ONTOLOGY-MEDIATED QUERY ANSWERING

Part 2: OMQA Problem and Algorithms

Meghyn Bienvenu (*CNRS & Université de Bordeaux*)

INTRODUCTION TO OMQA

USING ONTOLOGIES TO ACCESS DATA

Aim: enrich databases (DBs) with ontologies

- convenient vocabulary for users to specify queries
- link multiple datasets with different schemas
- knowledge in ontology can yield additional answers to queries

USING ONTOLOGIES TO ACCESS DATA

Aim: enrich databases (DBs) with ontologies

- convenient vocabulary for users to specify queries
- link multiple datasets with different schemas
- knowledge in ontology can yield additional answers to queries

Desiderata:

- efficiency is crucial – must scale up to huge datasets
- would like to use expressive queries like in DBs
  - conjunctive queries $\sim$ select-project-join queries in SQL

Note: henceforth assume ABox uses only concept and role names

CONJUNCTIVE QUERIES

An **atom** takes the form $A(t_1)$ or $r(t_1, t_2)$ where:

- A is a concept name, r a role name
- each t_i is either a variable or individual name

A **conjunctive query (CQ)** has the form

$$q(x_1, \ldots, x_k) \quad = \quad \exists y_1, \ldots, y_m \;\; \alpha_1 \wedge \ldots \wedge \alpha_n$$

where each α_i is an atom with variables drawn from $x_1, \ldots, x_k, y_1, \ldots, y_m$.

- $y_1, \ldots, y_m$ are called **quantified / existential variables**
- $x_1, \ldots, x_k$ are called **answer variables**

CONJUNCTIVE QUERIES

An **atom** takes the form $A(t_1)$ or $r(t_1, t_2)$ where:

- A is a concept name, r a role name
- each t_i is either a variable or individual name

A **conjunctive query (CQ)** has the form

$$q(x_1, \ldots, x_k) \quad = \quad \exists y_1, \ldots, y_m \;\; \alpha_1 \wedge \ldots \wedge \alpha_n$$

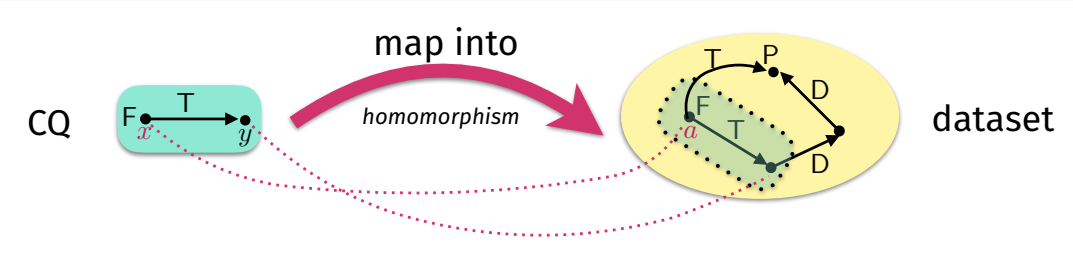
where each α_i is an atom with variables drawn from $x_1, \ldots, x_k, y_1, \ldots, y_m$.

- $y_1, \ldots, y_m$ are called **quantified / existential variables**
- $x_1, \ldots, x_k$ are called **answer variables**

Unions of conjunctive queries (UCQs):
disjunction of CQs sharing the same tuple of answer variables

QUERY ANSWERING: DATABASE VS ONTOLOGY SETTINGS

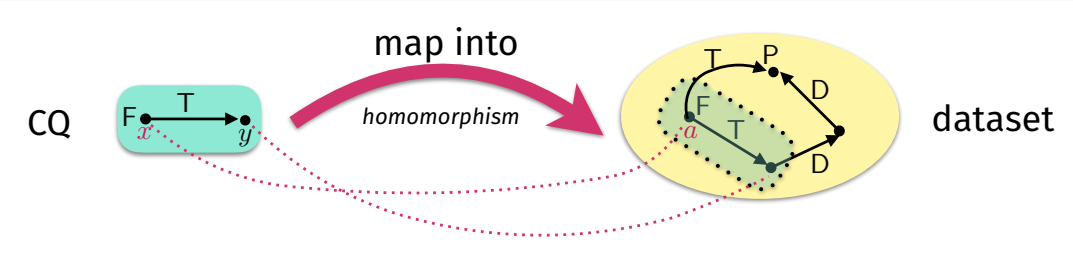
Answering CQs in database setting



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

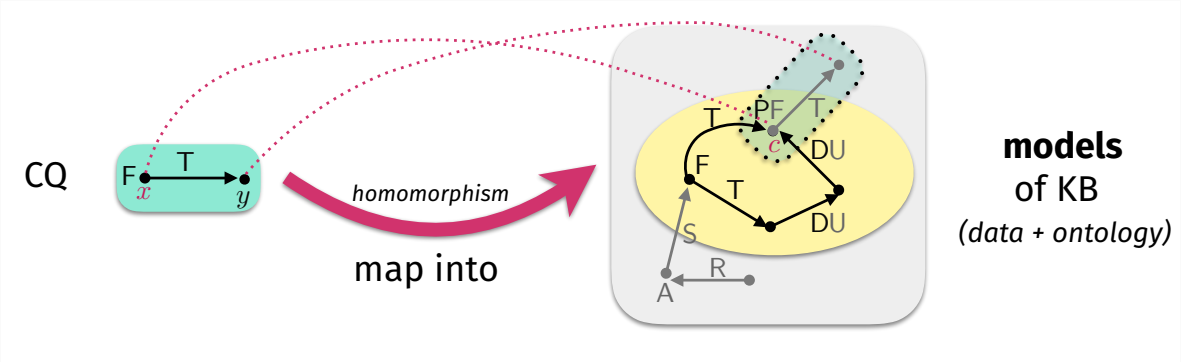
QUERY ANSWERING: DATABASE VS ONTOLOGY SETTINGS

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 $ans(q, \mathcal{I})$

A mapping $h : terms(q) \to \mathcal{I}$ is a **homomorphism of CQ q into $\mathcal{I}$** if:

- $h(a) = a^{\mathcal{I}}$, for every $a \in terms(q)$
- $A(t) \in q$ implies $h(t) \in A^{\mathcal{I}}$
- $r(t, t') \in q$ implies $(h(t), h(t')) \in r^{\mathcal{I}}$

Will also use notation $h : q \to \mathcal{I}$.

Answers to CQ $q(\vec{x})$ in interpretation $\mathcal{I}$:

$$ans(q, \mathcal{I}) = \{\vec{a} \subseteq \mathsf{N_I} \mid \text{ hom } h : q \to \mathcal{I} \text{ with } h(\vec{x}) = \vec{a}^{\mathcal{I}}\}$$

Equivalently, want tuples of individuals $\vec{a}$ such that $\mathcal{I} \models q[\vec{x}/\vec{a}]$.

Need to combine the answers from different models of a KB

Certain answers to CQ $q(\vec{x})$ w.r.t. $\mathcal{K} = (\mathcal{T}, \mathcal{A})$

$$cert(q, \mathcal{K}) = \{\vec{a} \subseteq \mathsf{Ind}(\mathcal{A}) \mid \vec{a} \in ans(q, \mathcal{I}) \text{ for every model } \mathcal{I} \text{ of } \mathcal{K}\}$$

Corresponds to a form of **entailment**, we'll also write $\mathcal{T}, \mathcal{A} \models q(\vec{a})$

Need to combine the answers from different models of a KB

Certain answers to CQ $q(\vec{x})$ w.r.t. $\mathcal{K} = (\mathcal{T}, \mathcal{A})$

$$cert(q, \mathcal{K}) = \{\vec{a} \subseteq \mathsf{Ind}(\mathcal{A}) \mid \vec{a} \in ans(q, \mathcal{I}) \text{ for every model } \mathcal{I} \text{ of } \mathcal{K}\}$$

Corresponds to a form of entailment, we'll also write $\mathcal{T}, \mathcal{A} \models q(\vec{a})$

Ontology-mediated query answering =
problem of computing / verifying certain answers

OMQA EXAMPLE

TBox:

$$\mathsf{Prof} \sqsubseteq \mathsf{Faculty} \qquad \mathsf{Fellow} \sqsubseteq \mathsf{Faculty} \qquad \mathsf{Prof} \sqsubseteq \neg \mathsf{Fellow}$$
$$\mathsf{Prof} \sqsubseteq \exists \mathsf{teaches}.\top \qquad \exists \mathsf{teaches}^{-}.\top \sqsubseteq \mathsf{Course}$$

ABox:

$$\{\mathsf{Prof}(\mathsf{anna}), \mathsf{Fellow}(\mathsf{tom}), \mathsf{teaches}(\mathsf{tom}, \mathsf{cs101})\}$$

Query: $q(x) = \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$

OMQA EXAMPLE

TBox:

$$\mathsf{Prof} \sqsubseteq \mathsf{Faculty} \qquad \mathsf{Fellow} \sqsubseteq \mathsf{Faculty} \qquad \mathsf{Prof} \sqsubseteq \neg\mathsf{Fellow}$$
$$\mathsf{Prof} \sqsubseteq \exists \mathsf{teaches}.\top \qquad \exists \mathsf{teaches}^{-}.\top \sqsubseteq \mathsf{Course}$$

ABox:

$$\{\mathsf{Prof}(\mathsf{anna}), \mathsf{Fellow}(\mathsf{tom}), \mathsf{teaches}(\mathsf{tom}, \mathsf{cs101})\}$$

Query: $q(x) = \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$

Get the following certain answers:

- anna — $\mathsf{Prof}(\mathsf{anna}) + \mathsf{Prof} \sqsubseteq \mathsf{Faculty} + \mathsf{Prof} \sqsubseteq \exists \mathsf{teaches}.\top$
- tom — $\mathsf{Fellow}(\mathsf{tom}) + \mathsf{Fellow} \sqsubseteq \mathsf{Faculty} + \mathsf{teaches}(\mathsf{tom}, \mathsf{cs101})$

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** $\mathcal{A}$, an $\mathcal{L}$-**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** $\mathcal{A}$, an $\mathcal{L}$**-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**

COMPLEXITY OF CQ ANSWERING

	data complexity	combined complexity
DL-Lite	in AC_0	NP-c
$\mathcal{EL}, \mathcal{ELHO}_\bot$	PTIME-c	NP-c
$\mathcal{ELI}$, Horn-$\mathcal{SHOIQ}$	PTIME-c	EXP-c
$\mathcal{ALC}$, $\mathcal{ALCHQ}$	coNP-c	EXP-c
$\mathcal{ALCI}$, $\mathcal{SHIQ}$	coNP-c	2EXP-c
$\mathcal{SHOIQ}$	coNP-h	coN2EXP-h

coNP-hardness for ALC

CQ answering has **coNP data complexity**, even for atomic CQs

Easy proof of **coNP hardness via (non)-3-colourability**:

Use ABox to encode graph, consider the following TBox axioms:

- $\top \sqsubseteq R \sqcup G \sqcup B$
- $B \sqcap \exists \mathsf{edge}.B \sqsubseteq \mathsf{Clash}$ (and similarly for R, G)

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

coNP-hardness for ALC

CQ answering has coNP data complexity, even for atomic CQs

Easy proof of coNP hardness via (non)-3-colourability:

Use ABox to encode graph, consider the following TBox axioms:

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

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

More generally: any form of (implicit or explicit) disjunction leads to coNP-hard data complexity for CQ answering

Intractability results led to proposal of DLs with lower complexity

DL-Lite family of DLs (basis for **OWL 2 QL**)

- designed with OMQA in mind
- capture main constructs from conceptual modelling
- key technique: query rewriting

DATA-TRACTABLE DLS

Intractability results led to proposal of DLs with lower complexity

DL-Lite family of DLs (basis for OWL 2 QL)

- designed with OMQA in mind
- capture main constructs from conceptual modelling
- key technique: query rewriting

$\mathcal{EL}$ family of DLs (basis for OWL 2 EL)

- designed to allow efficient reasoning with large ontologies
- well suited for medical and life science applications
- key technique: materialization

Intractability results led to proposal of DLs with lower complexity

DL-Lite family of DLs (basis for OWL 2 QL)

- designed with OMQA in mind
- capture main constructs from conceptual modelling
- key technique: query rewriting

$\mathcal{EL}$ family of DLs (basis for OWL 2 EL)

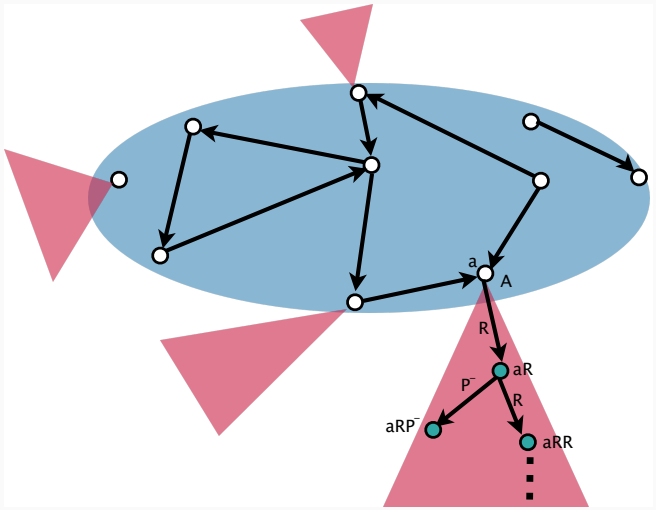
- designed to allow efficient reasoning with large ontologies
- well suited for medical and life science applications
- key technique: materialization

Commonality: cannot express disjunction (Horn logics),
existence of a canonical / universal model

CANONICAL MODELS

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

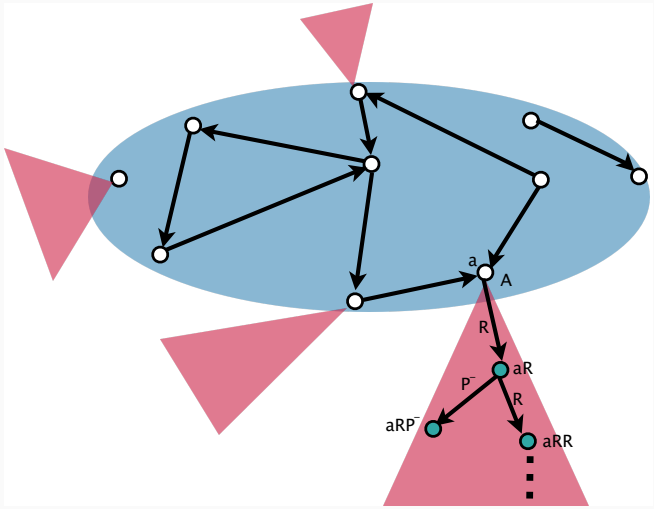
- idea: exhaustively apply ontology axioms to dataset (like the chase)
- possibly infinite ($A \sqsubseteq \exists r.A$)
- forest-shaped (dataset + new tree structures for $\exists$-axioms)
- universal in the sense that homomorphically embeddable into every model of the KB $\rightsquigarrow$ gives correct answer to all (U)CQs



CANONICAL MODELS

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

- idea: exhaustively apply ontology axioms to dataset (like the chase)
- possibly infinite ($A \sqsubseteq \exists r.A$)
- forest-shaped (dataset + new tree structures for $\exists$-axioms)
- universal in the sense that homomorphically embeddable into every model of the KB $\rightsquigarrow$ gives correct answer to all (U)CQs



OMQA with Horn DLs =

finding ways to
map the query into
the canonical model

HIGH-LEVEL VIEW OF OMQA ALGORITHMS

Query rewriting: Reduce OMQA to standard database query evaluation rewriting

HIGH-LEVEL VIEW OF OMQA ALGORITHMS

Query rewriting: Reduce OMQA to standard database query evaluation rewriting

Materialization: Complete the data with implicit information from the ontology

OMQA TECHNIQUES FOR DL-LITE

We present the dialect DL-Lite$_R$ (which underlies OWL2 QL profile).

DL-Lite$_R$ TBoxes contain

- concept inclusions $B_1 \sqsubseteq B_2$, $B_1 \sqsubseteq \neg B_2$
- role inclusions $R_1 \sqsubseteq R_2$, $R_1 \sqsubseteq \neg R_2$

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

DL-Lite$_{RDFS}$ ($\sim$ RDF Schema): disallow $\neg$ or $\exists$ on RHS of axioms

We present the **dialect DL-Lite$_R$** (which underlies **OWL2 QL profile**).

DL-Lite$_R$ TBoxes contain

- **concept inclusions** $B_1 \sqsubseteq B_2$, $B_1 \sqsubseteq \neg B_2$
- **role inclusions** $R_1 \sqsubseteq R_2$, $R_1 \sqsubseteq \neg R_2$

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

DL-Lite$_{RDFS}$ ($\sim$ RDF Schema): disallow $\neg$ or $\exists$ on RHS of axioms

Example TBox inclusions:

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

QUERY REWRITING

General idea: reduce OMQA to database query evaluation

- rewriting step: TBox $\mathcal{T}$ + query $q \rightsquigarrow$ database query q'
- evaluation step: evaluate query q' using database system

QUERY REWRITING

General idea: reduce OMQA to database query evaluation

- rewriting step: TBox $\mathcal{T}$ + query $q \rightsquigarrow$ database query q'
- evaluation step: evaluate query q' using database system

Formally: a query q' is a rewriting of q w.r.t. $\mathcal{T}$ iff for every ABox $\mathcal{A}$:

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

where $\mathcal{I}_{\mathcal{A}}$ is the finite interpretation (database) corresponding to $\mathcal{A}$, i.e. with domain $\mathsf{Ind}(\mathcal{A})$ & making true precisely the assertions in $\mathcal{A}$.

QUERY REWRITING

General idea: **reduce OMQA to database query evaluation**

- **rewriting step**: TBox $\mathcal{T}$ + query $q \rightsquigarrow$ **database query** q'
- **evaluation step**: evaluate query q' using **database system**

Formally: a query q' **is a rewriting of** q **w.r.t.** $\mathcal{T}$ iff for every ABox $\mathcal{A}$:

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

where $\mathcal{I}_{\mathcal{A}}$ is the finite interpretation (database) corresponding to $\mathcal{A}$, i.e. with domain $\mathsf{Ind}(\mathcal{A})$ & making true precisely the assertions in $\mathcal{A}$.

Need q' to belong to a **database query** language:

- **FO-rewritings**: q' is a first-order ($\sim$ SQL) query
- **Datalog rewriting**: q' is a Datalog query (more on this later)

QUERY REWRITING

General idea: **reduce OMQA to database query evaluation**

- **rewriting step**: TBox $\mathcal{T}$ + query $q \rightsquigarrow$ **database query** q'
- **evaluation step**: evaluate query q' using **database system**

Formally: a query q' **is a rewriting of** q **w.r.t.** $\mathcal{T}$ iff for every ABox $\mathcal{A}$:

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

where $\mathcal{I}_{\mathcal{A}}$ is the finite interpretation (database) corresponding to $\mathcal{A}$, i.e. with domain $\mathsf{Ind}(\mathcal{A})$ & making true precisely the assertions in $\mathcal{A}$.

Need q' to belong to a **database query** language:

- **FO-rewritings**: q' is a first-order ($\sim$ SQL) query
- **Datalog rewriting**: q' is a Datalog query (more on this later)

Note: **DL-Lite designed so that every CQ has an FO-rewriting**

EXAMPLE: QUERY REWRITING IN DL-LITE

Reconsider the DL-Lite TBox $\mathcal{T}$:

$$\text{Prof} \sqsubseteq \text{Faculty} \qquad \text{Fellow} \sqsubseteq \text{Faculty} \qquad \text{Prof} \sqsubseteq \neg\text{Fellow}$$
$$\text{Prof} \sqsubseteq \exists\text{teaches} \qquad \exists\text{teaches}^{-} \sqsubseteq \text{Course}$$

and the query $q(x) = \exists y.\text{Faculty}(x) \wedge \text{teaches}(x, y)$

EXAMPLE: QUERY REWRITING IN DL-LITE

Reconsider the DL-Lite TBox $\mathcal{T}$:

$$\mathsf{Prof} \sqsubseteq \mathsf{Faculty} \qquad \mathsf{Fellow} \sqsubseteq \mathsf{Faculty} \qquad \mathsf{Prof} \sqsubseteq \neg\mathsf{Fellow}$$
$$\mathsf{Prof} \sqsubseteq \exists\mathsf{teaches} \qquad \exists\mathsf{teaches}^{-} \sqsubseteq \mathsf{Course}$$

and the query $q(x) = \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$

The following query is a rewriting of $q(x)$ w.r.t. $\mathcal{T}$:

$$q(x) \quad \vee \quad \mathsf{Prof}(x) \quad \vee \quad \exists y.\mathsf{Fellow}(x) \wedge \mathsf{teaches}(x, y)$$

EXAMPLE: QUERY REWRITING IN DL-LITE

Reconsider the DL-Lite TBox $\mathcal{T}$:

$$\mathsf{Prof} \sqsubseteq \mathsf{Faculty} \qquad \mathsf{Fellow} \sqsubseteq \mathsf{Faculty} \qquad \mathsf{Prof} \sqsubseteq \neg \mathsf{Fellow}$$
$$\mathsf{Prof} \sqsubseteq \exists \mathsf{teaches} \qquad \exists \mathsf{teaches}^{-} \sqsubseteq \mathsf{Course}$$

and the query $q(x) = \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$

The following query is a rewriting of $q(x)$ w.r.t. $\mathcal{T}$:

$$q(x) \quad \vee \quad \mathsf{Prof}(x) \quad \vee \quad \exists y.\mathsf{Fellow}(x) \wedge \mathsf{teaches}(x, y)$$

Evaluating the rewritten query over the earlier ABox

$$\{\mathsf{Prof}(\mathsf{anna}), \mathsf{Fellow}(\mathsf{tom}), \mathsf{teaches}(\mathsf{tom}, \mathsf{cs101})\}$$

produces the two certain answers: anna and tom

QUERY REWRITING & SATISFIABILITY

Desirable to separate satisfiability check from query answering

Rewriting of q w.r.t. $\mathcal{T}$ for consistent ABoxes: defined as before, but only quantify over $\mathcal{A}$ such that $(\mathcal{T}, \mathcal{A})$ is satisfiable

Desirable to separate satisfiability check from query answering

Rewriting of q w.r.t. $\mathcal{T}$ for consistent ABoxes: defined as before, but only quantify over $\mathcal{A}$ such that $(\mathcal{T}, \mathcal{A})$ is satisfiable

Rewriting of unsatisfiability w.r.t. $\mathcal{T}$: q_{unsat} such that for all $\mathcal{A}$:

$$(\mathcal{T}, \mathcal{A}) \text{ unsatisfiable} \quad \Leftrightarrow \quad \mathcal{I}_{\mathcal{A}} \models q_{\text{unsat}}$$

QUERY REWRITING & SATISFIABILITY

Desirable to separate satisfiability check from query answering

Rewriting of q w.r.t. $\mathcal{T}$ for consistent ABoxes: defined as before, but only quantify over $\mathcal{A}$ such that $(\mathcal{T}, \mathcal{A})$ is satisfiable

Rewriting of unsatisfiability w.r.t. $\mathcal{T}$: q_{unsat} such that for all $\mathcal{A}$:

$$(\mathcal{T}, \mathcal{A}) \text{ unsatisfiable} \quad \Leftrightarrow \quad \mathcal{I}_{\mathcal{A}} \models q_{\text{unsat}}$$

Rewritings (w.r.t. arbitrary ABoxes) can be built from rewritings w.r.t. consistent ABoxes + rewriting of unsatisfiability

- can focus on rewriting the query w.r.t. consistent ABoxes

QUERY REWRITING STEP BY STEP

We will present a query rewriting approach by stages:

- rewritings of instance queries (single atoms)
- rewriting of unsatisfiability
- rewriting of CQs for RDFS fragment
- rewriting of CQs for DL-Lite

This will allow us to illustrate useful notions along the way

Also: approach can be extended to richer Horn DLs and queries

Rewriting of $A(x)$ w.r.t. $\mathcal{T}$:

$$rew^{\mathcal{T}}_{\mathsf{IQ}}(A(x)) = \bigvee_{\mathcal{T} \models A' \sqsubseteq A} A'(x) \vee \bigvee_{\mathcal{T} \models \exists r \sqsubseteq A} \exists y.r(x,y) \vee \bigvee_{\mathcal{T} \models \exists r^- \sqsubseteq A} \exists y.r(y,x)$$

Rewriting of $A(x)$ w.r.t. $\mathcal{T}$:

$$rew^{\mathcal{T}}_{\mathrm{IQ}}(A(x)) = \bigvee_{\mathcal{T} \models A' \sqsubseteq A} A'(x) \vee \bigvee_{\mathcal{T} \models \exists r \sqsubseteq A} \exists y.r(x,y) \vee \bigvee_{\mathcal{T} \models \exists r^- \sqsubseteq A} \exists y.r(y,x)$$

Rewriting of $r(x,y)$ w.r.t. $\mathcal{T}$:

$$rew^{\mathcal{T}}_{\mathrm{IQ}}(r(x,y)) = \bigvee_{\mathcal{T} \models s \sqsubseteq r} s(x,y) \vee \bigvee_{\mathcal{T} \models s^- \sqsubseteq r} s(y,x)$$

Rewriting of $A(x)$ w.r.t. $\mathcal{T}$:

$$rew^{\mathcal{T}}_{\mathrm{IQ}}(A(x)) = \bigvee_{\mathcal{T} \models A' \sqsubseteq A} A'(x) \vee \bigvee_{\mathcal{T} \models \exists r \sqsubseteq A} \exists y.r(x,y) \vee \bigvee_{\mathcal{T} \models \exists r^- \sqsubseteq A} \exists y.r(y,x)$$

Rewriting of $r(x,y)$ w.r.t. $\mathcal{T}$:

$$rew^{\mathcal{T}}_{\mathrm{IQ}}(r(x,y)) = \bigvee_{\mathcal{T} \models s \sqsubseteq r} s(x,y) \vee \bigvee_{\mathcal{T} \models s^- \sqsubseteq r} s(y,x)$$

Theorem: $rew^{\mathcal{T}}_{\mathrm{IQ}}(\alpha)$ is a rewriting of IQ α w.r.t. $\mathcal{T}$ and consistent ABoxes

Note: only polynomially many axioms to consider, and testing entailment in DL-Lite$_R$ is easy (NLOGSPACE)

Rewriting of unsatisfiability w.r.t. $\mathcal{T}$:

$$q^{\mathcal{T}}_{\text{unsat}} = \bigvee_{\mathcal{T} \models B_1 \sqsubseteq \neg B_2} (\rho_x(B_1) \wedge \rho_x(B_2)) \vee \bigvee_{\mathcal{T} \models R_1 \sqsubseteq \neg R_2} (\rho_{x,y}(R_1) \wedge \rho_{x,y}(R_2))$$

where use ρ_x and $\rho_{x,y}$ to translate concepts / roles into atoms:

- $\rho_x(A)$ is $A(x)$ if $A \in \mathsf{N_C}$
- $\rho_x(\exists r)$ is $\exists y.r(x,y)$
- $\rho_x(\exists r^-)$ is $\exists y.r(x,y)$
- $\rho_{x,y}(r)$ is $r(x,y)$
- $\rho_{x,y}(r^-)$ is $r(y,x)$

Shows that we can focus on rewritings w.r.t. consistent ABoxes

Is it enough to apply the IQ rewriting to every atom in the CQ?

$$rew^{\mathcal{T}}_{\text{naive}}(\exists \vec{y}.\alpha_1 \wedge \ldots \wedge \alpha_n) = \exists \vec{y}.rew^{\mathcal{T}}_{\text{IQ}}(\alpha_1) \wedge \ldots \wedge rew^{\mathcal{T}}_{\text{IQ}}(\alpha_n)$$

Is it enough to apply the IQ rewriting to every atom in the CQ?

$$rew^{\mathcal{T}}_{\text{naive}}(\exists \vec{y}.\alpha_1 \wedge \ldots \wedge \alpha_n) = \exists \vec{y}.rew^{\mathcal{T}}_{\text{IQ}}(\alpha_1) \wedge \ldots \wedge rew^{\mathcal{T}}_{\text{IQ}}(\alpha_n)$$

Theorem: $rew^{\mathcal{T}}_{\text{naive}}(q)$ **is a rewriting** of CQ q w.r.t. $\mathcal{T}$, provided that $\mathcal{T}$ is a **DL-Lite$_{RDFS}$ TBox**

Is it enough to apply the IQ rewriting to every atom in the CQ?

$$rew^{\mathcal{T}}_{\text{naive}}(\exists\vec{y}.\alpha_1 \wedge \ldots \wedge \alpha_n) = \exists\vec{y}.rew^{\mathcal{T}}_{\text{IQ}}(\alpha_1) \wedge \ldots \wedge rew^{\mathcal{T}}_{\text{IQ}}(\alpha_n)$$

Theorem: $rew^{\mathcal{T}}_{\text{naive}}(q)$ **is a rewriting** of CQ q w.r.t. $\mathcal{T}$, provided that $\mathcal{T}$ is a **DL-Lite$_{RDFS}$ TBox**

But this **doesn't work for full DL-Lite$_R$ logic**:

- ignores axioms $B \sqsubseteq \exists R$ which may allow us to map variables to non-ABox elements
- ignores interactions between conjuncts

Is it enough to apply the IQ rewriting to every atom in the CQ?

$$rew^{\mathcal{T}}_{\text{naive}}(\exists \vec{y}.\alpha_1 \wedge \ldots \wedge \alpha_n) = \exists \vec{y}.rew^{\mathcal{T}}_{\text{IQ}}(\alpha_1) \wedge \ldots \wedge rew^{\mathcal{T}}_{\text{IQ}}(\alpha_n)$$

Theorem: $rew^{\mathcal{T}}_{\text{naive}}(q)$ **is a rewriting** of CQ q w.r.t. $\mathcal{T}$, provided that $\mathcal{T}$ is a **DL-Lite$_{RDFS}$ TBox**

But this **doesn't work for full DL-Lite$_R$ logic**:

- ignores axioms $B \sqsubseteq \exists R$ which may allow us to map variables to non-ABox elements
- ignores interactions between conjuncts

Example: **miss** $\text{Prof}(x)$ as way to obtain $\exists y.\text{Faculty}(x) \wedge \text{teaches}(x, y)$

INTERLUDE: CANONICAL MODEL

Define **canonical model** $\mathcal{C}_\mathcal{K}$ for DL-Lite KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$

The **domain** $\Delta^{\mathcal{C}_\mathcal{K}}$ **of** $\mathcal{C}_\mathcal{K}$ contains all words $aR_1R_2 \dots R_n$ $(n \geq 0)$ where $a \in \mathsf{Ind}(\mathcal{A})$, the R_i are (possibly inverse) roles, and:

- if $n \geq 1$, then $\mathcal{K} \models \exists R_1(a)$
- for $1 \leq i \leq n$, then $\mathcal{T} \models \exists R_i^- \sqsubseteq \exists R_{i+1}$ and $R_i^- \neq R_{i+1}$

Interpretation of concept, role, and individual names:

- $a^{\mathcal{C}_\mathcal{K}} = a$
- $A^{\mathcal{C}_\mathcal{K}} = \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models A(a)\} \cup \{aR_1 \dots, R_n \mid \mathcal{T} \models \exists R_n^- \sqsubseteq A\}$
- $r^{\mathcal{C}_\mathcal{K}} = \{(a, b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models r(a, b)\} \cup$
  $\{e_1, e_2) \mid e_2 = e_1 S \text{ and } \mathcal{T} \models S \sqsubseteq r\} \cup$
  $\{e_2, e_1) \mid e_2 = e_1 S \text{ and } \mathcal{T} \models S \sqsubseteq r^-\}$

Can also define via exhaustive application of axioms (chase)

Recall that

$$\mathcal{K} \models q(\vec{a}) \text{ iff } \mathcal{C}_\mathcal{K} \models q(\vec{a}) \text{ iff hom } h : q(\vec{a}) \to \mathcal{C}_\mathcal{K}$$

Can use naïve rewriting to find all homomorphisms that map query variables to the core of $\mathcal{C}_\mathcal{K}$ (i.e. ABox individuals)

Next task will be to show how we can rewrite the query so that it is sufficient to only consider homomorphisms into the ABox part of $\mathcal{C}_\mathcal{K}$

This is done in iterative manner: 'move' the homomorphism closer and closer to the ABox

SINGLE REWRITING STEP

Write $q \rightsquigarrow q'$ if q' can be obtained from CQ q as follows:

1. Choose **'leaf' variable** x (intuition: no vars mapped below it in $\mathcal{C}_\mathcal{K}$)
   - not allowed to choose x if x is an answer variable or $s(x, x) \in q$

SINGLE REWRITING STEP

Write $q \rightsquigarrow q'$ if q' can be obtained from CQ q as follows:

1. Choose **'leaf' variable** x (intuition: no vars mapped below it in $\mathcal{C}_\mathcal{K}$)
   - not allowed to choose x if x is an answer variable or $s(x, x) \in q$
2. **Choose** R **that ensures satisfaction of query atoms involving** x
   - if $A(x) \in q$, then $\mathcal{T} \models \exists R^- \sqsubseteq A$ (note: if $R = u^-$, then R^- denotes u)
   - if $s(y, x) \in q$, then $\mathcal{T} \models R \sqsubseteq s$
   - if $s(x, y) \in q$, then $\mathcal{T} \models R \sqsubseteq s^-$

SINGLE REWRITING STEP

Write $q \rightsquigarrow q'$ if q' can be obtained from CQ q as follows:

1. Choose **'leaf' variable** x (intuition: no vars mapped below it in $\mathcal{C}_\mathcal{K}$)
   - not allowed to choose x if x is an answer variable or $s(x, x) \in q$
2. **Choose** R **that ensures satisfaction of query atoms involving** x
   - if $A(x) \in q$, then $\mathcal{T} \models \exists R^- \sqsubseteq A$ (note: if $R = u^-$, then R^- denotes u)
   - if $s(y, x) \in q$, then $\mathcal{T} \models R \sqsubseteq s$
   - if $s(x, y) \in q$, then $\mathcal{T} \models R \sqsubseteq s^-$
3. **Choose concept** B **such that** $\mathcal{T} \models B \sqsubseteq \exists R$

SINGLE REWRITING STEP

Write $q \rightsquigarrow q'$ if q' can be obtained from CQ q as follows:

1. Choose **'leaf' variable** x (intuition: no vars mapped below it in $\mathcal{C}_\mathcal{K}$)
   - not allowed to choose x if x is an answer variable or $s(x, x) \in q$
2. **Choose** R **that ensures satisfaction of query atoms involving** x
   - if $A(x) \in q$, then $\mathcal{T} \models \exists R^- \sqsubseteq A$ (note: if $R = u^-$, then R^- denotes u)
   - if $s(y, x) \in q$, then $\mathcal{T} \models R \sqsubseteq s$
   - if $s(x, y) \in q$, then $\mathcal{T} \models R \sqsubseteq s^-$
3. **Choose concept** B **such that** $\mathcal{T} \models B \sqsubseteq \exists R$
4. Let q' be the query obtained from q by:
   - **Unifying all 'neighbours' of** x (say y is this merged variable)
   - **Remove all atoms that mention** x
   - **Add new query atom to force** y **to belong to** B
     - $B(y)$, if $B \in \mathsf{N_C}$, or $u(z, y)$, if $B = \exists u^-$ (z fresh variable), similarly for $B = \exists u$

SINGLE REWRITING STEP

Write $q \rightsquigarrow q'$ if q' can be obtained from CQ q as follows:

1. Choose **'leaf' variable** x (intuition: no vars mapped below it in $\mathcal{C}_\mathcal{K}$)
   - not allowed to choose x if x is an answer variable or $s(x, x) \in q$
2. **Choose** R **that ensures satisfaction of query atoms involving** x
   - if $A(x) \in q$, then $\mathcal{T} \models \exists R^- \sqsubseteq A$ (note: if $R = u^-$, then R^- denotes u)
   - if $s(y, x) \in q$, then $\mathcal{T} \models R \sqsubseteq s$
   - if $s(x, y) \in q$, then $\mathcal{T} \models R \sqsubseteq s^-$
3. **Choose concept** B **such that** $\mathcal{T} \models B \sqsubseteq \exists R$
4. Let q' be the query obtained from q by:
   - **Unifying all 'neighbours' of** x (say y is this merged variable)
   - **Remove all atoms that mention** x
   - **Add new query atom to force** y **to belong to** B
     - $B(y)$, if $B \in \mathsf{N_C}$, or $u(z, y)$, if $B = \exists u^-$ (z fresh variable), similarly for $B = \exists u$

Use $q \rightsquigarrow^* q'$ if q' can be obtained by sequence of such steps

REWRITING FOR COMPLETE ABOXES

Now set $rew^{\mathcal{T}}_{\text{core}}(q) = \{q' \mid q \rightsquigarrow^* q'\}$

Now set $rew^{\mathcal{T}}_{\text{core}}(q) = \{q' \mid q \rightsquigarrow^* q'\}$

Theorem For every CQ $q(\vec{x})$, DL-Lite$_R$ TBox $\mathcal{T}$, ABox $\mathcal{A}$, the following are equivalent:

- there is a **homomorphism h of q into $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ with $h(\vec{x}) = \vec{a}$**
- there is a **homomorphism h of some $q' \in rew^{\mathcal{T}}_{\text{core}}(q)$ into $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ such that $h(\vec{x}) = \vec{a}$ and every variable is mapped to $\text{Ind}(\mathcal{A})$**

Call $\bigvee_{q' \in rew^{\mathcal{T}}_{\text{core}}(q)} q'$ a **rewriting w.r.t. complete ABoxes**

If we **complete $\mathcal{A}$ by adding all entailed assertions** (a form of **materialization**), then we can simply **use this rewriting**

To obtain rewriting w.r.t. possibly incomplete ABoxes, we can combined the previous components:

$$rew^{\mathcal{T}}(q) = \bigvee_{q' \in rew^{\mathcal{T}}_{\text{core}}(q)} rew^{\mathcal{T}}_{\text{naive}}(q')$$

Theorem: For every CQ $q(\vec{x})$ and DL-Lite$_R$ TBox $\mathcal{T}$, the query $rew^{\mathcal{T}}(q)$ is a rewriting of $q(\vec{x})$ w.r.t. consistent ABoxes

Note: also works if the input query is a UCQ

Theorem. In DL-Lite$_R$, IQ answering is NLOGSPACE-complete in combined complexity and in AC_0 in data complexity.

Note: $AC_0 \subsetneq LOGSPACE \subseteq NLOGSPACE \subseteq PTIME$

Theorem. For DL-Lite$_R$, CQ answering is NP-complete in combined complexity and in AC_0 in data complexity.

Some remarks:

- For combined upper bounds: guess relevant part of rewriting + how it maps onto ABox or guess homomorphism into initial portion of canonical model
- Data upper bounds: use that FO query evaluation in AC_0
- Same data & combined complexity as CQ answering in databases

OMQA TECHNIQUES FOR THE EL FAMILY

The logic $\mathcal{EL}$ and its extensions are designed for applications requiring very large ontologies.

This family of DLs is well suited for biomedical applications.

Examples of large biomedical ontologies:

- GO (Gene Ontology), around 20,000 concepts
- NCI (cancer ontology), around 30,000 concepts
- SNOMED (medical ontology), over 350,000 concepts (!)

$$\mathsf{Pericarditis} \sqsubseteq \mathsf{Inflammation} \sqcap \exists \mathsf{loc}.\mathsf{Pericardium}$$
$$\mathsf{Pericardium} \sqsubseteq \mathsf{Tissue} \sqcap \exists \mathsf{partOf}.\mathsf{Heart} \qquad \mathsf{Inflammation} \sqsubseteq \mathsf{Disease}$$
$$\mathsf{Disease} \sqcap \exists \mathsf{loc}.\exists \mathsf{partOf}.\mathsf{Heart} \sqsubseteq \mathsf{HeartDisease}$$

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

$$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists r.C$$

Only concept inclusions $C_1 \sqsubseteq C_2$ in the TBox

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

$$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists r.C$$

Only concept inclusions $C_1 \sqsubseteq C_2$ in the TBox

Some possible extensions:

- $\bot$ (to express disjoint classes)
- domain restrictions $\text{dom}(r) \sqsubseteq C$
- range restrictions $\text{range}(r) \sqsubseteq C$
- complex role inclusions $r_1 \circ \ldots \circ r_n \sqsubseteq r_{n+1}$ (transitivity:$r \circ r \sqsubseteq r$)

OWL 2 EL profile includes all these extensions

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

$$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists r.C$$

Only concept inclusions $C_1 \sqsubseteq C_2$ in the TBox

Some possible extensions:

- $\bot$ (to express disjoint classes)
- domain restrictions $\text{dom}(r) \sqsubseteq C$
- range restrictions $\text{range}(r) \sqsubseteq C$
- complex role inclusions $r_1 \circ \ldots \circ r_n \sqsubseteq r_{n+1}$ (transitivity: $r \circ r \sqsubseteq r$)

OWL 2 EL profile includes all these extensions

We will focus on plain $\mathcal{EL}$ (without these extensions)

$\mathcal{T}$ is in normal form if it contains only inclusions of the forms:

$$A \sqsubseteq B \quad A_1 \sqcap A_2 \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

where A, A_1, A_2, B are concept names (or $\top$).

NORMAL FORM FOR EL TBOXES

$\mathcal{T}$ is in normal form if it contains only inclusions of the forms:

$$A \sqsubseteq B \quad A_1 \sqcap A_2 \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

where A, A_1, A_2, B are concept names (or $\top$).

Fact: for every $\mathcal{EL}$ TBox $\mathcal{T}$, we can construct in PTIME a TBox $\mathcal{T}'$ in normal form (possibly using new concept names) that is a model-conservative extension of $\mathcal{T}$, meaning that:

- every model of $\mathcal{T}'$ is a model of $\mathcal{T}$
- every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$

NORMAL FORM FOR EL TBOXES

$\mathcal{T}$ is in **normal form** if it contains only inclusions of the forms:

$$A \sqsubseteq B \quad A_1 \sqcap A_2 \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

where A, A_1, A_2, B are concept names (or $\top$).

Fact: for every $\mathcal{EL}$ TBox $\mathcal{T}$, we can **construct in PTIME a TBox $\mathcal{T}'$ in normal form (possibly using new concept names)** that is a **model-conservative extension** of $\mathcal{T}$, meaning that:

- every model of $\mathcal{T}'$ is a model of $\mathcal{T}$
- every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$

Example: the inclusion $\exists r.(\exists s.A \sqcap H) \sqsubseteq B \sqcap D$ would be replaced by

$$\exists r.E \sqsubseteq B \quad \exists r.E \sqsubseteq D \quad E \sqsubseteq F \quad E \sqsubseteq H \quad F \sqcap H \sqsubseteq E \quad F \sqsubseteq \exists s.A \quad \exists s.A \sqsubseteq F$$

Assume w.l.o.g. TBox is in normal form

Rules for deriving ontology axioms

$$\frac{}{A \sqsubseteq A}\ \text{T1} \qquad \frac{}{A \sqsubseteq \top}\ \text{T2} \qquad \frac{A \sqsubseteq B \quad B \sqsubseteq D}{A \sqsubseteq D}\ \text{T3}$$

$$\frac{A \sqsubseteq B_1 \quad A \sqsubseteq B_2 \quad B_1 \sqcap B_2 \sqsubseteq D}{A \sqsubseteq D}\ \text{T4} \qquad \frac{A \sqsubseteq \exists r.B_1 \quad B_1 \sqsubseteq B_2 \quad \exists r.B_2 \sqsubseteq D}{A \sqsubseteq D}\ \text{T5}$$

Rules for deriving assertions

$$\frac{A \sqsubseteq B \quad A(c)}{B(c)}\ \text{A1} \qquad \frac{A_1 \sqcap A_2 \sqsubseteq B \quad A_1(c) \quad A_2(c)}{B(c)}\ \text{A2} \qquad \frac{\exists r.A \sqsubseteq B \quad r(c,d) \quad A(d)}{B(c)}\ \text{A3}$$

EXAMPLE: COMPLETION RULES IN EL

TBox $\mathcal{T}$ contains axioms:

(1) $\exists \mathsf{hasIngred}.\mathsf{Spicy} \sqsubseteq \mathsf{Spicy}$ (2) $\mathsf{Spicy} \sqcap \mathsf{Dish} \sqsubseteq \mathsf{SpicyDish}$

(3) $\mathsf{ArrabSauce} \sqsubseteq \exists \mathsf{hasIngred}.\mathsf{Peperonc}$ (4) $\mathsf{Peperonc} \sqsubseteq \mathsf{Spicy}$

ABox $\mathcal{A}$ contains:

(5) $\mathsf{Dish}(p)$ (6) $\mathsf{hasIngred}(p, s)$ (7) $\mathsf{ArrabSauce}(s)$

EXAMPLE: COMPLETION RULES IN EL

TBox $\mathcal{T}$ contains axioms:

(1) $\exists \mathsf{hasIngred}.\mathsf{Spicy} \sqsubseteq \mathsf{Spicy}$ (2) $\mathsf{Spicy} \sqcap \mathsf{Dish} \sqsubseteq \mathsf{SpicyDish}$

(3) $\mathsf{ArrabSauce} \sqsubseteq \exists \mathsf{hasIngred}.\mathsf{Peperonc}$ (4) $\mathsf{Peperonc} \sqsubseteq \mathsf{Spicy}$

ABox $\mathcal{A}$ contains:

(5) $\mathsf{Dish}(p)$ (6) $\mathsf{hasIngred}(p, s)$ (7) $\mathsf{ArrabSauce}(s)$

Saturation procedure adds the following axioms and assertions:

(8) $\mathsf{ArrabSauce} \sqsubseteq \mathsf{Spicy}$	using (1), (3), (4) and rule T5
(9) $\mathsf{Spicy}(s)$	using (7), (8), and rule A1
(10) $\mathsf{Spicy}(p)$	using (1), (6), (9), and rule A3
(11) $\mathsf{SpicyDish}(p)$	using (2), (5), (10), and rule A2

PROPERTIES OF COMPLETION RULES

Let close($\mathcal{T}$) be the result of exhaustively applying all the TBox completion rules, and adding derived axioms to $\mathcal{T}$

Let close$_\mathcal{T}(\mathcal{A})$ be the result of **adding to $\mathcal{A}$ all assertions that can be derived** using TBox + ABox rules

Theorem: can **compute** close($\mathcal{T}$) **and** close$_\mathcal{T}(\mathcal{A})$ **in polynomial time**

PROPERTIES OF COMPLETION RULES

Let close($\mathcal{T}$) be the result of exhaustively applying all the TBox completion rules, and adding derived axioms to $\mathcal{T}$

Let close$_\mathcal{T}(\mathcal{A})$ be the result of **adding to $\mathcal{A}$ all assertions that can be derived** using TBox + ABox rules

Theorem: can **compute** close($\mathcal{T}$) **and** close$_\mathcal{T}(\mathcal{A})$ **in polynomial time**

Theorem: For every $\mathcal{EL}$ KB $(\mathcal{T}, \mathcal{A})$:

- $\mathcal{T} \models A \sqsubseteq B$ $(A, B \in \mathsf{N_C})$ **iff** $A \sqsubseteq B \in$ close($\mathcal{T}$)
- $\mathcal{T}, \mathcal{A} \models A(b)$ $(A \in \mathsf{N_C})$ **iff** $A(b) \in$ close$_\mathcal{T}(\mathcal{A})$

This yields PTIME procedures for classification and IQ answering

FO-rewriting for IQS?

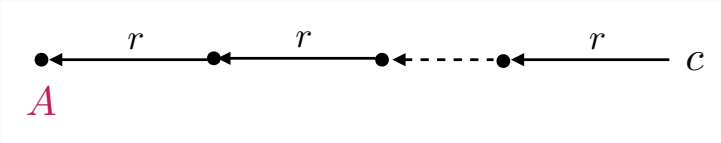
Could we alternatively use FO-rewriting, like for DL-Lite?

FO-rewriting for IQs?

Could we alternatively use FO-rewriting, like for DL-Lite?

No, since FO-rewritings need not exist:

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



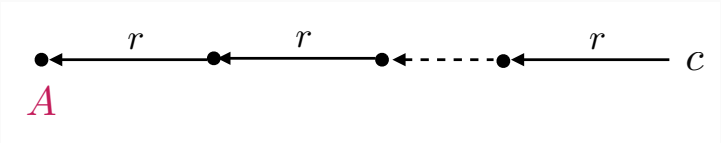
unbounded propagation of A along r

FO-rewriting for IQs?

Could we alternatively use FO-rewriting, like for DL-Lite?

No, since 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

However, we can instead give a Datalog rewriting

On the next slide, we recall what are Datalog queries

Datalog: logic programming language, studied in databases as natural means of expressing recursive queries

Datalog program Π is a finite set of rules of the form

$$\underbrace{P_0(\vec{t_0})}_{\text{head}} \leftarrow \underbrace{P_1(\vec{t_1}), \ldots, P_n(\vec{t_n})}_{\text{body}}$$

where the P_i are relation symbols (of any arity), and every variable in the head must occur in the body

Datalog: logic programming language, studied in databases as natural means of expressing recursive queries

Datalog program Π is a finite set of rules of the form

$$\underbrace{P_0(\vec{t_0})}_{\text{head}} \leftarrow \underbrace{P_1(\vec{t_1}), \ldots, P_n(\vec{t_n})}_{\text{body}}$$

where the P_i are relation symbols (of any arity), and every variable in the head must occur in the body

Datalog query = pair (Π, Goal), where Π program, Goal relation

Answers to (Π, Goal) on D:
tuples $\vec{a}$ such that $\mathit{Goal}(\vec{a})$ derivable / entailed

Let $\Pi_{\mathcal{T}}$ be the Datalog program with the following rules:

- $B(x) \leftarrow A(x)$, for every $A \sqsubseteq B \in \text{close}(\mathcal{T})$
- $B(x) \leftarrow A_1(x), A_2(x)$, for every $A_1 \sqcap A_2 \sqsubseteq B \in \text{close}(\mathcal{T})$
- $B(x) \leftarrow r(x, y), A(y)$, for every $\exists r.A \in \text{close}(\mathcal{T})$

These rules simply translate the ABox completion rules

Theorem: $(\Pi_{\mathcal{T}}, A)$ is a (Datalog) rewriting of $A(x)$ w.r.t. $\mathcal{T}$

Can evaluate rewriting using a Datalog engine (LogicBlox, RDFox)

Before moving to CQs, define **canonical model of $\mathcal{EL}$ KB** $\mathcal{K} = (\mathcal{T}, \mathcal{A})$

Domain $\Delta^{\mathcal{C}_\mathcal{K}}$ will again consist of words, this time of the form $ar_1A_1r_2A_2 \dots r_nA_n$ ($n \geq 0$), where $a \in \mathsf{Ind}(\mathcal{A})$, $r_i \in \mathsf{N_R}$, $A_i \in \mathsf{N_C}$, and:

- if $n > 0$, then $\mathcal{K} \models \exists r_1.A_1(a)$
- for every $1 \leq i < n$, $\mathcal{T} \models A_i \sqsubseteq \exists r_{i+1}.A_{i+1}$

Interpretation of concept, role, and individual names:

- $a^{\mathcal{C}_\mathcal{K}} = a$
- $B^{\mathcal{C}_\mathcal{K}} = \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models B(a)\} \cup \{ar_1A_1r_2A_2 \dots r_nA_n \mid \mathcal{T} \models A_n \sqsubseteq B\}$
- $r^{\mathcal{C}_\mathcal{K}} = \{(a, b) \in \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models r(a, b)\} \cup \{(e_1, e_2) \mid e_2 = e_1rA\}$

Can adapt the rewriting $rew^{\mathcal{T}}_{\text{core}}(q)$ from DL-Lite to $\mathcal{EL}$ as follows:

- in Step 2, pick a pair $r\,A$ (rather than role R), and simpler conditions to check (since no role inclusions)
- in Step 3, pick a concept name B such that $\mathcal{T} \models B \sqsubseteq \exists r.A$

Theorem: $rew^{\mathcal{T}}_{\text{core}}(q)$ is a UCQ-rewriting of q w.r.t. complete ABoxes

Can adapt the rewriting $rew^{\mathcal{T}}_{\text{core}}(q)$ from DL-Lite to $\mathcal{EL}$ as follows:

- in Step 2, pick a pair rA (rather than role R), and simpler conditions to check (since no role inclusions)
- in Step 3, pick a concept name B such that $\mathcal{T} \models B \sqsubseteq \exists r.A$

Theorem: $rew^{\mathcal{T}}_{\text{core}}(q)$ is a UCQ-rewriting of q w.r.t. complete ABoxes

Can combine with $\Pi_{\mathcal{T}}$ to get a Datalog rewriting

Theorem: The following Datalog query is a rewriting of CQ q w.r.t. $\mathcal{T}$

$$(\Pi_{\mathcal{T}} \cup \Pi_{q,\mathcal{T}}, Goal) \text{ where } \Pi_{q,\mathcal{T}} = \{Goal(\vec{x}) \leftarrow q'(\vec{x}) \mid q' \in rew^{\mathcal{T}}_{\text{core}}(q)\}$$

Theorem. Axiom entailment, classification, and instance checking over $\mathcal{EL}$ KBs are PTIME-complete in data and combined complexity.

- upper bounds from completion procedure from presented earlier
- very efficient in practice - can classify SNOMED in few seconds

Theorem. Axiom entailment, classification, and instance checking over $\mathcal{EL}$ KBs are PTIME-complete in data and combined complexity.

- upper bounds from completion procedure from presented earlier
- very efficient in practice - can classify SNOMED in few seconds

Theorem. CQ answering over $\mathcal{EL}$ KBs is PTIME-complete in data complexity and NP-complete in combined complexity.

- PTIME data complexity due to Datalog rewriting
- NP upper bound: guess homomorphism into canonical model

We can add all of the following without PTIME combined complexity:

- $\bot$
- $\mathrm{dom}(r) \sqsubseteq C$, $\mathrm{range}(r) \sqsubseteq C$
- role inclusions $r \sqsubseteq s$

We can add all of the following without PTIME combined complexity:

- $\bot$
- $\mathrm{dom}(r) \sqsubseteq C$, $\mathrm{range}(r) \sqsubseteq C$
- role inclusions $r \sqsubseteq s$

But adding any of the following makes reasoning EXPTIME-hard:

- negation $\neg$
- disjunction $\sqcup$
- at-least or at-most restrictions: $\geq 2R$, $\leq 1R$
- functional roles ($\mathrm{funct}\, R$)
- inverse roles R^-

MORE EXPRESSIVE HORN DLS

Can adapt approach to richer DLs like $\mathcal{ELI}$ and Horn-$\mathcal{SHIQ}$

(basis for Clipper system, Eiter et al, AAAI 2012)

MORE EXPRESSIVE HORN DLS

Can adapt approach to richer DLs like $\mathcal{ELI}$ and Horn-$\mathcal{SHIQ}$

(basis for Clipper system, Eiter et al, AAAI 2012)

Salient differences:

- canonical model contains words $aR_1M_1 \ldots R_nM_n$ where R_i roles and M_i are sets of concept names
- need different set of completion rules to handle e.g. inverse roles

$$\frac{A \sqsubseteq \exists R.D \quad \exists R^-.B \sqsubseteq E}{A \sqcap B \sqsubseteq \exists R.(D \sqcap E)}$$

- adapt rewriting to guess RM and M' (M, M' sets of concepts)

MORE EXPRESSIVE HORN DLS

Can adapt approach to richer DLs like $\mathcal{ELI}$ and Horn-$\mathcal{SHIQ}$

(basis for Clipper system, Eiter et al, AAAI 2012)

Salient differences:

- canonical model contains words $aR_1M_1 \dots R_nM_n$ where R_i roles and M_i are sets of concept names
- need different set of completion rules to handle e.g. inverse roles

$$\frac{A \sqsubseteq \exists R.D \quad \exists R^-.B \sqsubseteq E}{A \sqcap B \sqsubseteq \exists R.(D \sqcap E)}$$

- adapt rewriting to guess RM and M' (M, M' sets of concepts)

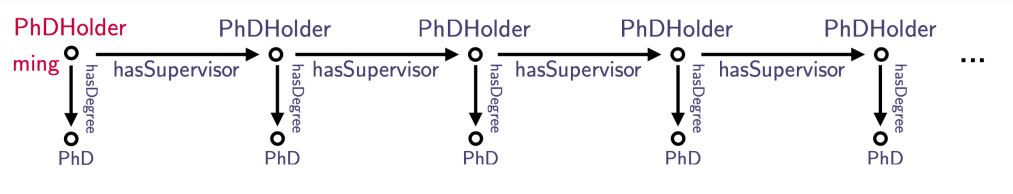
Still get a Datalog rewriting $\Rightarrow$ PTIME data complexity

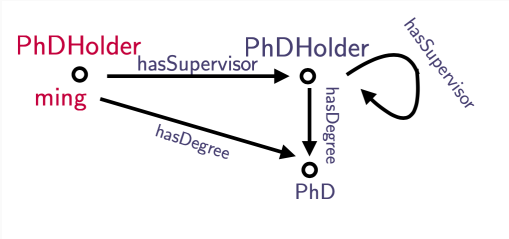
COMBINED APPROACH

Combined approach exploits the canonical model in a different way to reduce OMQA to database query evaluation

Combined approach exploits the canonical model in a different way to reduce OMQA to database query evaluation

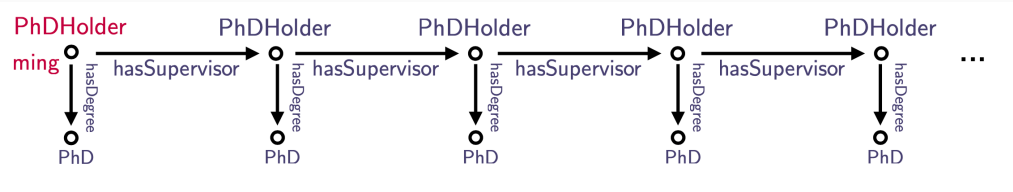
Idea: store a compact version of the canonical model

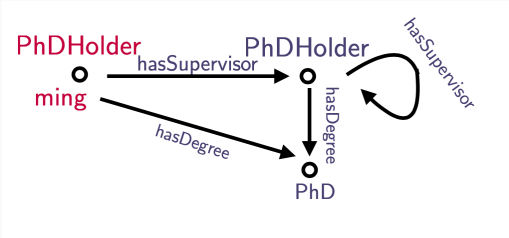




Combined approach exploits the canonical model in a different way to reduce OMQA to database query evaluation

Idea: store a compact version of the canonical model

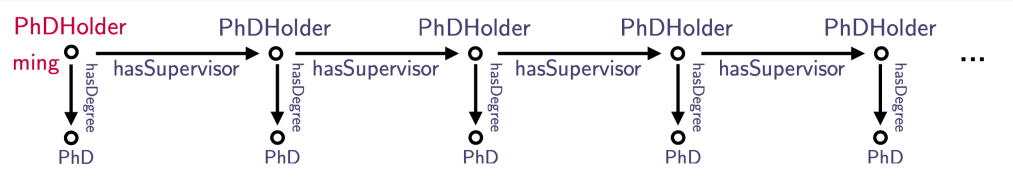


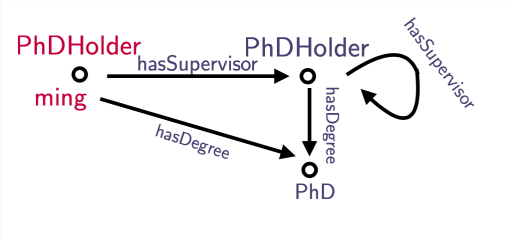


In the case of $\mathcal{EL}$: identify $ar_1A_1 \dots r_nA_n$ with same last concept

Combined approach exploits the canonical model in a different way to reduce OMQA to database query evaluation

Idea: store a compact version of the canonical model





In the case of $\mathcal{EL}$: identify $ar_1A_1 \dots r_nA_n$ with same last concept

Evaluate original query over compact canonical model

- superset of certain answers
- rewriting / filtering phase to block incorrect answers

First proposed for $\mathcal{EL}$, adapted to many different Horn DLs

COMPACT CANONICAL MODEL FOR EL KB

Consider an $\mathcal{EL}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with $\mathcal{T}$ in normal form.

We begin by defining an initial interpretation $\mathcal{I}_0$ as follows:

- $\Delta^{\mathcal{I}_0} = \mathsf{Ind}(\mathcal{A}) \cup \{w_A \mid A \text{ concept name appearing in } \mathcal{K}\} \cup \{w_\top\}$
- $A^{\mathcal{I}_0} = \{a \mid A(a) \in \mathcal{A}\} \cup \{w_A\}$
- $r^{\mathcal{I}_0} = \{(a, b) \mid r(a, b) \in \mathcal{A}\}$

We obtain $\mathcal{I}_{i+1}$ from $\mathcal{I}_i$ by applying one of the following rules:

R0 If $e \in A^{\mathcal{I}_i}$ and $A \sqsubseteq B \in \mathcal{T}$ and $e \notin B^{\mathcal{I}_i}$
then $B^{\mathcal{I}_{i+1}} = B^{\mathcal{I}_i} \cup \{e\}$

R1 If $e \in (A_1 \sqcap A_2)^{\mathcal{I}_i}$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $e \notin B^{\mathcal{I}_i}$,
then $B^{\mathcal{I}_{i+1}} = B^{\mathcal{I}_i} \cup \{e\}$

R2 If $e \in A^{\mathcal{I}_i}$ and $A \sqsubseteq \exists r.B \in \mathcal{T}$ and $(e, w_B) \notin r^{\mathcal{I}_i}$,
then $r^{\mathcal{I}_{i+1}} = r^{\mathcal{I}_i} \cup \{(e, w_B)\}$

R3 If $(d, e) \in r^{\mathcal{I}_i}$ and $e \in A^{\mathcal{I}_i}$ and $\exists r.A \sqsubseteq B \in \mathcal{T}$ and $d \notin B^{\mathcal{I}_i}$,
then $B^{\mathcal{I}_{i+1}} = B^{\mathcal{I}_i} \cup \{d\}$

Once no more rules apply, result is compact canonical model

REFERENCES: HORN DLS

DL-Lite family:

Calvanese, De Giacomo, Lembo, Lenzerini, Rosati: Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. J. Automated Reasoning, 2007.

$\mathcal{EL}$ family:

Baader, Brandt, Lutz: Pushing the EL Envelope. IJCAI 2005.

Detailed introduction to OMQA in Horn DLs:

Bienvenu and Ortiz: Ontology-Mediated Query Answering with Data-Tractable Description Logics. Lecture Notes of the 11th International Reasoning Web Summer School, 2015.

Short survey of OMQA (with mappings, database considerations):

Xiao, Calvanese, Kontchakov, Lembo, Poggi, Rosati, Zakharyaschev: Ontology-based data access: A survey. IJCAI 2018.

References: Algorithms & Systems

Seminal work on query rewriting for OMQA

- PerfectRef algo (JAR'07 previous slide), Quonto system

State-of-the-art DL-Lite system (employs tree-witness rewriting)

Calvanese, Cogrel, Komla-Ebri, Kontchakov, Lanti, Rezk, Rodriguez-Muro, Xiao: Ontop: Answering SPARQL queries over relational databases. Semantic Web, 2017.

First rewriting algorithm for Horn-$\mathcal{SHIQ}$: (basis for algo in this course)

Eiter, Ortiz, Simkus, Tran, Xiao: Query Rewriting for Horn-SHIQ Plus Rules. AAAI 2012. (Clipper system)

Combined approach:

Kontchakov, Lutz, oman, Wolter, Zakharyaschev: The Combined Approach to Ontology-Based Data Access. IJCAI 2011.

There are **many more algos + systems**, see the **surveys for pointers**!