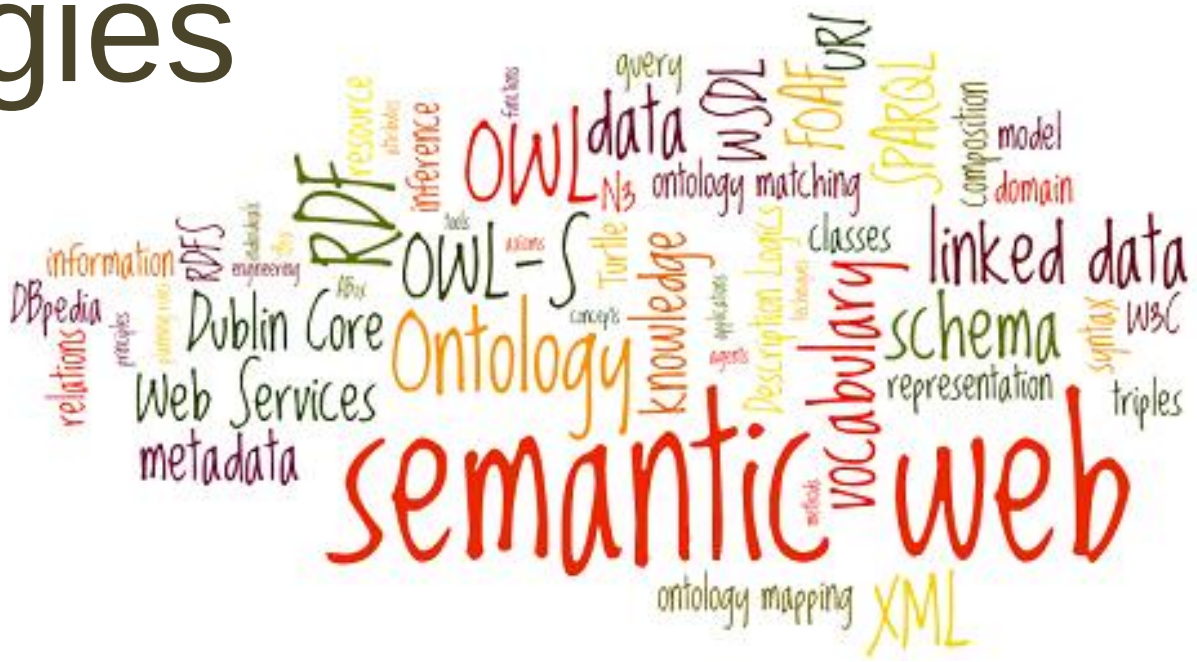
Ontologies

OWL2



POLITECNICO
DI TORINO

e-Lite

Outline

- Automated Reasoning
- OWL Semantics and Profiles
- Reasoning with Description Logics
- SWRL

Reasoning

```
@prefix ex: <http://example.org/>.

ex:Mammal rdf:type owl:Class.

# Canine is a subclass of Mammal
ex:Canine rdf:type owl:Class;
          rdfs:subClassOf ex:Mammal.

# Daisy is implicitly a member of the class Mammal
ex:Daisy rdf:type ex:Canine.
```

- Daisy is a Canine
  - Explicit fact
- Daisy is a Mammal
  - Implicit (implied) fact
- How to derive implied information?

Reasoners

- Applications that perform inference are called **reasoning engines**, or **reasoners**.
- A reasoning engine is a system that **infers new information** based on the contents of a knowledgebase
- Various reasoning **approaches**: rules and rule engine, triggers on database or RDF store, decision trees, tableau algorithms, hard-coded logic, …

Rules-based reasoning

- Combine the assertions contained in a knowledgebase with a set of logical rules in order to derive assertions or perform actions
- Rules are if-then statements:
  - Condition
  - Conclusion
- *Any time a set of statements **matches the conditions** of the rule, the statements in the **conclusion are implicit** in the knowledgebase.*

Example

```
[IF]
  ?class1 rfds:subClassOf ?class2
  AND
  ?instance rdf:type ?class1
[THEN]
  ?instance rdf:type ?class2

[IF]
  ?class2 rfds:subClassOf ?class1
  AND
  ?class3 rfds:subClassOf ?class2
[THEN]
  ?class3 rdf:type ?class1
```

Note: rules systems are different

- Different rules-based languaged offer different expressive power:
  - Conjunctive rules: A and B imply C
  - Disjuntive rules: A or B imply C
  - Negation as a failure: not A implies B

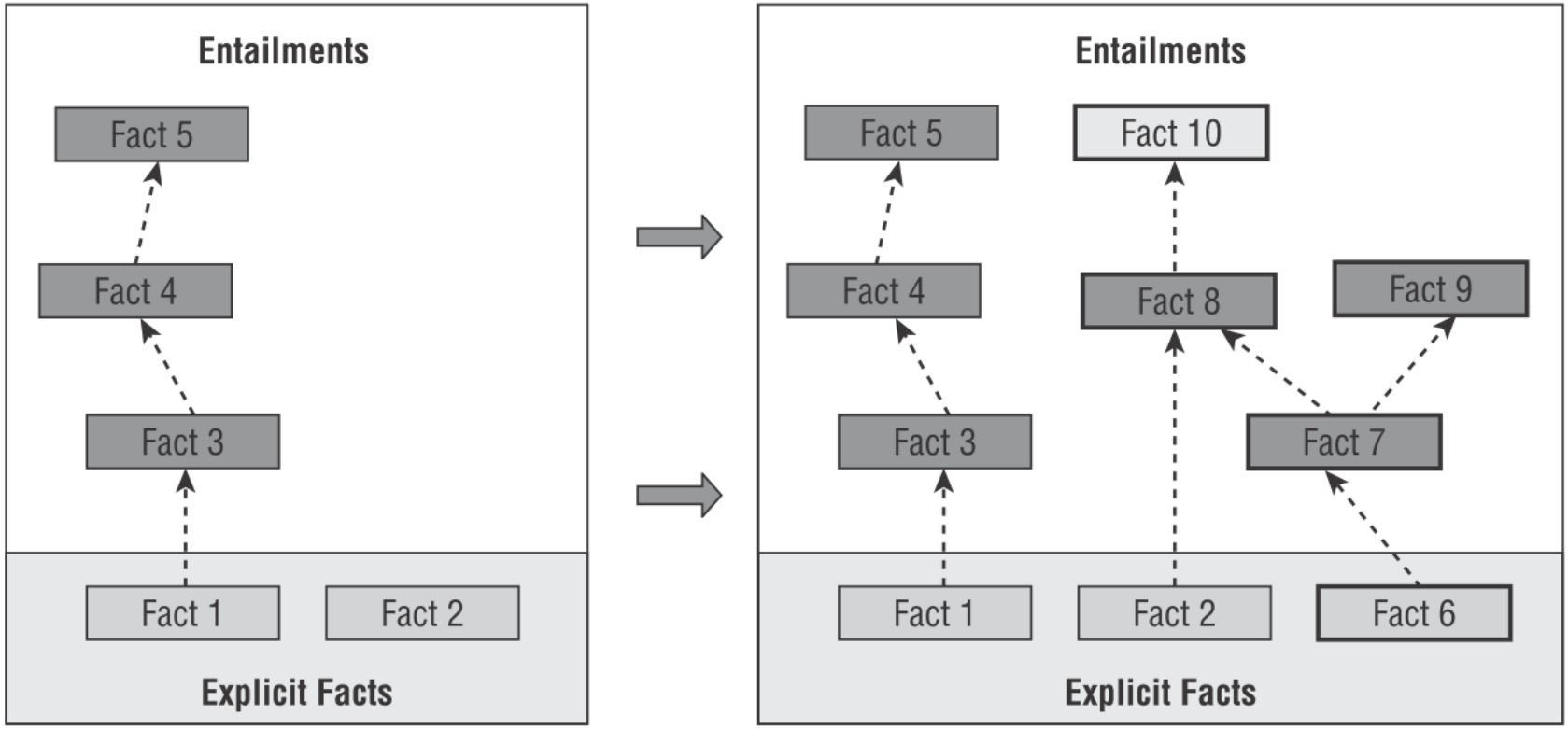
Note: Rule sets are different

- **Predefined** sets of rules (e.g. OWL semantics)
- **Custom** sets of rules (e.g. your own application)

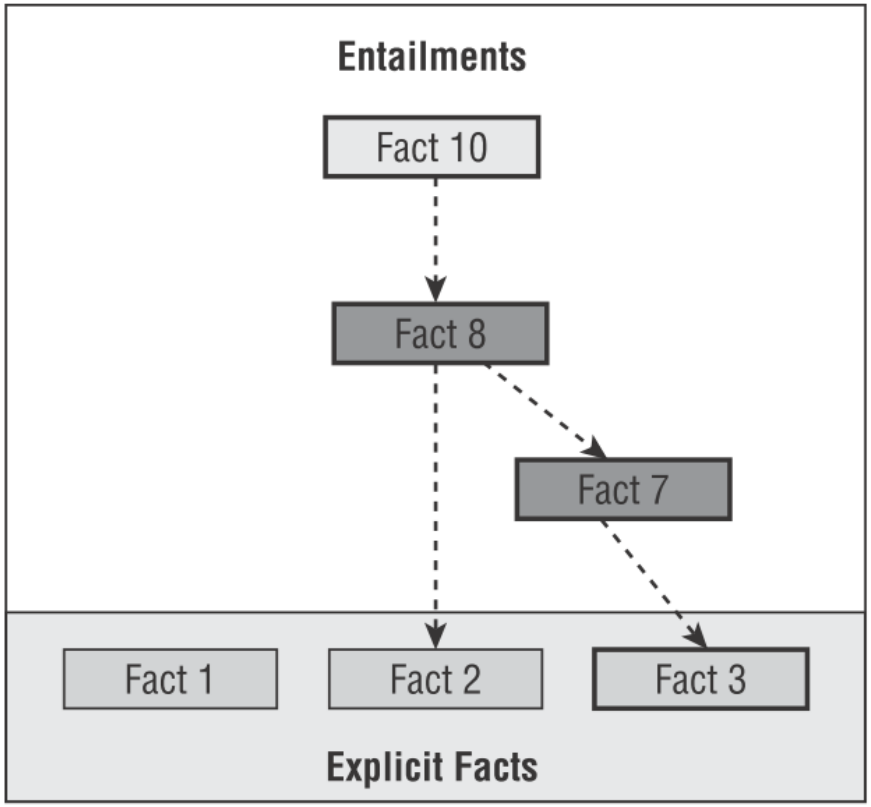
Inference

- Inference = Applying the set of rules to the knowledge base
- Problem: the huge space of all possible applicable rules
- Two main approaches:
  - **Forward Chaining** Inference: Compute all the facts that are entailed by the currently asserted facts
  - **Backward Chaining** Inference: Starting from an unknown fact that we want to know (whether it's true or not), try to construct a chain of entailments rooting back in the known facts

Forward Chaining



Backward chaining



Comparison

Forward Chaining

- After reasoning, all queries are straighforward
- Much memory may be needed for inferred model
- May be computationally intensive at startup
- Difficult to update when facts are removed/modified

Backward Chaining

- Does not compute whole model
- Usually faster
- Each query needs to re-compute part of the model (caching is essential)
- No start-up overhead
- Lower memory requirements
- Efficiency depending on exploration strategies/heuristics

Outline

- Automated Reasoning
- OWL Semantics and Profiles
- Reasoning with Description Logics
- SWRL

OWL2 semantics

- The Direct Semantics and the RDF-Based Semantics provide two alternative ways of assigning meaning to OWL 2 ontologies
  - A correspondence theorem provides a link between the two
- These two semantics are used by reasoners and other tools to answer class consistency, subsumption, instance retrieval queries, …

OWL 2 RDF-based semantics

- Assigns meaning directly to RDF graphs and so indirectly to ontology structures via the Mapping to RDF graphs
- The RDF-Based Semantics is fully compatible with the RDF Semantics, and extends the semantic conditions defined for RDF
- The RDF-Based Semantics can be applied to any OWL 2 Ontology, without restrictions, as any OWL 2 Ontology can be mapped to RDF
- "**OWL 2 Full**" is used informally to refer to RDF graphs considered as OWL 2 ontologies and interpreted using the RDF-Based Semantics
- **"OWL 2 Full" is not decidable**

OWL 2 direct semantics

- Assigns meaning directly to ontology structures, resulting in a semantics compatible with the model theoretic semantics of the SROIQ description logic
  - SROIQ is a fragment of first order logic
- The advantage of this close connection is that the extensive description logic literature and implementation experience can be directly exploited by OWL 2 tools
- Ontologies that satisfy these syntactic conditions are called **OWL 2 DL** ontologies
- **OWL-DL is decidable**

OWL-DL class constructors

Constructor	DL Syntax	Example	Modal Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male	$C_1 \wedge \ldots \wedge C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer	$C_1 \vee \ldots \vee C_n$
complementOf	$\neg C$	$\neg$Male	$\neg C$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} $\sqcup$ {mary}	$x_1 \vee \ldots \vee x_n$
allValuesFrom	$\forall P.C$	$\forall$hasChild.Doctor	$[P]C$
someValuesFrom	$\exists P.C$	$\exists$hasChild.Lawyer	$\langle P \rangle C$
maxCardinality	$\leqslant nP$	$\leqslant$1hasChild	$[P]_{n+1}$
minCardinality	$\geqslant nP$	$\geqslant$2hasChild	$\langle P \rangle_n$

OWL-DL axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg$ {peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+ \sqsubseteq$ ancestor
functionalProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant 1$hasMother
inverseFunctionalProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant 1$hasSSN$^-$

OWL-DL Reasoning Rules

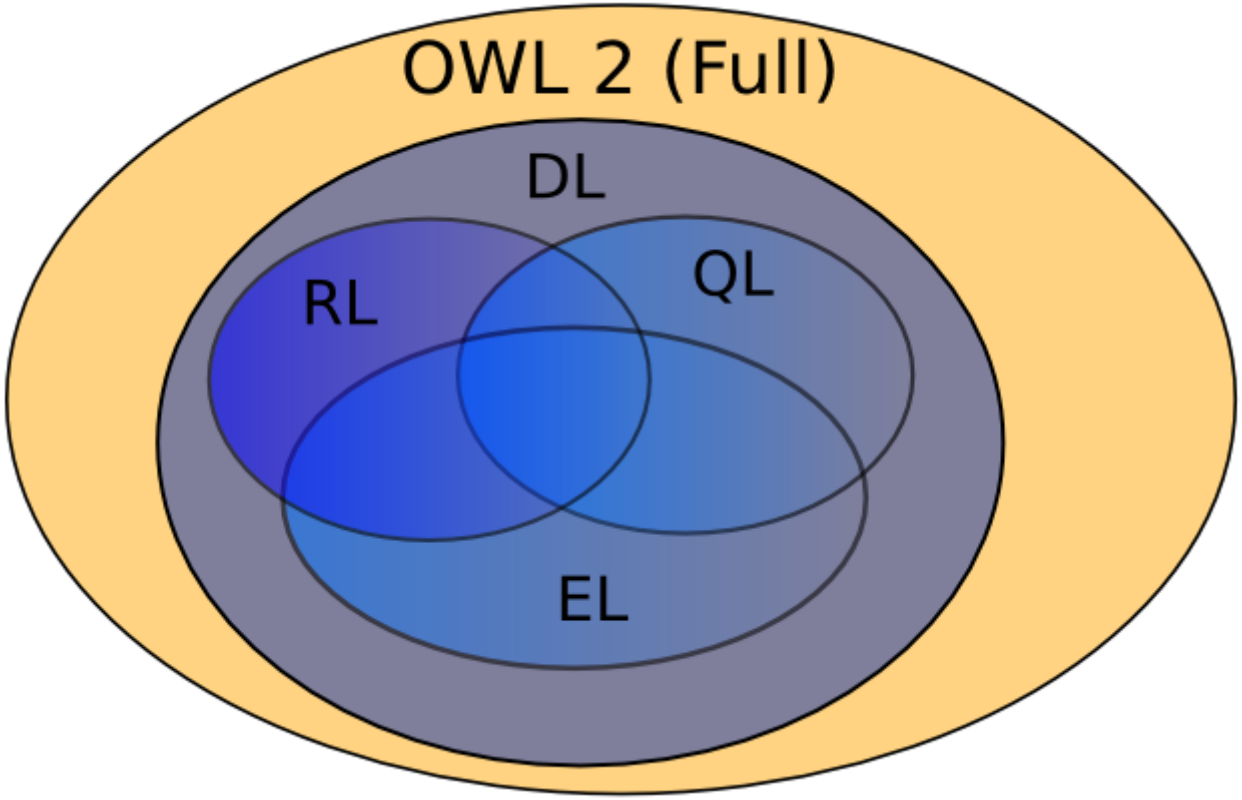
$\sqcap$-rule:	if	$C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$,
	then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule:	if	$C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{E\}$ for some $E \in \{C_1, C_2\}$
$\exists$-rule:	if	$\exists S.C \in \mathcal{L}(x)$, x is not blocked, and x has no S-neighbour y with $C \in \mathcal{L}(y)$
	then	create a new node y with $\mathcal{L}(\langle x, y\rangle) := \{S\}$ and $\mathcal{L}(y) := \{C\}$
Self–Ref-rule:	if	$\exists S.\mathsf{Self} \in \mathcal{L}(x)$ or $\mathsf{Ref}(S) \in \mathcal{R}_a$, x is not blocked, and $S \notin \mathcal{L}(\langle x, x\rangle)$
	then	add an edge $\langle x, x\rangle$ if it does not yet exist, and set $\mathcal{L}(\langle x, x\rangle) \longrightarrow \mathcal{L}(\langle x, x\rangle) \cup \{S\}$
$\forall_1$-rule:	if	$\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and $\forall \mathcal{B}_S.C \notin \mathcal{L}(x)$
	then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{\forall \mathcal{B}_S.C\}$
$\forall_2$-rule:	if	$\forall \mathcal{B}(p).C \in \mathcal{L}(x)$, x is not indirectly blocked, $p \xrightarrow{S} q$ in $\mathcal{B}(p)$, and there is an S-neighbour y of x with $\forall \mathcal{B}(q).C \notin \mathcal{L}(y)$,
	then	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall \mathcal{B}(q).C\}$
$\forall_3$-rule:	if	$\forall \mathcal{B}.C \in \mathcal{L}(x)$, x is not indirectly blocked, $\varepsilon \in L(\mathcal{B})$ and $C \notin \mathcal{L}(x)$
	then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$
choose-rule:	if	$(\leqslant n S.C) \in \mathcal{L}(x)$, x is not indirectly blocked, and there is an S-neighbour y of x with $\{C, \doteq\neg C\} \cap \mathcal{L}(y) = \emptyset$
	then	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \doteq\neg C\}$

$\geqslant$-rule:	if	1. $(\geqslant n S.C) \in \mathcal{L}(x)$, x is not blocked
		2. there are not n safe S-neighbours $y_1, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$ and $y_i \neq y_j$ for $1 \leq i < j \leq n$
	then	create n new nodes $y_1, \ldots, y_n$ with $\mathcal{L}(\langle x, y_i\rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$.
$\leqslant$-rule:	if	1. $(\leqslant n S.C) \in \mathcal{L}(z)$, z is not indirectly blocked
		2. $\sharp S^{\mathbf{G}}(z, C) > n$ and there are two S-neighbours x, y of z with $C \in \mathcal{L}(x) \cap \mathcal{L}(y)$, and not $x \neq y$
	then	1. if x is a nominal node then $\mathsf{Merge}(y, x)$
		2. else, if y is a nominal node or an ancestor of x then $\mathsf{Merge}(x, y)$
		3. else $\mathsf{Merge}(y, x)$
o-rule:	if	for some $o \in N_I$ there are 2 nodes x, y with $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$ and not $x \neq y$
	then	$\mathsf{Merge}(x, y)$
NN-rule:	if	1. $(\leqslant n S.C) \in \mathcal{L}(x)$, x is a nominal node, and there is a blockable S-neighbour y of x such that $C \in \mathcal{L}(y)$ and x is a successor of y,
		2. there is no m such that $1 \leqslant m \leqslant n$, $(\leqslant m S.C) \in \mathcal{L}(x)$, and there exist m nominal S-neighbours $z_1, \ldots, z_m$ of x with $C \in \mathcal{L}(z_i)$ and $z_i \neq z_j$ for all $1 \leq i < j \leq m$.
	then	1. guess m with $1 \leqslant m \leqslant n$, and set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{(\leqslant m S.C)\}$
		2. create m new nodes $y_1, \ldots, y_m$ with $\mathcal{L}(\langle x, y_i\rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C, o_i\}$, for each $o_i \in N_I$ new in $\mathbf{G}$, and $y_i \neq y_j$ for $1 \leq i < j \leq m$.

OWL2 profiles

- Decidable does not mean efficient nor convenient
- OWL 2 profiles are sub-languages (i.e. syntactic subsets) of OWL 2 that offer important advantages in particular application scenarios
- Three different profiles are defined
  - OWL 2 EL, OWL 2 QL, OWL 2 RL
- Each profile is a **syntactic** restriction of the OWL 2 Structural Specification, i.e., as a subset of the structural elements that can be used in a conforming ontology, and each is **more restrictive than OWL DL**
- Each of the profiles trades off different aspects of OWL **expressive power** in return for different **computational** and/or implementation **benefits**

OWL Profiles



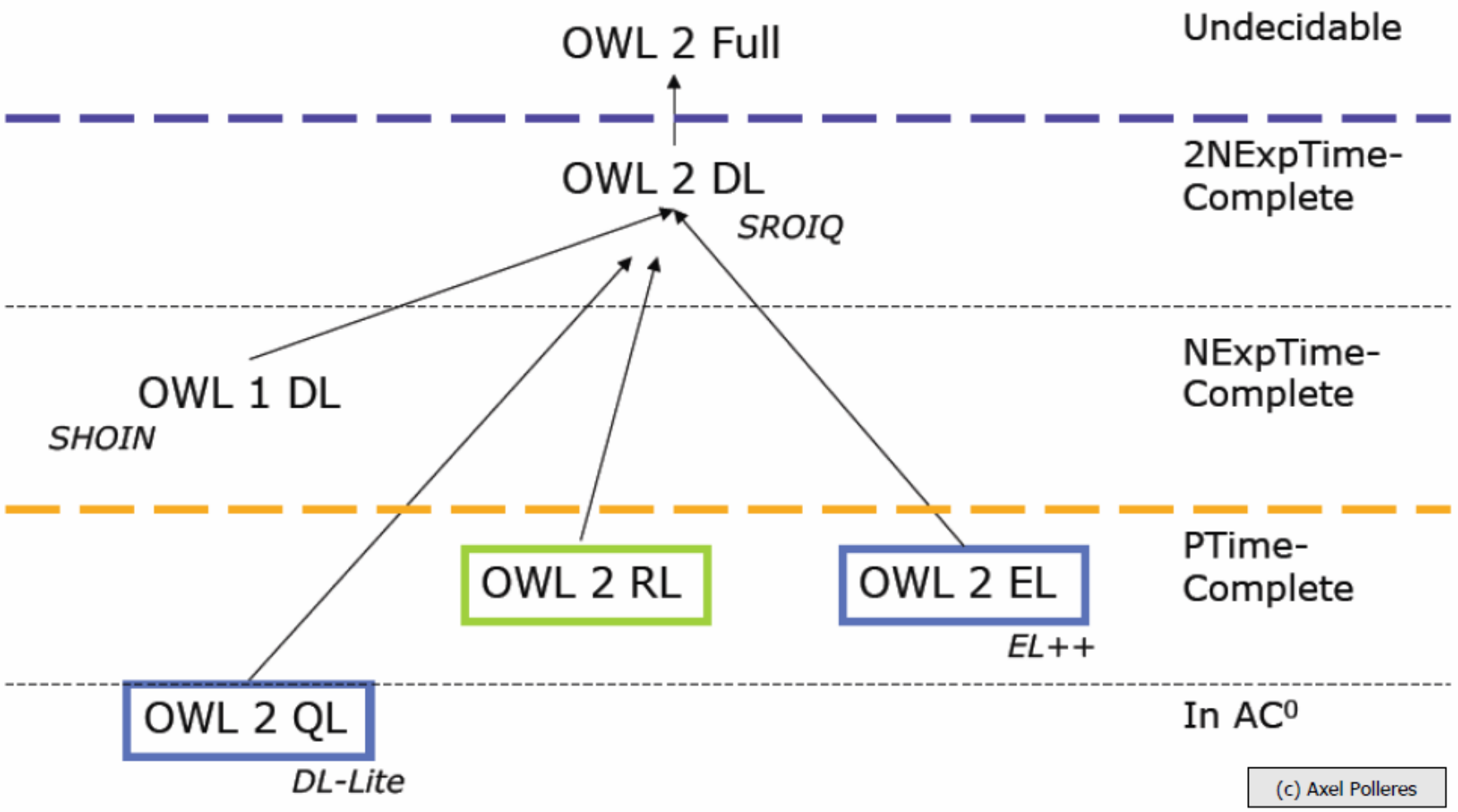
OWL2 profiles

- OWL 2 EL "Existential quantification Language"
  - Enables **polynomial** time algorithms for all the standard reasoning tasks
  - It is particularly suitable for applications where very **large** ontologies are needed, and where expressive power can be traded for performance guarantees
- OWL 2 QL "Query Language"
  - Enables **conjunctive queries** to be answered in LogSpace (more precisely, AC0) using standard relational **database** technology
  - It is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to access the data directly via relational queries (e.g., SQL)

OWL2 profiles

- OWL 2 RL "Rule Language"
  - Enables the implementation of polynomial time reasoning algorithms using rule-extended database technologies operating directly on RDF triples
  - It is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to operate directly on data in the form of RDF triples
- Any OWL 2 EL, QL or RL ontology is, of course, also an OWL 2 ontology and can be interpreted using either the Direct or RDF-Based Semantics

OWL2 semantics and profiles overview



OWL2-EL characteristics

- Ontologies with complex structural descriptions, huge numbers of classes, heavy use of classification, application of the resulting terminology to vast amounts of data
- Expressive class expression language, no restrictions on how they may be used in axioms
- Fairly expressive property expressions, including property chains, but excluding inverse
- **Forbidden**: negation, disjunction, universal quantification on properties, all kinds of role inverses

https://www.w3.org/TR/owl2-primer/

OWL2-EL Feature Overview

OWL 2 EL places restrictions on the type of class restrictions that can be used in axioms. In particular, the following types of class restrictions are supported:

- existential quantification to a class expression (**ObjectSomeValuesFrom**) or a data range (**DataSomeValuesFrom**)
- existential quantification to an individual (**ObjectHasValue**) or a literal (**DataHasValue**)
- self-restriction (**ObjectHasSelf**)
- enumerations involving a *single* individual (**ObjectOneOf**) or a *single* literal (**DataOneOf**)
- intersection of classes (**ObjectIntersectionOf**) and data ranges (**DataIntersectionOf**)

OWL 2 EL supports the following axioms, all of which are restricted to the allowed set of class expressions:

- class inclusion (**SubClassOf**)
- class equivalence (**EquivalentClasses**)
- class disjointness (**DisjointClasses**)
- object property inclusion (**SubObjectPropertyOf**) with or without property chains, and data property inclusion (**SubDataPropertyOf**)
- property equivalence (**EquivalentObjectProperties** and **EquivalentDataProperties**),
- transitive object properties (**TransitiveObjectProperty**)
- reflexive object properties (**ReflexiveObjectProperty**)
- domain restrictions (**ObjectPropertyDomain** and **DataPropertyDomain**)
- range restrictions (**ObjectPropertyRange** and **DataPropertyRange**)
- assertions (**SameIndividual**, **DifferentIndividuals**, **ClassAssertion**, **ObjectPropertyAssertion**, **DataPropertyAssertion**, **NegativeObjectPropertyAssertion**, and **NegativeDataPropertyAssertion**)
- functional data properties (**FunctionalDataProperty**)
- keys (**HasKey**)

The following constructs are not supported in OWL 2 EL:

- universal quantification to a class expression (**ObjectAllValuesFrom**) or a data range (**DataAllValuesFrom**)
- cardinality restrictions (**ObjectMaxCardinality**, **ObjectMinCardinality**, **ObjectExactCardinality**, **DataMaxCardinality**, **DataMinCardinality**, and **DataExactCardinality**)
- disjunction (**ObjectUnionOf**, **DisjointUnion**, and **DataUnionOf**)
- class negation (**ObjectComplementOf**)
- enumerations involving more than one individual (**ObjectOneOf** and **DataOneOf**)
- disjoint properties (**DisjointObjectProperties** and **DisjointDataProperties**)
- irreflexive object properties (**IrreflexiveObjectProperty**)
- inverse object properties (**InverseObjectProperties**)
- functional and inverse-functional object properties (**FunctionalObjectProperty** and **InverseFunctionalObjectProperty**)
- symmetric object properties (**SymmetricObjectProperty**)
- asymmetric object properties (**AsymmetricObjectProperty**)

https://www.w3.org/TR/owl2-profiles

OWL2-QL characteristics

- Can represent key features of Entity-relationship and UML diagrams, suitable for representing database schemas and for integrating them via query rewriting
- Can also be used directly as a high level database schema language
- Captures many commonly used features in RDFS and small extensions (such as inverse properties and subproperty hierarchies)
- Restricts class axioms asymmetrically
- **Forbidden**: existential quantification of roles to a class expression, property chain axioms and equality

https://www.w3.org/TR/owl2-primer/

OWL2-QL Feature Overview

OWL 2 QL is defined not only in terms of the set of supported constructs, but it also restricts the places in which these constructs are allowed to occur. The allowed usage of constructs in class expressions is summarized in Table 1.

Table 1. Syntactic Restrictions on Class Expressions in OWL 2 QL

Subclass Expressions	**Superclass Expressions**
a class existential quantification (**ObjectSomeValuesFrom**) where the class is limited to *owl:Thing* existential quantification to a data range (**DataSomeValuesFrom**)	a class intersection (**ObjectIntersectionOf**) negation (**ObjectComplementOf**) existential quantification to a class (**ObjectSomeValuesFrom**) existential quantification to a data range (**DataSomeValuesFrom**)

OWL 2 QL supports the following axioms, constrained so as to be compliant with the mentioned restrictions on class expressions:

- subclass axioms (**SubClassOf**)
- class expression equivalence (**EquivalentClasses**)
- class expression disjointness (**DisjointClasses**)
- inverse object properties (**InverseObjectProperties**)
- property inclusion (**SubObjectPropertyOf** not involving property chains and **SubDataPropertyOf**)
- property equivalence (**EquivalentObjectProperties** and **EquivalentDataProperties**)
- property domain (**ObjectPropertyDomain** and **DataPropertyDomain**)
- property range (**ObjectPropertyRange** and **DataPropertyRange**)
- disjoint properties (**DisjointObjectProperties** and **DisjointDataProperties**)
- symmetric properties (**SymmetricObjectProperty**)
- reflexive properties (**ReflexiveObjectProperty**)
- irreflexive properties (**IrreflexiveObjectProperty**)
- asymmetric properties (**AsymmetricObjectProperty**)
- assertions other than individual equality assertions and negative property assertions (**DifferentIndividuals**, **ClassAssertion**, **ObjectPropertyAssertion**, and **DataPropertyAssertion**)

The following constructs are not supported in OWL 2 QL:

- existential quantification to a class expression or a data range (**ObjectSomeValuesFrom** and **DataSomeValuesFrom**) in the subclass position
- self-restriction (**ObjectHasSelf**)
- existential quantification to an individual or a literal (**ObjectHasValue**, **DataHasValue**)
- enumeration of individuals and literals (**ObjectOneOf**, **DataOneOf**)
- universal quantification to a class expression or a data range (**ObjectAllValuesFrom**, **DataAllValuesFrom**)
- cardinality restrictions (**ObjectMaxCardinality**, **ObjectMinCardinality**, **ObjectExactCardinality**, **DataMaxCardinality**, **DataMinCardinality**, **DataExactCardinality**)
- disjunction (**ObjectUnionOf**, **DisjointUnion**, and **DataUnionOf**)
- property inclusions (**SubObjectPropertyOf**) involving property chains
- functional and inverse-functional properties (**FunctionalObjectProperty**, **InverseFunctionalObjectProperty**, and **FunctionalDataProperty**)
- transitive properties (**TransitiveObjectProperty**)
- keys (**HasKey**)
- individual equality assertions and negative property assertions

OWL 2 QL does not support individual equality assertions (**SameIndividual**): adding such axioms to OWL 2 QL would increase the data complexity of query answering, so that it is no longer first order rewritable, which means that query answering could not be implemented directly using relational database technologies. However, an ontology *O* that includes individual equality assertions, but is otherwise OWL 2 QL, could be handled by computing the reflexive–symmetric–transitive closure of the equality (**SameIndividual**) relation in *O* (this requires answering recursive queries and can be implemented in LOGSPACE w.r.t. the size of data) [*DL-Lite-bool*], and then using this relation in query answering procedures to simulate individual equality reasoning [*Automated Reasoning*].

https://www.w3.org/TR/owl2-profiles

OWL2-RL characteristics

- For applications that require scalable reasoning without sacrificing too much expressive power
- Designed to be as expressive as possible while allowing implementation using rules and a rule-processing system (only conjunctive rules)
- We cannot (easily) talk about unknown individuals in our superclass expressions
- Disallows statements where the existence of an individual enforces the existence of another individual
- Restricts class axioms asymmetrically

https://www.w3.org/TR/owl2-primer/

OWL2-RL Feature Overview

Restricting the way in which constructs are used makes it possible to implement reasoning systems using rule-based reasoning engines, while still providing desirable computational guarantees. These restrictions are designed so as to avoid the need to infer the existence of individuals not explicitly present in the knowledge base, and to avoid the need for nondeterministic reasoning. This is achieved by restricting the use of constructs to certain syntactic positions. For example in `SubClassOf` axioms, the constructs in the subclass and superclass expressions must follow the usage patterns shown in Table 2.

Table 2. Syntactic Restrictions on Class Expressions in OWL 2 RL

Subclass Expressions	**Superclass Expressions**
a class other than *owl:Thing* an enumeration of individuals (**ObjectOneOf**) intersection of class expressions (**ObjectIntersectionOf**) union of class expressions (**ObjectUnionOf**) existential quantification to a class expression (**ObjectSomeValuesFrom**) existential quantification to a data range (**DataSomeValuesFrom**) existential quantification to an individual (**ObjectHasValue**) existential quantification to a literal (**DataHasValue**)	a class other than *owl:Thing* intersection of classes (**ObjectIntersectionOf**) negation (**ObjectComplementOf**) universal quantification to a class expression (**ObjectAllValuesFrom**) existential quantification to an individual (**ObjectHasValue**) at-most 0/1 cardinality restriction to a class expression (**ObjectMaxCardinality** 0/1) universal quantification to a data range (**DataAllValuesFrom**) existential quantification to a literal (**DataHasValue**) at-most 0/1 cardinality restriction to a data range (**DataMaxCardinality** 0/1)

All axioms in OWL 2 RL are constrained in a way that is compliant with these restrictions. Thus, OWL 2 RL supports all axioms of OWL 2 apart from disjoint unions of classes (**DisjointUnion**) and reflexive object property axioms (**ReflexiveObjectProperty**).

https://www.w3.org/TR/owl2-profiles

Outline

- Automated Reasoning
- OWL Semantics and Profiles
- Reasoning with Description Logics
- SWRL

Description Logics: Syntax

- *Concepts* corresponds to classes
- *Roles* correspond to class properties
- *Constructors* mix of set notation and FO quantification

  Booleans: $\boldsymbol{C \sqcap D}$, $\boldsymbol{C \sqcup D}$, $\boldsymbol{\neg C}$
  Qualified quantification: $\boldsymbol{\forall R.C}$, $\boldsymbol{\exists R.C}$

- Variable free notation for concepts (classes)
  - $\boldsymbol{artist(x) = person(x) \wedge \exists y\, created(x,y) \wedge Artwork(y)}$
    is written as $Artist \sqsubseteq Person \sqcap \exists created.Artwork$

Description Logic: Semantics

- Interpretations are pairs $(\Delta, \cdot^{\mathcal{I}})$, with a universe Δ and a mapping $\mathcal{I}$ from
  - concept names to subsets of Δ
  - role names to binary relations
- Booleans:

$C \sqcap D$,	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}$

Qualified quantification:

$\forall R.C \quad \forall R.C^{\mathcal{I}} = \{x \in \Delta \mid \forall y \in \Delta : R^{\mathcal{I}}(x,y) \rightarrow y \in C^{\mathcal{I}})\}$

$\exists R.C \quad \exists R.C^{\mathcal{I}} = \{x \in \Delta \mid \exists y \in \Delta : R^{\mathcal{I}}(x,y) \& y \in C^{\mathcal{I}})\}$

Modular Definition of Description Logics

Constructor	Syntax	Semantics
concept name	C	$C^{\mathcal{I}}$
conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
univ. quant.	$\forall R.C$	$\{d_1 \mid \forall d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}} d_1 d_2 \rightarrow d_2 \in C^{\mathcal{I}})\}$
top	$\top$	$\Delta^{\mathcal{I}}$
negation ($\mathcal{C}$)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
disjunction ($\mathcal{U}$)	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
exist. quant. ($\mathcal{E}$)	$\exists R.C$	$\{d_1 \mid \exists d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}} d_1 d_2 \wedge d_2 \in C^{\mathcal{I}})\}$
number restr. ($\mathcal{N}$)	$\geq nR$	$\{d_1 \mid \lvert\{d_2 \mid R^{\mathcal{I}} d_1 d_2\}\rvert \geq n\}$
	$\leq nR$	$\{d_1 \mid \lvert\{d_2 \mid R^{\mathcal{I}} d_1 d_2\}\rvert \leq n\}$
one-of ($\mathcal{O}$)	$\{a_1, \ldots, a_n\}$	$\{d \mid d = a_i^{\mathcal{I}}$ for some $a_i\}$
filler ($\mathcal{B}$)	$\exists R.\{a\}$	$\{d \mid d = R^{\mathcal{I}} d a^{\mathcal{I}}\}$
role name	R	$R^{\mathcal{I}}$
role conj. ($\mathcal{R}$)	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$
inverse roles ($\mathcal{I}$)	R^{-1}	$\{(d_1, d_2) \mid R^{\mathcal{I}}(d_2, d_1)\}$

Reasoning

- With the definition of the semantics, we may now define some reasoning metods
  - Reasoning on the structure of the ontology
  - Reasoning on relationships among classes
  - Reasoning on instances

Concept Reasoning

Based on these semantics there are two basic reasoning services:

- *Concept satisfiability*, $\models C \neq \bot$.
  - Check whether for some interpretation $\mathcal{I}$ we have $C^{\mathcal{I}} \neq \varnothing$.
  - $\models \forall creates.Sculpture \sqcap \exists creates.(Artwork \sqcap \neg Sculpture) = \bot$.
- *Concept subsumption*, $\models C_1 \sqsubseteq C_2$.
  - Check whether for all interpretations $\mathcal{I}$ we have $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
  - $\forall creates.Painting \sqcap \exists creates.\top \sqsubseteq \exists creates.Painting$.

Terminological Reasoning

$$\begin{aligned}\mathcal{T} = \{ & \mathit{Painting} \sqsubseteq \mathit{Artwork} \sqcap \neg \mathit{Sculpture}, \\ & \mathit{Painter} \sqsubseteq \exists \mathit{creates}.\mathit{Paintings}, \\ & \mathit{Sculpturer} \sqsubseteq \exists \mathit{creates}.\mathit{Artwork} \sqcap \forall \mathit{creates}.\mathit{Sculpture} \ \}\end{aligned}$$

- *Concept satisfiability*, $\Sigma \models C \neq \bot$.
  - Check whether there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ and $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
  - *Concept unsatisfiability:* $\Sigma \models \mathit{Painter} \sqcap \mathit{Sculpturer} = \bot$.
- *Subsumption*, $\Sigma \models C_1 \sqsubseteq C_2$.
  - Check whether for all interpretations $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ we have $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
  - *Subsumption:* $\Sigma \models \mathit{Painter} \sqsubseteq \neg \mathit{Sculpurer}$

Assertional reasoning

$\mathcal{A} = \{rembrandt{:}Artist,\ nightwatch{:}Painting,\ (rembrandt,nightwatch){:}created\}$ and $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$

- *Consistency*, $\Sigma \not\models \bot$.
  - Check whether there exists $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$.
  - $\Sigma \models \mathcal{A} \neq \bot$ but $\Sigma \models \mathcal{A} \cup \{rembrandt{:}Sculpturor\} = \bot$
- *Instance Checking*, $\Sigma \models a{:}C$.
  - Check whether $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all interpretations $\mathcal{I} \models \Sigma$.
  - $rembrandt \in_{\Sigma} Painter$.
- Defined reasoning tasks:
  - *Retrieval:* $retrieve(Artists) = \{rembrandt\}$.
  - *Realization:* find most specific concepts in $\mathcal{T}$ for instances in $\mathcal{A}$ $realize(rembrandt) = Painter$

What is an OWL-DL reasoner

- The official normative definition:
  - An *OWL consistency checke*r takes a document as input, and returns one word being Consistent, Inconsistent, or Unknown. [J. J. Carroll, J. D. Roo, OWL Web Ontology Language Test Cases, W3C Recommendation http://www.w3.org/TR/owl-test/ (2004).]
  - Rather restrictive... and not very useful for ontology development, debug and querying

DL Jargon

Abbr.	Stands for	Meaning
ABox	**Assertional Box**	Component that contains assertions about individuals, i.e. OWL facts such as type, property-value, equality or inequality assertions.
TBox	**Terminological Box**	Component that contains axioms about classes, i.e. OWL axioms such as subclass, equivalent class or disjointness axioms.
KB	**Knowledge Base**	A combination of an ABox and a TBox, i.e. a complete OWL ontology.

Classical Types of Logic Inference

- **Consistency checking**, which ensures that an ontology does not contain any contradictory facts.
  - The OWL Abstract Syntax & Semantics document [S&AS] provides a formal definition of ontology consistency that Pellet uses.
  - In DL terminology, this is the operation to check the consistency of an ABox with respect to a Tbox.
  - Equivalent to OWL Consistency Checking

Classical Types of Logic Inference

- **Concept satisfiability**, which checks if it is possible for a class to have any instances. If class is unsatisfiable, then defining an instance of the class will cause the whole ontology to be inconsistent.

Classical Types of Logic Inference

- **Classification**, which computes the subclass relations between every named class to create the complete class hierarchy. The class hierarchy can be used to answer queries such as getting all or only the direct subclasses of a class.

Classical Types of Logic Inference

- **Realization**, which finds the most specific classes that an individual belongs to; or in other words, computes the direct types for each of the individuals. Realization can only be performed after classification since direct types are defined with

Some OWL reasoners

- Fact++
  - C++, OpenSource, OWL-DL, http://owl.man.ac.uk/factplusplus/
- Hermit
  - Java, Open Source, DL Safe, novel ‘tableau’ algorithm, http://www.hermit-reasoner.com/
- Kaon2
  - Java, OWL-DL, http://kaon2.semanticweb.org/
- Pellet
  - Java, Open Source + commercial support, OWL-DL, http://clarkparsia.com/pellet/
- RacerPro
  - Commercial, OWL-DL, http://www.racer-systems.com/products/racerpro/
- Vampire
  - Commercial, theorem prover, novel approach, still undergoing, www.cs.man.ac.uk/~tsarkov/papers/TRBH04a.pdf

Outline

- Automated Reasoning
- OWL Semantics and Profiles
- Reasoning with Description Logics
- SWRL

Semantic Web Rule Language (SWRL)

- Not an official W3C Recommedation
- Application to OWL of the RuleML (http://ruleml.org/) languages
- Extends OWL language by providing Horn clauses
- Defines an extension of the OWL model-theoretic semantics

SWRL structure

- The rules are of the form of an implication between an **antecedent** (body) and **consequent** (head).
- The intended meaning can be read as:
  - whenever the conditions specified in the antecedent hold,
  - then the conditions specified in the consequent must also hold.

General structure

- Both the antecedent (body) and consequent (head) consist of zero or more atoms.
  - An **empty antecedent** is treated as trivially **true** (i.e. satisfied by every interpretation), so the consequent must also be satisfied by every interpretation;
  - an **empty consequent** is treated as trivially **false** (i.e., not satisfied by any interpretation), so the antecedent must also not be satisfied by any interpretation.
- Multiple atoms are treated as a **conjunction**

Rule structure

- A SWRL rule contains an antecedent part, which is referred to as the *body*, and a consequent part, which is referred to as the *head*.
- Both the body and head consist of positive conjunctions of *atoms*
  - `atom ^ atom .... -> atom ^ atom`
- SWRL does not support negated atoms or disjunction.

Atoms

- Atoms in these rules can be of the form
  - **C(x)**, where C is an OWL description (class)
  - **P(x,y)**, where P is an OWL property
  - **sameAs(x,y)**
  - **differentFrom(x,y)**
  - where x, y are either variables, OWL individuals or OWL data values

Atoms

- p(arg1, arg2, ... Argn)
- *p* is a predicate symbol
  - OWL classes, properties or data types
- *arg1*, *arg2*, ..., *argn* are the terms or arguments of the expression
  - OWL individuals or data values,
  - variables referring to them
- All variables in SWRL are treated as universally quantified

Atom types

- Class Atoms
  - Person(?p)
  - Man(Fred)
- Individual Property atoms
  - hasBrother(?x, ?y)
  - hasSibling(Fred, ?y)
- Data Valued Property atoms
  - hasAge(?x, ?age)
  - hasHeight(Fred, ?h)
  - hasAge(?x, 232)
  - hasName(?x, "Fred")
- Different Individuals atoms
  - differentFrom(?x, ?y)
  - differentFrom(Fred, Joe)
- Same Individual atoms
  - sameAs(?x, ?y)
  - sameAs(Fred, Freddy)
- Built-in atoms
  - Runtime-provided functions
  - Core built-ins in swrlb:
- Data Range atoms

Syntax issues

- SWRL rules are defined according to different syntax forms
  - Abstract syntax (in functional form)
  - XML concrete syntax
  - RDF concrete syntax
  - Human-readable form (using logic predicates)

Example: uncle

- Human-readable syntax
  - `hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) -> hasUncle(?x1,?x3)`
- Abstract syntax
  - ```
    Implies(Antecedent(
     hasParent(I-variable(x1) I-variable(x2))
     hasBrother(I-variable(x2) I-variable(x3)))
    Consequent(
     hasUncle(I-variable(x1) I-variable(x3))))
    ```
- Example: if John has Mary as a parent and Mary has Bill as a brother then John has Bill as an uncle
    ```

Example: inheritance

- Human-readable syntax
  - `Student(?x1) -> Person(?x1)`
- Abstract syntax
  - `Implies(Antecedent(Student(I-variable(x1)))`
    `Consequent(Person(I-variable(x1))))`

- This is an **improper usage** of rules: it should be expressed directly in OWL, to make the information also available to an OWL reasoner
  - `SubClassOf(Student Person)`

Example: propagating properties

- Human-readable syntax
  - `Artist(?x) ^ artistStyle(?x,?y) & Style(?y) ^ creator(?z,?x) -> style/period(?z,?y)`
- Abstract syntax
  - `Implies(Antecedent( Artist(I-variable(x)) artistStyle(I-variable(x) I-variable(y)) Style(I-variable(y)) creator(I-variable(z) I-variable(x))) Consequent(style/period(I-variable(z) I-variable(y))))`
- Meaning: the style of an art object is the same as the style of the creator

SWRL versus OWL

- The last example may not be described in OWL
- In OWL, you declare relationships between Classes
- Such relationships are intended to apply on instances
  - You may add properties to instances to materialize such relationships
- OWL Inference only supports “forall” or “exists” in propagating properties
- In OWL you may not express “that specific instance that has such properties”!

OWL versus SWRL

- OWL has a declarative nature, while SWRL is more operational
  - Even if the semantics extends that of OWL, practical reasoners just “apply the rules”
- The consistency of the rules application relies on the rule designer’s infinite wisdom
- Example: If a property is declared as symmetric, then we must be careful to create all property instances to satisfy that

SWRL in Protege

- Enable "SWRL Tab"
- Uses the "Drools" rule engine
- 3-step process:
  - OWL + Rules transferred to Drools
  - Running Drools
  - Inferred statements transferred back to OWL (optional)

References

- Course material for “Practical Reasoning for the Semantic Web” course at the 17th European Summer School in Logic, Language and Information (ESSLLI)
  - http://www.few.vu.nl/~schlobac/
- E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, no. 2, pp. 51-53, June 2007. [Online]. Available: http://dx.doi.org/10.1016/j.websem.2007.03.004

References

- OWL Web Ontology Language: Semantics and Abstract Syntax – W3C Recommendation 10 February 2004 [S&AS]
  - http://www.w3.org/TR/owl-semantics/
- OWL 2 Web Ontology Language Profiles (Second Edition)
  - https://www.w3.org/TR/owl2-profiles/
- OWL 2 Web Ontology Language Primer (Second Edition)
  - https://www.w3.org/TR/owl2-primer/

References (SWRL)

- SWRL API in Protégé
  - https://github.com/protegeproject/swrlapi/wiki
- SWRL Language FAQ
  - https://github.com/protegeproject/swrlapi/wiki/SWRLLanguageFAQ

License