COMP318: OWL

www.csc.liv.ac.uk/~valli/Comp318

Dr Valentina Tamma
Room: Ashton 2.12
Dept of computer science
University of Liverpool
V.Tamma@liverpool.ac.uk

Recap

- OWL ontology language
- OWL class constructors
- OWL property restrictions

Three species of OWL 1.0

- OWL Lite
  - Classification hierarchy
  - Simple constraints
- OWL DL
  - Maximal expressiveness while maintaining decidability
  - Standard formalisation in a DL
- OWL Full
  - Very high expressiveness
  - Losing decidability
  - All syntactic freedom of RDF (self-modifying)

Comp 318

OWL Lite vs OWL DL vs OWL Full

- OWL Lite
  - (sub)classes, individuals
  - (sub)properties, domain,range
  - conjunction
  - (in)equality
  - cardinality 0/1
  - datatypes
  - inverse, transitive, symmetric properties
  - `someValuesFrom`
  - `allValuesFrom`
- OWL DL
  - Negation
  - Disjunction
  - Full cardinality
  - Enumerated types
  - `hasValue`
- OWL Full
  - Meta-classes
  - Modify language

OWL 2 profiles

- Sublanguages of OWL2 trading expressive power for efficient reasoning
  - Each supports different application scenarios
- OWL 2 EL
  - very large ontologies, efficient reasoning performance guaranteed at the expenses of expressive power;
- OWL 2 RL
  - subclass axioms understood as rule like implication, with head - superclass and body - subclass
  - different restrictions on subclasses and superclasses
  - allows the integration of OWL with rules
- OWL 2 QL
  - useful to query data rich applications
  - different restrictions on subclasses and superclasses
  - suitable for simple, lightweight ontologies with a large number of individuals and it is necessary to access the data directly via SQL queries
  - fast implementation on top of legacy DB systems, relational or RDF

Example

- Translate in turtle syntax the following statements, and add any axiom you think appropriate:
  - john is a lecturer
  - mary is an academic staff member
  - mary is 39 years old
  - COMP1111 is a course
  - each course is taught by at most one staff member
  - john teaches COMP1111
  - mary teaches COMP1111

Example 1 in Manchester syntax

```
ObjectProperty: TaughtBy
    Characteristics: Functional
    Domain: Course
    Range:   AcademicStaffMember

DataProperty: age
    Range: xsd:nonNegativeInteger

Class: AcademicStaffMember
   SubClassOf: Person

Class: Course
   DisjointWith: Person

Class: Lecturer
    SubClassOf: AcademicStaffMember

Individual: comp1111
    Types: Course
    Facts:  isTaughtBy john
                isTaughtBy mary

Individual: john
      Types:Lecturer
      DifferentFrom: mary

Individual: mary
    Types: AcademicStaffMember
    Facts: age "39"^^xsd:nonNegativeInteger
    DifferentFrom: john
```

Example (ctd)

- Is the model we obtain correct, or does it contain contradictory information?
  - If so, what are the statements that cause a contradiction?
  - how would you solve it?

Example in Manchester Syntax

```
ObjectProperty: TaughtBy
    ~~Characteristics: Functional~~
    Domain: Course
    Range:   AcademicStaffMember

DataProperty: age
    Range: xsd:nonNegativeInteger

Class: AcademicStaffMember
   SubClassOf: Person

Class: Course
   DisjointWith: Person

Class: Lecturer
    SubClassOf: AcademicStaffMember
```

```
Individual: comp1111
    Types: Course
    Facts:  ~~isTaughtBy john~~
            isTaughtBy mary

Individual: john
     Types:Lecturer
      DifferentFrom: mary

Individual: mary
    Types: AcademicStaffMember
    Facts: age
"39"^^xsd:nonNegativeInteger
    DifferentFrom: john
```

Example

- Translate in turtle syntax the following statements:
  - first year courses are courses taught only by professors
  - maths courses are taught by mary
  - all academic staff members must teach at least one undergraduate course
  - an undergraduate course is taught by someone

Example

```
FirstYearCourse
     rdf:type owl:Class
     rdfs:subClassOf [ rdf:type owl:Restriction;
                       owl:onProperty :isTaughtBy;
                       owl:AllValuesFrom
                                :Professors ] .
```

Example

- Maths courses are taught by Mary

```
:mathCourse
     rdf:type owl:Class
     rdfs:subClassOf ([ rdf:type owl:Restriction;
                       owl:onProperty :isTaughtBy;
                       owl:hasValue :mary ] ).
:mary rdf:type :Professor .
:mathsCourse rdf:type :FirstYearCourse .
```

Not necessary, should be inferred

Example

- all academic staff members must teach at least one undergraduate course

```
:AcademicStaffMember
     rdf:type owl:Class
     rdfs:subClassOf ([ rdf:type owl:Restriction;
                       owl:onProperty :teaches;
                       owl:someValuesFrom
                          :UndegraduateCourses ]).
```

Example

- an undergraduate course is taught by someone (Academic staff member)

```
:Course
     rdf:type owl:Class
     rdfs:subClassOf ([ rdf:type owl:Restriction;
                        owl:onProperty :isTaughtBy;
                       owl:minQualifiedCardinality 1;
                       owl:onClass :AcademicStaffMember]).
```

Example

- an undergraduate course is taught by someone

```
:Course
     rdf:type owl:Class
     rdfs:subClassOf ([ rdf:type owl:Restriction;
                       owl:onProperty :isTaughtBy;
                       owl:someValuesFrom
                                 :AcademicStaffMember] ).
```

Exercise

- a department has at least 10 members and at most 30 members

Exercise

- a department has at least 10 members and at most 30 members

```
:Department
     rdf:type owl:Class
     rdfs:subClassOf [ rdf:type owl:Restriction;
                       owl:onProperty :hasMember;
                       owl:minQualifiedCardinality 10;
                       owl:onClass :Member]
                    [ rdf:type owl:Restriction;
                       owl:onProperty :hasMember;
                       owl:maxQualifiedCardinality 30;
                       owl:onClass :Member]
```

Example 3

- Translate in turtle or Manchester syntax the following statements:
  - *no course is an academic staff member*
  - *define peopleAtUni as the union of staffMember and student*
  - *a faculty in CS is a faculty member who works in the CS department*
  - *adminStaff are those department staff members that are neither faculty nor technical support staff*

Example 3 in turtle syntax

```
[]  rdf:type     owl:AllDisjointClasses ;
     owl:members  ( :Course  :AcademicStaffMember ) .

PeopleAtUni owl:equivalentClass [
   rdf:type     owl:Class ;
   owl:unionOf ( :AcademicStaffMember :Student ) ] .

FacultyInCS owl:equivalentClass [
   rdf:type     owl:Class ;
   owl:intersectionOf (Faculty
                      [ rdf:type owl:Restriction;
                           owl:onProperty :belongsTo;
                        owl:someValuesFrom :CSDepartment] .

AdminStaff owl:equivalentClass [ rdf:type     owl:Class ;
                                 owl:intersectionOf (DepartmentMember
                                [ rdf:type owl:Class;
                              owl:complementOf [
                                                rdf:type     owl:Class ;
                                                owl:unionOf (Faculty TeachingSupportStaff ] ] )
```

Exercise

- Translate the following statements in Turtle syntax:
  - The class AcademicStaffMember does not share any element with the class TechnicalStaffMember
  - The class StaffMember includes elements that are in the class AcademicStaffMember or in the class TechnicalStaffMember
  - The role isResponsibleForTask is used to relate elements of the class StaffMember to elements of the class MgmtTasks

Sample solution

```
1. rdf:type    owl:AllDisjointClasses ;
   owl:members (:AcademicStaffMember :TechnicalStaffMember)
2. :StaffMember owl:equivalentClass [
      rdf:type  owl:Class ;
      owl:unionOf (:AcademicStaffMember :TechnicalStaffMember)
   ] .
3. :isResponsibleForTask rdfs:domain :StaffMember
                         rdfs:range  :MgmtTasks
```

Exercise

- Given the Knowledge Base described before, decide whether the following statements are reasonable and motivate your answer
  - teachesModule is functional
  - teachesModule is inverseFunctional

Sample solution

- `teachesModule` should not be `functional`
  - an `AcademicStaffMember` can teach more than one module
- `teachesModule` is not `inverseFunctional`,
  - since a specific instance of a `Module` can be taught by two different `AcademicStaffMembers`