Knowledge Graphs, Description Logics and Reasoning on Data

C. Bourgaux, M. Thomazo

Organization

- Lecturers
  - Camille Bourgaux, CNRS researcher at ENS
    - camille.bourgaux@ens.fr
  - Michaël Thomazo, Inria researcher at ENS
    - michael.thomazo@inria.fr
- Course day and room not the same each week, check ADE
- Evaluation: written exam
- Slides will be made available at: `https://www.di.ens.fr/camille.bourgaux/teaching/IASD-KGDL/index.html`

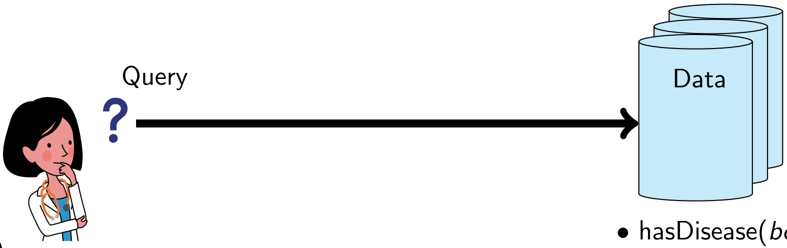
Organization

Knowledge Graphs, Description Logics and Reasoning on Data
1: Introduction

C. Bourgaux, M. Thomazo

Outline

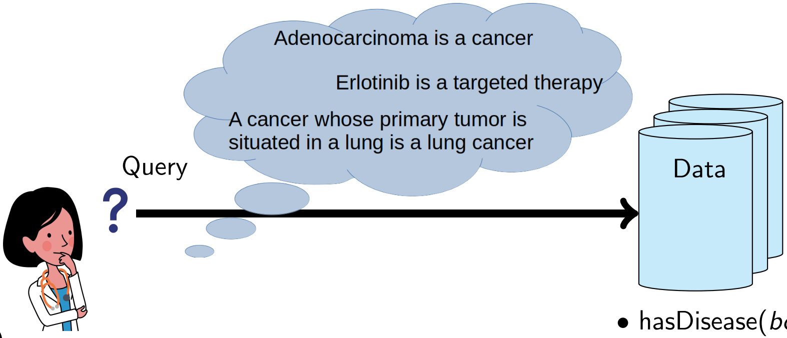
Motivation: Incomplete Data



$q(x) =$
$\text{hasDisease}(x, y) \land \text{LungCancer}(y)$
$\land \text{gotTreatment}(x, z) \land \text{TargetedTherapy}(z)$

- $\text{hasDisease}(bob, d_1)$
- $\text{Adenocarcinoma}(d_1)$
- $\text{primaryTumor}(d_1, o_1)$
- $\text{Lung}(o_1)$
- $\text{gotTreatment}(bob, t_1)$
- $\text{Erlotinib}(t_1)$

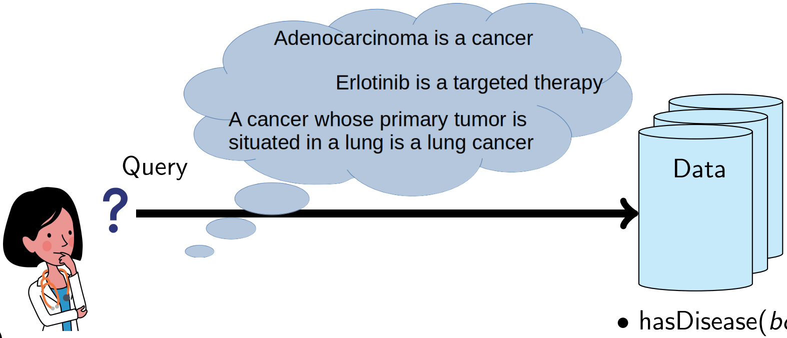
Motivation: Incomplete Data


Adenocarcinoma is a cancer
Erlotinib is a targeted therapy
A cancer whose primary tumor is situated in a lung is a lung cancer
Query
Data

$q(x) =$
$\text{hasDisease}(x, y) \wedge \text{LungCancer}(y)$
$\wedge\ \text{gotTreatment}(x, z) \wedge \text{TargetedTherapy}(z)$

- $\text{hasDisease}(bob, d_1)$
- $\text{Adenocarcinoma}(d_1)$
- $\text{primaryTumor}(d_1, o_1)$
- $\text{Lung}(o_1)$
- $\text{gotTreatment}(bob, t_1)$
- $\text{Erlotinib}(t_1)$

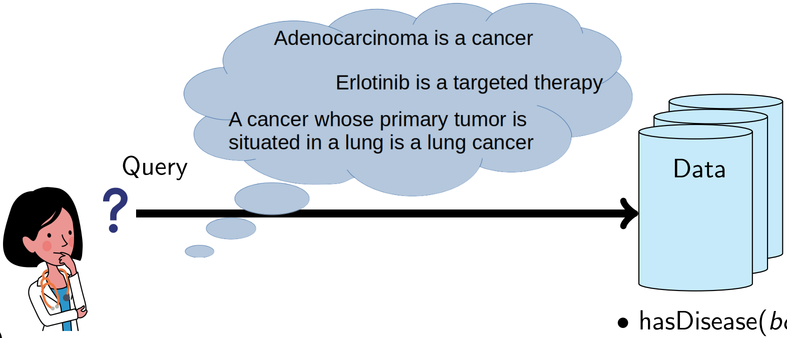
Motivation: Incomplete Data


Adenocarcinoma is a cancer
Erlotinib is a targeted therapy
A cancer whose primary tumor is situated in a lung is a lung cancer
Query
?
Data

$q(x) =$
$\quad \text{hasDisease}(x, y) \wedge \text{LungCancer}(y)$
$\quad \wedge \text{gotTreatment}(x, z) \wedge \text{TargetedTherapy}(z)$
$\vee \quad \text{hasDisease}(x, y) \wedge \text{Cancer}(y) \wedge \text{primaryTumor}(y, u)$
$\quad \wedge \text{Lung}(u) \wedge \text{gotTreatment}(x, z) \wedge \text{TargetedTherapy}(z)$

- $\text{hasDisease}(bob, d_1)$
- $\text{Adenocarcinoma}(d_1)$
- $\text{primaryTumor}(d_1, o_1)$
- $\text{Lung}(o_1)$
- $\text{gotTreatment}(bob, t_1)$
- $\text{Erlotinib}(t_1)$

Motivation: Incomplete Data


Adenocarcinoma is a cancer
Erlotinib is a targeted therapy
A cancer whose primary tumor is situated in a lung is a lung cancer
Query
Data

- $\text{hasDisease}(bob, d_1)$
- $\text{Adenocarcinoma}(d_1)$
- $\text{primaryTumor}(d_1, o_1)$
- $\text{Lung}(o_1)$
- $\text{gotTreatment}(bob, t_1)$
- $\text{Erlotinib}(t_1)$

$$
\begin{aligned}
q(x) = \; & \text{hasDisease}(x, y) \wedge \text{LungCancer}(y) \\
& \wedge \text{gotTreatment}(x, z) \wedge \text{TargetedTherapy}(z) \\
\vee \; & \text{hasDisease}(x, y) \wedge \text{Cancer}(y) \wedge \text{primaryTumor}(y, u) \\
& \wedge \text{Lung}(u) \wedge \text{gotTreatment}(x, z) \wedge \text{TargetedTherapy}(z) \\
\vee \; & \text{hasDisease}(x, y) \wedge \text{Adenocarcinoma}(y) \wedge \text{primaryTumor}(y, u) \\
& \wedge \text{Lung}(u) \wedge \text{gotTreatment}(x, z) \wedge \text{TargetedTherapy}(z) \\
\vee \; & \dots
\end{aligned}
$$

Motivation: Incomplete Data

- Complete the data using rules/triggers?
  - may not be feasible (access right...)
  - value creation/nulls?
  - size?
  - updates?

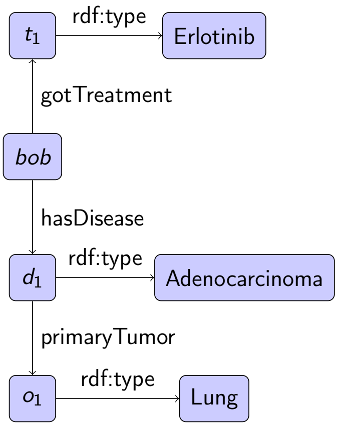
Motivation: Incomplete Data

- Complete the data using rules/triggers?
  - may not be feasible (access right...)
  - value creation/nulls?
  - size?
  - updates?
- Use a knowledge graph to model the expert knowledge?

Reminders: Knowledge Graphs and Semantic Web

- Semantic Web: extension of the World Wide Web that aims at making the web content "understandable" by computers
- Annotate websites data with meta-data
- Standards set by the World Wide Web Consortium (W3C)
  - IRI (Internationalized Resource Identifier) to name things
  - RDF (Resource Description Framework)
    - standard graph model for data interchange on the Web
    - RDF graph: set of triples (subject, predicate, object)
    - graph nodes are entities (vertices labelled with IRIs), data values (vertices labelled with literals), or blank nodes
  - SPARQL (SPARQL Protocol and RDF Query Language)
    - query (and update) language for RDF data
  - RDFS (RDF Schema)
    - data-modelling vocabulary for RDF data: allows to define classes and relations and specify subclasses and domain/range

Using Knowledge Graphs to Answer Complex Questions



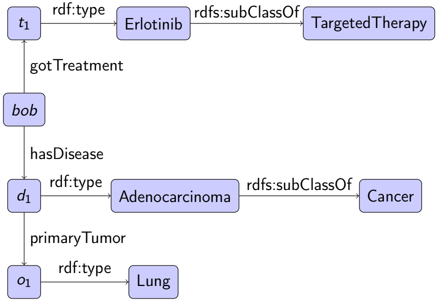
Data as an RDF graph

Using Knowledge Graphs to Answer Complex Questions

$$\begin{tikzcd}
t_1 \arrow[r, "\text{rdf:type}"] & \text{Erlotinib} \arrow[r, "\text{rdfs:subClassOf}"] & \text{TargetedTherapy} \\
\textit{bob} \arrow[u, "\text{gotTreatment}"'] \arrow[d, "\text{hasDisease}"] & & \\
d_1 \arrow[r, "\text{rdf:type}"] \arrow[d, "\text{primaryTumor}"] & \text{Adenocarcinoma} \arrow[r, "\text{rdfs:subClassOf}"] & \text{Cancer} \\
o_1 \arrow[r, "\text{rdf:type}"] & \text{Lung} &
\end{tikzcd}$$

Modelling expert knowledge with RDFS. What about “A cancer whose primary tumor is situated in a lung is a lung cancer” ?

Using Knowledge Graphs to Answer Complex Questions



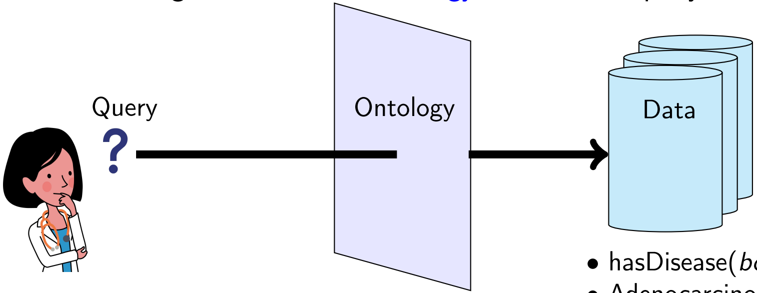
```
SELECT DISTINCT ?patient WHERE {
 { ?patient hasDisease ?disease .
   ?disease rdf:type/rdfs:subClassOf* LungCancer .
   ?patient gotTreatment  rdf:type/rdfs:subClassOf* TargetedTherapy
 }
 UNION
 { ?patient hasDisease ?disease .
   ?disease rdf:type/rdfs:subClassOf* Cancer .
   ?disease primaryTumor ?organ .
   ?organ rdf:type Lung .
   ?patient gotTreatment  rdf:type/rdfs:subClassOf* TargetedTherapy
 }
}
```

Using Knowledge Graphs to Answer Complex Questions

- Knowledge graphs and SPARQL queries allow us to get answers to complex queries
- But SPARQL queries may become very (too) complex
- Need for a way of formulating simpler queries, closer to the natural language of the user, and still get all the answers from the data
- Ontologies allow to formalize knowledge and delegate the reasoning to the machine

Ontology-Mediated Query Answering

- Add a semantic layer to abstract from the specific way data is stored and infer implicit information
- "Knowledge" stored in an ontology and used to query the data



$q(x) =$
$\text{hasDisease}(x, y)$
$\wedge\ \text{LungCancer}(y)$
$\wedge\ \text{gotTreatment}(x, z)$
$\wedge\ \text{TargetedTherapy}(z)$

- $\text{Adenocarcinoma}(x) \rightarrow \text{Cancer}(x)$
- $\text{Cancer}(x) \wedge \text{Lung}(y) \wedge \text{primaryTumor}(x, y) \rightarrow \text{LungCancer}(x)$
- $\text{Erlotinib}(x) \rightarrow \text{TargetedTherapy}(x)$

- $\text{hasDisease}(bob, d_1)$
- $\text{Adenocarcinoma}(d_1)$
- $\text{primaryTumor}(d_1, o_1)$
- $\text{Lung}(o_1)$
- $\text{gotTreatment}(bob, t_1)$
- $\text{Erlotinib}(t_1)$

Open-World Assumption

- To use an ontology to query the data, we need to change the data semantics to take into account its incompleteness
- Open-world assumption: the absence of a fact does not mean that it is false (different from closed-world assumption used for databases)
- The data is a partial description of the world

Ontologies

- “Originally, the term ontology comes from philosophy - it goes as far back as Aristotle’s attempt to classify the things in the world - where it is employed to describe the existence of beings in the world. [...] An ontology is a formal, explicit specification of a shared conceptualisation.
    - A ‘conceptualisation’ refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon.
    - ‘Explicit’ means that the type of concepts used, and the constraints on their use are explicitly defined. For example, in medical domains, the concepts are diseases and symptoms, the relations between them are causal and a constraint is that a disease cannot cause itself.
    - ‘Formal’ refers to the fact that the ontology should be machine readable, which excludes natural language.
    - ‘Shared’ reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group.”

(Struder et al. 1998)

Ontologies

- An ontology is generally a logical theory that models a domain of interest
- It defines the terminology (vocabulary) of the domain and the semantics relationships between terms
- Example (family domain)
  - Terms: parent, mother, sister, sibling, ...
  - Relationships between terms: "mother" is a subclass of "parent", "sister" is both in the domain and in the range of "has sibling"...

Reasons for Using Ontologies

- Standardize the terminology of an application domain: make it easy to share information – well-defined syntax and formal logic-based semantics (i.e. meaning)
  - complex industrial systems description, scientific knowledge (medicine, life science...)

Reasons for Using Ontologies

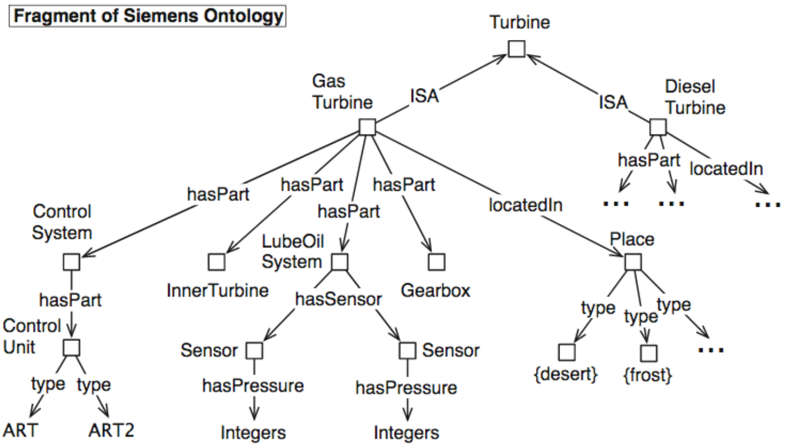
- Standardize the terminology of an application domain: make it easy to share information – well-defined syntax and formal logic-based semantics (i.e. meaning)
  - complex industrial systems description, scientific knowledge (medicine, life science...)
- Present an intuitive and unified view of data sources: make it easy to formulate queries
  - data integration, semantic web

Reasons for Using Ontologies

- Standardize the terminology of an application domain: make it easy to share information – well-defined syntax and formal logic-based semantics (i.e. meaning)
  - complex industrial systems description, scientific knowledge (medicine, life science...)
- Present an intuitive and unified view of data sources: make it easy to formulate queries
  - data integration, semantic web
- Support automated reasoning: logical inferences allow us to take advantage of implicit knowledge to answer queries – computational aspects can be studied to design ontology languages and tools that allow for efficient reasoning
  - expert systems, semantic web, ontology-based data access

Examples of Ontologies

In industry

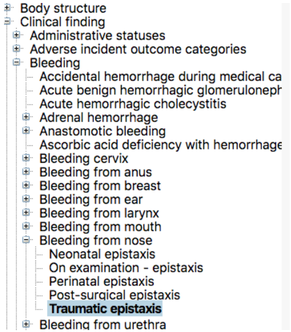


From: How Semantic Technologies Can Enhance Data Access at Siemens Energy, Kharlamov et al., ISWC 2014

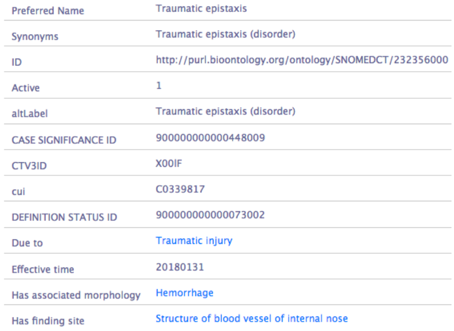
- Energy sector: turbines diagnostics

Examples of Ontologies

Medical ontologies



- Body structure
- Clinical finding
  - Administrative statuses
  - Adverse incident outcome categories
  - Bleeding
    - Accidental hemorrhage during medical ca
    - Acute benign hemorrhagic glomerulonepł
    - Acute hemorrhagic cholecystitis
    - Adrenal hemorrhage
    - Anastomotic bleeding
    - Ascorbic acid deficiency with hemorrhage
    - Bleeding cervix
    - Bleeding from anus
    - Bleeding from breast
    - Bleeding from ear
    - Bleeding from larynx
    - Bleeding from mouth
    - Bleeding from nose
      - Neonatal epistaxis
      - On examination - epistaxis
      - Perinatal epistaxis
      - Post-surgical epistaxis
      - **Traumatic epistaxis**
    - Bleeding from urethra

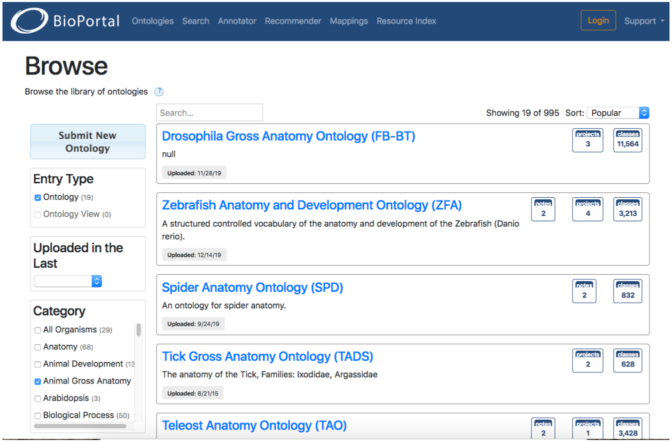


Preferred Name	Traumatic epistaxis
Synonyms	Traumatic epistaxis (disorder)
ID	http://purl.bioontology.org/ontology/SNOMEDCT/232356000
Active	1
altLabel	Traumatic epistaxis (disorder)
CASE SIGNIFICANCE ID	900000000000448009
CTV3ID	X00lF
cui	C0339817
DEFINITION STATUS ID	900000000000073002
Due to	Traumatic injury
Effective time	20180131
Has associated morphology	Hemorrhage
Has finding site	Structure of blood vessel of internal nose

- SNOMED CT: general medical ontology ($>$ 350 000 concepts)
  - multilingual, mapped to other international standards
  - used for recording medical information: information sharing, decision-making assistance systems, gathering data for clinical research, monitoring population health and clinical practices...
- NCI (National Cancer Institute Thesaurus), FMA (Foundational Model of Anatomy), GO (Gene Ontology) ...

Examples of Ontologies

Ontologies for life sciences



Browse

Browse the library of ontologies

Search...

Showing 19 of 995 Sort: Popular

Entry Type
- [✓ Ontology (19)]
- Ontology View (0)

Uploaded in the Last

Category
- All Organisms (29)
- Anatomy (68)
- Animal Development (13
- [✓ Animal Gross Anatomy]
- Arabidopsis (3)
- Biological Process (50)

Drosophila Gross Anatomy Ontology (FB-BT) — projects 3, classes 11,564
null
Uploaded: 11/28/19

Zebrafish Anatomy and Development Ontology (ZFA) — notes 2, projects 4, classes 3,213
A structured controlled vocabulary of the anatomy and development of the Zebrafish (Danio rerio).
Uploaded: 12/14/19

Spider Anatomy Ontology (SPD) — notes 2, classes 832
An ontology for spider anatomy.
Uploaded: 9/24/19

Tick Gross Anatomy Ontology (TADS) — projects 2, classes 628
The anatomy of the Tick, Families: Ixodidae, Argassidae
Uploaded: 8/21/15

Teleost Anatomy Ontology (TAO) — notes 2, projects 1, classes 3,428

- Bioportal repository contains hundreds of ontologies about biology and chemistry (`http://bioportal.bioontology.org/`)

The Research Field

- Symbolic AI:
  - uses symbols that represent real-world entities or concepts to explicitly define human knowledge and reasoning processes
  - make machine able to reason
  - allows for explanation of reasoning: every step of reasoning use human readable representation of the problem and exact reasoning rules
  - allows humans to control the set of knowledge used by the machine (some things cannot or should not be learnt: biases, laws, policies...)
  - examples of other subfields of symbolic AI: planning and scheduling, multi-agents systems, constraints satisfaction and optimization, search...

The Research Field

- Knowledge representation and reasoning (KR):
  - “Research in the field of knowledge representation and reasoning is usually focused on methods for providing high-level descriptions of the world that can be effectively used to build intelligent applications. In this context, “intelligent” refers to the ability of a system to find implicit consequences of its explicitly represented knowledge.” (Brachman and Nardi, 2003)
  - “field of AI dedicated to representing information about the world in a form that a computer system can use to solve complex tasks [...]. KR incorporates findings from psychology about how humans solve problems and represent knowledge in order to design formalisms that will make complex systems easier to design and build. KR also incorporates findings from logic to automate various kinds of reasoning, such as the application of rules or the relations of sets and subsets.” (Wikipedia, 2022)
  - Logic-based formalisms, expert systems, semantic networks...

Knowledge Representation and Reasoning

- Knowledge representation: define languages for specifying knowledge
  - Syntax (what can be written): determines precisely whether an expression is a well-formed statement
  - Semantics (meaning): determines what each well-formed statement means and whether a statement is a consequence of the knowledge represented
- Automated reasoning: provide inference tools (reasoners) to deduce implicit knowledge (consequences) of the knowledge explicitly represented
  - Independent from the specific knowledge and domain
  - Reasoners may solve different reasoning problems

Description Logics

- Description Logics is a family of ontology languages
  - each description logic describes a language
  - fragments of first-order logic (FOL)
  - decidable, often tractable
  - differ in expressive power and reasoning complexity
  - very well studied, lots of tools (highly optimized reasoners...)

Description Logics: Syntax

Basic building blocks

- Atomic concepts (unary predicates)
  - represent sets of entities
  - Parent, Mother, Sister ...
- Atomic roles (binary predicates)
  - represent relations between two entities
  - hasChild, isMarriedTo ...
- Individuals (constants)
  - represent entities
  - *BarackObama*, *SashaObama* ...
- These basic building blocks can be combined using constructors to describe complex concepts or roles
  - the set of entities who are mothers who don't have any male child who is married is captured by the complex concept: $\text{Mother} \sqcap \neg(\exists \text{hasChild}.(\text{Male} \sqcap \exists \text{isMarriedTo}.\top))$

Description Logics: Syntax

Most common concept constructors

Let C, D be (potentially complex) concepts, and R be a (potentially complex) role

- Conjunction: $C \sqcap D$ (entities that belong to concepts C and D, i.e., to the intersection of C and D)
  - Mother $\sqcap$ Sister: "mothers that are also sisters"
- Disjunction: $C \sqcup D$ (entities that belong to concepts C or D, i.e., to the union of C and D)
  - Mother $\sqcup$ Father: "mothers or fathers"
- Negation: $\neg C$ (entities that do not belong to concept C, i.e., that belong to the complement of C)
  - $\neg$Parent: "those that are not parents"
- Existential restriction: $\exists R.C$ (entities that are related by R to some entity in C)
  - $\exists$hasChild.Male: "those who have a male child"
- Universal restriction: $\forall R.C$ (entities that are related by R only to entities in C)
  - $\forall$hasChild.Male: "those who have only male children"

Description Logics: Syntax

More concept constructors...

Let C be a (potentially complex) concept, and R be a (potentially complex) role, and a be an individual name

- Number restriction: $\geq kR.C$ (entities that are related by R to at least k entities in C)
  - $\geq$ 2hasChild.Male: “those who have at least two male children”
- Number restriction: $\leq kR.C$ (entities that are related by R to at most k entities in C)
  - $\leq$ 2hasChild.Male: “those who have at most two male children”
- Nominals: $\{a\}$ (the set that contains only a)
  - $\exists$hasChild.$\{BarackObama\}$: “those who have child Barack Obama”

Description Logics: Syntax

Examples of role constructors

Let R and S be (potentially complex) roles

- Inverse: R^- (pairs of entities such that the second one is related by an R to the first one)
    - hasChild^-: "has parent"
- Negation: $\neg R$ (complement of R: all pairs of entities not related by R)
    - $\neg\text{hasChild}$: "has not child"
- Composition: $R \circ S$ (pairs of entities such that the first one is related by an R to some entity related by an S to the second one)
    - $\text{hasChild} \circ \text{hasChild}$: "has grand-child"

Description Logics: Syntax

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

TBox (terminological box) specifies knowledge at intensional level

- finite set of axioms
- describes general knowledge about the domain
- defines a set of conceptual elements (concepts, roles) and states constraints describing the relationships between them

ABox (assertional box) specifies knowledge at extensional level

- finite set of assertions
- contains facts about specific individuals

Note: the term ontology is sometimes used to refer to the whole knowledge base rather than to the TBox alone.

Description Logics: Syntax

TBox axioms

The TBox may contain

- (General) concept inclusions: $C \sqsubseteq D$ (every instance of C is an instance of D)
  - Mother $\sqsubseteq$ Parent: "all mothers are parents"
  - Spouse $\sqsubseteq \exists$isMarriedTo.$\top$: "all spouses are married to someone"
  - Spouse $\sqsubseteq \neg$Bachelor: "one cannot be both a spouse and a bachelor"

Description Logics: Syntax
TBox axioms

The TBox may contain

- (General) concept inclusions: $C \sqsubseteq D$ (every instance of C is an instance of D)
  - Mother $\sqsubseteq$ Parent: “all mothers are parents”
  - Spouse $\sqsubseteq \exists$isMarriedTo.$\top$: “all spouses are married to someone”
  - Spouse $\sqsubseteq \neg$Bachelor: “one cannot be both a spouse and a bachelor”
- Role inclusions: $R \sqsubseteq S$
  - hasParent $\sqsubseteq$ hasAncestor: “one’s parents are one’s ancestors”
  - hasParent $\sqsubseteq$ hasChild^{-}: “one has parents those who have oneseflf as child”
  - hasParent $\sqsubseteq \neg$hasChild: “one cannot has parent one’s child”

Description Logics: Syntax

TBox axioms

The TBox may contain

- (General) concept inclusions: $C \sqsubseteq D$ (every instance of C is an instance of D)
    - Mother $\sqsubseteq$ Parent: “all mothers are parents”
    - Spouse $\sqsubseteq \exists$isMarriedTo.$\top$: “all spouses are married to someone”
    - Spouse $\sqsubseteq \neg$Bachelor: “one cannot be both a spouse and a bachelor”
- Role inclusions: $R \sqsubseteq S$
    - hasParent $\sqsubseteq$ hasAncestor: “one’s parents are one’s ancestors”
    - hasParent $\sqsubseteq$ hasChild^{-}: “one has parents those who have oneseflf as child”
    - hasParent $\sqsubseteq \neg$hasChild: “one cannot has parent one’s child”
- Concept (or role) equivalence: $X \equiv Y$ (abbreviation for $X \sqsubseteq Y$ and $Y \sqsubseteq X$: X and Y define the same concept/role)

Description Logics: Syntax

TBox axioms

The TBox may also contain properties about roles

- functional: (*func* R) (if $R(x, y)$ and $R(x, z)$ then $y = z$)
  - (*func* isMarriedTo)
- transitive: (*trans* R) (if $R(x, y)$ and $R(y, z)$ then $R(x, z)$)
  - (*trans* hasAncestor)
- symmetric: (*sym* R) (if $R(x, y)$ then $R(y, x)$)
  - (*sym* isMarriedTo)
- ...

Description Logics: Syntax

ABox assertions

The ABox contains

- Concept assertions: $C(a)$ (a is a member of C)
  - Father($BarackObama$): “Barack Obama is a father”
- Role assertions: $R(a, b)$ (a is related to b by R)
  - hasParent($SashaObama, BarackObama$): “Sasha Obama has parent Barack Obama”

Defining a Particular DL

To define a particular DL, we need to specify

- which concept and role constructors can be used
- what types of statements can appear in the TBox

Defining a Particular DL

To define a particular DL, we need to specify

- which concept and role constructors can be used
- what types of statements can appear in the TBox

For example, the $\mathcal{ALC}$ DL ("Attributive Concept Language with Complements") is defined as follows:

- if A is an atomic concept, then A is an $\mathcal{ALC}$ concept
- if C, D are $\mathcal{ALC}$ concepts and R is an atomic role, then the following are $\mathcal{ALC}$ concepts:
  - $C \sqcap D$ (conjunction)
  - $C \sqcup D$ (disjunction)
  - $\neg C$ (negation)
  - $\exists R.C$ (existential restriction)
  - $\forall R.C$ (universal restriction)
- an $\mathcal{ALC}$ TBox contains only concept inclusions

Note that $A \sqcap \neg A$ can be abbreviated by $\bot$ and $A \sqcup \neg A$ by $\top$.

Description Logics: Semantics

Interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

- $\Delta^{\mathcal{I}}$ is a non-empty set called domain
- $\cdot^{\mathcal{I}}$ is a function which associates
  - each individual a with an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - each atomic concept A with a unary relation $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - each atomic role R with a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

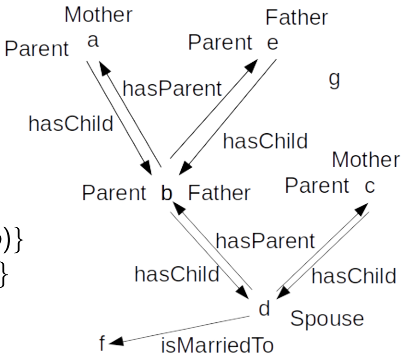
Description Logics: Semantics

Interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

- $\Delta^{\mathcal{I}}$ is a non-empty set called domain
- $\cdot^{\mathcal{I}}$ is a function which associates
  - each individual a with an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - each atomic concept A with a unary relation $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - each atomic role R with a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Example:

$\Delta^{\mathcal{I}} = \{a, b, c, d, e, f, g\}$
$alice^{\mathcal{I}} = a,\ bob^{\mathcal{I}} = b$
$\text{Mother}^{\mathcal{I}} = \{a, c\}$
$\text{Father}^{\mathcal{I}} = \{b, e\}$
$\text{Parent}^{\mathcal{I}} = \{a, b, c, e\}$
$\text{Spouse}^{\mathcal{I}} = \{d\}$
$\text{hasParent}^{\mathcal{I}} = \{(b, a), (b, e), (d, c), (d, b)\}$
$\text{hasChild}^{\mathcal{I}} = \{(a, b), (e, b), (c, d), (b, d)\}$
$\text{isMarriedTo}^{\mathcal{I}} = \{(d, f)\}$



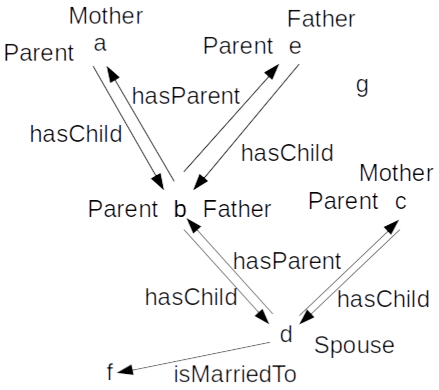
Description Logics: Semantics

The function $\cdot^{\mathcal{I}}$ is extended to complex concepts and roles to formalize the meaning of the constructors:

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
- $(\exists R.C)^{\mathcal{I}} = \{u \mid \text{ there exists } (u,v) \in R^{\mathcal{I}} \text{ such that } v \in C^{\mathcal{I}}\}$
- $(\forall R.C)^{\mathcal{I}} = \{u \mid \text{ for every } v, \text{ if } (u,v) \in R^{\mathcal{I}} \text{ then } v \in C^{\mathcal{I}}\}$
- $(\geq kR.C)^{\mathcal{I}} = \{u \mid \text{ there exists at least } k \ v \text{ such that } (u,v) \in R^{\mathcal{I}} \text{ and } v \in C^{\mathcal{I}}\}$
- $\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$
- $(R^{-})^{\mathcal{I}} = \{(u,v) \mid (v,u) \in R^{\mathcal{I}}\}$
- $(\neg R)^{\mathcal{I}} = (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$
- $(R \circ S)^{\mathcal{I}} = \{(u,v) \mid (u,w) \in R^{\mathcal{I}}, (w,v) \in S^{\mathcal{I}}\}$
- ...

Description Logics: Semantics

Example



$(\neg\mathsf{Parent})^{\mathcal{I}} = ?$

$(\exists \mathsf{hasParent}.\top)^{\mathcal{I}} = ?$

$(\mathsf{isMarriedTo}^{-})^{\mathcal{I}} = ?$

$(\mathsf{Spouse} \sqcup \mathsf{Mother})^{\mathcal{I}} = ?$

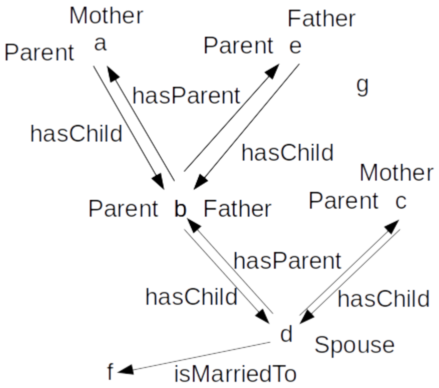
$(\forall \mathsf{hasChild}.\mathsf{Spouse})^{\mathcal{I}} = ?$

$((\forall \mathsf{hasChild}.\mathsf{Spouse}) \sqcap (\exists \mathsf{hasChild}.\top))^{\mathcal{I}} = ?$

$(\mathsf{Mother} \sqcap (\exists \mathsf{hasChild}.\exists \mathsf{hasChild}^{-}.\exists \mathsf{hasParent}.\mathsf{Father}))^{\mathcal{I}} = ?$

Description Logics: Semantics

Example



$(\neg\mathsf{Parent})^{\mathcal{I}} = \{d, f, g\}$

$(\exists\mathsf{hasParent}.\top)^{\mathcal{I}} = \{b, d\}$

$(\mathsf{isMarriedTo}^{-})^{\mathcal{I}} = \{(f, d)\}$

$(\mathsf{Spouse} \sqcup \mathsf{Mother})^{\mathcal{I}} = \{a, c, d\}$

$(\forall\mathsf{hasChild}.\mathsf{Spouse})^{\mathcal{I}} = \{b, c, d, f, g\}$

$((\forall\mathsf{hasChild}.\mathsf{Spouse}) \sqcap (\exists\mathsf{hasChild}.\top))^{\mathcal{I}} = \{b, c\}$

$(\mathsf{Mother} \sqcap (\exists\mathsf{hasChild}.\exists\mathsf{hasChild}^{-}.\exists\mathsf{hasParent}.\mathsf{Father}))^{\mathcal{I}} = \{c\}$

Description Logics: Semantics

Satisfaction of TBox axioms

- $\mathcal{I}$ satisfies a concept inclusion $C \sqsubseteq D$, written $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role inclusion $R \sqsubseteq S$, written $\mathcal{I} \models R \sqsubseteq S$, if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $(\mathit{func}\ R)$, written $\mathcal{I} \models (\mathit{func}\ R)$, if $R^{\mathcal{I}}$ is a functional relation
- ...

Description Logics: Semantics

Satisfaction of TBox axioms

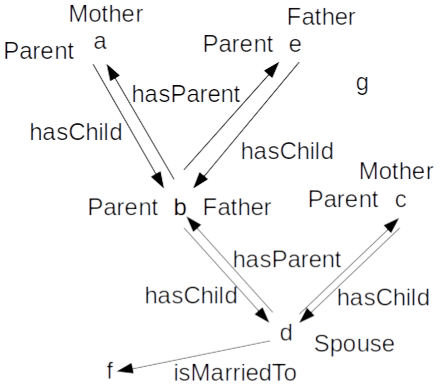
- $\mathcal{I}$ satisfies a concept inclusion $C \sqsubseteq D$, written $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a role inclusion $R \sqsubseteq S$, written $\mathcal{I} \models R \sqsubseteq S$, if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
- $\mathcal{I}$ satisfies $(\mathit{func}\ R)$, written $\mathcal{I} \models (\mathit{func}\ R)$, if $R^{\mathcal{I}}$ is a functional relation
- ...

Satisfaction of ABox assertions

- $\mathcal{I}$ satisfies
  - a concept assertion $C(a)$, written $\mathcal{I} \models C(a)$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$,
  - a role assertion $R(a, b)$, written $\mathcal{I} \models R(a, b)$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Description Logics: Semantics

Example



Assuming that $alice^{\mathcal{I}} = a$ and $bob^{\mathcal{I}} = b$:

$\mathcal{I} \models \text{Mother} \sqsubseteq \text{Parent}$?

$\mathcal{I} \models \exists\text{hasChild}.\top \sqsubseteq \exists\text{hasParent}.\top$?

$\mathcal{I} \models \text{Mother} \sqsubseteq \neg\text{Father}$?

$\mathcal{I} \models (\mathit{func}\ \text{hasChild})$?

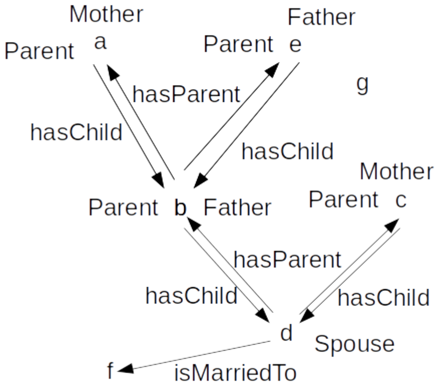
$\mathcal{I} \models \text{hasParent}(\mathit{bob}, \mathit{alice})$?

$\mathcal{I} \models \exists\text{hasChild}.(\text{Father} \sqcap \exists\text{hasChild}.\text{Spouse})(\mathit{alice})$?

$\mathcal{I} \models \forall\text{hasChild}.(\text{Father} \sqcap \forall\text{isMarriedTo}.\text{Spouse})(\mathit{alice})$?

Description Logics: Semantics

Example



Assuming that $alice^{\mathcal{I}} = a$ and $bob^{\mathcal{I}} = b$:

$\mathcal{I} \models \text{Mother} \sqsubseteq \text{Parent}$ ✓

$\mathcal{I} \models \exists\text{hasChild}.\top \sqsubseteq \exists\text{hasParent}.\top$ ✗

$\mathcal{I} \models \text{Mother} \sqsubseteq \neg\text{Father}$ ✓

$\mathcal{I} \models (\mathit{func}\ \text{hasChild})$ ✓

$\mathcal{I} \models \text{hasParent}(\mathit{bob}, \mathit{alice})$ ✓

$\mathcal{I} \models \exists\text{hasChild}.(\text{Father} \sqcap \exists\text{hasChild}.\text{Spouse})(\mathit{alice})$ ✓

$\mathcal{I} \models \forall\text{hasChild}.(\text{Father} \sqcap \forall\text{isMarriedTo}.\text{Spouse})(\mathit{alice})$ ✓

Description Logics: Semantics

Models

- $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it satisfies every axiom in $\mathcal{T}$
- $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies every assertion in $\mathcal{A}$
- $\mathcal{I}$ is a model of a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of $\mathcal{T}$ and $\mathcal{A}$
- Two KBs are equivalent if they have the same models

Description Logics: Semantics

Models

- $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it satisfies every axiom in $\mathcal{T}$
- $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies every assertion in $\mathcal{A}$
- $\mathcal{I}$ is a model of a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of $\mathcal{T}$ and $\mathcal{A}$
- Two KBs are equivalent if they have the same models

Satisfiability

- A KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, or consistent, if it has a model
- A concept C is satisfiable if there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$
- A concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$

Description Logics: Semantics

Models

- $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it satisfies every axiom in $\mathcal{T}$
- $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies every assertion in $\mathcal{A}$
- $\mathcal{I}$ is a model of a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of $\mathcal{T}$ and $\mathcal{A}$
- Two KBs are equivalent if they have the same models

Satisfiability

- A KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, or consistent, if it has a model
- A concept C is satisfiable if there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$
- A concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$

Entailment

- A TBox $\mathcal{T}$ entails an axiom α, written $\mathcal{T} \models \alpha$, if every model of $\mathcal{T}$ satisfies α
- A KB $\langle \mathcal{T}, \mathcal{A} \rangle$ entails an assertion α, written $\langle \mathcal{T}, \mathcal{A} \rangle \models \alpha$, if every model of $\langle \mathcal{T}, \mathcal{A} \rangle$ satisfies α

Description Logics: Semantics

Example

$$\begin{aligned}\mathcal{T} = \{\, & \mathsf{Female} \sqcap \exists \mathsf{hasChild}.\top \sqsubseteq \mathsf{Mother},\\ & \mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \exists \mathsf{hasChild}.\top,\\ & \mathsf{Parent} \equiv \exists \mathsf{hasChild}.\top,\\ & \top \sqsubseteq \exists \mathsf{hasChild}^{-}.\top \,\}\\ \mathcal{A} = \{\, & \mathsf{hasChild}(\mathit{alice}, \mathit{bob}),\ \mathsf{Female}(\mathit{alice}),\ \mathsf{Father}(\mathit{bob}) \,\}\end{aligned}$$

- $\mathcal{T} \models \mathsf{Father} \sqsubseteq \mathsf{Parent}$?
- $\mathcal{T} \models \mathsf{Mother} \sqsubseteq \mathsf{Parent}$?
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathsf{Mother}(\mathit{alice})$?
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathsf{Male}(\mathit{bob})$?
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \forall \mathsf{hasChild}.\mathsf{Male}(\mathit{alice})$?
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \exists \mathsf{hasChild}^{-}.\exists \mathsf{hasChild}^{-}.\top(\mathit{alice})$?

Description Logics: Semantics

Example

$$\begin{aligned}\mathcal{T} = \{\, & \mathsf{Female} \sqcap \exists \mathsf{hasChild}.\top \sqsubseteq \mathsf{Mother},\\ & \mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \exists \mathsf{hasChild}.\top,\\ & \mathsf{Parent} \equiv \exists \mathsf{hasChild}.\top,\\ & \top \sqsubseteq \exists \mathsf{hasChild}^{-}.\top \,\}\\ \mathcal{A} = \{\, & \mathsf{hasChild}(alice, bob),\ \mathsf{Female}(alice),\ \mathsf{Father}(bob)\,\}\end{aligned}$$

- $\mathcal{T} \models \mathsf{Father} \sqsubseteq \mathsf{Parent}$ ✓
- $\mathcal{T} \models \mathsf{Mother} \sqsubseteq \mathsf{Parent}$ ✗
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathsf{Mother}(alice)$ ✓
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathsf{Male}(bob)$ ✓
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \forall \mathsf{hasChild}.\mathsf{Male}(alice)$ ✗
- $\langle \mathcal{T}, \mathcal{A} \rangle \models \exists \mathsf{hasChild}^{-}.\exists \mathsf{hasChild}^{-}.\top(alice)$ ✓

Description Logics: Semantics
Example

$$\begin{aligned}\mathcal{T} = \{\, & \top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female},\\ & \exists \mathsf{friend}.(\mathsf{Female} \sqcap \exists \mathsf{loves}.\mathsf{Male}) \sqsubseteq A \,\}\end{aligned}$$

$$\begin{aligned}\mathcal{A} = \{\, & \mathsf{friend}(\mathit{john}, \mathit{susan}), && \mathsf{friend}(\mathit{john}, \mathit{andrea}),\\ & \mathsf{loves}(\mathit{susan}, \mathit{andrea}), && \mathsf{loves}(\mathit{andrea}, \mathit{bill}),\\ & \mathsf{Female}(\mathit{susan}), && \mathsf{Male}(\mathit{bill}) \,\}\end{aligned}$$

$$\langle \mathcal{T}, \mathcal{A} \rangle \models A(\mathit{john})\,?$$

Description Logics: Semantics

Example

$$\begin{aligned}\mathcal{T} = \{\, & \top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}, \\ & \exists \mathsf{friend}.(\mathsf{Female} \sqcap \exists \mathsf{loves}.\mathsf{Male}) \sqsubseteq A \,\}\end{aligned}$$

$$\begin{aligned}\mathcal{A} = \{\, & \mathsf{friend}(\mathit{john}, \mathit{susan}), && \mathsf{friend}(\mathit{john}, \mathit{andrea}), \\ & \mathsf{loves}(\mathit{susan}, \mathit{andrea}), && \mathsf{loves}(\mathit{andrea}, \mathit{bill}), \\ & \mathsf{Female}(\mathit{susan}), && \mathsf{Male}(\mathit{bill}) \,\}\end{aligned}$$

$$\langle \mathcal{T}, \mathcal{A} \rangle \models A(\mathit{john}) \checkmark$$

Exercises

Show the following statements:

- If $\mathcal{T} \subseteq \mathcal{T}'$, then $\mathcal{T} \models C \sqsubseteq D$ implies $\mathcal{T}' \models C \sqsubseteq D$.
- The following equivalence axioms are valid, i.e., are satisfied by every interpretation:
  - $\neg\top \equiv \bot$
  - $\neg(\neg C) \equiv C$
  - $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$
  - $\forall R.C \equiv \neg\exists R.\neg C$
  - $\neg(\exists R.C) \equiv \forall R.\neg C$
- If $\mathcal{T} \models C \sqsubseteq D$, then for every role R, $\mathcal{T} \models \exists R.C \sqsubseteq \exists R.D$ and $\mathcal{T} \models \forall R.C \sqsubseteq \forall R.D$.
- $\mathcal{T} \models \exists R^{-}.\top \sqsubseteq C$ if and only if $\mathcal{T} \models \top \sqsubseteq \forall R.C$.

Standard Reasoning Tasks

Especially useful to build and debug an ontology

- Concept satisfiability: Given a concept C and a TBox $\mathcal{T}$, decide whether C is satisfiable w.r.t. $\mathcal{T}$.
- KB satisfiability: Given a KB $\langle \mathcal{T}, \mathcal{A} \rangle$, decide whether $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable.
- Subsumption (or axiom entailment): Given a concept inclusion α (or in general an axiom α) and a TBox $\mathcal{T}$, decide whether $\mathcal{T} \models \alpha$.
- Classification: Given a TBox $\mathcal{T}$, decide for every pair of concept names A, B from $\mathcal{T}$, whether $\mathcal{T} \models A \sqsubseteq B$.
- Instance checking: Given a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ and a concept C, decide for every individual a from $\mathcal{A}$ whether $\langle \mathcal{T}, \mathcal{A} \rangle \models C(a)$.

Exercises: Reduction Between Standard Reasoning Tasks

Show the following statements:

- $\mathcal{T} \models C \sqsubseteq D$ iff $C \sqcap \neg D$ is not satisfiable w.r.t. $\mathcal{T}$.
- C is satisfiable w.r.t. $\mathcal{T}$ iff $\mathcal{T} \not\models C \sqsubseteq \bot$.
- C is satisfiable w.r.t. $\mathcal{T}$ iff $\langle \mathcal{T} \cup \{A \sqsubseteq C\}, \{A(a)\} \rangle$ is consistent.
- $\langle \mathcal{T}, \mathcal{A} \rangle \models C(a)$ iff $\langle \mathcal{T} \cup \{C \sqsubseteq \neg A\}, \mathcal{A} \cup \{A(a)\} \rangle$ is inconsistent.

Ontology-Mediated Query Answering

- Database-style queries, in particular conjunctive queries: $q(\vec{x}) = \exists \vec{y} \varphi$ where φ is a conjunction of atoms of the form $R(t, t')$ or $A(t)$ with t, t' variables from $\vec{x} \cup \vec{y}$ or constants.
- A tuple of constants $\vec{a}$ is an answer to $q(\vec{x})$ in an interpretation $\mathcal{I}$ if $\mathcal{I} \models q(\vec{a})$ where $q(\vec{a})$ is obtained by replacing answer variables from $\vec{x}$ by constants from $\vec{a}$.
- Ontology-mediated query answering: Find the certain answers to $q(\vec{x})$ over $\langle \mathcal{T}, \mathcal{A} \rangle$, i.e., those that hold in every model of $\langle \mathcal{T}, \mathcal{A} \rangle$.

Examples of Other Non-Standard Reasoning Tasks

- Axiom entailment explanation: Given a TBox $\mathcal{T}$ and an axiom α such that $\mathcal{T} \models \alpha$, find a minimal subset $\mathcal{T}'$ of $\mathcal{T}$ such that $\mathcal{T}' \models \alpha$.
- ABox repair: Given a satisfiable TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent, find a maximal subset $\mathcal{A}'$ of $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is satisfiable.
- ABox abduction: Given a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ and an assertion α such that $\langle \mathcal{T}, \mathcal{A} \rangle \not\models \alpha$, find a minimal set of assertions H such that $\langle \mathcal{T}, \mathcal{A} \cup H \rangle$ is consistent and $\langle \mathcal{T}, \mathcal{A} \cup H \rangle \models \alpha$.
- Module extraction: Given a signature Σ (set of concept and role names) and a TBox $\mathcal{T}$, extract a minimal subset (module) $\mathcal{T}_0$ from $\mathcal{T}$ such that $\mathcal{T}_0$ preserves all logical entailments that can be expressed in the description logic of $\mathcal{T}$ using only terms in Σ.
- ...

Expressivity vs Efficiency

- The complexity of the reasoning tasks depends on the description logic language considered.
- A crucial design principle in description logics is to establish a favorable trade-off between expressive power (what the language can express) and computational complexity.
- For example standard reasoning tasks are
    - in PTime for the description logic $\mathcal{EL}$: concept inclusions $C \sqsubseteq D$ where $C, D := \top \mid A \mid C \sqcap D \mid \exists R.C$
    - ExpTime-complete for $\mathcal{ELI} = \mathcal{EL}$ +inverse roles: $C, D := \top \mid A \mid C \sqcap D \mid \exists R.C \mid \exists R^{-}.C$
    - in PTime for DL-Lite: concept inclusions $B \sqsubseteq C$ where $C := B \mid \neg B,\ B := A \mid \exists S,\ S := R \mid R^{-}$

Relationship with First-Order Logic

DL KBs can be translated into first-order logic (FOL):

- atomic concepts and roles are unary and binary predicates
- complex concepts are FOL formula with one free variable
  - $\mathrm{Female} \sqcap \exists \mathrm{hasChild}.\top$ $\quad$ $\mathrm{Female}(x) \wedge \exists y \mathrm{hasChild}(x, y)$
- TBox and ABox axioms are FOL sentences
  - $\exists \mathrm{hasChild}.\top \sqsubseteq \mathrm{Parent}$ $\quad$ $\forall x (\exists y \mathrm{hasChild}(x, y) \Rightarrow \mathrm{Parent}(x))$

Relationship with First-Order Logic

Example: Translation of an $\mathcal{ALC}$ TBox

Concept C is translated into FOL formula with one free variable $\pi_x(C)$ inductively defined as follows

- $\pi_x(A) = A(x)$ for A atomic concept
- $\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$
- $\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$
- $\pi_x(\neg C) = \neg \pi_x(C)$
- $\pi_x(\exists R.C) = \exists y(R(x,y) \wedge \pi_y(C))$, y different from x
- $\pi_x(\forall R.C) = \forall y(R(x,y) \Rightarrow \pi_y(C))$, y different from x

Concept inclusion $C \sqsubseteq D$ is translated into FOL sentence $\pi(C \sqsubseteq D) = \forall x(\pi_x(C) \Rightarrow \pi_x(D))$

Relationship between DLs and OWL

OWL (Web Ontology Language): W3C standard

- "language to represent rich and complex knowledge about things, groups of things, and relations between things"
- Ontology language based on Description Logic

There are several variants of OWL

- OWL Full: no restrictions on the use of OWL features $\rightarrow$ undecidable !
- OWL 2 DL (decidable fragment of OWL Full) corresponds to the DL $\mathcal{SROIQ}(D)$
  - $\mathcal{S}$ stands for $\mathcal{ALC}$ extended with transitive roles ($(trans\ R)$)
  - $\mathcal{R}$: regular role hierarchies (role inclusions with some suitable acyclicity conditions) ($R_1 \sqsubseteq R_2$)
  - $\mathcal{O}$: nominals (concept $\{o\}$ where o is an individual)
  - $\mathcal{I}$: inverse roles (R^-)
  - $\mathcal{Q}$: qualified number restrictions ($\leq nR.C$, $\geq nR.C$)
  - (D): data types

Relationship between DLs and OWL

OWL 2 defines three profiles targeted towards a specific uses

- "The OWL 2 EL profile is particularly suitable for applications employing ontologies that define very large numbers of classes and/or properties, [...] and for which ontology consistency, class expression subsumption, and instance checking can be decided in polynomial time."
- "The OWL 2 QL profile is designed so that data (assertions) that is stored in a standard relational database system can be queried through an ontology via a simple rewriting mechanism"
- "The OWL 2 RL profile [...] is amenable to implementation using rule-based technologies"
- OWL 2 EL is based on the DL $\mathcal{EL}$
- OWL 2 QL is based on the DL DL-Lite
- OWL 2 RL was inspired by Description Logic Programs

`https://www.w3.org/TR/owl2-profiles`

Relationship between DLs and OWL

OWL adopts different terminology and syntax(es) than DLs

- OWL classes are concepts in DLs
- OWL (object) properties are roles in DLs
- OWL have many syntaxes that serve different purposes
  - functional syntax

    ```
    SubClassOf(:Parent ObjectSomeValuesFrom(:hasChild owl:Thing))
    ```

  - Manchester syntax

    ```
    Class: Parent
        SubClassOf:
            hasChild some owl:Thing
    ```

  - OWL/XML syntax

    ```
        <SubClassOf>
            <Class IRI="#Parent"/>
            <ObjectSomeValuesFrom>
                <ObjectProperty IRI="#hasChild"/>
                <Class abbreviatedIRI="owl:Thing"/>
            </ObjectSomeValuesFrom>
        </SubClassOf>
    ```

  - ...

Relationship between DLs and OWL

OWL expressions (functional syntax)	DL counterparts
Thing	$\top$
Nothing	$\bot$
ObjectIntersectionOf($C_1 \ \ldots \ C_n$)	$C_1 \sqcap \cdots \sqcap C_n$
ObjectUnionOf($C_1 \ \ldots \ C_n$)	$C_1 \sqcup \cdots \sqcup C_n$
ObjectComplementOf(C)	$\neg C$
ObjectSomeValuesFrom(R C)	$\exists R.C$
ObjectAllValuesFrom(R C)	$\forall R.C$
ObjectOneOf($a_1 \ \ldots \ a_n$)	$\{a_1\} \sqcup \cdots \sqcup \{a_n\}$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$
EquivalentClasses($C_1 \ \ldots \ C_n$)	$C_1 \equiv \cdots \equiv C_n$
DisjointClasses($C_1 \ \ldots \ C_n$)	$C_i \sqsubseteq \neg C_j$ $(1 \leq i \neq j \leq n)$
SubObjectPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$
InverseObjectPropertyOf(R_1 R_2)	$R_1 \equiv R_2^-$
ClassAssertion(C a)	$C(a)$
ObjectPropertyAssertion(R a b)	$R(a, b)$

...

Ontology Engineering

Real-world ontologies

- can be huge
- often represent knowledge that only domain experts have
- are usually developed by many people
- need to evolve

$\Longrightarrow$ difficult to build and maintain

Ontology Engineering

Real-world ontologies

- can be huge
- often represent knowledge that only domain experts have
- are usually developed by many people
- need to evolve

$\Longrightarrow$ difficult to build and maintain

Ontology engineering: methodologies for building, maintaining or debugging an ontology

- design patterns
- reuse of existing ontologies
  - modules
  - ontologies alignment
- automated knowledge acquisition
- debugging with reasoner and explanations

Mostly out of the scope of this course

Ontology Editors and Reasoners

A lot of reasoners, tools and libraries for developing ontologies have been implemented. Reasoners support various ontology languages and reasoning tasks, and implement various algorithms.

- List of DL reasoners:
  `http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/`
- List of OWL implementations (reasoners, editors, API...):
  `http://www.w3.org/2001/sw/wiki/OWL/Implementations`
- A toolkit for existential rules:
  `https://graphik-team.github.io/graal/`

Outline of the Course

- Algorithms for basic reasoning tasks in description logics (ontology building and debugging)
  - in $\mathcal{ALC}$
  - in $\mathcal{EL}$
- Ontology-mediated query answering
  - chase (materialization)
  - query rewriting
  - bounded treewidth sets of rules
- Some current research topics related to OMQA, for example among the following
  - finite controllability
  - aggregation queries
  - inconsistency-handling
  - descriptive complexity
  - ...

References
Knowledge graphs

- W3C standards and documentation:
    - RDF: https://www.w3.org/TR/rdf11-concepts/
    - SPARQL: https://www.w3.org/TR/rdf-sparql-query/
    - OWL: https://www.w3.org/TR/owl2-overview
    - OWL 2 profiles: https://www.w3.org/TR/owl2-profiles/
- Hogan (2014): Linked Data and the Semantic Web Standards (book chapter: https://aidanhogan.com/docs/ldmgmt_semantic_web_linked_data.pdf)
- Hogan: website associated to book The Web of Data (https://aidanhogan.com/webofdatabook/)
- Krötzsch (2019): course on Knowledge Graphs (lecture: https://iccl.inf.tu-dresden.de/web/Knowledge_Graphs_(WS2019/20)/en)

References

Description logics

- Baader, Calvanese, McGuinness, Nardi, Patel-Schneider (2003): The Description Logic Handbook: Theory, Implementation, and Applications (book, can be found online)
- Bienvenu (2022): Ontologies & Description Logics (lecture: `https://www.labri.fr/perso/meghyn/teaching/lola-2022/1-lola-intro-ontologies.pdf`)
- Baader (2019): course on Description Logics (lecture: `https://tu-dresden.de/ing/informatik/thi/lat/studium/lehrveranstaltungen/sommersemester-2019/description-logic`)