Knowledge Representation for the Semantic Web
Lecture 4: Description Logics III

Daria Stepanova

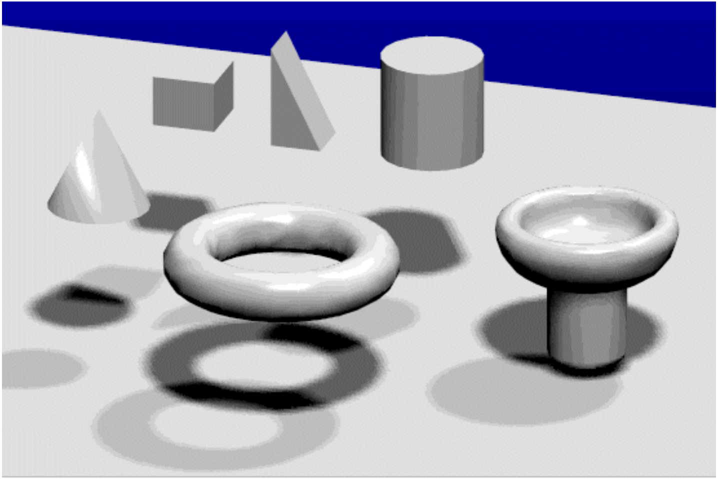
slides based on Reasoning Web 2011 tutorial "*Foundations of Description Logics and OWL*" by S. Rudolph

Max Planck Institute for Informatics
D5: Databases and Information Systems group

WS 2017/18

Unit Outline

Modeling



Modeling with DLs: Motivating Examples

- individual *angelina* belongs to the set of all actors:

Modeling with DLs: Motivating Examples

- individual *angelina* belongs to the set of all actors:

$$Actor(angelina)$$

Modeling with DLs: Motivating Examples

- individual *angelina* belongs to the set of all actors:

$$Actor(angelina)$$

- individuals *angelina* and *brad* are in the relation of being married:

Modeling with DLs: Motivating Examples

- individual *angelina* belongs to the set of all actors:

$$Actor(angelina)$$

- individuals *angelina* and *brad* are in the relation of being married:

$$married(angelina, brad)$$

Modeling with DLs: Motivating Examples

- individual *angelina* belongs to the set of all actors:

$$Actor(angelina)$$

- individuals *angelina* and *brad* are in the relation of being married:

$$married(angelina, brad)$$

- every actor is an artist:

Modeling with DLs: Motivating Examples

- individual *angelina* belongs to the set of all actors:

$$Actor(angelina)$$

- individuals *angelina* and *brad* are in the relation of being married:

$$married(angelina, brad)$$

- every actor is an artist:

$$Actor \sqsubseteq Artist$$

$$\forall x.Actor(x) \rightarrow Artist(x)$$

Modeling with DLs: Motivating Examples

- individual *angelina* belongs to the set of all actors:

$$Actor(angelina)$$

- individuals *angelina* and *brad* are in the relation of being married:

$$married(angelina, brad)$$

- every actor is an artist:

$$Actor \sqsubseteq Artist$$

$$\forall x.Actor(x) \to Artist(x)$$

- every actor who is a US governor is also a bodybuilder or not Austrian:

Modeling with DLs: Motivating Examples

- individual *angelina* belongs to the set of all actors:

$$Actor(angelina)$$

- individuals *angelina* and *brad* are in the relation of being married:

$$married(angelina, brad)$$

- every actor is an artist:

$$Actor \sqsubseteq Artist$$

$$\forall x. Actor(x) \to Artist(x)$$

- every actor who is a US governor is also a bodybuilder or not Austrian:

$$Actor \sqcap USGovernor \sqsubseteq Bodybuilder \sqcup \neg Austrian$$

$$\forall x. (Actor(x) \land USGovernor(x)) \to (BodyBuilder(x) \lor \neg Austrian(x))$$

Modeling with DLs: Motivating Examples, cont'd.

- everybody knowing some actor has only envious friends:

Modeling with DLs: Motivating Examples, cont'd.

- everybody knowing some actor has only envious friends:

$$\exists knows.Actor \sqsubseteq \forall hasfriend.Envious$$

$$\forall x\,(\exists y\,(knows(x,y) \wedge Actor(y)) \rightarrow \forall z\,(hasfriend(x,z) \rightarrow Envious(z)))$$

Modeling with DLs: Motivating Examples, cont'd.

- everybody knowing some actor has only envious friends:

$$\exists knows.Actor \sqsubseteq \forall hasfriend.Envious$$

$$\forall x\,(\exists y\,(knows(x,y) \land Actor(y)) \rightarrow \forall z\,(hasfriend(x,z) \rightarrow Envious(z)))$$

- everybody having a child is the child of only grandparents:

Modeling with DLs: Motivating Examples, cont'd.

- everybody knowing some actor has only envious friends:

$$\exists knows.Actor \sqsubseteq \forall hasfriend.Envious$$

$$\forall x\,(\exists y\,(knows(x,y) \land Actor(y)) \rightarrow \\ \forall z\,(hasfriend(x,z) \rightarrow Envious(z)))$$

- everybody having a child is the child of only grandparents:

$$\exists hasChild.\top \sqsubseteq \forall hasChild^{-}.Grandparent$$

$$\forall x\,(\exists y\,(hasChild(x,y)) \rightarrow \\ \forall z\,(hasChild(z,x) \rightarrow Grandparent(x)))$$

Modeling with DLs: Motivating Examples, cont'd.

- everybody knowing some actor has only envious friends:

$$\exists knows.Actor \sqsubseteq \forall hasfriend.Envious$$

$$\forall x\,(\exists y\,(knows(x,y) \land Actor(y)) \rightarrow \\ \forall z\,(hasfriend(x,z) \rightarrow Envious(z)))$$

- everybody having a child is the child of only grandparents:

$$\exists hasChild.\top \sqsubseteq \forall hasChild^{-}.Grandparent$$

$$\forall x\,(\exists y\,(hasChild(x,y)) \rightarrow \\ \forall z\,(hasChild(z,x) \rightarrow Grandparent(x)))$$

- a polygamist is married to at least two distinct individuals:

Modeling with DLs: Motivating Examples, cont'd.

- everybody knowing some actor has only envious friends:

$$\exists knows.Actor \sqsubseteq \forall hasfriend.Envious$$

$$\forall x\,(\exists y\,(knows(x,y) \land Actor(y)) \rightarrow \forall z\,(hasfriend(x,z) \rightarrow Envious(z)))$$

- everybody having a child is the child of only grandparents:

$$\exists hasChild.\top \sqsubseteq \forall hasChild^{-}.Grandparent$$

$$\forall x\,(\exists y\,(hasChild(x,y)) \rightarrow \forall z\,(hasChild(z,x) \rightarrow Grandparent(x)))$$

- a polygamist is married to at least two distinct individuals:

$$Polygamist \sqsubseteq \geq 2\ marriedTo.\top$$

$$\forall x\,(Polygamist(x) \rightarrow \exists y \exists z\,(marriedTo(x,y) \land marriedTo(x,z) \land y \neq z))$$

Modeling with DLs: Motivating Examples, cont'd.

- being married to Brad is a property only applying to Angelina:

Modeling with DLs: Motivating Examples, cont'd.

- being married to Brad is a property only applying to Angelina:

$$\exists marriedTo.\{brad\} \sqsubseteq \{angelina\}$$

$$\exists x\,(marriedTo(x, brad) \rightarrow x = angelina)$$

Modeling with DLs: Motivating Examples, cont'd.

- being married to Brad is a property only applying to Angelina:

$$\exists marriedTo.\{brad\} \sqsubseteq \{angelina\}$$

$$\exists x\,(marriedTo(x, brad) \rightarrow x = angelina)$$

- being married to somebody implies loving them:

Modeling with DLs: Motivating Examples, cont'd.

- being married to Brad is a property only applying to Angelina:

$$\exists marriedTo.\{brad\} \sqsubseteq \{angelina\}$$

$$\exists x\,(marriedTo(x, brad) \rightarrow x = angelina)$$

- being married to somebody implies loving them:

$$marriedTo \sqsubseteq loves$$

$$\forall x \forall y\ married(x, y) \rightarrow loves(x, y)$$

Modeling with DLs: Motivating Examples, cont'd.

- being married to Brad is a property only applying to Angelina:

$$\exists marriedTo.\{brad\} \sqsubseteq \{angelina\}$$

$$\exists x\,(marriedTo(x, brad) \rightarrow x = angelina)$$

- being married to somebody implies loving them:

$$marriedTo \sqsubseteq loves$$

$$\forall x \forall y\; married(x, y) \rightarrow loves(x, y)$$

- the child of somebody I am a child of is my sibling:

Modeling with DLs: Motivating Examples, cont'd.

- being married to Brad is a property only applying to Angelina:

$$\exists marriedTo.\{brad\} \sqsubseteq \{angelina\}$$

$$\exists x\,(marriedTo(x, brad) \rightarrow x = angelina)$$

- being married to somebody implies loving them:

$$marriedTo \sqsubseteq loves$$

$$\forall x \forall y\; married(x, y) \rightarrow loves(x, y)$$

- the child of somebody I am a child of is my sibling:

$$hasChild^{-} \circ hasChild \sqsubseteq hasSibling$$

$$\forall x \forall y \forall z\,(hasChild(y, x) \wedge hasChild(y, z) \rightarrow hasSibling(x, z))$$

Frequent Modeling Features

- domain $\exists authorOf.\top \sqsubseteq Person$
- range $\top \sqsubseteq \forall authorOf.Publication$
  or $\exists authorOf^{-}.\top \sqsubseteq Publication$
- concept disjointness $Male \sqcap Female \sqsubseteq \bot$
  or $Male \sqsubseteq \neg Female$
- role symmetry $marriedWith \sqsubseteq marriedWith^{-}$
- role transitivity $partOf \circ partOf \sqsubseteq partOf$

Number Restrictions

- allow for defining that a role is functional

$$\top \sqsubseteq \leq 1\ hasFather.\top$$

- ...or inverse functional

$$\top \sqsubseteq \leq 1\ hasFather^{-}.\top$$

- allow for enforcing an infinite domain

$$(\forall succ^{-}.\bot)(zero) \quad \top \sqsubseteq\ \exists succ.\top \quad \top \sqsubseteq \leq 1.succ^{-}.\top$$

- Consequently, DLs with number restrictions and inverses *do not have the finite model property*.

Nominal Concept and Universal Role

- allow to restrict the size of concepts

$$AtMostTwo \sqsubseteq \{one, two\} \qquad AtMostTwo \sqsubseteq \leq 2u.\top$$

- even allow to restrict the size of the domain

$$\top \sqsubseteq \{one, two\} \qquad \top \sqsubseteq \leq 2u.\top$$

Self-Restriction

- allows to define a role as reflexive

$$\top \sqsubseteq \exists knows.Self$$

- allows to define a role as irreflexive

$$\exists betterThan.Self \sqsubseteq \bot$$

- together with number restrictions, we can even axiomatize equality

$$\top \sqsubseteq \exists equals.Self \qquad \top \sqsubseteq \leq 1 equals.\top$$

Axioms vs. Constraints

Note: GCIs may not serve as constraints that eliminate models.

- every employee must have a social security number (SSN)
  - informal constraint: "if employee x has no social security number, then infer contradiction (falsity)"
  - transcribed into a DL axiom: $\underbrace{Employee \sqcap \neg\exists hasSSN \sqsubseteq \bot}_{\alpha}$
- let $KB = \{Employee(Joe),\ \alpha\}$
  - this knowledge base is consistent!
  - it assigns informally a *null*-value to *Joe*'s SSN
    - α is logically equivalent to $Employee \sqsubseteq \exists hasSSN$
- no possibility to express (independent of concrete data) that every employee has a *known* SSN

Axioms vs. Constraints, cont'd

Uniqueness constraints might not eliminate models either

- define that $hasSSN$ is inverse functional, and add some data:
- for

$$KB = \left\{ \begin{array}{l} \top \sqsubseteq \ \leq 1 \ hasSSN^{-}.\top, \\ hasSSN(Joe, 4711), \\ hasSSN(Jeff, 4711) \end{array} \right\}$$

  we can conclude that $KB \models Joe \approx Jeff$

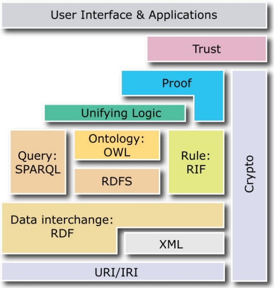
- there is no Unique Name Assumption (UNA) by default

Reminer: UNA

If c_1 and c_2 are two individuals such that $c_1 \neq c_2$, then $c_1^{\mathcal{I}} \neq c_2^{\mathcal{I}}$

Description Logics and OWL

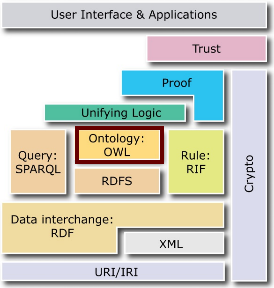
Semantic Web



http://www.w3.org/2007/03/layerCake.png

- Resource Description Framework (RDF): triple statements (S, P, O)
- RDF Schema (RDFS): simple classes/property taxonomies
- Web Ontology Language (OWL): more constructs, based on DLs
- SPARQL is a RDF query language (also used for RDFS and OWL)
- RIF is a rule interchange format

Semantic Web



`http://www.w3.org/2007/03/layerCake.png`

- Resource Description Framework (RDF): triple statements (S, P, O)
- RDF Schema (RDFS): simple classes/property taxonomies
- Web Ontology Language (OWL): more constructs, based on DLs
- SPARQL is a RDF query language (also used for RDFS and OWL)
- RIF is a rule interchange format

OWL Ontologies

- **2004:** Web Ontology Language (OWL): W3C standard
- Knowledge about concepts, individuals, their properties and relationships

OWL Ontologies

- **2004:** Web Ontology Language (OWL): W3C standard
- Knowledge about concepts, individuals, their properties and relationships
- Three increasingly expressive sublanguages

  - OWL Lite: concept hierarchies, simple constraint features ($\rightleftharpoons \mathcal{SHIF}$ with datatypes)
  - OWL DL : nominals, number restriction ($\rightleftharpoons \mathcal{SHOIN}$ with datatypes[a])
  - OWL Full: allow, e.g. to treat classes as individuals

[a]Datatype can be seen as a unary predicate with a built-in interpretation (e.g., the `xsd:integer` datatype is interpreted as the set of all integer values)

OWL Ontologies

- **2004:** Web Ontology Language (OWL): W3C standard
- Knowledge about concepts, individuals, their properties and relationships
- Three increasingly expressive sublanguages

  - OWL Lite: concept hierarchies, simple constraint features (≂ $\mathcal{SHIF}$ with datatypes)
  - OWL DL : nominals, number restriction (≂ $\mathcal{SHOIN}$ with datatypes[a])
  - OWL Full: allow, e.g. to treat classes as individuals

[a] Datatype can be seen as a unary predicate with a built-in interpretation (e.g., the `xsd:integer` datatype is interpreted as the set of all integer values)

- **2009:** OWL2: redefines OWL DL and adds profiles EL, QL, RL
- OWL syntax is based on RDF, common: RDF/XML, turtle syntax (in this lecture turtle is used just for demonstration)
  `http://www.w3.org/TR/turtle/`

How Do OWL2 and DLs Relate?

- OWL2 DL is essentially
  - $\mathcal{SROIQ}$ in disguise
  - plus extended datatype support
  - plus extralogical features such as annotations, versioning, etc.

How Do OWL2 and DLs Relate?

- OWL2 DL is essentially
  - $\mathcal{SROIQ}$ in disguise
  - plus extended datatype support
  - plus extralogical features such as annotations, versioning, etc.

- tractable OWL2 profiles EL, QL and RL correspond roughly to description logics $\mathcal{EL}^{++}$, DL-Lite, and description logic programs

How Do OWL2 and DLs Relate?

- OWL2 DL is essentially
  - $\mathcal{SROIQ}$ in disguise
  - plus extended datatype support
  - plus extralogical features such as annotations, versioning, etc.

- tractable OWL2 profiles EL, QL and RL correspond roughly to description logics $\mathcal{EL}^{++}$, DL-Lite, and description logic programs
- OWL and DL terminologies slightly differ for historical reasons

OWL	DL	FOL
class name	concept name	unary predicate
class	concept	formula with one free variable
object property name	role name	binary predicate
object property	role	formula with two free variables
ontology	knowledge base	theory
axiom	axiom	sentence
vocabulary	vocabulary/signature	signature

Translating DL into OWL

- Next to the logic part, an OWL ontology features a preamble and a declaration part (turtle syntax):

$$[\![\mathcal{KB}]\!] = \mathsf{Pre} + \mathsf{Dec}(\mathcal{KB}) + \Sigma_{\alpha \in \mathcal{KB}} [\![\alpha]\!]$$

$$\mathsf{Pre} = \begin{cases} \texttt{@prefix owl: <http://www.w3.org/2002/07/owl\#>} \\ \texttt{@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema\#>} \\ \texttt{@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>} \\ \texttt{@prefix xsd: <http://www.w3.org/2001/XMLSchema\#>} \end{cases}$$

$$\begin{aligned} \mathsf{Dec}(\mathcal{KB}) &= \Sigma_{A \in \mathsf{N}_C(\mathcal{KB})}\, A \ \texttt{rdf:type owl:Class} \\ &+ \Sigma_{r \in \mathsf{N}_R(\mathcal{KB})}\, r \ \texttt{rdf:type owl:ObjectProperty} \end{aligned}$$

Translating DL Axioms into OWL

- Following the Semantic Web rationale, OWL axioms are expressed using RDF, i.e. as triples. As far as possible, RDFS vocabulary is reused

$$
\begin{aligned}
r_1 \circ \ldots \circ r_n \sqsubseteq r \quad &\doteq \quad r\ \texttt{owl:propertyChainAxiom}\ (r_1, \ldots, r_n)\ . \\
\mathsf{Dis}(r, r') \quad &\doteq \quad r\ \texttt{owl:propertyDisjointWith}\ r'\ . \\
C \sqsubseteq D \quad &\doteq \quad C\ \texttt{owl:subClassOf}\ D\ . \\
C(a) \quad &\doteq \quad a\ \texttt{rdf:type}\ C\ . \\
r(a, b) \quad &\doteq \quad a\ r\ b\ . \\
r^-(a, b) \quad &\doteq \quad b\ r\ a\ . \\
\neg r(a, b) \quad &\doteq \quad [\,]\ \texttt{rdf:type owl:NegativePropertyAssertion}\ ; \\
&\qquad\ \texttt{owl:assertionProperty}\ r\ ; \\
&\qquad\ \texttt{owl:sourceIndividual}\ a\ ;\ \texttt{owl:targetValue}\ b\ . \\
a \approx b \quad &\doteq \quad a\ \texttt{owl:sameAs}\ b\ . \\
a \not\approx b \quad &\doteq \quad a\ \texttt{owl:differentFrom}\ b\ .
\end{aligned}
$$

Translating DL Axioms into OWL, cont'd.

$$
\begin{array}{rcl}
u & \doteq & \texttt{owl:topObjectProperty} \ . \\
r & \doteq & r \, . \\
r^{-} & \doteq & [\ \texttt{owl:inverseOf} \ \ {:}\, r \ ] \, . \\
A & \doteq & A \, . \\
\top & \doteq & \texttt{owl:Thing} \ . \\
\bot & \doteq & \texttt{owl:Nothing} \ . \\
\{a_1, \ldots, a_n\} & \doteq & [\ \texttt{rdf:type owl:Class ; owl:oneOf} \ ({:}\, a_1 \ldots {:}\, a_n) \ ] \, . \\
\neg C & \doteq & [\ \texttt{rdf:type owl:Class ; owl:complementOf} \ C \ ] \, . \\
C_1 \sqcap \ldots \sqcap C_n & \doteq & [\ \texttt{rdf:type owl:Class ; owl:intersectionOf} \ (C_1 \ldots C_n) \ ] \, . \\
C_1 \sqcup \ldots \sqcup C_n & \doteq & [\ \texttt{rdf:type owl:Class ; owl:unionOf} \ (C_1 \ldots C_n) \ ] \, .
\end{array}
$$

Translating DL Axioms into OWL ctd.

$\exists r.C \doteq$ [rdf:type owl:Restriction ;
owl:onProperty r ; owl:someValuesFrom C] .

$\forall r.C \doteq$ [rdf:type owl:Restriction ;
owl:onProperty r ; owl:allValuesFrom C] .

$\exists r.\mathsf{Self} \doteq$ [rdf:type owl:Restriction ;
owl:onProperty r ; owl:hasSelf ''true'' xsd:boolean] .

$\geq rn.C \doteq$ [rdf:type owl:Restriction ;
owl:minQualifiedCardinality n xsd:nonNegativeInteger;
owl:onProperty r ; owl:onClass C] .

$\leq rn.C \doteq$ [rdf:type owl:Restriction ;
owl:maxQualifiedCardinality n xsd:nonNegativeInteger;
owl:onProperty r ; owl:onClass C] .

Behind the Scenes: Cat Example

RBox $\mathcal{R}$

$owns \sqsubseteq caresFor$

"If somebody owns something, s/he cares for it."

TBox $\mathcal{T}$

$Healthy \sqsubseteq \neg Dead$

"Healthy beings are not dead."

$Cat \sqsubseteq Dead \sqcup Alive$

"Every cat is dead or alive."

$HappyCatOwner \sqsubseteq \exists owns.Cat \sqcap \forall caresFor.Healthy$

"A happy cat owner owns a cat and all beings he cares for are healthy."

ABox $\mathcal{A}$

$HappyCatOwner(schroedinger)$

"Schrödinger is a happy cat owner."

Behind the Scenes ctd.

```
:owns rdfs:SubPropertyOf :caresFor .

:Healthy  rdfs:subClassOf [ owl:complementOf :Dead ].

:Cat rdfs:subClassOf [ owl:unionOf (:Dead :Alive) ] .

:HappyCatOwner rdfs:subClassOf
                 [ owl:intersectionOf
                 ( [ rdf:type owl:restriction ;
                     owl:onProperty :owns ; owl:someValuesFrom :Cat ]
                   [ rdf:type owl:Restriction ;
                     owl:onProperty :caresFor ; owl:allValuesFrom :Healthy ] )
                 ] .
:schroedinger rdf:type :HappyCatOwner .
```

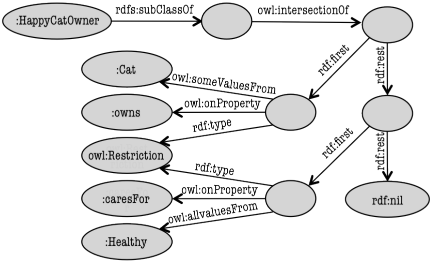
Behind the Scenes ctd.

```
:owns rdfs:SubPropertyOf :caresFor .

:Healthy  rdfs:subClassOf [ owl:complementOf :Dead ].

:Cat rdfs:subClassOf [ owl:unionOf (:Dead :Alive) ] .

:HappyCatOwner rdfs:subClassOf
                 [ owl:intersectionOf
                 ( [ rdf:type owl:restriction ;
                     owl:onProperty :owns ; owl:someValuesFrom :Cat ]
                   [ rdf:type owl:Restriction ;
                     owl:onProperty :caresFor ; owl:allValuesFrom :Healthy ] )
                 ] .
:schroedinger rdf:type :HappyCatOwner .
```



Paraphrasing OWL Axioms in DL

Axiom type	Turtle notation	DL paraphrase
Class Equivalence	C owl:equivalentClass D	$C \sqsubseteq D, D \sqsubseteq C$
Class Disjointness	C owl:disjointWith D	$C \sqcap D \sqsubseteq \bot$
Disjoint Classes	[] rdf:type owl:AllDisjointClasses ; owl:members $(C_1 \ldots C_n)$.	$C_i \sqcap C_j \sqsubseteq \bot$, for all $1 \leq i < j \leq n$
Disjoint Union	C owl:disjointUnionOf $(C_1 \ldots C_n)$.	$\bigsqcup_{i<j} C_i \sqsubseteq C$ $C_i \sqcap C_j \sqsubseteq \bot$, for all $1 \leq i < j \leq n$
Property Equivalence	r owl:equivalentProperty s .	$r \sqsubseteq s$, $s \sqsubseteq r$
Disjoint Properties	[] rdf:type owl:AllDisjointProperties . owl:members $(r_1 \ldots r_n)$.	Dis(r_i, r_j), for all $1 \leq i < j \leq n$
Inverse Properties	r owl:inverseOf s .	$Inv(r) \sqsubseteq s$
Property Domain	r rdfs:domain C .	$\exists r.\top \sqsubseteq C$
Property Range	r rdfs:range C .	$\top \sqsubseteq \forall r.C$

Paraphrasing OWL Axioms in DL ctd.

Axiom type	Turtle notation	DL paraphrase
Functional Property	r rdf:type owl:FunctionalProperty .	$\top \sqsubseteq 1r.\top$
Inverse Functional Property	r rdf:type owl:InverseFunctionalProperty .	$\top \sqsubseteq 1Inv(r).\top$
Reflexive Property	r rdf:type owl:ReflexiveProperty .	$\top \sqsubseteq \exists r.\mathsf{Self}$
Irreflexive Property	r rdf:type owl:IrreflexiveProperty .	$\exists r.\mathsf{Self} \sqsubseteq \bot$
Symmetric Property	r rdf:type owl:SymmetricProperty .	$Inv(r) \sqsubseteq r$
Asymmetric Property	r rdf:type owl:AsymmetricProperty .	$\mathsf{Dis}(Inv(r), r)$
Transitive Property	r rdf:type owl:TransitiveProperty .	$r \circ r \sqsubseteq r$
Different Individuals	[] rdf:type owl:AllDifferent ; owl:members$(a_1 \ldots a_n)$.	$a_i \not\approx a_j$, for all $1 \leq i < j \leq n$

OWL Profiles

- OWL2 is highly intractable in general!
  (standard reasoning is 2NEXPTIME-complete).
- Design principle for profiles: Identify maximal OWL sublanguages that are still implementable in PTime.
- Main source of intractability: non-determinism
  (reasoning requires guessing/backtracking)
  - `owl:unionOf`, or `owl:complementOf` and `owl:intersectionOf`
  - Max. cardinality restrictions
  - Combining existentials (`owl:someValuesFrom`) and universals (`owl:allValuesFrom`) in superclasses
  - Non-unary finite class expressions (`owl:oneOf`) or datatype expressions

→ features that are not allowed in any OWL profile.
Many further features can lead to non-determinism - care needed!

OWL2 EL

- OWL profile based on description logic $\mathcal{EL}^{++}$
- Intuition: focus on terminological expressivity used for light-weight ontologies
- Allow `owl:someValuesFrom` (existential) but not `owl:allValuesFrom` (universal)
- Property domains, class/property hierarchies, class intersections, disjoint classes/properties, property chains, `owl:hasSelf`, `owl:hasValue`, and keys fully supported
- No inverse or symmetric properties
- `rdfs:range` allowed but with some restrictions
- No `owl:unionOf` or `owl:complementOf`
- Various restrictions on available datatypes

OWL2 QL

- OWL profile that can be used to query data-rich applications
  - often used for OBDA (ontology based data access)
- Intuition: use OWL concepts as light-weight queries, allow query answering using rewriting in SQL on top of relational DBs
- Different restrictions on subclasses and superclasses of `rdfs:SubclassOf`
  - subclasses can only be class names or `owl:someValuesFrom` (existential) with unrestricted (`owl:Thing`) filler
  - superclasses can be class names, `owl:someValuesFrom` or `owl:intersectionOf` with superclass filler (recursive), or `owl:complementOf` with subclass filler

OWL2 QL ctd.

- Property hierarchies, disjointness, inverses, (a)symmetry supported, restrictions on range and domain
- Disjoint or equivalence of classes only for subclass-type expressions
- No `owl:unionOf`, `owl:allValuesFrom`, `owl:hasSelf`, `owl:hasKey`, `owl:hasValue`, `owl:oneOf`, `owl:sameAs`, `owl:propertyChainAxiom`, `owl:TransitiveProperty`, cardinalities, functional properties
- Some restrictions on available datatypes

OWL2 RL

- OWL profile that resembles an OWL-based rule language
- Intuition: subclass axioms in OWL RL can be understood as rule-like implications with head (superclass) and body (subclass)
- Different restrictions on subclasses and superclasses of `rdfs:SubclassOf`
  - subclasses can only be class names, `owl:oneOf`, `owl:hasValue`, `owl:intersectionOf`, `owl:unionOf`, `owl:someValuesFrom` if applied only to subclass-type expressions
  - superclasses can be class names, `owl:allValuesFrom` or `owl:hasValue`; also max. cardinalities of 0 or 1 are allowed, all with superclass-type filler expressions only

OWL2 RL cdt.

- Property domains and ranges only for subclass-type expressions
- Full support of property hierarchies, disjointness, inverses, (a)symmetry, transitivity, chains, (inverse)functionality, irreflexivity
- Disjoint classes and classes in keys need subclass-type expressions
- Equivalence only for expressions that are sub- and superclass-type, no restrictions on `owl:sameAs`
- Some restrictions on available datatypes

Important feature: as in relational databases, only "named" individuals matter:

$$Person \sqsubseteq \exists father \qquad Person(joe).$$

intuitively, father of Joe is "unnamed"; axiom is not allowed in RL.

Do We Really Need So Many OWLs?

Three new OWL profiles with somewhat complex descriptions...
Why not just one?

- The union of any two of the profiles is no longer light-weight!

  each of EL + RL, QL + EL, RL + EL is ExpTime-hard

- Restricting to fewer profiles: give up potentially useful feature combinations
- **Rationale:**
  - profiles are "maximal" (well, not quite) well-behaved fragments of OWL 2
  - pick suitable feature set for applications
- In particular, nobody is forced to implement *all* of a profile

OWL in Practice: Tools

- Most common editor: Protégé 4
- Other tools: TopBraid Composer, NeOn toolkit, etc.
- Special purpose apps, esp. for light-weight ontologies (e.g. FOAF editors)
- Reasoners
  - OWL DL: Pellet, HermiT, FaCT++, RacerPro
  - OWL2 EL: CEL, SHER, snorocket, ELK
  - OWL2 RL: OWLIM, Jena, Oracle Prime (part of O 11g)
  - OWL2 QL: Owlgres, QuOnto, Quill
- Many tools use the OWL API library (Java)
- Note: many other Semantic Web tools are found online

http://www.w3.org/2001/sw/wiki/OWL/Implementations

http://semanticweb.org/wiki/Tools.html

Summary

1. Modeling
   - Frequent modeling features
   - Number restrictions
   - Nominal concepts
   - Self restruction
2. Description Logics and OWL
   - Relation between DLs and OWL
   - Translating DLs into OWL
   - OWL profiles
   - Tools

References I

Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors.
The Description Logic Handbook: Theory, Implementation and Applications.
Cambridge University Press, 2007.

Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph.
Foundations of Semantic Web Technologies.
Chapman and Hall, 2010.

Sebastian Rudolph.
Foundations of description logics.
In Axel Polleres, Claudia d'Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, and Peter Patel-Schneider, editors, *Reasoning Web. Semantic Technologies for the Web of Data*, volume 6848 of *Lecture Notes in Computer Science*, pages 76–136. Springer Berlin / Heidelberg, 2011.