RDF & RDFS
Basics, Semantics, Schema, Vocabulary, and Usage

BASICS

Syntax

URI Prefix: Shorthand for namespace URI (written as: @prefix dbpedia: <http://dbpedia.org/resource/>)

Subject — Property/Predicate — Object

dbpedia:TheHague dbo:areaCode "015"^^xsd:double .

'Type' operator

dbpedia:The_Hague rdfs:label "The Hague"@en ,
dbpedia:The_Hague rdfs:label "Den Haag"@nl .

If unspecified, 'string' will be assumed by RDF

A full usage example is here

Short form

dbpedia:The_Hague dbo:areaCode "015"@en ;
rdfs:label "The Hague"@en ,
"Den Haag"@nl .

Tag: Provides further information about literals

1

Possible Components

An RDF graph is a set of triples like this.

Triples can be made up of...

	Subject	Property/Predicate	Object
URI References	✔	✔	✔
Literals	✖	✖	✔
Blank Nodes	✔	✖	✔

This creates "hypergraphs"—another dimension that connects to predicate and defines it (i.e., the resource page of the predicate itself, and the connections to it, etc). This is an important feature of RDF.

A blank node as a predicate would be just too meaningless; the relation between two things should be somehow specified.

Literals as subjects is not officially allowed in RDF specification, but it could arguably still be possible/make sense in some cases. E.g., "10 isA number"

Formal definition of the syntax of RDF:
RDF = (URI ∪ blankNodes) X (URI) X (URI ∪ blankNodes ∪ literals)

2

Example RDF File

```
@prefix vcard: <http://www.w3.org/2006/vcard/ns#>
@prefix sn: <http://www.snee.com/hr/>

sn:employee1 vcard:given-name "Heidi" .
sn:employee2 vcard:given-name "John" .
sn:employee3 vcard:given-name "Francis" .
sn:employee4 vcard:given-name "Jane" .

sn:employee1 vcard:family-name "Smith" .
sn:employee2 vcard:family-name "Smith" .
sn:employee3 vcard:family-name "Jones" .
sn:employee4 vcard:family-name "Berger" .

sn:employee1 vcard:title "CEO" .
sn:employee2 vcard:title "Engineer" .
sn:employee3 vcard:title "Vice President" .
sn:employee4 vcard:title "Sales" .

sn:employee1 sn:hiring-date "2018-01-13" .
sn:employee2 sn:hiring-date "2018-01-28" .
sn:employee3 sn:hiring-date "2018-02-13" .
sn:employee4 sn:hiring-date "2018-03-10" .

sn:employee1 vcard:orientation-date "2018-01-30" .
sn:employee2 vcard:orientation-date "2018-01-30" .
```

Official w3.org vocabulary

User-created, custom vocabulary

Full expression: <http://www.w3.org/2006/vcard/ns#given-name>

Assigning a prefix at the beginning of file allows writing a shorthand as: vcard:given-name

Further RDF syntax specifications in the dedicated 'Turtle Syntax' sheet.

3

From Tables and Trees to Triples

Any information type can be transformed into triples

	Subject	Property	Object
Tables	row id	column name	cell
Trees	parent	path	child

Tree: Parent (Parent object: Subject) —has→ Child 1, —has→ Child 2 (Path: Property; Child object: Object)

Column title: Property

Employee ID	given-name	family-name	title	hiring-date	orientation-date
employee1	Heidi	Smith	CEO	2018-01-13	2015-01-30
employee2	John	Smith	Engineer	2018-01-28	2015-01-30
employee3	Francis	Jones	Vice President	2018-02-13	
employee4	Jane	Berger	Sales	2018-03-10	

Row identifier: Subject

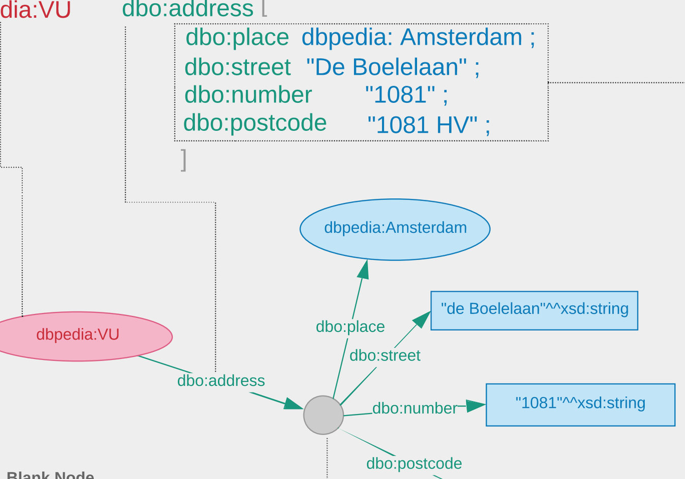
Cell: Object

sn:employee3 vcard:title "Vice President" .

4

Blank Nodes

```
dbpedia:VU dbo:address [
    dbo:place dbpedia: Amsterdam ;
    dbo:street "De Boelelaan" ;
    dbo:number "1081" ;
    dbo:postcode "1081 HV" ;
]
```



Blank Node
An object that has no URI, and is used in a similar fashion to the connector element of UML schemas. It serves as a terminal or intermediary between two or more elements.

Use them when resource is unknown, or has no natural identifier—when we cannot/don't want to give a name to an object

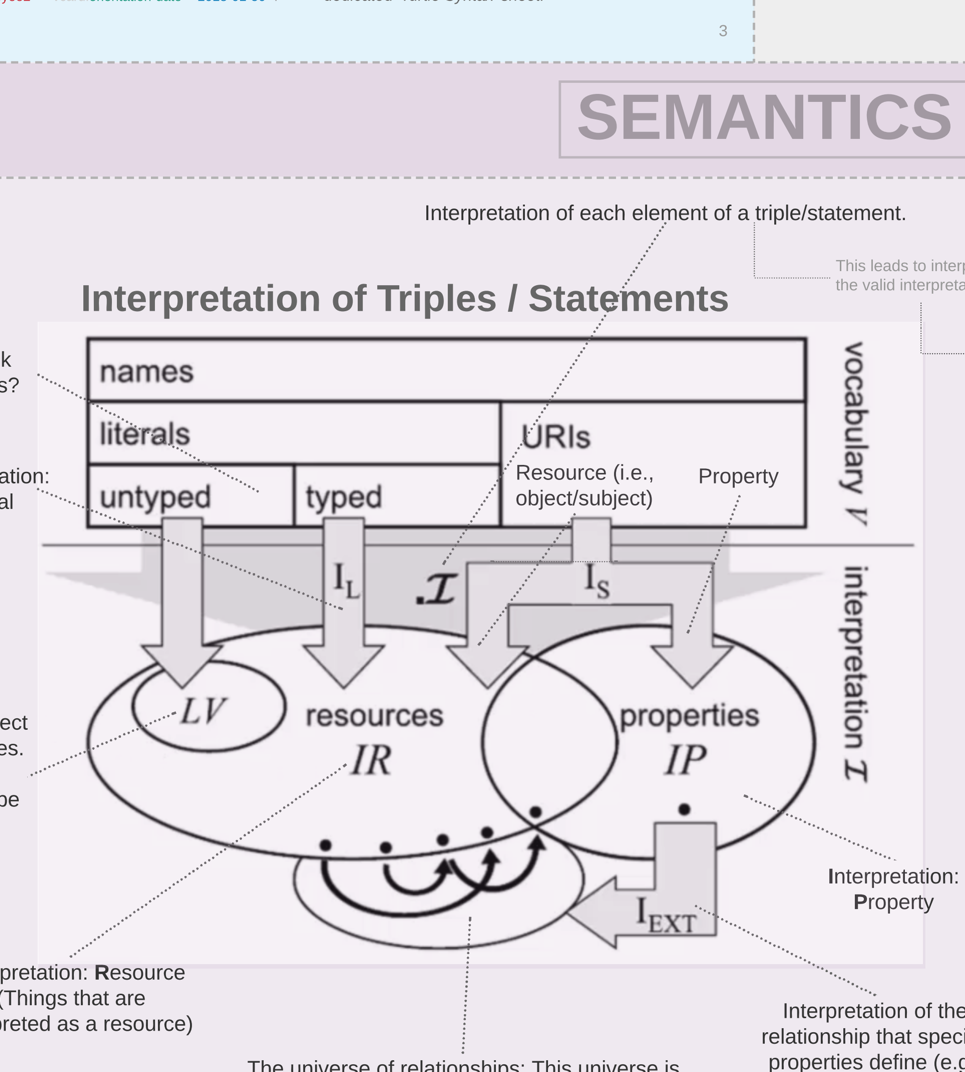
5

SEMANTICS

Interpretation of each element of a triple/statement.

This leads to interpretations of a knowledge graph, and the valid interpretations are used to model the statement.

Interpretation of Triples / Statements



Blank nodes?

Interpretation: Literal

Non-object resources. These tend to be literals.

Interpretation: Resource (Things that are interpreted as a resource)

The universe of relationships: This universe is an intersection of the objects from the IR universe whose relationships are defined by the IP universe, and the IP universe.

Interpretation: Property

Interpretation of the relationship that specific properties define (e.g., transitive relationship, symmetrical relationship, range, etc...).

This specificity of properties is not the case in Simple Knowledge Graph Logic (SKGL); but is the case in RDF. SKGL creates graphs while RDF creates Hypergraphs—graphs in which properties are also linked as objects and subjects while their properties is being defined.

Interpretation and Models

A model of a graph is an interpretation that satisfies all its triples.

Blank Nodes and Entailment

Modeling blank nodes with grounding

'**Grounding**' is the equivalent of value assignment in arithmetics (i.e., giving a value to x). Just like it is the case in arithmetics, entailment can only be proven if values are assigned to variables.

Ungrounded graphs (~Uninstantiated) — Grounded graphs (~Instantiated)

Model: VU —has→ ⊗ —place→ Amsterdam, —street→ De Boelelaan

Model: Amsterdam —has→ ⊗ —latitude→ 52.3702° N, —longitude→ 4.8952° E

Entailment:

largeGraph ⊧ smallGraph if and only if largeGraph ⊧ smallGraph(grounded) is true for at least in one grounded instance of the smallGraph

i.e., Models of graphs is created by assigning URIs to blank nodes, and thus by grounding them. After grounding, if all models of largeGraph is also a model of at least one grounding of smallGraph, then largeGraph entails smallGraph.

i.e., Nothing in the smallGraph (i.e., a version of it that has its variables replaced by values) violates anything in the largeGraph; and everything that is true in the largeGraph is true for the smallGraph.

~Ungrounded: lion as species
~Grounded: Lion as specimen

Analogous to saying that nothing that applies to a specimen of a lion violates anything that applies to mammals; and everything that is true for mammals is true for (at least) this sample specimen, and thus, it must be true for mammals. Mammals entail this lion specimen; thus, mammals entail lions.

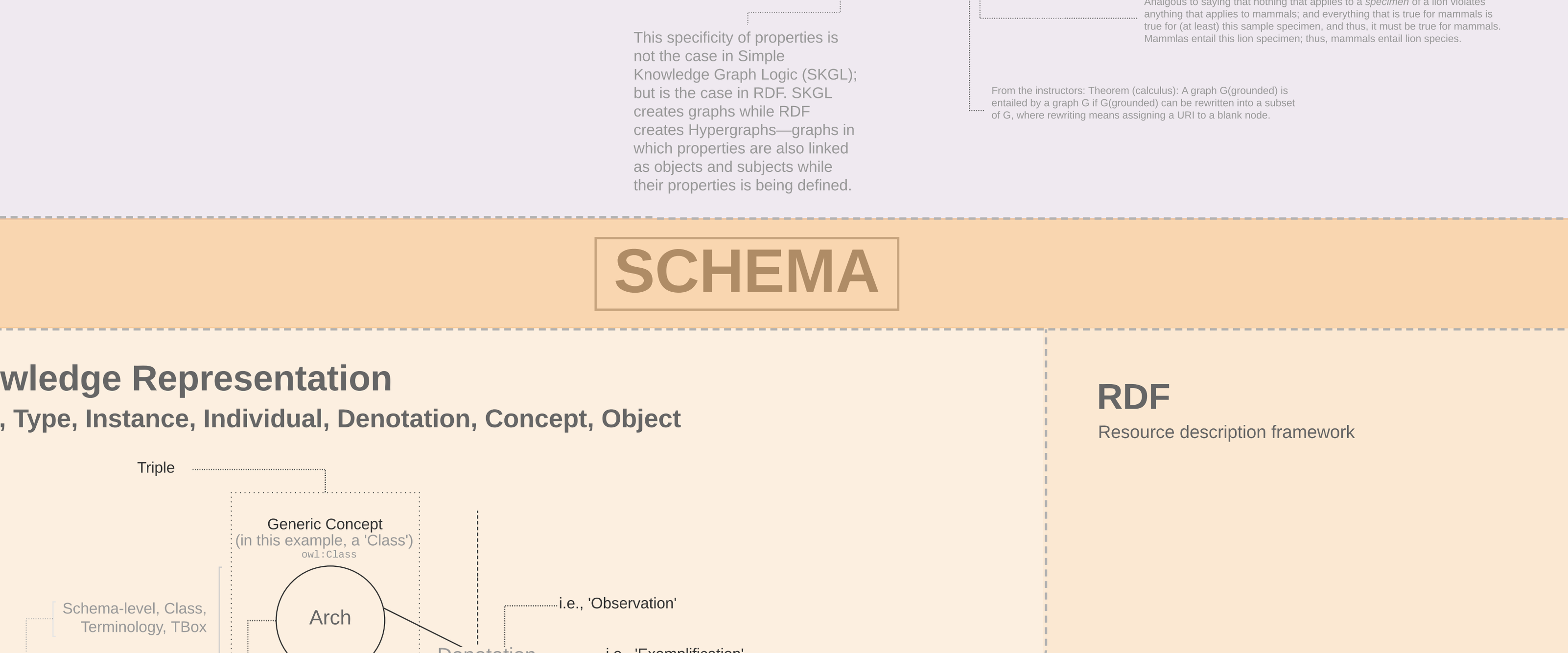
From the instructors: Theorem (calculus): A graph G(grounded) is entailed by a graph G if G(grounded) can be rewritten into a subset of G, where rewriting means assigning a URI to a blank node.

6

SCHEMA

Knowledge Representation

Class, Type, Instance, Individual, Denotation, Concept, Object



7

RDF

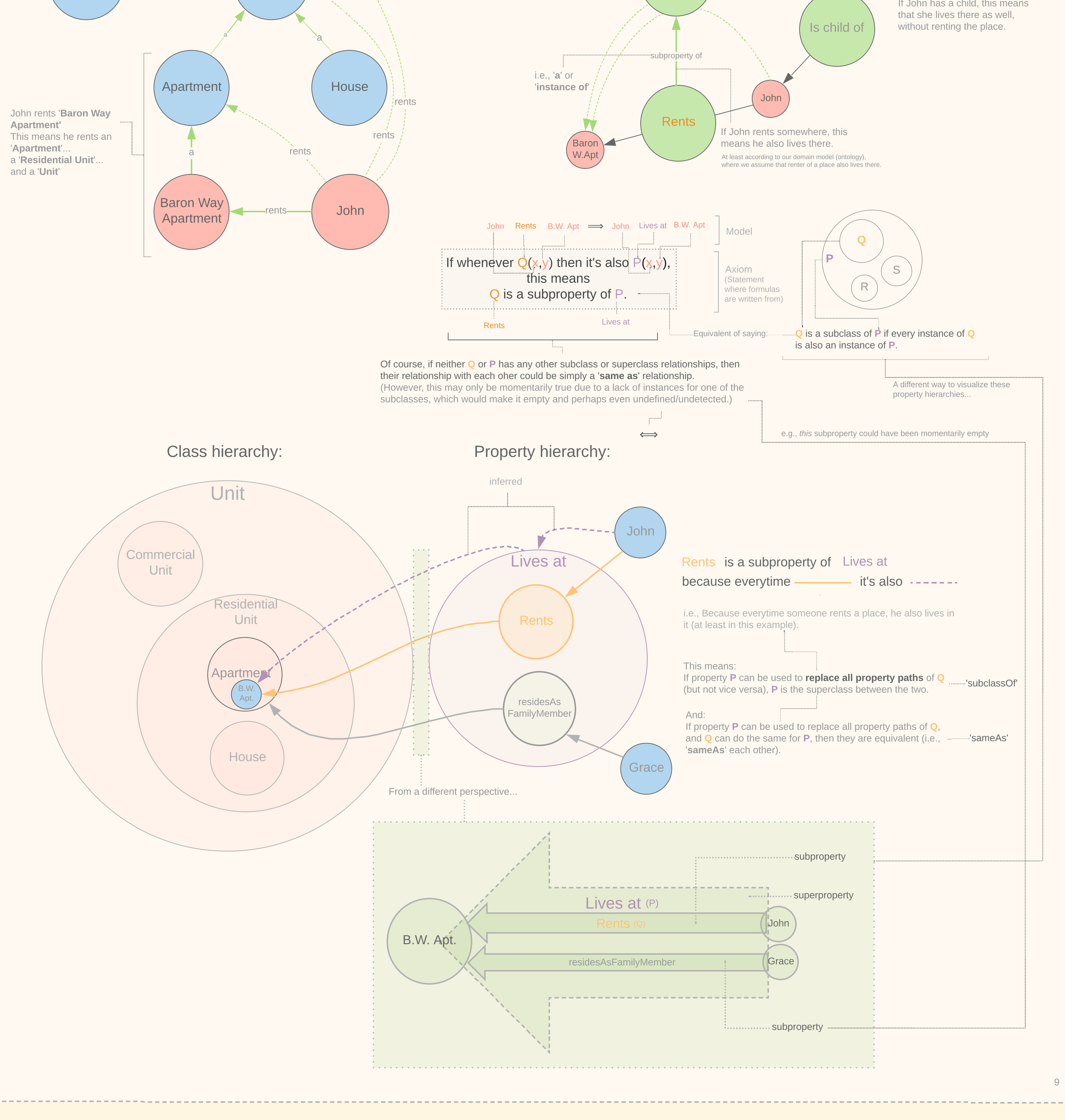
Resource description framework

8

RDFS (RDF Schema)

RDFS is a basic ontology for formally defining relationships in RDF. It is more limited, but also more efficient (e.g., when calculating entailments) than full-fledged ontology languages such as OWL.

Class Hierarchy / Property Hierarchy



Rents is a subproperty of Lives at because everytime ——— it's also - - - - -

i.e., Because everytime someone rents a place, he also lives in it (at least in this example).

This means:
If property P can be used to replace all property paths of Q (but not vice versa), P is the superclass between the two. — 'subclassOf'

And:
If property P can be used to replace all property paths of Q, and Q can do the same for P, then they are equivalent (i.e., 'sameAs' each other). — 'sameAs'

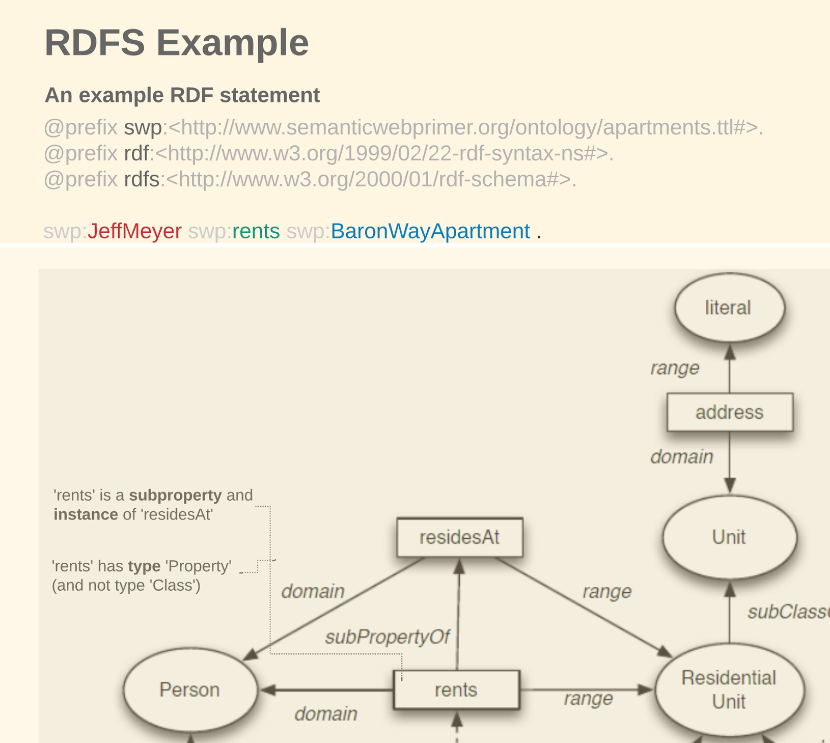
9

RDFS Example

An example RDF statement

@prefix swp:<http://www.semanticwebprimer.org/ontology/apartments.ttl#>.
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.

swp:JeffMeyer swp:rents swp:BaronWayApartment .



'House' is an **instance** and **subclass** of 'Residential Unit'

'Jeff Meyer' or 'Baron Way Aparment' is an **instance** (but **not a subclass**) of 'Person'

An example RDFS

```
@prefix swp:<http://www.semanticwebprimer.org/ontology/apartments.ttl#>.
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.

swp:Person            rdf:type           rdfs:Class .
swp:Person            rdfs:comment       "The class of people" .

swp:Unit              rdf:type           rdfs:Class .
swp:Unit              rdfs:comment       "A self contained section of accommodations in a larger building or group of buildings." .

swp:ResidentialUnit   rdf:type           rdfs:Class .
swp:ResidentialUnit   rdfs:subClassOf    swp:Unit .
swp:ResidentialUnit   rdfs:comment       "The class of all units or places where people live." .

swp:Apartment         rdf:type           rdfs:Class .
swp:Apartment         rdfs:subClassOf    swp:ResidentialUnit .
swp:Apartment         rdfs:comment       "The class of apartments" .

swp:House             rdf:type           rdfs:Class .
swp:House             rdfs:subClassOf    swp:ResidentialUnit .
swp:House             rdfs:comment       "The class of houses" .

swp:residesAt         rdf:type           rdfs:Property .
swp:residesAt         rdfs:comment       "Relates persons to their residence" .
swp:residesAt         rdfs:domain        swp:Person .
swp:residesAt         rdfs:range         swp:ResidentialUnit .

swp:rents             rdf:type           rdfs:Property .
swp:rents             rdfs:comment       "It inherits its domain (swp:Person) and range (swp:ResidentialUnit) from its superproperty (swp:residesAt)." .

swp:rents             rdfs:subPropertyOf swp:residesAt .

swp:address           rdf:type           rdfs:Property .
swp:address           rdfs:comment       "Is a property of units and takes literals as its value" .
swp:address           rdfs:domain        swp:Unit .
swp:address           rdfs:range         rdf:Literal .
```

'House' has **type** 'Class' and is a **subclassOf** 'Residential Unit'

'**type**' property defines the relationship between **class** and **instance**.

'**subclass**' property defines the relationship between **class** and **superclass** (and not between class and instance).

Example in Action: Class-based (weak) Inference with RDFS

Turtle File:

```
@prefix fb: <https://graph.facebook.com/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

fb:594486635        fb:likes      fb:289482145973 , fb:109327925759703.

fb:109327925759703  fb:name       "Waking Life"@en ;
                    fb:category   fb:movie .

fb:32985985640      fb:name       "Zeitgeist"@en ;
                    fb:category   fb:community .

fb:movie            rdfs:label    "Movie"@en .
fb:community        rdfs:label    "Community"@en .

fb:community        rdfs:subClassOf    fb:movie .
fb:category         rdfs:subPropertyOf rdf:type .
```

SPARQL Query:

```
PREFIX fb: <https://graph.facebook.com/>

SELECT ?title WHERE {
    ?movie a        fb:movie ;
           fb:name  ?title .
}
```

Query Result:

?title
"Waking Life"@en
"Zeitgeist"@en

This triple allowed Zeitgeist to be inferred as a movie (as it specified that each community is also a movie). Without this line, only 'Waking Life' would have returned.

10

VOCABULARY

RDF, RDFS and Other Vocabularies

Resource Description Framework
Resource Description Framework Schema

prefix rdf <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix rdfs <http://www.w3.org/2000/01/rdf-schema#>

RDF/RDFS Classes

Core Classes	Container Classes
rdfs:Resource	rdf:Bag
rdfs:Class	rdf:Seq
rdfs:Literal	rdf:Alt
rdfs:Property	rdfs:Container
rdfs:Statement	
rdfs:Datatype	

RDF/RDFS Properties

Defining Relationships	Restricting Properties	Utility	Reification
rdf:type	rdfs:domain	rdfs:seeAlso	rdf:subject
rdfs:subClassOf	rdfs:range	rdfs:isDefinedBy	rdf:predicate
rdfs:subPropertyOf		rdfs:comment	rdf:object
		rdfs:label	

geo:Amsterdam rdf:type rdf:Resource .
geo:containedIn a rdf:Property .
rdf:type a rdf:Property .

FOAF
Friend of a Friend
prefix foaf: <http://xmlns.com/foaf/0.1/>

Important Elements
foaf:Person
foaf:name
foaf:mbox
foaf:knows

vu:Rinke_Hoekstra rdf:type foaf:Person ;
foaf:knows vu:Albert_Merono ;
foaf:name "Rinke Hoekstra" .

DC/DCT
Dublin Core / Dublin Core Terms
prefix dct: <http://purl.org/dc/terms>

Important Elements
dct:creator
dct:created
dct:description
dct:title

vu:SWPrimer dct:creator vu:Rinke_Hoekstra ;
dct:created "2012"^^xsd:date,
dct:title "A Semantic Web ..." .

SKOS
Simple Knowledge Organization System
prefix skos <http://www.w3.org/2004/02/skos/core#>

Important Elements
skos:exactMatch

Good for using things that are *almost* the same. This is an informal way of establishing sameness.

fb:594486635 fb:likes fb:105638652803531 .
fb:105638652803531 skos:exactMatch imdb:tt0120669 .

For common prefixes and their namespaces: http://prefix.cc

Social Side of Semantics

Berners-Lee: Using shared vocabularies is the key to establishing consistent semantics.

URI's may not always be unique. They are not a global identification service, but simply strings.

Rule of Thumb: "Only create vocabularies/URIs in a namespace within your own domain." (Important for trust).

Example in Action: Linking Vocabularies

Turtle File:

```
@prefix fb: <https://graph.facebook.com/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

fb:594486635        fb:likes      fb:289482145973 , fb:109327925759703.

fb:109327925759703  fb:name       "Waking Life"@en ;
                    fb:category   fb:movie .

fb:32985985640      fb:name       "Zeitgeist"@en ;
                    fb:category   fb:community .

fb:movie            rdfs:label    "Movie"@en .
fb:community        rdfs:label    "Community"@en .

fb:community        rdfs:subClassOf    fb:movie .
fb:category         rdfs:subPropertyOf rdf:type .
```

This is Facebook's way of definig a 'type' relationship. It is used instead of 'rdf:type'.

This is triple aligns Facebook and RDF schema by bringing Facebook's 'category' property in line with 'type' property of of RDF.

Without this triple, results of the example SPARQL would have been empty, as there would not be any 'rdf:type' properties in the Turtle file. However, now that 'fb:category' is related to 'rdf:type', asking for 'rdf:type' in SPARQL also returns all categories.

SPARQL Query:

```
PREFIX fb: <https://graph.facebook.com/>

SELECT ?title WHERE {
    ?movie a        fb:movie ;
           fb:name  ?title .
}
```

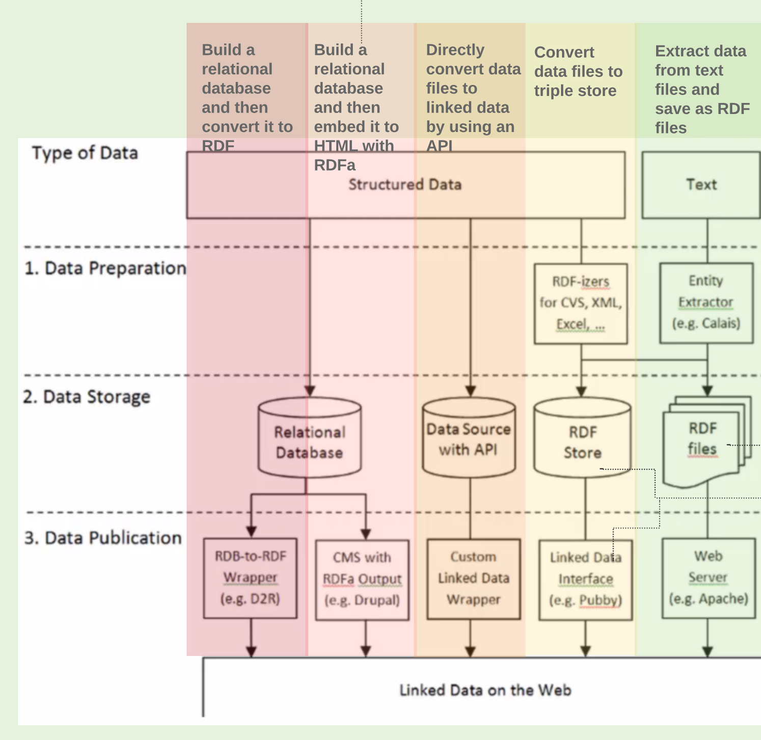
Query Result:

?title
"Waking Life"@en
"Zeitgeist"@en

11

USE CASES

Accessing and Publishing RDF & Linked Data



File publishing
This does not allow queries if not first serialized

Triple store publishing
This allows users to **query** with a query API like a SPARQL endpoint

12