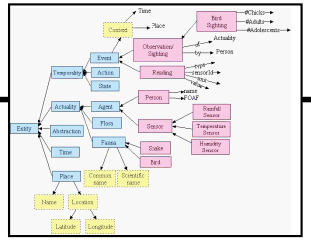
QUERY REWRITING:
Limits and Possibilities

Meghyn Bienvenu (*CNRS, University of Montpellier, Inria*)

ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)

incomplete database

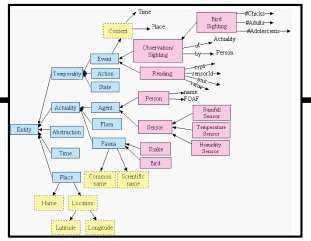


ontology

domain knowledge

user query

ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)

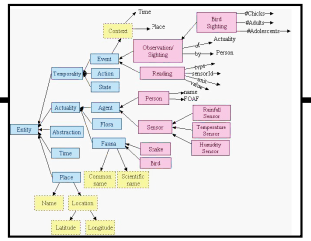


patient data
"Melanie has listeriosis"
"Paul has Lyme disease"

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)

SETTING WE CONSIDER

Ontologies formulated using description logics (DLs):

- family of decidable fragments of first-order logic
- basis for OWL web ontology language (W3C)
- range from fairly simple to highly expressive
- complexity of query answering well understood

In this talk, mainly focus on three particular DLs:

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

Consider two types of queries:

- conjunctive queries (CQs) - aka select-project-join queries
- instance queries (IQs)

BRIEF INTRO TO DLS & OMQA

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 ($\exists$, $\forall$)

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 ($\exists$, $\forall$)

An $\mathcal{ALC}$ TBox (ontology) is a set of concept inclusions $C \sqsubseteq D$, where

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

where A is an atomic concept, r an atomic role.

Intuitively, $C \sqsubseteq D$ means "everything that is a C is also a D"

EXAMPLES OF TBOX AXIOMS

Professors and lecturers are disjoint classes of faculty

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

Every grad student is supervised by a professor

$$\mathsf{GradSt} \sqsubseteq \exists \mathsf{supervisedBy}.\mathsf{Prof}$$

Grad students are students, and they only take graduate courses

$$\mathsf{GradSt} \sqsubseteq \mathsf{Student} \sqcap \forall \mathsf{takesC}.\mathsf{GradC}$$

FO translation: $\forall x\ (\mathsf{GradSt}(x) \rightarrow (\mathsf{Student}(x) \wedge \forall y\ \mathsf{takesC}(x, y) \rightarrow \mathsf{GradC}(y))$

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 many biomedical ontologies, including SNOMED
- importance witnessed by OWL 2 EL profile

Also consider $\mathcal{ELI}$ = $\mathcal{EL}$ + inverse roles (r^-)

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

DL-Lite$_R$ TBoxes contain

- 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

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

Example TBox inclusions:

- Every professor teaches something: $\mathsf{Prof} \sqsubseteq \exists \mathsf{teaches}$
- Everything that is taught is a course: $\exists \mathsf{teaches}^- \sqsubseteq \mathsf{Course}$
- Head of dept implies member of dept: $\mathsf{headOf} \sqsubseteq \mathsf{memberOf}$

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

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

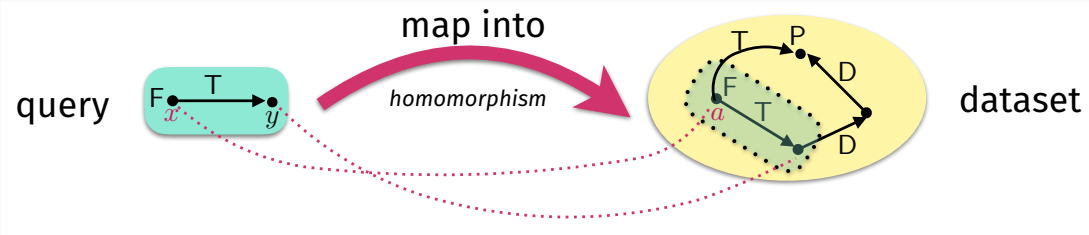
Conjunctive queries (CQs) $\sim$ SPJ queries in SQL, BGPs in SPARQL

conjunctions of atoms, some variables can be existentially quantified

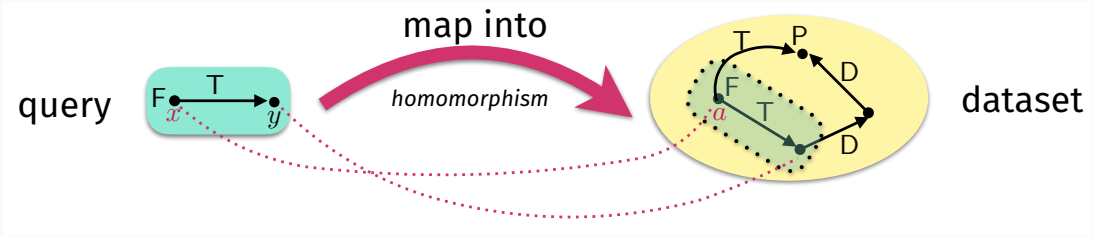
$$\exists y.\, \mathrm{Faculty}(x) \land \mathrm{teaches}(x, y)$$

(find all faculty members that teach something)

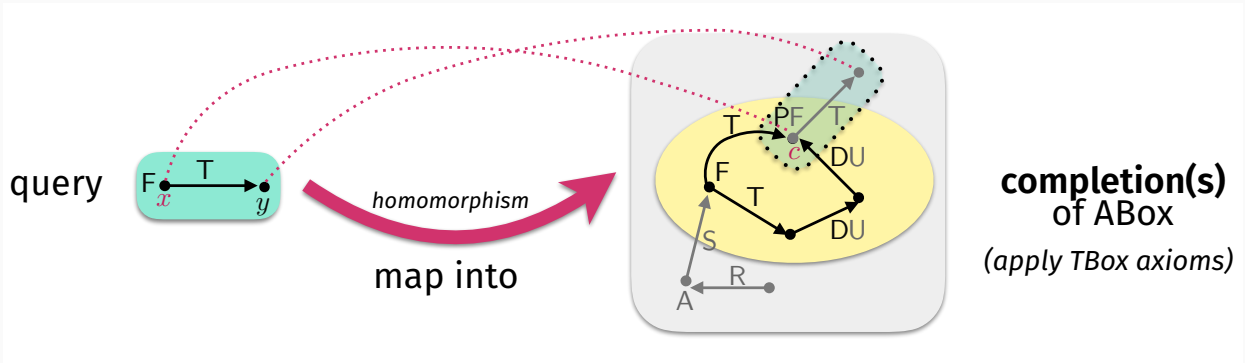
Answering CQs in database setting



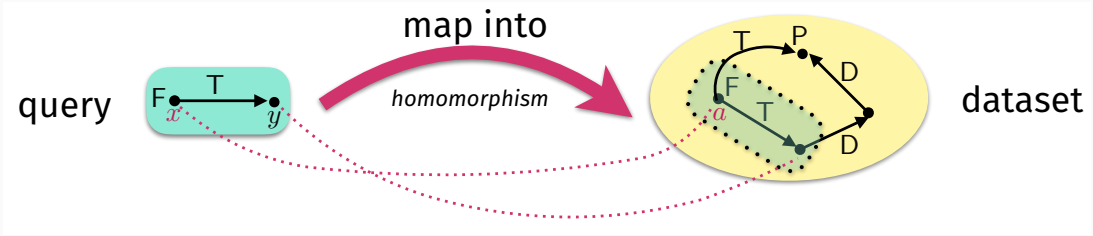
Answering CQs in database setting



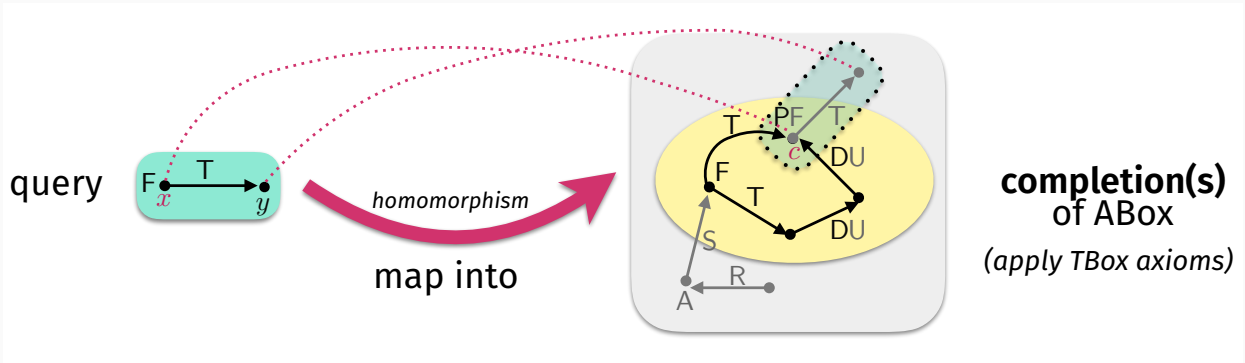
Answering CQs in the presence of a TBox (ontology)



Answering CQs in database setting

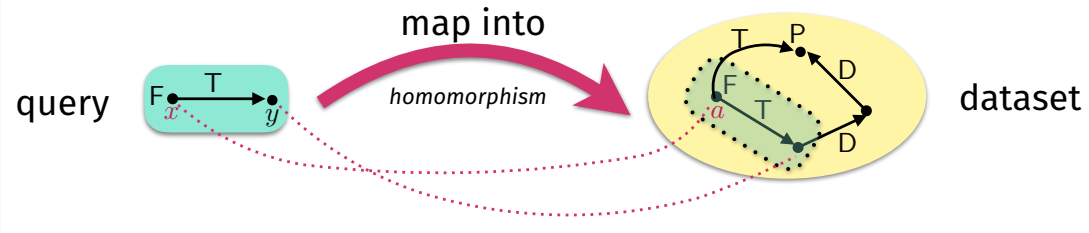


Answering CQs in the presence of a TBox (ontology)

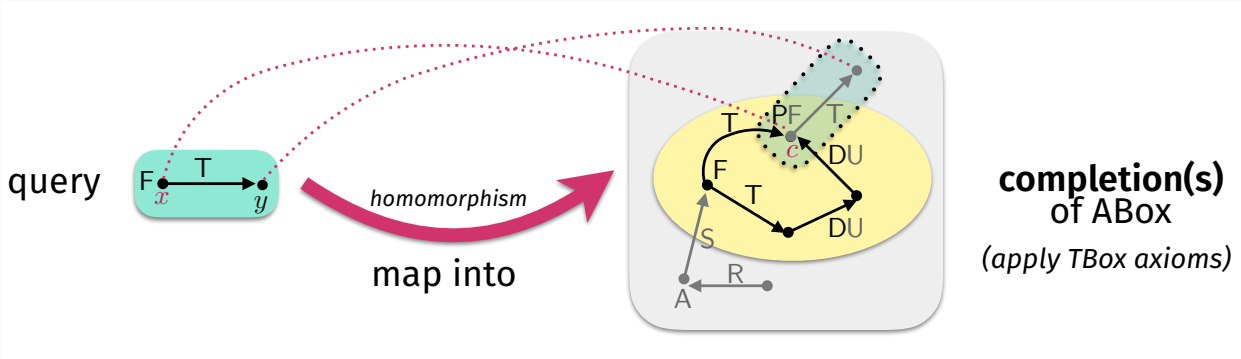


Certain answers: tuples $\vec{a}$ of individuals such that $\mathcal{T}, \mathcal{A} \models q(\vec{a})$

Answering CQs in database setting



Answering CQs in the presence of a TBox (ontology)



Certain answers: tuples $\vec{a}$ of individuals such that $\mathcal{T}, \mathcal{A} \models q(\vec{a})$

Ontology-mediated query (OMQ): pair $(\mathcal{T}, q)$ with $\mathcal{T}$ a TBox, q a query

QUERY REWRITING

QUERY REWRITING

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

QUERY REWRITING

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 FO-rewriting

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

Example:

TBox $\mathcal{T} = \{ \exists supervises \sqsubseteq Prof \; supervises \sqsubseteq involved \; 100S \sqsubseteq IntroC \}$

Query $q_0 = Prof(x) \wedge involved(x, y) \wedge IntroC(y)$

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

Example:

TBox $\mathcal{T} = \{ \exists supervises \sqsubseteq Prof \; supervises \sqsubseteq involved \; 100S \sqsubseteq IntroC \}$

Query $q_0 = Prof(x) \wedge involved(x, y) \wedge IntroC(y)$

Get FO-rewriting by taking disjunction of q_0 and following queries:

$$q_1 = \exists z \, supervises(x, z) \wedge involved(x, y) \wedge IntroC(y)$$
$$q_2 = supervises(x, y) \wedge IntroC(y)$$
$$q_3 = Prof(x) \wedge involved(x, y) \wedge 100S(y)$$
$$q_5 = \exists z \, supervises(x, z) \wedge involved(x, y) \wedge 100S(y)$$
$$q_6 = supervises(x, y) \wedge 100S(y)$$

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

Example:

TBox $\mathcal{T} = \{\exists supervises \sqsubseteq Prof \; supervises \sqsubseteq involved \; 100S \sqsubseteq IntroC\}$

Query $q_0 = Prof(x) \wedge involved(x, y) \wedge IntroC(y)$

Get FO-rewriting by taking disjunction of q_0 and following queries:

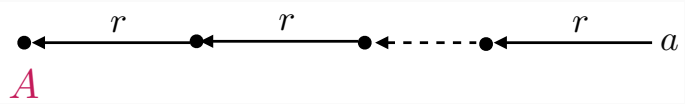
$$q_1 = \exists z \, supervises(x, z) \wedge involved(x, y) \wedge IntroC(y)$$
$$q_2 = supervises(x, y) \wedge IntroC(y)$$
$$q_3 = Prof(x) \wedge involved(x, y) \wedge 100S(y)$$
$$q_5 = \exists z \, supervises(x, z) \wedge involved(x, y) \wedge 100S(y)$$
$$q_6 = supervises(x, y) \wedge 100S(y)$$

Note: existence of FO-rewritings $\Rightarrow$ very low data complexity (AC_0)

WHAT ABOUT EL?

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

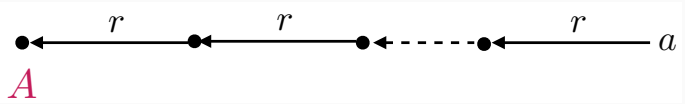
- no FO-rewriting of $A(x)$ w.r.t. $\{\exists r.A \sqsubseteq A\}$



WHAT ABOUT EL?

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

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



However, Datalog rewritings always exist:

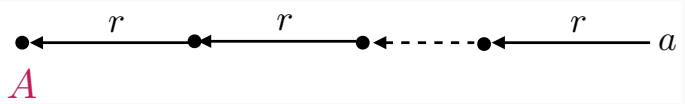
- Datalog program Π: $r(x,y) \wedge A(x) \rightarrow A(y)$ $A(x) \rightarrow \text{goal}(x)$
- $\mathcal{T}, \mathcal{A} \models A(a)$ iff can derive $\text{goal}(a)$ from $\mathcal{A}$ using Π

Can pass on rewriting to Datalog engine

WHAT ABOUT EL?

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

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



However, Datalog rewritings always exist:

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

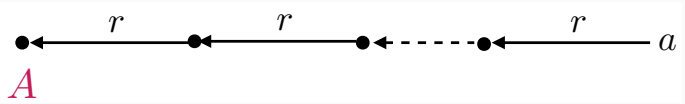
Can pass on rewriting to Datalog engine

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

WHAT ABOUT EL?

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

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



However, Datalog rewritings always exist:

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

Can pass on rewriting to Datalog engine

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

Note: also get Datalog rewritings for many extensions of $\mathcal{EL}$

WHAT ABOUT ALC?

Neither FO nor Datalog rewritings need exist

Culprit: presence of disjunction

WHAT ABOUT ALC?

Neither FO nor Datalog rewritings need exist

Culprit: presence of disjunction

Encoding of non-3-colourability:

TBox axioms:

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

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

UNDERSTANDING QUERY REWRITING

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. Optimality of rewritings — DL-Lite
   - Can we achieve optimal complexity via query rewriting?

3. 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)

Lots of rewriting algorithms for DL-Lite designed and tested

Most produce unions of conjunctive queries (UCQs)

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)

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: $A_1^0(x) \land \ldots \land A_n^0(x)$
- 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: $\bigvee_{(i_1,\ldots,i_n)\in\{0,1\}} A_1^{i_1}(x) \land A_1^{i_1}(x) \land \ldots \land A_1^{i_1}(x)$

Lots of rewriting algorithms for DL-Lite designed and tested

Most produce unions of conjunctive queries (UCQs)

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: $A_1^0(x) \land \ldots \land A_n^0(x)$
- 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: $\bigvee_{(i_1,\ldots,i_n)\in\{0,1\}} A_1^{i_1}(x) \land A_1^{i_1}(x) \land \ldots \land A_1^{i_1}(x)$

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

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

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

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

$$\begin{aligned}
\mathrm{goal}(x,y) &\leftarrow q_1(x,y), q_2(x), q_2(y) \\
q_1(x,y) &\leftarrow r(x,y) & q_2(x) &\leftarrow A(x) \\
q_1(x,y) &\leftarrow s(y,x) & q_2(x) &\leftarrow B(x), p(x,z)
\end{aligned}$$

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

$$\begin{aligned} \mathrm{goal}(x,y) &\leftarrow q_1(x,y), q_2(x), q_2(y) \\ q_1(x,y) &\leftarrow r(x,y) \qquad & q_2(x) &\leftarrow A(x) \\ q_1(x,y) &\leftarrow s(y,x) & q_2(x) &\leftarrow B(x), p(x,z) \end{aligned}$$

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

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

$$\begin{aligned} \mathrm{goal}(x,y) &\leftarrow q_1(x,y), q_2(x), q_2(y) \\ q_1(x,y) &\leftarrow r(x,y) \qquad & q_2(x) &\leftarrow A(x) \\ q_1(x,y) &\leftarrow s(y,x) & q_2(x) &\leftarrow B(x), p(x,z) \end{aligned}$$

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

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

(Note: focus on so-called pure rewritings - no special constants)

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

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

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 / chase

- $\mathcal{T}$ has finite depth $\leftrightarrow$ chase terminates for every KB $(\mathcal{T}, \mathcal{A})$

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

- $\mathcal{T}$ has finite depth $\leftrightarrow$ chase terminates for every KB $(\mathcal{T}, \mathcal{A})$

Does restricting the depth of TBoxes suffice for polysize rewritings?

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

- $\mathcal{T}$ has finite depth $\leftrightarrow$ chase terminates for every KB $(\mathcal{T}, \mathcal{A})$

Does restricting the depth of TBoxes suffice for polysize rewritings?

Unfortunately not...

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

- $\mathcal{T}$ has finite depth $\leftrightarrow$ chase terminates for every KB $(\mathcal{T}, \mathcal{A})$

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 NP/poly $\subseteq$ NC^1

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

- $\mathcal{T}$ has finite depth $\leftrightarrow$ chase terminates for every KB $(\mathcal{T}, \mathcal{A})$

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 $\mathrm{NP}/\mathrm{poly} \subseteq \mathrm{NC}^1$

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

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

- $\mathcal{T}$ has finite depth $\leftrightarrow$ chase terminates for every KB $(\mathcal{T}, \mathcal{A})$

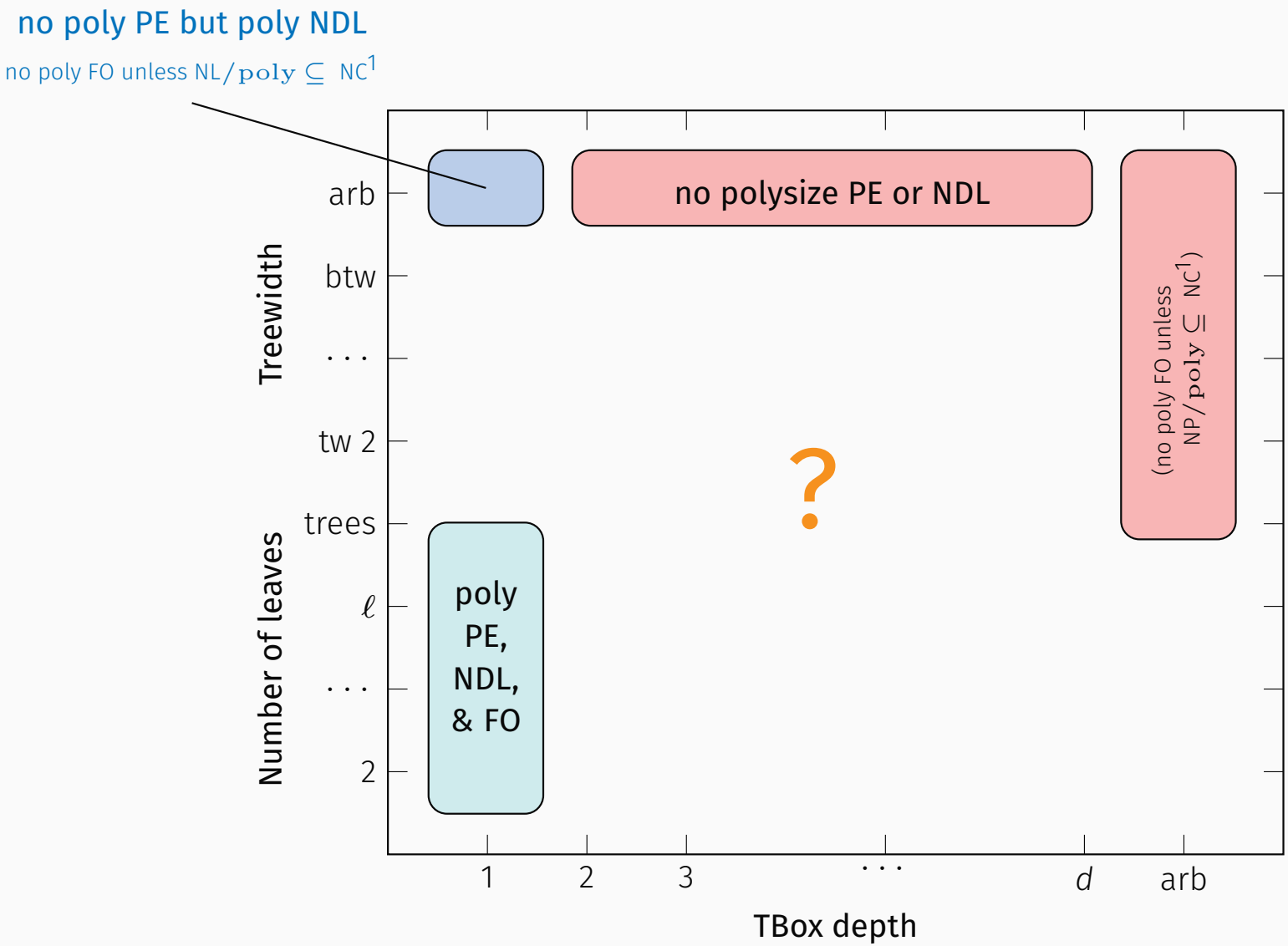
Does restricting the depth of TBoxes suffice for polysize rewritings?

Unfortunately not...

Depth 2 TBoxes:
- no polysize PE- or NDL-rewritings
- no polysize FO-rewritings unless $\mathrm{NP}/\mathrm{poly} \subseteq \mathrm{NC}^1$

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

MAP OF RESULTS SO FAR



GAPS IN THE LANDSCAPE

Two dimensions:

- type of TBox — *depth* 1, *depth* 2, ..., *arbitrary*
- type of query — *tree-shaped* / *arbitrary*

GAPS IN THE LANDSCAPE

Two dimensions:

- type of TBox — *depth* 1, *depth* 2, ..., *arbitrary*
- type of query — *tree-shaped* / *arbitrary*

What about tree-shaped queries & depth k TBoxes ($k > 1$) ?

GAPS IN THE LANDSCAPE

Two dimensions:

- type of TBox — *depth 1, depth 2, ..., arbitrary*
- type of query — *tree-shaped / arbitrary*

What about tree-shaped queries & depth k TBoxes ($k > 1$) ?

What about bounded treewidth queries?

GAPS IN THE LANDSCAPE

Two dimensions:

- type of TBox — *depth 1, depth 2, ..., arbitrary*
- type of query — *tree-shaped / arbitrary*

What about tree-shaped queries & depth k TBoxes ($k > 1$) ?

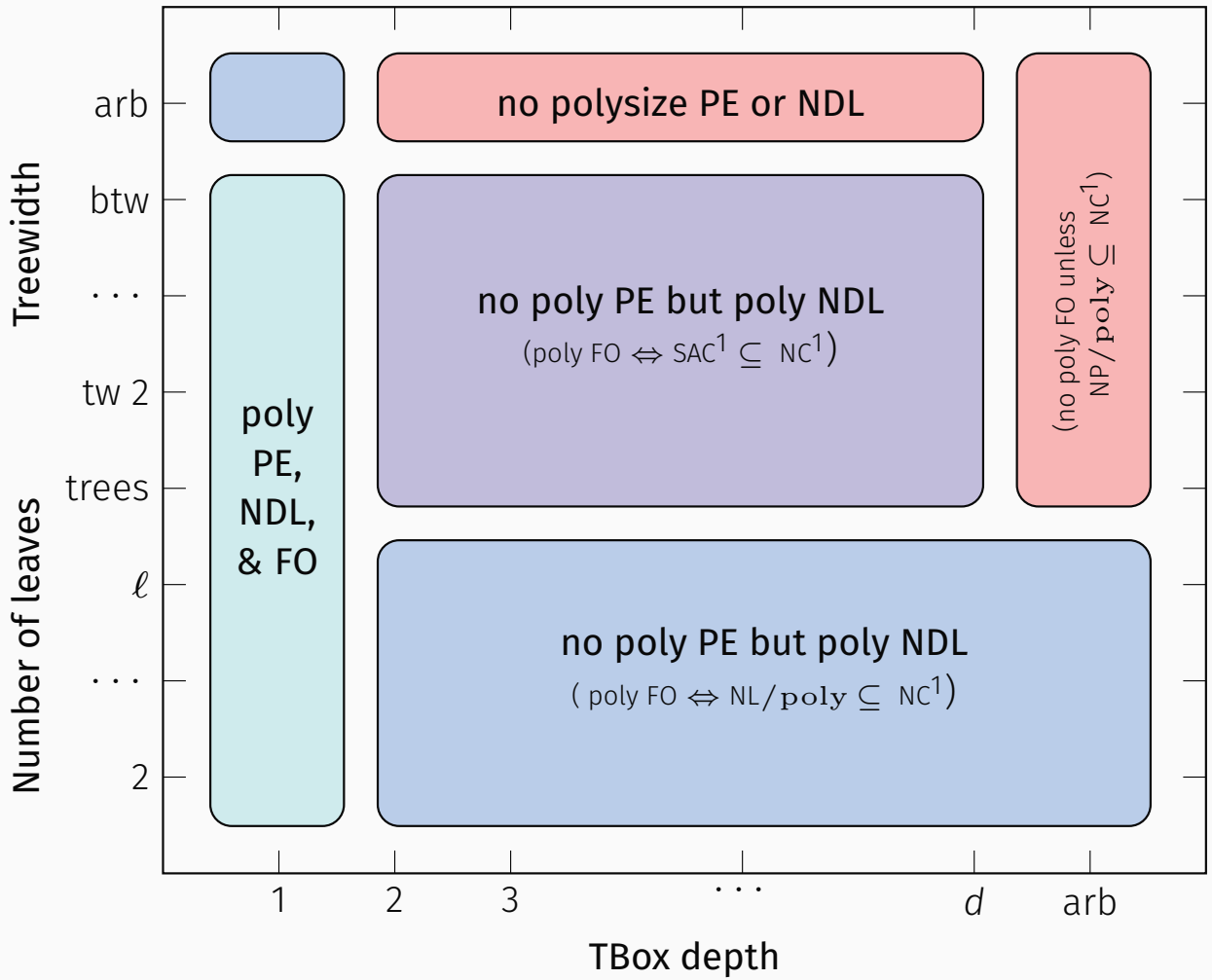
What about bounded treewidth queries?

What about restricted classes of tree-shaped queries?

- linear queries
- tree-shaped queries with fixed number of leaves

COMPLETING THE LANDSCAPE

[BKP15]



Bounded depth TBox + bounded treewidth CQs

- no polysize PE-rewritings
- polysize NDL-rewritings do exist
- no polysize FO-rewritings (unless $\mathsf{SAC}^1 \subseteq \mathsf{NC}^1$)

Tree-shaped CQs with bounded number of leaves

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

Negative results hold already for: depth 2 + linear queries

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

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 $\{0,1\}$

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)

Associate Boolean functions with query-TBox pair $(q, \mathcal{T})$

Associate Boolean functions with query-TBox pair $(q, \mathcal{T})$

Primitive evaluation function $f^{\text{prim}}_{q,\mathcal{T}}$

- input vector $\vec{u} \sim$ **single-individual ABox** $\mathcal{A}_{\vec{u}}$ ind a
  - p_A means concept A is present, p_r means r self-loop
- $f^{\text{prim}}_{q,\mathcal{T}}(\vec{u}) = 1$ **if** $(\mathcal{T}, \mathcal{A}_{\vec{u}}) \models q(a, \ldots, a)$

BRIEF GLIMPSE AT PROOF TECHNIQUES (2)

Associate Boolean functions with query-TBox pair $(q, \mathcal{T})$

Primitive evaluation function $f^{\text{prim}}_{q,\mathcal{T}}$

- input vector $\vec{u} \sim$ **single-individual ABox** $\mathcal{A}_{\vec{u}}$ ind a
  - p_A means concept A is present, p_r means r self-loop
- $f^{\text{prim}}_{q,\mathcal{T}}(\vec{u}) = 1$ **if** $(\mathcal{T}, \mathcal{A}_{\vec{u}}) \models q(a, \ldots, a)$

Tree-witness hypergraph function $f^{\text{tw}}_{q,\mathcal{T}}$:

- input vector $\vec{u} \sim$ **abstract description of how query mapped into canonical model**
  - p_α means atom α mapped into ABox
  - p_{t} means tree-witness (subquery) t mapped to existential part
- $f^{\text{tw}}_{q,\mathcal{T}}(\vec{u}) = 1$ **if** $\vec{u}$ **describes a partition of the atoms in** q

BRIEF GLIMPSE AT PROOF TECHNIQUES (3)

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

PE-rewritings	monotone Boolean formulas
NDL-rewritings	monotone Boolean circuits
FO-rewritings	Boolean formulas

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

PE-rewritings	monotone Boolean formulas
NDL-rewritings	monotone Boolean circuits
FO-rewritings	Boolean formulas

Primitive evaluation function $\Rightarrow$ lower bounds on rewriting size

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

Tree-witness hypergraph func. $\Rightarrow$ upper bounds on rewriting size

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

BRIEF GLIMPSE AT PROOF TECHNIQUES (3)

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

PE-rewritings	monotone Boolean formulas
NDL-rewritings	monotone Boolean circuits
FO-rewritings	Boolean formulas

Primitive evaluation function $\Rightarrow$ lower bounds on rewriting size

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

Tree-witness hypergraph func. $\Rightarrow$ upper bounds on rewriting size

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

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

OPTIMALITY OF REWRITINGS

WHAT DOES ALL THIS MEAN FOR COMPLEXITY OF QUERY ANSWERING?

WHAT DOES ALL THIS MEAN FOR COMPLEXITY OF QUERY ANSWERING?

Actually, not much!

Small rewritings do not guarantee low complexity

- need to consider cost of producing and evaluating the rewriting

WHAT DOES ALL THIS MEAN FOR COMPLEXITY OF QUERY ANSWERING?

Actually, not much!

Small rewritings do not guarantee low complexity

- need to consider cost of producing and evaluating the rewriting

Large rewritings do not guarantee high complexity

- maybe query rewriting is not the most efficient approach

WHAT DOES ALL THIS MEAN FOR COMPLEXITY OF QUERY ANSWERING?

Actually, not much!

Small rewritings do not guarantee low complexity

- need to consider cost of producing and evaluating the rewriting

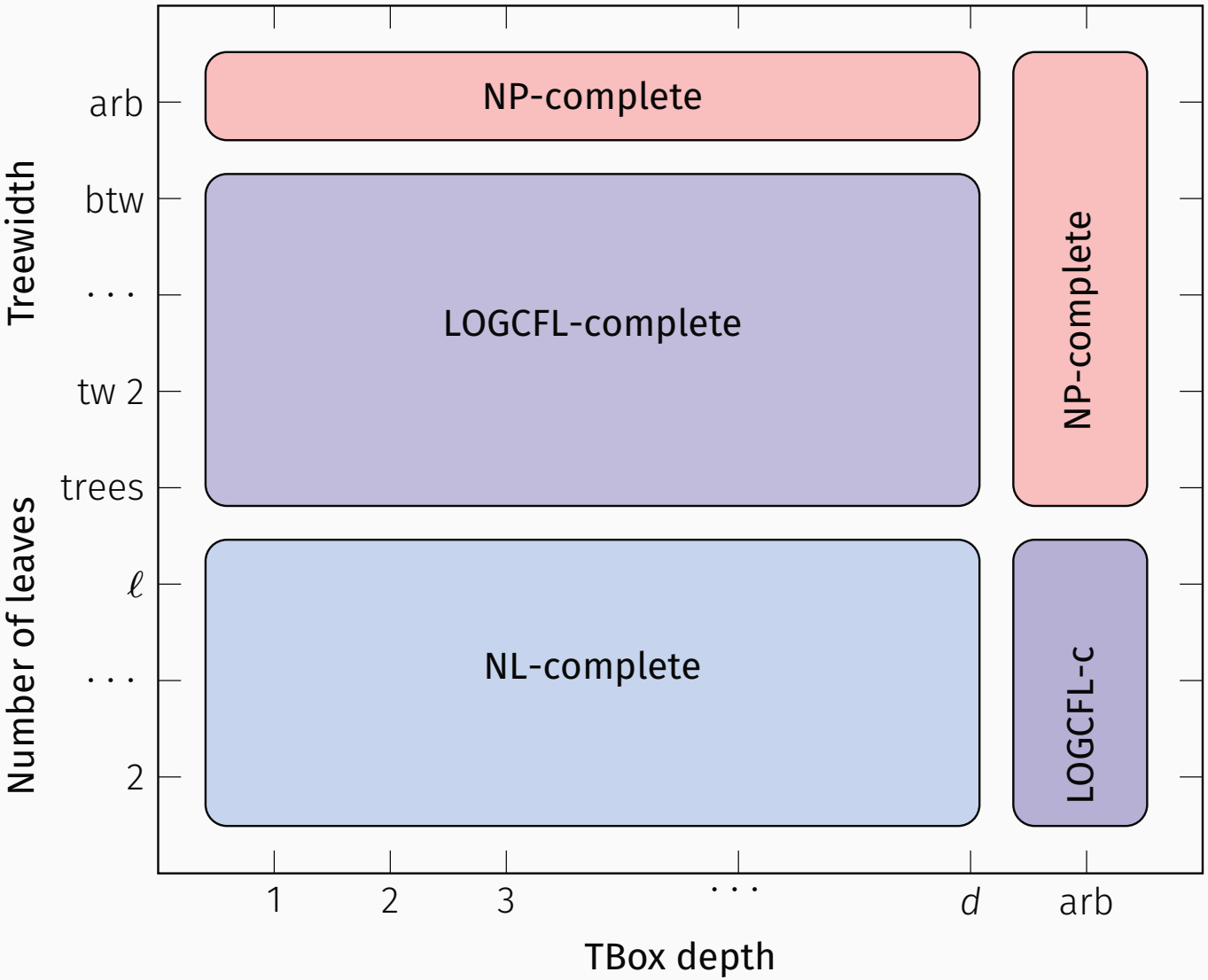
Large rewritings do not guarantee high complexity

- maybe query rewriting is not the most efficient approach

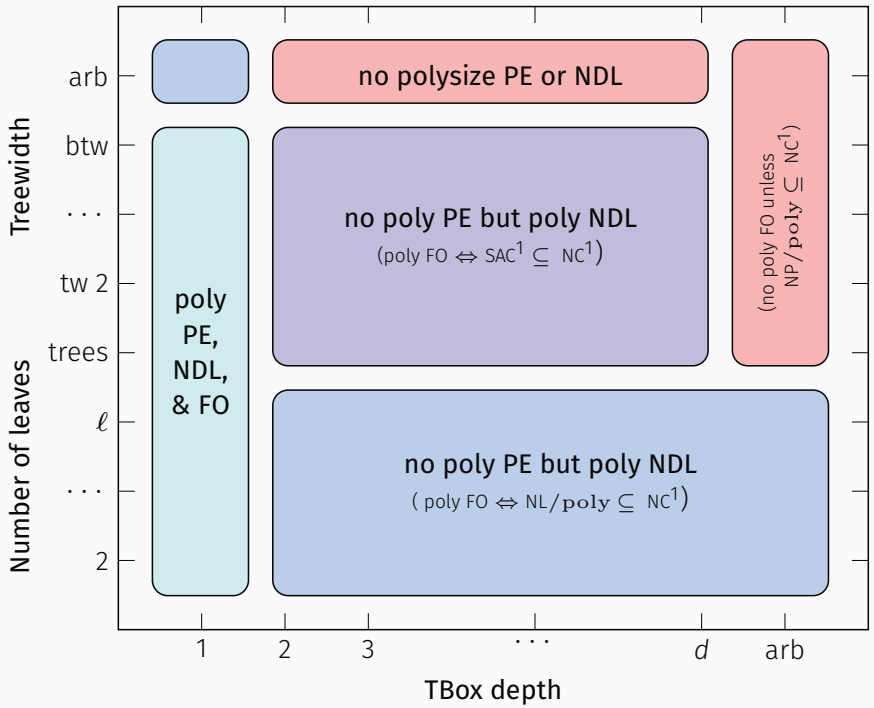
Motivated the study of the complexity landscape of query answering

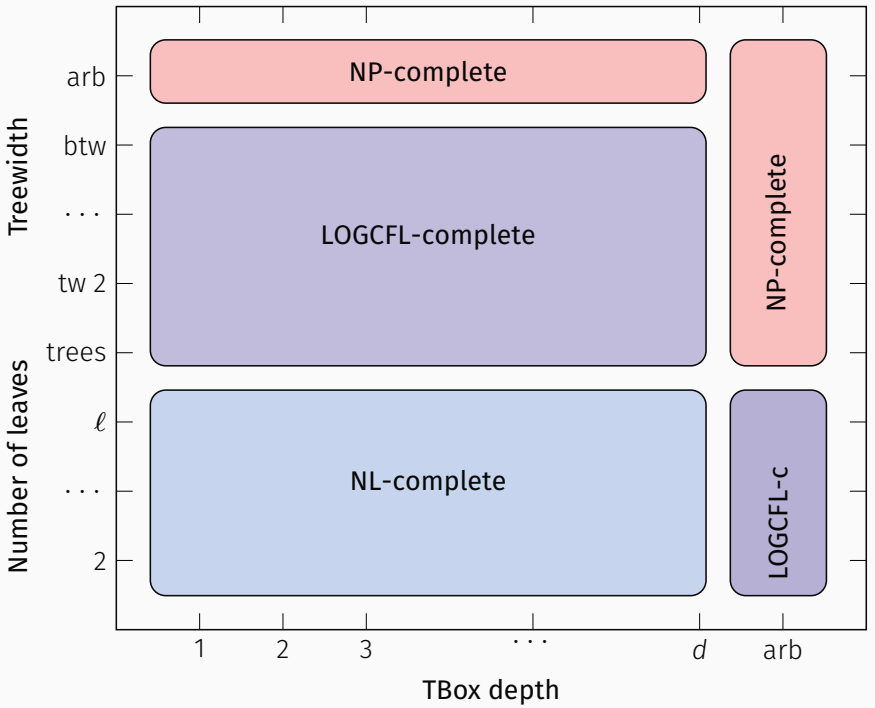
Consider combined complexity (data complexity is same in all cases)

COMBINED COMPLEXITY LANDSCAPE FOR DL-LITE [BKP15]



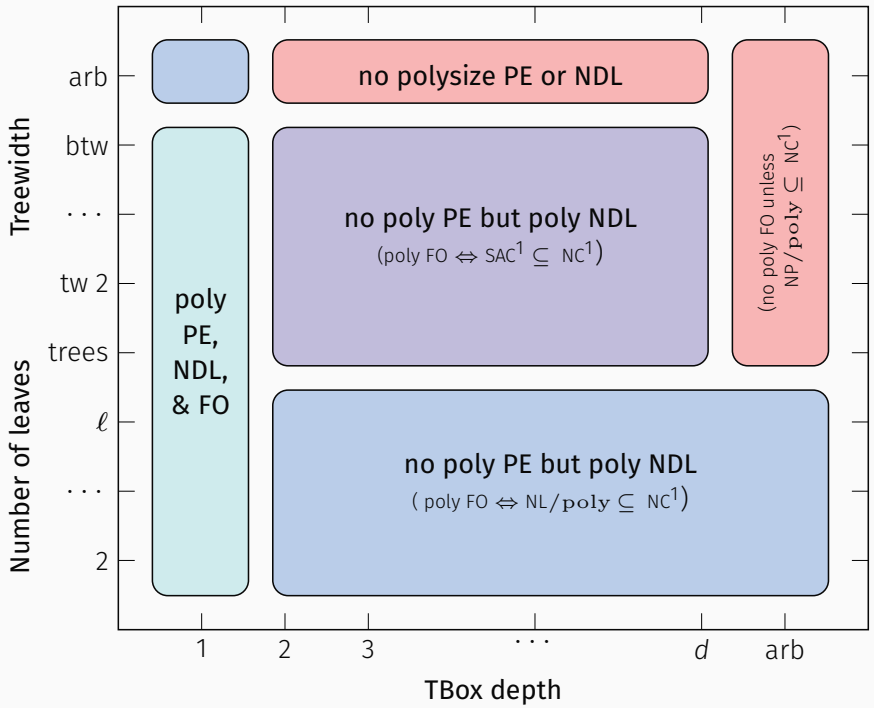
COMPARING THE LANDSCAPES

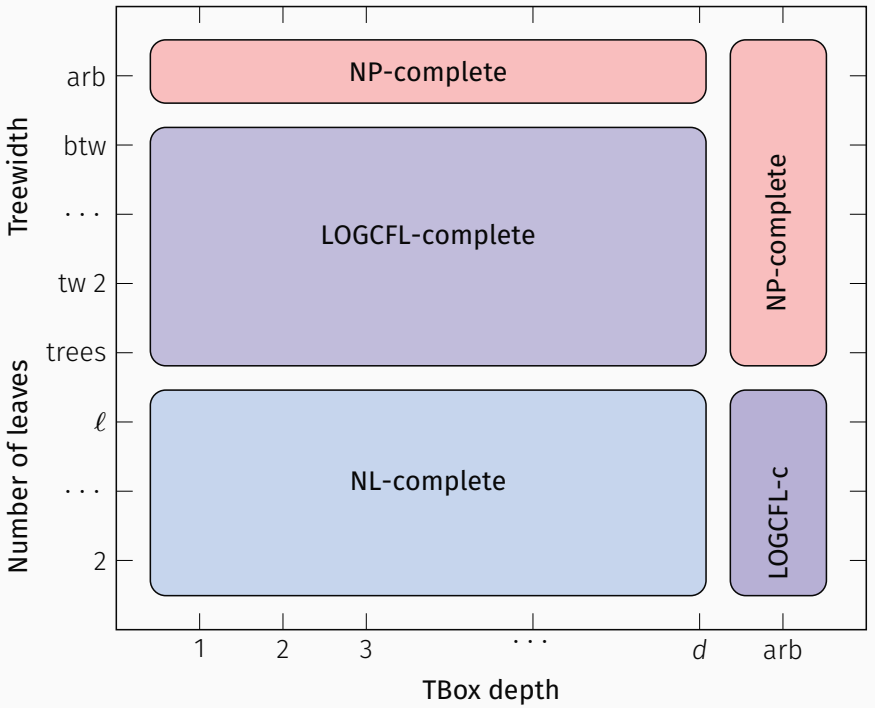




polysize NDL-rewritings ∼ polynomial (LOGCFL / NL) complexity

COMPARING THE LANDSCAPES





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 $C \in \{\mathsf{NL}, \mathsf{LOGCFL}\}$ 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

Key step: identify **classes of NDL programs** with **right complexity**

- NL: linear NDL-programs of bounded width (# vars per rule)

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

Key step: identify **classes of NDL programs** with **right complexity**

- NL: linear NDL-programs of bounded width (# vars per rule)

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**
- **our rewritings usually evaluated faster**

Upper bound on time needed to evaluate our NDL-rewritings:

- depth d / number of leaves ℓ occur in the exponent

Upper bound on time needed to evaluate our NDL-rewritings:

- depth d / number of leaves ℓ occur in the exponent

Is it possible to do better?

- formally: fixed-parameter tractable (FPT)? $f(d, \ell) \cdot p(|q|, |\mathcal{T}|, |\mathcal{A}|)$

Upper bound on time needed to evaluate our NDL-rewritings:

- depth d / number of leaves ℓ occur in the exponent

Is it possible to do better?

- formally: fixed-parameter tractable (FPT)? $f(d, \ell) \cdot p(|q|, |\mathcal{T}|, |\mathcal{A}|)$

Parameterized complexity of answering tree-shaped OMQs $(\mathcal{T}, q)$:

- parameters: depth d of $\mathcal{T}$, number ℓ of leaves in CQs

Upper bound on time needed to evaluate our NDL-rewritings:

- depth d / number of leaves ℓ occur in the exponent

Is it possible to do better?

- formally: fixed-parameter tractable (FPT)? $f(d, \ell) \cdot p(|q|, |\mathcal{T}|, |\mathcal{A}|)$

Parameterized complexity of answering tree-shaped OMQs $(\mathcal{T}, q)$:

- parameters: depth d of $\mathcal{T}$, number ℓ of leaves in CQs
- not FPT if depth d taken as parameter W[2]-hard
- not FPT if number of leaves ℓ taken as parameter W[1]-hard

Message: for good performance, want d and ℓ small

EXISTENCE OF REWRITINGS

QUERY REWRITING BEYOND DL-LITE

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 ontologies and queries encountered in practice

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

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

DECIDING EXISTENCE OF REWRITINGS

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 $\mathcal{Q}$ $\qquad \mathcal{Q} \in \{\mathsf{IQ}, \mathsf{CQ}\}$

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

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**

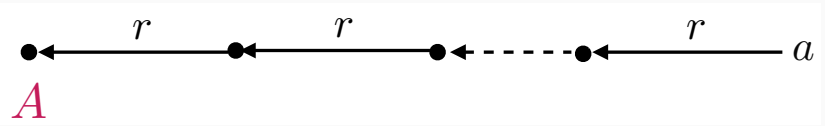
Datalog-rewritability decision problem can be **defined analogously**

We have the following results:

- FO-rewritability is **ExpTime-complete** in $(\mathcal{EL}, \mathrm{IQ})$ and $(\mathcal{ELI}, \mathrm{IQ})$
- FO-rewritability is **ExpTime-complete** in $(\mathcal{EL}, \mathrm{CQ})$
- FO-rewritability is **2ExpTime-complete** in $(\mathcal{ELI}, \mathrm{CQ})$

What makes FO-rewritability so difficult?

Consider TBox $\mathcal{T} = \{\exists r.A \sqsubseteq A, \exists r.\top \sqsubseteq A\}$ IQ $A(x)$

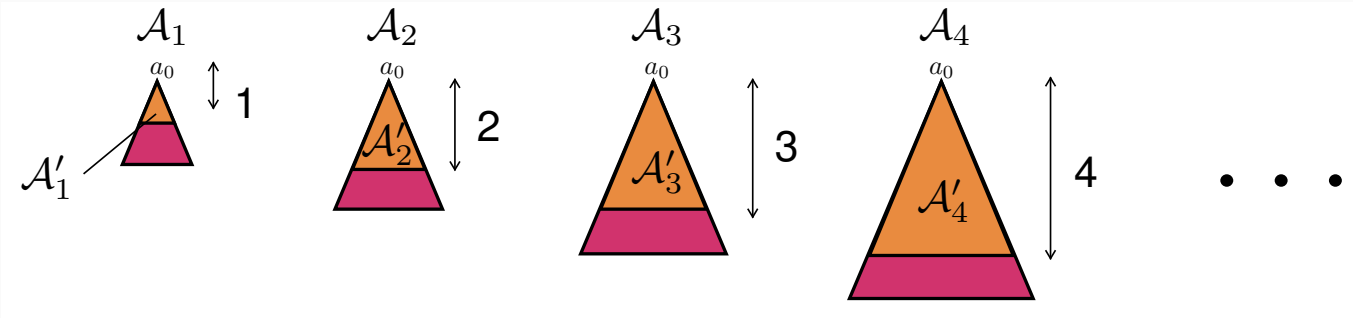


FO-rewriting exists: $A(x) \vee \exists y\, r(x, y)$

Reason: $\exists r.\top \sqsubseteq A$ **cancels effect of recursive** $\exists r.A \sqsubseteq A$

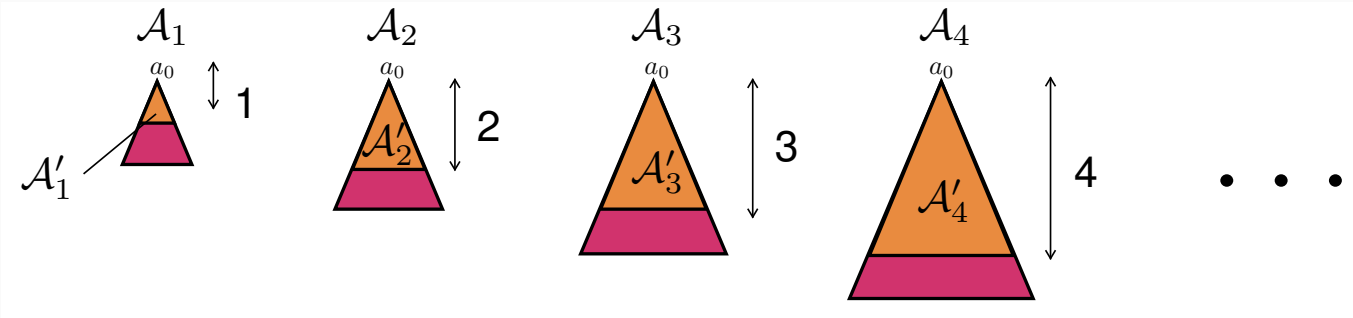
- such cancellations can be quite complicated and difficult to detect

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 $\mathcal{T}, \mathcal{A}'_i \not\models A(a_0)$

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

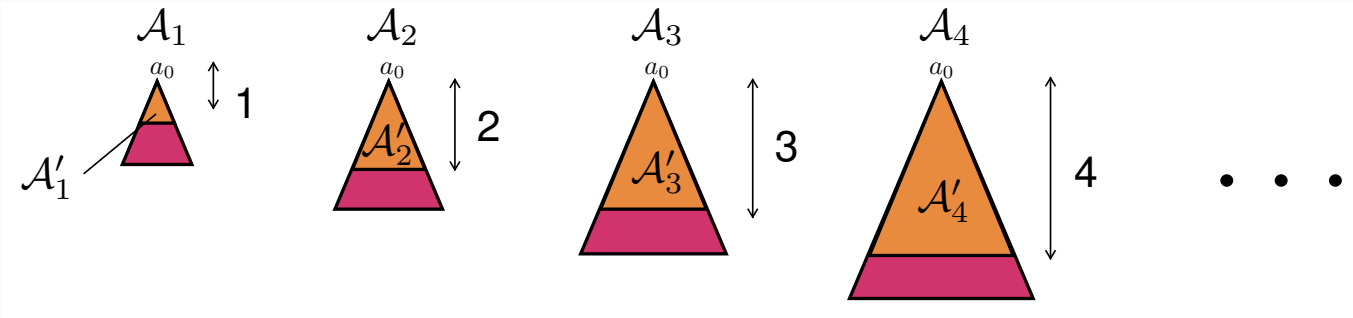


such that for all $i \geq 1$: $\mathcal{T}, \mathcal{A}_i \models A(a_0)$ and $\mathcal{T}, \mathcal{A}'_i \not\models A(a_0)$

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

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



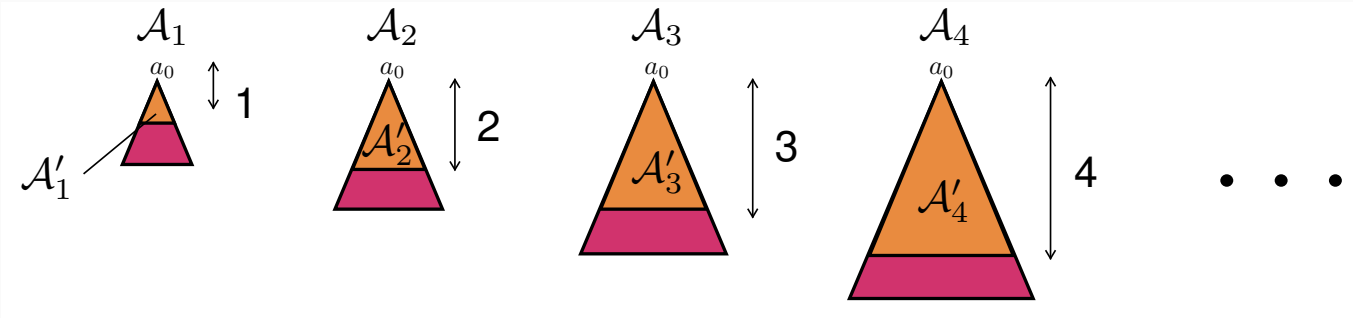
such that for all $i \geq 1$: $\mathcal{T}, \mathcal{A}_i \models A(a_0)$ and $\mathcal{T}, \mathcal{A}'_i \not\models A(a_0)$

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**

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

Use **existing backwards-chaining rewriting procedure**

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

Use **existing backwards-chaining rewriting procedure**

- if FO-rewriting does exist, terminates and outputs UCQ-rewriting
- 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

Use **existing backwards-chaining rewriting procedure**

- if FO-rewriting does exist, terminates and outputs UCQ-rewriting
- 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**

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

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

Lower bound: reduction from exponential grid tiling problem

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

Lower bound: reduction from exponential grid tiling problem

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 of $(\mathcal{ALC}, \text{UCQ})$ is 2NExptime-complete

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

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

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

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

OMQs from $(\mathcal{ALC}, \text{UCQ})$ $\sim$ complement of MMSNP formulas
$\sim$ monadic disjunctive Datalog

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

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

OMQs from $(\mathcal{ALC}, \text{UCQ})$ $\sim$ complement of MMSNP formulas
$\sim$ monadic disjunctive Datalog

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

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

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

OMQs from $(\mathcal{ALC}, \text{UCQ})$ $\sim$ complement of MMSNP formulas
$\sim$ monadic disjunctive Datalog

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)
- optimality of rewritings (Datalog fragments, param. complexity)
- existence of FO and Datalog rewritings (automata, CSP / MMSNP)

Lots left to do!

- experiment + optimize NDL-rewritings, evaluation strategies
- develop practical algorithms going beyond $\mathcal{EL}$ and IQs

Questions ?

References: Succinctness & Optimality of Rewritings

[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.

References: Succinctness & Optimality of Rewritings

[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.

[BKKPZ16a] M. Bienvenu, S. Kikot, R. Kontchakov, V. Podolskii, and M. Zakharyaschev: Theoretically Optimal Datalog Rewritings for OWL 2 QL Ontology-Mediated Queries. 29th International Workshop on Description Logics (**DL'16**), 2016.

[BKKPZ16b] M. Bienvenu, S. Kikot, R. Kontchakov, V. Podolskii, and M. Zakharyaschev: Ontology-Mediated Queries: Combined Complexity and Succinctness of Rewritings via Circuit Complexity. Under review, CoRR abs/1605.01207, 2016.

[BKKPRZ16] 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. Under review, 2016.

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.

REFERENCES: EXISTENCE OF REWRITINGS

[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.

[FKL16] C. Feier, A. Kuusisto, and C. Lutz: FO-Rewritability of Expressive Ontology-Mediated Queries. 29th International Workshop on Description Logics (**DL'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.

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.