• OWL: Overview and Rationale
• „Basic“ OWL Syntax and Semantics
  – Axioms
  – Constructors
• Modeling
• Tableaux-Based Reasoning
• **Web Ontology Language**
  – W3C Recommendation for the Semantic Web, 2004
  – OWL 2 (revised W3C Recommendation) forthcoming in 2009
    – we already present this here

• **Semantic Web KR language based on description logics (DLs)**
  – OWL DL is essentially DL SROIQ(D)
  – KR for web resources, using URIs.
  – Using web-enabled syntaxes, e.g. based on XML or RDF
  – We present RDF Turtle syntax
• Open World Assumption
• Favourable trade-off between expressivity and scalability
• Integrates with RDFS
• Purely declarative semantics

Features (for OWL 2 DL):
• Fragment of first-order predicate logic (FOL)
• Decidable
• Known complexity classes (N2ExpTime for OWL 2 DL)
• Reasonably efficient for real KBs
OWL Building Blocks

• individuals (written as URIs): ex:markus
  – aka: constants (FOL), resources (RDF)
• classes (also written as URIs): ex:Female
  – aka: concepts, unary predicates (FOL)
• properties (also written as URIs): ex:married
  – aka: roles (DL), binary predicates (FOL)
• model theory (aka extensional semantics)
• OWL DL Interpretation $\mathcal{I}$:
• OWL statements are written down as (sets of) RDF triples
• OWL facts (aka: assertions) are written down like in RDF
• some RDF language elements are reused
• new language elements from the OWL namespace
• more complex statements are constructed by using bnodes (we “hide” them for convenience)
Class Membership

- \text{induri} \text{ rdf:type classuri .}
- true in \mathcal{I}, \text{ if } I_I(\text{induri}) \in I_C(\text{classuri})
- Example:

  \text{ex:nicolas rdf:type ex:Male}
• induri1 propuri induri2.
• true in $\mathcal{I}$, if $\langle I_I(\text{induri1}), I_I(\text{induri2}) \rangle \in I_P(\text{propuri})$
• Example:
  \[
  \text{ex:carla} \text{ ex:marriedWith } \text{ex:nicolas}
  \]
Class Inclusion

- \texttt{classuri1} \texttt{rdfs:subClassOf} \texttt{classuri2}.
- true in \( \mathcal{I} \), if \( I_C(\texttt{classuri1}) \subseteq I_C(\texttt{classuri2}) \)
- Example:
  \texttt{ex:President rdfs:subClassOf ex:Politician}

• propuri1 $\text{rdfs:subPropertyOf}$ propuri2.
• true in $\mathcal{I}$, if $I_R(\text{propuri1}) \subseteq I_R(\text{propuri2})$
• Example:
  \[ \text{ex:sonOf} \text{rdfs:subPropertyOf} \text{ex:childOf} \]
• **owl:Thing** – the class containing everything
  – \( I_C(\text{owl:Thing}) = \Delta \)

• **owl:Nothing** – the empty class
  – \( I_C(\text{owl:Nothing}) = \emptyset \)

• **owl:topProperty** – the property connecting everything
  – \( I_R(\text{owl:topProperty}) = \Delta \times \Delta \)

• **owl:bottomProperty** – the empty property
  – \( I_R(\text{owl:bottomProperty}) = \emptyset \)
Complex Classes: Intersection

• \( [\text{owl:intersectionOf} (\text{class1}, \ldots, \text{classn})] \)
• \( I_C([\text{owl:intersectionOf} (\text{class1}, \ldots, \text{classn})]) = I_C(\text{class1}) \cap \ldots \cap I_C(\text{classn}) \)
• Example:
  \( [\text{owl:intersectionOf} (\text{ex:Actor}, \text{ex:Politician})] \)
Complex Classes: Union

- \([\text{owl:unionOf}(\text{class1}, \ldots, \text{classn})]\)
- \(I_C([\text{owl:unionOf}(\text{class1}, \ldots, \text{classn})]) = I_C(\text{class1}) \cup \ldots \cup I_C(\text{classn})\)

Example:

\([\text{owl:unionOf}(\text{ex:Actor}, \text{ex:Politician})]\)
Complex Classes: Complement

- $\text{[owl:complementOf class]}
- I_C([\text{owl:complementOf class}]) = \Delta - I_C(\text{class})$
- Example:
  $\text{[owl:complementOf ex:Politician]}$

Complex Classes: Existential Property Restriction

- \[ \text{[ rdf:type } \text{owl:Restriction } \text{; owl:onProperty } \text{prop } \text{; owl:someValuesFrom } \text{class } \}\]
- \( I_C(...) = \{ x | \langle x, y \rangle \in I_R(prop) \text{ for some } y \in I_C(class) \} \)
- **Example:** \[ \text{[ rdf:type } \text{owl:Restriction } \text{; owl:onProperty } \text{ex:parentOf } \text{; owl:someValuesFrom } \text{ex:Male} ]\]
Complex Classes: Universal Property Restriction

- \[
  \begin{aligned}
  &\text{[ rdf:type } \text{owl:Restriction ;} \\
  &\text{  owl:onProperty } prop ; \\
  &\text{  owl:allValuesFrom } class ]
  \end{aligned}
\]

- \[ I_C(...) = \{ x | \langle x, y \rangle \in I_R(prop) \implies y \in I_C(class) \} \]

- **Example:** \[
  \begin{aligned}
  &\text{[ rdf:type } \text{owl:Restriction ;} \\
  &\text{  owl:onProperty } ex:parentOf ; \\
  &\text{  owl:allValuesFrom } ex:Male] \\
  \end{aligned}
\]
Alternative Semantics: Translation into FOL

\[
\text{trans(} \text{induri} \ \text{rdf:type} \ \text{class}. \) \\
= \text{trans}_x(\text{class})[x/\text{induri}]
\]

\[
\text{trans(} \text{induri}_1 \ \text{propuri} \ \text{induri}_2. \) \\
= \text{propuri}(\text{induri}_1, \text{induri}_2)
\]

\[
\text{trans(} \text{class}_1 \ \text{rdfs:subClassOf} \ \text{class}_2. \) \\
= \forall x: \text{trans}_x(\text{class}_1) \rightarrow \text{trans}_x(\text{class}_2)
\]

\[
\text{trans(} \text{propuri}_1 \ \text{rdfs:subPropertyOf} \ \text{propuri}_2. \) \\
= \forall x,y: \text{propuri}_1(x,y) \rightarrow \text{propuri}_2(x,y)
\]

Alternative Semantics: Translation into FOL

\[\text{trans}_x(\text{owl:Thing}) = \text{true} \quad \text{trans}_x(\text{owl:Nothing}) = \text{false}\]

\[\text{trans}_x(\text{classuri}) = \text{classuri}(x)\]

\[\text{trans}_x([\text{owl:intersectionOf} (\text{class}_1, \ldots, \text{class}_n)]) = \text{trans}_x(\text{class}_1) \land \ldots \land \text{trans}_x(\text{class}_n)\]

\[\text{trans}_x([\text{owl:unionOf} (\text{class}_1, \ldots, \text{class}_n)]) = \text{trans}_x(\text{class}_1) \lor \ldots \lor \text{trans}_x(\text{class}_n)\]

\[\text{trans}_x([\text{owl:complementOf} \text{class}]) = \neg \text{trans}_x(\text{class})\]

\[\text{trans}_x([\text{rdf:type} \text{owl:Restriction} ; \text{owl:onProperty} \text{propuri} ; \text{owl:someValuesFrom} \text{class} ]) = \exists y: \text{propuri}(x,y) \land \text{trans}_y(\text{class})\]

\[\text{trans}_x([\text{rdf:type} \text{owl:Restriction} ; \text{owl:onProperty} \text{propuri} ; \text{owl:someValuesFrom} \text{class} ]) = \forall y: \text{propuri}(x,y) \rightarrow \text{trans}_y(\text{class})\]
class1 owl:disjointWith class2.

• same as:
  [owl:intersectionOf (class1, class2)]
  rdfs:subClassOf owl:Nothing.

propuri rdf:domain class.

• same as:
  [rdf:type owl:Restriction;
   owl:onProperty propuri;
   owl:someValuesFrom owl:Thing ] rdfs:subClassOf class.

propuri rdf:range class.

• same as:
  owl:Thing rdfs:subClassOf [rdf:type owl:Restriction;
   owl:onProperty propuri;
   owl:allValuesFrom class].
A simple modeling example

ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .

Healthy beings are not dead.

ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .

Every cat is alive or dead.

ex:owns rdfs:subPropertyOf ex:caresFor .

If somebody owns something, (s)he cares for it.

ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [ rdf:type owl:Restriction ; owl:onProperty ex:owns ;
    owl:someValuesFrom ex:Cat],
  [ rdf:type owl:Restriction ; owl:onProperty ex:caresFor ;
    owl:someValuesFrom ex:Healthy]
)] .

A happy cat owner owns a cat and all beings he cares for are healthy.

ex:schrödinger rdf:type ex:HappyCatOwner .

Schrödinger is a happy cat owner.
Behind the scenes...

- ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf ( [ rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat], [ rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:someValuesFrom ex:Healthy] )].
Given a knowledge base KB, we might want to know:
- whether the knowledge in KB is consistent,
- whether KB entails a class membership
  (e.g. ex:schrödinger rdf:type ex:Alive.),
- whether a class is (un)satisfiable
  (e.g. [owl:intersectionOf ( ex:Dead , ex:Alive)]),
- whether KB entails a subclass statement
  (e.g. ex:Alive rdfs:subClassOf ex:Healthy.),
- etc.
Many inference problems can be reduced to knowledge base consistency checking.

Technique: claim the opposite and look what happens...

**Class membership:**

KB entails

\[ \text{ex:schrödinger rdf:type ex:Alive} \]

iff adding

\[ \text{ex:schrödinger rdf:type [owl:complementOf ex:Alive]} \]

to KB makes it inconsistent.
• Many inference problems can be reduced to knowledge base consistency checking.
• Technique: claim the opposite and look what happens...

• **Class (un)satisfiability:**
  KB entails unsatisfiability of
  
  \[
  \text{[owl:intersectionOf ( ex:Dead , ex:Alive)]}
  \]
  iff adding
  
  ex:n rdf:type [owl:intersectionOf ( ex:Dead , ex:Alive)].
  to KB makes it inconsistent.
Reducing Inference Problems

- Many inference problems can be reduced to knowledge base consistency checking.
- Technique: claim the opposite and look what happens...

- **Subclass entailment:**
  
  KB entails
  
  \[
  \text{ex:Alive rdfs:subClassOf ex:Healthy}.
  \]
  
  iff adding
  
  \[
  \text{ex:n rdf:type [owl:intersectionOf}
  \[
  \text{( ex:Alive, [owl:complementOf ex:Healthy])].}
  \]
  
  to KB makes it inconsistent.
• But how to determine whether a KB is consistent?
• One option: translate to FOL and use standard methods.
• But: OWL is decidable while FOL isn’t.
• Still: FOL inferencing techniques (tableaux, resolution, type elimination) can be turned into decision procedures for OWL.
• Tableaux methods are most frequent.
• Basic idea: try to build a model of the given KB. If this fails, the KB is inconsistent, otherwise consistent.
• Warning! The following example is simplified for better presentation (but demonstrates the essential features of tableaux-based methods). Consult the literature for a comprehensive treatment.
| ex:Healthy | rdfs:subClassOf [owl:complementOf ex:Dead]. |
| ex:Cat     | rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)]. |
| ex:owns    | rdfs:subPropertyOf ex:caresFor. |
| ex:HappyCatOwner | rdfs:subClassOf [owl:intersectionOf ( |
|             | [ rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat], |
|             | [ rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:someValuesFrom ex:Healthy ] ) ]. |
| ex:schrödinger | rdf:type ex:HappyCatOwner. |
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (]
   [ rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
   [ rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:someValuesFrom ex:Healthy]) ].
ex:schrödinger rdf:type ex:HappyCatOwner.

Knowledge Base

Tableau
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (ex:owns rdfs:subPropertyOf ex:caresFor; owl:someValuesFrom ex:Healthy; ex:owns rdfs:subPropertyOf ex:caresFor; owl:someValuesFrom ex:Healthy)].
ex:schrödinger rdf:type ex:HappyCatOwner.
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (  
    [ rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],  
    [ rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:someValuesFrom ex:Healthy]) ] .
ex:schrödinger rdf:type ex:HappyCatOwner .

Knowledge Base

Tableau
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (]
  [ rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
  [ rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:someValuesFrom ex:Alive]) ].
ex:schrödinger rdf:type ex:HappyCatOwner.
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf(
    [ rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
    [ rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:someValuesFrom ex:Healthy])].
ex:schrödinger rdf:type ex:HappyCatOwner.

Knowledge Base

Tableau
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
    [ rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],
    [ rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:someValuesFrom ex:Healthy])].
ex:schrödinger rdf:type ex:HappyCatOwner.
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (ex:owns; owl:someValuesFrom ex:Cat), (ex:caresFor; owl:someValuesFrom ex:Healthy)].
ex:schrödinger rdf:type ex:HappyCatOwner.

Knowledge Base

Tableau
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (  
  [ rdf:type owl:Restriction; owl:onProperty ex:owns; owl:someValuesFrom ex:Cat],  
  [ rdf:type owl:Restriction; owl:onProperty ex:caresFor; owl:someValuesFrom ex:Healthy]) ].
ex:schrödinger rdf:type ex:HappyCatOwner.
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (ex:owns, ex:caresFor; owl:someValuesFrom ex:Cat), (ex:owns, ex:caresFor; owl:someValuesFrom ex:Healthy)] .
ex:schrödinger rdf:type ex:HappyCatOwner .
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (ex:Dead, ex:Alive)].

Knowledge Base

ex:schrödinger rdf:type ex:HappyCatOwner.

Tableau

ex:owns ex:caresFor

ex:HappyCatOwner [owl:intersectionOf (ex:Dead, ex:Alive)]

ex:Cat ex:Healthy [owl:unionOf (ex:Dead, ex:Alive)]

ex:owns ex:caresFor

ex:HappyCatOwner [owl:intersectionOf (ex:Dead, ex:Alive)]

ex:Cat ex:Healthy ex:Dead [owl:complementOf ex:Dead]

ex:owns ex:caresFor

ex:HappyCatOwner [owl:intersectionOf (ex:Dead, ex:Alive)]

ex:Cat ex:Healthy ex:Alive [owl:unionOf (ex:Dead, ex:Alive)]
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (ex:Dead, ex:Alive)].
ex:schrödinger rdf:type ex:HappyCatOwner.
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (ex:Dead, ex:Alive)].
ex:schrödinger rdf:type ex:HappyCatOwner.

Knowledge Base

Tableau
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (ex:owns, ex:caresFor)] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (ex:owns, ex:caresFor) .
   [rdf:type owl:Restriction ; owl:onProperty ex:owns ; owl:someValuesFrom ex:Cat],
   [rdf:type owl:Restriction ; owl:onProperty ex:caresFor ; owl:someValuesFrom ex:Healthy])] .
ex:schrödinger rdf:type ex:HappyCatOwner .

Knowledge Base
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead].
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)].
ex:owns rdfs:subPropertyOf ex:caresFor.
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (ex:owns; owl:someValuesFrom ex:Cat), (ex:caresFor; owl:someValuesFrom ex:Healthy)].
ex:schrödinger rdf:type ex:HappyCatOwner.

Knowledge Base

Satisfiable

Tableau
• possible termination problem:

\[
\text{ex:schrödinger rdf:type ex:Person . ex:Person rdfs:subClassOf [ rdf:type owl:Restriction ; owl:onProperty ex:hasParent ; owl:someValuesFrom ex:Person].}
\]

• can be solved by blocking
References

• W3C OWL Working Group, OWL 2 Web Ontology Language: Document Overview. http://www.w3.org/TR/owl2-overview/

• Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter Patel-Schneider, Sebastian Rudolph, OWL 2 Web Ontology Language: Primer. http://www.w3.org/TR/owl2-primer/
