Introduction to the Semantic Web Technologies

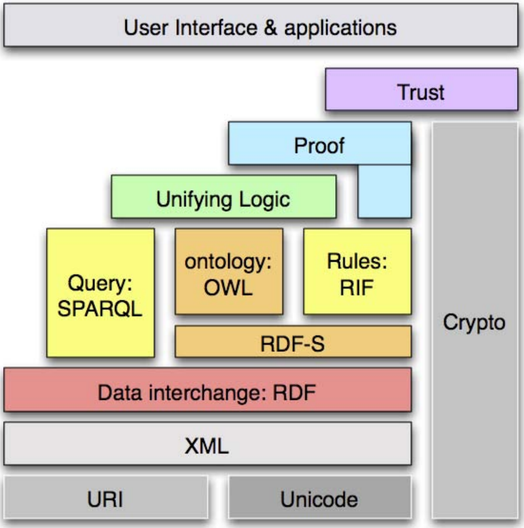
Kostyantyn Shchekotykhin

Alpen-Adria-Universität
Klagenfurt, Austria

Agenda

1. Semantic Modeling with OWL
   - General concepts
   - Boolean class constructors
   - Definition of Restrictions
2. Debugging ontologies
   - Basic definitions
   - Debugging methodology

Semantic Web Technology Stack



Capturing Knowledge for the Web

- People capture their knowledge about the world in form of some abstractions or models
- Models can be captured and exchanged in various ways
- Semantic Web aims at creation of infrastructure enabling people to formalize and share their knowledge

  Requirement people and computers should be able to collaborate on models and interpret them unambiguously

  Solution definition of expressive knowledge representation languages with formal semantics based on XML
- Ontologies and their elements are identified using Internationalized Resource Identifiers (IRIs)

Classes, Properties and Individuals

Classes are sets of individuals that share common properties
`Digicam`

Individuals represent actual objects `NikonD3`,
`NikonD3` **Types** `Digicam`

Propoerties are used to express relations between individuals
`Nikkor35 canBeUsedWith NikonD3`
and literals `Nikkor35 hasFocalLength 35`

In addition one can define domain and range of a property

Example

`canBeUsedWith` **Range** `Digicam`
`canBeUsedWith` **Domain** `Lens`

Subsumption and Equivalence

- Hierarchies of classes can be defined by using subsumption, i.e. **SubClassOf** relation
- All individuals of a subclass belong to a set of individuals of a super class

Example

`DSLR` **SubClassOf** `Digicam`
`ConsumerDSLR` **SubClassOf** `DSLR`

- Equivalent classes have the same set of individuals

Example

`LowendDSLR` **EquivalentTo** `ConsumerDSLR`

Conjunction, Disjunction and Negation I

Complex class expressions can be created using conjunction, disjunction and negation

Example: Conjunction

`ConsumerDSLR` **SubClassOf** `DSLR and CheapProduct`

Example: Disjunction

`BrandDSLR` **SubClassOf** `NikonDSLR or CanonDSLR`

Example: Negation

`ProfessionalDSLR` **SubClassOf** `not CheapProduct`

Conjunction, Disjunction and Negation II

Conjunction and disjunction have commutativity, associativity and distributivity properties

Example: Commutativity

`ConsumerDSLR` **SubClassOf** `DSLR and CheapProduct`
`ConsumerDSLR` **SubClassOf** `CheapProduct and DSLR`

Example: Distributivity

`SLR` **SubClassOf** `Camera and (Digital or Film)`
`SLR` **SubClassOf** `(Camera and Digital) or (Camera and Film)`

Conjunction, Disjunction and Negation III

De Morgan's laws can be applied to convert class expressions into the negation normal form

Example: De Morgan's Laws

`DSLR` **SubClassOf** `not (not Digital or Film)`
`DSLR` **SubClassOf** `Digital and not Film`

Problems with Negation

Careless usage of negation might result in inconsistencies

Example: Inconsistent Ontology

`ProfessionalCamera` **SubClassOf** `Digicam`
`NikonN90` **Types** `ProfessionalCamera`
`NikonN90` **Types** `not Digicam`

In the following case domain of the `ProfessionalCamera` camera is empty, i.e. there is no camera that is a professional one.

Example: Incoherent Theory

`ProfessionalCamera` **SubClassOf** `Digicam and FilmCamera`
`Digicam` **SubClassOf** `not ProfessionalCamera`

Class `ProfessionalCamera` is unsatisfiable

Disjointness

OWL supports declaration of disjoint classes and properties

Example: Disjoint Classes

`Digicam` **DisjointWith** `FilmCamera`

Example: Disjoint Properties

`compatibleLens` **DisjointWith** `incompatibleLens`

Individuals can also be different one from each other

Example: Different Individuals

`NikonD3` **DifferentFrom** `NikonD3X`

Problems with Disjointness

Just as with the negation overuse of disjointness can cause inconsistencies

Example: Inconsistent Ontology

`ProfessionalCamera` **SubClassOf** `Digicam and FilmCamera`
`Digicam` **DisjointWith** `FilmCamera`
`NikonN90` **Types** `ProfessionalCamera`

- `ProfessionalCamera` is incoherent, i.e. no individual can belong to this class
- `NikonN90` is of the type `ProfessionalCamera`
- Consequently, the ontology is inconsistent

Existential Quantification I

- `C` **SubClassOf** `P some R`
  restricts a class C to include only individuals for which at least one value of property P is of the class R

Example

SLR contains a mirror

`SLR` **SubClassOf** `contains some Mirror`

Example

Every digital camera has a sensor and a viewfinder

`Digicam` **SubClassOf** `(has some Sensor) and (has some Viewfinder)`

Existential Quantification II

Example

Some part of a digital camera is a viewfinder and a sensor at the same time

`Digicam` **SubClassOf** `has some (Sensor and Viewfinder)`

If one will add `Sensor` **DisjointWith** `Viewfinder` to the ontology defined above it will become inconsistent

Example

Digital cameras have some sensor or some viewfinder

`Digicam` **SubClassOf** `has some (Sensor or Viewfinder)`

Universal Quantification I

- `C` **SubClassOf** `P only R`
  restricts a class C to include only individuals for all values of property P is of the class R

Example

DSLR capture images using a sensor

`DSLR` **SubClassOf** `captureImages only Sensor`

Universal Quantification II

Incorrect Usage of Universal Quantification

DSLR includes a sensor

`DSLR` **SubClassOf** `includes only Sensor`

This restriction is satisfied in two cases

- for all `DSLR` individuals all values of the `includes` property are of the `Sensor` type
- for all `DSLR` individuals `includes` property has no values

Which one is correct?

SLR capture images using a sensor or a film

`SLR` **SubClassOf** `captureImages only (Sensor or Film)`

`SLR` **SubClassOf** `captureImages only Sensor or captureImages only Film`

Negation can be used with restrictions just as in FOL

Example

DSLR is a camera that cannot image using films or anything that is not a sensor

`Digicam` **SubClassOf** `Camera and not imagesUsing some (Film or not Sensor)`

Example

DSLR is a camera that can image only using sensor but not films

`Digicam` **SubClassOf** `Camera and imagesUsing only (not Film and Sensor)`

Negation and Disjointness with Restrictions

Restrictions should be used carefully if their class repressions include disjoint classes and/or negation

Example

DSLR is a camera that cannot image using films or anything that is not a sensor

`Digicam` **SubClassOf** `Camera and imagesUsing only (Film or not Sensor)`

Example

DSLR is a camera that can image only using sensor but not films

`Digicam` **SubClassOf** `Camera and imagesUsing only (not Film and Sensor)`

Cardinality Constraints

In some case one would like to count individuals fulfilling a requirement

OWL includes three types of cardinality restrictions: min , max and exactly

Example

A camera should have at least one flash

`Camera` **SubClassOf** `has min 1 Flash`

Example

A digital camera that can read exactly one type of either SD or CF cards is a consumer camera

`ConsumerDigicam` **SubClassOf** `Digicam and canRead exactly 1 (SDCard or CFCard)`

Advanced Properties I

In OWL one can define characteristics of properties and their relationships

Example: Transitive Properties

Camera resolution is a transitive property

`resolution` **Transitive**

Example: Inverse Properties

Cameras are owned by persons and persons own cameras

`owns` **InverseOf** `ownedBy`

Advanced Properties II

Properties can have sub-properties

Example

`hasNikonFMount` **SubPropertyOf** `hasLensMount`

Properties can be chained, i.e. a property exists any time its chain exists

Example

A camera has a flash if it has a hot shoe, which is compatible with the flash.

`hasFlash` **SubPropertyChain** `hasHotShoe o compatible`

`CameraA hasFlash FlashF` if

    `CameraA hasHotShoe HS` and `HS compatible FlashF`

ALPEN-ADRIA UNIVERSITÄT KLAGENFURT | WIEN GRAZ

A minimal conflict is an irreducible set of axioms of an ontology that are inconsistent

Example: Sample Ontology

ax_1: `ProfCamera` **SubClassOf** `Digicam and FilmCamera`
ax_2: `Digicam` **SubClassOf** `Camera and has some Sensor`
ax_3: `NikonN90` **Types** `ProfCamera`

- Let the sample ontology be extended with the axiom
  ax_4: `Digicam` **DisjointWith** `FilmCamera`
- assume also that axiom ax_4 is correct (background knowledge)

Example: Sample Ontology

ax_1: `ProfCamera` **SubClassOf** `Digicam and FilmCamera`
ax_2: `Digicam` **SubClassOf** `Camera and has some Sensor`
ax_3: `NikonN90` **Types** `ProfCamera`
ax_4: `Digicam` **DisjointWith** `FilmCamera`

Required	ontology should be consistent
Minimal Conflict	ax_1 ax_3
Solution	Remove/modify ax_1 or ax_3

A minimal diagnosis is an irreducible set of axioms, which have to be removed/modified in order to fulfill the requirements

Which diagnosis $[ax_1]$ or $[ax_3]$ is the target one?

We need tests, just as in software development!

Example: Entailed Test

We might require the target ontology not to entail
N: `ProfCamera` **SubClassOf** `FilmCamera`

- Removing ax_3 does not help since the resulting ontology entails N

Example: Sample Ontology

ax_1: `ProfCamera` **SubClassOf** `Digicam and FilmCamera`
ax_2: `Digicam` **SubClassOf** `Camera and has some Sensor`
ax_3: `NikonN90` **Types** `ProfCamera`
ax_4: `Digicam` **DisjointWith** `FilmCamera`

Debugging methodology II

- Removing ax_1 solves the problem and fulfills the test N

Example: Sample Ontology

ax_1: `ProfCamera` **SubClassOf** `Digicam and FilmCamera`
ax_2: `Digicam` **SubClassOf** `Camera and has some Sensor`
ax_3: `NikonN90` **Types** `ProfCamera`
ax_4: `Digicam` **DisjointWith** `FilmCamera`

- A possible correction would be to replace conjunction with disjunction
  `ProfCamera` **SubClassOf** `Digicam or FilmCamera`

- The corrected ontology might look as follows:

Example: Sample Ontology

ax_1: `ProfCamera` **SubClassOf** `Digicam or FilmCamera`
ax_2: `Digicam` **SubClassOf** `Camera and has some Sensor`
ax_3: `NikonN90` **Types** `ProfCamera`
ax_4: `Digicam` **DisjointWith** `FilmCamera`