# Web Ontology Language OWL – Part II

## VL Semantic Technologies

Bernd Neumayr
(with contributions from Dieter Steiner)

Department for Business Informatics – Data & Knowledge Engineering

based on: OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation 11 December 2012
http://www.w3.org/TR/owl-primer

# OWL 2 Part II – Agenda

1. Basic Notions: Modeling and Reasoning with OWL
2. Advanced Use of Properties
3. Data Ranges and Datatypes
4. Miscellaneous
5. OWL Language Variants: Syntaxes, Semantics, Profiles

# Basic Notions: Modeling and Reasoning with OWL

- Structure of OWL Ontologies
- Semantics of OWL – Basic Intuition

# OWL is a knowledge representation language

- formulate, exchange, and reason with knowledge about a domain of interest
- a general-purpose modeling language for certain parts of human knowledge
- a part of the Semantic Web: entities have IRIs as names

# Structure of OWL Ontologies

# OWL 2 Ontologies

An OWL 2 ontology is a formal description of a domain of interest and mainly consists of three different syntactic categories:

- **Entities**, such as classes, properties, and individuals, are identified by IRIs. They form the primitive terms of an ontology and constitute the basic elements of an ontology.
- **Expressions** represent complex notions in the domain being described. For example, a class expression describes a set of individuals in terms of the restrictions on the individuals' characteristics.
- **Axioms** are statements that are asserted to be true in the domain being described.

http://www.w3.org/TR/owl2-syntax/

**Axioms = Invariants**
statements that are always true with absolutely no exceptions

# Entities: atomic constituents of statements

- Individuals
  `John, Mary`
- Classes
  `Person, Father`
- Object properties
  `marriedTo, childOf`
- Data properties
  `hasAge, name`
- Literals
  `"27"^^xsd:integer`
- Datatypes
  `xsd:integer, personAgeInYears`

# Expressions: combinations of entities

- expressions can be seen as new entities and used where entities are used
- rich language for class expressions
  `(Man or Woman), (hasChild some Person)`
- weak language for property expressions
  `(inverse hasChild)`

# Axioms: statements that are true

OWL 2 ontologies are collections of axioms:

- Class expression axioms

```
Class: Woman
  EquivalentTo: Person and Female
```

- Object property axioms

```
ObjectProperty: hasHusband
  SubPropertyOf: hasSpouse
```

- Assertions

```
Individual: John
  Types: Person
```

- ...

**Axioms = Invariants**
Statements that are always true
with absolutely no exceptions.
All axioms are equal (no overriding ...)

# Semantics of OWL – Basic Intuition

# Interpreting an OWL ontology

Ontology $O$:

```
Class: Person
Individual: Mary
    Types: Person
Individual: John
```

# Interpreting an OWL ontology

Ontology $O$:

```
Class: Person
Individual: Mary
    Types: Person
Individual: John
```

Vocabulary $V$ of $O$:

```
Classes = {Person}
Individuals = { Mary, John }
```

# Interpreting an OWL ontology

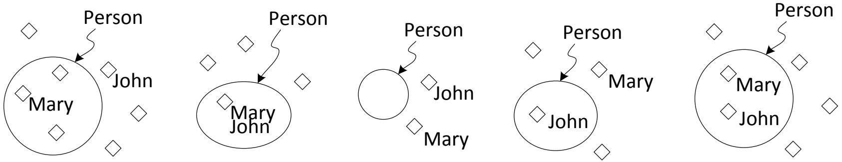
Ontology $O$:

```
Class: Person
Individual: Mary
    Types: Person
Individual: John
```

Vocabulary $V$ of $O$:

```
Classes = {Person}
Individuals = { Mary, John }
```

Some interpretations of vocabulary $V$:



# Interpreting an OWL ontology

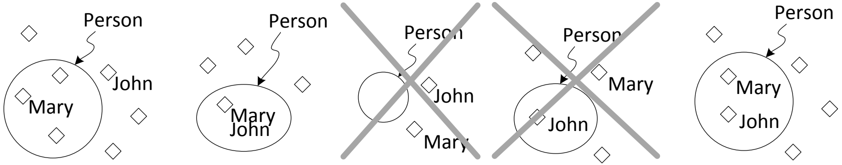
Ontology $O$:

```
Class: Person
Individual: Mary
    Types: Person
Individual: John
```

Vocabulary $V$ of $O$:

```
Classes = {Person}
Individuals = { Mary, John }
```

Some interpretations of vocabulary $V$ satisfying ontology $O$:
(models of $O$, possible state of affairs, possible worlds)



# OWL Reasoning Tasks

## automated by OWL reasoners

- Ontology Consistency (Satisfiability)
- Class Expression Satisfiability
- Subsumption Checking
- Query Answering (over incomplete data)
- ...

# Ontology consistency (satisfiability)

## Is there a possible world?

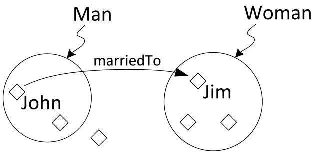
- An ontology is consistent (satisfiable), if there is a possible state of affairs (possible world, model)
- inconsistent, if there is no such state of affairs
- formal semantics of OWL specifies, in essence, for which possible "states of affairs" a particular set of OWL statements is true.

# Ontology consistency (satisfiability)

## Is there a possible world?

Consistent Ontology:

```
Class: Man
  SubClassOf:
    marriedTo only Woman
DisjointClasses: Man, Woman
Individual: John
  Types: Man
  Facts: marriedTo Jim
Individual: Jim
```




Inconsistent Ontology:

```
Class: Man
  SubClassOf:
    marriedTo only Woman
DisjointClasses: Man, Woman
Individual: John
  Types: Man
  Facts: marriedTo Jim
Individual: Jim
  Types: Man
```

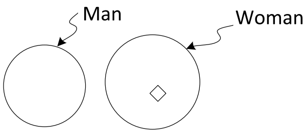
# Class Satisfiability

## Can we construct a world where the class has a member?

`Person` is satisfiable:

```
Class: Man
  DisjointWith: Woman

Class: Person
  EquivalentTo: Man or Woman
```




`Person` is not satisfiable:

```
Class: Man
  DisjointWith: Woman

Class: Person
  EquivalentTo: Man and Woman
```

# Entailment

- when humans think, they draw consequences from their knowledge
- OWL captures this aspect of human intelligence for the forms of knowledge it can represent
- An ontology entails a statement if this statement is true in every possible world
- OWL reasoners automatically compute consequences: Class Hierarchies, Class Membership, Assertions, . . .

# Subsumption Checking

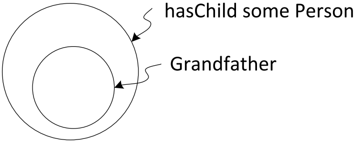
## Inferring the class hierarchy

A class $A$ subsumes another class $B$ if in every possible world, all members of $B$ are also members of $A$.

```
ObjectProperty: hasChild
  Domain: Person

Class: Grandfather
  EquivalentTo: Man and hasChild some
                       (hasChild some Person)
```

Does `hasChild some Person` subsume `Grandfather`?



# Open World Query Answering

## What are the common members of a class in every possible world?

Can we conclude that John has a daughter?

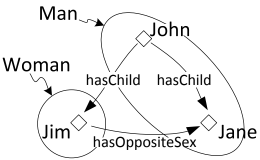
```
Class: Person
  EquivalentTo: Man or Woman
Class: Woman
  SubClassOf: hasOppositeSex only Man
Class: Man
  SubClassOf: hasOppositeSex only Woman
DisjointClasses: Woman, Man
ObjectProperty: hasOppositeSex
ObjectProperty: hasChild
  Domain: Person
  Range: Person
Individual: John
  Types: Man
  Facts: hasChild  Jane,
         hasChild  Jim
Individual: Jim
  Facts: hasOppositeSex Jane
Individual: Jane
```

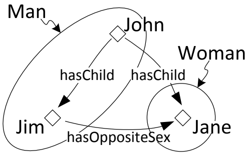
# Open World Query Answering

## What are the common members of a class in every possible world?

Can we conclude that John has a daughter?

```
Class: Person
  EquivalentTo: Man or Woman
Class: Woman
  SubClassOf: hasOppositeSex only Man
Class: Man
  SubClassOf: hasOppositeSex only Woman
DisjointClasses: Woman, Man
ObjectProperty: hasOppositeSex
ObjectProperty: hasChild
  Domain: Person
  Range: Person
Individual: John
  Types: Man
  Facts: hasChild  Jane,
         hasChild  Jim
Individual: Jim
  Facts: hasOppositeSex Jane
Individual: Jane
```



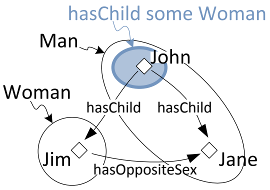



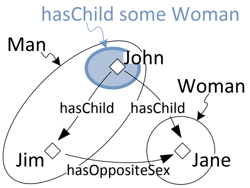
# Open World Query Answering

## What are the common members of a class in every possible world?

Can we conclude that John has a daughter?

```
Class: Person
  EquivalentTo: Man or Woman
Class: Woman
  SubClassOf: hasOppositeSex only Man
Class: Man
  SubClassOf: hasOppositeSex only Woman
DisjointClasses: Woman, Man
ObjectProperty: hasOppositeSex
ObjectProperty: hasChild
  Domain: Person
  Range: Person
Individual: John
  Types: Man
  Facts: hasChild  Jane,
         hasChild  Jim
Individual: Jim
  Facts: hasOppositeSex Jane
Individual: Jane
```



# Interaction between Axioms

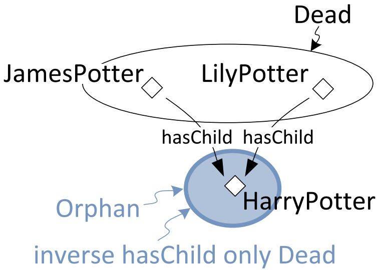
- the way ontological axioms interact can be very subtle and difficult to understand
- this is good, because:
  - OWL 2 tools can discover information that a person would not have spotted
  - allows to model more directly
  - system provides useful feedback and critique of the modeling
- this is bad, because:
  - difficult for humans to foresee the effect of various constructs in various combinations

# Advanced Use of Properties

- Property Expressions
- Property Characteristics
- Property Chains
- Keys

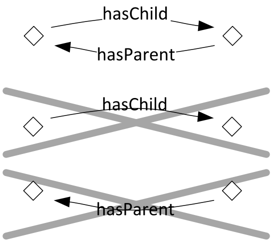
# Inverse Properties in Class Expressions

```
Class: Orphan
    EquivalentTo: inverse hasChild only Dead
```



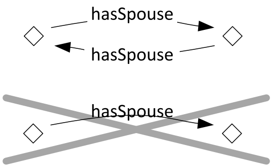
# Inverse Properties in Property Definitions

```
ObjectProperty: hasParent
  InverseOf: hasChild
```



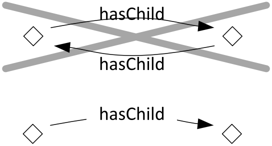
# Symmetric Properties

```
ObjectProperty: hasSpouse
  Characteristics: Symmetric
```



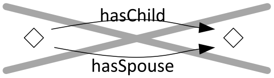
# Asymmetric Properties

```
ObjectProperty: hasChild
  Characteristics: Asymmetric
```



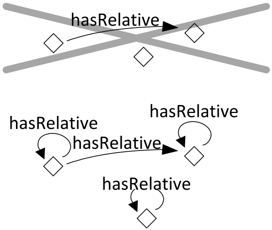
# Disjoint Properties

DisjointProperties: hasChild, hasSpouse



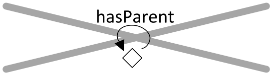
# Reflexive Properties

```
ObjectProperty: hasRelative
  Characteristics: Reflexive
```



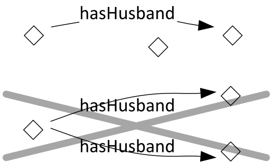
# Irreflexive Properties

```
ObjectProperty: parentOf
  Characteristics: Irreflexive
```



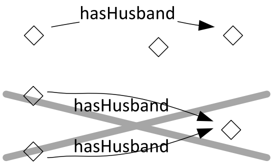
# Functional Properties

```
ObjectProperty: hasHusband
  Characteristics: Functional
```



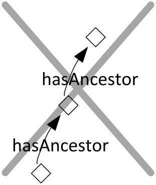
# Inverse Functional

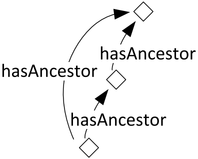
```
ObjectProperty: hasHusband
  Characteristics: InverseFunctional
```



# Transitive Properties

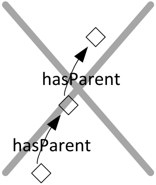
```
ObjectProperty: hasAncestor
  Characteristics: Transitive
```

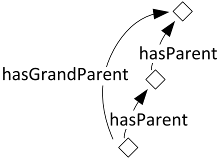


# Property Chains

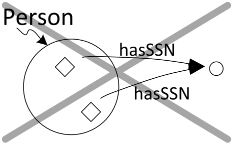
```
ObjectProperty: hasGrandparent
  SubPropertyChain: hasParent o hasParent
```



# Keys

```
Class: Person
  HasKey: hasSSN
```



# Data Ranges and Datatypes

# Datatypes

OWL 2 ontologies can refer to data values such as strings or integers. Each kind of such values is called a datatype. Each datatype is identified by an IRI and is defined by the following components:

- The value space is the set of values of the datatype. Elements of the value space are called data values.
- The lexical space is a set of strings that can be used to refer to data values. Each member of the lexical space is called a lexical form, and it is mapped to a particular data value.
- The facet space is a set of pairs of the form ( F , v ) where F is an IRI called a constraining facet, and v is an arbitrary data value called the constraining value. Each such pair is mapped to a subset of the value space of the datatype.

# Datatypes Example – xsd:integer

## value space

the set of all integers

## lexical space

integer has a lexical representation consisting of a finite-length sequence of one or more decimal digits with an optional leading sign. If the sign is omitted, “+” is assumed.
For example: -1, 0, 12678967543233, +100000.

## facet space (constraining facets)

`xsd:minInclusive`, `xsd:maxInclusive`,
`xsd:minExclusive`, `xsd:maxExclusive`

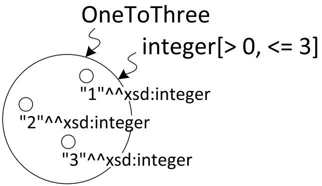
# Data Ranges and Datatype Definitions

- Data Range

```
integer[> 0, <= 3]
```

- Datatype Definition

```
Datatype: OneToThree
    EquivalentTo: integer[> 0, <= 3]
```



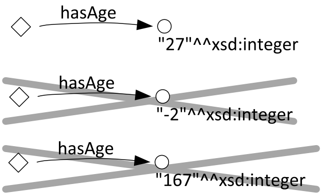
# Applying Data Ranges in Class Expressions

```
Class: Teenager
  EquivalentTo: Person and
       hasAge some integer[> 12 , <= 19]
```

# Defining Custom Datatypes

```
Datatype: PersonAgeInYears
  EquivalentTo: integer[>= 0 , <= 150]

DataProperty hasAge
  Domain: Person
  Range: PersonAgeInYears
```



# Miscellaneous

- Metamodeling
- Annotating Axioms and Entities
- Anonymous Individuals

# Metamodeling

An IRI can be used in an OWL 2 ontology to refer to more than one type of entity. In such cases, the entities that share the same IRI should be understood as different “views” of the same underlying notion identified by the IRI.

```
Individual: John
  Types: Father
Individual: Father
  Types: SocialRole
```

# Annotations

Annotations are used when the information attached to entities should not be considered a part of the domain and when it should not contribute to the logical consequences of an ontology.

```
Class: Person
  Annotations: rdfs:comment
            "Represents the set of all people."
```

# Anonymous Individuals

If an individual is not expected to be used outside a particular ontology, one can use an anonymous individual, which is identified by a local node ID rather than a global IRI. Anonymous individuals are analogous to blank nodes in RDF.

```
Individual: John
  Facts: livesAt  _:a1

Individual: _:a1
  Facts: city  Linz,
         state Austria
```

# OWL Language Variants: Syntaxes, Semantics, Profiles

- OWL Syntaxes
- OWL 2 DL vs OWL 2 Full
- OWL Profiles

# OWL Syntaxes

- Manchester Syntax
  easier for humans to read and write
- Description Logics Syntax
  for brevity, used in scientific publications
- Functional Syntax
  easier to see formal structure of ontologies
- OWL/XML
  easier to process with XML tools
- RDF/XML
  for interchange, mandatory for conformant OWL systems

# Manchester Syntax

```
Class: Woman
  EquivalentTo: Female and Person

Individual: Susan
    Types: Female, Person
```

# Description Logics Syntax

$\mathtt{Woman} \equiv \mathtt{Person} \sqcap \mathtt{Female}$
$\mathtt{Female(Susan)}$
$\mathtt{Person(Susan)}$

# Functional Syntax

```
EquivalentClasses(
    :Woman
    ObjectIntersectionOf(:Person :Female)
)
ClassAssertion(:Female :Susan)
ClassAssertion(:Person :Susan)
```

# OWL/XML

```
<EquivalentClasses>
  <Class IRI="#Woman"/>
  <ObjectIntersectionOf>
    <Class IRI="#Female"/>
    <Class IRI="#Person"/>
  </ObjectIntersectionOf>
</EquivalentClasses>

<ClassAssertion>
  <Class IRI="#Female"/>
  <NamedIndividual IRI="#Susan"/>
</ClassAssertion>
```

# RDF/XML

```
<owl:Class rdf:about="http://...#Woman">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://...#Female"/>
        <rdf:Description rdf:about="http://...#Person"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>

<owl:NamedIndividual rdf:about="http://...#Susan">
    <rdf:type rdf:resource="http://...#Female"/>
    <rdf:type rdf:resource="http://...#Person"/>
</owl:NamedIndividual>
```

# OWL 2 DL vs OWL 2 Full

## OWL 2 DL (Direct Semantics)

- allows sound and complete reasoning
- fully supported by OWL reasoners like HermiT
- based on Description Logic $\mathcal{SROIQ}$

## OWL 2 Full (RDF based Semantics)

- some extra inferences
- based on RDF semantics

# OWL Profiles

## language limitations that simplify reasoning

### OWL 2 EL

tractable reasoning over huge TBoxes, especially for large ontologies in the health domain

### OWL 2 QL

very efficient query answering (rewriting to SQL)

### OWL 2 RL

efficient reasoning, can be implemented on top of rule engines (e.g., Jess, Jena's Rule Engine, . . . )

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