Lecture 6

RDF (Continued)

Grigoris Antoniou

Frank van Harmelen

&

https://www.w3.org/RDF

https://www.w3schools.com/xml/xml_rdf.asp

Lecture Outline

1. **Basic Concepts of RDF Schema**
2. The Language of RDF Schema
3. Axiomatic Semantics for RDF and RDFS
4. Direct Semantics based on Inference Rules

Basic Ideas of RDF Schema

- RDF is a universal language that lets users describe resources in their own vocabularies
  - RDF does not assume, nor does it define semantics of any particular application domain
- The user can do so in RDF Schema using:
  - Classes and Properties
  - Class Hierarchies and Inheritance
  - Property Hierarchies

Classes and their Instances

- We must distinguish between
  - Concrete “things” (individual objects) in the domain: Discrete Maths, David Billington etc.
  - Sets of individuals sharing properties called classes: lecturers, students, courses etc.
- Individual objects that belong to a class are referred to as instances of that class
- The relationship between instances and classes in RDF is through **rdf:type**

Why Classes are Useful

- Impose restrictions on what can be stated in an RDF document using the schema

  - As in programming languages

  - E.g. A+1, where A is an array
  - Disallow nonsense from being stated

Nonsensical Statements disallowed through the Use of Classes

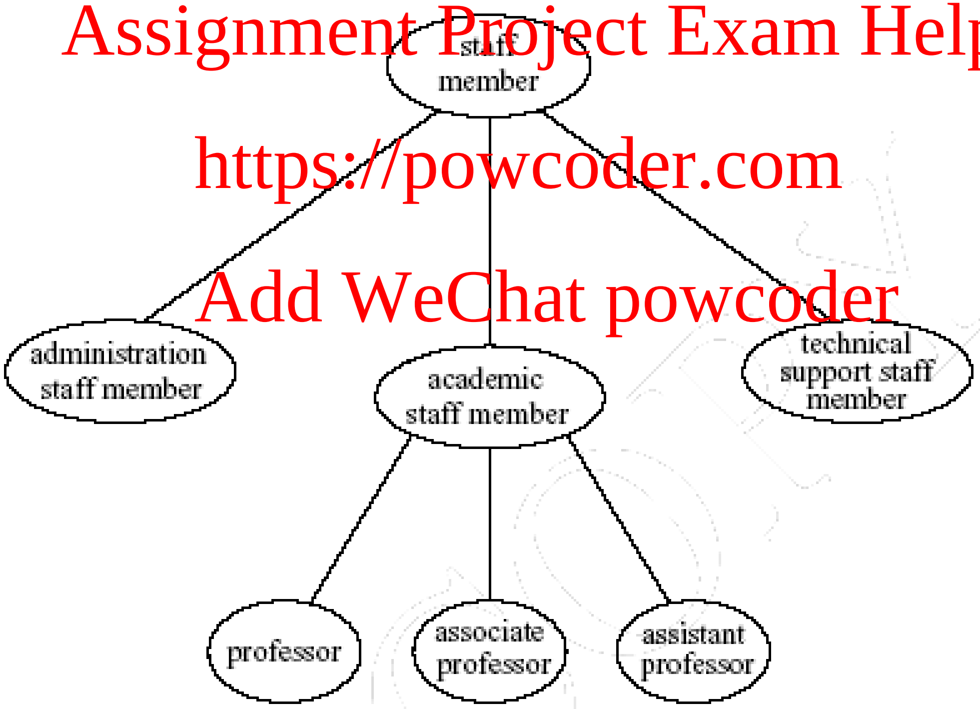
- Discrete Maths is taught by Concrete Maths
  - We want courses to be taught by lecturers only
  - Restriction on values of the property "is taught by" (**range restriction**)

- Room MZH5760 is taught by David Billington
  - Only courses can be taught
  - This imposes a restriction on the objects to which the property can be applied (**domain restriction**)

Class Hierarchies

- Classes can be organised in hierarchies
  - A is a subclass of B if every instance of A is also an instance of B
  - Then B is a superclass of A
- A subclass graph **does not have to** be a tree
- A class may have **multiple superclasses**

Class Hierarchy Example



Inheritance in Class Hierarchies

- Range restriction: **Courses must be taught by academic staff members only**
- Michael Maher is a professor
- He inherits the ability to teach from the class of academic staff members
- This is done in RDF Schema by fixing the semantics of “is a subclass of”

Property Hierarchies

- Hierarchical relationships for properties
  - E.g., “is taught by” is a subproperty of “involves”
  - If a course C is taught by an academic staff member A, then C also involves A

- The converse is not necessarily true

  - E.g., A may be the teacher of the course C, or
  - a tutor who marks student homework but does not teach C

- P is a subproperty of Q, if Q(x,y) is true whenever P(x,y) is true. Hence in the above example, “is taught by” is subproerty of “involve”

RDF Layer vs RDF Schema Layer

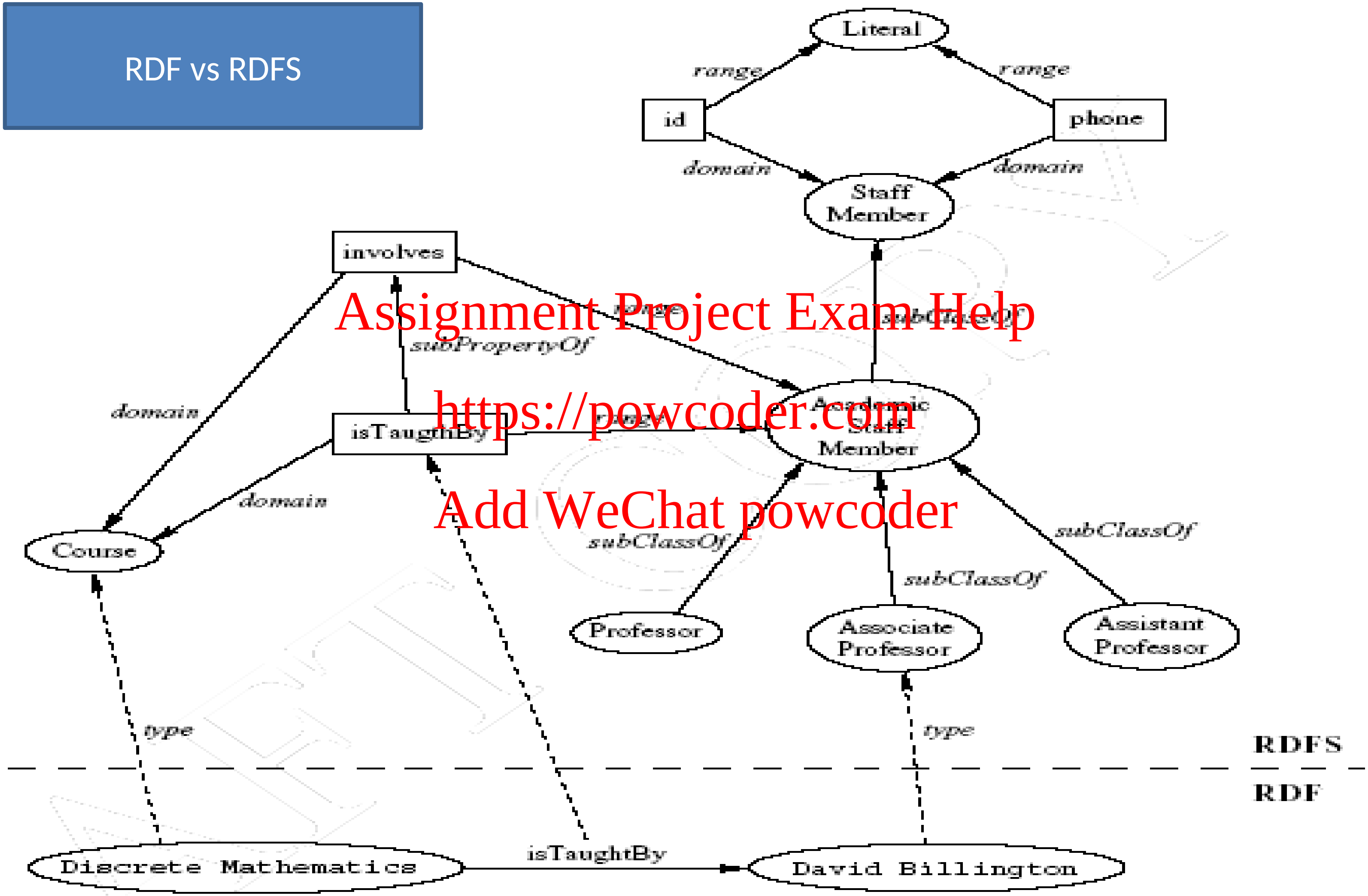
- Discrete Mathematics is taught by David Billington

- The schema is itself written in a formal language, RDF Schema, that can express its ingredients:

  – subClassOf, Class, Property, subPropertyOf, Resource, etc.

RDF Layer vs RDF Schema Layer (2)

RDF vs RDFS



Lecture Outline

1. Basic Concepts of RDF Schema
2. **The Language of RDF Schema**
3. Axiomatic Semantics for RDF and RDFS
4. Direct Semantics based on Inference Rules

RDF Schema in RDF

- The modeling primitives of RDF Schema are defined using resources and properties (RDF itself is used!)
- To declare that “lecturer” is a subclass of “academic staff member”
  - Define resources **lecturer**, **academicStaffMember**, and **subClassOf**
  - define property **subClassOf**
  - Write triple (**lecturer**,**subClassOf**,**academicStaffMember**)
- We use the XML-based syntax of RDF

Core Classes

- **rdfs:Resource**, the class of all resources
- **rdfs:Class**, the class of all classes
- **rdfs:Literal**, the class of all literals (strings)
- **rdf:Property**, the class of all properties.
- **rdf:Statement**, the class of all reified statements

Core Properties

- **rdf:type**, which relates a resource to its class
  - The resource is declared to be an instance of that class
- **rdfs:subClassOf**, which relates a class to one of its superclasses
  - All instances of a class are instances of its superclass
- **rdfs:subPropertyOf**, relates a property to one of its superproperties

Core Properties (2)

- **rdfs:domain**, which specifies the domain of a property P
  - The class of those resources that may appear as subjects in a triple with predicate P
  - If the domain is not specified, then any resource can be the subject
- **rdfs:range**, which specifies the range of a property P
  - The class of those resources that may appear as values in a triple with predicate P

Examples

```
<rdfs:Class rdf:about="#lecturer">
    <rdfs:subClassOf rdf:resource="#staffMember"/>
</rdfs:Class>

<rdf:Property rdf:ID="phone">
        <rdfs:domain rdf:resource="#staffMember"/>
        <rdfs:range rdf:resource="http://www.w3.org/
            2000/01/rdf-schema#Literal"/>
</rdf:Property>
```

Relationships Between Core Classes and Properties

- **rdfs:subClassOf** and **rdfs:subPropertyOf** are transitive, by definition
- ***rdfs:Class*** *is a subclass of* ***rdfs:Resource***
  - Because every class is a resource
- **rdfs:Resource** is an instance of **rdfs:Class**
  - Because rdfs:Resource is the class of all resources, so it is a class
- Every class is an instance of **rdfs:Class**
  - For the same reason

Reification and Containers

- **rdf:subject**, relates a reified statement to its subject
- **rdf:predicate**, relates a reified statement to its predicate
- **rdf:object**, relates a reified statement to its object
- **rdf:Bag**, the class of bags
- **rdf:Seq**, the class of sequences
- **rdf:Alt**, the class of alternatives
- **rdfs:Container**, which is a superclass of all container classes, including the three above

Utility Properties

- **rdfs:seeAlso** relates a resource to another resource that explains it
- **rdfs:isDefinedBy** is a subproperty of **rdfs:seeAlso** and relates a resource to the place where its definition, typically an RDF schema.
- **rdfs:comment**. Comments, typically longer text, can be associated with a resource
- **rdfs:label**. A human-friendly label (name) is associated with a resource

Example: A University

```
<rdfs:Class rdf:ID="lecturer">
        <rdfs:comment>
           The class of lecturers. All lecturers are academic staff members.
        </rdfs:comment>
        <rdfs:subClassOf rdf:resource="#academicStaffMember"/>
</rdfs:Class>
```


Note to the role of TWO Simplification rules mentioned in previous lectures:

I. Childless property elements within description elements may be replaced by XML attributes
II. For description elements with a typing element we can use the name specified in the rdf:type element instead of rdf:Description

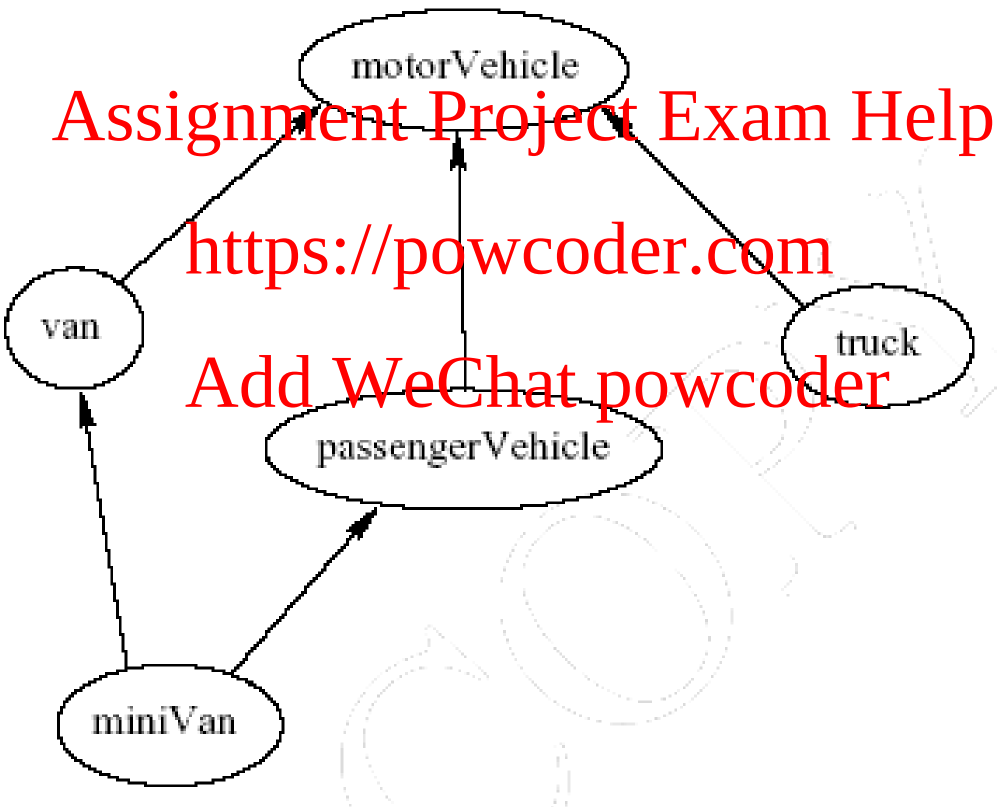
Example: A University (2)

```
<rdfs:Class rdf:ID="course">
    <rdfs:comment>The class of courses</rdfs:comment>
</rdfs:Class>
<rdf:Property rdf:ID="isTaughtBy">
    <rdfs:comment>
        Inherits its domain ("course") and range ("lecturer")
        from its superproperty "involves"
    </rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="#involves"/>
</rdf:Property>
```

Example: A University (3)

```
<rdf:Property rdf:ID="phone">
    <rdfs:comment>
        It is a property of staff members
        and takes literals as values.
    </rdfs:comment>
    <rdfs:domain rdf:resource="#staffMember"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
    schema#Literal"/>
</rdf:Property>
```

Class Hierarchy for the Motor Vehicles Example



https://www.youtube.com/watch?v=iuQrBf2Oq-E&t=106s&ab_channel=Ontotext

Lecture Outline

1. Basic Concepts of RDF Schema
2. The Language of RDF Schema
3. **Axiomatic Semantics for RDF and RDFS**
4. Direct Semantics based on Inference Rules

First order logic, also known as predicate logic, is a collection of formal systems used in mathematics, philosophy, linguistics, and **computer science**. First-order logic uses quantified variables over non-logical objects and allows the use of **sentences that contain variables**. In first-order logic, **there are predicates having predicates or functions as arguments**, or in which one or both of predicate quantifiers or function quantifiers are permitted.

The quantifier symbols used in First order logic are

∀ and ∃

, and the logical connectives used include ∧ for conjunction, ∨ for disjunction, → for implication, ↔ for biconditional, ¬ for negation.

Source: https://en.wikipedia.org/wiki/First-order_logic

First Order Logic

- https://www.youtube.com/watch?v=s7OMfvbg3gY&ab_channel=ArtificialIntelligence-AllinOne

Axiomatic Semantics is for the following three purposes:

I. Formalizing the meaning of the modeling primitives of RDF and RDF Schema by translating them into **first-order logic**

II. Making the semantics unambiguous and machine accessible

III. Providing a basis for reasoning support by automated reasoners manipulating logical formulas

The Approach

- All language primitives in RDF and RDF Schema are represented by constants:
  - **Resource**, **Class**, **Property**, **subClassOf**, etc.
- A few predefined predicates are used as a foundation for expressing relationships between the constants
- We use predicate logic with equality
- Variable names begin with ?

An Auxiliary Axiomatization of Lists

- Function symbols:
  - **nil** (empty list)
  - **cons(x,l)** (adds an element to the front of the list)
  - **first(l)** (returns the first element)
  - **rest(l)** (returns the rest of the list)
- Predicate symbols:
  - **item(x,l)** (tests if an element occurs in the list)
  - **list(l)** (tests whether l is a list)
- Lists are used to represent containers in RDF

Basic Predicates

- **PropVal(P,R,V)**
  - A predicate with 3 arguments, which is used to represent an RDF statement with resource **R**, property **P** and value **V**
  - An RDF statement (triple) **(P,R,V)** is represented as **PropVal(P,R,V)**.

- **Type(R,T)**
  - Short for **PropVal(type,R,T)**
  - Specifies that the resource **R** has the type **T**
- **Type(?r,?t) ↔ PropVal(type,?r,?t)**

RDF Classes

- Constants: **Class**, **Resource**, **Property**, **Literal**
  - All classes are instances of **Class**

Type(Class,Class)

Type(Resource,Class)

Type(Property,Class)

Type(Literal,Class)

↔ : Logical symbol representing iff called **biconditiional**

if and only if; iff; means the same as

In other words, either both right and left side statements are correct or both are wrong.

→: **Implication symbol**

If left-side statement is correct, then the right-side statement is also correct

RDF Classes (2)

- **Resource** is the most general class: every class and every property is a resource

Type(?p,Property) → Type(?p,Resource)

Type(?c,Class) → Type(?c,Resource)

- The predicate in an RDF statement must be a property
- **PropVal(?p,?r,?v) → Type(?p,Property)**

The type Property

- **type** is a property

PropVal(type,type,Property)

(Note that the three parameters of PropVal are (Predicate ,Resource , and Value))

- **type** can be applied to resources (domain) and has a class as its value (range)

Type(?r,?c) → (Type(?r,Resource) ∧ Type(?c,Class))

The Auxiliary FuncProp Property

- **P** is a functional property if, and only if,
  - it is a property, and
  - there are no **x, y1** and **y2** with **P(x,y1), P(x,y2)** and **y1≠y2**

Type(?p, FuncProp) ↔

(Type(?p, Property) ∧ ∀?r ∀?v1 ∀?v2

(PropVal(?p,?r,?v1) ∧

PropVal(?p,?r,?v2) → ?v1 = ?v2))

Containers

- Containers are lists:

Type(?c,Container) → list(?c)

- Containers are bags or sequences or alternatives:

Type(?c,Container) ↔

(Type(?c,Bag) ∨ Type(?c,Seq) ∨ Type(?c,Alt))

- Bags and sequences are disjoint:

¬(Type(?x,Bag) ∧ Type(?x,Seq))

Subclass

- **subClassOf** is a property:

Type(subClassOf,Property)

- If a class C is a subclass of a class C', then all instances of C are also instances of C':

PropVal(subClassOf,?c,?c') ↔

(Type(?c,Class) ∧ Type(?c',Class) ∧

∀?x (Type(?x,?c) → Type(?x,?c')))

By using implication symbol, write an implication formula for:

If the domain of **P** is **D**, then for every **P**(x,y), x∈**D**

Hint Complete the following:

PropVal(domain,?p,?d) →

ANSWER:

$$\mathrm{PropVal}(\mathrm{domain},?p,?d) \rightarrow$$

$$\forall ?x\ \forall ?y\ (\mathrm{PropVal}(?p,?x,?y) \rightarrow \mathrm{Type}(?x,?d))$$

By using implication symbol, write an implication formula for:

If the range of P is R, then for every P(x,y), y∈R

Hint : Complete the following

PropVal(range,?p,?r) →

ANSWER:

$$\mathbf{PropVal(range,?p,?r)} \rightarrow$$
$$\forall ?x\ \forall ?y\ (\mathbf{PropVal(?p,?x,?y)} \rightarrow \mathbf{Type(?y,?r)})$$

By using biconditional symbol, write a biconditional formula for:

P is a subproperty of P', if P'(x,y) is true whenever P(x,y) is true:

Note that P and P‘ both should be of type Property.

Hint: Complete the following:

PropVal(subPropertyOf,?p,?p') $\leftrightarrow$

ANSWER

PropVal(subPropertyOf,?p,?p') ↔

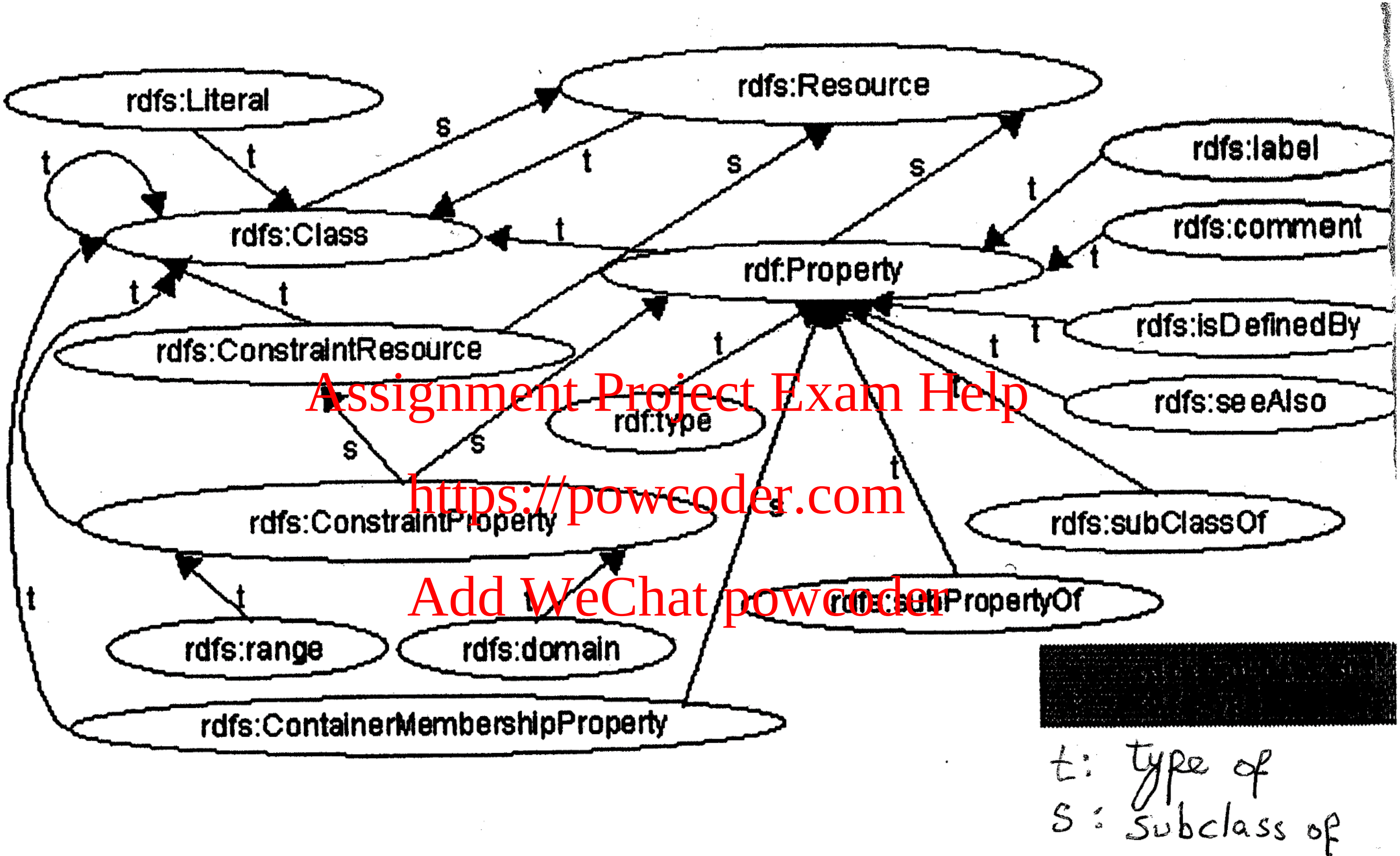
(Type(?p,Property) ∧ Type(?p',Property) ∧

∀?r ∀?v (PropVal(?p,?r,?v) →

PropVal(?p',?r,?v)))

Lecture Outline

1. Basic Concepts of RDF Schema
2. The Language of RDF Schema
3. Axiomatic Semantics for RDF and RDFS
4. **Direct Semantics based on Inference Rules**



Class Hierarchy for RDF Schema (Source: Brickley & Guha 1999)

Semantics based on Inference Rules

- Semantics in terms of RDF triples **instead of** restating RDF in terms of **first-order logic**
- This inference system consists of inference rules of the form:

IF E contains certain triples

THEN add to E certain additional triples

- where **E** is an arbitrary set of RDF triples

Examples of Inference Rules

IF E contains the triple (?x,?p,?y)
THEN E also contains (?p,rdf:type,rdf:property)

IF E contains the triples (?u,rdfs:subClassOf,?v) and (?v,rdfs:subclassOf,?w)
THEN E also contains the triple (?u,rdfs:subClassOf,?w)

IF E contains the triples (?x,rdf:type,?u) and (?u,rdfs:subClassOf,?v)
THEN E also contains the triple (?x,rdf:type,?v)

Lab Exercise 3

- Choose 10 (optional) queries from the following link, read the semantics of those queries and execute them at DbPedia endpoint http://dbpedia.org/sparql using Apache Jena (use Lecture5.java uploaded on Moodle, Week 5).

- https://aifb-ls3-kos.aifb.kit.edu/projects/spartiqulator/examples.htm

- Submit the results of the queries (before the next lab) as a single Word/PDF document on Moodle. You can limit/reduce long results.