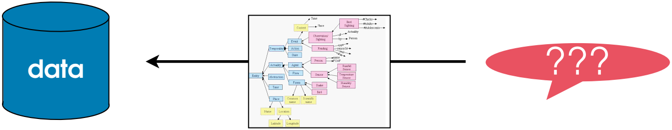
ONTOLOGY-MEDIATED QUERY ANSWERING

Meghyn Bienvenu (*CNRS & Université de Bordeaux*)

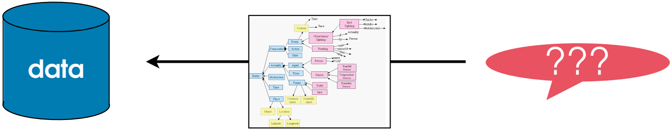


incomplete database
(ground facts)

ontology
(logical theory)

user query

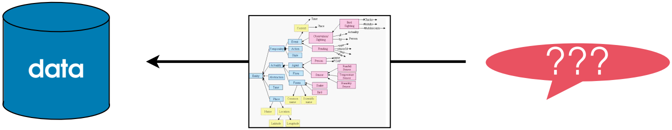
ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)



patient data	**medical knowledge**	**user query**
"Melanie has listeriosis" "Paul has Lyme disease"	"Listeriosis & Lyme disease are bacterial infections"	"Find all patients with bacterial infections"

The ontology (logical theory) specifies:

- terminology (or vocabulary) of the domain
- semantic relationships between terms
  - relations of specificity or generality, equivalence, disjointness, …



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**

The ontology (logical theory) specifies:

- terminology (or vocabulary) of the domain
- semantic relationships between terms
  - relations of specificity or generality, equivalence, disjointness, …

To **standardize the terminology** of an application domain

- meaning of terms is constrained, so less misunderstandings
- by adopting a common vocabulary, easy to share information

WHAT ARE ONTOLOGIES GOOD FOR?

To **standardize the terminology** of an application domain

- meaning of terms is constrained, so less misunderstandings
- by adopting a common vocabulary, easy to share information

To present an **intuitive and unified view of data sources**

- ontology can be used to **enrich the data vocabulary**, making it easier for users to formulate their queries
- especially useful when **integrating multiple data sources**

WHAT ARE ONTOLOGIES GOOD FOR?

To **standardize the terminology** of an application domain

- meaning of terms is constrained, so less misunderstandings
- by adopting a common vocabulary, easy to share information

To present an **intuitive and unified view of data sources**

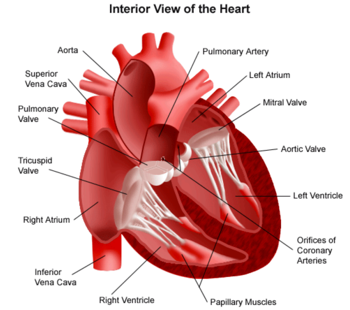
- ontology can be used to **enrich the data vocabulary**, making it easier for users to formulate their queries
- especially useful when **integrating multiple data sources**

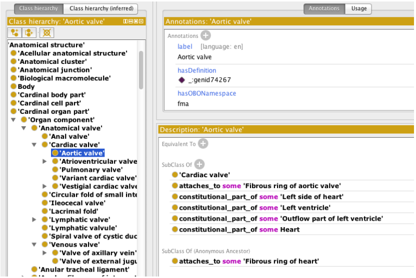
To support **automated reasoning**

- uncover implicit connections between terms, errors in modelling
- **exploit knowledge in the ontology during query answering**, to get back a **more complete set of answers** to queries

General medical ontologies: SNOMED CT ($\sim$ 400,000 terms!), GALEN

Specialized ontologies: FMA (anatomy), NCI (cancer), ...



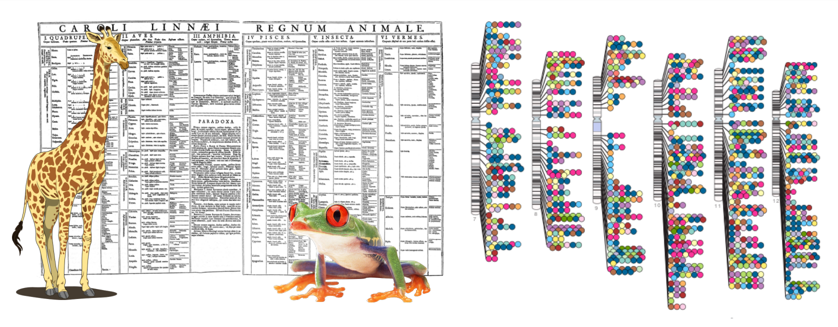


Querying & exchanging medical records (find patients for medical trials)

- myocardial infarction vs. MI vs. heart attack vs. 410.0

Supports tools for annotating and visualizing patient data (scans, x-rays)

Hundreds of ontologies at BioPortal (http://bioportal.bioontology.org/):
Gene Ontology (GO), Cell Ontology, Pathway Ontology, Plant Anatomy, ...



Help scientists share, query, & visualize experimental data

APPLICATIONS OF OMQA: ENTREPRISE INFORMATION SYSTEMS

Companies and organizations have lots of data

- need easy and flexible access to support decision-making

Example industrial projects:

- Public debt data: Sapienza Univ. & Italian Department of Treasury
- Energy sector: Optique EU project (several univ, StatOil, Siemens)

Ontologies typically described using logic-based formalisms

Description logics (DLs)

- family of decidable fragments of first-order logic (FO)
- concise variable-free syntax
- only unary and binary predicates

Ontologies typically described using logic-based formalisms

Description logics (DLs)

- family of decidable fragments of first-order logic (FO)
- concise variable-free syntax
- only unary and binary predicates

Existential rules (aka Datalog$^{+/-}$, tuple-generating dependencies)

- family of languages of rules of the form $\forall x(\exists \vec{y}\, \varphi(\vec{x}, \vec{y}) \rightarrow \exists \vec{z}\, \psi(\vec{x}, \vec{z}))$ where $\varphi(\vec{x}, \vec{y})$ and $\psi(\vec{x}, \vec{z})$ are conjunctions of atoms
- can have predicates of arbitrary arity

Ontologies typically described using logic-based formalisms

Description logics (DLs)

- family of decidable fragments of first-order logic (FO)
- concise variable-free syntax
- only unary and binary predicates

Existential rules (aka Datalog$^{+/-}$, tuple-generating dependencies)

- family of languages of rules of the form $\forall x(\exists \vec{y}\, \varphi(\vec{x}, \vec{y}) \rightarrow \exists \vec{z}\, \psi(\vec{x}, \vec{z}))$ where $\varphi(\vec{x}, \vec{y})$ and $\psi(\vec{x}, \vec{z})$ are conjunctions of atoms
- can have predicates of arbitrary arity

Two approaches are incomparable and complementary

Description logics (DLs) are:

- family of knowledge representation languages
- popular means for specifying ontologies
- range from fairly simple to highly expressive
- basis of the web ontology language OWL (W3C standard)

Description logics (DLs) are:

- family of knowledge representation languages
- popular means for specifying ontologies
- range from fairly simple to highly expressive
- basis of the web ontology language OWL (W3C standard)

Formally: correspond to decidable fragments of first-order logic

- inherit well-defined semantics
- succinct, variable-free syntax

Description logics (DLs) are:

- family of knowledge representation languages
- popular means for specifying ontologies
- range from fairly simple to highly expressive
- basis of the web ontology language OWL (W3C standard)

Formally: correspond to decidable fragments of first-order logic

- inherit well-defined semantics
- succinct, variable-free syntax

Computational properties well understood (decidability, complexity)

Description logics (DLs) are:

- family of knowledge representation languages
- popular means for specifying ontologies
- range from fairly simple to highly expressive
- basis of the web ontology language OWL (W3C standard)

Formally: correspond to decidable fragments of first-order logic

- inherit well-defined semantics
- succinct, variable-free syntax

Computational properties well understood (decidability, complexity)

Many implemented reasoners and tools available for use

Sets of **existential rules** (aka tuple-generating dependencies):

$$\forall\vec{x}\ \varphi(\vec{x}) \rightarrow \exists\vec{y}\ \psi(\vec{x},\vec{y})$$

Sets of existential rules (aka tuple-generating dependencies):

$$\forall \vec{x}\ \varphi(\vec{x}) \rightarrow \exists \vec{y}\ \psi(\vec{x}, \vec{y})$$

Undecidable in general, different restrictions to achieve decidability

- forward chaining (chase) halts
- backward chaining (rewriting) halts
- tree-like models

Sets of existential rules (aka tuple-generating dependencies):

$$\forall \vec{x}\ \varphi(\vec{x}) \rightarrow \exists \vec{y}\ \psi(\vec{x}, \vec{y})$$

Undecidable in general, different restrictions to achieve decidability

- forward chaining (chase) halts
- backward chaining (rewriting) halts
- tree-like models

Many complexity & decidability results, few implemented algorithms

Sets of existential rules (aka tuple-generating dependencies):

$$\forall \vec{x}\ \varphi(\vec{x}) \rightarrow \exists \vec{y}\ \psi(\vec{x}, \vec{y})$$

Undecidable in general, different restrictions to achieve decidability

- forward chaining (chase) halts
- backward chaining (rewriting) halts
- tree-like models

Many complexity & decidability results, few implemented algorithms

Can consider extensions allowing equality, disjunction in rule heads

- such extensions less well understood

Introduction to DLs

Introduction to OMQA

Techniques for OMQA with Lightweight DLs

Research Questions in OMQA

INTRODUCTION TO DLS

Building blocks:

- concept names (unary predicates, classes) Prof Fellow Course
- role names (binary predicates, properties) teaches headOf
- individual names (constants) anna, cs101

Building blocks:

- concept names (unary predicates, classes) — Prof Fellow Course
- role names (binary predicates, properties) — teaches headOf
- individual names (constants) — anna, cs101

Constructors to build complex concepts and roles $\sqcup, \sqcap, \neg, \forall, \exists, \ldots$

$\mathsf{Faculty} \sqcap \neg\mathsf{Prof}$ $\exists\mathsf{teaches}.\mathsf{GradCourse}$ $\mathsf{teaches}^{-}$

Note: allowed constructors **depends on chosen DL**

DL knowledge base (KB) = ABox (data) + TBox (ontology)

ABox = finite set of **concept and role assertions (facts)**

Prof(anna) teaches(tom, cs101)

TBox (ontology) = finite set of **axioms**

DL knowledge base (KB) = ABox (data) + TBox (ontology)

ABox = finite set of concept and role assertions (facts)

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

TBox (ontology) = finite set of axioms

- concept inclusions $C \sqsubseteq D$ (C, D possibly complex concepts)

  $\mathsf{Prof} \sqsubseteq \mathsf{Faculty}$ $\exists\mathsf{teaches}.\mathsf{GradCourse} \sqsubseteq \mathsf{Prof}$

- role inclusions $R \sqsubseteq S$ (R, S possibly complex roles)

  $\mathsf{taughtBy} \sqsubseteq \mathsf{teaches}^-$ $\mathsf{headOf} \sqsubseteq \mathsf{memberOf}$

Note: allowed axioms **depends on chosen DL**

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 **course** is **either an undergrad or grad course**

$$\mathsf{Course} \sqsubseteq \mathsf{UCourse} \sqcup \mathsf{GCourse}$$

The relation takesCourse **connects students to courses**

$$\exists \mathsf{takesCourse}.\top \sqsubseteq \mathsf{Student} \quad \exists \mathsf{takesCourse}^{-}.\top \sqsubseteq \mathsf{Course}$$

Every student takes **at least 2** and **at most 5 courses**

$$\mathsf{Student} \sqsubseteq\ \geq 2\mathsf{takesCourse}.\top \sqcap \leq 5\mathsf{takesCourse}.\top$$

MORE EXAMPLES OF TBOX AXIOMS

Every grad student is supervised by some faculty member

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

The academic ancestor relation is transitive

$$\mathsf{supervisedBy} \sqsubseteq \mathsf{academicAnc} \qquad \mathit{trans}(\mathsf{academicAnc})$$

Students who only take grad-level courses are grad students

$$\mathsf{Student} \sqcap \forall \mathsf{takesCourse}.\mathsf{GCourse} \sqsubseteq \mathsf{GStudent}$$

FO translation:
$\forall x \; (\mathsf{Student}(x) \land (\forall y \; \mathsf{takesCourse}(x,y) \rightarrow \mathsf{GCourse}(y)) \rightarrow \mathsf{GStudent}(x)$

Interpretation $\mathcal{I}$ ("possible world") (like in first-order logic)

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

Interpretation $\mathcal{I}$ ("possible world") (like in first-order logic)

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

Interpretation function $\cdot^{\mathcal{I}}$ extends to complex concepts and roles:

$\top$	$\Delta^{\mathcal{I}}$
$\bot$	$\emptyset$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$C_1 \sqcap C_2$	$C_1{}^{\mathcal{I}} \cap C_2{}^{\mathcal{I}}$
$\exists R.C$	$\{d_1 \mid \text{there exists } (d_1, d_2) \in R^{\mathcal{I}} \text{ with } d_2 \in C^{\mathcal{I}}\}$
$\forall R.C$	$\{d_1 \mid d_2 \in C^{\mathcal{I}} \text{ for all } (d_1, d_2) \in R^{\mathcal{I}}\}$
r^-	$\{(d_2, d_1) \mid (d_1, d_2) \in r^{\mathcal{I}}\}$

Satisfaction in an interpretation

- $\mathcal{I}$ satisfies $C \sqsubseteq D \quad \Leftrightarrow \quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $R \sqsubseteq S \quad \Leftrightarrow \quad R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $A(a) \quad \Leftrightarrow \quad a^{\mathcal{I}} \in A^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $r(a, b) \quad \Leftrightarrow \quad (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

Satisfaction in an interpretation

- $\mathcal{I}$ satisfies $C \sqsubseteq D \quad \Leftrightarrow \quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $R \sqsubseteq S \quad \Leftrightarrow \quad R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $A(a) \quad \Leftrightarrow \quad a^{\mathcal{I}} \in A^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $r(a, b) \quad \Leftrightarrow \quad (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

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

$\mathcal{K}$ **is satisfiable** = $\mathcal{K}$ has at least one model

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

Satisfaction in an interpretation

- $\mathcal{I}$ satisfies $C \sqsubseteq D \quad \Leftrightarrow \quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $R \sqsubseteq S \quad \Leftrightarrow \quad R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $A(a) \quad \Leftrightarrow \quad a^{\mathcal{I}} \in A^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $r(a, b) \quad \Leftrightarrow \quad (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

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

$\mathcal{K}$ **is satisfiable** = $\mathcal{K}$ has at least one model

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

Basic reasoning tasks:

- KB satisfiability: decide whether $\mathcal{K}$ is satisfiable
- Axiom entailment: decide whether $\mathcal{T} \models \alpha$ (with α an axiom)
  - Classification: decide $\mathcal{T} \models A \sqsubseteq B$ for every pair A, B of concept names

ABoxes are interpreted under the **open-world assumption**

- facts that are not in the ABox may still be true (e.g. can be inferred by exploiting information in the TBox)
- differs from the typical closed-world assumption from databases, where absent facts are interpreted as false

DIFFERENCES WITH DATABASE SETTING

ABoxes are interpreted under the **open-world assumption**

- facts that are not in the ABox may still be true (e.g. can be inferred by exploiting information in the TBox)
- differs from the typical closed-world assumption from databases, where absent facts are interpreted as false

Semantics in terms of **arbitrary (possibly infinite) interpretations**

- differs from finite models considered in databases
- may consider an alternative semantics based upon finite models
  - for some DLs, two semantics behave the same
  - for others, often possible to reduce to arbitrary model reasoning

For today's talk, we'll focus on standard semantics (arbitrary models)

Prototypical expressive description logic $\mathcal{ALC}$:

- Concepts: $C := \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$
- TBox axioms: only concept inclusions

Prototypical expressive description logic $\mathcal{ALC}$:

- Concepts: $C := \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$
- TBox axioms: only concept inclusions

Highly expressive description logic $\mathcal{SHOIQ}$ ($\sim$ OWL 2)

- Extends $\mathcal{ALC}$ with:
  - number restrictions $\leq nR.C$ $\geq nR.C$ and nominals $\{a\}$
  - role inclusions $R_1 \sqsubseteq R_2$ and transitivity axioms $trans(R)$
  - inverse roles r^- (allowed in all types of axioms)

Prototypical expressive description logic $\mathcal{ALC}$:

- Concepts: $C := \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$
- TBox axioms: only concept inclusions

Highly expressive description logic $\mathcal{SHOIQ}$ ($\sim$ OWL 2)

- Extends $\mathcal{ALC}$ with:
  - number restrictions $\leq nR.C$ $\geq nR.C$ and nominals $\{a\}$
  - role inclusions $R_1 \sqsubseteq R_2$ and transitivity axioms $trans(R)$
  - inverse roles r^- (allowed in all types of axioms)

"Lightweight" description logic $\mathcal{EL}$ ($\sim$ OWL 2 EL)

- Fragment of $\mathcal{ALC}$ with concepts $C := \top \mid A \mid C \sqcap C \mid \exists r.C$

Prototypical expressive description logic $\mathcal{ALC}$:

- Concepts: $C := \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$
- TBox axioms: only concept inclusions

Highly expressive description logic $\mathcal{SHOIQ}$ (∼ OWL 2)

- Extends $\mathcal{ALC}$ with:
  - number restrictions $\leq nR.C$ $\geq nR.C$ and nominals $\{a\}$
  - role inclusions $R_1 \sqsubseteq R_2$ and transitivity axioms $trans(R)$
  - inverse roles r^- (allowed in all types of axioms)

“Lightweight” description logic $\mathcal{EL}$ (∼ OWL 2 EL)

- Fragment of $\mathcal{ALC}$ with concepts $C := \top \mid A \mid C \sqcap C \mid \exists r.C$

$\mathcal{ALCI}$ = extension of $\mathcal{ALC}$ with inverse roles ($\mathcal{I}$)

$\mathcal{ELH}$ = extension of $\mathcal{EL}$ with role inclusions ($\mathcal{H}$)

INTRODUCTION TO OMQA

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

Faculty(x)   teaches(x, y)

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

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

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

conjunctions of atoms, some variables can be existentially quantified

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

(find all faculty members that teach something)

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

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

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

conjunctions of atoms, some variables can be existentially quantified

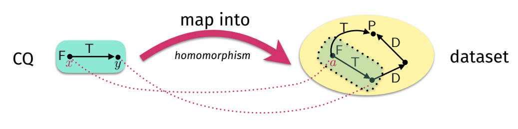
$$\exists y.\, \text{Faculty}(x) \land \text{teaches}(x, y)$$

(find all faculty members that teach something)

Ontology-mediated query (OMQ):

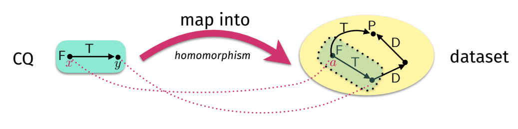
pair $(\mathcal{T}, q)$ with $\mathcal{T}$ a TBox and q a query (IQ / CQ)

Answering CQs in database setting



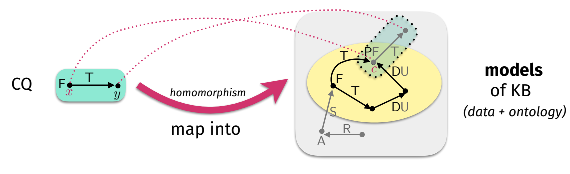
database D + query q $\rightsquigarrow$ set of answers $ans(q, D)$

Answering CQs in database setting



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

Answering CQs in the presence of a TBox (ontology)



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

Question: how to combine the answers from different models?

Certain answers:

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

Question: how to combine the answers from different models?

Certain answers:

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

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

OMQA EXAMPLE

TBox (ontology):

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

ABox (data):

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

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

OMQA EXAMPLE

TBox (ontology):

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

ABox (data):

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

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

INTRACTABILITY FOR EXPRESSIVE DLS

Recall the DL $\mathcal{ALC}$: $C := \top \mid \bot \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$

Satisfiability, IQ answering, and CQ answering in $\mathcal{ALC}$ are:

- EXPTIME-complete in combined complexity
- coNP-complete in data complexity

Even worse news for CQ answering in $\mathcal{ALCI}$ (= $\mathcal{ALC}$ + inverse roles):

- 2EXPTIME-complete in combined complexity
- coNP-complete in data complexity

OMQA WITH LIGHTWEIGHT DLS

Negative results led to proposal of **new DLs with lower complexity**

Negative results led to proposal of **new 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 ($\sim$ backward chaining)

Negative results led to proposal of new 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 ($\sim$ backward chaining)

$\mathcal{EL}$ 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: saturation ($\sim$ forward chaining)

Negative results led to proposal of new 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 ($\sim$ backward chaining)

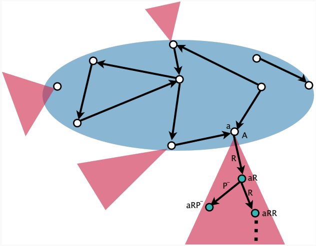
$\mathcal{EL}$ family of DLs (basis for OWL 2 EL)

- designed to allow efficient reasoning with large ontologies
- well suited for medical and life science applications
- key technique: saturation ($\sim$ forward chaining)

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

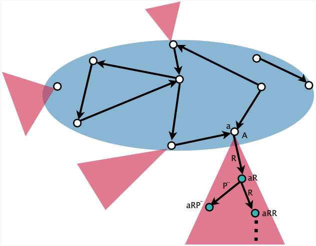
For Horn ontologies (no form of disjunction) like DL-Lite, $\mathcal{EL}$:
enough to consider a single canonical model

- 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)
- give correct answer to all CQs



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)
- give correct answer to all CQs



OMQA with Horn DLs =

finding ways to
map the query into
the canonical model

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

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

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: $\mathsf{headOf} \sqsubseteq \mathsf{memberOf}$

Idea: reduce OMQA to database query evaluation

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

Advantage: harness efficiency of relational database systems

Idea: reduce OMQA to database query evaluation

- rewriting step: TBox $\mathcal{T}$ + query $q \rightsquigarrow$ 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 q w.r.t. TBox $\mathcal{T}$ 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: TBox $\mathcal{T}$ + query $q \rightsquigarrow$ 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 q w.r.t. TBox $\mathcal{T}$ 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

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

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

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

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

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

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

$$\begin{array}{lll} \mathsf{Prof} \sqsubseteq \mathsf{Faculty} & \mathsf{Fellow} \sqsubseteq \mathsf{Faculty} & \mathsf{Prof} \sqsubseteq \neg\mathsf{Fellow} \\ \mathsf{Prof} \sqsubseteq \exists\mathsf{teaches} & \exists\mathsf{teaches}^{-} \sqsubseteq \mathsf{Course} & \end{array}$$

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

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

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

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

Can focus w.l.o.g. on rewritings over consistent ABoxes

"Classic" approach works roughly as follows on input $(q, \mathcal{T})$:

- Start with $S = \{q\}$
- Until S stabilizes, pick some $q' \in S$ and do one of the following:
  - Apply an axiom in $\mathcal{T}$ to an atom in q', and add the result to S
  - Merge two variables x and y in q', and add the result to S
- Output the UCQ $\bigvee_{q' \in S} q'$

Can focus w.l.o.g. on rewritings over consistent ABoxes

"Classic" approach works roughly as follows on input $(q, \mathcal{T})$:

- Start with $S = \{q\}$
- Until S stabilizes, pick some $q' \in S$ and do one of the following:
  - Apply an axiom in $\mathcal{T}$ to an atom in q', and add the result to S
  - Merge two variables x and y in q', and add the result to S
- Output the **UCQ** $\bigvee_{q' \in S} q'$

Alternative "semantic" approach (also producing a UCQ):

- Big $\bigvee$ over possible decompositions of q into ABox and "tree parts"
- For atoms mapped to ABox: check if find implying atom in data
- For subqueries mapped to tree parts:
  - ensure generating atom in data, merge "root" variables

Data complexity:

- rewriting takes constant time, yields FO query
- upper bound from FO query evaluation: AC_0 ($AC_0 \subseteq$ LogSpace $\subseteq$ P)
- CQ answering is in AC_0 for data complexity

Data complexity:

- rewriting takes constant time, yields FO query
- upper bound from FO query evaluation: AC_0 ($\mathrm{AC}_0 \subseteq \textsc{LogSpace} \subseteq \mathrm{P}$)
- CQ answering is in AC_0 for data complexity

Combined complexity:

- 'guess' a disjunct of UCQ-rewriting and how to map it into ABox
- CQ answering is NP-complete (same as for DBs)
- IQ answering is NLogSpace-complete ($\textsc{NLogSpace} \subseteq \mathrm{P}$)

Many of the proposed rewriting algorithms produce unions of conjunctive queries (UCQs = $\vee$ of CQs)

Many of the proposed rewriting algorithms produce unions of conjunctive queries (UCQs = $\vee$ of CQs)

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

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

Many of the proposed rewriting algorithms produce unions of conjunctive queries (UCQs = $\vee$ of CQs)

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

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

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

Many of the proposed rewriting algorithms produce unions of conjunctive queries (UCQs = $\vee$ of CQs)

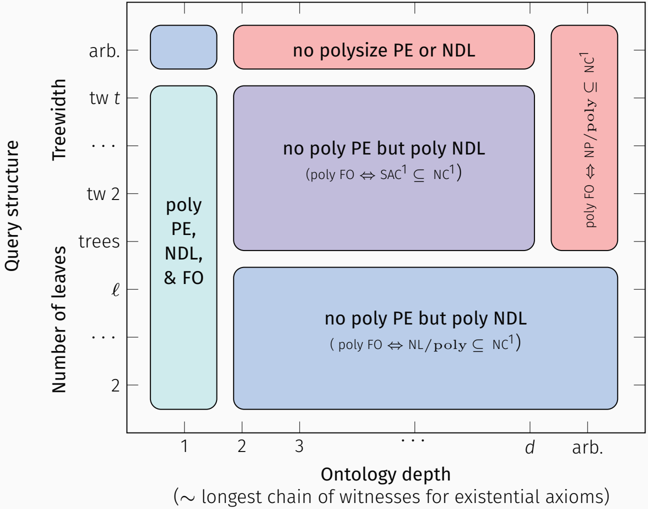
Not hard to see smallest UCQ-rewriting may be exponentially large:

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

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

What happens if we adopt other representations?

- positive existential queries (PE), non-recursive Datalog (NDL), first-order queries (FO) (for DL-Lite$_{\mathcal{R}}$ ontologies, so-called ‘pure’ rewritings)



Next consider the logic $\mathcal{EL}$:

- Concepts: $C := \top \mid A \mid C \sqcap C \mid \exists r.C$
- Only concept inclusions in the TBox, no inverse roles

Next consider the logic $\mathcal{EL}$:

- Concepts: $C := \top \mid A \mid C \sqcap C \mid \exists r.C$
- Only concept inclusions in the TBox, no inverse roles

Cannot use FO query rewriting approach for $\mathcal{EL}$:

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

Next consider the logic $\mathcal{EL}$:

- Concepts: $C := \top \mid A \mid C \sqcap C \mid \exists r.C$
- Only concept inclusions in the TBox, no inverse roles

Cannot use FO query rewriting approach for $\mathcal{EL}$:

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

We start with IQs and present a saturation-based approach.

Convenient to assume $\mathcal{EL}$ TBoxes given in normal form:

$$A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

(A, A_i, B concept names or $\top$)

TBox rules

$$\frac{A \sqsubseteq B_i \ (1 \leq i \leq n) \quad B_1 \sqcap \ldots \sqcap B_n \sqsubseteq D}{A \sqsubseteq D}\ \textsf{T1} \qquad \frac{A \sqsubseteq B \quad B \sqsubseteq \exists r.D}{A \sqsubseteq \exists r.D}\ \textsf{T2}$$

$$\frac{A \sqsubseteq \exists r.B \quad B \sqsubseteq D \quad \exists r.D \sqsubseteq E}{A \sqsubseteq E}\ \textsf{T3}$$

ABox rules

$$\frac{A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \quad A_i(a) \ (1 \leq i \leq n)}{B(a)}\ \textsf{A1} \qquad \frac{\exists r.B \sqsubseteq A \quad r(a,b) \quad B(b)}{A(a)}\ \textsf{A2}$$

Algorithm: apply rules exhaustively, check if $A(a)$ ($r(a,b)$) is present

EXAMPLE: SATURATION IN EL

$$\mathsf{PenneArrabiata} \sqsubseteq \exists \mathsf{hasIngred.ArrabiataSauce} \quad (1)$$
$$\mathsf{PenneArrabiata} \sqsubseteq \mathsf{PastaDish} \quad (2)$$
$$\mathsf{PastaDish} \sqsubseteq \mathsf{Dish} \quad (3)$$
$$\mathsf{PastaDish} \sqsubseteq \exists \mathsf{hasIngred.Pasta} \quad (4)$$
$$\mathsf{ArrabiataSauce} \sqsubseteq \exists \mathsf{hasIngred.Peperoncino} \quad (5)$$
$$\mathsf{Peperoncino} \sqsubseteq \mathsf{Spicy} \quad (6)$$
$$\exists \mathsf{hasIngred.Spicy} \sqsubseteq \mathsf{Spicy} \quad (7)$$
$$\mathsf{Spicy} \sqcap \mathsf{Dish} \sqsubseteq \mathsf{SpicyDish} \quad (8)$$
$$\mathsf{PenneArrabiata}(p). \quad (9)$$

EXAMPLE: SATURATION IN EL

$$\text{PenneArrabiata} \sqsubseteq \exists \text{hasIngred}.\text{ArrabiataSauce} \quad (1)$$
$$\text{PenneArrabiata} \sqsubseteq \text{PastaDish} \quad (2)$$
$$\text{PastaDish} \sqsubseteq \text{Dish} \quad (3)$$
$$\text{PastaDish} \sqsubseteq \exists \text{hasIngred}.\text{Pasta} \quad (4)$$
$$\text{ArrabiataSauce} \sqsubseteq \exists \text{hasIngred}.\text{Peperoncino} \quad (5)$$

$$\text{Peperoncino} \sqsubseteq \text{Spicy} \quad (6)$$
$$\exists \text{hasIngred}.\text{Spicy} \sqsubseteq \text{Spicy} \quad (7)$$
$$\text{Spicy} \sqcap \text{Dish} \sqsubseteq \text{SpicyDish} \quad (8)$$
$$\text{PenneArrabiata}(p). \quad (9)$$

$\text{ArrabSauce} \sqsubseteq \text{Spicy}$	T3 : (5), (6), (7)	(10)
$\text{PenneArrab} \sqsubseteq \text{Spicy}$	T3 : (1), (10), (7)	(11)
$\text{PenneArrab} \sqsubseteq \text{Dish}$	T1 : (2), (3)	(12)
$\text{PenneArrab} \sqsubseteq \exists \text{hasIngred}.\text{Pasta}$	T2 : (2), (4)	(13)
$\text{PenneArrab} \sqsubseteq \text{SpicyDish}$	T1 : (11), (12), (8)	(14)
$\text{Spicy}(\mathbf{p})$	A1 : (11), (9)	(15)
$\text{Dish}(\mathbf{p})$	A1 : (12), (9)	(16)
$\text{SpicyDish}(\mathbf{p})$	A1 : (16), (15)	(17)

Saturation approach is sound: everything derived is entailed

Saturation approach is sound: everything derived is entailed

Also complete for IQs, since for every ABox assertion α, we have:

$$\mathcal{K} \models \alpha \qquad \text{iff} \qquad \alpha \in \mathsf{sat}(\mathcal{K})$$

Saturation approach is **sound**: everything derived is entailed

Also **complete for IQs**, since for every ABox assertion α, we have:

$$\mathcal{K} \models \alpha \qquad \text{iff} \qquad \alpha \in \mathsf{sat}(\mathcal{K})$$

Note: does **not** make **all** consequences explicit

- can have infinitely many implied axioms $\rightsquigarrow$ would **not terminate!**
- so: only complete for some reasoning tasks

Saturation approach is sound: everything derived is entailed

Also complete for IQs, since for every ABox assertion α, we have:

$$\mathcal{K} \models \alpha \quad \text{iff} \quad \alpha \in \mathsf{sat}(\mathcal{K})$$

Note: does not make all consequences explicit

- can have infinitely many implied axioms $\rightsquigarrow$ would not terminate!
- so: only complete for some reasoning tasks

Procedure runs in polynomial time in $|\mathcal{K}|$. This is optimal:

IQ answering in $\mathcal{EL}$ is P-complete for data & combined complexity

Saw earlier that FO-rewritings may not exist

Saw earlier that FO-rewritings may not exist

Can show **Datalog rewritings exist for all CQs and $\mathcal{EL}$ ontologies**

- rules that generate all entailed facts over original individuals
  - can be obtained from axioms in sat($\mathcal{T}$)
- rules that check whether query holds
  - idea: rewrite query so enough to consider matches to original inds

Saw earlier that FO-rewritings may not exist

Can show **Datalog rewritings exist for all CQs and $\mathcal{EL}$ ontologies**

- rules that generate all entailed facts over original individuals
  - can be obtained from axioms in sat($\mathcal{T}$)
- rules that check whether query holds
  - idea: rewrite query so enough to consider matches to original inds

Complexity of CQ answering in $\mathcal{EL}$:

- **P-complete** in **data complexity**
- **NP-complete** in **combined complexity**

Saw earlier that FO-rewritings may not exist

Can show **Datalog rewritings exist for all CQs and $\mathcal{EL}$ ontologies**

- rules that generate all entailed facts over original individuals
  - can be obtained from axioms in sat($\mathcal{T}$)
- rules that check whether query holds
  - idea: rewrite query so enough to consider matches to original inds

Complexity of CQ answering in $\mathcal{EL}$:

- **P-complete** in **data complexity**
- **NP-complete** in **combined complexity**

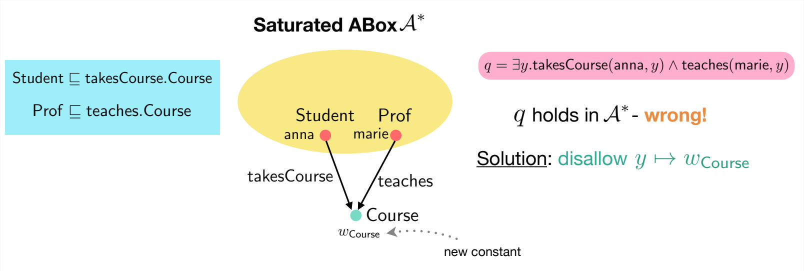
P data complexity extends to much richer Horn DLs

Way to use relational DBs to do CQ answering in $\mathcal{EL}$:

- saturate ABox using TBox axioms
  - introduce new individuals to witness existentials on LHS ($A \sqsubseteq \exists r.B$)
  - to ensure finite: reuse individuals as witnesses
  - can be viewed as compact representation of canonical model

Way to use relational DBs to do CQ answering in $\mathcal{EL}$:

- saturate ABox using TBox axioms
  - introduce new individuals to witness existentials on LHS ($A \sqsubseteq \exists r.B$)
  - to ensure finite: reuse individuals as witnesses
  - can be viewed as compact representation of canonical model
- evaluate query on saturated ABox $\Rightarrow$ superset of certain answers
- two strategies to block unsound answers:
  - add extra conditions to query
  - post-processing to identify and remove false answers



Lack of FO-rewritings is a worst-case result

- possible that rewritings do exist for many real-world OMQs

Lack of FO-rewritings is a worst-case result

- possible that rewritings do exist for many real-world OMQs

Motivates looking at FO-rewritability problem for $(\mathcal{L}, \mathcal{Q})$:

- input: OMQ $(\mathcal{T}, q)$ with $\mathcal{T}$ an $\mathcal{L}$-TBox and $q \in \mathcal{Q}$
- problem: decide if there exists an FO-rewriting of $(\mathcal{T}, q)$

Form of static analysis, related to Datalog boundedness

Lack of FO-rewritings is a worst-case result

- possible that rewritings do exist for many real-world OMQs

Motivates looking at FO-rewritability problem for $(\mathcal{L}, \mathcal{Q})$:

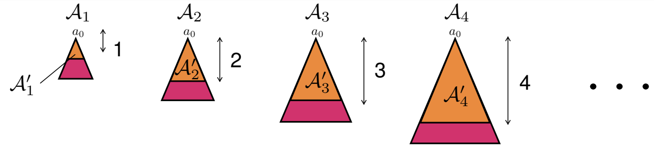
- input: OMQ $(\mathcal{T}, q)$ with $\mathcal{T}$ an $\mathcal{L}$-TBox and $q \in \mathcal{Q}$
- problem: decide if there exists an FO-rewriting of $(\mathcal{T}, q)$

Form of static analysis, related to Datalog boundedness

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

Characterization of non-existence of FO-rewriting

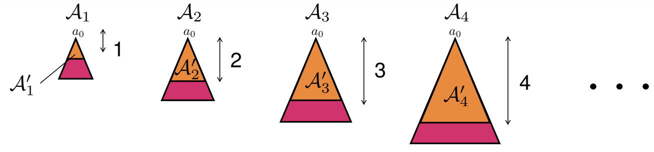
OMQ $(\mathcal{T}, A(x))$ is **not FO-rewritable** iff there exist tree-shaped ABoxes



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

Characterization of non-existence of FO-rewriting

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



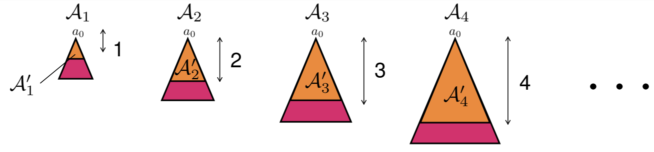
such that for all $i \geq 1$: $\mathcal{T}, \mathcal{A}_i \models A(a_0)$ and $\mathcal{T}, \mathcal{A}'_i \not\models A(a_0)$

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

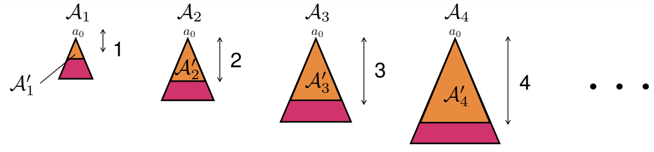
Pumping argument: enough to find ABox of **particular finite size** k_0

- 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

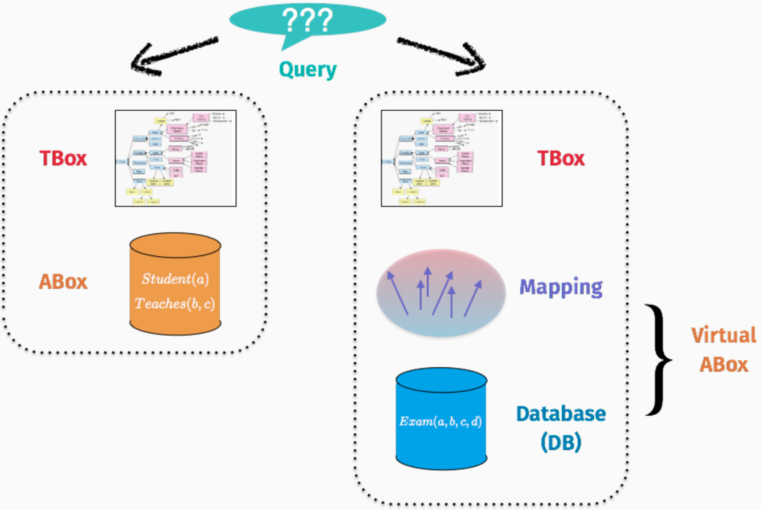
- desired ABox $\mathcal{A}_{k_0}$ exists $\Rightarrow$ can construct full sequence of ABoxes

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

Can **generalize this technique to handle CQs and richer Horn DLs**

LINKING TO EXISTING DATA VIA MAPPINGS

TWO VARIANTS OF OMQA



So far: data given as ABox (unary + binary facts using TBox predicates)

Problem: how to apply the approach to existing relational data (arbitrary arity, different vocabulary)?

Solution: use mapping that specifies relationship between the database relations and the ontology predicates

Formally: mapping assertions of the form $\forall \vec{x}\, \varphi(\vec{x}) \rightarrow \psi(\vec{x})$ where:

- $\varphi(\vec{x})$ is an query formulated using DB relations
- $\psi(\vec{x})$ is a query in the ontology vocabulary

Global-as-view (GAV) mappings: ψ is an atom (without $\exists$-vars)

Database D + mapping $\mathcal{M}$ $\rightsquigarrow$ ABox $\mathcal{A}_{\mathcal{M},D}$

Models of $\langle \mathcal{T}, \mathcal{M}, D \rangle$ = models of the KB $\langle \mathcal{T}, \mathcal{A}_{\mathcal{M},D} \rangle$

Certain answers to q **w.r.t.** $\langle \mathcal{T}, \mathcal{M}, D \rangle$ =
tuples of **constants a from** $D \cup \mathcal{M}$ such that $\langle \mathcal{T}, \mathcal{A}_{\mathcal{M},D} \rangle \models q(\mathbf{a})$

Database D + mapping $\mathcal{M}$ $\rightsquigarrow$ ABox $\mathcal{A}_{\mathcal{M},D}$

Models of $\langle \mathcal{T}, \mathcal{M}, D \rangle$ = models of the KB $\langle \mathcal{T}, \mathcal{A}_{\mathcal{M},D} \rangle$

Certain answers to q w.r.t. $\langle \mathcal{T}, \mathcal{M}, D \rangle$ =
tuples of constants $\mathbf{a}$ from $D \cup \mathcal{M}$ such that $\langle \mathcal{T}, \mathcal{A}_{\mathcal{M},D} \rangle \models q(\mathbf{a})$

Handling mappings:

- apply mappings to generate ABox, proceed as usual
- virtual ABox: combine query rewriting with an extra unfolding step to get rewriting over DB relations

EXAMPLE: MAPPINGS

Suppose course data is stored in the two database tables:

$\texttt{UndergradCourses}$[CourseID, Year, Lecturer, Room, Timeslot]

$\texttt{GradCourses}$[CourseID, Lecturer, Room, Timeslot]

and employee data is stored in the table

$\texttt{Employee}$[EmpID,Name,Position,Dept]

The mapping could contain statements like: (initial $\forall$ omitted)

$$\exists y,r,t\, \texttt{UndergradCourses}(\mathsf{c}, y, \mathsf{l}, r, t) \rightarrow \mathsf{teaches}(\mathsf{l}, \mathsf{c})$$

$$\exists y,r,t\, \texttt{GradCourses}(\mathsf{c}, \mathsf{l}, r, t) \rightarrow \mathsf{teaches}(\mathsf{l}, \mathsf{c})$$

$$\exists n,d\, \texttt{Employee}(\mathsf{x}, n, Professor, d) \rightarrow \mathsf{Prof}(\mathsf{x})$$

to populate the ontology predicates teaches and Prof

EXAMPLE: MAPPING UNFOLDING

Unfolding of query q (over ontology vocabulary) w.r.t. mapping:

- find ways of unifying atoms in q with head atoms in mapping rules
- replace atoms in q by the bodies of (unified) matching rules

Unfolding of query q (over ontology vocabulary) w.r.t. mapping:

- find ways of unifying atoms in q with head atoms in mapping rules
- replace atoms in q by the bodies of (unified) matching rules

For $q(x, y) = \mathrm{Prof}(x) \wedge \mathrm{teaches}(x, y)$, unfolding yields the following:

$$\exists n,d,z,r,t\ \mathtt{Employee}(\mathbf{x}, n, \mathit{Professor}, d) \wedge \mathtt{UndergradCourses}(\mathbf{y}, z, \mathbf{x}, r, t)$$
$$\vee\ \exists n,d,r,t\ \mathtt{Employee}(\mathbf{x}, n, \mathit{Professor}, d) \wedge \mathtt{GradCourses}(\mathbf{y}, \mathbf{x}, r, t)$$

Observe result uses only DB predicates, can be converted to SQL

Unfolding of query q (over ontology vocabulary) w.r.t. mapping:

- find ways of unifying atoms in q with head atoms in mapping rules
- replace atoms in q by the bodies of (unified) matching rules

For $q(x, y) = \mathsf{Prof}(x) \wedge \mathsf{teaches}(x, y)$, unfolding yields the following:

$$\exists n, d, z, r, t\, \mathtt{Employee}(\mathbf{x}, n, \mathit{Professor}, d) \wedge \mathtt{UndergradCourses}(\mathbf{y}, z, \mathbf{x}, r, t)$$
$$\vee\ \exists n, d, r, t\, \mathtt{Employee}(\mathbf{x}, n, \mathit{Professor}, d) \wedge \mathtt{GradCourses}(\mathbf{y}, \mathbf{x}, r, t)$$

Observe **result uses only DB predicates**, can be converted to SQL

Opportunities for optimization: simplify rewriting by exploiting the fact that only needs to work for ABoxes induced by the mapping

OMQA RESEARCH

RESEARCH TOPICS IN OMQA

Ontology-mediated query answering:

- new paradigm for intelligent information systems
- offers many advantages, but also computational challenges
- active area with lots left to explore!

Ontology-mediated query answering:

- new paradigm for intelligent information systems
- offers many advantages, but also computational challenges
- active area with lots left to explore!

Efficient OMQA algorithms:

- optimized rewriting algorithms: compact rewritings, exploit mapping structure, cost-based rewriting selection
- tackling more expressive DLs: identify easier cases (existence of rewritings), use upper + lower approximations

Ontology-mediated query answering:

- new paradigm for intelligent information systems
- offers many advantages, but also computational challenges
- active area with lots left to explore!

Efficient OMQA algorithms:

- optimized rewriting algorithms: compact rewritings, exploit mapping structure, cost-based rewriting selection
- tackling more expressive DLs: identify easier cases (existence of rewritings), use upper + lower approximations

Support for building and maintaining OMQA systems

- ontology + mapping bootstrapping, module extraction, debugging, ontology evolution and versioning
- inspired new reasoning tasks: query inseparability, query emptiness, justification finding, logical difference, ...

Improving the usability of OMQA systems

- interfaces and support for query formulation
- explaining query (non-)answers

Improving the usability of OMQA systems

- interfaces and support for query formulation
- explaining query (non-)answers

Broadening the scope:

- new data formats: graph data, key-value stores, temporal data
- further query languages: regular path queries, streaming queries

Improving the usability of OMQA systems

- interfaces and support for query formulation
- explaining query (non-)answers

Broadening the scope:

- new data formats: graph data, key-value stores, temporal data
- further query languages: regular path queries, streaming queries

Beyond classical OMQA:

- inconsistency-tolerant query answering
- probabilistic query answering
- privacy-aware query answering

REFERENCES

Recent textbook on DLs:

An Introduction to Description Logic. By F. Baader, I. Horrocks, C. Lutz, and U. Sattler. Cambridge University Press (2017).

Lecture notes on OMQA (**look here for further refs!**):

M. Bienvenu and M. Ortiz. **Ontology-mediated query answering with data-tractable description logics**. Lecture Notes of Reasoning Web. Springer LNCS, 2015.

Short recent survey, focusing on linking to DBs:

G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev. **Ontology-based data access: A survey**. Proc. of IJCAI, 2018.

Original PerfectRef algorithm presented here:

D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati: **Tractable reasoning and efficient query answering in description logics: The DL-Lite family**. Journal of Automated Reasoning (JAR) 39(3), 385–429, 2007.

"Semantic" tree witness rewriting is described here:

S. Kikot, R. Kontchakov, M. Zakharyaschev: **Conjunctive query answering with OWL 2 QL**. Proc. of KR, 2012.

(far from an exhaustive list, refer to RW chapter for further refs)

The following paper presents the landscape for pure rewritings:

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

Optimal NDL-rewritings presented here:

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**. Proc. of PODS, 2017.

Polynomial impure rewritings can be found here:

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.

Combined approach for $\mathcal{EL}$:

C. Lutz, D. Toman, F. Wolter: **Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system**. Proc. of IJCAI, 2009.

Datalog rewriting approach that works for Horn-$\mathcal{SHIQ}$:

T. Eiter, M. Ortiz, M. Simkus, T. Tran, G. Xiao: **Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules**. Proc. of AAAI, 2012.

(just two examples of algorithms, see RW chapter for more refs)

M. Bienvenu, C. Lutz, and F. Wolter: First Order-Rewritability of Atomic Queries in Horn Description Logics. Proc. of IJCAI, 2013.

P. Hansen, C. Lutz, I. Seylan, and F. Wolter: Efficient Query Rewriting in the Description Logic EL and Beyond. Proc. of IJCAI, 2015.

M. Bienvenu, P. Hansen, C. Lutz, and F. Wolter: First Order-Rewritability and Containment of Conjunctive Queries in Horn Description Logics. Proc. of IJCAI, 2016.

P. Hansen and C. Lutz: Computing FO-Rewritings in $\mathcal{EL}$ in Practice: from Atomic to Conjunctive Queries. Proc. of ISWC, 2017.