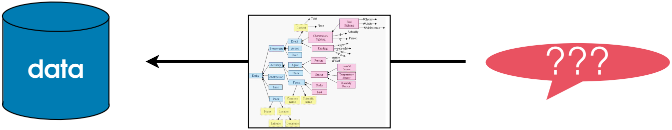
ONTOLOGY-MEDIATED QUERY ANSWERING WITH DESCRIPTION LOGICS

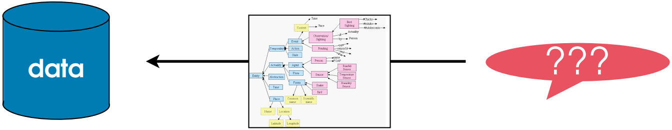
Meghyn Bienvenu (LIRMM: *CNRS & Université de Montpellier*)



incomplete database
(ground facts)

ontology
(logical theory)

user query



patient data

"Melanie has listeriosis"
"Paul has Lyme disease"

medical knowledge

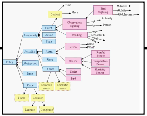
"Listeriosis & Lyme disease are bacterial infections"

user query

"Find all patients with bacterial infections"

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**

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

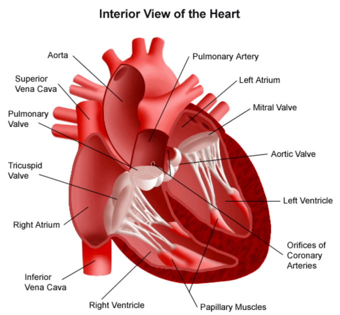
To present an intuitive and unified view of data sources

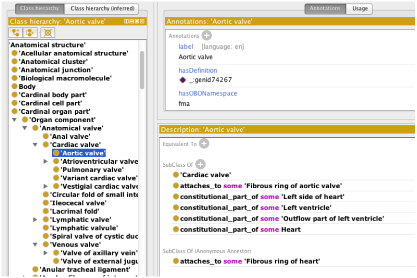
- ontology can be used to enrich the data vocabulary, making it easier for users to formulate their queries
- especially useful when integrating multiple data sources

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

General medical ontologies: SNOMED CT ($\sim$ 400,000 terms!), GALEN
Specialized ontologies: FMA (anatomy), NCI (cancer), ...



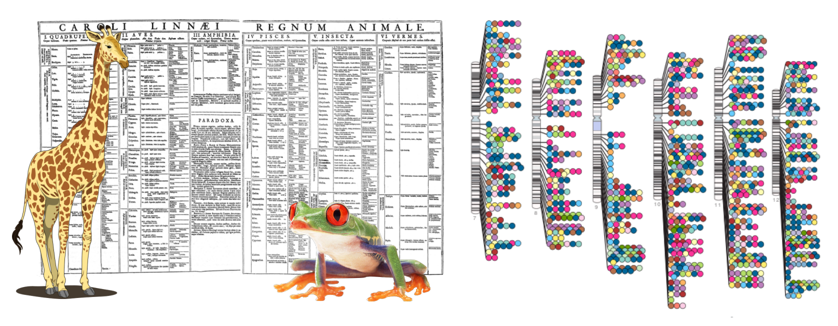


Querying & exchanging medical records (find patients for medical trials)

- 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

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 formulated using description logics (DLs):

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

Ontologies formulated using description logics (DLs):

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

Of particular interest: Horn description logics

- DL-Lite$_R$, $\mathcal{EL}$, $\mathcal{ELHI}$, Horn-$\mathcal{SHIQ}$, ...
- good computational properties, well suited for OMQA
- still expressive enough for interesting applications
- basis for OWL 2 QL and OWL 2 EL profiles

Basics of DLs

Introduction to OMQA

OMQA with Lightweight DLs

Research Trends in OMQA

BASICS OF DLS

Building blocks of DLs:

- concept names (unary predicates, classes)

  IceCream, Pizza, Meat, SpicyDish, Dish, Menu, Restaurant, ...

- role names (binary predicates, properties)

  hasIngred, hasCourse, hasDessert, serves, ...

- individual names (constants)

  menu32, pastadish17, d3, rest156, r12, ...

  (specific menus, dishes, restaurants ...)

N_C / N_R / N_I: set of all concept / role / individual names

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

ABox contains **facts about specific individuals**

- finite set of **concept assertions** $A(a)$ and **role assertions** $r(a, b)$
- IceCream(d_2): dish d_2 *is of type* IceCream
- hasDessert(m, d_2): *menu* m *is connected via* hasDessert *to dish* d_2

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

ABox contains **facts about specific individuals**

- finite set of **concept assertions** $A(a)$ and **role assertions** $r(a, b)$
- IceCream(d_2): dish d_2 *is of type* IceCream
- hasDessert(m, d_2): *menu* m *is connected via* hasDessert *to dish* d_2

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

- finite set of **axioms** (details on syntax to follow)
- IceCream *is a subclass of* Dessert
- hasCourse *connects* Menus *to* Dishes
- *every* Menu *is connected to at least one dish via* hasCourse

Can build complex concepts and roles using constructors:

- conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$)

$$\text{Dessert} \sqcap \neg\text{IceCream} \qquad \text{Pizza} \sqcup \text{PastaDish}$$

Can build complex concepts and roles using constructors:

- conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$)

$$\mathsf{Dessert} \sqcap \neg\mathsf{IceCream} \qquad \mathsf{Pizza} \sqcup \mathsf{PastaDish}$$

- restricted forms of existential and universal quantification ($\exists$, $\forall$)

$$\exists\mathsf{contains}.\mathsf{Meat} \qquad \exists\mathsf{hasCourse}.\top \qquad \mathsf{Dish} \sqcap \forall\mathsf{contains}.\neg\mathsf{Meat}$$

($\top$ acts as a “wildcard”, denotes set of all things)

Can build complex concepts and roles using constructors:

- conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$)

  $\mathsf{Dessert} \sqcap \neg\mathsf{IceCream}$ $\quad$ $\mathsf{Pizza} \sqcup \mathsf{PastaDish}$

- restricted forms of existential and universal quantification ($\exists$, $\forall$)

  $\exists\mathsf{contains}.\mathsf{Meat}$ $\quad$ $\exists\mathsf{hasCourse}.\top$ $\quad$ $\mathsf{Dish} \sqcap \forall\mathsf{contains}.\neg\mathsf{Meat}$

  ($\top$ acts as a "wildcard", denotes set of all things)

- inverse ($^{-}$) and composition ($\cdot$) of roles

  $\mathsf{hasCourse}^{-}$ $\quad$ $\mathsf{contains} \cdot \mathsf{contains}$

Can build complex concepts and roles using constructors:

- conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$)

  $\mathsf{Dessert} \sqcap \neg\mathsf{IceCream} \qquad \mathsf{Pizza} \sqcup \mathsf{PastaDish}$

- restricted forms of existential and universal quantification ($\exists$, $\forall$)

  $\exists\mathsf{contains}.\mathsf{Meat} \qquad \exists\mathsf{hasCourse}.\top \qquad \mathsf{Dish} \sqcap \forall\mathsf{contains}.\neg\mathsf{Meat}$

  ($\top$ acts as a “wildcard”, denotes set of all things)

- inverse ($^-$) and composition ($\cdot$) of roles

  $\mathsf{hasCourse}^- \qquad \mathsf{contains} \cdot \mathsf{contains}$

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

TBOX AXIOMS

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

$\mathsf{IceCream} \sqsubseteq \mathsf{Dessert}$ $\quad$ $\mathsf{Menu} \sqsubseteq \exists \mathsf{hasCourse}.\top$ $\quad$ $\mathsf{Spicy} \sqcap \mathsf{Dish} \sqsubseteq \mathsf{SpicyDish}$

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

$\mathsf{hasIngred} \sqsubseteq \mathsf{contains}$ $\quad$ $\mathsf{ingredOf}^{-} \sqsubseteq \mathsf{hasIngred}$ $\quad$ $\mathsf{hasDessert} \sqsubseteq \mathsf{hasCourse}$

Note: type and syntax of axioms **depends on the particular DL**!

EXAMPLE DLS

"Standard" expressive description logic $\mathcal{ALC}$:

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

"Lightweight" description logic $\mathcal{EL}$

- Concept constructors: $C := \top \mid A \mid C \sqcap C \mid \exists r.C$
- TBox axioms: only concept inclusions

EXAMPLE DLS

"Standard" expressive description logic $\mathcal{ALC}$:

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

"Lightweight" description logic $\mathcal{EL}$

- Concept constructors: $C := \top \mid A \mid C \sqcap C \mid \exists r.C$
- TBox axioms: only concept inclusions

$\mathcal{ALCI}$ = extension of $\mathcal{ALC}$ with inverse roles (r^-)

$\mathcal{ELH}$ = $\mathcal{EL}$ + role inclusions ($r \sqsubseteq s$)

Interpretation $\mathcal{I}$ ("possible world")

- 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")

- 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 $R \sqsubseteq S \quad \Leftrightarrow \quad R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

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

Note: ABoxes are interpreted under the open-world assumption

INTRODUCTION TO OMQA

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

(aka **atomic queries**)

$\mathsf{A}(\mathsf{x})$ where $A \in \mathsf{N_C}$ — **concept** instance query

$\mathsf{r}(\mathsf{x}, \mathsf{y})$ where $r \in \mathsf{N_R}$ — **role** instance query

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

(aka atomic queries)

$\mathsf{A(x)}$ where $A \in \mathsf{N_C}$	concept instance query
$\mathsf{r(x, y)}$ where $r \in \mathsf{N_R}$	role instance query

To query for a complex concept C, take $A_C(x)$ for fresh $A_C \in \mathsf{N_C}$ and add $C \sqsubseteq A_C$ to the TBox

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

$\mathsf{A}(\mathsf{x})$ where $A \in \mathsf{N_C}$	**concept** instance query
$\mathsf{r}(\mathsf{x},\mathsf{y})$ where $r \in \mathsf{N_R}$	**role** instance query

To query for a **complex concept** C, take $A_C(x)$ for fresh $A_C \in \mathsf{N_C}$ and add $C \sqsubseteq A_C$ to the TBox

Remarks:

- Instance query answering is often called *instance checking*
- **Focus of OMQA until mid-2000s**

(UNIONS OF) CONJUNCTIVE QUERIES

IQs are quite restricted: no selections and joins as in DB queries

IQs are quite restricted: no selections and joins as in DB queries

Most work on OMQA adopts (unions of) conjunctive queries (CQs).

IQs are quite restricted: no selections and joins as in DB queries

Most work on OMQA adopts (unions of) conjunctive queries (CQs).

A conjunctive query (CQ) takes the form

$$q(\vec{x}) = \exists \vec{y}. P_1(\vec{t_1}) \wedge \cdots \wedge P_n(\vec{t_n})$$

where every P_i is a concept or role name
and $\vec{t_i}$ contains individual names and/or variables from $\vec{x} \cup \vec{y}$

IQs are quite restricted: no selections and joins as in DB queries

Most work on OMQA adopts (unions of) conjunctive queries (CQs).

A conjunctive query (CQ) takes the form

$$q(\vec{x}) = \exists \vec{y}. P_1(\vec{t_1}) \wedge \cdots \wedge P_n(\vec{t_n})$$

where every P_i is a concept or role name
and $\vec{t_i}$ contains individual names and/or variables from $\vec{x} \cup \vec{y}$

A union of CQs (UCQ) takes the form of a disjunction of CQs:

$$q_1(\vec{x}) \vee \cdots \vee q_n(\vec{x})$$

Query q of arity n + **interpretation** $\mathcal{I} \rightsquigarrow$ set of **answers** $\mathsf{ans}(q, \mathcal{I})$
(n-tuples of elements from $\mathcal{I}$)

Query q of arity n + **interpretation** $\mathcal{I} \rightsquigarrow$ set of **answers** $\mathsf{ans}(q, \mathcal{I})$
(n-tuples of elements from $\mathcal{I}$)

Each KB gives rise to multiple interpretations (its models)

- want tuple to be an answer w.r.t. all models of KB

Query q of arity n + **interpretation** $\mathcal{I} \rightsquigarrow$ set of **answers** $\mathrm{ans}(q, \mathcal{I})$
(n-tuples of elements from $\mathcal{I}$)

Each KB gives rise to multiple interpretations (its models)

- want tuple to be an answer w.r.t. all models of KB

Formally: Call a tuple $\vec{a} = (a_1, \ldots, a_n)$ of individuals from $\mathcal{A}$ a **certain answer to n-ary query q over DL KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$** if

$$(a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}) \in \mathrm{ans}(q, \mathcal{I}) \text{ for every model } \mathcal{I} \text{ of } \mathcal{K}$$

in which case we write $\mathcal{K} \models q(\vec{a})$

Query q of arity n + interpretation $\mathcal{I} \rightsquigarrow$ set of answers $\mathsf{ans}(q, \mathcal{I})$
(n-tuples of elements from $\mathcal{I}$)

Each KB gives rise to multiple interpretations (its models)

- want tuple to be an answer w.r.t. all models of KB

Formally: Call a tuple $\vec{a} = (a_1, \ldots, a_n)$ of individuals from $\mathcal{A}$ a certain answer to n-ary query q over DL KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if

$$(a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}) \in \mathsf{ans}(q, \mathcal{I}) \text{ for every model } \mathcal{I} \text{ of } \mathcal{K}$$

in which case we write $\mathcal{K} \models q(\vec{a})$

Ontology-mediated query answering (OMQA)
= computing certain answers to queries

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

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

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

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

Combined complexity: in terms of size of whole input

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

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

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

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:

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

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

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

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: no disjunction, existence of canonical model

OMQA WITH LIGHTWEIGHT DLS

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** $S_1 \sqsubseteq S_2$, $S_1 \sqsubseteq \neg S_2$

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

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

DL-Lite$_R$ TBoxes contain

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

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

Example TBox inclusions:

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

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

TBox:

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

Query:

$$q(x) = \mathsf{Dish}(x)$$

We compute a rewriting of $q(x)$ w.r.t. $\mathcal{T}$ step by step:

EXAMPLE OF QUERY REWRITING

TBox:

$$
\begin{aligned}
\mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\
\mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\
\mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\
\exists \mathsf{hasCourse}^{-} &\sqsubseteq \mathsf{Dish} \\
\mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\
\mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse}
\end{aligned}
$$

Query:

$$q(x) = \mathsf{Dish}(x)$$

We compute a rewriting of $q(x)$ w.r.t. $\mathcal{T}$ step by step:

$$q'(x) = \mathsf{Dish}(x)$$

EXAMPLE OF QUERY REWRITING

TBox:

$$
\begin{aligned}
\mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\
\mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\
\mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\
\exists \mathsf{hasCourse}^{-} &\sqsubseteq \mathsf{Dish} \\
\mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\
\mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse}
\end{aligned}
$$

Query:

$$q(x) = \mathsf{Dish}(x)$$

We compute a rewriting of $q(x)$ w.r.t. $\mathcal{T}$ step by step:

$$q'(x) = \mathsf{Dish}(x) \vee \mathsf{ItalDish}(x)$$

EXAMPLE OF QUERY REWRITING

TBox:

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

Query:

$$q(x) = \mathsf{Dish}(x)$$

We compute a rewriting of $q(x)$ w.r.t. $\mathcal{T}$ step by step:

$$q'(x) = \mathsf{Dish}(x) \vee \mathsf{ItalDish}(x) \vee \mathsf{VegDish}(x)$$

EXAMPLE OF QUERY REWRITING

TBox:

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

Query:

$$q(x) = \mathsf{Dish}(x)$$

We compute a rewriting of $q(x)$ w.r.t. $\mathcal{T}$ step by step:

$$q'(x) = \mathsf{Dish}(x) \vee \mathsf{ItalDish}(x) \vee \mathsf{VegDish}(x) \vee \exists y.\mathsf{hasCourse}(y, x)$$

TBox:

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

Query:

$$q(x) = \mathsf{Dish}(x)$$

We compute a rewriting of $q(x)$ w.r.t. $\mathcal{T}$ step by step:

$$\begin{aligned} q'(x) = {} & \mathsf{Dish}(x) \vee \mathsf{ItalDish}(x) \vee \mathsf{VegDish}(x) \vee \exists y.\mathsf{hasCourse}(y, x) \\ & \vee \exists y.\mathsf{hasMain}(y, x) \end{aligned}$$

TBox:

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

Query:

$$q(x) = \mathsf{Dish}(x)$$

We compute a rewriting of $q(x)$ w.r.t. $\mathcal{T}$ step by step:

$$\begin{aligned} q'(x) = \mathsf{Dish}(x) &\vee \mathsf{ItalDish}(x) \vee \mathsf{VegDish}(x) \vee \exists y.\mathsf{hasCourse}(y,x) \\ &\vee \exists y.\mathsf{hasMain}(y,x) \vee \exists y.\mathsf{hasDessert}(y,x) \end{aligned}$$

EXAMPLE OF QUERY REWRITING

TBox:

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

ABox:

$$\begin{aligned} &\mathsf{hasMain}(m, d_1) \\ &\mathsf{hasDessert}(m, d_2) \\ &\mathsf{VegDish}(d_3) \end{aligned}$$

$$\begin{aligned} q'(x) = \mathsf{Dish}(x) \lor \mathsf{ItalDish}(x) \lor \mathsf{VegDish}(x) \lor \exists y.\mathsf{hasCourse}(y, x) \\ \lor \exists y.\mathsf{hasMain}(y, x) \lor \exists y.\mathsf{hasDessert}(y, x) \end{aligned}$$

EXAMPLE OF QUERY REWRITING

TBox:

$$\begin{aligned} \text{ItalDish} &\sqsubseteq \text{Dish} \\ \text{VegDish} &\sqsubseteq \text{Dish} \\ \text{Dish} &\sqsubseteq \exists \text{hasIngred} \\ \exists \text{hasCourse}^- &\sqsubseteq \text{Dish} \\ \text{hasMain} &\sqsubseteq \text{hasCourse} \\ \text{hasDessert} &\sqsubseteq \text{hasCourse} \end{aligned}$$

ABox:

$$\begin{aligned} &\text{hasMain}(m, d_1) \\ &\text{hasDessert}(m, d_2) \\ &\text{VegDish}(d_3) \end{aligned}$$

$$q'(x) = \text{Dish}(x) \lor \text{ItalDish}(x) \lor \text{VegDish}(x) \lor \exists y.\text{hasCourse}(y, x)$$
$$\lor \exists y.\text{hasMain}(y, x) \lor \exists y.\text{hasDessert}(y, x)$$

Certain answers:
d_1, because of the disjunct $\exists y.\text{hasMain}(y, x)$
d_2, because of the disjunct $\exists y.\text{hasDessert}(y, x)$
d_3, because of the disjunct $\text{VegDish}(x)$

TBox:

$\{\ \exists\text{coordinates} \sqsubseteq \text{Prof} \quad \text{coordinates} \sqsubseteq \text{involved} \quad \text{100S} \sqsubseteq \text{IntroC}\}$

Query: $\text{Prof}(x) \wedge \text{involved}(x, y) \wedge \text{IntroC}(y)$

ANOTHER REWRITING EXAMPLE

TBox:

$\{\ \exists \mathsf{coordinates} \sqsubseteq \mathsf{Prof} \quad \mathsf{coordinates} \sqsubseteq \mathsf{involved} \quad \mathsf{100S} \sqsubseteq \mathsf{IntroC}\}$

Query: $\mathsf{Prof}(x) \wedge \mathsf{involved}(x, y) \wedge \mathsf{IntroC}(y)$

Obtain FO-rewriting by taking disjunction of q_0 and the CQs:

$$q_1 = \exists z\, \mathsf{coordinates}(x, z) \wedge \mathsf{involved}(x, y) \wedge \mathsf{IntroC}(y)$$
$$q_2 = \mathsf{coordinates}(x, y) \wedge \mathsf{IntroC}(y)$$
$$q_3 = \mathsf{Prof}(x) \wedge \mathsf{coordinates}(x, y) \wedge \mathsf{100S}(y)$$
$$q_4 = \exists z\, \mathsf{coordinates}(x, z) \wedge \mathsf{involved}(x, y) \wedge \mathsf{100S}(y)$$
$$q_5 = \mathsf{coordinates}(x, y) \wedge \mathsf{100S}(y)$$

Data complexity:

- rewriting takes constant time, yields FO query
- upper bound from FO query evaluation: AC_0 $\quad$ ($AC_0 \subseteq$ 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 ($AC_0 \subseteq$ 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 ABox
- CQ answering is NP-complete (same as for DBs)
- IQ answering is NLOGSPACE-complete (NLOGSPACE $\subseteq$ P)

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

Note: Same bounds hold for several other DL-Lite dialects

Next consider IQ answering in $\mathcal{EL}$.

Assume $\mathcal{EL}$ TBoxes given in normal form: axioms of the forms

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

$(A, A_i, B \in \mathsf{N_C})$

Next consider IQ answering in $\mathcal{EL}$.

Assume $\mathcal{EL}$ TBoxes given in normal form: axioms of the forms

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

$$(A, A_i, B \in \mathsf{N_C})$$

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 IQ answering in $\mathcal{EL}$.

Assume $\mathcal{EL}$ TBoxes given in normal form: axioms of the forms

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

$(A, A_i, B \in \mathsf{N_C})$

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 present a saturation-based approach.

SATURATION RULES FOR EL

TBox rules

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

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

ABox rules

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

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

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

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

$\text{ArrabSauce} \sqsubseteq \text{Spicy}$	T3 : (5), (6), (7)	(10)
$\text{PenneArrab} \sqsubseteq \text{Spicy}$	T3 : (1), (10), (7)	(11)
$\text{PenneArrab} \sqsubseteq \text{Dish}$	T1 : (2), (3)	(12)
$\text{PenneArrab} \sqsubseteq \exists\text{hasIngred.Pasta}$	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 instance checking:

Theorem Let $\mathcal{K}$ be an $\mathcal{EL}$ knowledge base, and let $\mathcal{K}'$ be the result of saturating $\mathcal{K}$. For every ABox assertion α, we have:

$$\mathcal{K} \models \alpha \qquad \text{iff} \qquad \alpha \in \mathcal{K}'$$

Saturation approach is sound: everything derived is entailed

Also complete for instance checking:

Theorem Let $\mathcal{K}$ be an $\mathcal{EL}$ knowledge base, and let $\mathcal{K}'$ be the result of saturating $\mathcal{K}$. For every ABox assertion α, we have:

$$\mathcal{K} \models \alpha \qquad \text{iff} \qquad \alpha \in \mathcal{K}'$$

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 instance checking**:

Theorem Let $\mathcal{K}$ be an $\mathcal{EL}$ knowledge base, and let $\mathcal{K}'$ be the result of saturating $\mathcal{K}$. For every ABox assertion α, we have:

$$\mathcal{K} \models \alpha \qquad \text{iff} \qquad \alpha \in \mathcal{K}'$$

Note: does not make all consequences explicit

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

Runs in **polynomial time** in $|\mathcal{K}|$. This is **optimal**:

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

CQ ANSWERING IN EL

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

- P-complete in data complexity (scale polynomially in $|\mathcal{A}|$)
  - can be shown e.g. by rewriting into Datalog
- NP-complete in combined complexity

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

- P-complete in data complexity (scale polynomially in $|\mathcal{A}|$)
  - can be shown e.g. by rewriting into Datalog
- NP-complete in combined complexity

Combined approach:

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

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

- P-complete in data complexity (scale polynomially in $|\mathcal{A}|$)
  - can be shown e.g. by rewriting into Datalog
- NP-complete in combined complexity

Combined approach:

- 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

RESEARCH TRENDS IN OMQA

Lots of work on developing and implementing efficient OMQA algorithms

Focus mostly on DL-Lite (and related dialects):

- First algorithm PerfectRef proposed in mid-2000's
- Rewrites into UCQs, implemented in QUONTO
- Improved versions proposed in REQUIEM, PRESTO, RAPID, ...
- Some algorithms rewrite into positive existential queries or Datalog programs instead of UCQs
- Resulting queries are smaller, can be easier to evaluate

Tractable classes, fragments of lower complexity

Rewriting engines for other Horn DLs also developed, e.g.,

- REQUIEM and the related KYRIE cover several $\mathcal{EL}$ dialects
- CLIPPER, and recently RAPID cover Horn-$\mathcal{SHIQ}$

They usually rewrite into Datalog programs

Much attention devoted to understanding the limits of rewritability and size of rewritings

When are polynomial-size rewritings possible?

Can we give bounds on the size of rewritings?

Which non-DL-Lite ontologies can be rewritten into FO-queries?

$\rightsquigarrow$ related to non-uniform complexity:

- study specific pairs $(q, \mathcal{T})$, called ontology-mediated queries

Beyond classical OMQA

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

Support for building and maintaining OMQA systems

- module extraction
- ontology evolution
- query inseparability and emptiness

Improving the usability of OMQA systems

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

Questions ?

30th anniversary DL workshop

July 18-21, 2017

Montpellier