Ontology Design: Semantics and OWL

Aldo Gangemi
Lecture 2 @ Corso Dottorato 2011
Dipartimento di Informatica, Bologna, Italy
Natural and Formal Languages

- Formal interpretation gives us a precise way to establish what we are talking about, and therefore to provide reliable automated inferences when needed.
- Natural language is able to describe very different types of facts with the same structure, for example ...
... different types of facts ...

- Wile buys from ACME [ground fact]
- ACME has been reported for abusive discharge [reported fact]
- Wile is blonde [attributive fact]
- Wile is a coyote [classification fact]
- To discharge is to release someone from a job [meaning fact]
- To discharge is a transitive verb [terminological fact]
- Discharge is nine characters long [information fact]
- Discharge is a class [formal fact]
- Discharge can be abusive in Italy [contextual fact]
- Discharge represents a failure [interpretive fact]
Liberality of RDF

• You can declare any fact in RDF, even “unusual”, “counterintuitive”, or “abnormal” ones:
  ? Wile is blonde but also brunette
  ? ACME has been fired by ACME
  ? ACME fired the third character of Wile
  ? ACME is a class

• Liberality is great, but has costs: no way to make a machine detect undesired facts
What semantics?

- **Linguistic semantics**
  - From NLP/IR technologies: advanced search, information extraction, automatic tagging, text classification, ...
  - Mainly informal background knowledge
- **Formal semantics**
  - From SW/AI technologies: data modelling, reengineering, linking, automated reasoning via query, rules, classification, ...
  - Ontologies as background knowledge
- **Trend towards hybridization**
  - SW used as background knowledge in NLP/IR
  - Linguistic semantics extracted by NLP/IR tools, and “reconstructed” in an enriched SW
  - Advanced reasoning performed in the enriched SW
From NL and RDF to FL

• The functionality of natural language and RDF cannot be easily reproduced for machine interpretation
• The formal interpretation of OWL can do something
• More is provided by best practices
• Some arbitrariness is unavoidable
  –Requirements, requirements, requirements
OWL

- Web Ontology Language
- Formal semantics over RDF
- Large use of small vocabularies
  - FOAF, SIOC, SKOS, VoID, content patterns
- **OWL representation and reasoning** is the next step at a web scale, but already a reality for intrawebs and small domains
  - Automatic concept classification, consistency and coherence checking, materialization of knowledge
OWL constraints

• In order to limit (and to guide) the design of ontologies, OWL restricts the expressivity of RDF.
• No more is any triple allowed, but only those that respect the constraints of OWL formal semantics (this lecture).
• Undesired facts will then be detected, if the design of the ontology reflects the conceptualization of the users.
• That’s why we need good design (lecture 3).
Formal interpretation (extensional)

$sellsTo^I \subseteq \text{Thing}^I \times \text{Thing}^I$
$\text{Coyote}^I \subseteq \text{Thing}^I$
$\text{Company}^I \subseteq \text{Thing}^I$
$\text{Company}^I \cap \text{Coyote}^I = \emptyset$
$\text{Wile}^I \in \text{Coyote}^I$
$\text{ACME}^I \in \text{Company}^I$
$(\text{ACME}, \text{Wile})^I \in sellsTo^I$
OWL exercise I: types, subclasses, and inheritance

> Wile is a Coyote
> ACME is a Company
> Company is a class
> Company is a subclass of Institution
> Institution is a subclass of Organization
> Organization is a Concept
Inheritance

• Notice that some inferences are now allowed

- ACME is a company; companies are institutions
- ACME is an institution

Company is a subclass of Institution; Institution is a subclass of Organization
- Company is a subclass of Organization

• This is called inheritance
Prevented inheritance

• But some inferences are prevented

\[ \text{ACME is a Company; Company is an (OWL) class} \]
\[ \not \equiv \text{ACME is an (OWL) class} \]
Lesson learnt

• In OWL we must distinguish when “is a” means “rdf:type”, and when it means “rdfs:subClassOf”
• There are patterns to deal with hybrid entities (individual/class)
• OWL2 (like RDFS) allows more freedom
  – Aldo rdf:type Tutor
  – Tutor rdf:type Role
• But beware of sense-making!
OWL exercise II: disjointness and consistency

use Radon or Pellet to check the ontology after each addition

> *Companies are not coyotes

• In formal languages, we can usually state when two classes are disjoint
• This provides greater clarity, and works fine for formal checking, i.e. “coherency” or “consistency”
Consistency

• Equipped with the owl:disjointWith axiom, we can now exclude the following:
  ➢ Wile is a company
  ➢ ACME is a coyote

Since they would make an ontology inconsistent
Coherence

• We can also exclude the following:
  ➢ Coyompanies are coyotes and companies
Since it would make the ontology *incoherent*
OWL exercise III:
*domain, ranges, inverses, subproperties*

> Companies sell to coyotes
> Selling to someone implies buying from someone
> Wile buys from ACME
> Buying implies getting
Materialization

• Some new inferences are now allowed:
  
  \[ Wile \text{ buys from ACME} \]
  \[ \vdash ACME \text{ sells to Wile} \]
  
  \[ Wile \text{ buys from ACME} \]
  \[ \vdash Wile \text{ gets from ACME} \]

• This is called \textit{materialization}
Classification

*Wile chases Beep; (as far as we know) only predators prey; chases is a kind of preying*

\[ \vdash \text{Wile is a predator} \]

*This is called *classification*
Open World Assumption

• Since the Web is an open world, if we say something that is not explicitly put in our axioms, we cannot exclude it, and then we have to add a new axiom, e.g. a disjointness axiom, to obtain an “integrity check”:

\[ \text{Wile sells to ACME; ACME buys from Wile} \]
\[ \neg \text{Wile is also a company, and ACME is also a coyote} \]

• But this generates an inconsistency, since Coyote and Company are disjoint classes
OWL exercise IV: sameness and difference

• Due to the open world assumption, it is important to explicitly declare if any two individuals are the same or are different (when this is known)
  > Wile is the same as “The funny coyote”
  > Wile is different from Beep
Advanced modelling: restrictions and definitions in OWL

• The formal semantics of OWL is currently based on so called *description logics*. An important feature of description logics is the ability to declare “unnamed” constructs that are interpreted as classes

• ---) changing example domain to “cetaceans”
OWL exercise V: boolean class constructors

> Mammals that are also aquatic organisms
> Aquatic mammals, fishes or crustaceans
> Non-fishes
OWL exercise VI: 
*relational class constructors*

> Things that *only* live in a marine habitat
  - livesIn *only* MarineHabitat

> Things that live in *at least one* marine habitat
  - livesIn *some* MarineHabitat

> Things that live in the Indian Ocean
  - livesIn *only* {IndianOcean}

> Things that live in *either* the Indian *or* Pacific Ocean
  - livesIn *only* {IndianOcean PacificOcean}

> Things that have taxonomic species “Monodon monoceros”
  - livesIn *value* MonodonMonoceros
OWL exercise VII: “enumerating” class constructors

- The class of the following things: Indian, Pacific, and Atlantic Oceans
  - \{IndianOcean PacificOcean AtlanticOcean\}
OWL exercise VIII

equivalence and automated subsumption/classification

• Differently from other conceptual modeling languages, OWL allows to state formal equivalence between two classes

• Equivalence works implicitly with class constructors
  – ontology APIs generate internal classes that are equivalent to class constructors

• but equivalence can be stated explicitly
Aquatic mammals are mammals and aquatic organisms

Aquatic organisms only include aquatic mammals, fishes and crustaceans

Whales are not fishes

Omnivore animals are animals that eat any other animals or plants

Carnivore animals are animals that only eat other animals, and eat some of them
Fuzzy OWL

• Not for all concepts or not in all contexts/ applications it is possible to assume crisp conditions, under which something is e.g. a legal subject, or an omnivore animal
• If a user needs to represent fuzzy concepts, should use a fuzzy variety of OWL
• But compare necessary conditions, sufficient conditions, scope of the ontology, and real fuzziness
... cont. ...

- Necessary condition on *OmnivoreAnimal*
  - Omnivore animals eat any other animals or plants
- Sufficient condition to *OmnivoreAnimal*
  - Animals that eat any other animals or plants are omnivore animals
- Definition of *OmnivoreAnimal*
  - Omnivore animals are animals that eat any other animals or plants
- Fuzzy condition on *OmnivoreAnimal*
  - Omnivore animals, under certain (but not necessarily all) relevant circumstances, eat any other animal or plant
• Object properties can be symmetric, or transitive.
• “Sibling” is an example of a symmetric object property
• “Part” is an example of a transitive object property
> siblingOrder and siblingSpecies are symmetric and transitive

> Cetacea is sibling order of Artyodactyla within superorder Laurasiatheria

\[ \mathcal{A} \] \hspace{1cm} \textit{Artyodactyla is sibling order of Cetacea}

> Within Cetacea, Monodon monoceros is a sibling species of Orcynus Orca, which is a sibling species of Balaenoptera musculus

\[ \mathcal{A} \] \hspace{1cm} \textit{Monodon monoceros is sibling species of Balaenoptera musculus}
When dealing with XSD datatypes, OWL uses **Datatype Properties**

> Moby Dick is 23 metres long

When dealing with annotations, OWL uses **Annotation Properties**

> Moby Dick has been inspired by a long hunting for the *albino sperm whale Mocha Dick* near the coast of Chile (more than 100 battles with whalers)
XML Datatypes in OWL

• OWL supports XML Schema primitive datatypes
• Clean separation between “object” classes and datatypes
  – Disjoint interpretation domain: \( d^I \subseteq \Delta^I_D \), and \( \Delta^I_D \cap \Delta^I_O = \emptyset \)
  – Disjoint datatype property universe: \( p^I_D \subseteq \Delta^I_O \times \Delta^I_D \)
• Other reasons:
  – Datatypes structured by built-in predicates
  – Not appropriate to form new datatypes using ontology language
• Practical reasons:
  – Ontology language remains simple and compact
  – Semantic integrity of ontology language not compromised
  – Implementability not compromised — can use hybrid reasoner
    • Only need sound and complete decision procedure for \( d^I_1 \cap \ldots \cap d^I_n \),
      where \( d^I_i \) is a (possibly negated) datatype
• Different treatment in OWL 2
OWL exercise XI

- Everything that is true of Cetaceans ontology is true for Moby Dick ontology
OWL Syntaxes

- **Abstract Syntax**
  - Used in the definition of the language and the DL/Lite semantics

- **OWL in RDF (the “official” concrete syntax)**
  - RDF/XML presentation

- **XML Presentation Syntax**
  - XML Schema definition

- **Other, Human readable syntaxes**
  - Manchester OWL Syntax
  - Sydney OWL Syntax
  - Rabbit

*borrowed from Sean Bechhofer’s SSSW08 slides*
Common Misconceptions

• Disjointness of primitives
• Interpreting domain and range
• And and Or
• Quantification
• Closed and Open Worlds

borrowed from Sean Bechhofer’s SSSW08 slides
Disjointness

• By default, primitive classes are not disjoint.
• Unless we explicitly say so, the description (Animal and Vegetable) is not inconsistent.
• Similarly with individuals -- the so-called Unique Name Assumption (often present in DL languages) does not hold, and individuals are not considered to be distinct unless explicitly asserted to be so.
Domain and Range

• OWL allows us to specify the domain and range of properties.
• Note that this is not interpreted as a constraint.
• Rather, the domain and range assertions allow us to make inferences about individuals.
• Consider the following:
  – ObjectProperty: employs
    Domain: Company
    Range: Person
  Individual: IBM
  Facts: employs Jim
• If we haven’t said anything else about IBM or Jim, this is not an error. However, we can now infer that IBM is a Company and Jim is a Person.
And/Or and Quantification

• The logical connectives And and Or often cause confusion
  – Tea or Coffee?
  – Milk and Sugar?

• Quantification can also be contrary to our intuition.
  – Universal quantification over an empty set is true.
    – Sean is a member of hasChild only Martian
  – Existential quantification may imply the existence of an individual that we don’t know the name of.

borrowed from Sean Bechhofer’s SSSW08 slides
Closed and Open Worlds

• The standard semantics of OWL makes an Open World Assumption (OWA).
  – We cannot assume that all information is known about all the individuals in a domain
  – Facilitates reasoning about the intensional definitions of classes
  – Sometimes strange side effects

• Closed World Assumption (CWA)
  – Named individuals are the only individuals in the domain

• Negation as failure
  – If we can’t deduce that x is an A, then we know it must be a (not A)
  – Facilitates reasoning about a particular state of affairs

borrowed from Sean Bechhofer’s SSSW08 slides
SPARQL

• Query language over RDF
• Has sources, namespaces
• Has operators: DESCRIBE, SELECT, CONSTRUCT
• Has lists of conditions, filters, etc.
• PREFIX : <http://example.org/querytest.rdf>
• FROM <http://example.org/querytest.rdf>

• DESCRIBE : Paracetamol
• SELECT ?c
WHERE {
  ?c :hasDrugComponent :Paracetamol }
• CONSTRUCT { ?c :commodityFor ?f }
WHERE {
  ?c :hasDrugComponent ?d .
  ?d :hasFunction ?f }
Rules (SWRL)

- Reified in RDF
- Implications with head and body:
  
  \( (?x \text{ isLocatedIn} \ ?z) \leftarrow ((?x \text{ isLocatedIn} \ ?y) \ (\ ?y \text{ isPartOf} \ ?z)) \)
OWL 2

• Several new features, three are especially important
  – Property chains
    • [brotherOf o motherOf] subPropertyOf uncleOf
    • Mustafa is brother of Saida; Saida is mother of Rashid
    • ⊢ Mustafa is uncle of Rashid
  – Keys
    • Person hasKey: identityNumber
    • Person hasKey: SSN
    • Transplantation hasKey: [donor, recipient, organ]
  – Multiple interpretations of a constant ("punning")
    • Supervisor(Aldo)
    • Role(Supervisor)
    • Supervisor(Aldo, Andrea)