OWL expressivity with respect to logical patterns

- What was the necessity for the creation of OWL profiles?
  - Hard reasoning due to high complexities of problems
- What's the definition of profiles in OWL?
  - Subsets of OWL 2 which restrict expressivity in order to have better computational properties (reasoning problems w.r.t. complexity)
- Profiles of OWL 2:
  - OWL 2 EL
  - OWL 2 QL
  - OWL 2 RL
- Logical patterns:
  - Representing classes as property values on the semantic web (5 approaches)
  - Defining N-ary relations on the semantic web: use with individuals (3 approaches)
  - Representing Specified collection of Values in OWL: "value partitions" and "value sets " (2 approaches)

OWL EL

- Useful for applications where the basic reasoning problems can be decided in a polynomial time
- It has the following characteristics:
  - 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)
  - 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)

OWL EL(cont.)

- The following constructs are not supported in OWL 2 EL:
  - universal quantification to a class expression (ObjectAllValuesFrom) or a data range (DatAllaValuesFrom)
  - 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)

OWL QL

- OWL QL has as main reasoning task the query answering
- It has the following characteristics:
  - 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)
  - assertions other than the equality assertions (DifferentIndividuals, ClassAssertion, ObjectPropertyAssertion, and DataPropertyAssertion)

OWL QL (cont.)

- The following constructs are not supported in OWL 2 QL:
  - existential quantification to a class expression or a data range (ObjectSomeValuesFrom 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)
  - reflexive properties (ReflexiveObjectProperty)
  - irreflexive properties (IrreflexiveObjectProperty)
  - asymmetric properties (AsymmetricObjectProperty)
  - keys (HasKey)

OWL R

- OWL R is using technologies based on rules and also serves RDFS applications that need some added OWL expressivity
- OWL RL can use most of the structures of OWL 2 except from:
  - cardinality
  - minCardinality
  - NegativeObjectPropertyAssertion
  - NegativeDataPropertyAssertion
  - Owl:complementOf

Representing classes as property values

- Approaches 1 (Use classes directly as property values), 2 (Create special instances of the class to be used as property values), 3 (Create a parallel hierarchy of instances as property values) correspond to OWL EL because:
  - In OWL QL there is disallowance of existential quantification (`ObjectSomeValuesFrom`)
  - IN OWL R there is disallowance of defining a class as the subclass of the union of two other classes
- Approach 4 (Create a special restriction instead of using a specific value for an instance) corresponds to OWL R because:
  - The other two dialects are having disallowed anonymous classes (keyword `Restriction`)

Defining N-ary relations

- Approaches 1 (where additional attributes describe a relation), 2 (where we have different aspects of the same relation) correspond to OWL R because:
  - The other two dialects don't support enumerations involving more than one individual [eg. class1 has equivalent class: {individual1, individual2, ...}]
  - They don't support the `FunctionalObjectProperty`
- Approach 3 (where we have N-ary relation with no distinguished participant) doesn't correspond to any of the dialects because:
  - It makes use of cardinalities

Representing Specified collection of Values in OWL: "value partitions" and "value sets "

- Approaches 1 (where values are represented as sets of individuals), 2 (where values are represented as disjoint subclasses partitioning a "feature") correspond to OWL R because:
  - The other two dialects don't allow the `FunctionalObjectProperty`

References

- http://www.w3.org/TR/swbp-n-aryRelations/
- http://www.w3.org/TR/swbp-classes-as-values/
- http://www.w3.org/TR/swbp-specified-values/
- http://www.w3.org/TR/owl2-profiles/#Introduction
- http://www.w3.org/2007/OWL/wiki/Profile_Explanations
- http://www.w3.org/TR/owl2-profiles/