TECHNISCHE UNIVERSITÄT DRESDEN

Stefan Borgwardt
Institute of Theoretical Computer Science, Chair of Automata Theory

Logic-Based Ontology Engineering

Part 1: Introduction to Ontology Engineering, OWL 2, and Description Logics

Organizational Matters

Web page:
`https://tu-dresden.de/ing/informatik/thi/lat/studium/vorlesungen/sommersemester-2018/logic-based-ontology-engineering`

- 2 SWS lectures (Dr.-Ing. Stefan Borgwardt) and
  2 SWS tutorials (PD Anni-Yasmin Turhan)
- Wednesdays 9:20–10:50 (2. DS) and 13:00–14:30 (4. DS)
- not on 23 May and 6 June
- for the exact schedule see the web page
- slides, exercise sheets on the web page
- some exercises use the ontology editor Protégé
- exam regulations: see module descriptions
- MCL-ILS: written exam together with Deduction Systems
- feel free to ask questions

Goals

After this lecture, you will know more about

- what an (OWL) ontology is
- how to create an ontology
- how to link ontologies
- how to debug ontologies
- logic-based techniques for these tasks

What do you already know about

- first-order logic?
- description logics?
- OWL?
- the Semantic Web?
- ontology tools?

Outline

Part 1: Introduction

1.1 Introduction to Ontologies

1.2 Ontology Engineering

1.3 OWL 2 and Description Logics

Part 2: Ontology Creation

Part 3: Ontology Integration

Part 4: Ontology Maintenance

1.1 Introduction to Ontologies

Ontology in Philosophy

> “The branch of metaphysics concerned with the nature and relations of being.” (Oxford Dictionaries)

- What kinds of things exist?
- What categories of things, similarities, differences, exist?
- What is the identity of an object? When does it start and end to exist?
- How can things be related to each other?
- Ontology does not talk about specific objects.

Computer Science adopted this notion via Mathematical Logic, Knowledge Representation and Reasoning, Artificial Intelligence.

Ontology in Computer Science

Computer scientists are more pragmatic:

> "For AI systems, what 'exists' is exactly that which can be represented." (Gruber, 1993)

> "An ontology is a formal, explicit specification of a shared conceptualization." (Staab, Studer, 2009)

- An ontology represents an abstract, simplified view of the relevant entities (objects, concepts, and relations) that exist in the domain of interest.
- It is a computational artifact designed for a specific purpose.
- In contrast to Ontology in Philosophy, data, i.e., knowledge about specific objects, plays a central role.
- A shared, formal language allows for automated processing, reuse and integration.

Conceptualization

Objects / Individuals:

> Stefan Borgwardt, LBOE, APB/E005, TU Dresden

Concepts / Classes / Categories:

> Person, Lecture, Room, Building, University

Relations / Properties / Attributes:

> attends, gives, is part of, is a, belongs to, is employed by

Axioms / Constraints:

> APB/E005 is part of APB, which belongs to TU Dresden.
> Every lecture is a course. Every room is part of a building.
> Every lecture is a given by a person employed by a university.

Shared Conceptualization

An ontology is founded on a shared understanding of the domain terms: objects, concepts, and relations should be interpreted in the same way by every user.

> What is a company? Is it a person? Is a university a company?
>
> This depends on the context: application domain, purpose of the ontology, legal system in which it is used, ...

- Such information can be part of an informal consensus between the involved parties: domain experts, ontology developers, end users.
- To allow automated processing of the ontology, however, also the computer needs access to this information.
- This is where the formal specification comes in.

Specification

There are many specification languages, from informal to formal ones:

- Lists of terms, glossaries
- Folksonomies, collaborative tagging
- Thesauri, informal hierarchies
- XML Document Type Description (DTD)
- Database schemas, XML Schema
- Entity Relationship Model (ERM), Unified Modeling Language (UML)
- Resource Description Framework Schema (RDFS), Formal taxonomies
- Logic Programming, Frame logic (F-logic)
- Description logics, Web Ontology Language (OWL)
- Modal logics, First-order logic
- Higher-order logics, Common Logic

Formal Specification

Most popular are relational specification languages, based on first-order logic (but the syntax and semantics may differ).

- objects are constants
- concepts are unary predicates
- relations are n-ary predicates, $n \geq 2$

$$\mathsf{partOf}(\mathsf{APB/E005}, \mathsf{APB})$$
$$\mathsf{belongsTo}(\mathsf{APB}, \mathsf{TU\ Dresden})$$
$$\forall x.\mathsf{Lecture}(x) \rightarrow \mathsf{Course}(x)$$
$$\forall x.\mathsf{Room}(x) \rightarrow \exists y.\mathsf{partOf}(x, y) \land \mathsf{Building}(y)$$

An ontology represents a set of possible worlds (a.k.a. models).

Our knowledge is usually incomplete, i.e., we don't know which of these models describes (an abstract view of) the real world.

On what level the abstraction takes place depends on the application.

Using Ontologies

The power of ontologies lies in automated inference mechanisms.

- Concepts from an ontology can be used to annotate data (databases, web pages, text), to make it easier to search and browse information.

> In a university database, a search for "**Course**" can automatically return all lectures, seminars, etc.

- Structured queries can retrieve complex information, similar to SQL.

> $$\mathsf{SeminarRoom}(x) \land \mathsf{partOf}(x, \mathsf{APB}) \land \exists y.\mathsf{Course}(y) \land \mathsf{takesPlace}(y, x, \mathsf{Wed\ 2.DS})$$

- Formal properties of the ontology can hint at modeling errors in the domain knowledge.

> If the ontology is inconsistent, then either the ontology does not correctly reflect the domain knowledge, or the knowledge itself is faulty.

Application Areas of Ontologies

Research:

- Artificial Intelligence
- Databases
- Natural Language Processing
- Software Engineering
- Biology, Medicine

Industry:

- E-Commerce
- Semantic Web
- Library Systems
- Geographic Information Systems

Popular Ontologies

Upper ontologies:

- WordNet
- Basic Formal Ontology
- Cyc, SUMO, DOLCE

Core ontologies:

- Common Core Ontologies (Time Ontology, Agent Ontology, . . .)
- Dublin Core (metadata, e.g., for library systems)
- FOAF Core (people)

Domain ontologies:

- NCI Thesaurus, UMLS Metathesaurus (medicine)
- GoodRelations (e-commerce)

Application ontologies:

- Gene Ontology (biological processes, gene functions, interactions)
- ICD, SNOMED CT (medical billing, statistics)

WordNet Ontology

- Developed at Princeton University, Departments of Psychology / Computer Science, since the 1980s
- 155.000 English words are grouped into 117.000 sets of synonyms ("synsets") according to their meanings
- Concepts: **Word**, **WordSense**, **Synset**, . . .
- Relations: **word**, **containsWordSense**, **antonymOf**, **hyponymOf**, . . .

WordNet Axioms

"'funny' can be have the sense 'humorous', as opposed to 'humorless'."

word(funny-sense-1, funny)

containsWordSense(synset-humorous-1, funny-sense-1)

containsWordSense(synset-humorous-1, amusing-sense-2)

gloss(synset-humorous-1, "provoking laughter")

antonymOf(funny-sense-1, humorless-sense-1)

word(funny-sense-2, funny)

containsWordSense(synset-strange-1, funny-sense-2)

containsWordSense(synset-strange-1, odd-sense-4)

gloss(synset-strange-1, "deviating from the usual or expected")

antonymOf(funny-sense-2, familiar-sense-2)

Basic Formal Ontology (BFO)

- Developed by a community of researchers, started around 2003
- Contains definitions of high-level classes (35) and relations.
- Concepts: **Occurrent**, **Continuant**, **MaterialEntity**, **TemporalRegion**, . . .
- Relations: **continuantPartOf**, **hasContinuantPart**, **existsAt**, . . .

BFO Axioms

"Every material entity exists at some time."

$\forall x.\text{MaterialEntity}(x) \rightarrow \exists t.\text{TemporalRegion}(t) \land \text{existsAt}(x, t)$

"The parts of a material entity must be material entities."

$\forall x, y.\text{MaterialEntity}(x) \land \text{hasContinuantPart}(x, y) \rightarrow \text{MaterialEntity}(y)$

"A process is an occurrent that has temporal proper parts and that specifically depends on some material entity at some time."

$$\forall x.\text{Process}(x) \leftrightarrow \big(\text{Occurent}(x) \land \exists y.\text{properTemporalPartOf}(y, x) \land \exists z, t.\text{MaterialEntity}(z) \land \text{specificallyDependsOnAt}(x, z, t)\big)$$

Common Core Ontologies

- Developed by non-profit R&D company CUBRC, since 2010
- Extensions of BFO to more specialized domains

Time Ontology:

"A day is a temporal interval. An hour occurs during a day. The relation 'during' is transitive."

$\forall x.\mathsf{Day}(x) \rightarrow \mathsf{OneDimensionalTemporalRegion}(x)$

$\forall x.\mathsf{Hour}(x) \rightarrow \exists y.\mathsf{intervalDuring}(x, y) \land \mathsf{Day}(y)$

$\forall x, y, z.\mathsf{intervalDuring}(x, y) \land \mathsf{intervalDuring}(y, z) \rightarrow \mathsf{intervalDuring}(x, z)$

Common Core Ontologies

- Developed by non-profit R&D company CUBRC, since 2010
- Extensions of BFO to more specialized domains

Agent Ontology:

> "An agent is an organization or person that acts in some process. A group of agents consists only of agents, and contains at least one agent."
>
> $$\forall x.\mathsf{Agent}(x) \leftrightarrow \big((\mathsf{Organization}(x) \lor \mathsf{Person}(x)) \land \exists y.\mathsf{agentIn}(x,y) \land \mathsf{Process}(y)\big)$$
>
> $$\forall x.\mathsf{GroupOfAgents}(x) \to \mathsf{ObjectAggregate}(x) \land (\exists y.\mathsf{hasPart}(x,y) \land \mathsf{Agent}(y)) \land (\forall z.\mathsf{hasPart}(x,z) \to \mathsf{Agent}(z))$$

NCI Thesaurus

- Medical Terminology developed by the US National Cancer Institute (NCI)
- Contains 133.000 concepts and 100 relations

"A cellular process is a biological process that takes place in a cell."

$$\forall x.\mathsf{CellularProcess}(x) \rightarrow \mathsf{BiologicalProcess}(x) \wedge \exists y.\mathsf{hasAssociatedLocation}(x, y) \wedge \mathsf{Cell}(y)$$

"The concepts 'gene' and 'organism' are disjoint."

$$\forall x.\mathsf{Gene}(x) \rightarrow \neg\mathsf{Organism}(x)$$

"A cancer gene is a gene that plays a role in the formation of a cancer."

$$\forall x.\mathsf{CancerGene}(x) \rightarrow \mathsf{Gene}(x) \wedge \exists y.\mathsf{playsRoleIn}(x, y) \wedge \mathsf{Tumorigenesis}(y)$$

Gene Ontology (GO)

- Developed by the Gene Ontology consortium since 1998
- Knowledge about biological processes and their interactions
- Contains 63.000 concepts and 300 relations

$$\forall x.\mathsf{DNAMetabolicProcess}(x) \leftrightarrow \big(\mathsf{MetabolicProcess}(x) \land \exists y.\mathsf{hasParticipant}(x,y) \land \mathsf{DNA}(y)\big)$$

$$\forall x.\mathsf{MAPKCascade}(x) \rightarrow \mathsf{MetabolicProcess}(x) \land \exists y.\mathsf{partOf}(x,y) \land \mathsf{CellCommunication}(y)$$

GO Annotations

- Associate concrete genes to their biological functions, as supported by the current biological knowledge
- Not formally a part of GO, but can be formulated as axioms in the vocabulary of GO

> **annotatedWith**(**A-kinase anchor protein 9**, **x**, **Homo sapiens**,
> **R-HSA-5673001**, **2017/11/18**)
> **MAPK cascade**(**x**)

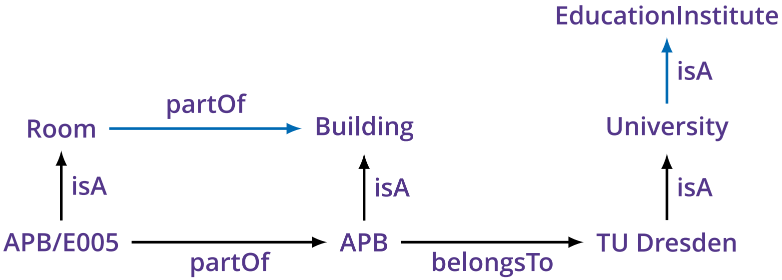
Biomedical Ontology Repositories

In the biomedical area, there is a large number of specialized ontologies for many disciplines.

- BioPortal: `https://bioportal.bioontology.org/`
- The OBO Foundry: `http://www.obofoundry.org/`

Ontologies as Graphs of Knowledge

The term “Knowledge Graph” is often used when talking about ontologies, but is not quite the same. Such a graph represents objects and concepts as nodes, and (binary) relations as edges in a graph.

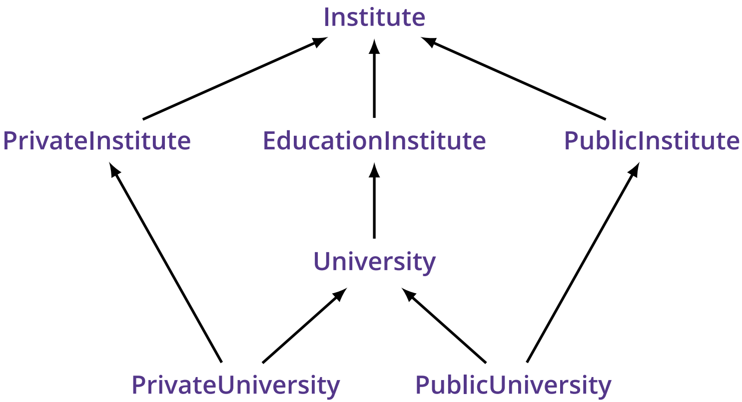


Drawbacks:

- Difficult to represent $\forall x.\text{Room}(x) \rightarrow \exists y.\text{partOf}(x, y) \wedge \text{Building}(y)$.
- Difficult to distinguish objects from concepts.

The Concept Hierarchy (a.k.a. Taxonomy)

Abstracting even further, an ontology is reduced to a hierarchy of concepts, which is a directed acyclic graph. This is the backbone of the ontology.



Outline

Part 1: Introduction

1.1 Introduction to Ontologies

1.2 Ontology Engineering

1.3 OWL 2 and Description Logics

Part 2: Ontology Creation

Part 3: Ontology Integration

Part 4: Ontology Maintenance

1.2 Ontology Engineering

Challenges

- How to build ontologies with 100.000+ concepts?
- How to make sure that every user understands the concepts in the same way?
- How to link ontologies together?

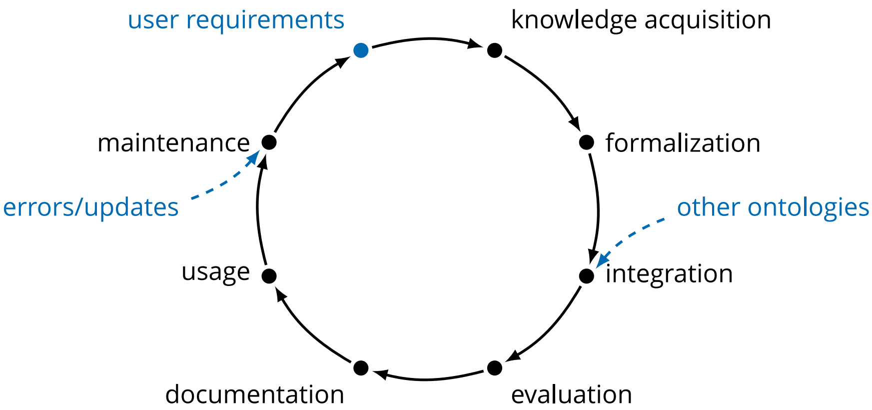
> Both the NCI Thesaurus and GO contain a concept called CellularProcess, but they are different entities.

- How to repair ontologies when they are faulty?

> An old version of SNOMED CT implied that every AmputationOfFinger is an AmputationOfHand, via a combination of 6 out of 350,000+ axioms.

- How to ensure that an ontology stays up-to-date?
- How to add new concepts/axioms to an ontology without affecting the old inferences?
- How to display large ontologies to the user?

The Ontology Life Cycle



Logic-Based Ontology Engineering

- Ontology engineering methods support knowledge engineers throughout the ontology life cycle.
- Logic-based techniques can automate some tasks.
- In this lecture, we will discuss some techniques for the following tasks:
  - ontology creation (from user requirements to a formalization)
  - ontology integration (linking to other ontologies)
  - ontology maintenance (handling errors and updates)

  ... based on OWL 2 and Description Logics.
- There are many other approaches with different advantages and drawbacks.
- Creating and maintaining ontologies is similar to large software engineering projects. We will not discuss project management (feasibility analysis, scheduling, risk management, etc.) in this lecture.

Ontology Editors

- Protégé https://protege.stanford.edu/
- Vitro https://github.com/vivo-project/Vitro/
- TopBraid Composer https://www.topquadrant.com/tools/
- OntoStudio http://www.semafora-systems.com/
- NeOn toolkit http://neon-toolkit.org/
- SWOOP https://github.com/ronwalf/swoop/

Plugins for Protégé:

- Ontograf https://github.com/protegeproject/ontograf/
- VOWL http://vowl.visualdataweb.org/
- OWLAx https://github.com/md-k-sarker/OWLAx
- DL-Learner
https://github.com/SmartDataAnalytics/DL-Learner-Protege-Plugin

Outline

Part 1: Introduction

1.1 Introduction to Ontologies

1.2 Ontology Engineering

1.3 OWL 2 and Description Logics

Part 2: Ontology Creation

Part 3: Ontology Integration

Part 4: Ontology Maintenance

1.3 OWL 2 and Description Logics

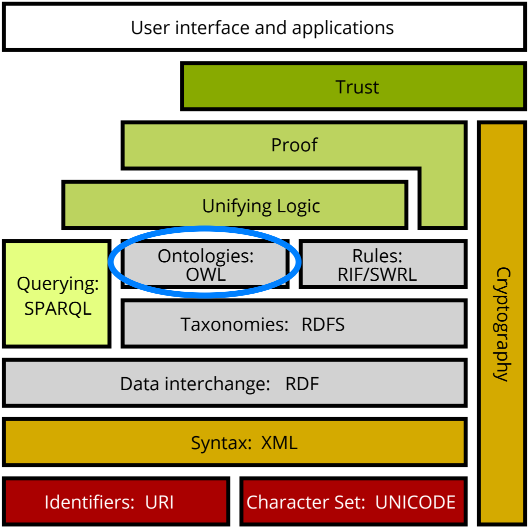
Web Ontology Language (OWL)

OWL is a World Wide Web Consortium (W3C) Recommendation and one of the most successful ontology languages.

The OWL 2 Direct Semantics (a.k.a. OWL 2 DL) is given by description logics (DLs), which are decidable fragments of first-order logic.

`http://www.w3.org/TR/owl2-overview/`

OWL in the Semantic Web



History of OWL

1956 Semantic Networks

1992 Description Logics

2001 DAML+OIL

2004 RDF, RDF/S

2004 OWL 1, OWL Lite, OWL DL, OWL Full

2008 (OWL 1.1)

2009/2012 OWL 2, Profiles: OWL 2 QL, OWL 2 RL, OWL 2 EL

> We will cover only the main features of OWL 2 here.

Syntaxes

Functional-Style Syntax

```
SubClassOf( Lecture Course )
```

RDF/XML Syntax

```
<owl:Class rdf:about="Lecture">
  <rdfs:subClassOf rdf:resource="Course">
</owl:Class>
```

OWL/XML Syntax

```
<SubClassOf>
  <Class IRI="Lecture">
  <Class IRI="Course">
</SubClassOf>
```

Syntaxes

Turtle Syntax

> **Lecture** rdfs:subClassOf **Course**

Manchester Syntax (used by Protégé)

> **Class: Lecture**
> **SubClassOf: Course**

(DL Syntax)

> $\mathsf{Lecture} \sqsubseteq \mathsf{Course}$

(FOL Syntax)

> $\forall x.\mathsf{Lecture}(x) \to \mathsf{Course}(x)$

Entity Declarations

Every entity of an OWL ontology must be declared to be of a certain type:

Individual: APB/E005

Class: Room

ObjectProperty: belongsTo

In description logics, these are represented by the following disjoint sets:

individual names: $\mathbf{I} = \{\text{APB/E005}, \dots\}$

concept names: $\mathbf{C} = \{\text{Room}, \dots\}$

role names: $\mathbf{R} = \{\text{belongsTo}, \dots\}$

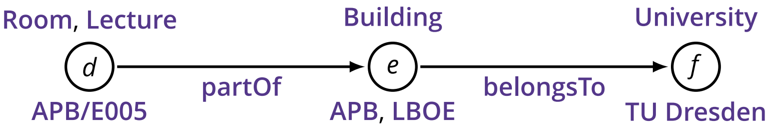
Together, these sets form the vocabulary of the ontology.

Interpretations

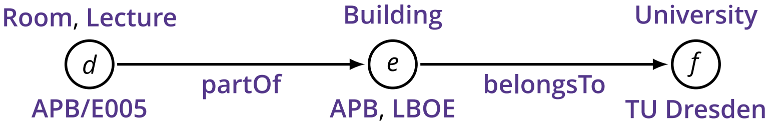
A DL interpretation is a tuple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where

- $\Delta^\mathcal{I}$ is a non-empty set, called the domain of $\mathcal{I}$,
- $\cdot^\mathcal{I}$ is an interpretation function that assigns meanings to names:
  - each $a \in \mathbf{I}$ is interpreted as an element $a^\mathcal{I} \in \Delta^\mathcal{I}$,
  - each $A \in \mathbf{C}$ is interpreted as a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$,
  - each $r \in \mathbf{R}$ is interpreted as a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$.

Interpretations represent possible worlds:



Interpretations: Example



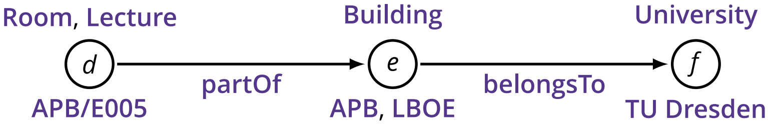
$\mathbf{I} = \{$APB/E005, APB, LBOE, TU Dresden$\}$

$\mathbf{C} = \{$Room, Lecture, Building, University, Company$\}$

$\mathbf{R} = \{$partOf, belongsTo$\}$

$\Delta^{\mathcal{I}} = \{d, e, f\}$	$\text{Room}^{\mathcal{I}} = \{d\}$	$\text{partOf}^{\mathcal{I}} = \{(d, e)\}$
$\text{APB/E005}^{\mathcal{I}} = d$	$\text{Lecture}^{\mathcal{I}} = \{d\}$	$\text{belongsTo}^{\mathcal{I}} = \{(e, f)\}$
$\text{APB}^{\mathcal{I}} = e$	$\text{Building}^{\mathcal{I}} = \{e\}$	
$\text{LBOE}^{\mathcal{I}} = e$	$\text{University}^{\mathcal{I}} = \{f\}$	
$\text{TU Dresden}^{\mathcal{I}} = f$	$\text{Company}^{\mathcal{I}} = \emptyset$	

Interpretations: Example



f is called **belongsTo**-successor / **belongsTo**-filler of e

e is called **belongsTo**-predecessor of f

The purpose of the ontology's axioms is to specify which interpretations are permitted, e.g., by stating that rooms cannot be lectures.

Complex Expressions

Before we can define axioms, we need to introduce more complex expressions built from classes and object properties.

Class expressions are interpreted as sets, and object property expressions are interpreted as binary relations.

An object property expression is either an object property or an inverse object property of an object property r:

Syntax:	**inverse** r
DL syntax:	r^-
DL name:	inverse role
Semantics:	$(r^-)^\mathcal{I} = \{(d,e) \mid (e,d) \in r^\mathcal{I}\}$

inverse belongsTo

Basic Class Expressions

Apart from declared classes, OWL 2 contains the following built-in classes:

Syntax:	owl:Thing	owl:Nothing
DL syntax:	$\top$	$\bot$
DL name:	top concept	bottom concept
Semantics:	$\Delta^{\mathcal{I}}$	$\emptyset$

All classes are class expressions. Given two class expressions C, D, the following are also class expressions:

Name:	conjunction	disjunction	negation
Syntax:	C **and** D	C **or** D	**not** C
DL syntax:	$C \sqcap D$	$C \sqcup D$	$\neg C$
Semantics:	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$

Room **and** owl:Thing   **not** Building

DL Notation

In Description Logics, different terms are used:

(named) object property	$\rightsquigarrow$	role name
object property (expression)	$\rightsquigarrow$	role
(named) class	$\rightsquigarrow$	concept name
class (expression)	$\rightsquigarrow$	concept (description)

Class Expressions: Object Property Restrictions

Class expressions can define restrictions on outgoing object properties, i.e., restrict the classes of object property successors.

If C is a class expression and r is an object property expression, then the following are also class expressions:

Name:	existential restriction	value restriction
Syntax:	r **some** C	r **only** C
DL syntax:	$\exists r.C$	$\forall r.C$
Semantics:	$\{x \mid \exists y.(x,y) \in r^{\mathcal{I}} \land y \in C^{\mathcal{I}}\}$	$\{x \mid \forall y.(x,y) \in r^{\mathcal{I}} \to y \in C^{\mathcal{I}}\}$

partOf **some** Building (**inverse** partOf) **only** MaterialEntity

Class Expressions: Cardinality Restrictions

Cardinality restrictions (also called number restrictions) can restrict the number of outgoing object property connections.

If C is a class expression, r is an object property expression, and n is a natural number, then the following are also class expressions:

Name:	at-least restriction	at-most restriction
Syntax:	r **min** n C	r **max** n C
DL syntax:	$\geq n\, r.C$	$\leq n\, r.C$
Semantics:	$\{x \mid \#\{y \in C^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \geq n\}$	$\{x \mid \#\{y \in C^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \leq n\}$

Student **and** (attends **min** 2 Lecture) belongsTo **max** 1 owl:Thing

Class Expressions: Nominals and Self Restrictions

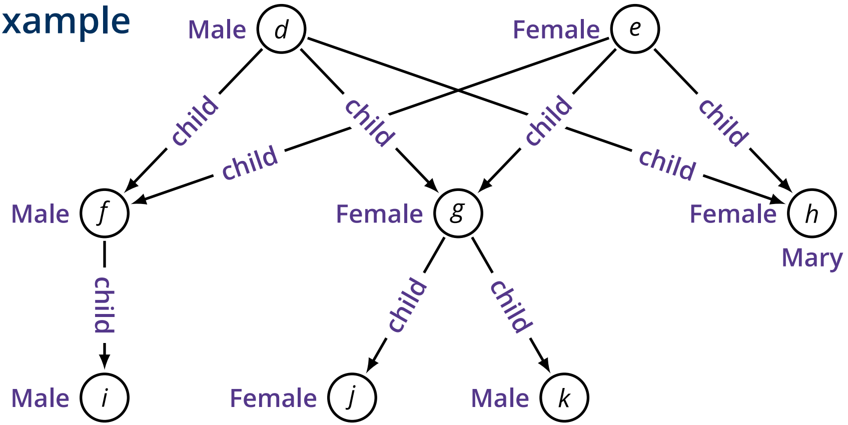
Given an individual a and an object property expression r, the following are also class expressions:

Name:	nominal	self restriction
Syntax:	**{**a**}**	r **Self**
DL syntax:	$\{a\}$	$\exists r.\mathrm{Self}$
Semantics:	$\{a^{\mathcal{I}}\}$	$\{x \mid (x,x) \in r^{\mathcal{I}}\}$

- "r **some {**a**}**" can also be written as "r **value** a".
- "**{**a_1**} or** . . . **or {**a_n**}**" can also be written as "**{**a_1**,**. . .**,**a_n**}**".

partOf **some {**APB**}**   partOf **value** APB   loves **Self**

Example



$(\exists \text{child}.\top)^{\mathcal{I}} = \{d, e, f, g\}$

$(\text{Female} \sqcap \exists \text{child}.\top)^{\mathcal{I}} = \{e, g\}$

$(\geq 2\, \text{child}.\text{Female})^{\mathcal{I}} = \{d, e\}$

$(\exists \text{child}.\text{Self})^{\mathcal{I}} = \emptyset$

$(\text{child}^{-})^{\mathcal{I}} = \{(f, d), (f, e), (i, f), \ldots\}$

$(\text{Male} \sqcap \exists \text{child}^{-}.\exists \text{child}.\text{Female})^{\mathcal{I}} = \{f, k\}$

$(\neg \exists \text{child}^{-}.\top)^{\mathcal{I}} = \{d, e\}$

$(\exists \text{child}.\{\text{Mary}\})^{\mathcal{I}} = \{d, e\}$

Class Axioms

We can use class and object property expressions to formulate axioms.

An axiom α defines a set of models, which are interpretations $\mathcal{I}$ that satisfy the axiom, written $\mathcal{I} \models \alpha$.

If C and D are class expressions, then the following is a class axiom:

Name:	general concept inclusion (GCI)
Syntax:	**Class:** C **SubClassOf:** $D, \ldots$
DL syntax:	$C \sqsubseteq D$
Semantics:	$\mathcal{I} \models C \sqsubseteq D$ holds iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

In Manchester syntax, axioms are grouped under entity declarations, into a frame. This means that C can only be a named class.

The ontology editor Protégé allows general class axioms with the syntax C **SubClassOf** D, where C can be a complex class expression.

Class Axioms II

If C and D are class expressions, then the following is a class axiom:

Name:	equivalence axiom
Syntax:	**Class:** C
	EquivalentTo: D
DL syntax:	$C \equiv D$
Semantics:	$C^{\mathcal{I}} = D^{\mathcal{I}}$

A special equivalence axiom is a class definition $C \equiv D$, where C is a named class.

Lecture $\sqsubseteq$ **Course**

Room $\equiv$ **Structure** $\sqcap \exists$**partOf.Building** $\sqcap \exists$**partOf**$^{-}$.**Door**

$\exists$**hasNiece**.$\top \sqsubseteq \exists$**hasSibling**.$\top$

$C \equiv D$ is equivalent to the two GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$.

Object Property Axioms

If $r, s, s_1, \ldots, s_n$ are object property expressions, then the following are object property axioms:

Name:	role inclusion	complex role inclusion
Syntax:	**ObjectProperty:** r **SubPropertyOf:** s	**ObjectProperty:** r **SubPropertyChain:** s_1 **o** ... **o** s_n
DL syntax:	$r \sqsubseteq s$	$s_1 \circ \cdots \circ s_n \sqsubseteq r$
Semantics:	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$	$s_1^{\mathcal{I}} \circ \cdots \circ s_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$

$\mathsf{owns} \sqsubseteq \mathsf{belongsTo}^-$ $\quad$ $\mathsf{belongsTo}^- \sqsubseteq \mathsf{owns}$ $\quad$ $\mathsf{partOf} \circ \mathsf{partOf} \sqsubseteq \mathsf{partOf}$

Object Property Axioms II

If r, s are object property expressions, then the following are also object property axioms:

Name:	role disjointness	role reflexivity
Syntax:	**ObjectProperty:** r **DisjointWith:** s	**ObjectProperty:** r **Characteristics: Reflexive**
DL syntax:	$\mathrm{Dis}(r, s)$	$\mathrm{Ref}(r)$
Semantics:	$r^{\mathcal{I}} \cap s^{\mathcal{I}} = \emptyset$	$\{(x, x) \mid x \in \Delta^{\mathcal{I}}\} \subseteq r^{\mathcal{I}}$

Dis(hasDaughter, hasSon)   Ref(hasRelative)

Assertions I

Assertions are axioms about named individuals, also called facts.

Given $a, b \in \mathbf{I}$, a concept C, and a role r, the following are assertions:

Name:	class assertion	[negative] object property assertion
Syntax:	**Individual:** a **Types:** C	**Individual:** a **Facts:** **[not]** $r\ b$
DL syntax:	$a : C$	$(a, b) : [\neg]r$
Semantics:	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ $[(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}]$

Assertions II

Given $a, b \in \mathbf{I}$, the following are also assertions:

Name:	individual equality	individual inequality
Syntax:	**Individual:** a **SameAs:** b	**Individual:** a **DifferentFrom:** b
DL syntax:	$a \approx b$	$a \not\approx b$
Semantics:	$a^{\mathcal{I}} = b^{\mathcal{I}}$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

$\mathsf{APB/E005} : \mathsf{Room}$ $\quad (\mathsf{APB}, \mathsf{TU\ Dresden}) : \mathsf{belongsTo}$ $\quad \mathsf{APB} \not\approx \mathsf{APB/E005}$

Additional Axioms: Syntactic Sugar

Syntax	DL syntax	Equivalent axioms
DisjointWith:	$\mathsf{Dis}(C, D)$	$C \sqsubseteq \neg D$ or $C \sqcap D \sqsubseteq \bot$
DisjointUnionOf:		$C \equiv D_1 \sqcup \cdots \sqcup D_n, D_1 \sqsubseteq \neg D_2, \ldots$
EquivalentTo:	$r \equiv s$	$r \sqsubseteq s, s \sqsubseteq r$
Domain:	$\mathsf{Dom}(r) \sqsubseteq C$	$\top \sqsubseteq \forall r^-.C$ or $\exists r.\top \sqsubseteq C$
Range:	$\mathsf{Ran}(r) \sqsubseteq C$	$\top \sqsubseteq \forall r.C$ or $\exists r^-.\top \sqsubseteq C$
InverseOf:		$r \equiv s^-$
Characteristics:		
Irreflexive	$\mathsf{Irr}(r)$	$\exists r.\mathsf{Self} \sqsubseteq \bot$
Functional	$\mathsf{Fun}(r)$	$\top \sqsubseteq \leq 1\, r.\top$
Symmetric	$\mathsf{Sym}(r)$	$r \sqsubseteq r^-$
Asymmetric	$\mathsf{Asy}(r)$	$\mathsf{Dis}(r, r^-)$
Transitive	$\mathsf{Tra}(r)$	$r \circ r \sqsubseteq r$

Models and Reasoning

A DL ontology is a triple $\mathcal{O} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$, where

- $\mathcal{A}$ is an ABox, a set of assertions,
- $\mathcal{T}$ is a TBox, a set of class axioms,
- $\mathcal{R}$ is an RBox, a set of object property axioms.

An interpretation is a model of $\mathcal{O}$ if it is a model of all its axioms.

We sometimes write ontologies as sets $\mathcal{O} = \mathcal{A} \cup \mathcal{T} \cup \mathcal{R}$.

$\mathcal{O}$ is consistent if it has a model.

$\mathcal{O}$ entails an axiom α ($\mathcal{O} \models \alpha$) if every model of $\mathcal{O}$ is also a model of α.

An inconsistent ontology entails all axioms (also $\top \sqsubseteq \bot$)!

In fact, $\mathcal{O}$ is inconsistent iff $\mathcal{O} \models \top \sqsubseteq \bot$.

Other Reasoning Problems

Let C, D be concepts and $a \in \mathbf{I}$.

- If $\mathcal{O} \models C \sqsubseteq D$, we say that C is subsumed by D w.r.t. $\mathcal{O}$. $C \sqsubseteq_{\mathcal{O}} D$
- If $\mathcal{O} \models C \equiv D$, we say that C is equivalent to D w.r.t. $\mathcal{O}$. $C \equiv_{\mathcal{O}} D$
- If $C \sqsubseteq_{\mathcal{O}} D$ and $C \not\equiv_{\mathcal{O}} D$, C is strictly subsumed by D w.r.t. $\mathcal{O}$. $C \sqsubset_{\mathcal{O}} D$
- If $\mathcal{O} \models C \sqcap D \sqsubseteq \bot$, we say that C and D are disjoint w.r.t. $\mathcal{O}$.

- If $\mathcal{O} \models a : C$, then a is an instance of C w.r.t. $\mathcal{O}$.
- If $\mathcal{O} \not\models C \sqsubseteq \bot$, then C is satisfiable w.r.t. $\mathcal{O}$.
- If all concept names in $\mathcal{O}$ are satisfiable w.r.t. $\mathcal{O}$, then $\mathcal{O}$ is coherent.
- Classification is the task of computing all entailments of the form $\mathcal{O} \models A \sqsubseteq B$, where $A, B \in \mathbf{C}$.
- Materialization is the task of computing all entailments of the form $\mathcal{O} \models a : A$ and $\mathcal{O} \models (a, b) : r$, where $a, b \in \mathbf{I}$, $A \in \mathbf{C}$, and $r \in \mathbf{R}$.

Examples

The ontology

$\{\mathsf{Felix} : \mathsf{Cat}, \quad \mathsf{Cat} \sqsubseteq \mathsf{Animal}, \quad (\mathsf{Felix}, \mathsf{Toby}) : \mathsf{hasFather},$
$\exists \mathsf{hasFather}.\top \sqsubseteq \mathsf{Human}\}$

is consistent and coherent, and entails $\mathsf{Felix} : \mathsf{Human}$.

$\{\mathsf{Felix} : \mathsf{Cat}, \quad \mathsf{Cat} \sqsubseteq \mathsf{Animal}, \quad (\mathsf{Felix}, \mathsf{Toby}) : \mathsf{hasFather},$
$\exists \mathsf{hasFather}.\top \sqsubseteq \mathsf{Human}, \quad \mathsf{Human} \sqsubseteq \neg\mathsf{Animal}\}$

is inconsistent.

$\{\mathsf{Human} \sqsubseteq \neg\mathsf{Animal}, \quad \mathsf{Werewolf} \sqsubseteq \mathsf{Human} \sqcap \mathsf{Wolf}, \quad \mathsf{Wolf} \sqsubseteq \mathsf{Animal}\}$

is consistent, but not coherent, because Werewolf is unsatisfiable.

Disjointness axioms are very useful for debugging ontologies.
Inconsistent or incoherent ontologies indicate modeling errors.

Reasoning without an Ontology

Certain equivalences and subsumptions hold w.r.t. any ontology (in particular the empty ontology $\emptyset$).

We write $\equiv$ instead of $\equiv_\emptyset$ and $\sqsubseteq$ instead of $\sqsubseteq_\emptyset$.

Examples:

$$C \sqsubseteq \top$$
$$\exists r.C \sqsubseteq \exists r.\top$$
$$C \sqcap D \sqsubseteq C$$
$$\exists r.(C \sqcap D) \sqsubseteq (\exists r.C) \sqcap (\exists r.D)$$

$$\neg(C \sqcap D) \equiv \neg C \sqcup \neg D$$
$$\exists r.C \equiv \neg \forall r.\neg C$$
$$\leq n\, r.C \equiv \neg(\geq (n+1)\, r.C)$$
$$\geq 1\, r.C \equiv \exists r.C$$
$$\leq 0\, r.C \equiv \forall r.\neg C$$

OWL 2 DL

OWL 2 DL corresponds to the description logic $\mathcal{SROIQ}$. To retain decidability, this logic imposes several restrictions on the use of roles.

- The RBox must be regular.
- Number restrictions, self restrictions, and disjoint role axioms can only contain simple roles.

Regular RBoxes

Let $\mathbf{R}^-(\mathcal{O})$ be the set of all roles in $\mathcal{O} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ and their inverses, where the inverse of r^- is r.

The RBox $\mathcal{R}$ is regular if there is a strict partial order $<$ on $\mathbf{R}^-(\mathcal{O})$ such that

- $r < s$ iff $r^- < s$ for all $r, s \in \mathbf{R}^-(\mathcal{O})$, and
- for all $w \sqsubseteq r \in \mathcal{R}$, w has one of the following forms:
  - $r \circ r$ (transitivity),
  - r^- (symmetry),
  - $r_1 \circ \dots \circ r_n$, $r \circ r_1 \circ \dots \circ r_n$, or $r_1 \circ \dots \circ r_n \circ r$ such that $r_i < r$ for all $i \in \{1, \dots, n\}$.

Intuitively, there should be no non-trivial cyclic relationships between roles.

> The RBox $\{\mathsf{hasFather} \circ \mathsf{hasBrother} \sqsubseteq \mathsf{hasUncle},$
> $\mathsf{hasChild} \circ \mathsf{hasUncle} \sqsubseteq \mathsf{hasBrother}\}$ is not regular.

Simple Roles

OWL 2 DL corresponds to the description logic $\mathcal{SROIQ}$. To retain decidability, this logic imposes several restrictions on the use of roles.

- The RBox must be regular.
- Number restrictions, self restrictions, and disjoint role axioms can only contain simple roles.

The set of non-simple roles is inductively defined as follows:

- If $r_1 \circ \cdots \circ r_n \sqsubseteq r \in \mathcal{R}$ with $n \geq 2$, then r and r^- are non-simple.
- If $s \sqsubseteq r \in \mathcal{R}$ and s is non-simple, then r and r^- are non-simple.

All other roles are simple.

Transitive roles and roles that have transitive subroles are not simple.

Query Answering and SPARQL

SPARQL is a very expressive SQL-like query language that can be used to query RDF data and OWL ontologies.

For OWL, this requires more complex reasoning than entailment of axioms.

In DLs (and database theory), this corresponds to conjunctive queries.

$$\mathsf{SeminarRoom}(x) \land \mathsf{partOf}(x, \mathsf{APB}) \land \exists y.\mathsf{Lecture}(y) \land \mathsf{takesPlace}(y, x, \mathsf{Wed\ 2.DS})$$

The decidability of answering conjunctive queries is unknown for OWL 2 DL.

Conjunctive query answering is not well supported by automated reasoners.

OWL 2 Profiles

In full $\mathcal{SROIQ}$, reasoning is 2-NEXPTIME-complete. Further restricting the expressivity of the logic improves the complexity of reasoning.

There are three OWL 2 Profiles, called OWL 2 EL, OWL 2 QL, and OWL 2 RL, which roughly correspond to description logics of the $\mathcal{EL}$ and *DL-Lite* families, and to Description Logic Programs (DLP).

Reasoning in these profiles is possible in polynomial time.

OWL 2 EL

OWL 2 EL allows only the following:

- roles: only role names
- concepts: concept names, conjunction, existential restriction, nominals, self restriction
- axioms: GCIs, concept disjointness, complex role inclusions, domain and range restrictions, reflexive roles, all assertions

This profile covers many biomedical ontologies with a large number of concepts and roles.

$$\mathsf{LiverCancer} \equiv \mathsf{TumorOfLiver} \sqcap \exists \mathsf{associatedMorphology}.\mathsf{MalignantNeoplasm} \sqcap \exists \mathsf{findingSite}.\mathsf{Liver}$$

$$\mathsf{findingSite} \circ \mathsf{partOf} \sqsubseteq \mathsf{findingSite}$$

OWL 2 QL

OWL 2 QL allows only the following:

- roles: role names and inverse roles
- concepts: concept names, unqualified existential restriction (only with owl:Thing)
- axioms: GCIs, concept disjointness, role inclusions (but not complex ones), domain and range restrictions, role disjointness, reflexive and irreflexive roles, all assertions except individual equality and negated role assertions

This profile is suitable for SPARQL query answering in applications with a large number of individuals and assertions (a.k.a. "data").

$\exists \text{employedBy} \sqsubseteq \text{Employee}$ $\quad$ $\exists \text{employedBy}^- \sqsubseteq \text{Company}$

$\text{Employee} \sqcap \text{Company} \sqsubseteq \bot$ $\quad$ $\text{employedBy}^- \sqsubseteq \text{employs}$

OWL 2 RL

OWL 2 RL allows all roles and axioms except equivalence axioms, but not $\top$, and only GCIs of the form $C \sqsubseteq D$, where

- C may be a concept name, nominal, conjunction, disjunction, existential restriction
- D may be a concept name, value restriction, existential restriction over a nominal, at-most restriction with $n = 0$ or $n = 1$

($\top$ can be used inside role restrictions)

This profile trades the expressivity of existential restrictions and disjunctions against faster reasoning, and can be implemented using rule-based reasoning engines (cf. Datalog).

$\mathsf{Human} \sqcap \mathsf{Male} \sqsubseteq \mathsf{Man}$ $\qquad$ $\mathsf{Man} \sqcap \exists \mathsf{hasChild}.\top \sqsubseteq \mathsf{Father}$

$\mathsf{Human} \sqsubseteq \forall \mathsf{hasChild}.\mathsf{Human}$ $\qquad$ $\mathsf{Human} \sqsubseteq \leq 1\ \mathsf{hasFather}.\top$

OWL 2 Reasoners

OWL 2 DL:

- Konclude http://derivo.de/en/products/konclude/
- Pellet https://github.com/stardog-union/pellet/
- FaCT++ https://bitbucket.org/dtsarkov/factplusplus/
- HermiT https://github.com/phillord/hermit-reasoner/
- PAGOdA https://www.cs.ox.ac.uk/isg/tools/PAGOdA/

OWL 2 EL:

- ELK https://github.com/liveontologies/elk-reasoner/
- CEL https://lat.inf.tu-dresden.de/systems/cel/

OWL 2 QL:

- ontop https://github.com/ontop/ontop/
- Mastro http://www.dis.uniroma1.it/~mastro/

OWL 2 RL:

- RDFox http://www.cs.ox.ac.uk/isg/tools/RDFox/

Additional Features: Datatypes

OWL 2 also includes the datatypes defined by XML Schema:

xsd:integer   xsd:decimal   xsd:float   xsd:string

A literal represents a constant value of a specific datatype.

"Lecture"   "Lecture"@en-US   -1.2E-2F   "-10"^^xsd:integer

In description logics:

A concrete domain is a set Δ^D of values, together with collections of datatypes and literals.

Each literal ℓ is associated to a unique value $\ell^D \in \Delta^D$, e.g., "-10"^^xsd:integer represents the number -10.

A datatype T is interpreted as a set of values $T^D \subseteq \Delta^D$, e.g., xsd:integer represents the set of all integers.

Data Properties and Axioms

Using data properties, individuals can be assigned data values:

DataProperty: hasSize

In DLs, they are called concrete role names: $\mathbf{R_c} = \{\text{hasSize}, \dots\}$

The definition of interpretations $\mathcal{I}$ is extended to assign each $r_c \in \mathbf{R_c}$ a binary relation $r_c^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^D$.

Data properties can also have **SubPropertyOf** and **DisjointWith** axioms, but not **SubPropertyChain** and no **Characteristics** other than **Functional**.

There are also data property assertions about values for named individuals.

$(\text{APB/E005}, 30) : \text{hasSize}$ $\qquad$ $\text{Fun}(\text{hasSize})$

Class Expressions: Data Property Restrictions

The expressions **some**, **only**, **min**, and **max** can also be used for data properties, but with a datatype in place of a class.

For example, if v is a data property and T a datatype, then the following is a class expression:

Name:	existential (datatype) restriction
Syntax:	v **some** T
DL syntax:	$\exists v.T$
Semantics:	$\{x \mid \exists y.(x,y) \in v^{\mathcal{I}} \wedge y \in T^{D}\}$

hasSize **some** xsd:integer   hasName **max** 1 xsd:string

Similar to nominals, if ℓ is a literal, then **{**ℓ**}** denotes a datatype that represents the singleton set $\{\ell^{D}\} \subseteq \Delta^{D}$.

hasSize **some {**30**}**   hasSize **value** 30

OWL 2 is more than Description Logics

In addition to being a modeling language for ontologies (with semantics based on DLs), OWL 2 has several features for managing ontologies:

- All entities are identified by an International Resource Identifier (IRI) that is unique across ontologies.
- Ontologies can be imported into other ontologies.
- All entities can be annotated by non-logical statements containing additional information.

OWL 2 Features: IRIs

Every entity (class, property, individual) and even the ontology itself is uniquely identified by an IRI, often in the form of a URL.

http://inf.tu-dresden.de/university-ontology#Lecture	uo:Lecture
http://inf.tu-dresden.de/university-ontology#Course	uo:Course

URLs should be dereferenceable, i.e., lead to a web page about the entity.

Prefix definitions are used to abbreviate IRIs.

Prefix: uo: http://inf.tu-dresden.de/university-ontology#

Standard prefixes:

rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
xsd:	http://www.w3.org/2001/XMLSchema#
owl:	http://www.w3.org/2002/07/owl#

OWL 2 Features: Imports

Like other entities, ontologies are declared using their IRI.

They can import all entities and axioms of other ontologies (via the IRI).

> **Ontology:** http://inf.tu-dresden.de/university-ontology.owl
> **Import:** http://purl.obolibrary.org/obo/bfo.owl

Imports are transitive, so importing an ontology that imports the BFO also grants access to all BFO entities and axioms.

> The import closure of an ontology $\mathcal{O}$ is a set containing $\mathcal{O}$ and all the ontologies that $\mathcal{O}$ imports (directly or indirectly).
>
> The axiom closure of $\mathcal{O}$ is the smallest set that contains all the axioms from each ontology $\mathcal{O}'$ in the import closure of $\mathcal{O}$.

Imports vs. IRIs

Importing an ontology adds all its entities and axioms to the current ontology.

One can always use the entities of another ontology by referring to their unique IRI, without importing its axioms.

This allows to "overwrite" existing axioms, since the new ontology is a completely new collection of axioms over the same vocabulary.

This is problematic, as there are no automated checks for inconsistency/incoherence of the new axioms w.r.t. the old ones.

Referring to entities of other ontologies without importing them is commonplace when dealing with pure vocabularies (without axioms).

For example, the standard prefixes rdf:, rdfs:, xsd:, and owl: contain only entity declarations, and nearly every OWL ontology uses them.

OWL 2 Features: Annotations

Annotations provide additional information about ontologies, entities, and axioms via annotation properties.

Class: Course
    Annotations: rdfs:label "Course"@en-US
        rdfs:label "Kurs"@de-DE
        rdfs:comment "A course offered by a university."
        dc:creator **Stefan Borgwardt**
        dc:issued "2018-03-07"^^xsd:date
        rdfs:seeAlso ubo:Course

Annotations have no semantics, but help the user to keep track of provenance, intuitive meaning, and external links to other ontologies.

Protégé uses rdfs:label annotations to display entities, if they are available.

Note: Binary vs. *n*-ary Relations

OWL 2 and description logics can only express unary and binary relations. This is not without loss of generality, but *n*-ary relations with $n \geq 3$ can partially be simulated.

annotatedWith(A-kinase anchor protein 9, x, Homo sapiens)

$\leadsto$

Annotation(a112),
hasAnnotation(A-kinase anchor protein 9, a112),
annotationProcess(a112, x),
annotationSpecies(a112, Homo sapiens)

This process is called reification.

Of course, it is more cumbersome to formulate axioms over this representation.

Note: Open World vs. Closed World

OWL and DLs make the open-world assumption, i.e., facts that are not explicitly stated are simply unknown.

> The ontology $(\{(\mathsf{Bob}, \mathsf{Fred}) : \mathsf{hasChild}\}, \emptyset, \emptyset)$ does not entail $\mathsf{Bob} : \mathsf{Father}$ nor $\mathsf{Bob} : \neg\mathsf{Father}$.
>
> The ontology $(\{(\mathsf{Bob}, \mathsf{Fred}) : \mathsf{hasChild}\}, \{\exists \mathsf{hasChild}.\top \sqsubseteq \mathsf{Father}\}, \emptyset)$ entails $\mathsf{Bob} : \mathsf{Father}$.

Databases make the closed-world assumption, i.e., facts that are not explicitly stated are assumed to be false.

> Consider the database that contains only the fact $\mathsf{hasChild}(\mathsf{Bob}, \mathsf{Fred})$.
>
> The formula $\neg\mathsf{Father}(\mathsf{Bob})$ is satisfied in this database.
>
> The formula $\forall x.(\exists y.\mathsf{hasChild}(x, y)) \to \mathsf{Father}(x)$ is not satisfied.

A database represents only one interpretation. An ontology has a large number of possible interpretations, which are constrained by axioms.

A Complete Ontology

```
Prefix: uo: http://inf.tu-dresden.de/university-ontology.owl#
Ontology: http://inf.tu-dresden.de/university-ontology.owl
  ObjectProperty: uo:partOf
    Characteristics: Transitive
  Class: uo:Building
    DisjointWith: uo:University, uo:Room
  Class: uo:Room
    DisjointWith: uo:University
    SubClassOf: uo:partOf some uo:Building
  Class: uo:University
  Individual: uo:APB/E005
    Types: uo:Room
    Facts: uo:partOf uo:APB
  Individual: uo:APB
```

A Complete Ontology

$\mathcal{O} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ with

$\mathcal{A} = \{\text{APB/E005} : \text{Room},\ (\text{APB/E005}, \text{APB}) : \text{partOf}\}$

$\mathcal{T} = \{\text{Room} \sqsubseteq \neg\text{University},\ \text{Room} \sqsubseteq \exists\text{partOf.Building},$
$\quad \text{Building} \sqsubseteq \neg\text{University},\ \text{Building} \sqsubseteq \neg\text{Room}\}$

$\mathcal{R} = \{\text{partOf} \circ \text{partOf} \sqsubseteq \text{partOf}\}$

In the rest of the lecture, we will mainly use DL syntax.