Ontology Engineering

Lecture 1: Foundations of ontology engineering

Dr. C. Maria Keet

email: mkeet@cs.uct.ac.za

home: http://www.meteck.org

Department of Computer Science
University of Cape Town, South Africa

Foundations and recent trends on ontology engineering
Universitat Politècnica de Catalunya, 2017

Outline

Administrivia (1/2)

- This course consists of two lectures of four hours, exercises, and a mini-project assignment in small groups
- Labs&self study about 4-6 hours
- Mini-project [100%], 8 hours (rough timeframe)
  - Topics will be distributed after this lecture
  - Need to from groups of 2-3 people and choose topic by next lecture
  - Deadline hand-in: 14 June
- Following the lectures will be easier and beneficial when you have read the recommended and required reading beforehand, and at least the 10-20 pages/chapter of the lecture notes

Administrivia (2/2)

- Slides, reading material, files for exercises, and answers will be made available
  - The slides serve as a teaching aid, not as a neat summary

Administrivia (2/2)

- Slides, reading material, files for exercises, and answers will be made available
  - The slides serve as a teaching aid, not as a neat summary
- The topics covered in this course are of an introductory nature and only a selection of core and elective topics will be addressed; *this is an active research field....*
- ... so there is no single textbook (yet) that covers all topics for the novice ontologist, has exercises with given, clear answers etc, but...

Administrivia (2/2)

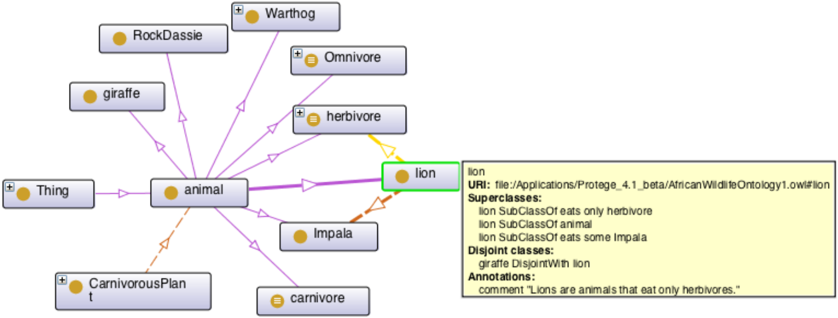
- Slides, reading material, files for exercises, and answers will be made available
  - The slides serve as a teaching aid, not as a neat summary
- The topics covered in this course are of an introductory nature and only a selection of core and elective topics will be addressed; *this is an active research field....*
- ... so there is no single textbook (yet) that covers all topics for the novice ontologist, has exercises with given, clear answers etc, but...
- ... there are lecture notes (though you still have to read some scientific literature)

Outline

An ontology (very informally)

- classes, relationships between them, and constraints that hold between/for them, with possibly individuals and their relations
- as a representation of a particular subject domain

'pretty' picture of a section of the AWO



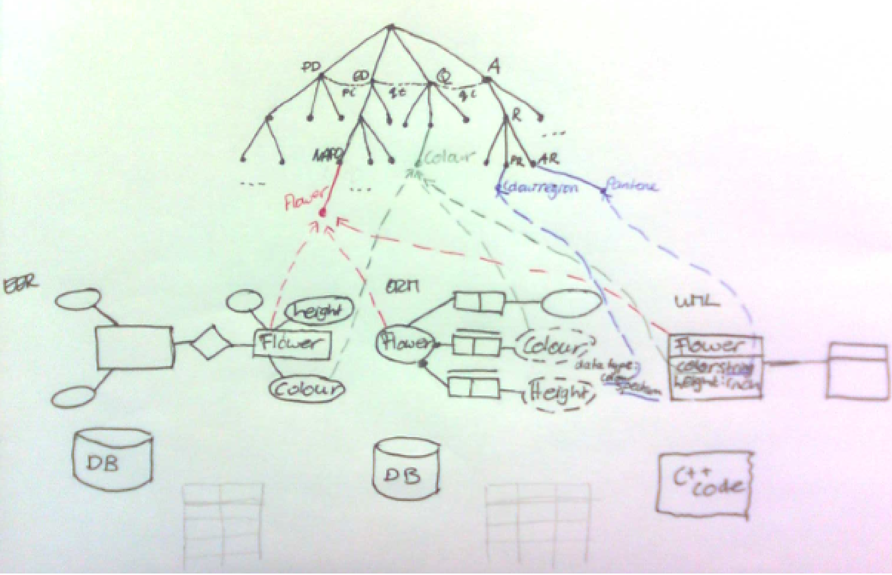
¡ there's a lot going on behind the scenes !

Conceptual data models vs ontologies

- Main differences:
  - Information needs for one application vs. representing the knowledge of a subject domain (regardless the particular application)
  - Formalization in a logic language (though one could do that for conceptual models as well)

Conceptual data models vs ontologies

- Main differences:
  - Information needs for one application vs. representing the knowledge of a subject domain (regardless the particular application)
  - Formalization in a logic language (though one could do that for conceptual models as well)
- An ontology as a layer on top of conceptual data models
  - To improve the quality of a conceptual data model (hence, the software)
  - To facilitate database integration, or prevent the usual data integration problems


PD
ED
Q
A
R
NAPO
Colour
PR
AR
Flower
EER
ORM
UML
Height
Flower
Colour
Flower
Colour
Height
Flower
DB
DB
C++ code

Databases vs. Knowledge bases

- Main differences:
  - Representation of the knowledge
  - Rules
  - Reasoning to infer new or implicit knowledge, detect inconsistencies of the knowledge base
  - Open World Assumption (vs. Closed World Assumption in databases)

What is the usefulness of an ontology?

- Making, more or less precisely, the (dis-)agreement among people explicit
- Enrich software applications with the additional semantics ⇒ *ontology-driven information systems*
- Thus, practically, improving computer-computer, computer-human, and human-human communication

Outline

Examples ontologies in information systems

- **e-learning** with *Inquire Biology* [Chaudhri et al., 2013]: textbook annotated with terms of the ontology, generates questions and answers.
- **data integration, cultural heritage**: combining resources of data and querying them, on the ever interesting topic of food [Calvanese et al., 2016]
- **publishing** of scientific papers, books: enable navigation and understanding of scholarly documents [Di Iorio et al., 2014]
- **semantic meta-mining of data mining experiments** (sections 1 and 5 of [Keet et al., 2015]): mine the (ontology-based) annotations of the data mining experiments, reason over that to have it propose the optimal data mining experiment

More Examples

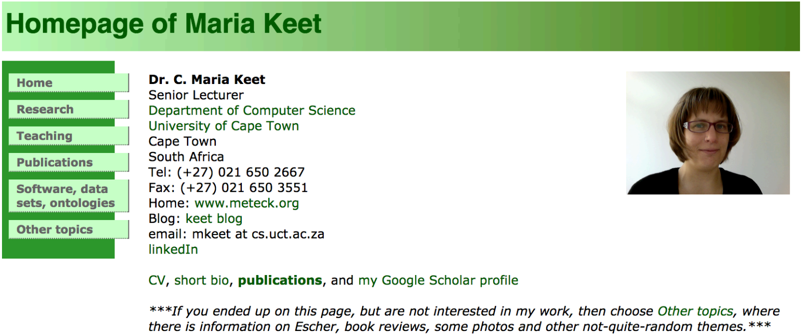
- **For science** Inside the scientific method: Outperforming humans (ontology+reasoner): classification of protein phosphatases [Wolstencroft et al., 2007]
- **Deep Question-Answering** with Watson beating human top-performers in 'Jeopardy!'; uses over 100 techniques, including ontologies for integration
- **Ontology-driven conceptual data modelling**: being more precise than just drawing diagrams, e.g., on those 'shared' and 'composite' aggregations in UML Class diagrams [Keet & Artale, 2008], finding contradictions.

The Semantic Web – Introduction (some motivations for ontologies and knowledge bases)

- AI put to the test in the (uncontrollable?) very large field
- Adding meaning to plain HTML pages and Web 2.0 by using theory and technologies of KBs and ontologies

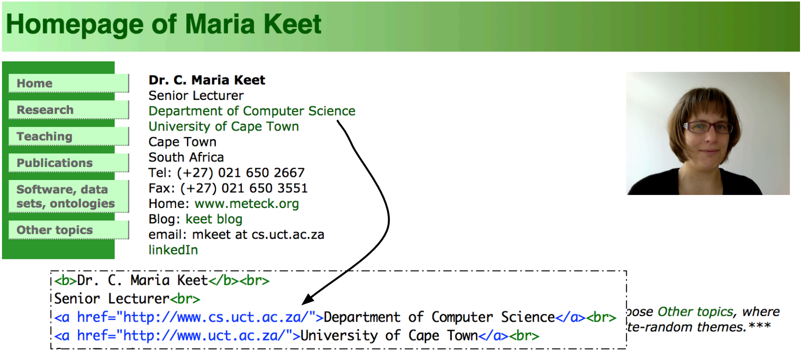
The Semantic Web – Introduction (some motivations for ontologies and knowledge bases)

- AI put to the test in the (uncontrollable?) very large field
- Adding meaning to plain HTML pages and Web 2.0 by using theory and technologies of KBs and ontologies



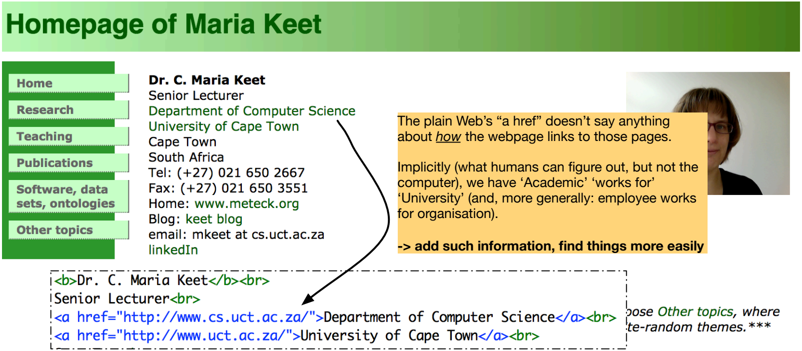
The Semantic Web – Introduction (some motivations for ontologies and knowledge bases)

- AI put to the test in the (uncontrollable?) very large field
- Adding meaning to plain HTML pages and Web 2.0 by using theory and technologies of KBs and ontologies



The Semantic Web – Introduction (some motivations for ontologies and knowledge bases)

- AI put to the test in the (uncontrollable?) very large field
- Adding meaning to plain HTML pages and Web 2.0 by using theory and technologies of KBs and ontologies



Generalising from the examples, ontologies are used for:

- Data(base) linking and integration
- Instance classification
- Matchmaking and services
- Querying, information retrieval
  - Ontology-Based Data Access
  - Ontologies to improve NLP
- Bringing more quality criteria into conceptual data modelling to develop a better model (hence, a better quality software system)

Outline

Background

– Aristotle and colleagues: **O**ntology
– Engineering: ontolog**ies** (count noun)

– Investigating reality, representing it
– Putting an engineering artefact to use

What then, is this engineering artefact?

(Guarino, 2002)

First, let's look at an artefact: a text file....

AfricanWildlifeOntology1.owl

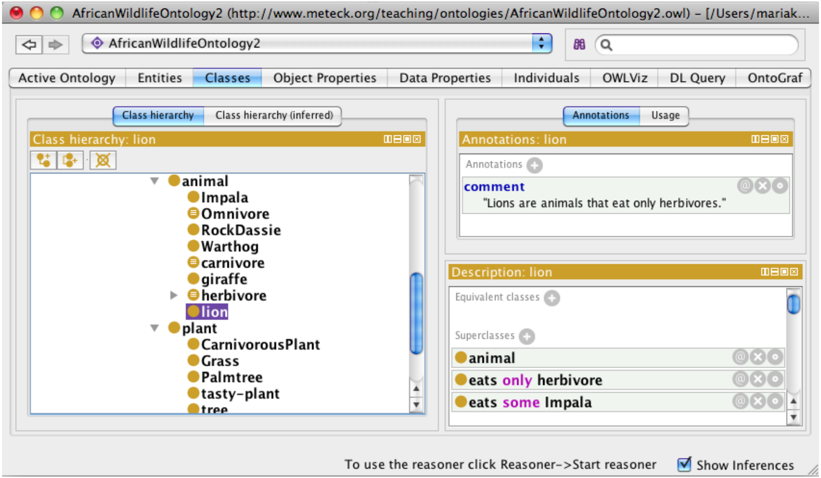
```
<owl:Class rdf:about="&AfricanWildlifeOntology1;lion">
    <rdfs:subClassOf rdf:resource="&AfricanWildlifeOntology1;animal"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&AfricanWildlifeOntology1;eats"/>
            <owl:allValuesFrom rdf:resource="&AfricanWildlifeOntology1;herbivore"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&AfricanWildlifeOntology1;eats"/>
            <owl:someValuesFrom rdf:resource="#Impala"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment>Lions are animals that eat only herbivores.</rdfs:comment>
</owl:Class>



<!-- file:/Applications/Protege_4.1_beta/AfricanWildlifeOntology1.owl#plant -->

<owl:Class rdf:about="&AfricanWildlifeOntology1;plant">
    <rdfs:comment>Plants are disjoint from animals.</rdfs:comment>
</owl:Class>
```

… and rendered in an ontology editor



A few definitions on what the text in the file is supposed to stand for

- Most cited (but very inadequate definition): "An ontology is a specification of a conceptualization" (by Tom Gruber, 1993)
- "a formal specification of a shared conceptualization" (by Borst, 1997)
- "An ontology is a formal, explicit specification of a shared conceptualization" (Studer et al., 1998)

A few definitions on what the text in the file is supposed to stand for

- Most cited (but very inadequate definition): "An ontology is a specification of a conceptualization" (by Tom Gruber, 1993)
- "a formal specification of a shared conceptualization" (by Borst, 1997)
- "An ontology is a formal, explicit specification of a shared conceptualization" (Studer et al., 1998)
- What is a *conceptualization*, and a *formal, explicit specification*? Why *shared*?

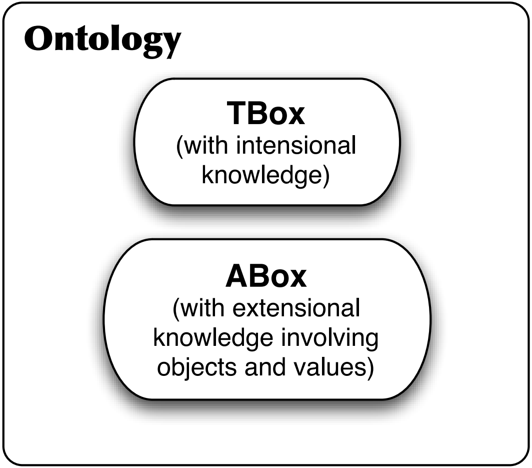
More definitions

- More detailed: "An ontology is a logical theory accounting for the *intended meaning* of a formal vocabulary, i.e. its *ontological commitment* to a particular *conceptualization* of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models." (Guarino, 1998)

More definitions

- More detailed: "An ontology is a logical theory accounting for the *intended meaning* of a formal vocabulary, i.e. its *ontological commitment* to a particular *conceptualization* of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models." (Guarino, 1998)
- And back to a simpler definition: "with an ontology being equivalent to a Description Logic knowledge base" (Horrocks et al, 2003)

Description Logic knowledge base



From logical to ontological level (1/2)

- Logical level (no structure, no constrained meaning[1]):
  - $\exists x (Apple(x) \wedge Green(x))$
  - "there exists an object that is an apple and it is green"

adapted from (Guarino, 2008)

[1] meaning in the sense of subject domain semantics, not formal semantics

[2] DL has a model-theoretic semantics, so the axioms have a meaning in that sense of 'meaning/semantics'

From logical to ontological level (1/2)

- Logical level (no structure, no constrained meaning[1]):
  - $\exists x(Apple(x) \wedge Green(x))$
  - "there exists an object that is an apple and it is green"
- Epistemological level (structure, no constrained meaning):
  - $\exists x : apple\ Green(x)$ (many-sorted logics)
  - "there exists an apple-object that is green"

adapted from (Guarino, 2008)

[1] meaning in the sense of subject domain semantics, not formal semantics

[2] DL has a model-theoretic semantics, so the axioms have a meaning in that sense of 'meaning/semantics'

From logical to ontological level (1/2)

- Logical level (no structure, no constrained meaning[1]):
  - $\exists x(Apple(x) \wedge Green(x))$
  - "there exists an object that is an apple and it is green"
- Epistemological level (structure, no constrained meaning):
  - $\exists x : apple\ Green(x)$ (many-sorted logics)
  - "there exists an apple-object that is green"
  - ~~$\exists x : green\ Apple(x)$~~
  - "there exists a green-object that is an apple"

adapted from (Guarino, 2008)

[1] meaning in the sense of subject domain semantics, not formal semantics

[2] DL has a model-theoretic semantics, so the axioms have a meaning in that sense of 'meaning/semantics'

From logical to ontological level (1/2)

- Logical level (no structure, no constrained meaning[1]):
    - $\exists x(Apple(x) \wedge Green(x))$
    - "there exists an object that is an apple and it is green"
- Epistemological level (structure, no constrained meaning):
    - $\exists x : apple\ Green(x)$ (many-sorted logics)
    - "there exists an apple-object that is green"
    - ~~$\exists x : green\ Apple(x)$~~
    - "there exists a green-object that is an apple"
    - $Apple(a)$ and $hasColor(a, green)$ (description logics[2])
    - "object a is an apple and that object a has the colour green"

adapted from (Guarino, 2008)

[1] meaning in the sense of subject domain semantics, not formal semantics

[2] DL has a model-theoretic semantics, so the axioms have a meaning in that sense of 'meaning/semantics'

From logical to ontological level (1/2)

- Logical level (no structure, no constrained meaning[1]):
  - $\exists x(Apple(x) \land Green(x))$
  - "there exists an object that is an apple and it is green"
- Epistemological level (structure, no constrained meaning):
  - $\exists x : apple\ Green(x)$ (many-sorted logics)
  - "there exists an apple-object that is green"
  - ~~$\exists x : green\ Apple(x)$~~
  - "there exists a green-object that is an apple"
  - $Apple(a)$ and $hasColor(a, green)$ (description logics[2])
  - "object a is an apple and that object a has the colour green"
  - ~~$Green(a)$ and $hasShape(a, apple)$~~
  - "object a is a green and that object a has the shape of an apple"

adapted from (Guarino, 2008)

[1] meaning in the sense of subject domain semantics, not formal semantics

[2] DL has a model-theoretic semantics, so the axioms have a meaning in that sense of 'meaning/semantics'

From logical to ontological level (2/2)

- Ontological level (structure, constrained meaning):
  - Some structuring choices are excluded because of ontological constraints
  - e.g., 'apple objects' seems bester than 'green objects'
  - e.g., objects having the colour green seems more reasonable than having an 'apple-shape'

adapted from (Guarino, 2008)

From logical to ontological level (2/2)

- Ontological level (structure, constrained meaning):
  - Some structuring choices are excluded because of ontological constraints
  - e.g., 'apple objects' seems bester than 'green objects'
  - e.g., objects having the colour green seems more reasonable than having an 'apple-shape'
  - There are reasons for that:
    - *Apple* carries an identity condition, so one can identify the object somehow (it is a 'sortal'),
    - *Green* does not (is a value ['qualia'] of the attribute ['quality'] *hasColor* that a thing has)

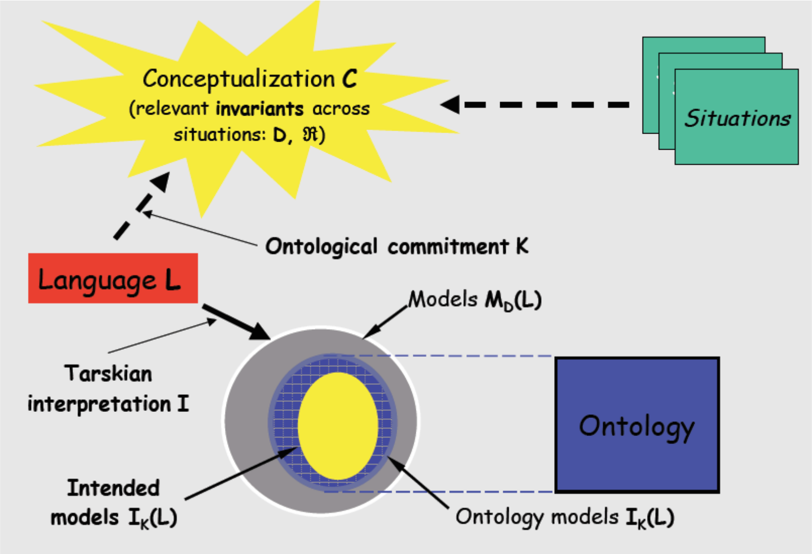
adapted from (Guarino, 2008)

From logical to ontological level (2/2)

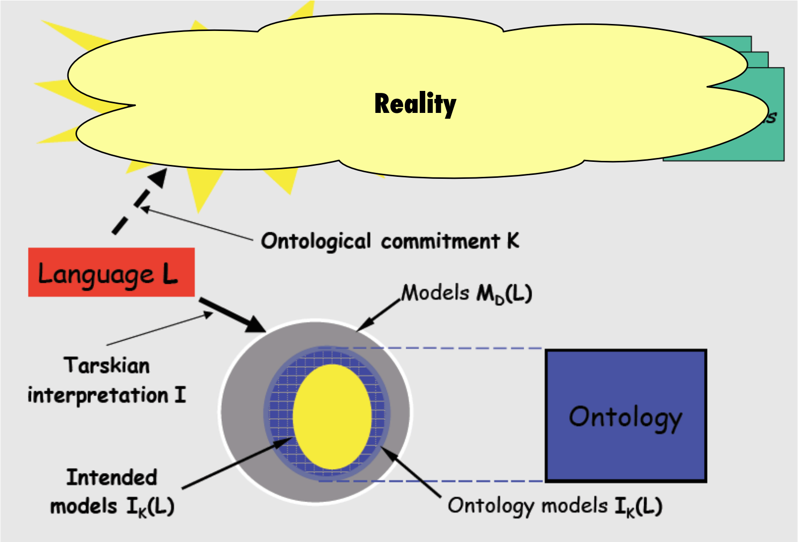
- Ontological level (structure, constrained meaning):
  - Some structuring choices are excluded because of ontological constraints
  - e.g., 'apple objects' seems bester than 'green objects'
  - e.g., objects having the colour green seems more reasonable than having an 'apple-shape'
  - There are reasons for that:
    - *Apple* carries an identity condition, so one can identify the object somehow (it is a 'sortal'),
    - *Green* does not (is a value ['qualia'] of the attribute ['quality'] *hasColor* that a thing has)
- Put differently: one way of representing things turn out to be *better* than others.

adapted from (Guarino, 2008)

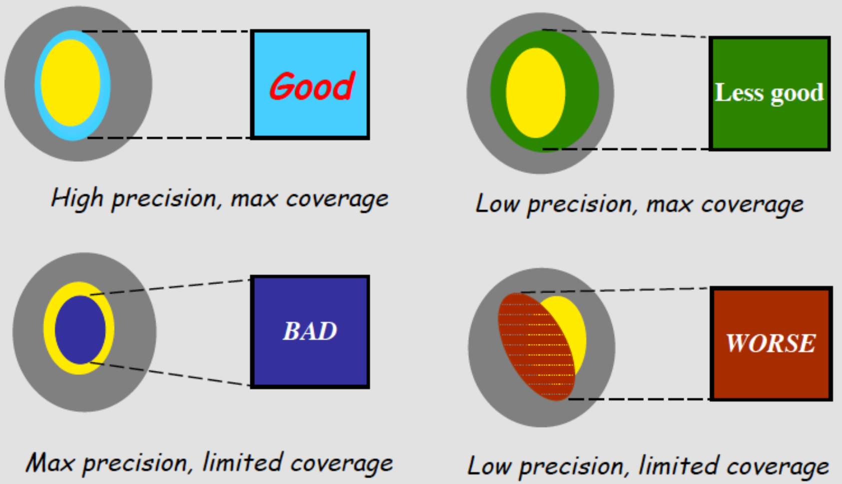
Ontologies and meaning



Ontologies and reality



Quality of the ontology



(Guarino, 2002)

Outline

Preliminary note

- There are only a few core concepts to get the general idea
- There are very many details
- Here we focus on the core concepts and some details and how that works out in computing
- More logic and details in the lecture notes

How to formalise it?

- Logics have a:
  - Syntax
    - Alphabet
    - Languages constructs
    - Sentences to assert knowledge
  - Semantics
    - Formal meaning

Several ontology languages

- W3C-standardised Web Ontology Language OWL, comes in many ‘species’
  - Description Logics-based OWL species
  - OWL full and OWL 2 full (RDF-based semantics)
- Common logic, CLIF
- First order logic
- Fuzzy and temporal extensions and variants

DLs are structured fragments of FOL

- Recall that full FOL is undecidable
- This is unpleasant for automated reasoning

DLs are structured fragments of FOL

- Recall that full FOL is undecidable
- This is unpleasant for automated reasoning
- Approach: find a fragment—a *sublanguage*—of FOL that is decidable
- Take some features, prove the computational complexity of some problem

DLs are structured fragments of FOL

- We end up with *trade-offs* of features in a DL
- Some features always will make the language undecidable (e.g., true role composition, $R \circ S \equiv T$)
- Other features are only ‘problematic’ (computationally less desirable) when taken together with another

DLs are structured fragments of FOL

- We end up with *trade-offs* of features in a DL
- Some features always will make the language undecidable (e.g., true role composition, $R \circ S \equiv T$)
- Other features are only 'problematic' (computationally less desirable) when taken together with another
- E.g., one could define a language where:
  - it is prohibited to use $\neg$ (negation) in an axiom, or
  - only $> 2\, R.\top$ (no range specified) but not $> 2\, R.D$, or
  - $\exists R$ only on the rhs of the inclusion but not on the lhs

DLs are structured fragments of FOL

- We end up with *trade-offs* of features in a DL
- Some features always will make the language undecidable (e.g., true role composition, $R \circ S \equiv T$)
- Other features are only 'problematic' (computationally less desirable) when taken together with another
- E.g., one could define a language where:
  - it is prohibited to use $\neg$ (negation) in an axiom, or
  - only $\geq 2\, R.\top$ (no range specified) but not $\geq 2\, R.D$, or
  - $\exists R$ only on the rhs of the inclusion but not on the lhs
- There are *many* DLs, and most combinations have been investigated over the past 25 years
- Roughly: the fewer features and the more restrictions, the more 'computationally well-behaved' the language is

On those OWL ‘species’

- OWL standardised in 2004
- OWL 2 standardised in 2009

On those OWL 'species'

- OWL standardised in 2004
  - OWL Lite
  - OWL DL
  - OWL full
- OWL 2 standardised in 2009

On those OWL 'species'

- OWL standardised in 2004
  - OWL Lite
  - OWL DL
  - OWL full
- OWL 2 standardised in 2009
  - OWL 2 profiles
    - OWL 2 EL
    - OWL 2 QL
    - OWL 2 RL
  - OWL 2 DL
  - OWL 2 full

OWL 2 DL Overview

- Has more features than OWL DL
- Computationally more ‘costly’ (N2EXPTIME-complete cf EXPTIME-complete)
- Based on the DL language $\mathcal{SROIQ}$
- Main novelty especially w.r.t. modelling practices: qualified cardinality constraints, more ‘characteristics’ of object properties

OWL 2 EL Overview

- Intended for large ‘simple’ ontologies
- Focussed on type-level knowledge (TBox)
- Better computational behaviour than OWL 2 DL (polynomial vs. exponential/open)
- Based on the DL language $\mathcal{EL}^{++}$ (PTime complete)
- Reasoner: e.g. CEL `http://code.google.com/p/cel/`
- $\sqcap \exists \top \bot \quad \sqsubseteq \quad \sqcap \exists \top \bot$

OWL 2 QL Overview

- Query answering over a large amount of instances with same kind of performance as relational databases (Ontology-Based Data Access)
- Expressive features cover several used features of UML Class diagrams and ER models ('COnceptual MOdel-based Data Access')
- Based on *DL-Lite*$_{\mathcal{R}}$ (*more is possible with UNA and in some implementations*)

OWL 2 RL Overview

- Development motivated by: what fraction of OWL 2 DL can be expressed by rules (with equality)?
- Scalable reasoning in the context of RDF(S) application
- Rule-based technologies (forward chaining rule system, over *instances*)
- Inspired by Description Logic Programs and pD*
- Reasoning in PTime
- No $\forall$ and $\neg$ on lhs, and $\exists$ and $\sqcup$ on rhs of $\sqsubseteq$

OWL Syntax—Many notations, actually

- Making those DL symbols usable by a computer
  - e.g., not "$\sqsubseteq$" to process inclusion, but "`SubClassOf`"

OWL Syntax—Many notations, actually

- Making those DL symbols usable by a computer
  - e.g., not "$\sqsubseteq$" to process inclusion, but "`SubClassOf`"
- RDF/XML
  - Official exchange syntax
  - Hard for humans to read (and RDF parsers are hard to write)
- OWL/XML
  - Not the RDF syntax
  - Still hard for humans, but more XML than RDF tools available
- Abstract syntax
  - To some, considered human readable
- "User-usable" ones
  - e.g., Manchester syntax, informal and limited matching with UML, pseudo-NL verbalisations

Example correspondences

- ‘Each C is a D’ / ‘All Cs are Ds’
  - `SubClassOf(C D)`
  - $C \sqsubseteq D$
  - $\forall x(C(x) \rightarrow D(x))$

e.g.: $Giraffe \sqsubseteq Animal$

Example correspondences

- ‘Each C is a D’ / ‘All Cs are Ds’
  - `SubClassOf(C D)`
  - $C \sqsubseteq D$
  - $\forall x(C(x) \rightarrow D(x))$

  e.g.: $Giraffe \sqsubseteq Animal$
- ‘Each C R at least one D’ / ‘Each C R some D’
  - `SubClassOf(C ObjectSomeValueFrom(R D))`
  - $C \sqsubseteq \exists R.D$
  - $\forall x(C(x) \rightarrow \exists y(R(x, y) \wedge D(y))$

  e.g.: $Elephant \sqsubseteq \exists eats.AmarulaFruit$

Example correspondences

- ‘Each C is a D’ / ‘All Cs are Ds’
  - `SubClassOf(C D)`
  - $C \sqsubseteq D$
  - $\forall x(C(x) \rightarrow D(x))$

  e.g.: $Giraffe \sqsubseteq Animal$
- ‘Each C R at least one D’ / ‘Each C R some D’
  - `SubClassOf(C ObjectSomeValueFrom(R D))`
  - $C \sqsubseteq \exists R.D$
  - $\forall x(C(x) \rightarrow \exists y(R(x,y) \wedge D(y))$

  e.g.: $Elephant \sqsubseteq \exists eats.AmarulaFruit$
- ‘C and D are disjoint’ / ‘each C is not a D’
  - `DisjointClasses(C D)` / `SubClassOf(C ObjectComplementOf(D))`
  - $C \sqcap D \sqsubseteq \bot$ (disj.) / $C \sqsubseteq \neg D$ (complement)
  - $\forall x(C(x) \rightarrow \neg D(x))$

  e.g.: $Herbivore \sqsubseteq \neg Carnivore$

Semantics (DL-based OWL species)

- Model-theoretic semantics

Semantics (DL-based OWL species)

- Model-theoretic semantics
- Domain Δ is a non-empty set of objects
- Interpretation: $\cdot^{\mathcal{I}}$ is the *interpretation function*, domain $\Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every concept name A to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every role name R to a subset $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every individual name a to elements of $\Delta^{\mathcal{I}}$: $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- Note: $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$

Semantics (DL-based OWL species)

- Model-theoretic semantics
- Domain Δ is a non-empty set of objects
- Interpretation: $\cdot^{\mathcal{I}}$ is the *interpretation function*, domain $\Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every concept name A to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every role name R to a subset $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every individual name a to elements of $\Delta^{\mathcal{I}}$: $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- Note: $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$
- An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model of a knowledge base $\mathcal{KB}$ if every axiom of $\mathcal{KB}$ is satisfied by $\mathcal{I}$
- A knowledge base $\mathcal{KB}$ is said to be satisfiable if it admits a model

Essential to automated reasoning (Ch2 of LN)

- The choice of the class of problems the software program has to solve
- The formal language in which to represent the problems
- The way how the program has to compute the solution
- How to do this efficiently

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of C (resp. R) contained in the extension of D (resp. S) in every model of $\mathcal{T}$?

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of C (resp. R) contained in the extension of D (resp. S) in every model of $\mathcal{T}$?
- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is a (resp. (a, b)) a member of concept C (resp. R) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?

Reasoning services for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of C (resp. R) contained in the extension of D (resp. S) in every model of $\mathcal{T}$?
- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is a (resp. (a, b)) a member of concept C (resp. R) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?
- Instance retrieval ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - find all members of C in $\mathcal{KB}$, i.e., compute all individuals a s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$

Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of ϕ

Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of ϕ
- Example:
  TBox: $\exists\mathtt{TEACHES}.\mathtt{Course} \sqsubseteq \neg\mathtt{Undergrad} \sqcup \mathtt{Professor}$
  ABox: $\mathtt{TEACHES}(\mathtt{John}, \mathtt{cs101})$, $\mathtt{Course}(\mathtt{cs101})$, $\mathtt{Undergrad}(\mathtt{John})$

Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of ϕ
- Example:
  TBox: $\exists \mathtt{TEACHES.Course} \sqsubseteq \neg \mathtt{Undergrad} \sqcup \mathtt{Professor}$
  ABox: $\mathtt{TEACHES}(\mathtt{John}, \mathtt{cs101})$, $\mathtt{Course}(\mathtt{cs101})$, $\mathtt{Undergrad}(\mathtt{John})$
- $\mathcal{KB} \models \mathtt{Professor}(\mathtt{John})$

Logical implication

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of ϕ
- Example:
  TBox: $\exists \mathtt{TEACHES.Course} \sqsubseteq \neg \mathtt{Undergrad} \sqcup \mathtt{Professor}$
  ABox: $\mathtt{TEACHES(John, cs101)}$, $\mathtt{Course(cs101)}$, $\mathtt{Undergrad(John)}$
- $\mathcal{KB} \models \mathtt{Professor(John)}$
- What if:
  TBox: $\exists \mathtt{TEACHES.Course} \sqsubseteq \mathtt{Undergrad} \sqcup \mathtt{Professor}$
  ABox: $\mathtt{TEACHES(John, cs101)}$, $\mathtt{Course(cs101)}$, $\mathtt{Undergrad(John)}$
- $\mathcal{KB} \models \mathtt{Professor(John)}$? or perhaps $\mathcal{KB} \models \neg\mathtt{Professor(John)}$?

Automated reasoning techniques

- How do we compute, say, satisfiability?

Automated reasoning techniques

- How do we compute, say, satisfiability?
- Truth tables are too cumbersome
- Several techniques are more efficient
- Current ‘winner’ is *tableau reasoning*

The idea, same as for FOL

- A sound and complete procedure deciding satisfiability is all we need, and the **tableaux method is a decision procedure which checks the existence of a model**
- It exhaustively looks at all the possibilities, so that it can eventually prove that no model could be found for unsatisfiable formulas.
- $\phi \models \psi$ iff $\phi \land \neg\psi$ is NOT satisfiable—if it is satisfiable, we have found a counterexample
- Decompose the formula in top-down fashion

The idea, same as for FOL

- A sound and complete procedure deciding satisfiability is all we need, and the **tableaux method is a decision procedure which checks the existence of a model**
- It exhaustively looks at all the possibilities, so that it can eventually prove that no model could be found for unsatisfiable formulas.
- $\phi \models \psi$ iff $\phi \wedge \neg\psi$ is NOT satisfiable—if it is satisfiable, we have found a counterexample
- Decompose the formula in top-down fashion
- Following slide *simplified* process (thanks to Markus Krötzsch & Sebastian Rudolph ESSLLI 2009 Bordeaux)

Automated reasoning

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 (
[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 .

Tableau

Automated reasoning

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 (
[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 .

Tableau

ex:HappyCatOwner

Automated reasoning

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 (
[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 .

Tableau

ex:HappyCatOwner
[owl:intersectionOf (,)]

Automated reasoning

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 (
[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 .

Tableau

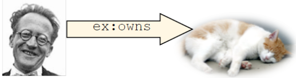
ex:HappyCatOwner
[owl:intersectionOf (,)]

Automated reasoning

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 (
    [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 .

Tableau



ex:HappyCatOwner
[owl:intersectionOf (,)]

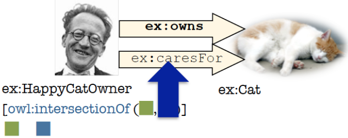
ex:Cat

Automated reasoning

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 (
[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 .

Tableau

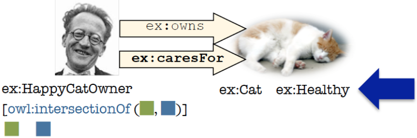


Automated reasoning

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 (
    [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 .

Tableau

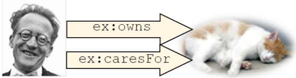


Automated reasoning

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 (
    [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 .

Tableau



ex:HappyCatOwner
[owl:intersectionOf (,)]

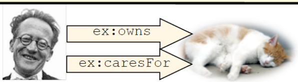
ex:Cat ex:Healthy
[owl:unionOf
(ex:Dead, ex:Alive)]

Automated reasoning

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 (
    [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 .

Tableau



ex:HappyCatOwner
[owl:intersectionOf (,)]

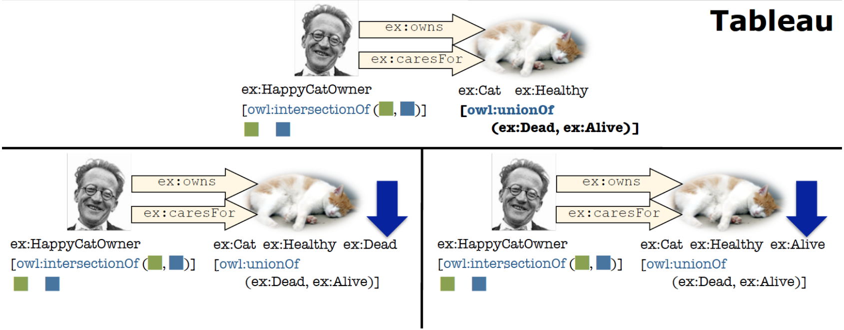
ex:Cat ex:Healthy
[owl:unionOf (ex:Dead, ex:Alive)]

Automated reasoning

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 (
  [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 .

Tableau

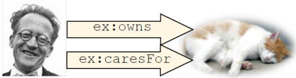


Automated reasoning

Knowledge Base

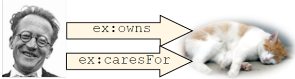
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
[rdf:type owl:Restriction ; owl:onProperty ex:owns ; owl:someValuesFrom ex:Cat],
[rdf:type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

Tableau



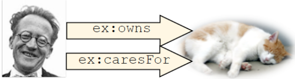
ex:HappyCatOwner
[owl:intersectionOf (,)]

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



ex:HappyCatOwner
[owl:intersectionOf (,)]

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



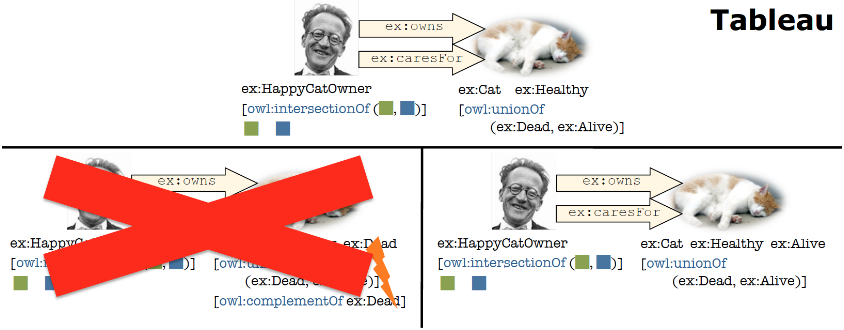
ex:HappyCatOwner
[owl:intersectionOf (,)]

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

Automated reasoning

Knowledge Base

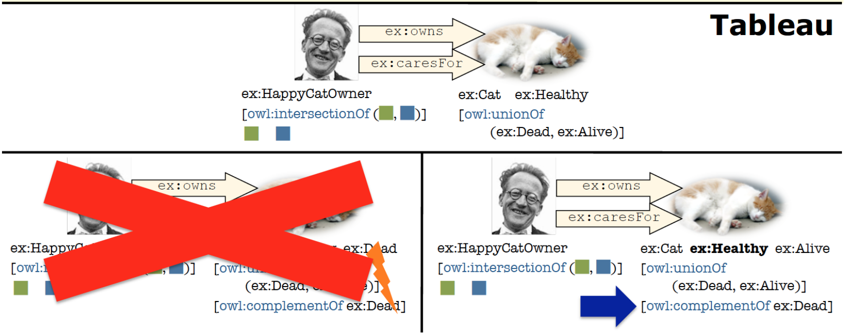
ex:Healthy rdfs:subClassOf [owl:complementOf ex:Dead] .
ex:Cat rdfs:subClassOf [owl:unionOf (ex:Dead, ex:Alive)] .
ex:owns rdfs:subPropertyOf ex:caresFor .
ex:HappyCatOwner rdfs:subClassOf [owl:intersectionOf (
  [rdf:type owl:Restriction ; owl:onProperty ex:owns ; owl:someValuesFrom ex:Cat],
  [rdf:type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



Automated reasoning

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 (
[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 .

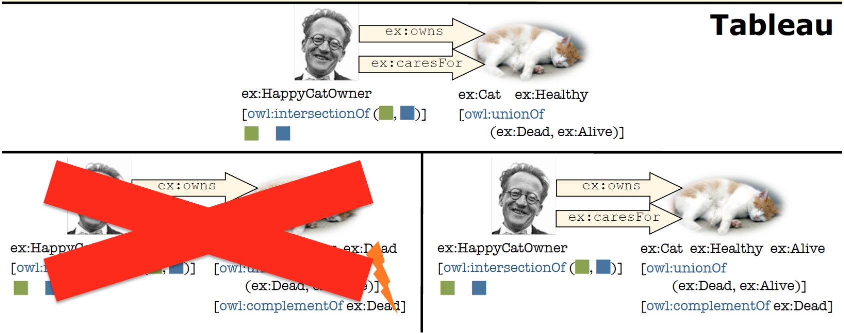


Automated reasoning

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 (
[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 .

Tableau



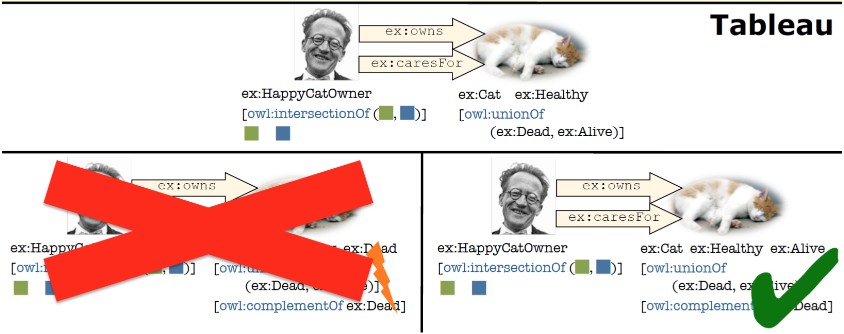
Automated reasoning

Knowledge Base

Satisfiable

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 .

Tableau



Summary

1. Introduction
2. Where is it used?
3. What is an Ontology?
4. Logic and automated reasoning
   - Representation languages
   - Automated reasoning

Exercises

- Chapter 1: exercises 1 and 2
- Chapter 3: exercises 5 (optionally 6)
- Chapter 4: exercises 8, 10, 14-21
- Read the mini-project topics (optionally also looking up what some of those terms mean) and select one

Exercises

- Chapter 1: exercises 1 and 2
- Chapter 3: exercises 5 (optionally 6)
- Chapter 4: exercises 8, 10, 14-21
- Read the mini-project topics (optionally also looking up what some of those terms mean) and select one
- The tutorial ontologies are available from `http://www.meteck.org/teaching/ontologies/`

Additional references

Calvanese, D., Liuzzo, P., Mosca, A., Remesal, J, Rezk, M., Rull, G. Ontology-Based Data Integration in EPNet: Production and Distribution of Food During the Roman Empire. Engineering Applications of Artificial Intelligence, 2016. To appear.

Chaudhri, V.K., Cheng, B., Overholtzer, A, Roschelle, J., Spaulding, A., Clark, P., Greaves, M., Gunning, D.. Inquire Biology: A Textbook that Answers Questions. AI Magazine, 2013, 34(3): 55-72.

Di Iorio, A., Peroni, S., Vitali, F., Zingoni, J. (2014). Semantic lenses to bring digital and semantic publishing together. In Zhao, J., van Erp, M., Kessler, C., Kauppinen, T., van Ossenbruggen, J., van Hage, W. R. (Eds.), Proceedings of the 4th Workshop on Linked Science (LISC 2014), CEUR Workshop Proceedings 1282: 12?23. Aachen, Germany: CEUR-WS.org.

Keet, C.M., Ławrynowicz, A., d'Amato, C., Kalousis, A., Nguyen, P., Palma, R., Stevens, R., Hilario, M. The Data Mining OPtimization ontology. Web Semantics: Science, Services and Agents on the World Wide Web, 2015, 32:43-53.

Keet, C.M., Artale, A. Representing and Reasoning over a Taxonomy of Part-Whole Relations. Applied Ontology, 2008, 3(1-2):91-110.

Wolstencroft, K., Stevens, R., and Haarslev, V.. Applying OWL reasoning to genomic data. In Baker, C. and Cheung, H., (Eds.). Semantic Web: revolutionizing knowledge discovery in the life sciences. pp225-248. Springer: New York. 2007.

Supported class restrictions OWL 2 EL

- existential quantification to a class expression or a data range
- existential quantification to an individual or a literal
- self-restriction
- enumerations involving a single individual or a single literal
- intersection of classes and data ranges

Supported axioms, restricted to allowed set of class expressions OWL 2 EL

- class inclusion, equivalence, disjointness
- object property inclusion and data property inclusion
- property equivalence
- transitive object properties
- reflexive object properties
- domain and range restrictions
- assertions
- functional data properties
- keys
- In short: $\sqcap\ \exists\ \top\ \bot\ \sqsubseteq\ \sqcap\ \exists\ \top\ \bot$

NOT supported in OWL 2 EL

- universal quantification to a class expression or a data range
- cardinality restrictions
- disjunction
- class negation
- enumerations involving more than one individual
- disjoint properties
- irreflexive, symmetric, and asymmetric object properties
- inverse object properties, functional and inverse-functional object properties

Supported Axioms in OWL 2 QL, restrictions

- Subclass expressions restrictions:
  - a class
  - existential quantification (ObjectSomeValuesFrom) where the class is limited to owl:Thing
  - existential quantification to a data range (DataSomeValuesFrom)
- Super expressions restrictions:
  - a class
  - intersection (ObjectIntersectionOf)
  - negation (ObjectComplementOf)
  - existential quantification to a class (ObjectSomeValuesFrom)
  - existential quantification to a data range (DataSomeValuesFrom)

Supported Axioms in OWL 2QL

- Restrictions on class expressions, object and data properties occurring in functionality assertions cannot be specialized
- subclass axioms
- class expression equivalence (involving subClassExpression), disjointness
- inverse object properties
- property inclusion (not involving property chains and SubDataPropertyOf)
- property equivalence
- property domain and range
- disjoint properties
- symmetric, reflexive, irreflexive, asymmetric properties
- assertions other than individual equality assertions and negative property assertions (DifferentIndividuals, ClassAssertion, ObjectPropertyAssertion, and DataPropertyAssertion)

NOT supported in OWL 2 QL

- existential quantification to a class expression or a data range in the subclass position
- self-restriction
- existential quantification to an individual or a literal
- enumeration of individuals and literals
- universal quantification to a class expression or a data range
- cardinality restrictions
- disjunction
- property inclusions involving property chains
- functional and inverse-functional properties
- transitive properties
- keys
- individual equality assertions and negative property assertions

Supported in OWL 2 RL

- More restrictions on class expressions (see table 2, e.g. no SomeValuesFrom on the right-hand side of a subclass axiom)
- All axioms in OWL 2 RL are constrained in a way that is compliant with the restrictions in Table 2.
- Thus, OWL 2 RL supports all axioms of OWL 2 apart from disjoint unions of classes and reflexive object property axioms.
- No $\forall$ and $\neg$ on lhs, and $\exists$ and $\sqcup$ on rhs of $\sqsubseteq$