OWL and Description Logics

CUSO Winter School

Oscar Corcho
Ontology Engineering Group
Universidad Politécnica de Madrid, Spain

Acknowledgements: Ian Horrocks, Sean Bechoffer, and many others that we may have omitted.

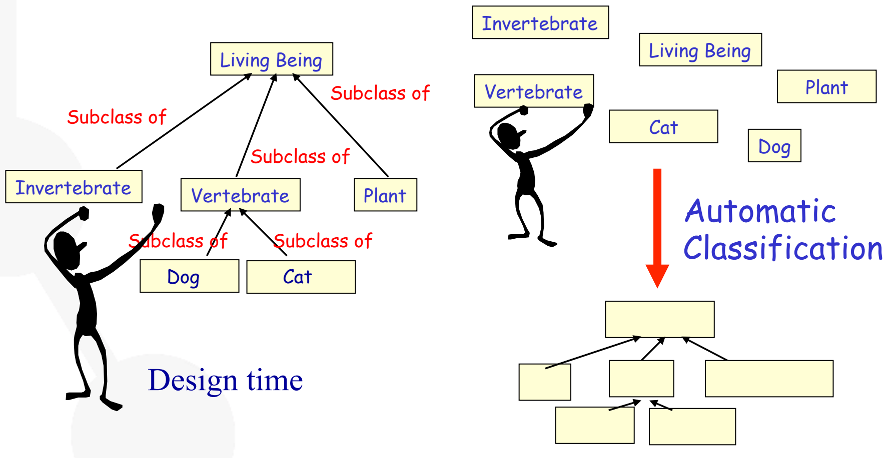
ocorcho@fi.upm.es
@ocorcho

07/02/2018
Champery, Switzerland

License

Description Logics

- **Automatic classification,** **done by the inference engine, at run-time**



What is Description Logic?

- A family of logic based Knowledge Representation formalisms
  - Descendants of semantic networks and KL-ONE
  - Describe domain in terms of concepts (classes), roles (relationships) and individuals
    - Specific languages characterised by the constructors and axioms used to assert knowledge about classes, roles and individuals.
    - Example: ALC (the least expressive language in DL that is propositionally closed)
      - Constructors: boolean (and, or, not)
      - Role restrictions

- Distinguished by:
  - Formal semantics (typically model theoretic)
    - Decidable fragments of FOL
    - Closely related to Propositional Modal & Dynamic Logics
  - Provision of inference services
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimised)

Motivation. Why Description Logic?

- DL and Semantic Networks / Frames
  - Semantic networks and frames allow describing knowledge in terms of concepts, properties and instances, and organising it in hierarchies
  - However, they lack from a formal support
    - Hence reasoning is not always understood in the same way
    - Especially important in multiple classification and exception handling
- DL and First Order Logic
  - First-order logic is undecidable
  - First-order logic is not focused on the definition of terminological knowledge bases (concepts, properties and instances)

Structure of DL Ontologies

- A DL ontology can be divided into two parts:
  - **Tbox** (Terminological KB): a set of axioms that describe the structure of a domain :
    - Doctor $\subseteq$ Person
    - Person $\subseteq$ Man $\cup$ Woman
    - HappyFather $\subseteq$ Man $\cap$ $\forall$hasDescendant.(Doctor $\cup$ $\forall$hasDescendant.Doctor)
  - **Abox** (Assertional KB): a set of axioms that describe a specific situation :
    - John $\in$ HappyFather
    - hasDescendant (John, Mary)
  - Other terms that have been used:
    - **RBox**
    - **EBox** (extensional box)

DL constructors

<table>
<tr><th>Construct</th><th>Syntax</th><th colspan="4">Language</th></tr>
<tr><td>Concept</td><td>A</td><td rowspan="4">FL_0</td><td rowspan="5">FL^-</td><td rowspan="8">AL</td><td rowspan="11">S[14]</td></tr>
<tr><td>Role name</td><td>R</td></tr>
<tr><td>Intersection</td><td>$C \cap D$</td></tr>
<tr><td>Value restriction</td><td>$\forall R.C$</td></tr>
<tr><td>Limited existential quantification</td><td>$\exists R$</td><td></td></tr>
<tr><td>Top or Universal</td><td>$\top$</td><td colspan="2"></td></tr>
<tr><td>Bottom</td><td>$\perp$</td><td colspan="2"></td></tr>
<tr><td>Atomic negation</td><td>$\neg A$</td><td colspan="2"></td></tr>
<tr><td>Negation[15]</td><td>$\neg C$</td><td colspan="3">C</td></tr>
<tr><td>Union</td><td>$C \cup D$</td><td colspan="3">U</td></tr>
<tr><td>Existential restriction</td><td>$\exists R.C$</td><td colspan="3">E</td></tr>
<tr><td>Number restrictions</td><td>$(\geq n\ R)\ (\leq n\ R)$</td><td colspan="4">N</td></tr>
<tr><td>Nominals</td><td>$\{a_1 \ldots a_n\}$</td><td colspan="4">O</td></tr>
<tr><td>Role hierarchy</td><td>$R \subseteq S$</td><td colspan="4">H</td></tr>
<tr><td>Inverse role</td><td>R^-</td><td colspan="4">I</td></tr>
<tr><td>Qualified number restriction</td><td>$(\geq n\ R.C)\ (\leq n\ R.C)$</td><td colspan="4">Q</td></tr>
</table>

≥3 hasChild, ≤1 hasMother

{Colombia, Argentina, México, ...} → MercoSur countries

hasChild⁻ (hasParent)

≤2 hasChild.Female, ≥1 hasParent.Male

[12] Names previously used for Description Logics were: terminological knowledge representation languages, concept languages, term subsumption languages, and KL-ONE-based knowledge representation languages.
[13] In this table, we use A to refer to atomic concepts (concepts that are the basis for building other concepts), C and D to any concept definition, R to atomic roles and S to role definitions. FL is used for structural DL languages and AL for attributive languages (Baader et al., 2003).
[14] S is the name used for the language ALC_{R+}, which is composed of ALC plus transitive roles.
[15] ALC and ALCUE are equivalent languages, since union (U) and existential restriction (E) can be represented using negation (C).

Other:

Concrete datatypes: hasAge.(<21)

Transitive roles: hasChild* (descendant)

Role composition: hasParent o hasBrother (uncle)

Most common constructors in class definitions

- Intersection: $C_1 \cap \ldots \cap C_n$ — Human ∩ Male
- Union: $C_1 \cup \ldots \cup C_n$ — Doctor ∪ Lawyer
- Negation: ¬C — ¬Male
- Nominals: $\{x_1\} \cup \ldots \cup \{x_n\}$ — {john} ∪ ... ∪ {mary}
- Universal restriction: ∀P.C — ∀hasChild.Doctor
- Existential restriction: ∃P.C — ∃hasChild.Lawyer
- Maximum cardinality: ≤nP — ≤3hasChild
- Minimum cardinality: ≥nP — ≥1hasChild
- Specific Value: ∃P.{x} — ∃hasColleague.{Matthew}

- Nesting of constructors can be arbitrarily complex
  - Person ∩ ∀hasChild.(Doctor ∪ ∃hasChild.Doctor)
- Lots of redundancy
  - A∪B is equivalent to ¬(¬ A ∩ ¬B)
  - ∃P.C is equivalent to ¬∀P. ¬C

Most common axioms

- Classes

Subclass	$C1 \subseteq C2$	$Human \subseteq Animal \cap Biped$
Equivalence	$C1 \equiv C2$	$Man \equiv Human \cap Male$
Disjointness	$C1 \cap C2 \subseteq \perp$	$Male \cap Female \subseteq \perp$

- Properties/roles

Subproperty	$P1 \subseteq P2$	$hasDaughter \subseteq hasChild$
Equivalence	$P1 \equiv P2$	$cost \equiv price$
Inverse	$P1 \equiv P2^-$	$hasChild \equiv hasParent^-$
Transitive	$P+ \subseteq P$	$ancestor+ \subseteq ancestor$
Functional	$T \subseteq \leq 1P$	$T \subseteq \leq 1hasMother$
InverseFunctional	$T \subseteq \leq 1P^-$	$T \subseteq \leq 1hasPassportID^-$

- Individuals

Equivalence	$\{x1\} \equiv \{x2\}$	$\{oeg:OscarCorcho\} \equiv \{img:Oscar\}$
Different	$\{x1\} \equiv \neg\{x2\}$	$\{john\} \equiv \neg\{peter\}$

- Most axioms are reducible to inclusion ($\cup$)
  - $C \equiv D$ iff both $C \subseteq D$ and $D \subseteq C$
  - C disjoint D iff $C \subseteq \neg D$

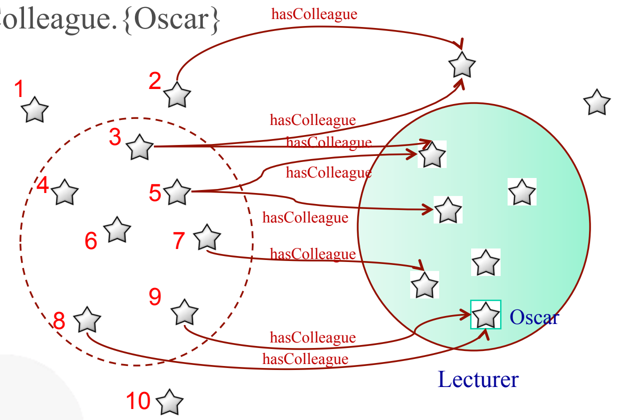
Description Logics

Understand the meaning of universal and existential restrictions

- Decide which is the set that we are defining with different expressions, taking into account Open and Close World Assumptions

Do we understand these constructors?

- ∃hasColleague.Lecturer
- ∀hasColleague.Lecturer
- ∃hasColleague.{Oscar}



- X must be Y, X is an Y that... → $X \subseteq Y$
- X is exactly Y, X is the Y that... → $X \equiv Y$
- X is not Y (*not the same as X is whatever it is not Y*) → $X \subseteq \neg Y$
- X and Y are disjoint → $X \cap Y \subseteq \bot$
- X is Y or Z → $X \subseteq Y \cup Z$
- X is Y for which property P has only instances of Z as values → $X \subseteq Y \cap (\forall P.Z)$
- X is Y for which property P has at least an instance of Z as a value → $X \subseteq Y \cap (\exists P.Z)$
- X is Y for which property P has at most 2 values → $X \subseteq Y \cap (\leq 2.P)$
- Individual X is a Y → $X \in Y$

Chunk 1. Formalize in DL, and then in OWL DL

1. Concept definitions:
   Grass and trees must be plants. Leaves are parts of a tree but there are other parts of a tree that are not leaves. A dog must eat bones, at least. A sheep is an animal that must only eat grass. A giraffe is an animal that must only eat leaves. A mad cow is a cow that eats brains that can be part of a sheep.

2. Restrictions:
   Animals or part of animals are disjoint with plants or parts of plants.

3. Properties:
   Eats is applied to animals. Its inverse is eaten_by.

4. Individuals:
   Tom.
   Flossie is a cow.
   Rex is a dog and is a pet of Mick.
   Fido is a dog.
   Tibbs is a cat.

Chunk 2. Formalize in DL, and then in OWL DL

1. Concept definitions:
   Bicycles, buses, cars, lorries, trucks and vans are vehicles. There are several types of companies: bus companies and haulage companies.
   An elderly person must be adult. A kid is (exactly) a person who is young. A man is a person who is male and is adult. A woman is a person who is female and is adult. A grown up is a person who is an adult. And old lady is a person who is elderly and female. Old ladies must have some animal as pets and all their pets are cats.

2. Restrictions:
   Youngs are not adults, and adults are not youngs.

3. Properties:
   Has mother and has father are subproperties of has parent.

4. Individuals:
   Kevin is a person.
   Fred is a person who has a pet called Tibbs.
   Joe is a person who has at most one pet. He has a pet called Fido.
   Minnie is a female, elderly, who has a pet called Tom.

Chunk 3. Formalize in DL, and then in OWL DL

1. Concept definitions:
   A magazine is a publication. Broadsheets and tabloids are newspapers. A quality broadsheet is a type of broadsheet. A red top is a type of tabloid. A newspaper is a publication that must be either a broadsheet or a tabloid.
   White van mans must read only tabloids.

2. Restrictions:
   Tabloids are not broadsheets, and broadsheets are not tabloids.

3. Properties:
   The only things that can be read are publications.

4. Individuals:
   Daily Mirror
   The Guardian and The Times are broadsheets
   The Sun is a tabloid

Chunk 4. Formalize in DL, and then in OWL DL

1. Concept definitions:
   A pet is a pet of something. An animal must eat something. A vegetarian is an animal that does not eat animals nor parts of animals. Ducks, cats and tigers are animals. An animal lover is a person who has at least three pets. A pet owner is a person who has animal pets. A cat liker is a person who likes cats. A cat owner is a person who has cat pets. A dog liker is a person who likes dogs. A dog owner is a person who has dog pets.

2. Restrictions:
   Dogs are not cats, and cats are not dogs.

3. Properties:
   Has pet is defined between persons and animals. Its inverse is is_pet_of.

4. Individuals:
   Dewey, Huey, and Louie are ducks.
   Fluffy is a tiger.
   Walt is a person who has pets called Huey, Louie and Dewey.

Chunk 5. Formalize in DL, and then in OWL DL

1. Concept definitions

A driver must be adult. A driver is a person who drives vehicles. A lorry driver is a person who drives lorries. A haulage worker is who works for a haulage company or for part of a haulage company. A haulage truck driver is a person who drives trucks ans works for part of a haulage company. A van driver is a person who drives vans. A bus driver is a person who drives buses. A white van man is a man who drives white things and vans.

2. Restrictions:

--

3. Properties:

The service number is an integer property with no restricted domain

4. Individuals:

Q123ABC is a van and a white thing.
The42 is a bus whose service number is 42.
Mick is a male who read Daily Mirror and drives Q123ABC.

Chunk 1. Formalisation in DL

$$grass \subseteq plant$$

$$tree \subseteq plant$$

$$leaf \subseteq \exists partOf.tree$$

$$dog \subseteq \exists eats.bone$$

$$sheep \subseteq animal \cap \forall eats.grass$$

$$giraffe \subseteq animal \cap \forall eats.leaf$$

$$madCow \equiv cow \cap \exists eats.(brain \cap \exists partOf.sheep)$$

$$(animal \cup \exists partOf.animal) \cap (plant \cup \exists partOf.plant) \subseteq \perp$$

Chunk 2. Formalisation in DL

$bicycle \sqsubseteq vehicle; bus \sqsubseteq vehicle; car \sqsubseteq vehicle; lorry \sqsubseteq vehicle; truck \sqsubseteq vehicle$

$busCompany \sqsubseteq company; haulageCompany \sqsubseteq company$

$elderly \sqsubseteq person \sqcap adult$

$kid \equiv person \sqcap young$

$man \equiv person \sqcap male \sqcap adult$

$woman \equiv person \sqcap female \sqcap adult$

$grownUp \equiv person \sqcap adult$

$oldLady \equiv person \sqcap female \sqcap elderly$

$oldLady \sqsubseteq \exists hasPet.animal \sqcap \forall hasPet.cat$

$young \sqcap adult \sqsubseteq \bot$

$hasMother \sqsubseteq hasParent$

$hasFather \sqsubseteq hasParent$

Chunk 3. Formalisation in DL

$magazine \sqsubseteq publication$

$broadsheet \sqsubseteq newspaper$

$tabloid \sqsubseteq newspaper$

$qualityBroadsheet \sqsubseteq broadsheet$

$redTop \sqsubseteq tabloid$

$newspaper \sqsubseteq publication \sqcap (broadsheet \sqcup tabloid)$

$whiteVanMan \sqsubseteq \forall reads.tabloid$

$tabloid \sqcap broadsheet \sqsubseteq \bot$

Chunk 4. Formalisation in DL

$pet \equiv \exists isPetOf.\mathrm{T}$

$animal \subseteq \exists eats.\mathrm{T}$

$vegetarian \equiv animal \cap \forall eats.\neg animal \cap \forall eats.\neg(\exists partOf.animal)$

$duck \subseteq animal; cat \subseteq animal; tiger \subseteq animal$

$animalLover \equiv person \cap (\geq 3hasPet)$

$petOwner \equiv person \cap \exists hasPet.animal$

$catLike \equiv person \cap \exists likes.cat; catOwner \equiv person \cap \exists hasPet.cat$

$dogLike \equiv person \cap \exists likes.dog; dogOwner \equiv person \cap \exists hasPet.dog$

$dog \cap cat \subseteq \perp$

Chunk 5. Formalisation in DL

$driver \subseteq adult$

$driver \equiv person \cap \exists drives.vehicle$

$lorryDriver \equiv person \cap \exists drives.lorry$

$haulageWorke \equiv \exists worksFor.(haulageCompany \cup \exists partOf.haulageCompany)$

$haulageTruckDriver \equiv person \cap \exists drives.truck \cap$

$\quad \exists worksFor.(\exists partOf.haulageCompany)$

$vanDriver \equiv person \cap \exists drives.van$

$busDriver \equiv person \cap \exists drives.bus$

$whiteVanMan \equiv man \cap \exists drives.(whiteThing \cap van)$

Inference. Basic Inference Tasks

- Subsumption – check knowledge is correct (captures intuitions)
  - Does C subsume D w.r.t. ontology O? (in *every* model I of O, $C^I \subseteq D^I$)
- Equivalence – check knowledge is minimally redundant (no unintended synonyms)
  - Is C equivalent to D w.r.t. O? (in *every* model I of O, $C^I = D^I$)
- Consistency – check knowledge is meaningful (classes can have instances)
  - Is C satisfiable w.r.t. O? (there exists *some* model I of O s.t. $C^I \neq \emptyset$)
- Instantiation and querying
  - Is x an instance of C w.r.t. O? (in *every* model I of O, $x^I \in C^I$)
  - Is (x,y) an instance of R w.r.t. O? (in *every* model I of O, $(x^I, y^I) \in R^I$)
- All reducible to KB satisfiability or concept satisfiability w.r.t. a KB
- Can be decided using highly optimised tableaux reasoners

Interesting results (I). Automatic classification

And old lady is a person who is elderly and female.
Old ladies must have some animal as pets and all their pets are cats.

$$elderly \subseteq person \cap adult$$

$$woman \equiv person \cap female \cap adult$$

$$catOwner \equiv person \cap \exists hasPet.cat$$

$$oldLady \equiv person \cap female \cap elderly$$

$$oldLady \subseteq \exists hasPet.animal \cap \forall hasPet.cat$$

We obtain:
Old ladies must be women.
Every old lady must have a pet cat
Hence, every old lady must be a cat owner

$$oldLady \subseteq woman \cap elderly \cap catOwner$$

Interesting results (II). Instance classification

A pet owner is a person who has animal pets
Old ladies must have some animal as pets and all their pets are cats.
Has pet has domain person and range animal
Minnie is a female, elderly, who has a pet called Tom.

$$petOwner \equiv person \cap \exists hasPet.animal$$
$$oldLady \subseteq \exists hasPet.animal \cap \forall hasPet.cat$$
$$hasPet \subseteq (person, animal)$$
$$Minnie \in female \cap elderly$$
$$hasPet(Minnie, Tom)$$

We obtain:

Minnie is a person
Hence, Minnie is an old lady
Hence, Tom is a cat

$$Minnie \in person; Tom \in animal$$
$$Minnie \in petOwner$$
$$Minnie \in oldLady$$
$$Tom \in cat$$

Interesting results (III). Instance classification and redundancy detection

An animal lover is a person who has at least three pets
Walt is a person who has pets called Huey, Louie and Dewey.

$$animalLover \equiv person \cap (\geq 3 hasPet)$$
$$Walt \in person$$
$$hasPet(Walt, Huey)$$
$$hasPet(Walt, Louie)$$
$$hasPet(Walt, Dewey)$$

We obtain:

Walt is an animal lover
Walt is a person is redundant

$$Walt \in animalLover$$

Interesting results (IV). Instance classification

A van is a type of vehicle
A driver must be adult
A driver is a person who drives vehicles
A white van man is a man who drives vans and white things
White van mans must read only tabloids
Q123ABC is a white thing and a van
Mick is a male who reads Daily Mirror and drives Q123ABC

$$van \sqsubseteq vehicle$$
$$driver \sqsubseteq adult$$
$$driver \equiv person \sqcap \exists drives.vehicle$$
$$whiteVanMan \equiv man \sqcap \exists drives.(van \sqcap whiteThing)$$
$$whiteVanMan \sqsubseteq \forall reads.tabloid$$
$$Q123ABC \in whiteThing \sqcap van$$
$$Mick \in male$$
$$reads(Mick, DailyMirror)$$
$$drives(Mick, Q123ABC)$$

We obtain:

Mick is an adult
Mick is a white van man
Daily Mirror is a tabloid

$$Mick \in adult$$
$$Mick \in whiteVanMan$$
$$DailyMirror \in tabloid$$

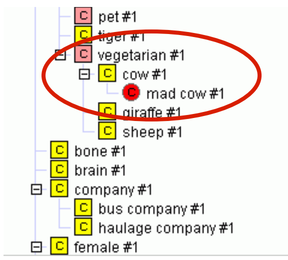
Interesting results (V). Consistency checking

Cows are vegetarian.
A vegetarian is an animal that does not eat animals nor parts of animals.
A mad cow is a cow that eats brains that can be part of a sheep

$$cow \sqsubseteq vegetarian$$
$$vegetarian \equiv animal \sqcap \forall eats.\neg animal \sqcap$$
$$\forall eats.\neg(\exists partOf.animal))$$
$$madCow \equiv cow \sqcap \exists eats.(brain \sqcup \exists partOf.sheep)$$
$$(animal \sqcup \exists partOf.animal) \sqcap (plant \sqcup \exists partOf.plant) \sqsubseteq \perp$$

We obtain:

Mad cow is unsatisfiable



Tableaux Algorithms

- Try to prove satisfiability of a knowledge base
- How do they work
  - They try to build a model of input concept C
    - Tree model property
      - If there is a model, then there is a tree shaped model
    - If no tree model can be found, then input concept unsatisfiable
  - Decompose C syntactically
    - Work on concepts in negation normal form (De Morgan's laws)
    - Use of tableaux expansion rules
    - If non-deterministic rules are applied, then there is search
  - Stop (and backtrack) if clash
    - E.g. A(x), ¬A(x)
  - Blocking (cycle check) ensures termination for more expressive logics

- The algorithm finishes when no more rules can be applied or a conflict is detected

Tableaux rules for ALC and for transitive roles

$x \bullet \{C_1 \sqcap C_2, \ldots\}$	$\rightarrow_\sqcap$	$x \bullet \{C_1 \sqcap C_2, C_1, C_2, \ldots\}$
$x \bullet \{C_1 \sqcup C_2, \ldots\}$	$\rightarrow_\sqcup$	$x \bullet \{C_1 \sqcup C_2, C, \ldots\}$ for $C \in \{C_1, C_2\}$
$x \bullet \{\exists R.C, \ldots\}$	$\rightarrow_\exists$	$x \bullet \{\exists R.C, \ldots\}$ $\xrightarrow{R}$ $y \bullet \{C\}$
$x \bullet \{\forall R.C, \ldots\}$ $\xrightarrow{R}$ $y \bullet \{\ldots\}$	$\rightarrow_\forall$	$x \bullet \{\forall R.C, \ldots\}$ $\xrightarrow{R}$ $y \bullet \{C, \ldots\}$
$x \bullet \{\forall R.C, \ldots\}$ $\xrightarrow{R}$ $y \bullet \{\ldots\}$	$\rightarrow_{\forall_+}$	$x \bullet \{\forall R.C, \ldots\}$ $\xrightarrow{R}$ $y \bullet \{\forall R.C, \ldots\}$

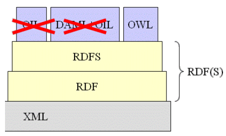
Tableaux examples and exercises

- Example
  - $\exists S.C \wedge \forall S.(\neg C \vee \neg D) \wedge \exists R.C \wedge \forall R.(\exists R.C)$

- Exercise 1
  - $\exists R.(\exists R.D) \wedge \exists S.\neg D \wedge \forall S.(\exists R.D)$

- Exercise 2
  - $\exists R.(C \vee D) \wedge \forall R.\neg C \wedge \neg\exists R.D$

OWL (Web Ontology Language) 1



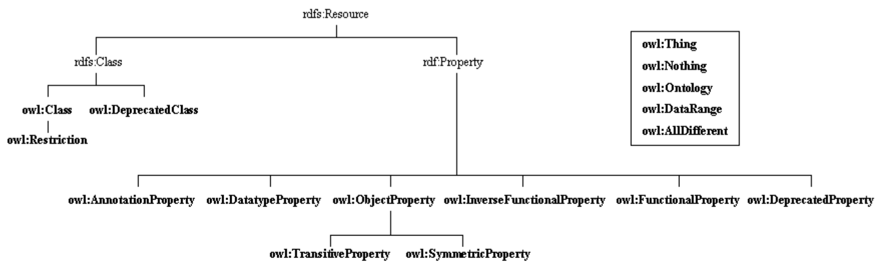
W3C Recommendation (February 2004)

Built on top of RDF(S)

3 layers:
- OWL Lite
  - A small subset of primitives
  - Easier for frame-based tools to transition to
- OWL DL
  - Description logic
  - Decidable reasoning
- OWL Full
  - RDF extension, allows metaclasses

Several syntaxes:
- Abstract syntax
- Manchester syntax
- RDF/XML

Class taxonomy of the OWL KR ontology



Property list of the OWL KR ontology

Property name	domain	range
owl:intersectionOf	owl:Class	rdf:List
owl:unionOf	owl:Class	rdf:List
owl:complementOf	owl:Class	owl:Class
owl:oneOf	owl:Class	rdf:List
owl:onProperty	owl:Restriction	rdf:Property
owl:allValuesFrom	owl:Restriction	rdfs:Class
owl:hasValue	owl:Restriction	*not specified*
owl:someValuesFrom	owl:Restriction	rdfs:Class
owl:minCardinality	owl:Restriction	xsd:nonNegativeInteger OWL Lite: {0,1} OWL DL/Full: {0,..,N}
owl:maxCardinality	owl:Restriction	xsd:nonNegativeInteger OWL Lite: {0,1} OWL DL/Full: {0,..,N}
owl:cardinality	owl:Restriction	xsd:nonNegativeInteger OWL Lite: {0,1} OWL DL/Full: {0,..,N}
owl:inverseOf	owl:ObjectProperty	owl:ObjectProperty
owl:sameAs	owl:Thing	owl:Thing
owl:equivalentClass	owl:Class	owl:Class
owl:equivalentProperty	rdf:Property	rdf:Property
owl:sameIndividualAs	owl:Thing	owl:Thing
owl:differentFrom	owl:Thing	owl:Thing
owl:disjointWith	owl:Class	owl:Class
owl:distinctMembers	owl:AllDifferent	rdf:List
owl:versionInfo	*not specified*	*not specified*
owl:priorVersion	owl:Ontology	owl:Ontology
owl:incompatibleWith	owl:Ontology	owl:Ontology
owl:backwardCompatibleWith	owl:Ontology	owl:Ontology
owl:imports	owl:Ontology	owl:Ontology

OWL: Most common constructors in class definitions and axioms for

Intersection:	$C_1 \cap ... \cap C_n$	intersectionOf	Human ∩ Male
Union:	$C_1 \cup ... \cup C_n$	unionOf	Doctor ∪ Lawyer
Negation:	$\neg C$	complementOf	¬Male
Nominals:	$\{x_1\} \cup ... \cup \{x_n\}$	oneOf	{john} ∪ ... ∪ {mary}
Universal restriction:	$\forall P.C$	allValuesFrom	∀hasChild.Doctor
Existential restriction:	$\exists P.C$	someValuesFrom	∃hasChild.Lawyer
Maximum cardinality:	$\leq nP$	maxCardinality	≤3hasChild
Minimum cardinality:	$\geq nP$	minCardinality	≥1hasChild
Specific Value:	$\exists P.\{x\}$	hasValue	∃hasColleague.{Matthew}

Subclass	$C1 \subseteq C2$	subClassOf	Human ⊆ Animal ∩ Biped
Equivalence	$C1 \equiv C2$	equivalentClass	Man ≡ Human ∩ Male
Disjointness	$C1 \cap C2 \subseteq \perp$	disjointWith	Male ∩ Female ⊆ ⊥
Subproperty	$P1 \subseteq P2$	subPropertyOf	hasDaughter ⊆ hasChild
Equivalence	$P1 \equiv P2$	equivalentProperty	cost ≡ price
Inverse	$P1 \equiv P2-$	inverseOf	hasChild ≡ hasParent-
Transitive	$P+ \subseteq P$	TransitiveProperty	ancestor+ ⊆ ancestor
Functional	$T \subseteq \leq 1P$	FunctionalProperty	T ⊆ ≤1hasMother
InverseFunctional	$T \subseteq \leq 1P-$	InverseFunctionalProperty	T ⊆ ≤1hasPassportID-
Equivalence	$\{x1\} \equiv \{x2\}$	sameIndividualAs	{oeg:OscarCorcho}≡{img:Oscar}
Different	$\{x1\} \equiv \neg \{x2\}$	differentFrom, AllDifferent	{john} ≡ ¬ {peter}

OWL DL

Class expressions allowed in: rdfs:domain, rdfs:range, rdfs:subClassOf
owl:intersectionOf, owl:equivalentClass, owl:allValuesFrom, owl:someValuesFrom

Values are not restricted (0..N) in: owl:minCardinality, owl:maxCardinality, owl:cardinality

owl:DataRange, rdf:List, rdf:first, rdf:rest, rdf:nil

owl:hasValue (*daml:hasValue*)
owl:oneOf (*daml:oneOf*)
owl:unionOf (*daml:unionOf*), owl:complementOf (*daml:complementOf*)
owl:disjointWith (*daml:disjointWith*)

OWL Lite

owl:Ontology (*daml:Ontology*),
owl:versionInfo (*daml:versionInfo*),
owl:imports (*daml:imports*),
owl:backwardCompatibleWith,
owl:incompatibleWith, owl:priorVersion,
owl:DeprecatedClass,
owl:DeprecatedProperty

owl:Class (*daml:Class*),
owl:Restriction (*daml:Restriction*),
owl:onProperty (*daml:onProperty*),
owl:allValuesFrom (*daml:toClass*) (only with class identifiers and named datatypes),
owl:someValuesFrom (*daml:hasClass*) (only with class identifiers and named datatypes),
owl:minCardinality (*daml:minCardinality*; restricted to {0,1}),
owl:maxCardinality (*daml:maxCardinality*; restricted to {0,1}),
owl:cardinality (*daml:cardinality*; restricted to {0,1})

owl:intersectionOf (only with class identifiers and property restrictions)

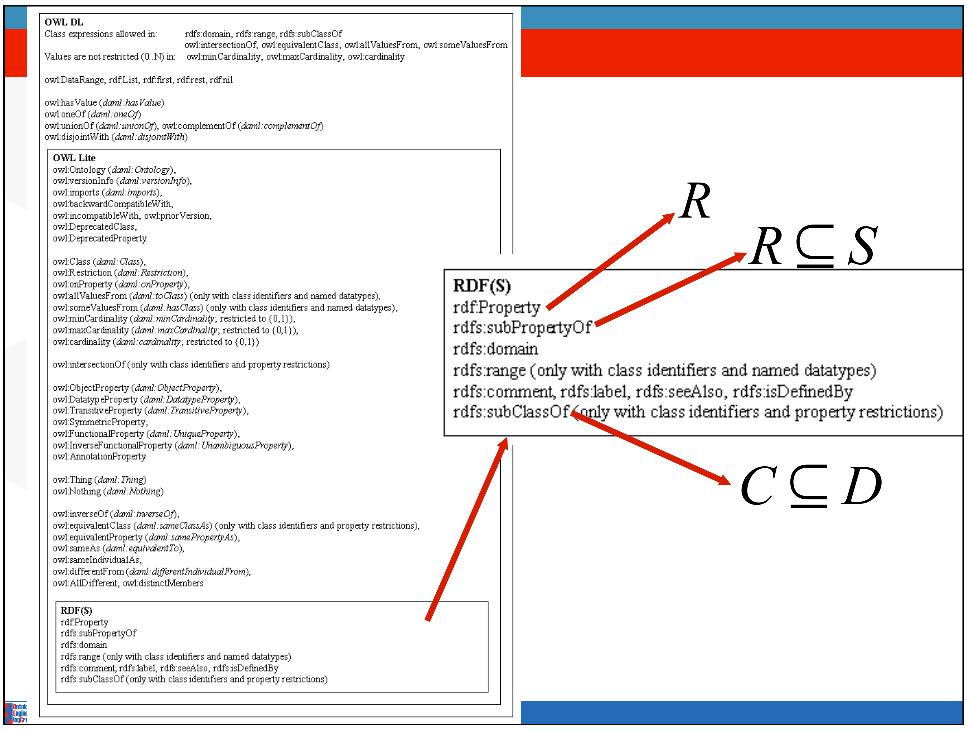
owl:ObjectProperty (*daml:ObjectProperty*),
owl:DatatypeProperty (*daml:DatatypeProperty*),
owl:TransitiveProperty (*daml:TransitiveProperty*),
owl:SymmetricProperty,
owl:FunctionalProperty (*daml:UniqueProperty*),
owl:InverseFunctionalProperty (*daml:UnambiguousProperty*),
owl:AnnotationProperty

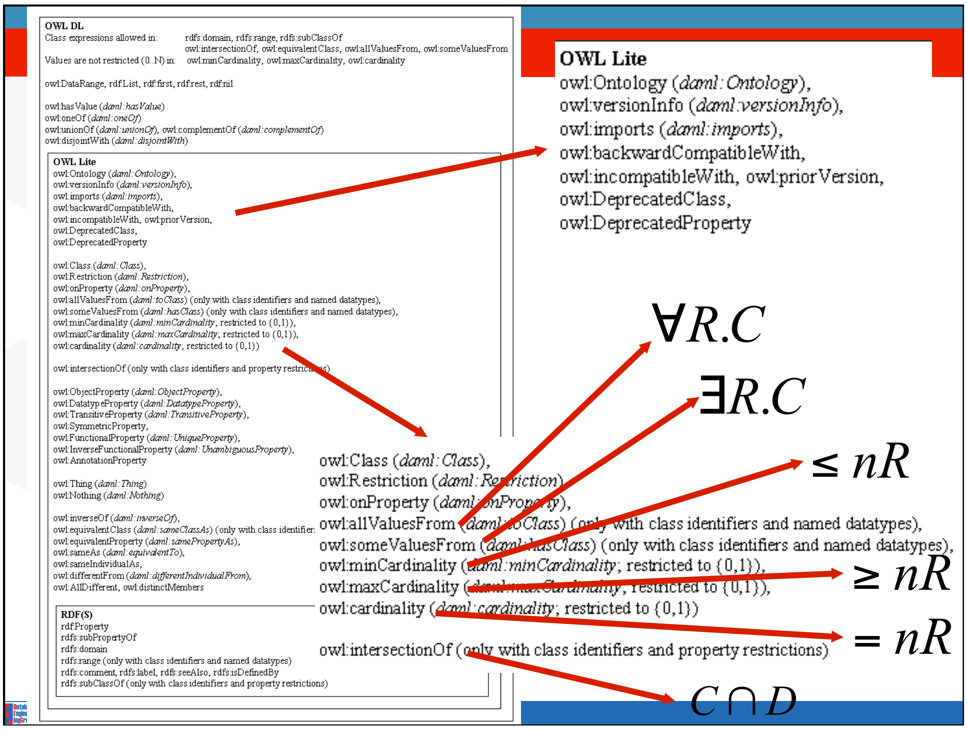
owl:Thing (*daml:Thing*)
owl:Nothing (*daml:Nothing*)

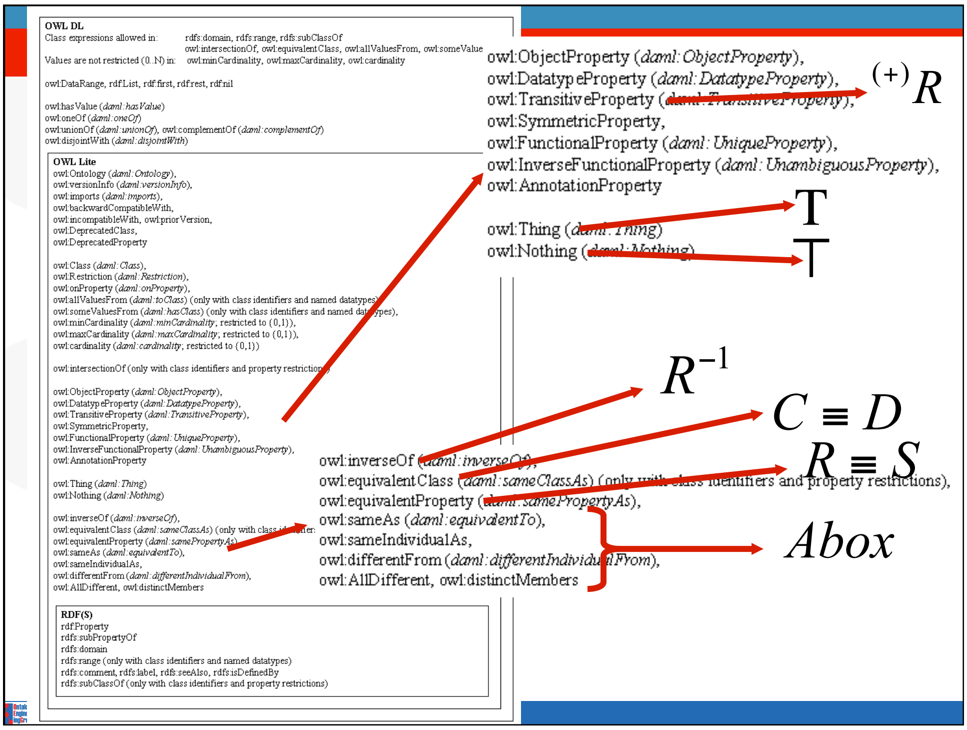
owl:inverseOf (*daml:inverseOf*),
owl:equivalentClass (*daml:sameClassAs*) (only with class identifiers and property restrictions),
owl:equivalentProperty (*daml:samePropertyAs*),
owl:sameAs (*daml:equivalentTo*),
owl:sameIndividualAs,
owl:differentFrom (*daml:differentIndividualFrom*),
owl:AllDifferent, owl:distinctMembers

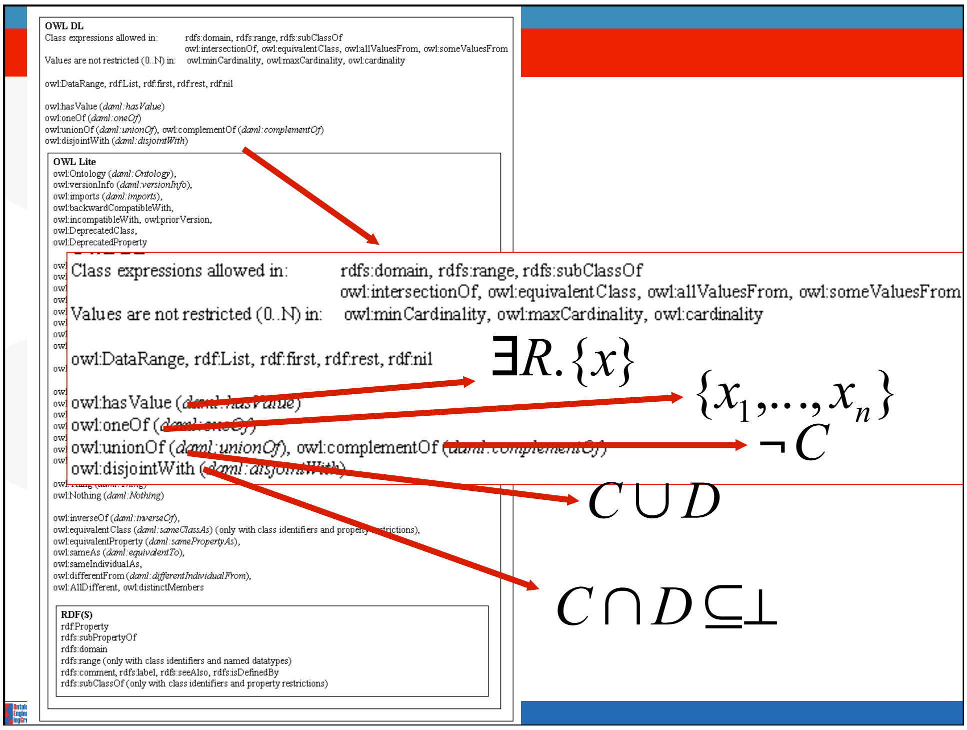
RDF(S)

rdf:Property
rdfs:subPropertyOf
rdfs:domain
rdfs:range (only with class identifiers and named datatypes)
rdfs:comment, rdfs:label, rdfs:seeAlso, rdfs:isDefinedBy
rdfs:subClassOf (only with class identifiers and property restrictions)


OWL DL
Class expressions allowed in: rdfs:domain, rdfs:range, rdfs:subClassOf
owl:intersectionOf, owl:equivalentClass, owl:allValuesFrom, owl:someValuesFrom
Values are not restricted (0..N) in: owl:minCardinality, owl:maxCardinality, owl:cardinality
owl:DataRange, rdf:List, rdf:first, rdf:rest, rdf:nil
owl:hasValue (daml:hasValue)
owl:oneOf (daml:oneOf)
owl:unionOf (daml:unionOf), owl:complementOf (daml:complementOf)
owl:disjointWith (daml:disjointWith)
OWL Lite
owl:Ontology (daml:Ontology),
owl:versionInfo (daml:versionInfo),
owl:imports (daml:imports),
owl:backwardCompatibleWith,
owl:incompatibleWith, owl:priorVersion,
owl:DeprecatedClass,
owl:DeprecatedProperty
owl:Class (daml:Class),
owl:Restriction (daml:Restriction),
owl:onProperty (daml:onProperty),
owl:allValuesFrom (daml:toClass) (only with class identifiers and named datatypes),
owl:someValuesFrom (daml:hasClass) (only with class identifiers and named datatypes),
owl:minCardinality (daml:minCardinality, restricted to {0,1}),
owl:maxCardinality (daml:maxCardinality, restricted to {0,1}),
owl:cardinality (daml:cardinality, restricted to {0,1})
owl:intersectionOf (only with class identifiers and property restrictions)
owl:ObjectProperty (daml:ObjectProperty),
owl:DatatypeProperty (daml:DatatypeProperty),
owl:TransitiveProperty (daml:TransitiveProperty),
owl:SymmetricProperty,
owl:FunctionalProperty (daml:UniqueProperty),
owl:InverseFunctionalProperty (daml:UnambiguousProperty),
owl:AnnotationProperty
owl:Thing (daml:Thing)
owl:Nothing (daml:Nothing)
owl:inverseOf (daml:inverseOf),
owl:equivalentClass (daml:sameClassAs) (only with class identifiers and property restrictions),
owl:equivalentProperty (daml:samePropertyAs),
owl:sameAs (daml:equivalentTo),
owl:sameIndividualAs,
owl:differentFrom (daml:differentIndividualFrom),
owl:AllDifferent, owl:distinctMembers
RDF(S)
rdf:Property
rdfs:subPropertyOf
rdfs:domain
rdfs:range (only with class identifiers and named datatypes)
rdfs:comment, rdfs:label, rdfs:seeAlso, rdfs:isDefinedBy
rdfs:subClassOf (only with class identifiers and property restrictions)
R
R ⊆ S
RDF(S)
rdf:Property
rdfs:subPropertyOf
rdfs:domain
rdfs:range (only with class identifiers and named datatypes)
rdfs:comment, rdfs:label, rdfs:seeAlso, rdfs:isDefinedBy
rdfs:subClassOf (only with class identifiers and property restrictions)
C ⊆ D

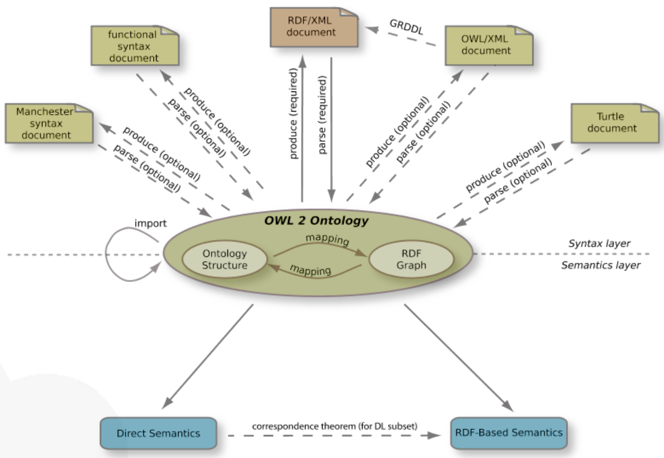

OWL Lite
owl:Ontology (daml:Ontology),
owl:versionInfo (daml:versionInfo),
owl:imports (daml:imports),
owl:backwardCompatibleWith,
owl:incompatibleWith, owl:priorVersion,
owl:DeprecatedClass,
owl:DeprecatedProperty
∀R.C
∃R.C
≤ nR
owl:Class (daml:Class),
owl:Restriction (daml:Restriction),
owl:onProperty (daml:onProperty),
owl:allValuesFrom (daml:toClass) (only with class identifiers and named datatypes),
owl:someValuesFrom (daml:hasClass) (only with class identifiers and named datatypes),
owl:minCardinality (daml:minCardinality; restricted to {0,1}),
owl:maxCardinality (daml:maxCardinality; restricted to {0,1}),
owl:cardinality (daml:cardinality; restricted to {0,1})
≥ nR
= nR
owl:intersectionOf (only with class identifiers and property restrictions)
C ∩ D


OWL DL
OWL Lite
owl:ObjectProperty (daml:ObjectProperty),
owl:DatatypeProperty (daml:DatatypeProperty),
owl:TransitiveProperty (daml:TransitiveProperty),
owl:SymmetricProperty,
owl:FunctionalProperty (daml:UniqueProperty),
owl:InverseFunctionalProperty (daml:UnambiguousProperty),
owl:AnnotationProperty
owl:Thing (daml:Thing)
owl:Nothing (daml:Nothing)
$^{(+)}R$
$\top$
$\bot$
R^{-1}
$C \equiv D$
$R \equiv S$
owl:inverseOf (daml:inverseOf),
owl:equivalentClass (daml:sameClassAs) (only with class identifiers and property restrictions),
owl:equivalentProperty (daml:samePropertyAs),
owl:sameAs (daml:equivalentTo),
owl:sameIndividualAs,
owl:differentFrom (daml:differentIndividualFrom),
owl:AllDifferent, owl:distinctMembers
Abox
RDF(S)


OWL DL
OWL Lite
Class expressions allowed in: rdfs:domain, rdfs:range, rdfs:subClassOf
owl:intersectionOf, owl:equivalentClass, owl:allValuesFrom, owl:someValuesFrom
Values are not restricted (0..N) in: owl:minCardinality, owl:maxCardinality, owl:cardinality
owl:DataRange, rdf:List, rdf:first, rdf:rest, rdf:nil
owl:hasValue (daml:hasValue)
owl:oneOf (daml:oneOf)
owl:unionOf (daml:unionOf), owl:complementOf (daml:complementOf)
owl:disjointWith (daml:disjointWith)
$\exists R.\{x\}$
$\{x_1, \ldots, x_n\}$
$\neg C$
$C \cup D$
$C \cap D \sqsubseteq \bot$
RDF(S)

OWL 2

- W3C Recommendation (December 2012)



SROIQ – OWL 2 DL

OWL 2 Full

OWL 2 vs OWL 1

- OWL 2 adds, with respect to OWL 1,
  - Syntactic sugar (e.g., disjoint union of classes)
  - Keys
  - Property chains
  - Richer datatypes, data ranges
  - Qualified cardinality restrictions
  - Asymmetric, reflexive, and disjoint properties
  - Enhanced annotation capabilities

- 3 profiles (http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/)
  - EL (ontologies with many classes and properties, inference in polynomial time)
  - QL (complete query answering in LOGSPACE wrt size of data, DL-Lite family)
  - RL (scalable reasoning, implementable with rules)

OWL2 class expressions

Predefined and Named Classes

Language Feature	Functional Syntax	RDF Syntax
named class	CN	CN
universal class	owl:Thing	owl:Thing
empty class	owl:Nothing	owl:Nothing

Boolean Connectives and Enumeration of Individuals

Language Feature	Functional Syntax	RDF Syntax
intersection	ObjectIntersectionOf(C_1 … C_n)	_:x rdf:type owl:Class. _:x owl:intersectionOf (C_1 … C_n).
union	ObjectUnionOf(C_1 … C_n)	_:x rdf:type owl:Class. _:x owl:unionOf (C_1 … C_n).
complement	ObjectComplementOf(C)	_:x rdf:type owl:Class. _:x owl:complementOf C.
enumeration	ObjectOneOf(a_1 … a_n)	_:x rdf:type owl:Class. _:x owl:oneOf (a_1 … a_n).

OWL2 class expressions

Object Property Restrictions

Language Feature	Functional Syntax	RDF Syntax
universal	ObjectAllValuesFrom(P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:allValuesFrom C
existential	ObjectSomeValuesFrom(P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:someValuesFrom C
individual value	ObjectHasValue(P a)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:hasValue a.
local reflexivity	ObjectHasSelf(P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:hasSelf "true"^^xsd:boolean.
exact cardinality	ObjectExactCardinality(n P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:cardinality n.
qualified exact cardinality	ObjectExactCardinality(n P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:qualifiedCardinality n. _:x owl:onClass C.
maximum cardinality	ObjectMaxCardinality(n P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:maxCardinality n.
qualified maximum cardinality	ObjectMaxCardinality(n P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:maxQualifiedCardinality n. _:x owl:onClass C.
minimum cardinality	ObjectMinCardinality(n P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:minCardinality n.
qualified minimum cardinality	ObjectMinCardinality(n P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:minQualifiedCardinality n. _:x owl:onClass C.

Data Property Restrictions

Language Feature	Functional Syntax	RDF Syntax
universal	DataAllValuesFrom(R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:allValuesFrom D.
existential	DataSomeValuesFrom(R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:someValuesFrom D.
literal value	DataHasValue(R v)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:hasValue v.
exact cardinality	DataExactCardinality(n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:cardinality n.
qualified exact cardinality	DataExactCardinality(n R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:qualifiedCardinality n. _:x owl:onDataRange D.
maximum cardinality	DataMaxCardinality(n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:maxCardinality n.
qualified maximum cardinality	DataMaxCardinality(n R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:maxQualifiedCardinality n. _:x owl:onDataRange D.
minimum cardinality	DataMinCardinality(n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:minCardinality n.
qualified minimum cardinality	DataMinCardinality(n R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:minQualifiedCardinality n. _:x owl:onDataRange D.

OWL2 class expressions

Restrictions Using n-ary Data Range

In the following table 'D^n' is an n-ary data range.

Language Feature	Functional Syntax	RDF Syntax
n-ary universal	DataAllValuesFrom(R_1 … R_n D^n)	_:x rdf:type owl:Restriction. _:x owl:onProperties (R_1 … R_n). _:x owl:allValuesFrom D^n.
n-ary existential	DataSomeValuesFrom(R_1 … R_n D^n)	_:x rdf:type owl:Restriction. _:x owl:onProperties (R_1 … R_n). _:x owl:someValuesFrom D^n.

OWL2. Property Expressions

Object Property Expressions

Language Feature	Functional Syntax	RDF Syntax
named object property	PN	PN
universal object property	owl:topObjectProperty	owl:topObjectProperty
empty object property	owl:bottomObjectProperty	owl:bottomObjectProperty
inverse property	ObjectInverseOf(PN)	_:x owl:inverseOf PN

Data Property Expressions

Language Feature	Functional Syntax	RDF Syntax
named data property	R	R
universal data property	owl:topDataProperty	owl:topDataProperty
empty data property	owl:bottomDataProperty	owl:bottomDataProperty

2.3 Individuals & Literals

Language Feature	Functional Syntax	RDF Syntax
named individual	aN	aN
anonymous individual	_:a	_:a
literal (datatype value)	"abc"^^DN	"abc"^^DN

Data Ranges

2.4 Data Ranges

Data Range Expressions

Language Feature	Functional Syntax	RDF Syntax
named datatype	DN	DN
data range complement	DataComplementOf(D)	_:x rdf:type rdfs:Datatype. _:x owl:datatypeComplementOf D.
data range intersection	DataIntersectionOf($D_1 \ldots D_n$)	_:x rdf:type rdfs:Datatype. _:x owl:intersectionOf ($D_1 \ldots D_n$).
data range union	DataUnionOf($D_1 \ldots D_n$)	_:x rdf:type rdfs:Datatype. _:x owl:unionOf ($D_1 \ldots D_n$).
literal enumeration	DataOneOf($v_1 \ldots v_n$)	_:x rdf:type rdfs:Datatype. _:x owl:oneOf ($v_1 \ldots v_n$).
datatype restriction	DatatypeRestriction(DN f_1 v_1 … f_n v_n)	_:x rdf:type rdfs:Datatype. _:x owl:onDatatype DN. _:x owl:withRestrictions (_:x_1 ... _:x_n). _:x_j f_j v_j. $j=1 \ldots n$

OWL2 Axioms

Class Expression Axioms

Language Feature	Functional Syntax	RDF Syntax
subclass	SubClassOf(C_1 C_2)	C_1 rdfs:subClassOf C_2.
equivalent classes	EquivalentClasses(C_1 … C_n)	C_j owl:equivalentClass C_{j+1}. j=1…n-1
disjoint classes	DisjointClasses(C_1 C_2)	C_1 owl:disjointWith C_2.
pairwise disjoint classes	DisjointClasses(C_1 … C_n)	_:x rdf:type owl:AllDisjointClasses. _:x owl:members (C_1 … C_n).
disjoint union	DisjointUnionOf(CN C_1 … C_n)	CN owl:disjointUnionOf (C_1 … C_n).

Object Property Axioms

Language Feature	Functional Syntax	RDF Syntax
subproperty	SubObjectPropertyOf(P_1 P_2)	P_1 rdfs:subPropertyOf P_2.
property chain inclusion	SubObjectPropertyOf(ObjectPropertyChain(P_1 … P_n) P)	P owl:propertyChainAxiom (P_1 … P_n).
property domain	ObjectPropertyDomain(P C)	P rdfs:domain C.
property range	ObjectPropertyRange(P C)	P rdfs:range C.
equivalent properties	EquivalentObjectProperties(P_1 … P_n)	P_j owl:equivalentProperty P_{j+1}. j=1…n-1
disjoint properties	DisjointObjectProperties(P_1 P_2)	P_1 owl:propertyDisjointWith P_2.
pairwise disjoint properties	DisjointObjectProperties(P_1 … P_n)	_:x rdf:type owl:AllDisjointProperties. _:x owl:members (P_1 … P_n).
inverse properties	InverseObjectProperties(P_1 P_2)	P_1 owl:inverseOf P_2.
functional property	FunctionalObjectProperty(P)	P rdf:type owl:FunctionalProperty.
inverse functional property	InverseFunctionalObjectProperty(P)	P rdf:type owl:InverseFunctionalProperty.
reflexive property	ReflexiveObjectProperty(P)	P rdf:type owl:ReflexiveProperty.
irreflexive property	IrreflexiveObjectProperty(P)	P rdf:type owl:IrreflexiveProperty.
symmetric property	SymmetricObjectProperty(P)	P rdf:type owl:SymmetricProperty.
asymmetric property	AsymmetricObjectProperty(P)	P rdf:type owl:AsymmetricProperty.
transitive property	TransitiveObjectProperty(P)	P rdf:type owl:TransitiveProperty.

OWL2 Axioms

Data Property Axioms

Language Feature	Functional Syntax	RDF Syntax
subproperty	SubDataPropertyOf(R_1 R_2)	R_1 rdfs:subPropertyOf R_2.
property domain	DataPropertyDomain(R C)	R rdfs:domain C.
property range	DataPropertyRange(R D)	R rdfs:range D.
equivalent properties	EquivalentDataProperties(R_1 … R_n)	R_j owl:equivalentProperty R_{j+1}. j=1…n-1
disjoint properties	DisjointDataProperties(R_1 R_2)	R_1 owl:propertyDisjointWith R_2.
pairwise disjoint properties	DisjointDataProperties(R_1 … R_n)	_:x rdf:type owl:AllDisjointProperties. _:x owl:members (R_1 … R_n).
functional property	FunctionalDataProperty(R)	R rdf:type owl:FunctionalProperty.

Datatype Definitions

Language Feature	Functional Syntax	RDF Syntax
datatype definition	DatatypeDefinition(DN D)	DN owl:equivalentClass D.

Assertions

Language Feature	Functional Syntax	RDF Syntax
individual equality	SameIndividual(a_1 … a_n)	a_j owl:sameAs a_{j+1}. j=1…n-1
individual inequality	DifferentIndividuals(a_1 a_2)	a_1 owl:differentFrom a_2.
pairwise individual inequality	DifferentIndividuals(a_1 … a_n)	_:x rdf:type owl:AllDifferent. _:x owl:members (a_1 … a_n).
class assertion	ClassAssertion(C a)	a rdf:type C.
positive object property assertion	ObjectPropertyAssertion(PN a_1 a_2)	a_1 PN a_2.
positive data property assertion	DataPropertyAssertion(R a v)	a R v.
negative object property assertion	NegativeObjectPropertyAssertion(P a_1 a_2)	_:x rdf:type owl:NegativePropertyAssertion. _:x owl:sourceIndividual a_1. _:x owl:assertionProperty P. _:x owl:targetIndividual a_2.
negative data property assertion	NegativeDataPropertyAssertion(R a v)	_:x rdf:type owl:NegativePropertyAssertion. _:x owl:sourceIndividual a. _:x owl:assertionProperty R. _:x owl:targetValue v.

Keys

Language Feature	Functional Syntax	RDF Syntax
Key	HasKey(C (P_1 … P_m) (R_1 … R_n))	C owl:hasKey (P_1 … P_m R_1 … R_n). m+n>0

3.1 Built-in Datatypes

Universal Datatype	rdfs:Literal			
Numbers	owl:rational		owl:real	
	xsd:double	xsd:float	xsd:decimal	xsd:integer
	xsd:long	xsd:int	xsd:short	xsd:byte
	xsd:nonNegativeInteger		xsd:nonPositiveInteger	
	xsd:positiveInteger		xsd:negativeInteger	
	xsd:unsignedLong		xsd:unsignedInt	
	xsd:unsignedShort		xsd:unsignedByte	
Strings	rdf:PlainLiteral (RDF plain literals)			
	xsd:string	xsd:NCName	xsd:Name	xsd:NMTOKEN
	xsd:token	xsd:language	xsd:normalizedString	
Boolean Values	xsd:boolean (value space: *true* and *false*)			
Binary Data	xsd:base64Binary		xsd:hexBinary	
IRIs	xsd:anyURI			
Time Instants	xsd:dateTime (optional time zone offset)			
	xsd:dateTimeStamp (required time zone offset)			
XML Literals	rdf:XMLLiteral			

3.2 Facets

Facet	Value	Applicable Datatypes	Explanation
xsd:minInclusive xsd:maxInclusive xsd:minExclusive xsd:maxExclusive	literal in the corresponding datatype	Numbers, Time Instants	Restricts the value-space to greater than (equal to) or lesser than (equal to) a value
xsd:minLength xsd:maxLength	Non-negative integer	Strings, Binary Data,	Restricts the value-space based on the lengths of the literals
xsd:pattern	xsd:string literal as a regular expression	Strings, IRIs	Restricts the value space to literals that> match the regular expression
rdf:langRange	xsd:string literal as a regular expression	rdf:PlainLiteral	Restricts the value space to literals with language tags that match the reg

ular expression