + |\mathcal{T}_n|$
- FO-rewritings of $(\mathcal{T}_n, q_n)$ superpolynomial unless $\mathsf{NP}/\mathrm{poly} \subseteq \mathsf{NC}^1$

Key proof step: reduce CNF satisfiability to CQ answering in DL-Lite$_R$

- TBox generates full binary tree, leaves represent prop. valuations
  - depth of tree = number of variables
- tree-shaped query selects valuation, checks clauses are satisfied
  - number of leaves / branches in query = number of clauses

Depth of TBox =
maximum depth of generated trees in canonical model

- $\mathcal{T}$ has finite depth $\leftrightarrow$ applying axioms in $\mathcal{T}$ always terminates

Depth of TBox =
maximum depth of generated trees in canonical model

- $\mathcal{T}$ has finite depth $\leftrightarrow$ applying axioms in $\mathcal{T}$ always terminates

Does restricting the depth of TBoxes suffice for polysize rewritings?

Depth of TBox =
maximum depth of generated trees in canonical model

- $\mathcal{T}$ has finite depth $\leftrightarrow$ applying axioms in $\mathcal{T}$ always terminates

Does restricting the depth of TBoxes suffice for polysize rewritings?
Unfortunately not...

Depth of TBox = maximum depth of generated trees in canonical model

- $\mathcal{T}$ has finite depth $\leftrightarrow$ applying axioms in $\mathcal{T}$ always terminates

Does restricting the depth of TBoxes suffice for polysize rewritings?

Unfortunately not…

Depth 2 TBoxes:

- no polysize PE- or NDL-rewritings
- no polysize FO-rewritings unless $\mathsf{NP}/\mathrm{poly} \subseteq \mathsf{NC}^1$

Depth 1 TBoxes:

- no polysize PE- or NDL-rewritings
- no polysize FO-rewritings unless $\mathsf{NL}/\mathrm{poly} \subseteq \mathsf{NC}^1$

Depth of TBox =
maximum depth of generated trees in canonical model

- $\mathcal{T}$ has finite depth $\leftrightarrow$ applying axioms in $\mathcal{T}$ always terminates

Does restricting the depth of TBoxes suffice for polysize rewritings?

Unfortunately not...

Depth 2 TBoxes:

- no polysize PE- or NDL-rewritings
- no polysize FO-rewritings unless $\mathsf{NP}/\mathrm{poly} \subseteq \mathsf{NC}^1$

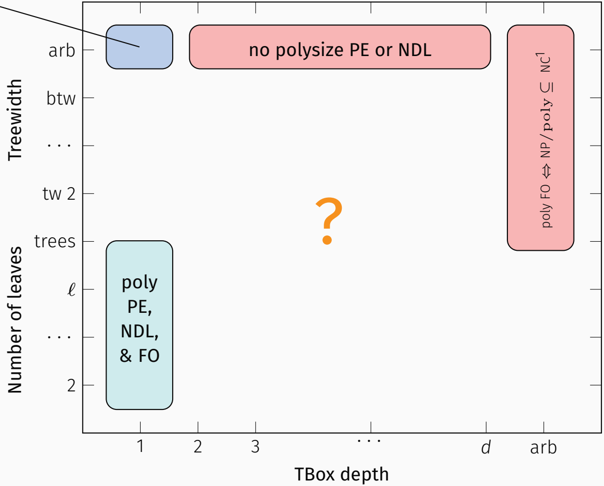
Depth 1 TBoxes:

- no polysize PE- or NDL-rewritings
- no polysize FO-rewritings unless $\mathsf{NL}/\mathrm{poly} \subseteq \mathsf{NC}^1$
- but: polysize PE-rewritings for tree-shaped queries

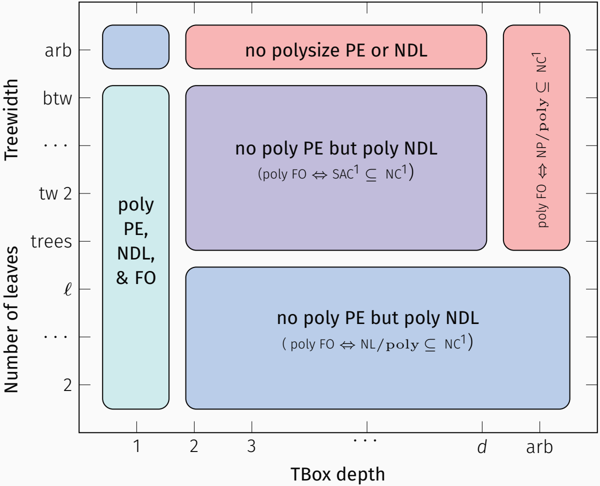
MAP OF RESULTS SO FAR

no poly PE but poly NDL

no poly FO unless NL/$\mathbf{poly} \subseteq$ NC[1]



COMPLETING THE LANDSCAPE [BKP15], [BKKPZ18]



Strong negative result for PE-rewritings

- no polysize PE-rewritings for depth 2 TBoxes + linear CQs

Conditional negative results for FO-rewritings

- polysize FO-rewritings exist iff
  - $\mathrm{SAC}^1 \subseteq \mathrm{NC}^1$ — bounded depth + bounded treewidth CQs
  - $\mathrm{NL}/\mathrm{poly} \subseteq \mathrm{NC}^1$ — bounded-leaf tree-shaped CQs

Positive results for NDL-rewritings

- bounded depth TBox + bounded treewidth CQs
- bounded-leaf tree-shaped CQs (+ arbitrary TBox)

Takeaway: NDL good target language for rewritings

BRIEF GLIMPSE AT PROOF TECHNIQUES (1)

Standard **computational complexity not the right tool**

- can be used to show no polytime-computable rewriting
- … but not that no polysize rewriting exists

Standard **computational complexity not the right tool**

- can be used to show no polytime-computable rewriting
- … but not that no polysize rewriting exists

Instead: establish **tight connections to circuit complexity**

- branch of complexity that classifies Boolean functions wrt. size / depth of Boolean circuits / formulas that compute them
- recall k-ary Boolean function maps tuples from $\{0,1\}^k$ to $\{0,1\}$

BRIEF GLIMPSE AT PROOF TECHNIQUES (1)

Standard **computational complexity not the right tool**

- can be used to show no polytime-computable rewriting
- … but not that no polysize rewriting exists

Instead: establish **tight connections to circuit complexity**

- branch of complexity that classifies Boolean functions wrt. size / depth of Boolean circuits / formulas that compute them
- recall k-ary Boolean function maps tuples from $\{0,1\}^k$ to $\{0,1\}$

Example: **function $\textsc{Reach}_n$**

- **input**: a Boolean vector representing the adjacency matrix of a directed graph G with n vertices including special vertices s and t
- **output**: 1 iff encoded graph G contains a directed path from s to t

BRIEF GLIMPSE AT PROOF TECHNIQUES (1)

Standard **computational complexity not the right tool**

- can be used to show no polytime-computable rewriting
- … but not that no polysize rewriting exists

Instead: establish **tight connections to circuit complexity**

- branch of complexity that classifies Boolean functions wrt. size / depth of Boolean circuits / formulas that compute them
- recall k-ary Boolean function maps tuples from $\{0,1\}^k$ to $\{0,1\}$

Example: **function $\textsc{Reach}_n$**

- **input**: a Boolean vector representing the adjacency matrix of a directed graph G with n vertices including special vertices s and t
- **output**: 1 iff encoded graph G contains a directed path from s to t

No family of polysize mon. Boolean formulas computing $\textsc{Reach}_n$

BRIEF GLIMPSE AT PROOF TECHNIQUES (2)

Types of rewritings $\rightsquigarrow$	ways of representing Boolean functions
PE-rewritings	monotone Boolean formulas $(\land, \lor)$
NDL-rewritings	monotone Boolean circuits ($\lor$- and $\land$-gates)
FO-rewritings	Boolean formulas $(\land, \lor, \neg)$

BRIEF GLIMPSE AT PROOF TECHNIQUES (2)

Types of rewritings $\rightsquigarrow$ ways of representing Boolean functions

PE-rewritings	monotone Boolean formulas $(\wedge, \vee)$
NDL-rewritings	monotone Boolean circuits ($\vee$- and $\wedge$-gates)
FO-rewritings	Boolean formulas $(\wedge, \vee, \neg)$

Associate Boolean functions with OMQ $(\mathcal{T}, q)$

BRIEF GLIMPSE AT PROOF TECHNIQUES (2)

Types of rewritings $\rightsquigarrow$	ways of representing Boolean functions
PE-rewritings	monotone Boolean formulas $(\wedge, \vee)$
NDL-rewritings	monotone Boolean circuits ($\vee$- and $\wedge$-gates)
FO-rewritings	Boolean formulas $(\wedge, \vee, \neg)$

Associate Boolean functions with OMQ $(\mathcal{T}, q)$

'Lower bound' function $f^{\mathsf{LB}}_{q,\mathcal{T}}$ $\Rightarrow$ lower bounds on rewriting size

- transform rewriting of $q, \mathcal{T}$ into formula / circuit that computes $f^{\mathsf{LB}}_{q,\mathcal{T}}$

'Upper bound' function $f^{\mathsf{UB}}_{q,\mathcal{T}}$ $\Rightarrow$ upper bounds on rewriting size

- transform formula / circuit that computes $f^{\mathsf{UB}}_{q,\mathcal{T}}$ into rewriting of $q, \mathcal{T}$

BRIEF GLIMPSE AT PROOF TECHNIQUES (2)

Types of rewritings $\rightsquigarrow$ ways of representing Boolean functions

PE-rewritings	monotone Boolean formulas $(\wedge, \vee)$
NDL-rewritings	monotone Boolean circuits ($\vee$- and $\wedge$-gates)
FO-rewritings	Boolean formulas $(\wedge, \vee, \neg)$

Associate Boolean functions with OMQ $(\mathcal{T}, q)$

'Lower bound' function $f^{\mathsf{LB}}_{q,\mathcal{T}}$ $\Rightarrow$ lower bounds on rewriting size

- transform rewriting of $q, \mathcal{T}$ into formula / circuit that computes $f^{\mathsf{LB}}_{q,\mathcal{T}}$

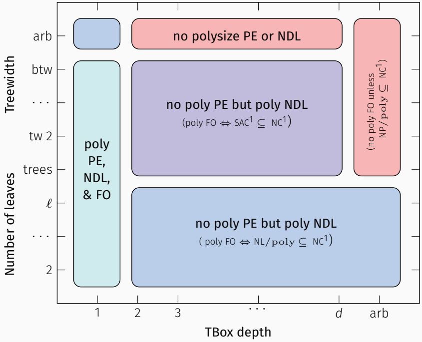
'Upper bound' function $f^{\mathsf{UB}}_{q,\mathcal{T}}$ $\Rightarrow$ upper bounds on rewriting size

- transform formula / circuit that computes $f^{\mathsf{UB}}_{q,\mathcal{T}}$ into rewriting of $q, \mathcal{T}$

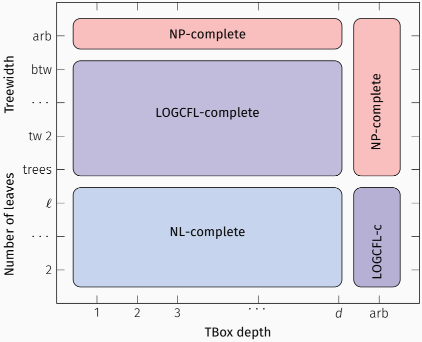
Exploit circuit complexity results about (in)existence of small formulas / circuits computing different classes of Boolean functions

- which functions expressible as $f^{\mathsf{LB}}_{q,\mathcal{T}}$ / $f^{\mathsf{UB}}_{q,\mathcal{T}}$ for given OMQ class?
  - intermediate computational model: hypergraph programs

Size of rewritings

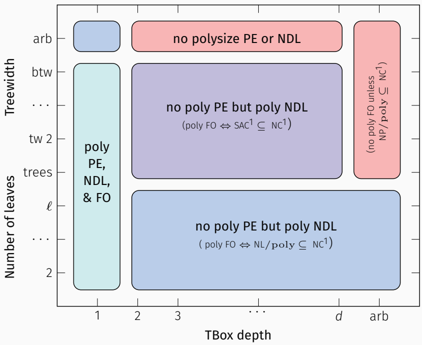


Combined complexity of OMQA

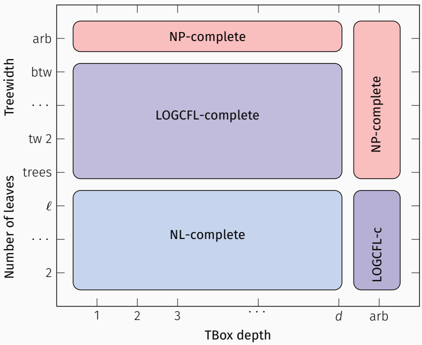


polysize NDL-rewritings ∼ polynomial (LOGCFL / NL) complexity

Size of rewritings



Combined complexity of OMQA



polysize NDL-rewritings ~ polynomial (LOGCFL / NL) complexity

Can we marry the positive succinctness & complexity results?

For the three well-behaved classes of OMQs, define **NDL-rewritings of optimal complexity**:

- rewriting can be constructed by L^C transducer
- evaluating the rewriting can be done in C

with $C \in \{\mathrm{NL}, \mathrm{LOGCFL}\}$ the complexity of the OMQ class

For the three well-behaved classes of OMQs, define **NDL-rewritings of optimal complexity**:

- rewriting can be constructed by L^C transducer
- evaluating the rewriting can be done in C

with $C \in \{\mathrm{NL}, \mathrm{LOGCFL}\}$ the complexity of the OMQ class

Preliminary experiments with **simple OMQs** (**depth 1, linear CQs**):

- compared with other NDL-rewritings (Clipper, Rapid, Presto)
- **our rewritings grow linearly** with increasing query size
- **other systems** produce rewritings that **grow exponentially**

EXISTENCE OF REWRITINGS

We have seen that:

- for $\mathcal{EL}$ ontologies, FO-rewritings need not exist
- for $\mathcal{ALC}$ ontologies, FO- and Datalog rewritings may not exist

But these are worst-case results

- only say that some OMQ that does not have a rewriting
- possible that rewritings exist for many OMQs encountered in practice

To extend the applicability of query rewriting beyond DL-Lite:

- devise ways of identifying 'good cases'
- construct rewritings when they exist

Use $(\mathcal{L}, \mathcal{Q})$ to denote set of **OMQs** $(\mathcal{T}, q)$ where:

- $\mathcal{T}$ is an $\mathcal{L}$-TBox
- q is a query from $\mathcal{Q}$ $\qquad \mathcal{Q} \in \{\mathrm{IQ}, \mathrm{CQ}\}$

For example: $(\mathcal{EL}, \mathrm{CQ})$, $(\mathcal{ALC}, \mathrm{IQ})$

FO-rewritability in $(\mathcal{L}, \mathcal{Q})$

- Input: OMQ $(\mathcal{T}, q)$ from $(\mathcal{L}, \mathcal{Q})$
- Problem: decide whether $(\mathcal{T}, q)$ has an FO-rewriting

Datalog-rewritability decision problem can be defined analogously

$\mathcal{EL} : \sqcap, \exists r.C$

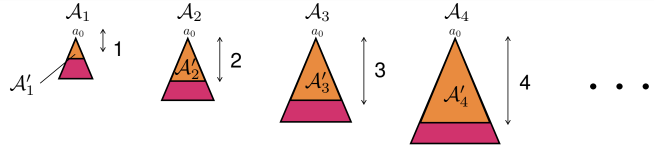
FO-rewritability is **EXPTIME-complete** in $(\mathcal{EL}, \mathrm{IQ})$ and $(\mathcal{EL}, \mathrm{CQ})$

$\mathcal{EL}$: $\sqcap, \exists r.C$

FO-rewritability is **EXPTIME-complete** in $(\mathcal{EL}, \mathrm{IQ})$ and $(\mathcal{EL}, \mathrm{CQ})$

Characterization of non-existence of FO-rewriting

OMQ $(\mathcal{T}, A(x))$ is **not FO-rewritable** iff there exist tree-shaped ABoxes



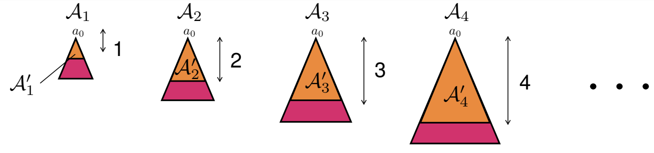
such that for all $i \geq 1$: $\mathcal{T}, \mathcal{A}_i \models A(a_0)$ and $\mathcal{T}, \mathcal{A}'_i \not\models A(a_0)$

$\mathcal{EL}: \sqcap, \exists r.C$

FO-rewritability is **EXPTIME-complete** in $(\mathcal{EL}, \mathrm{IQ})$ and $(\mathcal{EL}, \mathrm{CQ})$

Characterization of non-existence of FO-rewriting

OMQ $(\mathcal{T}, A(x))$ is **not FO-rewritable** iff there exist tree-shaped ABoxes



such that for all $i \geq 1$: $\mathcal{T}, \mathcal{A}_i \models A(a_0)$ and $\mathcal{T}, \mathcal{A}'_i \not\models A(a_0)$

Pumping argument: enough to find ABox of **particular finite size** k_0

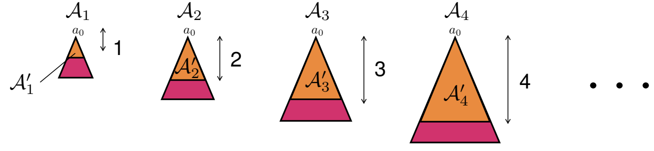
- desired ABox $\mathcal{A}_{k_0}$ exists $\Rightarrow$ can construct full sequence of ABoxes

$\mathcal{EL}: \sqcap, \exists r.C$

FO-rewritability is **EXPTIME-complete** in $(\mathcal{EL}, \mathrm{IQ})$ and $(\mathcal{EL}, \mathrm{CQ})$

Characterization of non-existence of FO-rewriting

OMQ $(\mathcal{T}, A(x))$ is **not FO-rewritable** iff there exist tree-shaped ABoxes



such that for all $i \geq 1$: $\mathcal{T}, \mathcal{A}_i \models A(a_0)$ and $\mathcal{T}, \mathcal{A}'_i \not\models A(a_0)$

Pumping argument: enough to find ABox of **particular finite size** k_0

- desired ABox $\mathcal{A}_{k_0}$ exists $\Rightarrow$ can construct full sequence of ABoxes

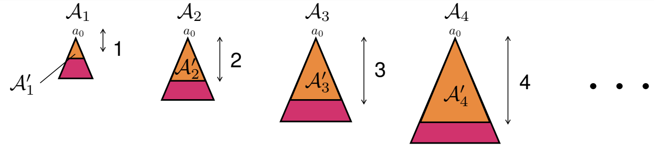
Use **tree automata** to check whether **such a witness ABox exists**

$\mathcal{EL}: \sqcap, \exists r.C$

FO-rewritability is **EXPTIME-complete** in $(\mathcal{EL}, \mathrm{IQ})$ and $(\mathcal{EL}, \mathrm{CQ})$

Characterization of non-existence of FO-rewriting

OMQ $(\mathcal{T}, A(x))$ is **not FO-rewritable** iff there exist tree-shaped ABoxes



such that for all $i \geq 1$: $\mathcal{T}, \mathcal{A}_i \models A(a_0)$ and $\mathcal{T}, \mathcal{A}'_i \not\models A(a_0)$

Pumping argument: enough to find ABox of **particular finite size** k_0

- desired ABox $\mathcal{A}_{k_0}$ exists $\Rightarrow$ can construct full sequence of ABoxes

Use **tree automata** to check whether **such a witness ABox exists**

Can **generalize this technique to handle CQs** as well

Idea for IQs: use **existing backwards-chaining rewriting procedure**

- if FO-rewriting does exist, terminates
- to ensure termination in general: use characterization result

To make practical: **decomposed algorithm**

- allows for structure sharing
- produces (succinct) NDL-rewriting instead of UCQ-rewriting

Experimental results are **very encouraging**:

- terminates quickly, produced rewritings are typically small
- suggests that in practice **FO-rewritings do exist** for **majority of IQs**

Recently **extended to handle CQs** with promising results

$\mathcal{ALC}$: $\neg, \sqcup, \sqcap, \exists r.C, \forall r.C$

FO-rewritability and Datalog-rewritability of $(\mathcal{ALC}, \mathrm{IQ})$ are both NEXPTIME-complete.

REWRITABILITY FOR (ALC, IQ) [BCLW13] [BCLW14]

$\mathcal{ALC}: \neg, \sqcup, \sqcap, \exists r.C, \forall r.C$

FO-rewritability and Datalog-rewritability of $(\mathcal{ALC}, \mathrm{IQ})$ are both NEXPTIME-complete.

Upper bound: connection to constraint satisfaction problems (CSPs)

- CSP($\mathfrak{B}$): decide if homomorphism from input structure $\mathcal{D}$ into $\mathfrak{B}$
- (Boolean) OMQs in $(\mathcal{ALC}, \mathrm{IQ}) \sim$ (complement of) CSPs
- exponential reduction to problem of deciding whether a CSP is definable in FO / Datalog
- use NP upper bounds for latter problems [LLT07] [FKKMMW09]

FO-REWRITABILITY FOR (ALC, UCQ) [FKL17]

FO-rewritability of $(\mathcal{ALC}, \mathrm{UCQ})$ is 2NEXPTIME-complete

FO-rewritability of $(\mathcal{ALC}, \mathrm{UCQ})$ is 2NExpTime-complete

Instead of CSP, uses MMSNP (monotone monadic strict NP):
fragment of monadic second-order logic that generalizes CSP

OMQs from $(\mathcal{ALC}, \mathrm{UCQ})$ $\sim$ complement of MMSNP formulas
$\sim$ monadic disjunctive Datalog [BCLW13] [BCLW14]

FO-rewritability of $(\mathcal{ALC}, \mathrm{UCQ})$ is 2NExpTime-complete

Instead of CSP, uses MMSNP (monotone monadic strict NP):
fragment of monadic second-order logic that generalizes CSP

OMQs from $(\mathcal{ALC}, \mathrm{UCQ})$ $\sim$ complement of MMSNP formulas
$\sim$ monadic disjunctive Datalog [BCLW13] [BCLW14]

FO-expressibility of (co)MMSNP not studied in CSP literature

FO-rewritability of $(\mathcal{ALC}, \mathrm{UCQ})$ is 2NExpTime-complete

Instead of CSP, uses MMSNP (monotone monadic strict NP):
fragment of monadic second-order logic that generalizes CSP

OMQs from $(\mathcal{ALC}, \mathrm{UCQ})$ $\sim$ complement of MMSNP formulas
$\sim$ monadic disjunctive Datalog [BCLW13] [BCLW14]

FO-expressibility of (co)MMSNP not studied in CSP literature

Recently: shown to be decidable and 2NExpTime-complete

CONCLUDING REMARKS

Ontology-mediated query answering:

- new paradigm for intelligent information systems
- offers many advantages, but also computational challenges

Query rewriting promising algorithmic approach

Many interesting problems related to OMQA and query rewriting:

- succinctness of rewritings (Boolean functions, circuit complexity)
- existence of FO and Datalog rewritings (automata, CSP / MMSNP)
- other tools: parameterized complexity, word rewriting

Active area with lots left to explore!

Questions?

Joint work with:

Balder ten Cate, Peter Hansen, Carsten Lutz, Stanislav Kikot,

Roman Kontchakov, Vladimir Podolskii, Vladislav Ryzhikov,

Frank Wolter, and Michael Zakharyaschev

[KKPZ12] S. Kikot, R. Kontchakov, V. Podolskii, and M. Zakharyaschev: Exponential Lower Bounds and Separation for Query Rewriting. 39th International Colloquium on Automata, Languages, and Programming (**ICALP'12**), 2012.

[GS12] G. Gottlob and T. Schwentick: Rewriting Ontological Queries into Small Nonrecursive Datalog Programs. 13th International Conference on the Principles of Knowledge Representation and Reasoning (**KR'12**), 2012.

[GKKPSZ14] G. Gottlob, S. Kikot, R. Kontchakov, V. Podolskii, T. Schwentick, and M. Zakharyaschev: The Price of Query Rewriting in Ontology-based Data Access. Artificial Intelligence (**AIJ**), 2014.

[KKPZ14] S. Kikot, R. Kontchakov, V. Podolskii, and M. Zakharyaschev: On the Succinctness of Query Rewriting over Shallow Ontologies. 29th Annual ACM/IEEE Symposium on Logic in Computer Science (**LICS'14**), 2014.

[BKP15] M. Bienvenu, S. Kikot, V. Podolskii: Tree-like Queries in OWL 2 QL: Succinctness and Complexity Results. 30th Annual ACM/IEEE Symposium on Logic in Computer Science (**LICS'15**), 2015.

[BKKPRZ17] M. Bienvenu, S. Kikot, R. Kontchakov, V. Podolskii, V. Ryzhikov and M. Zakharyaschev: The Complexity of Ontology-Based Data Access with OWL 2 QL and Bounded Treewidth Queries. Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (**PODS'17**), 2017.

[BKKPZ18] M. Bienvenu, S. Kikot, R. Kontchakov, V. Podolskii, and M. Zakharyaschev: Ontology-Mediated Queries: Combined Complexity and Succinctness of Rewritings via Circuit Complexity. Journal of the ACM (**JACM**), 2018.

REFERENCES: EXISTENCE OF REWRITINGS

[BCLW13] M. Bienvenu, B. ten Cate, C. Lutz, and F. Wolter: Ontology-based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP. 32nd International Conference on the Principles of Database Systems (**PODS'13**), 2013.

[BLW13] M. Bienvenu, C. Lutz, and F. Wolter: First Order-Rewritability of Atomic Queries in Horn Description Logics. 23rd International Joint Conference on Artificial Intelligence (**IJCAI'13**), 2013.

[BCLW14] M. Bienvenu, B. ten Cate, C. Lutz, and F. Wolter: Ontology-based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP. Transactions on Database Systems (**TODS**), 2014.

[KNG14] M. Kaminski, Y. Nenov, and B. Cuenca Grau: Datalog Rewritability of Disjunctive Datalog Programs and its Applications to Ontology Reasoning. 28th AAAI Conference on Artificial Intelligence (**AAAI'14**), 2014.

[HLSW15] P. Hansen, C. Lutz, I. Seylan, and F. Wolter: Efficient Query Rewriting in the Description Logic EL and Beyond. 24th International Joint Conference on Artificial Intelligence (**IJCAI'15**), 2015.

[BL16] P. Bourhis and C. Lutz: Containment in Monadic Disjunctive Datalog, MMSNP, and Expressive Description Logics. 15th International Conference on the Principles of Knowledge Representation and Reasoning (**KR'16**), 2016.

[BCLW16] M. Bienvenu, P. Hansen, C. Lutz, and F. Wolter: First Order-Rewritability and Containment of Conjunctive Queries in Horn Description Logics. 25th International Joint Conference on Artificial Intelligence (**IJCAI'16**), 2016.

[FKL17] C. Feier, A. Kuusisto, and C. Lutz: Rewritability in Monadic Disjunctive Datalog, MMSNP, and Expressive Description Logics. 20th International Conference on Database Theory (**ICDT'17**), 2017.

[HL17] P. Hansen and C. Lutz: Computing FO-Rewritings in $\mathcal{EL}$ in Practice: from Atomic to Conjunctive Queries. 16th International Semantic Web Conference (**ISWC'17**), 2017.

REFERENCES: DEFINABILITY OF CSPS

[LLT07] B. Larose, C. Loten, and C. Tardif. A characterisation of first-order constraint satisfaction problems. Logical Methods in Computer Science (**LMCS**), 2007.

[FKKMMW09] R. Freese, M. Kozik, A. Krokhin, M. Maroti, R. Mckenzie, and R. Willard. On maltsev conditions associated with omitting certain types of local structures. Available at: http://www.math.hawaii. edu/∼ralph/Classes/619/OmittingTypesMaltsev.pdf, 2009.

[CL17] H. Chen and B. Larose. Asking the Metaquestions in Constraint Tractability. ACM Transactions on Computation Theory (**TOCT**) 9(3), pages 1-27, 2017.