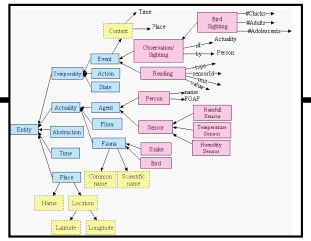
A BRIEF INTRODUCTION TO ONTOLOGY-MEDIATED QUERY ANSWERING

Meghyn Bienvenu (LaBRI - *CNRS & Université de Bordeaux*)

ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)

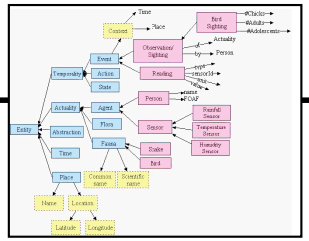
incomplete database
(ground facts)



ontology
(logical theory)

user query

ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)



patient data

"Melanie has listeriosis"
"Paul has Lyme disease"

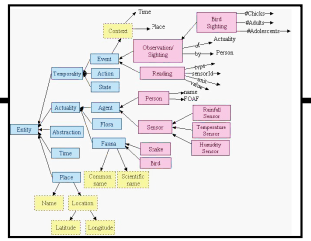
medical knowledge

"Listeriosis & Lyme disease are bacterial infections"

user query

"Find all patients with bacterial infections"

ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)



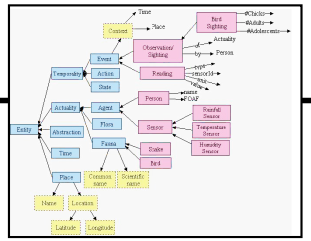
patient data
"Melanie has listeriosis"
"Paul has Lyme disease"

medical knowledge
"Listeriosis & Lyme disease are bacterial infections"

user query
"Find all patients with bacterial infections"

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

ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)



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

medical knowledge
"Listeriosis & Lyme disease are bacterial infections"

user query
"Find all patients with bacterial infections"

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

The ontology (logical theory) specifies:

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

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

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**

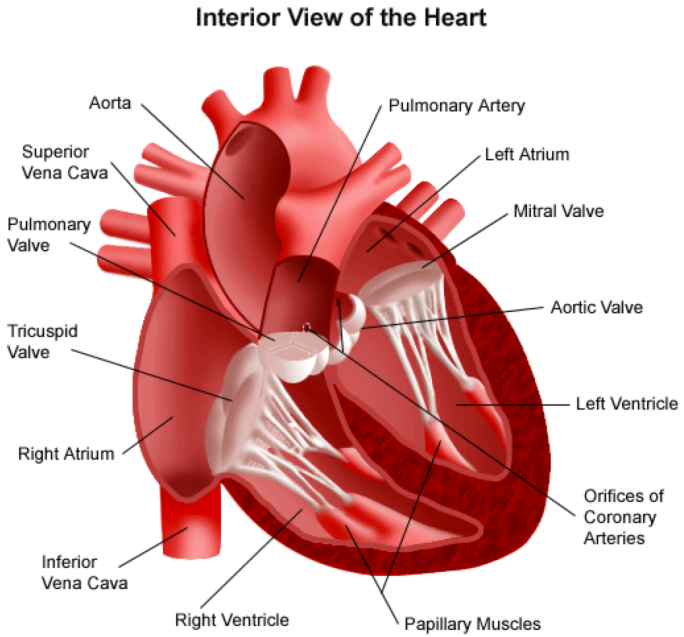
- ontology can be used to **enrich the data vocabulary**, making it easier for users to formulate their queries
- especially useful when **integrating multiple data sources**

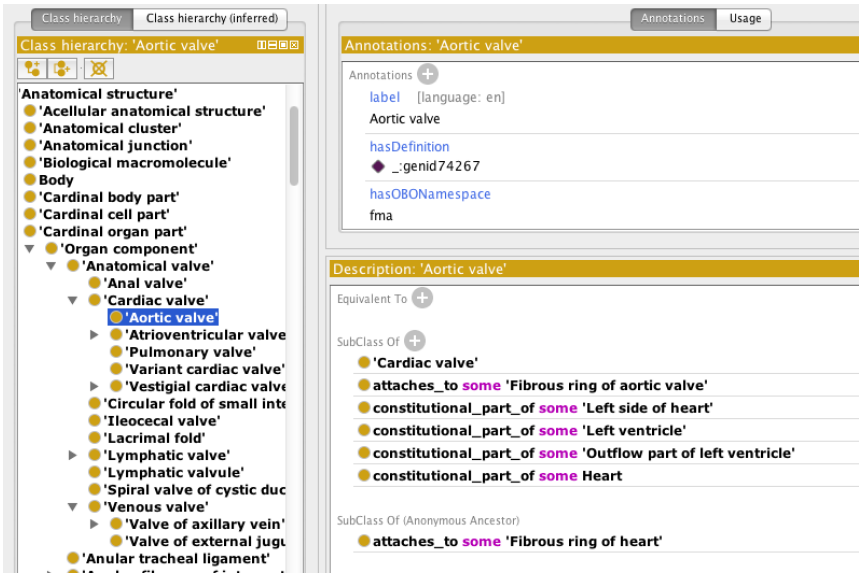
To support **automated reasoning**

- 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

APPLICATIONS OF OMQA: MEDICINE

General medical ontologies: SNOMED CT, GALEN
Specialized ontologies: FMA (anatomy), NCI (cancer), ...



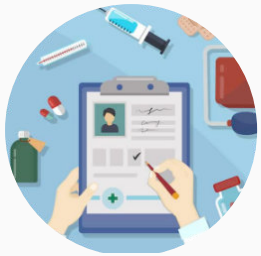


Querying & exchanging medical records (find patients for medical trials)

Supports analysis of health data for medical research

Large-scale comprehensive medical ontology

- more than 350,000 concepts on all aspects of clinical healthcare:
  - symptoms, diagnosis, medical procedures, body structures, organisms, substances, pharmaceutical products, etc.



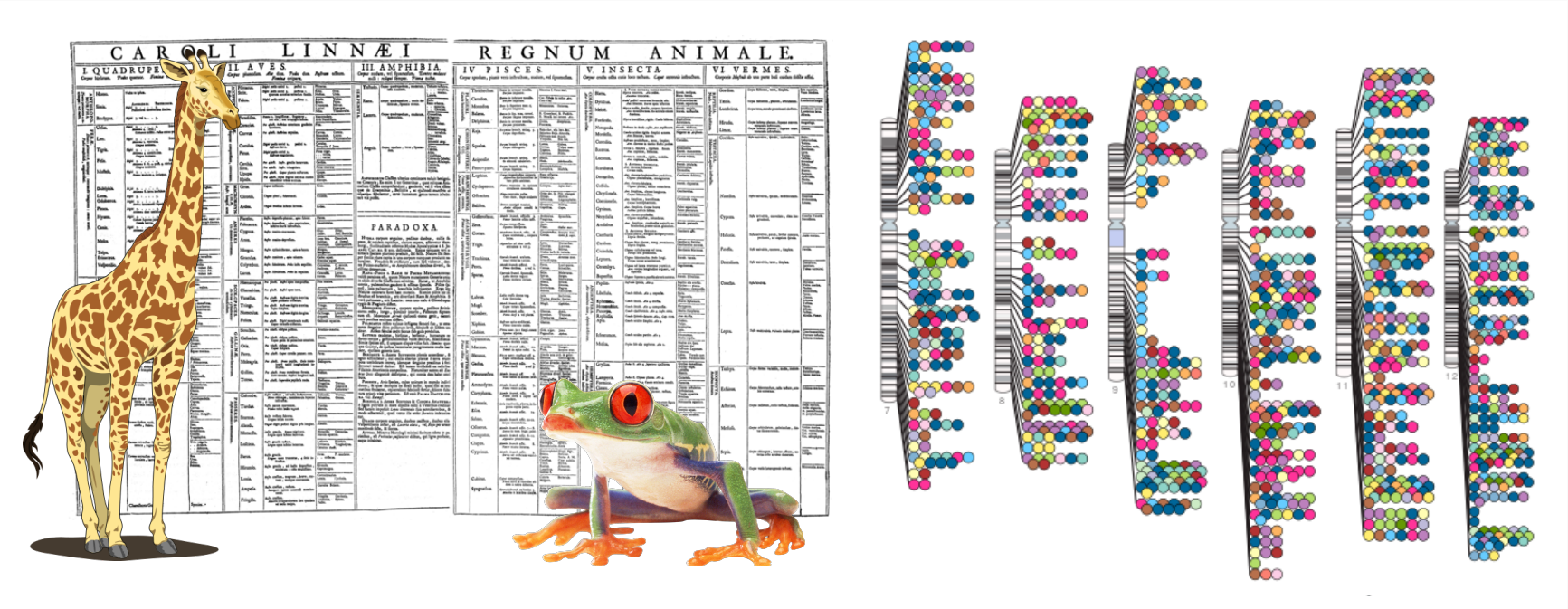


Widely adopted standard for healthcare terminology

- in use in $>$ 80 countries (including U.S., Canada, UK)
- utilized in health IT (e.g. IBM Watson Health, Babylon Health)

APPLICATIONS OF OMQA: LIFE SCIENCES

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

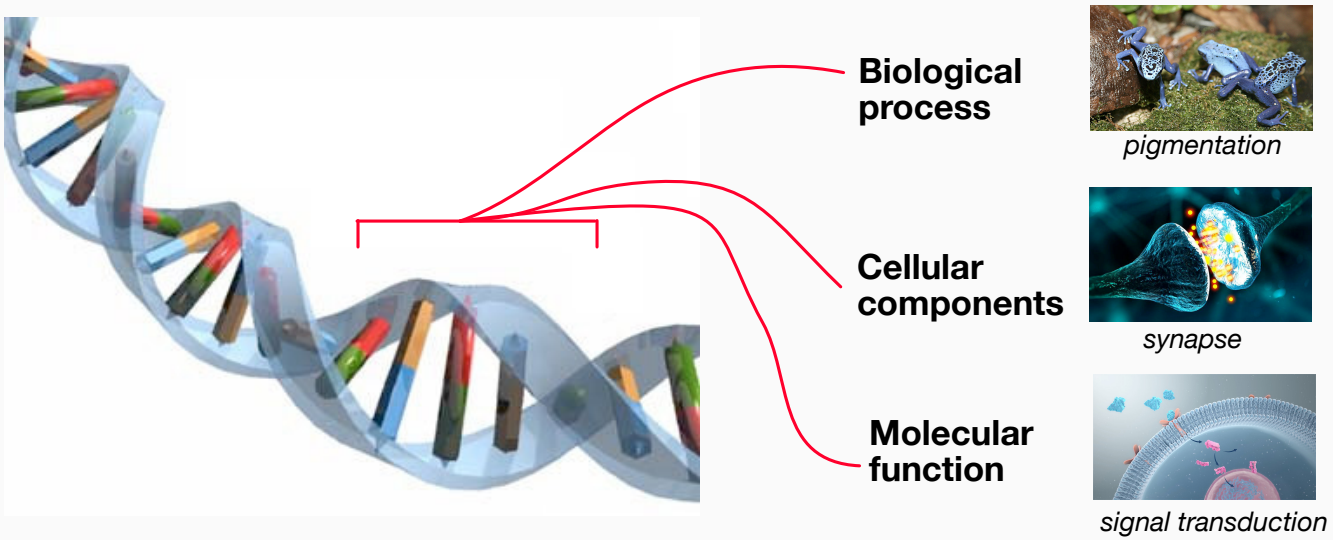


Help scientists share, query, & visualize experimental data

ZOOM ON: GENE ONTOLOGY

Aim: rigorous shared vocabulary to describe the roles of genes across different organisms

Annotations: evidence-based statements relating specific gene product to specific ontology terms



Very successful endeavour:

- > 100K published scientific articles with keyword “Gene Ontology”
- > 700K experimentally-supported annotations

APPLICATIONS OF OMQA: ENTREPRISE INFORMATION SYSTEMS

Companies and organizations have lots of data

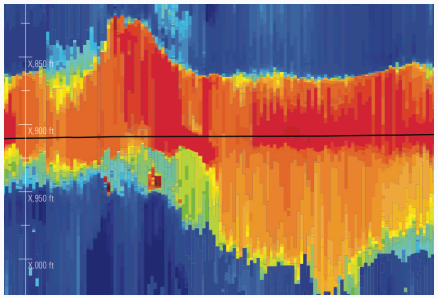
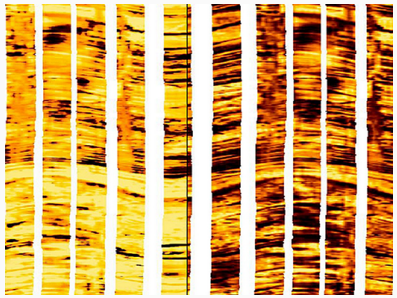
- need easy and flexible access to support decision-making

Example industrial projects:

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

ZOOM ON: STATOIL USECASE

Goal: aid geologists in gathering information for analysis


Difficulties:

- relevant data stored across $>$ 3000 database tables
  - geologist question = complex query (thousands of terms, 50-200 joins)
- must ask IT staff, may take several days to get answer

Solution:

- ontology provides familiar vocabulary for query formulation
- mappings link database tables to ontology terms
- use reasoning to automatically transform ontology query into executable database query

OUR FOCUS: DESCRIPTION LOGIC ONTOLOGIES

Ontologies typically described using logic-based formalisms

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)

OUR FOCUS: DESCRIPTION LOGIC ONTOLOGIES

Ontologies typically described using logic-based formalisms

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

OUR FOCUS: DESCRIPTION LOGIC ONTOLOGIES

Ontologies typically described using logic-based formalisms

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)

OUR FOCUS: DESCRIPTION LOGIC ONTOLOGIES

Ontologies typically described using logic-based formalisms

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

Introduction to DLs and OMQA

OMQA with Data-Tractable DLs

- DL-Lite ($\sim$ OWL 2 QL)
- $\mathcal{EL}$ ($\sim$ OWL 2 EL)

Research Questions in OMQA

INTRODUCTION TO DLS AND OMQA

Building blocks:

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

Constructors to build complex descriptions $\sqcup, \sqcap, \neg, \forall, \exists, \ldots$

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

DESCRIPTION LOGIC ONTOLOGIES

Building blocks:

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

Constructors to build complex descriptions — $\sqcup, \sqcap, \neg, \forall, \exists, \ldots$

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

Ontology (aka TBox) = set of axioms

- concept inclusions

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

- role inclusions

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

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

Expressive $\mathcal{ALC}$ $\rightsquigarrow$ OWL (2)

- only concept inclusions, with concepts formed as follows:

$$\top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$$

- reasoning provably intractable (EXPTIME)

Expressive $\mathcal{ALC}$ ⇝ OWL (2)

- only concept inclusions, with concepts formed as follows:

  $$\top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$$

- reasoning provably intractable (EXPTIME)

Lightweight $\mathcal{EL}$ ⇝ OWL 2 EL

- concept constructors: $\top \mid A \mid C \sqcap D \mid \exists R.C$
- suitable for large biomedical ontologies

Expressive $\mathcal{ALC}$ ⇝ OWL (2)

- only concept inclusions, with concepts formed as follows:

  $$\top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$$

- reasoning provably intractable (EXPTIME)

Lightweight $\mathcal{EL}$ ⇝ OWL 2 EL

- concept constructors: $\top \mid A \mid C \sqcap D \mid \exists R.C$
- suitable for large biomedical ontologies

Lightweight DL-Lite ⇝ OWL 2 QL

- concept inclusions $B_1 \sqsubseteq (\neg) B_2$ with $B_i = A \mid \exists R \mid \exists R^-$
- role inclusions $S_1 \sqsubseteq (\neg) S_2$ with $S_i := R \mid R^-$
- designed for data-intensive applications

EXAMPLES OF ONTOLOGY AXIOMS

Professors and lecturers are disjoint classes of faculty

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

Every Master's student is supervised by some faculty member

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

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

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

FO translation:

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

Focus on two common types of queries for DL ontologies:

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

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

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

conjunctions of atoms, some variables can be existentially quantified

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

(find all faculty members that teach something)

Focus on two common types of queries for DL ontologies:

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

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

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

conjunctions of atoms, some variables can be existentially quantified

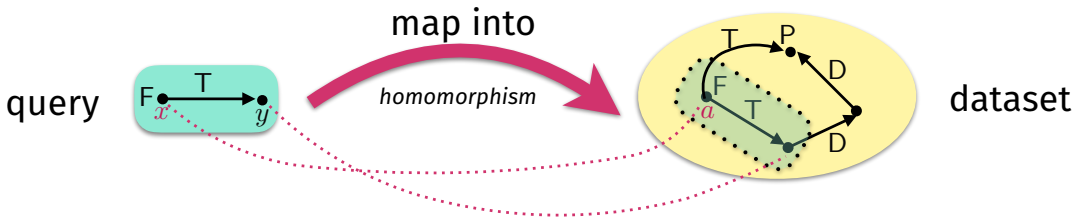
$$\exists y.\, \text{Faculty}(x) \land \text{teaches}(x, y)$$

(find all faculty members that teach something)

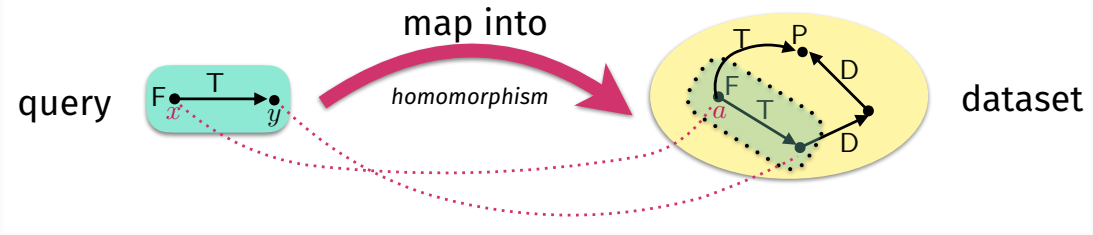
Assume data is set of unary and binary facts in ontology vocabulary

- different format / vocabulary $\Rightarrow$ use mappings

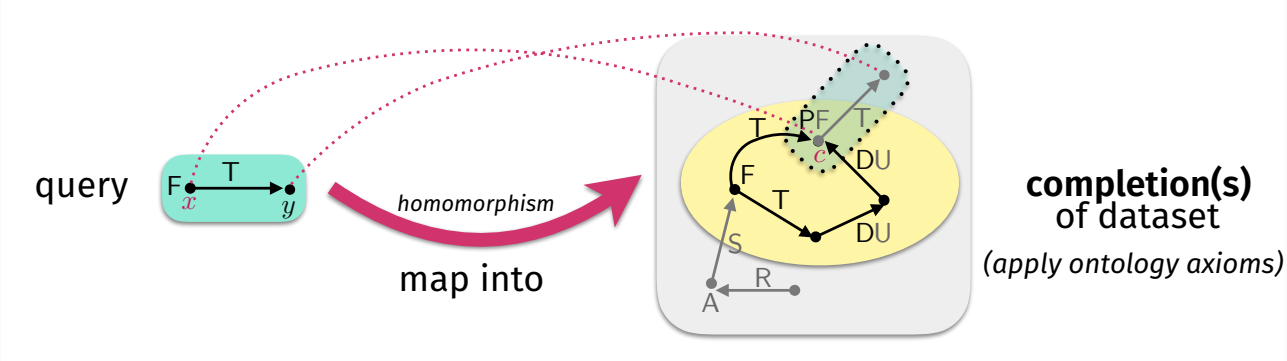
Answering CQs in database setting



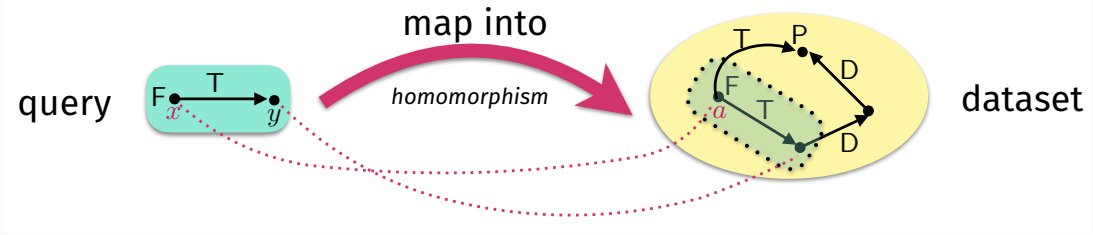
Answering CQs in database setting



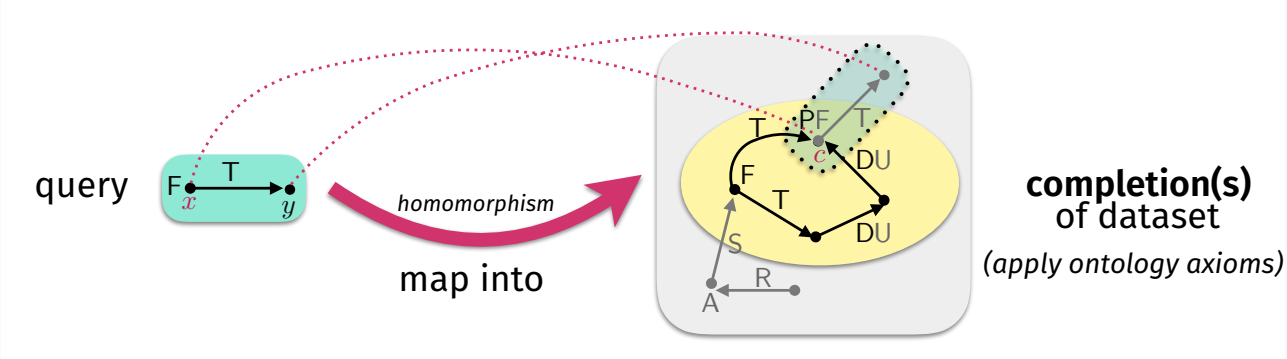
Answering CQs in the presence of an ontology



Answering CQs in database setting

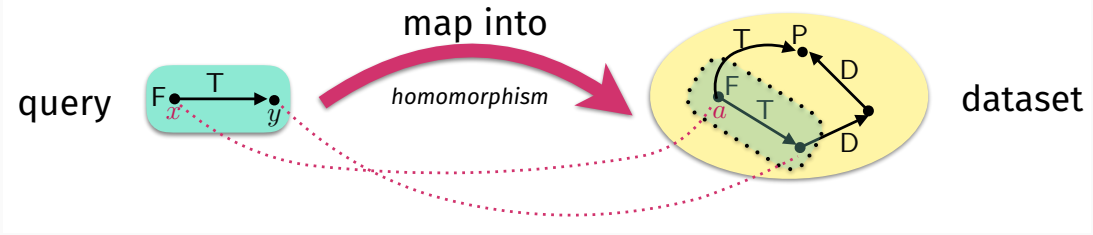


Answering CQs in the presence of an ontology

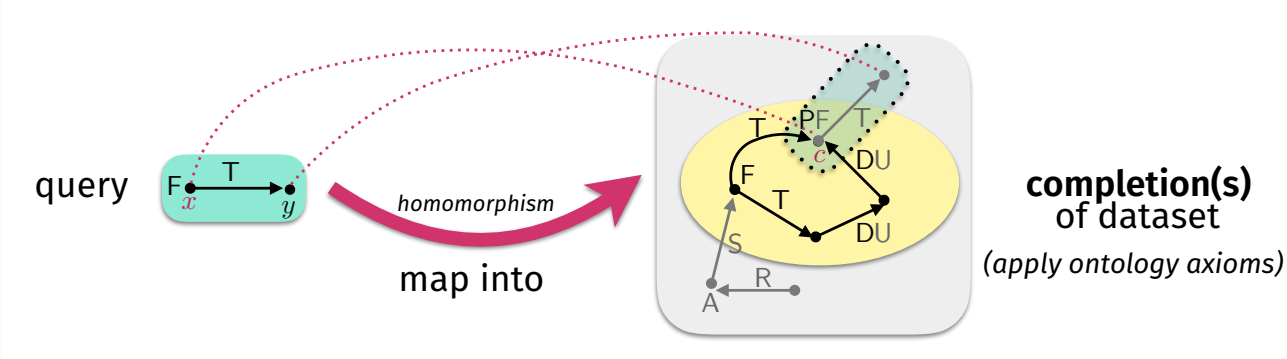


Certain answers: tuples $\vec{a}$ of individuals such that $\mathcal{O}, \mathcal{D} \models q(\vec{a})$

Answering CQs in database setting



Answering CQs in the presence of an ontology



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

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

EXAMPLE: OMQA WITH DL-LITE ONTOLOGIES

Ontology, expressed in **DL-Lite**: (lightweight DL designed for OMQA)

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

Dataset:

$$\mathcal{D}_1 = \{\mathsf{Prof}(\mathsf{anna}), \mathsf{Fellow}(\mathsf{tom}), \mathsf{teaches}(\mathsf{tom}, \mathsf{cs101})\}$$

Query: $q_1(x) = \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$

EXAMPLE: OMQA WITH DL-LITE ONTOLOGIES

Ontology, expressed in **DL-Lite**: (lightweight DL designed for OMQA)

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

Dataset:

$$\mathcal{D}_1 = \{\mathsf{Prof}(\mathsf{anna}), \mathsf{Fellow}(\mathsf{tom}), \mathsf{teaches}(\mathsf{tom}, \mathsf{cs101})\}$$

Query: $q_1(x) = \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$

Get the following answers:

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

COMPLEXITY OF OMQA

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** $q \in \mathcal{Q}$, a **data instance** $\mathcal{D}$, an $\mathcal{L}$**-ontology** $\mathcal{O}$, and a **tuple** $\vec{a} \in \mathsf{Ind}(\mathcal{A})^n$

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

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** $q \in \mathcal{Q}$, a **data instance** $\mathcal{D}$, an $\mathcal{L}$**-ontology** $\mathcal{O}$, and a **tuple** $\vec{a} \in \mathsf{Ind}(\mathcal{A})^n$

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

Combined complexity: in terms of **size of whole input**

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

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

$$\textbf{data complexity} \leq \textbf{combined complexity}$$

INTRACTABILITY OF OMQA IN ALC AND ALCI

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

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

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

The situation is even worse for $\mathcal{ALCI}$ (= $\mathcal{ALC}$ + inverse roles):

- 2EXPTIME-complete(!) in combined complexity
- coNP-complete in data complexity

OMQA WITH DATA-TRACTABLE DLS

DATA-TRACTABLE 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 (like chase in DBs)

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

DL-Lite$_R$ axioms:

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

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

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

DL-Lite$_R$ axioms:

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

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

Example axioms:

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

QUERY REWRITING

Idea: reduce OMQA to database query evaluation

- rewriting step: ontology $\mathcal{O}$ + query $q \rightsquigarrow$ first-order (SQL) query q'
- evaluation step: evaluate query q' using relational DB system

Advantage: harness efficiency of relational database systems

QUERY REWRITING

Idea: reduce OMQA to database query evaluation

- rewriting step: ontology $\mathcal{O}$ + 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. $\mathcal{O}$ iff for every dataset $\mathcal{D}$:

$$\mathcal{O}, \mathcal{D} \models q(\vec{a}) \quad \Leftrightarrow \quad DB(\mathcal{D}) \models q'(\vec{a})$$

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

QUERY REWRITING

Idea: reduce OMQA to database query evaluation

- rewriting step: ontology $\mathcal{O}$ + 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. $\mathcal{O}$ iff for every dataset $\mathcal{D}$:

$$\mathcal{O}, \mathcal{D} \models q(\vec{a}) \quad \Leftrightarrow \quad DB(\mathcal{D}) \models q'(\vec{a})$$

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

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

EXAMPLE: QUERY REWRITING IN DL-LITE

Reconsider the DL-Lite ontology $\mathcal{O}$:

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

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

EXAMPLE: QUERY REWRITING IN DL-LITE

Reconsider the DL-Lite ontology $\mathcal{O}$:

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

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. $\mathcal{O}$:

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

EXAMPLE: QUERY REWRITING IN DL-LITE

Reconsider the DL-Lite ontology $\mathcal{O}$:

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

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. $\mathcal{O}$:

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

Evaluating the rewritten query over the earlier dataset

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

produces the two certain answers: anna and tom

COMPLEXITY OF OMQA IN DL-LITE

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

COMPLEXITY OF OMQA IN DL-LITE

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

Combined complexity:

- 'guess' a disjunct of the rewriting and how to map it into data
- CQ answering is NP-complete (same as for DBs)
- IQ answering is NLogSpace-complete (NLogSpace $\subseteq$ P)

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

- Concepts: $C := \top \mid A \mid C \sqcap C \mid \exists r.C$
- Only concept inclusions in the ontology

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

- Concepts: $C := \top \mid A \mid C \sqcap C \mid \exists r.C$
- Only concept inclusions in the ontology

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

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

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

- Concepts: $C := \top \mid A \mid C \sqcap C \mid \exists r.C$
- Only concept inclusions in the ontology

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

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

We present a saturation-based approach.

Convenient to assume $\mathcal{EL}$ ontologies 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 $\top$)

Deriving new ontology axioms

$$\frac{A \sqsubseteq B_i \ (1 \leq i \leq n) \quad B_1 \sqcap \ldots \sqcap B_n \sqsubseteq D}{A \sqsubseteq D} \ \text{O1} \qquad \frac{A \sqsubseteq B \quad B \sqsubseteq \exists r.D}{A \sqsubseteq \exists r.D} \ \text{O2}$$

$$\frac{A \sqsubseteq \exists r.B \quad B \sqsubseteq D \quad \exists r.D \sqsubseteq E}{A \sqsubseteq E} \ \text{O3}$$

Deriving new facts

$$\frac{A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \quad A_i(a) \ (1 \leq i \leq n)}{B(a)} \ \text{D1} \qquad \frac{\exists r.B \sqsubseteq A \quad r(a,b) \quad B(b)}{A(a)} \ \text{D2}$$

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

EXAMPLE: SATURATION IN EL

$$\mathsf{PenneArrabiata} \sqsubseteq \exists \mathsf{hasIngred}.\mathsf{ArrabiataSauce} \quad (1)$$

$$\mathsf{PenneArrabiata} \sqsubseteq \mathsf{PastaDish} \quad (2)$$

$$\mathsf{PastaDish} \sqsubseteq \mathsf{Dish} \quad (3)$$

$$\mathsf{PastaDish} \sqsubseteq \exists \mathsf{hasIngred}.\mathsf{Pasta} \quad (4)$$

$$\mathsf{ArrabiataSauce} \sqsubseteq \exists \mathsf{hasIngred}.\mathsf{Peperoncino} \quad (5)$$

$$\mathsf{Peperoncino} \sqsubseteq \mathsf{Spicy} \quad (6)$$

$$\exists \mathsf{hasIngred}.\mathsf{Spicy} \sqsubseteq \mathsf{Spicy} \quad (7)$$

$$\mathsf{Spicy} \sqcap \mathsf{Dish} \sqsubseteq \mathsf{SpicyDish} \quad (8)$$

$$\mathsf{PenneArrabiata}(p). \quad (9)$$

EXAMPLE: SATURATION IN EL

$$\text{PenneArrabiata} \sqsubseteq \exists \text{hasIngred}.\text{ArrabiataSauce} \quad (1)$$

$$\text{PenneArrabiata} \sqsubseteq \text{PastaDish} \quad (2)$$

$$\text{PastaDish} \sqsubseteq \text{Dish} \quad (3)$$

$$\text{PastaDish} \sqsubseteq \exists \text{hasIngred}.\text{Pasta} \quad (4)$$

$$\text{ArrabiataSauce} \sqsubseteq \exists \text{hasIngred}.\text{Peperoncino} \quad (5)$$

$$\text{Peperoncino} \sqsubseteq \text{Spicy} \quad (6)$$

$$\exists \text{hasIngred}.\text{Spicy} \sqsubseteq \text{Spicy} \quad (7)$$

$$\text{Spicy} \sqcap \text{Dish} \sqsubseteq \text{SpicyDish} \quad (8)$$

$$\text{PenneArrabiata}(p). \quad (9)$$

$$\text{ArrabSauce} \sqsubseteq \text{Spicy} \quad \text{O3} : (5), (6), (7) \quad (10)$$

$$\text{PenneArrab} \sqsubseteq \text{Spicy} \quad \text{O3} : (1), (10), (7) \quad (11)$$

$$\text{PenneArrab} \sqsubseteq \text{Dish} \quad \text{O1} : (2), (3) \quad (12)$$

$$\text{PenneArrab} \sqsubseteq \exists \text{hasIngred}.\text{Pasta} \quad \text{O2} : (2), (4) \quad (13)$$

$$\text{PenneArrab} \sqsubseteq \text{SpicyDish} \quad \text{O1} : (11), (12), (8) \quad (14)$$

$$\text{Spicy}(\mathbf{p}) \quad \text{D1} : (11), (9) \quad (15)$$

$$\text{Dish}(\mathbf{p}) \quad \text{D1} : (12), (9) \quad (16)$$

$$\text{SpicyDish}(\mathbf{p}) \quad \text{D1} : (16), (15) \quad (17)$$

Saturation approach is sound: everything derived is entailed

Saturation approach is sound: everything derived is entailed

Also complete for IQs, since for every fact α, we have:

$$\mathcal{O}, \mathcal{D} \models \alpha \qquad \text{iff} \qquad \alpha \in \mathsf{sat}(\mathcal{O}, \mathcal{D})$$

COMPLEXITY OF IQ ANSWERING IN EL

Saturation approach is sound: everything derived is entailed

Also complete for IQs, since for every fact α, we have:

$$\mathcal{O}, \mathcal{D} \models \alpha \qquad \text{iff} \qquad \alpha \in \mathsf{sat}(\mathcal{O}, \mathcal{D})$$

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

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

Different techniques required to handle conjunctive queries

Combined approach (another way to exploit DBs):

- saturate ABox using the TBox axioms
  - introduce new individuals to witness existentials on LHS ($A \sqsubseteq \exists r.B$)

Different techniques required to handle conjunctive queries

Combined approach (another way to exploit DBs):

- saturate ABox using the TBox axioms
  - introduce new individuals to witness existentials on LHS ($A \sqsubseteq \exists r.B$)
  - to ensure finite: reuse individuals as witnesses
- 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

Different techniques required to handle conjunctive queries

Combined approach (another way to exploit DBs):

- saturate ABox using the TBox axioms
  - introduce new individuals to witness existentials on LHS ($A \sqsubseteq \exists r.B$)
  - to ensure finite: reuse individuals as witnesses
- 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

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

- P-complete in data complexity (scale polynomially in $|\mathcal{A}|$)
- NP-complete in combined complexity

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!

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!

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

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!

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, ...

RESEARCH TOPICS IN OMQA

Improving the usability of OMQA systems

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

RESEARCH TOPICS IN OMQA

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

RESEARCH TOPICS IN OMQA

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

Questions ?