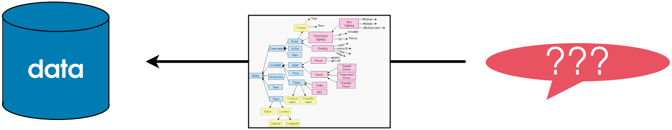
# QUERY ANSWERING WITH DESCRIPTION LOGIC ONTOLOGIES

Meghyn Bienvenu (*CNRS & Université de Montpellier*)
Magdalena Ortiz (*Vienna University of Technology*)

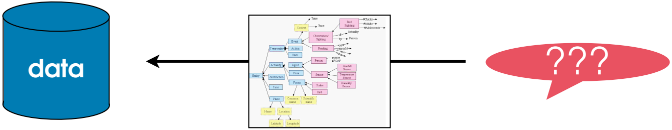
# ONTOLOGY-MEDIATED QUERY ANSWERING (OMQA)




**incomplete database**
(ground facts)

**ontology**
(logical theory)

**user query**



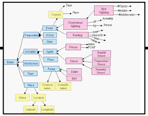

**patient data**
“Melanie has listeriosis”
“Paul has Lyme disease”

**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**

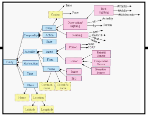
"Listeriosis & Lyme disease are bacterial infections"


**user query**

"Find all patients with bacterial infections"

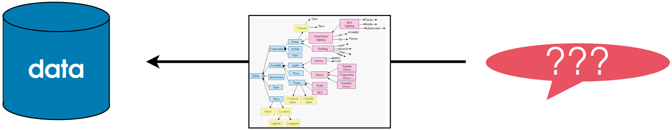
***expected answers*: Melanie, Paul**


**employee data**
"Marie is a professor"
"Mark teaches CS200"



**org. knowledge**
"Professors are teaching staff"
"Someone who teaches is part of the teaching staff"


**user query**
"Find all teaching staff"




**employee data**
“Marie is a professor”
“Mark teaches CS200”

**org. knowledge**
“Professors are teaching staff”
“Someone who teaches is part of the teaching staff”

**user query**
“Find all teaching staff”

***expected answers*: Marie, Mark**

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

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

# WHAT ARE ONTOLOGIES GOOD FOR?

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

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

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 standardize the terminology of an application domain

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

To present an intuitive and unified view of data sources

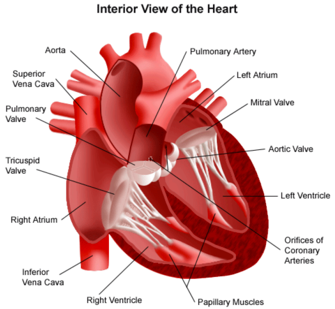
- ontology can be used to enrich the data vocabulary, making it easier for users to formulate their queries
- especially useful when integrating multiple data sources

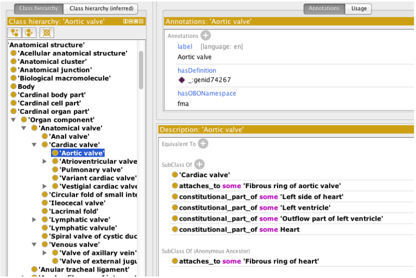
To support automated reasoning

- 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 (∼ 400,000 terms!), GALEN

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



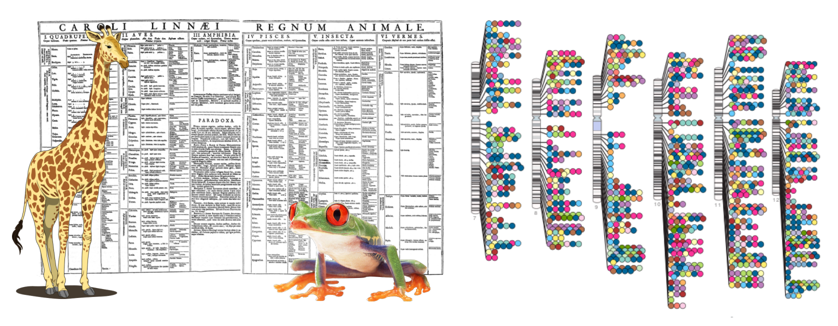




Querying & exchanging medical records (find patients for medical trials)

- 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 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

In this tutorial, focus on 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

Consider various types of queries

- Horn Description Logics
- Basics of OMQA

- Instance Queries
- Conjunctive Queries
- Navigational Queries
- Queries with Negation and Recursion

- Research Trends in OMQA

- Practical OMQA Systems: Ontop

# HORN DESCRIPTION LOGICS

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**

($\mathsf{Ind}(\mathcal{A})$: individuals appearing in ABox $\mathcal{A}$)

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

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

ABox contains **facts about specific individuals**

($\mathsf{Ind}(\mathcal{A})$: individuals appearing in ABox $\mathcal{A}$)

- finite set of **concept assertions** $A(a)$ and **role assertions** $r(a, b)$
- $\mathsf{IceCream}(d_2)$: dish $d_2$ *is of type* IceCream
- $\mathsf{hasDessert}(m, d_2)$: *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 ($\neg$)

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

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 ($\exists$, $\forall$)

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

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

Can build complex concepts and roles using constructors:

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

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

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

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

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

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

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

  (use $\mathsf{N}_\mathsf{R}^\pm$ for set of role names and inverse roles)

  (use $\mathsf{inv}(r)$ to toggle $^-$: $\mathsf{inv}(r) = r^-$, $\mathsf{inv}(r^-) = r$ )

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 ($\exists$, $\forall$)

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

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

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

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

  (use $\mathsf{N}_\mathsf{R}^\pm$ for set of role names and inverse roles)

  (use $\mathsf{inv}(r)$ to toggle $^-$: $\mathsf{inv}(r) = r^-$, $\mathsf{inv}(r^-) = r$ )

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

**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)

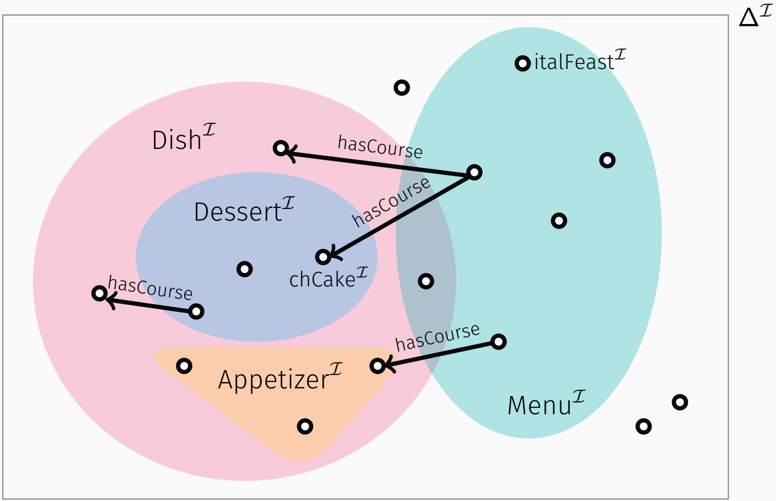
$\mathsf{hasIngred} \sqsubseteq \mathsf{contains}$ $\quad$ $\mathsf{ingredOf}^{-} \sqsubseteq \mathsf{hasIngred}$ $\quad$ $\mathsf{hasDessert} \sqsubseteq \mathsf{hasCourse}$

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

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 $a^\mathcal{I} \in \Delta^\mathcal{I}$

# EXAMPLE: INTERPRETATION




**4 concept names**: Dish, Dessert, Appetizer, Menu

**1 role name**: hasCourse **2 individual names**: italFeast, chCake

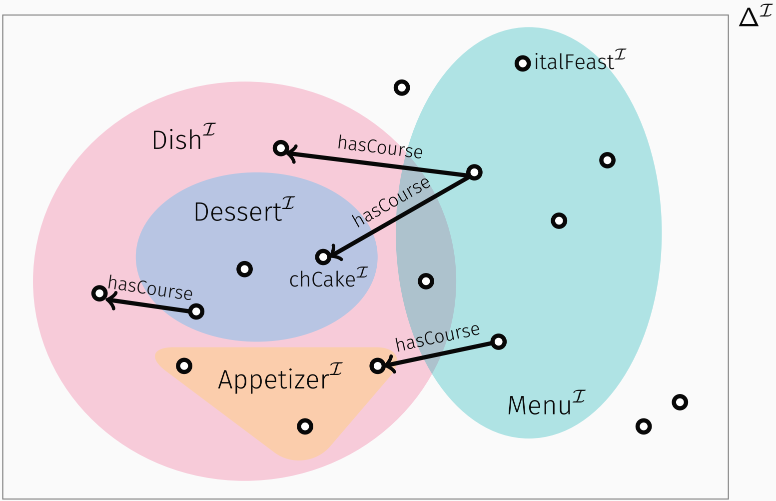
Interpretation $\mathcal{I}$ ("possible world")

- 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:

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

## BACK TO THE EXAMPLE




Dish $\sqcap$ Menu    Dessert $\sqcap$ Appetizer    $\exists$hasCourse.$\top$    $\exists$hasCourse$^{-}$.Dessert

## 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 $r(a, b) \quad \Leftrightarrow \quad (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

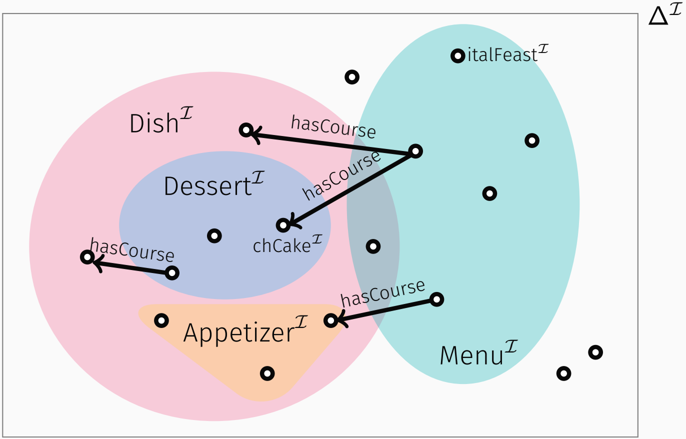
## Satisfaction in an interpretation

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




Which of the following assertions / axioms is satisfied in $\mathcal{I}$?

$\mathsf{Dessert} \sqsubseteq \mathsf{Dish}$ $\quad$ $\mathsf{Dish} \sqcap \mathsf{Menu} \sqsubseteq \bot$ $\quad$ $\mathsf{Menu} \sqsubseteq \exists \mathsf{hasCourse}.\top$

$\exists \mathsf{hasCourse}^{-}.\top \sqsubseteq \mathsf{Dish}$ $\quad$ $\mathsf{Menu}(\mathsf{italFeast})$ $\quad$ $\mathsf{hasCourse}(\mathsf{italFeast}, \mathsf{chCake})$

**Idea**: Horn DLs cannot express disjunction (*explicitly or implicitly*)

- better computational properties than non-Horn DLs (more on this later)

Idea: Horn DLs cannot express disjunction (*explicitly or implicitly*)

- better computational properties than non-Horn DLs (more on this later)

DL-Lite$_R$

- concept inclusions $B_1 \sqsubseteq (\neg)B_2$ — $B_1, B_2$ either $A \in \mathsf{N_C}$ or $\exists R$ ($R \in \mathsf{N}_\mathsf{R}^{\pm}$)
- role inclusions $R_1 \sqsubseteq (\neg)R_2$ — $R_1, R_2 \in \mathsf{N}_\mathsf{R}^{\pm}$

**Idea**: Horn DLs cannot express disjunction (*explicitly or implicitly*)

- better computational properties than non-Horn DLs (more on this later)

**DL-Lite$_R$**

- concept inclusions $B_1 \sqsubseteq (\neg)B_2$ — $B_1, B_2$ either $A \in \mathsf{N_C}$ or $\exists R$ ($R \in \mathsf{N_R^{\pm}}$)
- role inclusions $R_1 \sqsubseteq (\neg)R_2$ — $R_1, R_2 \in \mathsf{N_R^{\pm}}$

$\mathcal{EL}$

- allows only $\top$, $\sqcap$, and $\exists r.C$ as constructors
- only concept inclusions in TBox

Idea: Horn DLs cannot express disjunction (*explicitly or implicitly*)

- better computational properties than non-Horn DLs (more on this later)

**DL-Lite$_R$**

- concept inclusions $B_1 \sqsubseteq (\neg)B_2$ — $B_1, B_2$ either $A \in \mathsf{N_C}$ or $\exists R$ ($R \in \mathsf{N_R^{\pm}}$)
- role inclusions $R_1 \sqsubseteq (\neg)R_2$ — $R_1, R_2 \in \mathsf{N_R^{\pm}}$

$\mathcal{EL}$

- allows only $\top$, $\sqcap$, and $\exists r.C$ as constructors
- only concept inclusions in TBox

$\mathcal{ELHI}_\bot$

- additionally allows for $\bot$ and inverse roles ($r^-$)
- can also have role inclusions

# SOME IMPORTANT HORN DLS

Idea: Horn DLs cannot express disjunction (*explicitly or implicitly*)

- better computational properties than non-Horn DLs (more on this later)

**DL-Lite$_R$**

- concept inclusions $B_1 \sqsubseteq (\neg)B_2$ — $B_1, B_2$ either $A \in \mathsf{N_C}$ or $\exists R$ ($R \in \mathsf{N_R^{\pm}}$)
- role inclusions $R_1 \sqsubseteq (\neg)R_2$ — $R_1, R_2 \in \mathsf{N_R^{\pm}}$

**$\mathcal{EL}$**

- allows only $\top$, $\sqcap$, and $\exists r.C$ as constructors
- only concept inclusions in TBox

**$\mathcal{ELHI}_\bot$**

- additionally allows for $\bot$ and inverse roles ($r^-$)
- can also have role inclusions

**Horn-$\mathcal{SHIQ}$**

- limited use of $\neg$, $\forall r.C$, and number restrictions ($\geq nR.C$, $\leq nR.C$)
- also have transitivity axioms (e.g. assert contains is transitive)

# BASICS OF OMQA

## ABOXES VS. DATABASES

ABoxes and databases (DBs) and are syntactically similar:

- ABox = finite set of assertions (unary and binary facts)
- Database = finite set of facts of arbitrary arity

# ABOXES VS. DATABASES

ABoxes and databases (DBs) and are syntactically similar:

- ABox = finite set of assertions (unary and binary facts)
- Database = finite set of facts of arbitrary arity

ABoxes interpreted under open world assumption:

- every assertion in the ABox is assumed to hold (true)
- assertions not present in the ABox may hold or not (unknown)

Each ABox gives rise to many interpretations (its models)

- models can be infinite, can have infinitely many models

ABoxes and databases (DBs) and are syntactically similar:

- ABox = finite set of assertions (unary and binary facts)
- Database = finite set of facts of arbitrary arity

ABoxes interpreted under open world assumption:

- every assertion in the ABox is assumed to hold (true)
- assertions not present in the ABox may hold or not (unknown)

Each ABox gives rise to many interpretations (its models)

- models can be infinite, can have infinitely many models

Databases interpreted under closed world assumption:

- every fact in the DB is assumed to hold (true)
- every fact not in the DB is assumed not to hold (false)

In other words, each DB corresponds to single finite interpretation

- domain of the interpretation = set of constants in DB

Database query $q$ of arity $n$ maps (Boolean query = arity 0)

$$\text{Database } \mathcal{D} \quad \rightsquigarrow \quad \mathsf{ans}(q, \mathcal{D}) = \text{ set of } n\text{-tuples of constants from } \mathcal{D}$$

Database query $q$ of arity $n$ maps (Boolean query = arity 0)

$$\text{Database } \mathcal{D} \quad \rightsquigarrow \quad \mathsf{ans}(q, \mathcal{D}) = \text{ set of } n\text{-tuples of constants from } \mathcal{D}$$

$$\text{Interpretation } \mathcal{I} \quad \rightsquigarrow \quad \mathsf{ans}(q, \mathcal{I}) = \text{ set of } n\text{-tuples of elements from } \mathcal{I}$$

Database query $q$ of arity $n$ maps (Boolean query = arity 0)

Database $\mathcal{D}$ $\rightsquigarrow$ $\mathsf{ans}(q, \mathcal{D}) =$ set of $n$-tuples of constants from $\mathcal{D}$

Interpretation $\mathcal{I}$ $\rightsquigarrow$ $\mathsf{ans}(q, \mathcal{I}) =$ set of $n$-tuples of elements from $\mathcal{I}$

First-order (FO) query = first-order formula

- arity of FO query = number of free variables
- answers = substitutions for free vars that make formula hold
- example: $\mathsf{Dish}(x) \wedge \forall y.(\mathsf{contains}(x, y) \rightarrow \neg\mathsf{Spicy}(y))$

Database query $q$ of arity $n$ maps (Boolean query = arity 0)

Database $\mathcal{D}$ $\rightsquigarrow$ $\mathsf{ans}(q, \mathcal{D}) =$ set of $n$-tuples of constants from $\mathcal{D}$

Interpretation $\mathcal{I}$ $\rightsquigarrow$ $\mathsf{ans}(q, \mathcal{I}) =$ set of $n$-tuples of elements from $\mathcal{I}$

First-order (FO) query = first-order formula

- arity of FO query = number of free variables
- answers = substitutions for free vars that make formula hold
- example: $\mathsf{Dish}(x) \wedge \forall y.(\mathsf{contains}(x, y) \rightarrow \neg\mathsf{Spicy}(y))$

Datalog queries = finite set of Datalog rules + 'goal' relation

- arity of Datalog query = arity of goal relation
- answers = exhaustively apply rules to DB / interpretation, collect tuples in goal relation
- example: rules $\mathsf{contains}(x, z) \leftarrow \mathsf{contains}(x, y), \mathsf{contains}(y, z)$ and $\underline{\mathsf{SpicyDish}(\mathbf{x})} \leftarrow \mathsf{Dish}(x), \mathsf{contains}(x, y), \mathsf{Spicy}(y)$

Problem: each KB gives rise to multiple interpretations (its models), but DB query semantics defines answers w.r.t. a single interpretation

**Problem**: each KB gives rise to multiple interpretations (its models), but DB query semantics defines answers w.r.t. a single interpretation

**Solution**: adopt certain answer semantics

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

**Problem**: each KB gives rise to multiple interpretations (its models), but DB query semantics defines answers w.r.t. a single interpretation

**Solution**: adopt certain answer semantics

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

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

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

**Problem**: each KB gives rise to multiple interpretations (its models), but DB query semantics defines answers w.r.t. a single interpretation

**Solution**: adopt certain answer semantics

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

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

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

**Question**: what happens if $\mathcal{K}$ is unsatisfiable?

**Problem**: each KB gives rise to multiple interpretations (its models), but DB query semantics defines answers w.r.t. a single interpretation

**Solution**: adopt certain answer semantics

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

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

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

**Question**: what happens if $\mathcal{K}$ is unsatisfiable?

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

Consider the query $\mathsf{q}(\mathsf{x}) = \mathsf{Dessert}(\mathsf{x})$ and the DL-Lite$_R$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

$$\mathcal{T} = \{\mathsf{Cake} \sqsubseteq \mathsf{Dessert} \quad \mathsf{IceCream} \sqsubseteq \mathsf{Dessert} \quad \mathsf{hasDessert} \sqsubseteq \mathsf{hasCourse}$$
$$\exists \mathsf{hasCourse} \sqsubseteq \mathsf{Menu} \quad \exists \mathsf{hasDessert}^- \sqsubseteq \mathsf{Dessert}\}$$

$$\mathcal{A} = \{\mathsf{Cake}(d_1) \quad \mathsf{IceCream}(d_2) \quad \mathsf{Dessert}(d_3) \quad \mathsf{hasDessert}(m, d_4)\}$$

# EXAMPLE: CERTAIN ANSWERS

Consider the query $\mathsf{q}(\mathsf{x}) = \mathsf{Dessert}(\mathsf{x})$ and the DL-Lite$_R$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

$\mathcal{T} = \{\mathsf{Cake} \sqsubseteq \mathsf{Dessert} \quad \mathsf{IceCream} \sqsubseteq \mathsf{Dessert} \quad \mathsf{hasDessert} \sqsubseteq \mathsf{hasCourse}$
$\quad \exists\mathsf{hasCourse} \sqsubseteq \mathsf{Menu} \quad \exists\mathsf{hasDessert}^{-} \sqsubseteq \mathsf{Dessert}\}$

$\mathcal{A} = \{\mathsf{Cake}(d_1) \quad \mathsf{IceCream}(d_2) \quad \mathsf{Dessert}(d_3) \quad \mathsf{hasDessert}(m, d_4)\}$

**Certain answers** to $q$ w.r.t. $\mathcal{K}$:

Consider the query $\mathbf{q}(\mathbf{x}) = \mathsf{Dessert}(\mathbf{x})$ and the DL-Lite$_R$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

$\mathcal{T} = \{\mathsf{Cake} \sqsubseteq \mathsf{Dessert} \quad \mathsf{IceCream} \sqsubseteq \mathsf{Dessert} \quad \mathsf{hasDessert} \sqsubseteq \mathsf{hasCourse}$
$\quad \exists\mathsf{hasCourse} \sqsubseteq \mathsf{Menu} \quad \exists\mathsf{hasDessert}^- \sqsubseteq \mathsf{Dessert}\}$

$\mathcal{A} = \{\mathsf{Cake}(d_1) \quad \mathsf{IceCream}(d_2) \quad \mathsf{Dessert}(d_3) \quad \mathsf{hasDessert}(m, d_4)\}$

**Certain answers** to $q$ w.r.t. $\mathcal{K}$:

- $d_1 \in \mathrm{cert}(q, \mathcal{K})$ $\qquad \mathsf{Cake}(d_1) \in \mathcal{A}$, $\mathsf{Cake} \sqsubseteq \mathsf{Dessert} \in \mathcal{T}$
- $d_2 \in \mathrm{cert}(q, \mathcal{K})$ $\qquad \mathsf{IceCream}(d_2) \in \mathcal{A}$, $\mathsf{IceCream} \sqsubseteq \mathsf{Dessert} \in \mathcal{T}$
- $d_3 \in \mathrm{cert}(q, \mathcal{K})$ $\qquad \mathsf{Dessert}(d_3) \in \mathcal{A}$
- $d_4 \in \mathrm{cert}(q, \mathcal{K})$ $\qquad \mathsf{hasDessert}(m, d_4) \in \mathcal{A}$, $\mathsf{hasDessert}^- \sqsubseteq \mathsf{Dessert} \in \mathcal{T}$

Consider the query $\mathbf{q}(\mathbf{x}) = \mathsf{Dessert}(\mathbf{x})$ and the DL-Lite$_R$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

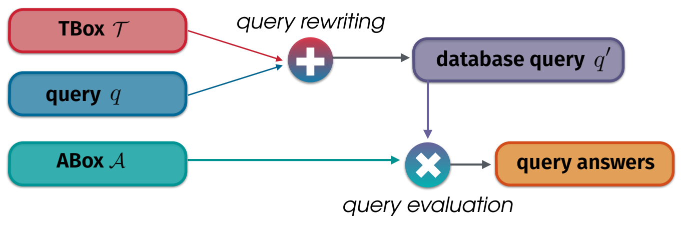
$$\mathcal{T} = \{\mathsf{Cake} \sqsubseteq \mathsf{Dessert} \quad \mathsf{IceCream} \sqsubseteq \mathsf{Dessert} \quad \mathsf{hasDessert} \sqsubseteq \mathsf{hasCourse}$$
$$\exists\mathsf{hasCourse} \sqsubseteq \mathsf{Menu} \quad \exists\mathsf{hasDessert}^{-} \sqsubseteq \mathsf{Dessert}\}$$
$$\mathcal{A} = \{\mathsf{Cake}(d_1) \quad \mathsf{IceCream}(d_2) \quad \mathsf{Dessert}(d_3) \quad \mathsf{hasDessert}(m, d_4)\}$$

**Certain answers** to $q$ w.r.t. $\mathcal{K}$:

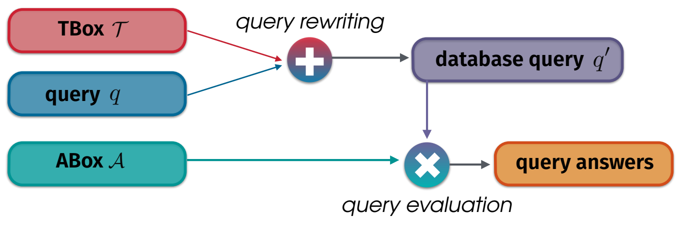
- $d_1 \in \mathsf{cert}(q, \mathcal{K})$ — $\mathsf{Cake}(d_1) \in \mathcal{A}$, $\mathsf{Cake} \sqsubseteq \mathsf{Dessert} \in \mathcal{T}$
- $d_2 \in \mathsf{cert}(q, \mathcal{K})$ — $\mathsf{IceCream}(d_2) \in \mathcal{A}$, $\mathsf{IceCream} \sqsubseteq \mathsf{Dessert} \in \mathcal{T}$
- $d_3 \in \mathsf{cert}(q, \mathcal{K})$ — $\mathsf{Dessert}(d_3) \in \mathcal{A}$
- $d_4 \in \mathsf{cert}(q, \mathcal{K})$ — $\mathsf{hasDessert}(m, d_4) \in \mathcal{A}$, $\mathsf{hasDessert}^{-} \sqsubseteq \mathsf{Dessert} \in \mathcal{T}$

The fifth individual $m$ **is not a certain answer**: can construct model $\mathcal{J}$ of $\mathcal{K}$ in which $m^{\mathcal{J}} \notin \mathsf{Dessert}^{\mathcal{J}}$

Query rewriting: Reduces problem of finding certain answers to standard DB query evaluation (⇝ exploit existing DB systems)



Query rewriting: Reduces problem of finding certain answers to standard DB query evaluation (⇝ exploit existing DB systems)

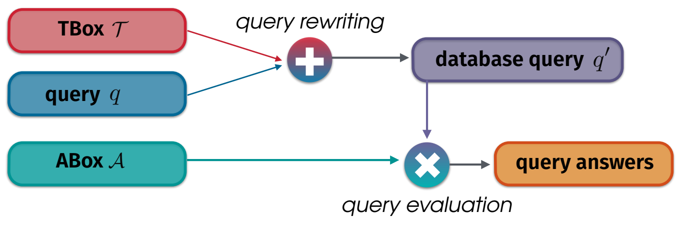



Call $q'(\vec{x})$ a rewriting of $q(\vec{x})$ and $\mathcal{T}$ iff for every ABox $\mathcal{A}$ and tuple $\vec{a}$

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

($\mathcal{I}_{\mathcal{A}}$ = treat $\mathcal{A}$ as DB)

Query rewriting: Reduces problem of finding certain answers to standard DB query evaluation (⇝ exploit existing DB systems)




Call $q'(\vec{x})$ a rewriting of $q(\vec{x})$ and $\mathcal{T}$ iff for every ABox $\mathcal{A}$ and tuple $\vec{a}$

$$\mathcal{T}, \mathcal{A} \models q(\vec{a}) \quad \Leftrightarrow \quad \vec{a} \in \mathsf{ans}(q'(\vec{x}), \mathcal{I}_\mathcal{A}) \qquad (\mathcal{I}_\mathcal{A} = \text{treat } \mathcal{A} \text{ as DB})$$

Types of rewritings: FO-rewritings (SQL), Datalog rewritings, ...

**Saturation**: Render explicit (some of) the implicit information contained in the KB, making it available for query evaluation

Saturation: Render explicit (some of) the implicit information contained in the KB, making it available for query evaluation

Simple use of saturation: (works e.g. for RDFS ontologies)

- use saturation to 'complete' the ABox by adding those assertions that are logically entailed from the KB
- then evaluate the query over the saturated ABox

**Saturation**: Render explicit (some of) the implicit information contained in the KB, making it available for query evaluation

**Simple use of saturation**: (works e.g. for RDFS ontologies)

- use saturation to 'complete' the ABox by adding those assertions that are logically entailed from the KB
- then evaluate the query over the saturated ABox

**More complex uses**:

- enrich the ABox in other ways (e.g. add new ABox individuals to witness the existential restrictions $\exists R.C$)
- combine saturation with query rewriting

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** $\vec{a}$ **belong to** $\mathsf{cert}(q, (\mathcal{T}, \mathcal{A}))$?

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 $\vec{a}$ belong to** $\mathsf{cert}(q, (\mathcal{T}, \mathcal{A}))$? |

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.)

We will mention the following standard classes:

| | |
|---|---|
| P | problems solvable in deterministic polynomial time |
| NP | problems solvable in non-det. polynomial time |
| coNP | problems whose complement is solvable in non-deterministic polynomial time |
| LogSpace | problems solvable in deterministic logarithmic space |
| NLogSpace | problems solvable in non-det. logarithmic space |
| PSpace | problems solvable in polynomial space (note: =NPSpace) |
| Exp | problems solvable in deterministic exponential time |

We will mention the following standard classes:

| | |
|---|---|
| P | problems solvable in **deterministic polynomial time** |
| NP | problems solvable in **non-det. polynomial time** |
| coNP | problems whose **complement** is solvable in **non-deterministic polynomial time** |
| LogSpace | problems solvable in **deterministic logarithmic space** |
| NLogSpace | problems solvable in **non-det. logarithmic space** |
| PSpace | problems solvable in **polynomial space** (note: =NPSpace) |
| Exp | problems solvable in **deterministic exponential time** |

Another less known but important class:

| | |
|---|---|
| $AC_0$ | problems solvable by **uniform family of polynomial-size constant-depth circuits** |

Relationships between classes:

$$AC_0 \subsetneq \text{LogSpace} \subseteq \text{NLogSpace} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSpace} \subseteq \text{Exp}$$