FOUNDATIONS OF SEMANTIC WEB TECHNOLOGIES

OWL & Description Logics

Markus Krötzsch

Dresden, 16 May 2014
# Content

<table>
<thead>
<tr>
<th>Topic</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview &amp; XML</td>
<td>11 APR</td>
<td>DS5</td>
</tr>
<tr>
<td>Introduction into RDF</td>
<td>11 APR</td>
<td>DS6</td>
</tr>
<tr>
<td>RDFS – Syntax &amp; Intuition</td>
<td>16 APR</td>
<td>DS6</td>
</tr>
<tr>
<td>Tutorial 1</td>
<td>23 APR</td>
<td>DS6</td>
</tr>
<tr>
<td>RDFS – Semantics</td>
<td>25 APR</td>
<td>DS5</td>
</tr>
<tr>
<td>RDFS Rule-based Reasoning</td>
<td>25 APR</td>
<td>DS6</td>
</tr>
<tr>
<td>Tutorial 2</td>
<td>30 APR</td>
<td>DS6</td>
</tr>
<tr>
<td>SPARQL – Syntax &amp; Intuition</td>
<td>02 MAY</td>
<td>DS5</td>
</tr>
<tr>
<td>SPARQL – Semantics</td>
<td>02 MAY</td>
<td>DS6</td>
</tr>
<tr>
<td>SPARQL Algebra</td>
<td>09 MAY</td>
<td>DS5</td>
</tr>
<tr>
<td>Tutorial 3</td>
<td>09 MAY</td>
<td>DS6</td>
</tr>
<tr>
<td>OWL – Syntax &amp; Intuition</td>
<td>14 MAY</td>
<td>DS6</td>
</tr>
<tr>
<td>OWL &amp; Description Logics</td>
<td>16 MAY</td>
<td>DS5</td>
</tr>
<tr>
<td>Tutorial 4</td>
<td>16 MAY</td>
<td>DS6</td>
</tr>
<tr>
<td>Tableau I</td>
<td>23 MAY</td>
<td>DS6</td>
</tr>
<tr>
<td>Tableau II</td>
<td>30 MAY</td>
<td>DS5</td>
</tr>
<tr>
<td>Tutorial 5</td>
<td>30 MAY</td>
<td>DS6</td>
</tr>
<tr>
<td>Hypertableau I</td>
<td>4 JUN</td>
<td>DS6</td>
</tr>
<tr>
<td>Hypertableau II</td>
<td>6 JUN</td>
<td>DS5</td>
</tr>
<tr>
<td>Tutorial 6</td>
<td>6 JUN</td>
<td>DS6</td>
</tr>
<tr>
<td>SPARQL 1.1</td>
<td>18 JUN</td>
<td>DS6</td>
</tr>
<tr>
<td>SPARQL Entailment</td>
<td>20 JUN</td>
<td>DS5</td>
</tr>
<tr>
<td>Tutorial 7</td>
<td>20 JUN</td>
<td>DS6</td>
</tr>
<tr>
<td>OWL &amp; Rules</td>
<td>25 JUN</td>
<td>DS6</td>
</tr>
<tr>
<td>Ontology Editing</td>
<td>27 JUL</td>
<td>DS5</td>
</tr>
<tr>
<td>Ontology Engineering</td>
<td>27 JUL</td>
<td>DS6</td>
</tr>
<tr>
<td>Tutorial 8</td>
<td>2 JUL</td>
<td>DS6</td>
</tr>
<tr>
<td>Linked Data &amp; Applications</td>
<td>4 JUL</td>
<td>DS5</td>
</tr>
<tr>
<td>Q&amp;A Session</td>
<td>9 JUL</td>
<td>DS6</td>
</tr>
<tr>
<td>Q&amp;A Session</td>
<td>11 JUL</td>
<td>DS5</td>
</tr>
</tbody>
</table>

TU Dresden, 16 May 2014 - Foundations of Semantic Web Technologies
OWL & Description Logics
OWL & Description Logics

User Interface & applications

Trust

Proof

Unifying Logic

Query: SPARQL

ontology: OWL

Rules: RIF

RDF-S

Data interchange: RDF

XML

URI

Unicode

Crypto
Agenda

- Motivation
- Introduction Description Logics
- The Description Logic \( \mathcal{ALC} \)
- Extensions of \( \mathcal{ALC} \)
- Inference Problems
Agenda

- Motivation
- Introduction Description Logics
- The Description Logic $\mathcal{ALC}$
- Extensions of $\mathcal{ALC}$
- Inference Problems
Description Logics

- Description logics (DLs) are one of the current KR paradigms
- Have significantly influenced the standardization of Semantic Web languages
  - OWL is essentially based on DLs
- Numerous reasoners
  - Quonto, JFact, FaCT++, RacerPro
  - Owlgres, Pellet, SHER, snorocket
  - OWLIM, Jena, Oracle Prime, QuOnto
  - Trowl, Hermit, condor, CB
  - ELK, konclude, RScale
OWL Tools

Tools and editors for OWL:

- Protégé, http://protege.stanford.edu (free editor)

- TopBraid Composer, http://www.topquadrant.com (commercial editor)

- OWL API, http://owlapi.sourceforge.net/ (free Java library)

- OWL Syntax Converter, http://mowl-power.cs.man.ac.uk:8080/converter/ (conversion/validation service)
Description Logics

- Origin of DLs: semantic networks and frame-based systems
- Downside of the former: only intuitive semantics - diverging interpretations
- DLs provide a formal semantics on logical grounds
- Can be seen as decidable fragments of first-order logic (FOL), closely related to modal logics
- Significant portion of DL-related research devoted to clarifying the computational effort of reasoning tasks in terms of their worst-case complexity
- Despite high complexities, even for expressive DLs exist optimized reasoning algorithms with good average case behavior
Agenda

• Motivation
• Introduction Description Logics
• The Description Logic $\mathcal{ALC}$
• Extensions of $\mathcal{ALC}$
• Inference Problems
DL building blocks

- **individuals:** birte, cs63.800, sebastian, etc.
  - constants in FOL, resources in RDF
- **concept names:** Person, Course, Student, etc.
  - unary predicates in FOL, classes in RDF
- **role names:** hasFather, attends, worksWith, etc.
  - binary predicates in FOL, properties in RDF
  - can be subdivided into abstract and concrete roles (object und data properties)

The set of all individual, concept and role names is called signature or vocabulary
Constituents of a DL Knowledge Base

- **TBox** $\mathcal{T}$: information about concepts and their taxonomic dependencies
- **ABox** $\mathcal{A}$: information about individuals, their concept and role memberships
- **RBox** $\mathcal{R}$: information about roles and their mutual dependencies

in more expressive DLs also:
Agenda

- Motivation
- Introduction Description Logics
- The Description Logic $\mathcal{ALC}$
- Extensions of $\mathcal{ALC}$
- Inference Problems
AŁC, Attribute Language with Complement, is the simplest DL that is Boolean closed

we define (complex) AŁC concepts as follows:

- every concept name is a concept,
- ⊤ and ⊥ are concepts,
- for concepts C and D, ¬C, C ∩ D, and C ∪ D are concepts,
- for a role r and a concept C, ∃r.C and ∀r.C are concepts

Example: Student ∩ ∀attendsCourse.MasterCourse

Intuitively: describes the concept comprising all students that attend only master courses
Concept Constructors vs. OWL

- $\top$ corresponds to `owl:Thing`
- $\bot$ corresponds to `owl:Nothing`
- $\sqcap$ corresponds to `owl:intersectionOf`
- $\sqcup$ corresponds to `owl:unionOf`
- $\neg$ corresponds to `owl:complementOf`
- $\forall$ corresponds to `owl:allValuesFrom`
- $\exists$ corresponds to `owl:someValuesFrom`
Concept Axioms

For concepts $C, D$, a general concept inclusion (GCI) axiom has the form

$$C \sqsubseteq D$$

- $C \equiv D$ is an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$
- a TBox (terminological Box) consists of a set of GCIs

TBox $\mathcal{T}$
ABox

an $\mathcal{ALC}$ ABox assertion can be of one of the following forms

- $C(a)$, called concept assertion
- $r(a, b)$, called role assertion

an ABox consists of a set of ABox assertions
The Description Logic $\mathcal{ALC}$

- $\mathcal{ALC}$ is a syntactic variant of the modal logic $\mathcal{K}$ with multiple modalities
- semantics defined in a model-theoretic way, that is, via interpretations
- can be expressed in first-order predicate logic
- a DL interpretation $\mathcal{I}$ consists of a domain $\Delta^\mathcal{I}$ and a function $\cdot^\mathcal{I}$, that maps
  - individual names $a$ to domain elements $a^\mathcal{I} \in \Delta^\mathcal{I}$
  - concept names $C$ to sets of domain elements $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$
  - role names $r$ to sets of pairs of domain elements $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
Schematic Representation of an Interpretation

- Individual names: \( \ldots a \ldots \)
- Concept names: \( \ldots C \ldots \)
- Role names: \( \ldots r \ldots \)
the interpretation of complex concepts is defined inductively:

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>⊤</td>
<td>$\Delta^\mathcal{I}$</td>
</tr>
<tr>
<td>bottom</td>
<td>⊥</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>negation</td>
<td>$\neg C$</td>
<td>$\Delta^\mathcal{I} \setminus C^\mathcal{I}$</td>
</tr>
<tr>
<td>conjunction</td>
<td>$C \sqcap D$</td>
<td>$C^\mathcal{I} \cap D^\mathcal{I}$</td>
</tr>
<tr>
<td>disjunction</td>
<td>$C \sqcup D$</td>
<td>$C^\mathcal{I} \cup D^\mathcal{I}$</td>
</tr>
<tr>
<td>universal quantifier</td>
<td>$\forall r.C$</td>
<td>${ x \in \Delta^\mathcal{I} \mid (x, y) \in r^\mathcal{I} \text{ implies } y \in C^\mathcal{I} }$</td>
</tr>
<tr>
<td>existential quantifier</td>
<td>$\exists r.C$</td>
<td>${ x \in \Delta^\mathcal{I} \mid \text{there is some } y \in \Delta^\mathcal{I}, \text{ such that } (x, y) \in r^\mathcal{I} \text{ and } y \in C^\mathcal{I} }$</td>
</tr>
</tbody>
</table>
Interpretation of Axioms

interpretation can be extended to axioms:

<table>
<thead>
<tr>
<th>name</th>
<th>syntax</th>
<th>semantic</th>
<th>notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>inclusion</td>
<td>$C \sqsubseteq D$</td>
<td>holds if $C^\mathcal{I} \subseteq D^\mathcal{I}$</td>
<td>$\mathcal{I} \models C \sqsubseteq D$</td>
</tr>
<tr>
<td>equivalence</td>
<td>$C \equiv D$</td>
<td>holds if $C^\mathcal{I} = D^\mathcal{I}$</td>
<td>$\mathcal{I} \models C \equiv D$</td>
</tr>
<tr>
<td>concept assertion</td>
<td>$C(a)$</td>
<td>holds if $a^\mathcal{I} \in C^\mathcal{I}$</td>
<td>$\mathcal{I} \models C(a)$</td>
</tr>
<tr>
<td>role assertion</td>
<td>$r(a, b)$</td>
<td>holds if $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$</td>
<td>$\mathcal{I} \models r(a, b)$</td>
</tr>
</tbody>
</table>
Logical Entailment in Knowledge Bases

- Let $\mathcal{I}$ be an interpretation, $\mathcal{T}$ a TBox, $\mathcal{A}$ an Abox and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ a knowledge base.
- $\mathcal{I}$ is a model for $\mathcal{T}$, if $\mathcal{I} \models ax$ for every axiom $ax$ in $\mathcal{T}$, written $\mathcal{I} \models \mathcal{T}$.
- $\mathcal{I}$ is a model for $\mathcal{A}$, if $\mathcal{I} \models ax$ for every assertion $ax$ in $\mathcal{A}$, written $\mathcal{I} \models \mathcal{A}$.
- $\mathcal{I}$ is a model for $\mathcal{K}$, if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$.
- An axiom $ax$ follows from $\mathcal{K}$, written $\mathcal{K} \models ax$, if every model $\mathcal{I}$ of $\mathcal{K}$ is also a model of $ax$. 
Semantics via Translation into FOL

translation of TBox axioms into first-order predicate logics through the mapping $\pi$ with $C, D$ complex classes, $r$ a role and $A$ an atomic class:

$$
\pi(C \sqsubseteq D) = \forall x. (\pi_x(C) \rightarrow \pi_x(D)) \quad \pi(C \equiv D) = \forall x. (\pi_x(C) \leftrightarrow \pi_x(D))
$$
Semantics via Translation into FOL

translation of TBox axioms into first-order predicate logics through the mapping $\pi$ with $C, D$ complex classes, $r$ a role and $A$ an atomic class:

$$\pi(C \sqsubseteq D) = \forall x. (\pi_x(C) \rightarrow \pi_x(D)) \quad \pi(C \equiv D) = \forall x. (\pi_x(C) \leftrightarrow \pi_x(D))$$

$$\pi_x(A) = A(x)$$
$$\pi_x(\neg C) = \neg \pi_x(C)$$
$$\pi_x(C \cap D) = \pi_x(C) \land \pi_x(D)$$
$$\pi_x(C \cup D) = \pi_x(C) \lor \pi_x(D)$$
$$\pi_x(\forall r.C) = \forall y. (r(x, y) \rightarrow \pi_y(C))$$
$$\pi_x(\exists r.C) = \exists y. (r(x, y) \land \pi_y(C))$$
Semantics via Translation into FOL

translation of TBox axioms into first-order predicate logics through the mapping \( \pi \) with \( C, D \) complex classes, \( r \) a role and \( A \) an atomic class:

\[
\begin{align*}
\pi(C \sqsubseteq D) &= \forall x.(\pi_x(C) \rightarrow \pi_x(D)) & \pi(C \equiv D) &= \forall x.(\pi_x(C) \leftrightarrow \pi_x(D)) \\
\pi_x(A) &= A(x) & \pi_y(A) &= A(y) \\
\pi_x(\neg C) &= \neg \pi_x(C) & \pi_y(\neg C) &= \neg \pi_y(C) \\
\pi_x(C \cap D) &= \pi_x(C) \land \pi_x(D) & \pi_y(C \cap D) &= \pi_y(C) \land \pi_y(D) \\
\pi_x(C \cup D) &= \pi_x(C) \lor \pi_x(D) & \pi_y(C \cup D) &= \pi_y(C) \lor \pi_y(D) \\
\pi_x(\forall r.C) &= \forall y.(r(x,y) \rightarrow \pi_y(C)) & \pi_y(\forall r.C) &= \forall x.(r(y,x) \rightarrow \pi_x(C)) \\
\pi_x(\exists r.C) &= \exists y.(r(x,y) \land \pi_y(C)) & \pi_y(\exists r.C) &= \exists x.(r(y,x) \land \pi_x(C))
\end{align*}
\]
Semantics via Translation into FOL

- translation only requires two variables

$\models \mathcal{ALC}$ is a fragment of FOL with two variables $\mathcal{L}_2$

$\models$ satisfiability checking of sets of $\mathcal{ALC}$ axioms is decidable
Agenda

• Motivation
• Introduction Description Logics
• The Description Logic $\mathcal{ALC}$
• Extensions of $\mathcal{ALC}$
• Inference Problems
Inverse Roles

- a role can be
  - a role name \( r \) or
  - an inverse role \( r^- \)

- the semantics of inverse roles is defined as follows:
  \[
  (r^-)^I = \{(y, x) \mid (x, y) \in r^I\}
  \]

- the extension of \( ALC \) by inverse roles is denoted as \( ALCI \)
- corresponds to \( owl:\text{inverseOf} \)
Parts of a Knowledge Base

- **TBox** $\mathcal{T}$: information about concepts and their taxonomic dependencies
- **ABox** $\mathcal{A}$: information about individuals, their concepts and role connections

In more expressive DLs also:

- **RBox** $\mathcal{R}$: information about roles and their mutual dependencies
Role Axioms

- for \( r, s \) roles, a role inclusion axiom – \( RIA \) has the form \( r \sqsubseteq s \)
- \( r \equiv s \) is the abbreviation for \( r \sqsubseteq s \) and \( s \sqsubseteq r \)
- an RBox (role box) or role hierarchy consists of a set of role axioms
- \( r \sqsubseteq s \) holds in an interpretation \( \mathcal{I} \) if \( r^\mathcal{I} \subseteq s^\mathcal{I} \), written \( \mathcal{I} \models r \sqsubseteq s \)
- the extension of \( ALC \) by role hierarchies is denoted with \( ALC\mathcal{H} \), if we also have inverse roles: \( ALC\mathcal{HI} \)
- corresponds to \( \text{owl:subPropertyOf} \)
An Example Knowledge Base

**RBox R**

own ⊑ careFor

**TBox T**

Healthy ⊑ ¬Dead

Cat ⊑ Dead ⊔ Alive

HappyCatOwner ⊑ ∃owns.Cat ⊓ ∀caresFor.Healthy

**ABox A**

HappyCatOwner (schrödinger)
### An Example Knowledge Base

<table>
<thead>
<tr>
<th><strong>RBox R</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>own ⊑ careFor</td>
</tr>
<tr>
<td>“If somebody owns something, they care for it.”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>TBox T</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy ⊑ ¬Dead</td>
</tr>
<tr>
<td>“Healthy beings are not dead.”</td>
</tr>
<tr>
<td>Cat ⊑ Dead ⊔ Alive</td>
</tr>
<tr>
<td>“Every cat is dead or alive.”</td>
</tr>
<tr>
<td>HappyCatOwner ⊑ ∃owns.Cat ⊓ ∀caresFor.Healthy</td>
</tr>
<tr>
<td>“A happy cat owner owns a cat and everything he cares for is healthy.”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>ABox A</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>HappyCatOwner (schrödinger)</td>
</tr>
<tr>
<td>“Schrödinger is a happy cat owner.”</td>
</tr>
</tbody>
</table>
Role Transitivity

- for $r$ a role, a **transitivity axiom** has the form $\text{Trans}(r)$

- $\text{Trans}(r)$ holds in an interpretation $\mathcal{I}$ if $r^\mathcal{I}$ is a transitive relation, i.e., $(x, y) \in r^\mathcal{I}$ and $(y, z) \in r^\mathcal{I}$ imply $(x, z) \in r^\mathcal{I}$, written $\mathcal{I} \models \text{Trans}(r)$

- the extension of $\mathcal{ALC}$ by transitivity axioms is denoted by $S$ (after the modal logic $S_4$ – although it is really based on $K_4$)

- corresponds to $\text{owl:TransitiveProperty}$
Role Functionality

- for $r$ a role, a **functionality axiom** has the form $\text{Func}(r)$
- $\text{Func}(r)$ holds in an interpretation $\mathcal{I}$ if $(x, y_1) \in r^\mathcal{I}$ and $(x, y_2) \in r^\mathcal{I}$ imply $y_1 = y_2$, written $\mathcal{I} \models \text{Func}(r)$
- translation into FOL requires equality ($=$)
- the extension of $\mathcal{ALC}$ by functionality axioms is denoted by $\mathcal{ALCF}$
- corresponds to $\text{owl:FunctionalProperty}$
Simple and Non-Simple Roles

- Given a role hierarchy $\mathcal{R}$, we let $\sqsubseteq_{\mathcal{R}}$ be the smallest relation such that:
  - if $r \sqsubseteq s$ then $r \sqsubseteq_{\mathcal{R}} s$
  - $r \sqsubseteq_{\mathcal{R}} r$ for every role $r$ (reflexive)
  - if $r \sqsubset_{\mathcal{R}} s$ and $s \sqsubset_{\mathcal{R}} t$ then $r \sqsubset_{\mathcal{R}} t$ (transitive)
  - if $r \sqsubset_{\mathcal{R}} s$ then $r^- \sqsubset_{\mathcal{R}} s^-$, where “$(r^-)^-$” means $r$

- For a role hierarchy $\mathcal{R}$, we can distinguish the roles in $\mathcal{R}$ into simple and non-simple roles:
  - a role $r$ is non-simple w.r.t. $\mathcal{R}$, if there is a role $t$ such that $\text{Trans}(t) \in \mathcal{R}$ and $t \sqsubset_{\mathcal{R}} r$ holds
  - all other roles are are simple

- Example: $\mathcal{R} = \{u \sqsubseteq t, \quad t \sqsubseteq s, \quad s^- \sqsubseteq r^-, \quad q \sqsubseteq r, \quad \text{Trans}(t)\}$

\[ u \longrightarrow t \longrightarrow s \longrightarrow r \]

non-simple:
Simple and Non-Simple Roles

- Given a role hierarchy $\mathcal{R}$, we let $\sqsupseteq^*_{\mathcal{R}}$ be the smallest relation such that:
  - if $r \sqsubseteq s$ then $r \sqsupseteq^*_{\mathcal{R}} s$
  - $r \sqsupseteq^*_{\mathcal{R}} r$ for every role $r$ (reflexive)
  - if $r \sqsupseteq^*_{\mathcal{R}} s$ and $s \sqsupseteq^*_{\mathcal{R}} t$ then $r \sqsupseteq^*_{\mathcal{R}} t$ (transitive)
  - if $r \sqsupseteq^*_{\mathcal{R}} s$ then $(r^-)^- = s^-$, where “$(r^-)^-$” means $r$

- For a role hierarchy $\mathcal{R}$, we can distinguish the roles in $\mathcal{R}$ into simple and non-simple roles:
  - a role $r$ is non-simple w.r.t. $\mathcal{R}$, if there is a role $t$ such that $\text{Trans}(t) \in \mathcal{R}$ and $t \sqsupseteq^*_{\mathcal{R}} r$ holds
  - all other roles are are simple

- Example: $\mathcal{R} = \{ u \sqsubseteq t, \quad t \sqsubseteq s, \quad s^- \sqsubseteq r^-, \quad q \sqsubseteq r, \quad \text{Trans}(t) \}$

non-simple: $t$

TU Dresden, 16 May 2014    Foundations of Semantic Web Technologies
Simple and Non-Simple Roles

- Given a role hierarchy $\mathcal{R}$, we let $\mathcal{R}$ be the smallest relation such that:
  - if $r \sqsubseteq s$ then $r \mathcal{R} s$
  - $r \mathcal{R} r$ for every role $r$ (reflexive)
  - if $r \mathcal{R} s$ and $s \mathcal{R} t$ then $r \mathcal{R} t$ (transitive)
  - if $r \mathcal{R} s$ then $r^- \mathcal{R} s^-$, where “($r^-)$” means $r$

- For a role hierarchy $\mathcal{R}$, we can distinguish the roles in $\mathcal{R}$ into simple and non-simple roles:
  - a role $r$ is non-simple w.r.t. $\mathcal{R}$, if there is a role $t$ such that $\text{Trans}(t) \in \mathcal{R}$ and $t \mathcal{R} r$ holds
  - all other roles are are simple

- Example: $\mathcal{R} = \{u \sqsubseteq t, \ t \sqsubseteq s, \ s^- \sqsubseteq r^-, \ q \sqsubseteq r, \ \text{Trans}(t)\}$

non-simple: $t, s$
Simple and Non-Simple Roles

- Given a role hierarchy \( R \), we let \( \sqsubseteq_{R} \) be the smallest relation such that:
  - if \( r \sqsubseteq s \) then \( r \sqsubseteq_{R} s \)
  - \( r \sqsubseteq_{R} r \) for every role \( r \) (reflexive)
  - if \( r \sqsubseteq_{R} s \) and \( s \sqsubseteq_{R} t \) then \( r \sqsubseteq_{R} t \) (transitive)
  - if \( r \sqsubseteq_{R} s \) then \( r^{-} \sqsubseteq_{R} s^{-} \), where \("(r^{-})^{-}\" means \( r \)

- For a role hierarchy \( R \), we can distinguish the roles in \( R \) into simple and non-simple roles:
  - a role \( r \) is non-simple w.r.t. \( R \), if there is a role \( t \) such that \( \text{Trans}(t) \in R \) and \( t \sqsubseteq_{R} r \) holds
  - all other roles are are simple

- Example: \( R = \{ u \sqsubseteq t, \ t \sqsubseteq s, \ s^{-} \sqsubseteq r^{-}, \ q \sqsubseteq r, \ \text{Trans}(t) \} \)

\[ q \]
\[ u \rightarrow t \rightarrow s \rightarrow r \]

non-simple: \( t, s, r \)
Simple and Non-Simple Roles

- Given a role hierarchy $\mathcal{R}$, we let $\sqsubseteq^*_\mathcal{R}$ be the smallest relation such that:
  - if $r \sqsubseteq s$ then $r \sqsubseteq^*_\mathcal{R} s$
  - $r \sqsubseteq^*_\mathcal{R} r$ for every role $r$ (reflexive)
  - if $r \sqsubseteq^*_\mathcal{R} s$ and $s \sqsubseteq^*_\mathcal{R} t$ then $r \sqsubseteq^*_\mathcal{R} t$ (transitive)
  - if $r \sqsubseteq^*_\mathcal{R} s$ then $(r^-)^- \sqsubseteq^*_\mathcal{R} s^-$, where "$(r^-)^-"$ means $r$

- For a role hierarchy $\mathcal{R}$, we can distinguish the roles in $\mathcal{R}$ into simple and non-simple roles:
  - a role $r$ is non-simple w.r.t. $\mathcal{R}$, if there is a role $t$ such that $\text{Trans}(t) \in \mathcal{R}$ and $t \sqsubseteq^*_\mathcal{R} r$ holds
  - all other roles are are simple

- Example: $\mathcal{R} = \{ u \sqsubseteq t, \ t \sqsubseteq s, \ s^- \sqsubseteq r^-, \ q \sqsubseteq r, \ \text{Trans}(t) \}$

\begin{center}
\begin{tikzpicture}
  \node (q) at (0,0) {$q$};
  \node (u) at (-2,0) {$u$};
  \node (t) at (-1,0) {$t$};
  \node (s) at (0,0) {$s$};
  \node (r) at (1,0) {$r$};
  \draw (q) -- (r);
  \draw (u) -- (t);
  \draw (t) -- (s);
\end{tikzpicture}
\end{center}

non-simple: $t, s, r$  simple: $q, u$

TU Dresden, 16 May 2014  Foundations of Semantic Web Technologies
(Unqualified) Number Restrictions

- For a simple role $s$ and a natural number $n$, $\leq n s$, $\geq n s$ and $= n s$ are concepts
- Semantics defined by:

$$\leq n s^I = \{ x \in \Delta^I \mid \#\{ y \in \Delta^I \mid (x, y) \in s^I \} \leq n \}$$
$$\geq n s^I = \{ x \in \Delta^I \mid \#\{ y \in \Delta^I \mid (x, y) \in s^I \} \geq n \}$$
$$= n s^I = \{ x \in \Delta^I \mid \#\{ y \in \Delta^I \mid (x, y) \in s^I \} = n \}$$

- Extension of $\mathcal{ALC}$ by (unqualified) number restrictions denoted by $\mathcal{ALCN}$
- Correspond to $\text{owl:maxCardinality}$, $\text{owl:minCardinality}$, and $\text{owl:cardinality}$
- Restriction to simple roles ensures decidability, e.g., for checking knowledge base satisfiability
- Definition of TBox requires an RBox being already defined
(Unqualified) Number Restrictions in FOL

- Translation into FOL requires equality or counting quantifiers
- Translation defined as follows (likewise for $\pi_y$):
  \[
  \pi_x(\leq n \cdot s) = \exists^{\leq n} y. (s(x, y)) \\
  \pi_x(\geq n \cdot s) = \exists^{\geq n} y. (s(x, y)) \\
  \pi_x(= n \cdot s) = \exists^{\leq n} y. (s(x, y)) \land \exists^{\geq n} y. (s(x, y))
  \]
- The following equivalences hold:
  \[
  \neg (\leq n \cdot s) \equiv (n + 1) \cdot s \\
  \neg (\geq 0 \cdot s) \equiv \bot \\
  \leq 0 \cdot s \equiv \forall s. \bot \\
  \neg (\geq n \cdot s) \equiv (n - 1) \cdot s, \quad n \geq 1 \\
  \geq 1 \cdot s \equiv \exists s. \top \\
  \top \sqsubseteq \leq 1 \cdot s \equiv \text{Func}(s)
  \]
Nominals or Closed Classes

- A **nominal** for an individual $a$ is a concept of the form $\{a\}$; it denotes a class with a single element (denoted by $a$).
- **Closed class** with multiple elements expressed as union of nominals; $\{a_1\} \sqcup \ldots \sqcup \{a_n\}$ can also be abbreviated as $\{a_1, \ldots, a_n\}$.
- Semantics defined as follows (the special case $n = 1$ covers nominals):
  
  - **DL**: $\left(\{a_1, \ldots, a_n\}\right)^I = \{a_1^I, \ldots, a_n^I\}$
  
  - **FOL**: $\pi_x(\{a_1, \ldots, a_n\}) = (x = a_1 \lor \ldots \lor x = a_n)$

- Extension of $\mathcal{ALC}$ by nominals denoted as $\mathcal{ALCO}$
- Corresponds to $\text{owl:oneOf}$
Nominals for Encoding Further OWL Constructors

- **owl:hasValue** "forces" role to a certain individual

  ```xml
  <owl:Class rdf:ID="Woman">
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasGender"/>
        <owl:hasValue rdf:resource="#female"/>
      </owl:Restriction>
    </owl:equivalentClass>
  </owl:Class>
  
  - In description logic:

    \[ \text{Woman} \equiv \exists \text{hasGender. \{female\}} \]
Further Kinds of ABox Assertions

ABox assertions can have one of the following forms:

- $C(a)$ (concept assertion)
- $r(a, b)$ (role assertion)
- $\neg r(a, b)$ (negative role assertion)
- $a \approx b$ (equality assertion)
- $a \not\approx b$ (inequality assertion)
Further Kinds of ABox Assertions

ABox assertions can have one of the following forms:

- \( C(a) \) (concept assertion)
- \( r(a, b) \) (role assertion)
- \( \neg r(a, b) \) (negative role assertion)
- \( a \approx b \) (equality assertion)
- \( a \not\approx b \) (inequality assertion)
Internalization of ABox Assertions

If nominals are supported, every knowledge base with an ABox can be transformed into an equivalent one without ABox:

\[ C(a) \equiv \{a\} \sqsubseteq C \]
\[ r(a, b) \equiv \{a\} \sqsubseteq \exists r.\{b\} \]
\[ \neg r(a, b) \equiv \{a\} \sqsubseteq \forall r.\neg \{b\} \]
\[ a \approx b \equiv \{a\} \equiv \{b\} \]
\[ a \not\approx b \equiv \{a\} \sqsubseteq \neg \{b\} \]
Overview Nomenclature

\( ALC \) Attribute Language with Complement

\( S \) \( ALC \) + role transitivity

\( \mathcal{H} \) subroles

\( \mathcal{O} \) nominals

\( \mathcal{I} \) inverse roles

\( \mathcal{N} \) (unqualified) number restrictions

\( (D) \) datatypes

\( \mathcal{F} \) functional roles

OWL DL is \( SHOIN(D) \) and OWL Lite is \( SHIF(D) \)
Different Terms in DLs and in OWL

<table>
<thead>
<tr>
<th>OWL</th>
<th>DL</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>concept</td>
</tr>
<tr>
<td>property</td>
<td>role</td>
</tr>
<tr>
<td>object property</td>
<td>abstract role</td>
</tr>
<tr>
<td>data property</td>
<td>concrete role</td>
</tr>
<tr>
<td>oneOf*</td>
<td>nominal</td>
</tr>
<tr>
<td>ontology</td>
<td>knowledge base</td>
</tr>
<tr>
<td>–</td>
<td>TBox, RBox, ABox</td>
</tr>
</tbody>
</table>

*) with one individual
Example: A More Complex Knowledge Base

Human $\sqsubseteq$ Animal $\sqcap$ Biped
Man $\equiv$ Human $\sqcap$ Male
Male $\sqsubseteq$ $\neg$Female
\{President\_Obama\} $\equiv$ \{Barack\_Obama\}
\{john\} $\sqsubseteq$ $\neg$\{peter\}
hasDaughter $\sqsubseteq$ hasChild
hasChild $\equiv$ hasParent$^-$
cost $\equiv$ price
Trans(ancestor)
Func(hasMother)
Func(hasSSN$^-$)
Open versus Closed World Assumption

**OWA** Open World Assumption
- the existence of further individuals is possible, if they are not explicitly excluded
- OWL uses the OWA

**CWA** Closed World Assumption
- it is assumed that the knowledge base contains all individuals and facts
Open versus Closed World Assumption

OWA  Open World Assumption
– the existence of further individuals is possible, if they are not explicitly excluded
– OWL uses the OWA

CWA  Closed World Assumption
– it is assumed that the knowledge base contains all individuals and facts

Are all of Bill’s children male?    DL answers    Prolog answers

\(\text{child(bill, bob)}\) \(\models ? \ (\forall \text{child.Man})(\text{bill})\)

\(\leq 1 \text{ child}(\text{bill})\) \(\models ? \ (\forall \text{child.Man})(\text{bill})\)
Open versus Closed World Assumption

OWA  Open World Assumption

– the existence of further individuals is possible, if they are not explicitly excluded
– OWL uses the OWA

CWA  Closed World Assumption

– it is assumed that the knowledge base contains all individuals and facts

Are all of Bill’s children male?  DL answers Prolog answers

| child(bill, bob) | Man(bob) | \( \models ? \ (\forall \text{ child.Man})(\text{bill}) \) | don’t know |

| \( \leq 1 \text{ child})(\text{bill}) | \( \models ? \ (\forall \text{ child.Man})(\text{bill}) \) |
### Open versus Closed World Assumption

**OWA  Open World Assumption**
- the existence of further individuals is possible, if they are not explicitly excluded
- OWL uses the OWA

**CWA  Closed World Assumption**
- it is assumed that the knowledge base contains all individuals and facts

<table>
<thead>
<tr>
<th>Are all of Bill’s children male?</th>
<th>DL answers</th>
<th>Prolog answers</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>child(bill, bob)</code> <code>Man(bob)</code></td>
<td>⎕? <code>(∀ child.Man)(bill)</code></td>
<td>don’t know      yes</td>
</tr>
<tr>
<td><code>(≤ 1 child)(bill)</code></td>
<td>⎕? <code>(∀ child.Man)(bill)</code></td>
<td></td>
</tr>
</tbody>
</table>
Open versus Closed World Assumption

OWA Open World Assumption
- the existence of further individuals is possible, if they are not explicitly excluded
- OWL uses the OWA

CWA Closed World Assumption
- it is assumed that the knowledge base contains all individuals and facts

<table>
<thead>
<tr>
<th>Are all of Bill’s children male?</th>
<th>DL answers</th>
<th>Prolog answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>child(bill, bob) Man(bob)</td>
<td>=? (∀ child.Man)(bill)</td>
<td>don’t know yes</td>
</tr>
<tr>
<td>(≤ 1 child)(bill)</td>
<td>=? (∀ child.Man)(bill)</td>
<td>yes</td>
</tr>
</tbody>
</table>

TU Dresden, 16 May 2014 Foundations of Semantic Web Technologies
Open versus Closed World Assumption

**OWA  Open World Assumption**
- the existence of further individuals is possible, if they are not explicitly excluded
- OWL uses the OWA

**CWA  Closed World Assumption**
- it is assumed that the knowledge base contains all individuals and facts

<table>
<thead>
<tr>
<th>Are all of Bill’s children male?</th>
<th>DL answers</th>
<th>Prolog answers</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>child(bill, bob)</code></td>
<td><code>Man(bob)</code></td>
<td>don’t know</td>
</tr>
<tr>
<td><code>\leq 1 child(bill)</code></td>
<td><code>\forall child. Man(bill)</code></td>
<td>yes</td>
</tr>
</tbody>
</table>
Agenda

- Motivation
- Introduction Description Logics
- The Description Logic $\mathcal{ALC}$
- Extensions of $\mathcal{ALC}$
- Inference Problems
Important Inference Problems for a Knowledge Base $\mathcal{K}$

- **global consistency of the knowledge base:** $\mathcal{K} \models \text{false}\? \mathcal{K} \models \text{T} \subseteq \bot\?$
  - Is the knowledge base “plausible”?  
- **class consistency:** $\mathcal{K} \models C \subseteq \bot\?$
  - Is the class $C$ necessarily empty?  
- **class inclusion (subsumption):** $\mathcal{K} \models C \subseteq D\?$
  - Taxonomic structure of the knowledge base  
- **class equivalence:** $\mathcal{K} \models C \equiv D\?$
  - Do two classes comprise the same individual sets?  
- **class disjointness:** $\mathcal{K} \models C \cap D \subseteq \bot\?$
  - Are two classes disjoint?  
- **class membership:** $\mathcal{K} \models C(a)\?$
  - Is the individual $a$ contained in class $C$?  
- **instance retrieval:** find all $x$ with $\mathcal{K} \models C(x)$
  - Find all (known!) members of the class $C$.  

Decidability of OWL DL

- decidability means that there is a terminating algorithm for all the aforementioned inference problems
- OWL DL is a fragment of FOL, thus FOL inference procedures could be used in principle (Resolution, Tableaux)
  - but these are not guaranteed to terminate!
- problem: find algorithms that are guaranteed to terminate
- no “naive” solutions for this
OWL 2: Outlook

- OWL 2 extends the fragments introduced here by further constructors
- OWL 2 also defines simpler fragments (PTime for standard inferencing problems)
- diverse tools for automated inferencing
- editors support creation of ontologies / knowledge bases
Further Literature

- Description Logic Primer
  A gentle introduction to DLs and pointers to further literature
  Available at http://arxiv.org/abs/1201.4089