• Web Ontology Language
  – W3C Recommendation for the Semantic Web, 2004
  – OWL 2 (currently Proposed W3C Recommendation) forthcoming this October
    • 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 as identifiers
  – Using web-enabled syntaxes, e.g. based on XML or RDF
    • We mostly use concise DL syntax, some RDF syntax examples
  – Many technical and extra-logical aspects, e.g. datatypes
    • We focus on the logical core language
OWL Rationale

An ontology language for the Web ...

- Open World Assumption
- Reasonable trade-off between expressivity and scalability
- Integrates with RDF and RDF Schema
- Fully 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 $\mathcal{I}$:

\[\Delta \subset \mathcal{I}, \mathcal{I}_I(\text{uri}), \mathcal{I}_C(\text{uri}), \mathcal{I}_R(\text{uri})\]
On the OWL Syntax

- 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

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

\[
\text{ex:nicolas rdf:type ex:Male}
\]

\( I_I \)

\( I_C \)
Property Membership

- \( \text{induri}_1 \text{ propuri } \text{induri}_2 \).
- true in \( \mathcal{I} \), if 
  \[ \langle I_1(\text{induri}_1), I_1(\text{induri}_2) \rangle \in I_p(\text{propuri}) \]
- Example:
  ex:carla ex:marriedWith ex:nicolas
Class Inclusion

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

- \( \text{propuri1} \text{ rdfs:subPropertyOf } \text{propuri2} \).
- True in \( \mathcal{I} \), if \( I_R(\text{propuri1}) \subseteq I_R(\text{propuri2}) \).
- Example:
  - \( \text{ex:sonOf} \text{ rdfs:subPropertyOf } \text{ex:childOf} \)
Predefined Classes & Properties

- **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

- \([\text{owl:intersectionOf}(\text{class}_1, \ldots, \text{class}_n)]\)
- \(I_C([\text{owl:intersectionOf}(\text{class}_1, \ldots, \text{class}_n)]) = I_C(\text{class}_1) \cap \ldots \cap I_C(\text{class}_n)\)
- Example:
  - \([\text{owl:intersectionOf}(\text{ex:Actor}, \text{ex:Politician})]\)
Complex Classes: Union

- $[\text{owl:unionOf}(\text{class}_1, \ldots, \text{class}_n)]$
- $I_C([\text{owl:unionOf}(\text{class}_1, \ldots, \text{class}_n)])$
  $= I_C(\text{class}_1) \cup \ldots \cup I_C(\text{class}_n)$

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

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

- \[ [ \text{rdf:type} \quad \text{owl:Restriction} ; \\]
  \[ \text{owl:onProperty} \quad prop ; \]
  \[ \text{owl:someValuesFrom} \quad \text{class} \ ] \]

- \( I_C(...)=\{x|\langle x,y\rangle \in I_R(prop) \text{ for some } y \in I_C(class)\} \)

- Example: \[ [ \text{rdf:type} \quad \text{owl:Restriction} ; \\]
  \[ \text{owl:onProperty} \quad \text{ex:parentOf} ; \]
  \[ \text{owl:someValuesFrom} \quad \text{ex:Male} \] \]
Complex Classes: Universal Property Restriction

- \( [ \text{rdf:type} \, \text{owl:Restriction} ; \, \text{owl:onProperty} \, prop ; \, \text{owl:allValuesFrom} \, \text{class} ] \)
- \( I_C(...) = \{ x | \langle x, y \rangle \in I_R(prop) \text{ implies } y \in I_C(class) \} \)
- Example: \( [ \text{rdf:type} \, \text{owl:Restriction} ; \, \text{owl:onProperty} \, \text{ex:parentOf} ; \, \text{owl:allValuesFrom} \, \text{ex:Male} ] \)
Syntactic Sugar: Disjointness, Domain & Range Statements

\( \text{class1} \ \text{owl:disjointWith} \ \text{class2} \).

- same as:
  \[ \text{[owl:intersectionOf (class1, class2)]} \]
  \( \text{rdfs:subClassOf} \ \text{owl:Nothing} \).

\( \text{propuri} \ \text{rdf:domain} \ \text{class} \).

- same as:
  \[ \text{[rdf:type owl:Restriction; owl:onProperty propuri; owl:someValuesFrom owl:Thing]} \]
  \( \text{rdfs:subClassOf} \ \text{class} \).

\( \text{propuri} \ \text{rdf:range} \ \text{class} \).

- same as:
  \( \text{owl:Thing} \ \text{rdfs:subClassOf [rdf:type owl:Restriction; owl:onProperty propuri; owl:allValuesFrom class]} \).
• Advanced Features of OWL
  – more class constructors
  – extended property modeling
  – handling of data values
More Complex Classes: Qualified At-Least Restriction

- \[ \text{[ rdf:type owl:Restriction; owl:minQualifiedCardinality } "n"^\text{xsd:nonNegativeInteger}; owl:onProperty } prop; owl:onClass } class \] 

- \( I_C(...)=\{x| \#\{\langle x,y\rangle \in I_R(prop) | y \in I_C(class)\} \geq n\} \)

- Example:
  \[ \text{[ rdf:type owl:Restriction; owl:minQualifiedCardinality } "2"^\text{xsd:nonNegativeInteger}; owl:onClass } ex:Male; owl:onProperty } ex:parentOf \]
More Qualified Cardinalities

• in analogy to at-least restrictions:
  – at-most:
    \texttt{owl:maxQualifiedCardinality}
  – exact cardinality:
    \texttt{owl:QualifiedCardinality}
More Complex Classes: Enumeration of Individuals

- \([\text{owl:oneOf} \ (\text{induri1}, \ldots, \text{indurin})]\)
- \(I_C([\text{owl:oneOf} \ (\text{induri1}, \ldots, \text{indurin})]) = \{I_I(\text{induri1}), \ldots, I_I(\text{indurin})\}\)
- Example:
  \([\text{owl:oneOf} \ (\text{ex:georgec}, \text{ex:arnolds})]\)
More Complex Classes: Self Restriction

- \[
  \text{[ rdf:type } \text{owl:Restriction} ; \text{owl:onProperty } prop ; \text{owl:hasSelf } "true"^{xsd:boolean} ]}
- \[I_C(...) = \{ x | \langle x, x \rangle \in I_R(prop) \} \]
- Example: \[
  \text{[ rdf:type } \text{owl:Restriction} ; \text{owl:onProperty } ex:hasKilled ; \text{owl:hasSelf } "true"^{xsd:boolean} ]}
\]
Inverse Properties

• \([\text{owl:inverseOf } prop]\)
• \(I_R([\text{owl:inverseOf } prop]) = \{\langle y, x \rangle | \langle x, y \rangle \in I_R(prop)\}\)
• Example: \([\text{owl:inverseOf ex:childOf}]\)
Property Chain Axioms

- \( prop \ quad \text{owl:propertyChainAxiom} ( \text{prop1}, \ldots, \text{propn}) \).
- true in \( \mathcal{I} \), if \( I_R(\text{prop1}) \circ \ldots \circ I_R(\text{propn}) \subseteq I_R(\text{prop}) \)
- Example:

\( \text{ex:siblingOf} \quad \text{owl:propertyChainAxiom} \quad (\text{ex:childOf}, \text{ex:parentOf}) \).
Decidability problems

- property chain axioms can easily lead to undecidability
- in order to retain decidability, two global constraints are imposed on OWL DL ontologies:
  - the set of property chain axioms and subproperty statements must be regular
  - properties used in cardinality and self restrictions must be simple properties
in the following, we abbreviate

\[ R \text{owl:propertyChainAxiom}(S_1 \ldots S_n) \text{ by } S_1 \circ \ldots \circ S_n \sqsubseteq R \]

\[ S \text{owlrdfs:subPropertyOf} R \text{ by } S \sqsubseteq R \]

regularity restriction:

– there must be a strict linear order \( \prec \) on the properties

– every property chain or subproperty axiom has to have one of the following forms where \( S_i \prec R \) for all \( i = 1, 2, \ldots, n \):

\[
R \circ R \sqsubseteq R \quad [\text{owl:inverseOf } R] \sqsubseteq R \\
R \circ S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R \\
S_i \circ S_2 \circ \ldots \circ S_n \circ R \sqsubseteq R
\]

– Example 1: \( R \circ S \sqsubseteq R \quad S \circ S \sqsubseteq S \quad R \circ S \circ R \sqsubseteq T \)

– regular with order \( S \prec R \prec T \)

– Example 2: \( R \circ T \circ S \sqsubseteq T \)

– not regular because form not admissible

– Example 3: \( R \circ S \sqsubseteq S \quad S \circ R \sqsubseteq R \)

– not regular because no adequate order exists

Property Chain Axioms:

Regularity
• combining property chain axioms and cardinality or self restrictions may lead to undecidability
• restriction: use only *simple* properties in cardinality expressions (i.e. those which cannot be – directly or indirectly – inferred from property chains)
• technically:
  – for any property chain axiom $S_1 \circ S_2 \circ ... \circ S_n \sqsubseteq R$ with $n>1$, $R$ is non-simple
  – for any subproperty axiom $S \sqsubseteq R$ with $S$ non-simple, $R$ is non-simple
  – all other properties are simple
• Example:

$Q \circ P \sqsubseteq R \quad R \circ P \sqsubseteq R \quad R \sqsubseteq S \quad P \sqsubseteq R \quad Q \sqsubseteq S$

non-simple: $R, S$  simple: $P, Q$
Property Characteristics

- OWL also allows for specifying that properties are:
  - disjoint from another
  - functional
  - inverse functional
  - transitive
  - symmetric
  - asymmetric
  - reflexive
  - irreflexive

  \[\text{syntactic sugar w.r.t.} \]

  \[\text{already introduced} \]

  \[\text{modeling features} \]
Datatypes in OWL

- like in RDF, properties can also be used to associate individuals with data values:
  
  ex:john ex:hasAge "42"^^xsd:integer.

- those datatype properties must not be used as individual-interrelating object properties at the same time.

- datatypes supported by OWL:
  
Simple Data Integration in OWL

• Practical problem: given ontologies from different sources, which identifiers refer to the same individuals?

• Typical approaches in OWL:
  – Explicitly specify equality (owl:sameAs)
  – Use inverse functional properties (“same values → same individual”)

• Problems:
  – equality requires explicit mappings (rare on the Web)
  – OWL DL disallows inverse functional datatype properties (complicated interplay with datatype definitions!)
  – Only one property used globally for identification, no property combinations (Example: “All Informatik 2009 participants with the same name and birthday are the same.”)
OWL 2 Keys

OWL 2 provides a way to model
“All Informatik 2009 participants with same name and birthday are the same.“

→ **Keys** (expressed with `owl:hasKey`)

- **Restriction**: Keys apply only to named individuals – objects of the interpretation domain to which a constant symbol refers.
- This is not an expressive feature of description logics!
Other OWLs

• OWL 1 contained three “species” of OWL:
  – **OWL DL**: a DL-based KR language with an RDF syntax
    • not all RDF documents are OWL DL ontologies
  – **OWL Lite**: a restricted version of OWL DL
  – **OWL Full**: an extension of RDF to give semantics to the OWL keywords
    • intended to behave “similar” to OWL DL but applicable to all RDF documents
    • entailment problem undecidable (if the semantics is non-contradictory)

• OWL 2: OWL 2 DL and OWL 2 Full to extend OWL 1 species
Quo Vadis, OWL Lite?

**OWL Lite as failure:**

- Defined as fragment of OWL 1 DL, intended to be simpler
- However: almost as complex as OWL DL (ExpTime)
- Complex syntax hides real expressive power
- Current usage in ontologies coincidental rather than intentionally

Original goal: simpler implementation and usage

→ approach in OWL 2: three simpler **language profiles:**

- **OWL 2 QL**
- **OWL 2 EL**
- **OWL 2 RL**
OWL 2 Profiles

Original goal: simpler implementation and usage

→ approach in OWL 2: three simpler language profiles:
  OWL 2 QL    OWL 2 EL    OWL 2 RL

Design principle for profiles:
  Identify maximal OWL 2 sublanguages that are still implementable in Ptime.

Main source of intractability: non-determinism (requires guessing/backtracking)

• disjunction, or negation + conjunction
• Max. cardinality restrictions
• Combining existentials and universals in superclasses
• Non-unary finite class expressions (nominals) or datatype expressions (not discussed here)

→ features that are not allowed in any OWL 2 profile

Many further features can lead to non-determinism – care needed!
OWL 2 EL

OWL profile based on description logic EL++

- Intuition: focus on terminological expressivity used for light-weight ontologies
- Allow existential but not universal, only \texttt{rdfs:range} (special kind of universals) allowed with restrictions
- Property domains, class/property hierarchies, class intersections, disjoint classes/properties, property chains, \texttt{Self}, nominals (singleton classes), and keys fully supported
- No inverse or symmetric properties, no disjunctions or negations
OWL 2 EL: Features

- Standard reasoning in OWL 2 EL: PTime-complete
- Used by practically relevant ontologies: Prime example is SNOMED CT (clinical terms ontology with classes and properties in the order of $10^5$)
- Fast implementations available: full classification of SNOMED-CT in <1 min; real-time responsivity when preprocessed (modules)
OWL profile that resembles an OWL-based rule language:

- Intuition: subclass axioms in OWL RL can be understood as rule-like implications with head (superclass) and body (subclass)

- Different restrictions on subclasses and superclasses:
  - subclasses can only be class names, nominals, conjunctsions, disjunctions, existentials if applied only to subclass-type expressions
  - superclasses can be class names, universals or nominals; also max. cardinalities of 0 or 1 are allowed, all with superclass-type filler expressions only

- Property domains and ranges only for subclass-type expressions; property hierarchies, disjointness, inverses, (a)symmetry, transitivity, chains, (inverse)functionality, irreflexivity fully supported

- Disjoint classes and classes in keys need subclass-type expressions, equivalence only for expressions that are sub- and superclass-type, no restrictions on equality
OWL 2 RL: Features

- Standard reasoning in OWL 2 RL: PTime-complete
- Rule-based reading simplifies modelling and implementation:
  even naïve implementations can be useful
- Fast and scalable implementations exist

Also: possibly useful for combining OWL with rules
**OWL 2 QL**

**OWL profile that can be used to query data-rich applications:**

- **Intuition:** use OWL concepts as light-weight queries, allow query answering using rewriting in SQL on top of relational DBs
- **Different restrictions on subclasses and superclasses:**
  - subclasses can only be class names or existentials with unrestricted (⊤) filler
  - superclasses can be class names, existentials or conjunctions with superclass filler (recursive), or negations with subclass filler
- **Property hierarchies, disjointness, inverses, (a)symmetry supported, restrictions on range and domain**
- **Disjoint or equivalence of classes only for subclass-type expressions**
- **No disjunctions, universals, Self, keys, nominals, equality, property chains, transitive properties, cardinalities, or functional properties**
OWL 2 QL: Features

- Standard reasoning in OWL 2 QL: PTime, instance retrieval even LogSpace (actually $\text{AC}_0$) w.r.t. size of data
- Convenient light-weight interface to legacy data
- Fast implementations on top of legacy database systems (relational or RDF): highly scalable to very large datasets
Do We Really Need So Many OWLs?

Three new OWL profiles with somewhat complex descriptions … why not just one?

- The union of any two of the profiles is no longer light-weight!
  QL+RL, QL+EL, RL+EL all ExpTime-hard
- Restricting to fewer profiles = giving up useful feature combinations
- Rationale: profiles are “maximal” (well, not quite) well-behaved fragments of OWL 2
  → Pick suitable feature set for applications
- In particular, nobody is forced to implement all of a profile
OWL in Practice: Tools

- Editors ([http://semanticweb.org/wiki/Editors](http://semanticweb.org/wiki/Editors))
  - Most common editor: Protégé 4
  - Other tools: TopBraid Composer ($), NeOn toolkit
  - Special purpose apps, esp. for light-weight ontologies (e.g. FOAF editors)
- Reasoners ([http://semanticweb.org/wiki/Reasoners](http://semanticweb.org/wiki/Reasoners))
  - OWL DL: Pellet, HermiT, FaCT++, RacerPro ($)
  - OWL EL: CEL, SHER, snorocket ($), ELLY (extension of IRIS)
  - OWL RL: OWLIM, jena, Oracle OWL Reasoner (part of O11g) ($),
  - OWL QL: Owlgres, QuOnto, Quill
- Many tools use the OWL API library (Java)
- Note: many other Semantic Web tools are found online
Non-standard Reasoning in OWL

There is more to do than editing and inferencing:

• **Explanation**: reasoning task of providing axiom sets to explain a conclusion (important for editing and debugging)

• ** Conjunctive querying**: check entailment of complex query patterns

• **Modularisation**: extract sub-ontologies that suffice for (dis)proving a certain conclusion

• **Repair**: determine ways to repair inconsistencies (related to explanation)

• **Least Common Subsumer**: assuming that class unions are not available, find the smallest class expression that subsumes two given classes

• **Abduction**: given an observed conclusion, derive possible input facts that would lead to this conclusion

• …

→ All implemented, tasks on top common in standard tools today
Summary and Outlook

• OWL: an expressive ontology language with practical impact
• Structurally representable in RDF
• Reasoning typical based on extensional ("direct") semantics:
  – closely related to description logics and first-order logic (with equality)
  – different from RDF semantics, but compatible for many purposes
• Various flavours for different applications:
  – OWL Full provides RDF-based semantics (undecidable)
  – OWL DL decidable but complex (N2ExpTime)
  – OWL profiles for light-weight reasoning (in PTime)

Version 2 of the Web Ontology Language almost complete:

Official specification expected by Oct 2009
Further Reading

- P. Hitzler, S. Rudolph, M. Krötzsch: *Foundations of Semantic Web Technologies*. CRC Press, 2009. (Chapter 4 and 5 closely related to this lecture)

Selected research articles:

- F. Baader, S. Brandt, C. Lutz: *Pushing the EL envelope*. In Proc. of the 19th Joint Int. Conf. on Artificial Intelligence (IJCAI 2005), 2005. (paper introducing description logic EL++ underlying OWL EL)