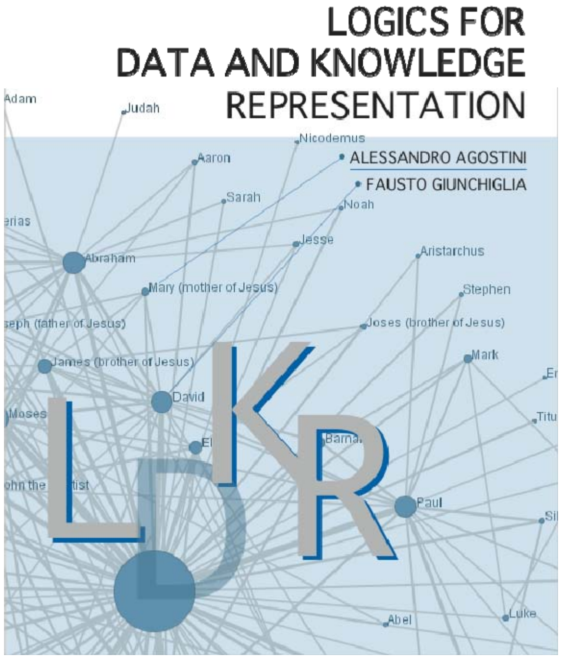


Logics for Data and Knowledge Representation

Web Ontology Language (OWL)

Fausto Giunchiglia and Biswanath Dutta

Outline

- Introduction
- OWL
- Syntax
  - Exchange Syntax
  - Abstract Syntax
    - Constructors
    - Axioms and facts
- Demo
- Semantics
- Reasoning
- Tool Support for OWL

Limitations of RDFS

- Is too weak in describing resources with sufficient details
  - No localised range and domain constraints
    - Cannot say that the range of teachBy is only professor when applied to professors and lecturer when applied to lecturers
  - No cardinality constraints
    - Cannot say that a course is taught by at least one professor, or persons have exactly 2 parents
  - No transitive, inverse or symmetrical properties
    - Cannot say that isPartOf is a transitive property, that hasSupervisor is the inverse of isSupervisorOf, and, that friendOf is symmetrical
  - Disjoint classes
    - Cannot say that Graduate and PhD. Students are two disjoint classes
  - Boolean combinations of classes
    - Sometimes we may need to build new classes by combining other classes using union, intersection, and complement (e.g. person is the disjoint union of the classes male and female)

Ontology Languages

- Wide variety of onotlogy languages for explicit specification
  - Graphical notations
    - Semantic networks, Topic Maps, UML, RDF
  - Logic based
    - Description Logics (e.g., OIL, DAML+OIL, OWL), Rules (e.g., RuleML, SWRL, N3Logic, LP/Prolog), First Order Logic (e.g., KIF), Conceptual graphs, (Syntactically) higher order logics (e.g., LBase), Non-classical logics (e.g., Flogic, Non-Mon, modalities)
  - Probabilistic/fuzzy
- However, degree of formality varies widely
  - Increased formality makes languages more amenable to machine processing (e.g., automated reasoning)

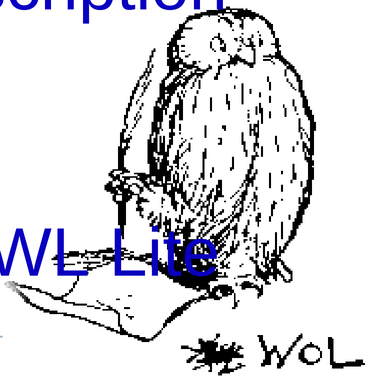
Important: XMLS is not an ontology language

Ontology Language Requirements

- Well defined syntax
- Extends existing Web standards
  - Like, XML, RDF, RDFS
- Easy to understand and use
  - Should be based on familiar KR idioms
- Adequate expressive power
  - Important: the richer the language is, the more inefficient the reasoning support becomes
- Formal semantics
- Efficient reasoning support

Web Ontology Language OWL

- Semantic Web led to requirement for a Web Ontology Language
- OWL is a W3C recommended, semantic markup language for publishing and sharing ontologies on Web
- OWL is developed as vocabulary extension of RDF and RDFS
- OWL is based on the earlier languages OIL and DAML+OIL
- OIL, DAML+OIL and OWL are based on Description Logics (DL)
  - OWL is a Web-friendly syntax for SHOIN
- Three species of OWL: OWL Full, OWL DL and OWL Lite
- All OWL species use the open world assumption



OWL Full

- It uses all the OWL languages primitives
- It allows free mixing of OWL with RDF Schema
- So, expressive that does not enforce a strict separation of classes, properties, individuals and data values
  - A class can be treated simultaneously as a collection of individuals and as an individual in its own right
- It is fully upward-compatible with RDF, both syntactically and semantically
- Unlikely to have complete (or efficient) reasoning support by the reasoning software
- Important: RDF documents will generally be in OWL Full, unless they are specifically constructed to be in OWL DL or OWL Lite

OWL DL (Description Logic)

- It is a sublanguage of OWL Full
- Provides maximum expressivity, while retaining computational completeness (i.e, all conclusions are guaranteed to be computable) and decidability
- Includes all OWL language constructs with certain restrictions
  - E.g., a sets of class, property and individual names must be disjoint
  - While a class may be a subclass of many classes, a class cannot be an individual of another class
- It permits efficient reasoning support
- Important: we lose full compatibility with RDF
- Note
  - Every RDF document is a legal OWL DL document
  - Every legal OWL DL document is a legal RDF document
  - Every legal OWL DL ontology is a legal OWL Full ontology
  - Every valid OWL DL conclusion is a valid OWL Full conclusion

OWL Lite

- It is a sublanguage (i.e., lighter version) of OWL DL, supports only a subset of the OWL language constructs
- Putting further restrictions, limits OWL DL to a subset of the OWL language constructors
  - E.g., OWL Lite excludes enumerated classes, disjointness statements, and arbitrary cardinality (only permits cardinality values of 0 or 1)
- Advantage of OWL Lite are
  - Easy to grasp
  - Easy to implement for tool builders
  - Provides a quick migration path for thesauri and other taxonomies
- Disadvantage is restricted expressivity
- Important:
  - OWL Lite is not simply an extension of RDF Schema
  - Every legal OWL Lite ontology is a legal OWL DL ontology
  - Every valid OWL Lite conclusion is a valid OWL DL conclusion

OWL Ontology Elements

- OWL ontology concern of,
  - Classes,
  - Properties,
  - Instances of classes, and
  - Relationships between the objects

- Synonymous terms in DL
  - Classes -> Concepts
  - Properties -> Roles
  - Object -> Individuals

Class

- A class defines a group of individuals that belong together and the classes are defined using owl:Class
- Important
  - owl:Thing- a built-in most general class and is the class of all individuals and is a superclass of all OWL classes in the OWL World
  - owl:Nothing- a built-in most specific class and is the class that has no instances (i.e., empty object class) and a subclass of all OWL classes

$$Thing = Nothing \cup \overline{Nothing}$$

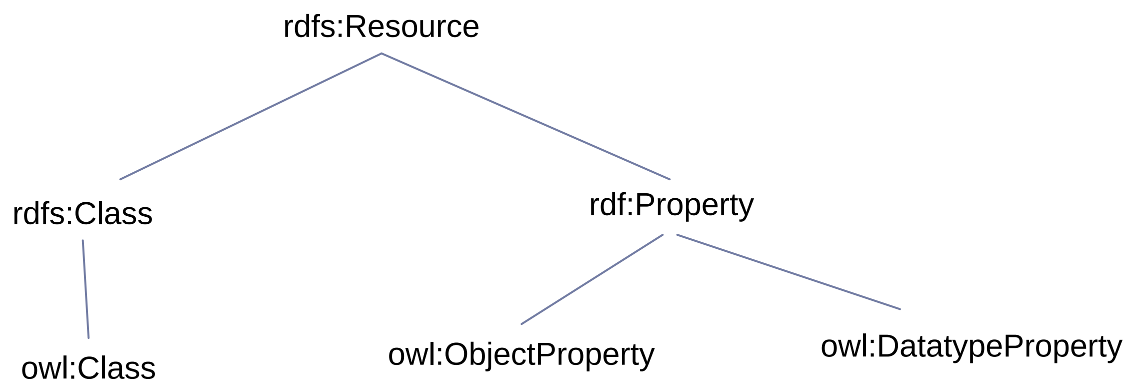
$$Nothing = \overline{Thing} = \overline{Nothing \cup \overline{Nothing}} = \overline{Nothing} \cap \overline{\overline{Nothing}} = \varnothing$$

- Note:
  - owl:Class is a subclass of rdfs:Class
  - in OWL, class hierarchy can be built using the rdfs:subClassOf

Properties

- OWL defines the properties,
  - Object property- relate individuals to other individuals (e.g. isTaughtBy, supervises, isStudentOf, isLocatedIn)
    - An object property is defined as an instance of the built-in OWL class owl:ObjectProperty
  - Datatype property- relate individuals to datatype values (e.g. author, title, phone, age, etc.)
    - A datatype property is defined as an instance of the built-in OWL class owl:DatatypeProperty
  - Annotation property- use to add uninterpreted information (e.g., versioning information, comment) to individuals, classes, and properties
- Important: both owl:ObjectProperty and owl:DatatypeProperty are *subclasses* of the RDF class rdf:Property

OWL Class and OWL Properties



Exchange Syntax

- OWL builds on RDF and uses RDF's XML based syntax
  - An OWL ontology turns into is a set of RDF triples
- Like wise any RDF graph, an OWL ontology graph can be written in many different syntactic forms of RDF/XML
- Alternative syntactic forms for OWL have also been defined
  - More readable XML based syntax
    - E.g., <owl:Class rdf:ID="Person"/>
    - The above can be alternatively represented by the following,

```
<rdf:Description rdf:about="#Person">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
```

- Important: A graphic syntax based on the conventions of UML (Unified Modelling Language)

Abstract Syntax

Abstract Syntax	DL Syntax
Descriptions (C)	
A	A
owl:Thing	$\top$
owl:Nothing	$\bot$
intersectionOf($C_1 \ldots C_n$)	$C_1 \sqcap \ldots \sqcap C_n$
unionOf($C_1 \ldots C_n$)	$C_1 \sqcup \ldots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf($o_1 \ldots o_n$)	$\{o_1\} \sqcup \ldots \sqcup \{o_n\}$
restriction(R someValuesFrom(C))	$\exists R.C$
restriction(R allValuesFrom(C))	$\forall R.C$
restriction(R hasValue(o))	$R : o$
restriction(R minCardinality(n))	$\geqslant n\, R$
restriction(R minCardinality(n))	$\leqslant n\, R$
restriction(U someValuesFrom(D))	$\exists U.D$
restriction(U allValuesFrom(D))	$\forall U.D$
restriction(U hasValue(v))	$U : v$
restriction(U minCardinality(n))	$\geqslant n\, U$
restriction(U maxCardinality(n))	$\leqslant n\, U$
Data Ranges (D)	
D	D
oneOf($v_1 \ldots v_n$)	$\{v_1\} \sqcup \ldots \sqcup \{v_n\}$
Object Properties (R)	
R	R
inv(R)	R^-
Datatype Properties (U)	
U	U
Individuals (o)	
o	o
Data Values (v)	
v	v

OWL DL Descriptions(C), Data Ranges(D), Object properties(R), Individuals(o), Datatype properties(U) and Data Values(v)

Property Restrictions

- In OWL we can declare that the class C satisfies certain conditions
  - All instances of C satisfy the conditions
- A (restriction) class is achieved through an owl:Restriction element
- This element contains an owl:onProperty element and one or more restriction declarations
- Defines restrictions on the kinds of values the property may take, owl:allValuesFrom, owl:someValuesFrom, owl:hasvalue
- We can specify minimum and maximum number using owl:minCardinality and owl:maxCardinality
  - Also, possible to specify a precise number using the same minimum and maximum number, by owl:cardinality

Property Restrictions (examples)

```
<owl:Class rdf:about="#PhD">
    <rdfs:subClassOf>
      <owl:Restriction>
            <owl:onProperty rdf:resource="#isSuperviseBy"/>
            <owl:allValuesFrom rdf:resource="#Professor"/>
     </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:about="#AcademicStaffMember">
    <rdfs:subClassOf>
      <owl:Restriction>
           <owl:onProperty rdf:resource="#teaches"/>
           <owl:someValuesFrom rdf:resource="#undergraduateCourse"/>
      </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Property Restrictions (examples)

```
<owl:Class rdf:about="#Person">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasParents"/>
            <owl:maxCardinality rdf:datatype=
                "&xsd;nonNegativeInteger“>2
            </owl:maxCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Examples

owl:equivalentClass defines equivalence of classes

```
<owl:Class rdf:ID="faculty">
   <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```

Enumeration using owl:oneOf

```
<owl:Class rdf:ID="weekdays">
      <owl:oneOf rdf:parseType="Collection">
         <owl:Thing rdf:about="#Monday"/>
         <owl:Thing rdf:about="#Tuesday"/>
         <owl:Thing rdf:about="#Wednesday"/>
         <owl:Thing rdf:about="#Thursday"/>
         <owl:Thing rdf:about="#Friday"/>
         <owl:Thing rdf:about="#Saturday"/>
         <owl:Thing rdf:about="#Sunday"/>
      </owl:oneOf>
</owl:Class>
```

Boolean Combinations (Examples)

Classes can be combined using Boolean operations (union, intersection, complement)

```
<owl:Class rdf:about="#course">
    <rdfs:subClassOf>
        <owl:Restriction>
                <owl:complementOf rdf:resource="#staffMember"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Axioms and Facts (OWL DL)

Abstract Syntax	DL Syntax
Class(A partial $C_1 \ldots C_n$)	$A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$
Class(A complete $C_1 \ldots C_n$)	$A \equiv C_1 \sqcap \ldots \sqcap C_n$
EnumeratedClass(A $o_1 \ldots o_n$)	$A \equiv \{o_1\} \sqcup \ldots \sqcup \{o_n\}$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$
EquivalentClasses($C_1 \ldots C_n$)	$C_1 \equiv \ldots \equiv C_n$
DisjointClasses($C_1 \ldots C_n$)	$C_i \sqcap C_j \sqsubseteq \bot, i \neq j$
Datatype(D)	
ObjectProperty(R super(R_1)...super(R_n)	$R \sqsubseteq R_i$
domain(C_1)...domain(C_m)	$\geqslant 1\, R \sqsubseteq C_i$
range(C_1)...range(C_ℓ)	$\top \sqsubseteq \forall R.C_i$
[inverseOf(R_0)]	$R \equiv R_0^-$
[Symmetric]	$R \equiv R^-$
[Functional]	$\top \sqsubseteq\, \leqslant 1\, R$
[InverseFunctional]	$\top \sqsubseteq\, \leqslant 1\, R^-$
[Transitive])	$Tr(R)$
SubPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$
EquivalentProperties($R_1 \ldots R_n$)	$R_1 \equiv \ldots \equiv R_n$
DatatypeProperty(U super(U_1)...super(U_n)	$U \sqsubseteq U_i$
domain(C_1)...domain(C_m)	$\geqslant 1\, U \sqsubseteq C_i$
range(D_1)...range(D_ℓ)	$\top \sqsubseteq \forall U.D_i$
[Functional])	$\top \sqsubseteq\, \leqslant 1\, U$
SubPropertyOf(U_1 U_2)	$U_1 \sqsubseteq U_2$
EquivalentProperties($U_1 \ldots U_n$)	$U_1 \equiv \ldots \equiv U_n$
AnnotationProperty(S)	
OntologyProperty(S)	
Individual(o type(C_1)...type(C_n)	$o \in C_i$
value(R_1 o_1)...value(R_n o_n)	$\langle o, o_i \rangle \in R_i$
value(U_1 v_1)...value(U_n v_n))	$\langle o, v_i \rangle \in U_i$
SameIndividual($o_1 \ldots o_n$)	$\{o_1\} \equiv \ldots \equiv \{o_n\}$
DifferentIndividuals($o_1 \ldots o_n$)	$\{o_i\} \sqsubseteq \neg\{o_j\}, i \neq j$

Axioms and Facts (examples)

E.g.1: Class Axioms,

Class(ed:Person partial owl:Thing)

Class(ed:Student partial ed:Person)

Class(ed:Country partial owl:Thing)

Class(ed:Italian complete ed:Person hasValue(ed:nationality ed:Italy))

E.g.2: Property Axioms,

DatatypeProperty(ed:age domain(ed:Person) range(xsd:integer))

ObjectProperty(ed:nationality domain(ed:Person) range(ed:Country)

E.g.3: Individual Axioms,

Individual(ed:India type(ed:Country))

Individual(ed:Italy type(ed:Country))

Individual(ed:Fausto type(ed:Italian)
value(ed:age “53”^^xsd:integer))

Individual(value(ed:nationality ed:India)
value(ed:age “32”^^xsd:integer))

Axioms and Facts

- A Class Axioms specifies the
  - Name of the class being described
  - A modality of “partial”, or “complete”
  - A sequence of property restrictions
  - Names of more general classes

Axioms and Facts

- A Property axiom specifies the
  - Name of the property
  - Its various features
- Individual Axiom specifies the
  - Name of the individual
  - Individual type
  - Object property and its value
  - Datatype property and data values
  - Identity of individuals

Class Axioms : owl:disjointWith

- Each owl:disjointWith statement asserts that the class extensions of the two class descriptions involved have no individuals in common
  - E.g., Student ⊓ Teacher ≡ ⊥
    - "Student is disjoint with Teacher"
- Axioms with rdfs:disjointWith declaring that two classes to be disjoint is a partial definition: it imposes a necessary but not sufficient condition on the class
- Implications:
    - a reasoner can deduce an inconsistency when an individual, A is stated to be an instance of both
    - similarly, a reasoner can also deduce that if A is an instance of class Teacher, then A is not an instance of class Student
- Important: use of owl:disjointWith is not allowed in OWL Lite

Individuals Axioms

- Individuals are defined with individual axioms (also called "facts"),
  - Facts about class membership and property values of individuals,

    Individual(ed:John type(ed:Student)
        value(ed:learningStyle ed:concrete-generic)
        value(vcard:FN “John Smith”^^xsd:string)
        value(stu:age “32”^^xsd:integer))

  - Facts about individual identity
    - OWL does not make unique name assumption
    - OWL provides three constructs for stating facts about the identity of individuals: owl:sameAs, owl:differentFrom, owl:AllDifferent

Special Properties

- owl:TransitiveProperty (transitive property)
  - E.g. “has better grade than”, “is ancestor of”
- owl:SymmetricProperty (symmetry)
  - E.g. “has same grade as”, “is sibling of”
- owl:FunctionalProperty defines a property that has at most one value for each object
  - E.g. “age”, “height”, “directSupervisor”
- owl:InverseFunctionalProperty defines a property for which two different objects cannot have the same value

Important: Not all of these can be specified for a particular object property as to retain the decidability of OWL DL properties (e.g., an object property specified as transitive, and their super-properties and their inverses cannot have their cardinality restricted, either via a functional part of property axioms or in cardinality restrictions)

Datatypes

- OWL supports XML Schema primitive datatypes
  - E.g., integer, real, string, …
- Strict separation between “object” classes and datatypes

Namespace

- Starts with a set of XML namespace declarations enclosed in an opening rdf:RDF tag
- Provide a means to unambiguously interpret identifiers and make the rest of the ontology presentation much more readable
- OWL depends on constructs defined by RDF, RDFS, and XML Schema datatypes

```
<rdf:RDF
    xmlns="http://www.disi.unitn.it/student#"
    xmlns:stu="http://www.disi.unitn.it/student#”
    xmlns:base=“http://www.disi.unitn.it/student#”
    xmlns:doc=“http://www.disi.unitn.it/document#”
    xmlns:owl ="http://www.w3.org/2002/07/owl#"
    xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd ="http://www.w3.org/2001/XLMSchema#”
  >
```

Namespace

- As an aid to writing lengthy URLs, it useful to provide a set of entity definitions in a document type declaration (DOCTYPE) that precedes the ontology definitions

```
<!DOCTYPE rdf:RDF [
   <!ENTITY stu  "http://www.disi.unitn.it/student#" >
   <!ENTITY doc "http://www.disi.unitn.it/student#" >
………….
]>

<rdf:RDF
   xmlns     ="&stu;"
   xmlns:stu ="&stu;"
   xml:base  ="&stu;"
   xml:doc   ="&doc;”
…
>
```

Namespace

- Advantage of DOCTYPE
  - changes made to the entity declarations will propagate through the ontology consistently
  - Allows referring ontology identifiers using attribute values
  - <owl:Class rdf:about="&stu;Qualification"/>, where, "&stu:Student" can be written in its expanded form as, http://www.disi.unitn.it/student#Qualification"
- Important: The names defined by the namespace declarations only have significance as parts of XML tags

Header

- An OWL ontology may start (after the namespace inclusion) with a collection of assertions for housekeeping purposes using owl:Ontology element

<owl:Ontology rdf:about="">

<rdfs:comment>A educational OWL ontology</rdfs:comment>

<owl:priorVersion rdf:resource="http://disi.unitn.it/course-ontology-26092010"/>

<owl:imports rdf:resource="http://drtc.isibang.ac.in/education/course"/>

<rdfs:label>Educational ontology</rdfs:label>

…

</owl:Ontology>

A complete OWL DL example (Demo)

Semantics

- Provides well defined semantics very similar to the semantics provided for DL
- OWL Mapping to equivalent DL
  - OWL Lite closely corresponds to SHIF(D)
  - OWL DL closely corresponds to SHOIN(D)
- However, what makes (???), OWL (specifically OWL DL) a SW language when semantics for this is very similar to the DL
  - Use of URI references for names
  - Use of XML Schema datatypes for data values
  - Allow the use of annotation properties
  - Frame-like abstract syntax
  - Ability to connect to documents in the Web

Reasoning

- Reasoning about Knowledge in Ontology

- Significance of reasoning:
  - checking consistency of the ontology and the knowledge
  - checking for unintended relationships between classes
  - automatically classifying instances in classes

Reasoning

- Consistency
  - x instance of classes A and B, but A and B are disjoint
  - This is an indication of an error in the ontology
- Classification
  - Certain property-value pairs are a sufficient condition for membership in a class A; if an individual x satisfies such conditions, we can conclude that x must be an instance of A
- Class membership
  - If x is an instance of a class C, and C is a subclass of D, then we can infer that x is an instance of D
- Equivalence of classes
  - If class A is equivalent to class B, and class B is equivalent to class C, then A is equivalent to C

Tool Support for OWL

- Ontology editors
  - Protege (http://protege.stanford.edu/)
  - OilEd (http://oiled.man.ac.uk/)
  - ...
- APIs
  - OWL-API (http://owlapi.sourceforge.net)
  - Jena (http://jena.sourceforge.net)
  - …
- OWL makes use of the reasoners such as,
  - FaCT++ (http://owl.man.ac.uk/factplusplus/)
  - Pellet (http://clarkparsia.com/pellet/)
  - KAON2 (http://kaon2.semanticweb.org/)
  - …

Further Readings

- OWL Web Ontology Language Semantics and Abstract Syntax. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/
- Horrocks, I., Patel-Schneider, Peter F., McGuinness, D. L., and Welty, C. A. OWL: a description logic based ontology language for the semantic web. Deborah L. McGuinness and Peter F. Patel-Schneider. From Description Logic Provers to Knowledge Representation Systems. In *The Description Logic Handbook: Theory, Implementation and Applications*, ed. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. Cambridge University Press, 2nd edition, August 2007, pp. 458--486.
- Antoniou, G. and Harmelen, F. V. A semantic web primer. http://www.emu.edu.tr/aelci/Courses/D-588/MIT.Press.A.Semantic.Web.Primer.eBook-TLFeBOOK.pdf