RDF and RDF Schema

- Basic Concepts of RDF Schema
- Core Classes of RDFs
- Core properties of RDFs

Basic Concept of RDF

- RDF is a universal language that lets users describe resources in their own vocabularies

- But it does not give any special meaning (**semantics**) to vocabulary such as subClassOf or type.

- RDFS describes the **vocabulary** of the RDF data model.
- The user can do so in RDF Schema using:
  - Classes and Properties
  - Class Hierarchies and Inheritance
  - Property Hierarchies

RDF Schema (RDFS)

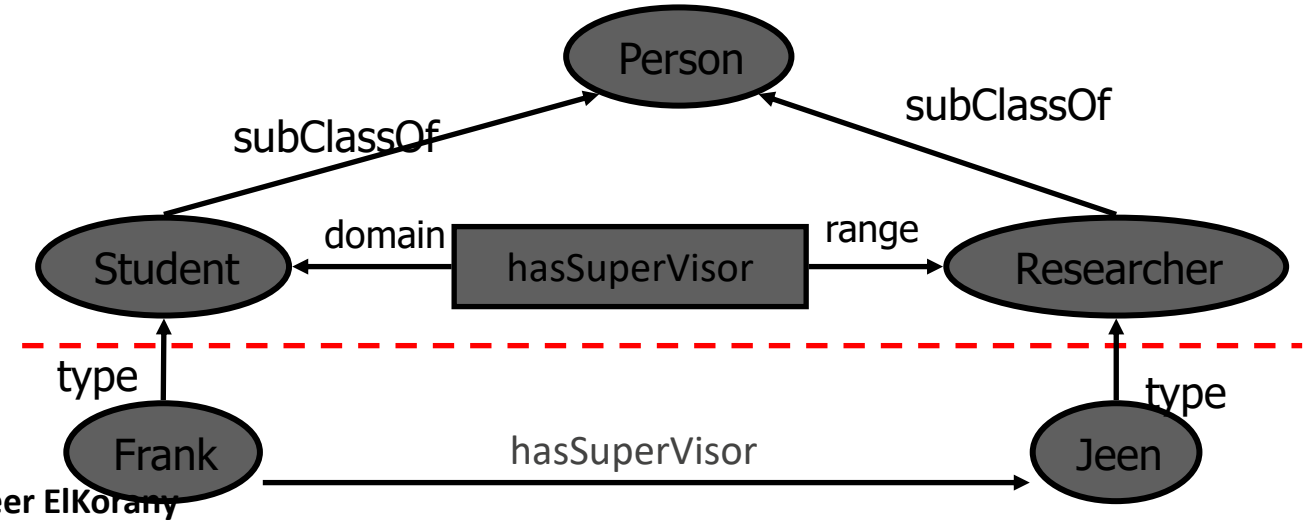
- RDFS describes the **vocabulary** of the RDF data model.
- The equivalent to the database schema.
- **Vocabulary == ontology!**

Classes and their Instances

- We must distinguish between
  - Sets of individuals sharing properties called **classes**: lecturers, students, courses etc.
    - These are described in **RDFS**
  - Individual **objects** in the domain: Discrete Maths, Abeer Mohamed etc.
    - These are described in **RDF**

RDF Schema

- Describe vocabulary for RDF:
  - Class, subClassOf, type
  - Property, subPropertyOf
  - domain, range
- Vocabulary can be used to define other vocabularies for your application domain



RDF Schema (RDFS)

- The class and property concepts in RDF are similar to the type systems of object-oriented programming languages such as Java.
- In object-oriented systems a class is usually defined in terms of the properties its instances may have.
- In the RDF vocabulary properties are defined in terms of the classes of resource to which they apply.
- This is the role of the domain and range mechanisms.
- For example, we could define the eg:Author-by property to have a domain of eg:Article and a range of eg:Person.

RDF Schema

- **RDF Schema**
  - provides the framework to define application-specific classes and properties.
  - allows resource to be defined as instances of classes
  - provides facility to define the ***subclass*** relationship of classes and properties.
- **RDF** describes rescources with subject, predicate and object (class, property and value)

RDF Schema (language for simple ontologies)

RDF schema is a semantic extension of RDF used for simple ontologies' design. The RDF schema language is used for declaring basic class and types when describing the terms used in RDF and are used to determine characteristics of other resources, such as the domains and ranges of properties.

- rdfs:Resource
- rdfs:Class
- rdfs:domain
- rdfs:range
- rdfs:subClassOf
- rdfs:subPropertyOf
- …

RDF Schema provides a data-modelling vocabulary for RDF data.

http://www.w3.org/TR/rdf-schema/

The RDF Schema class and property system is similar to the type systems of object-oriented programming languages such as Java. RDF Schema differs from many such systems in that instead of defining a class in terms of the properties its instances may have, RDF Schema describes properties in terms of the classes of resource to which they apply.

The namespace is identified by:
http://www.w3.org/2000/01/rdf-schema#

Core Elements of RDFs

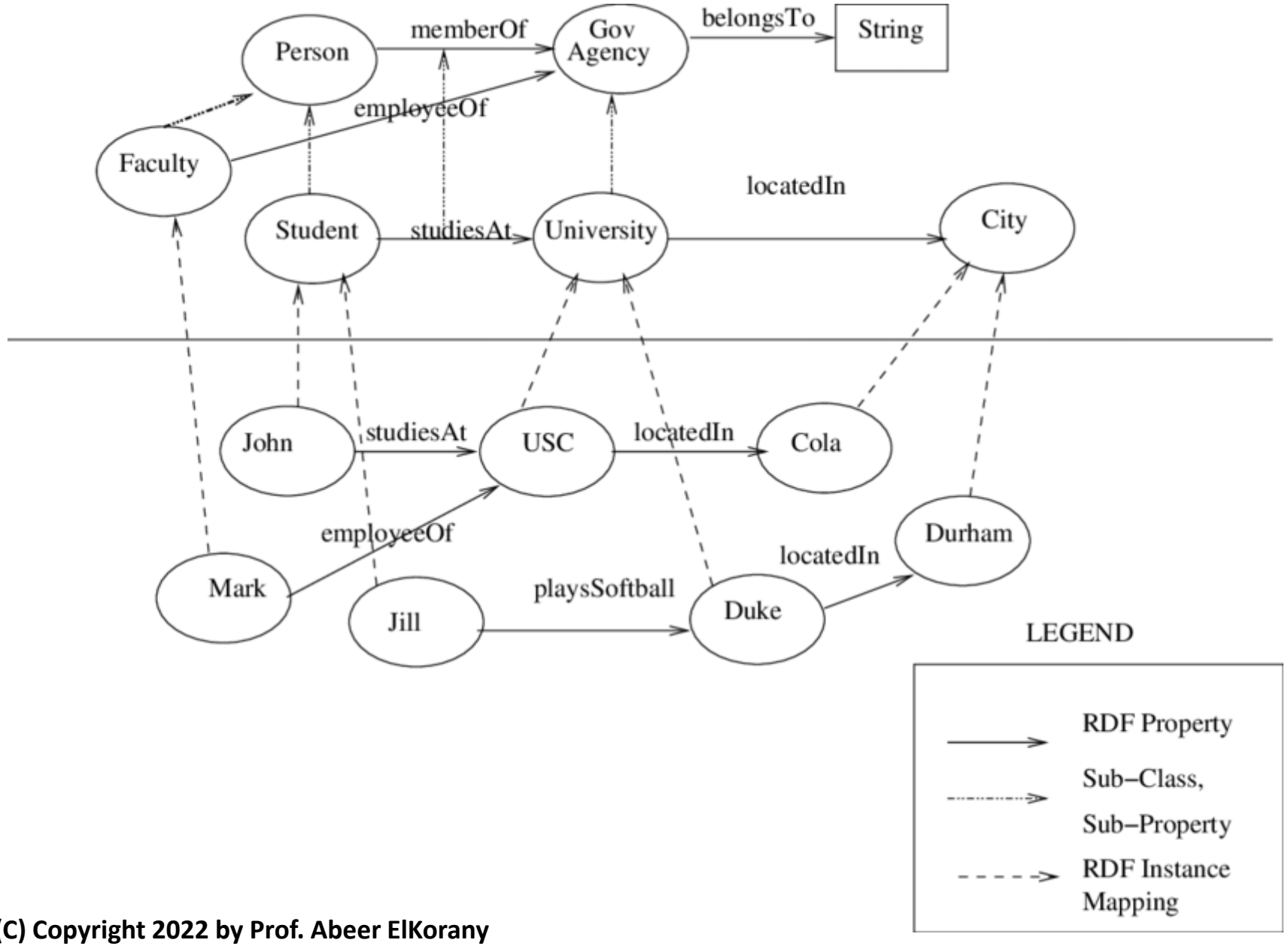
- Used to start from an ontology design.
- **rdfs:Resource**, the class of all resources.
- **rdfs:Class**, the class of all classes.
- **rdfs:Literal**, the class of all literals (strings).
- **rdf:Property**, the class of all properties.
- **rdf:Statement**, the class of all reified statements.

Core elements

- **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 elements (2)

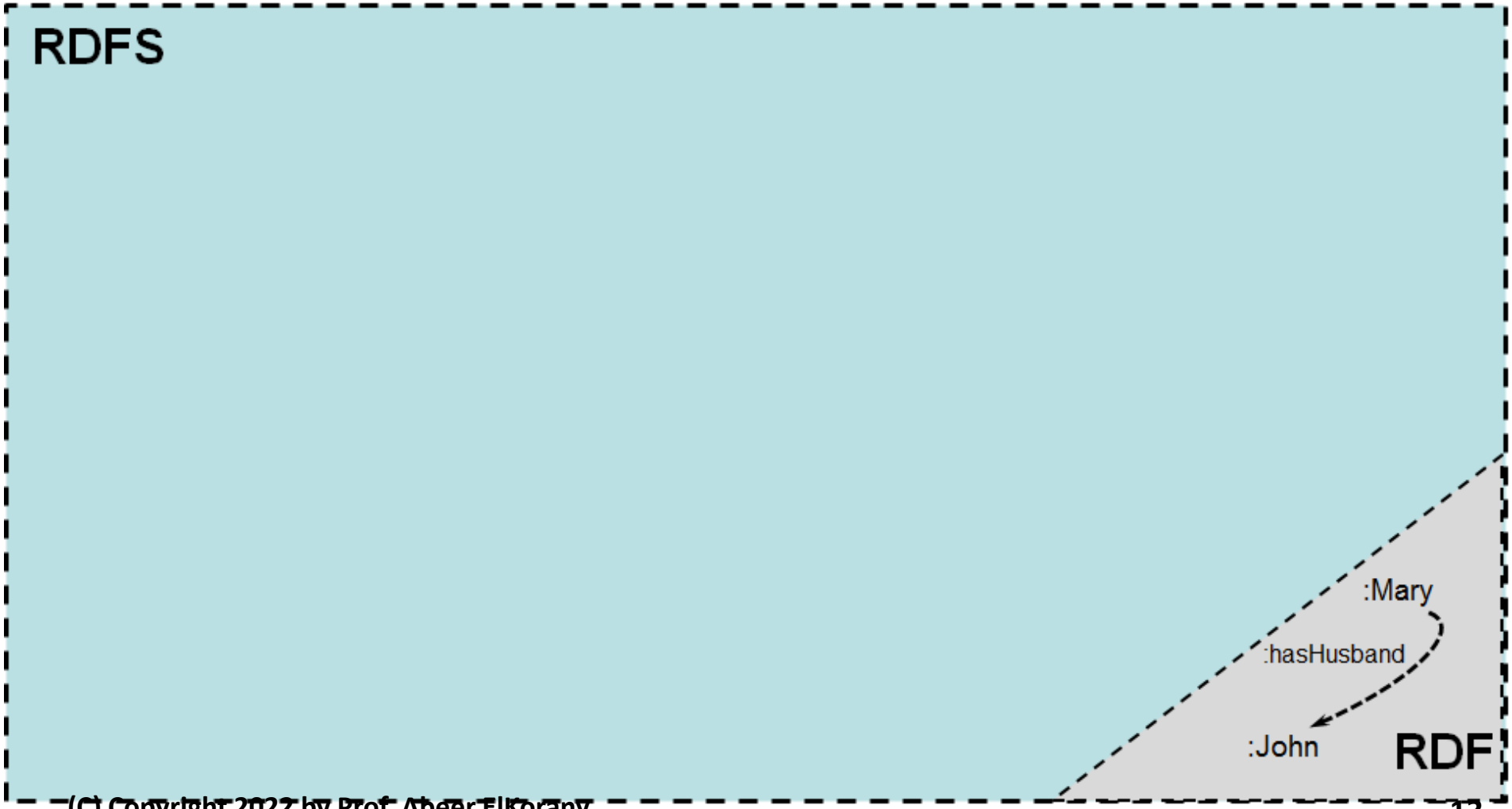
- **rdfs:domain**, which specifies the domain of a property P
  - The class of those resources that may appear as subjects in a triple with predicate P
  - If the domain is not specified, then any resource can be the subject
- **rdfs:range**, which specifies the range of a property P
  - The class of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



RDFS example

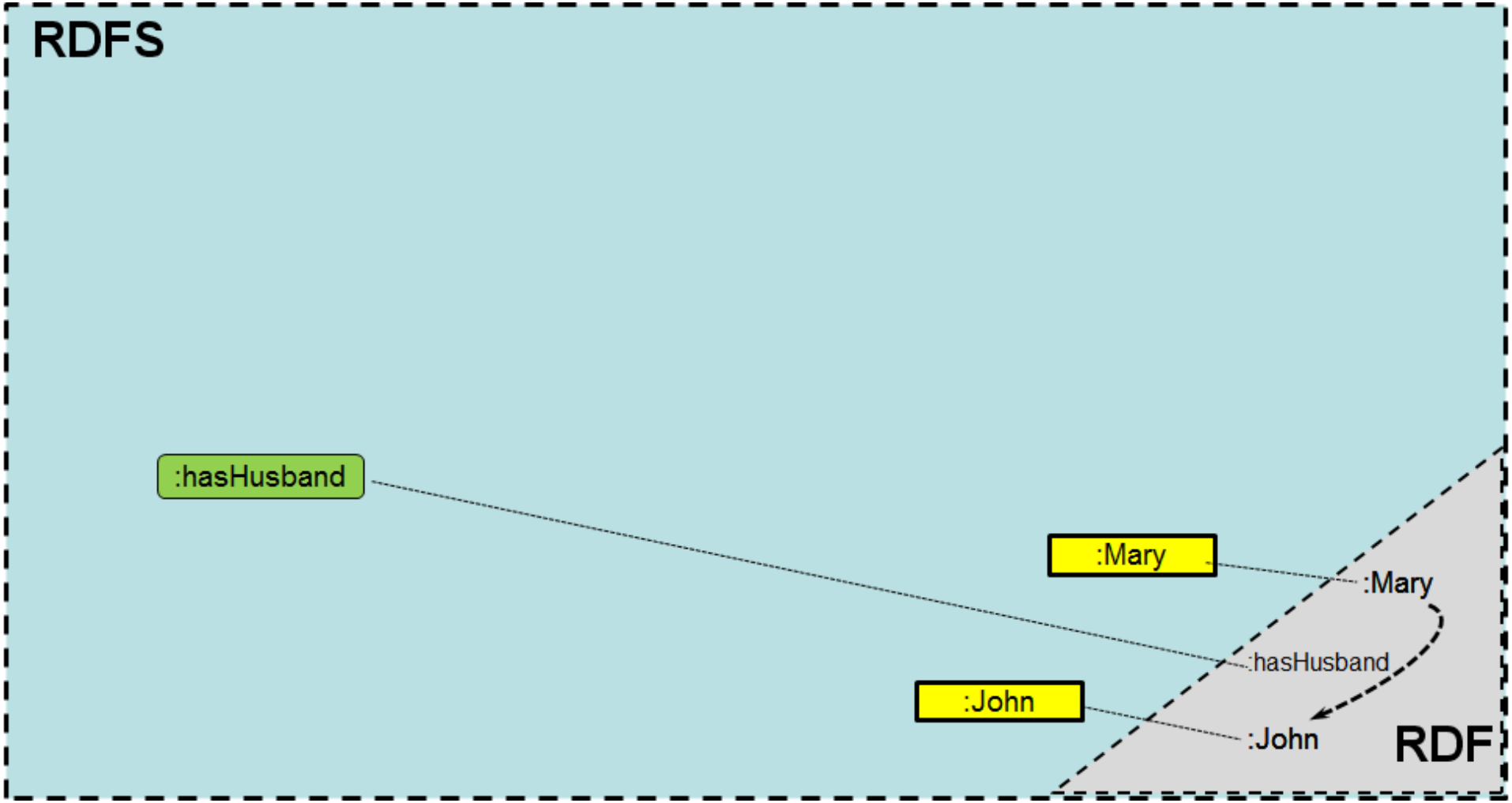
That's the only we know from the RDF statement:

```
…
:Mary      :hasHusband
:John .
…
```



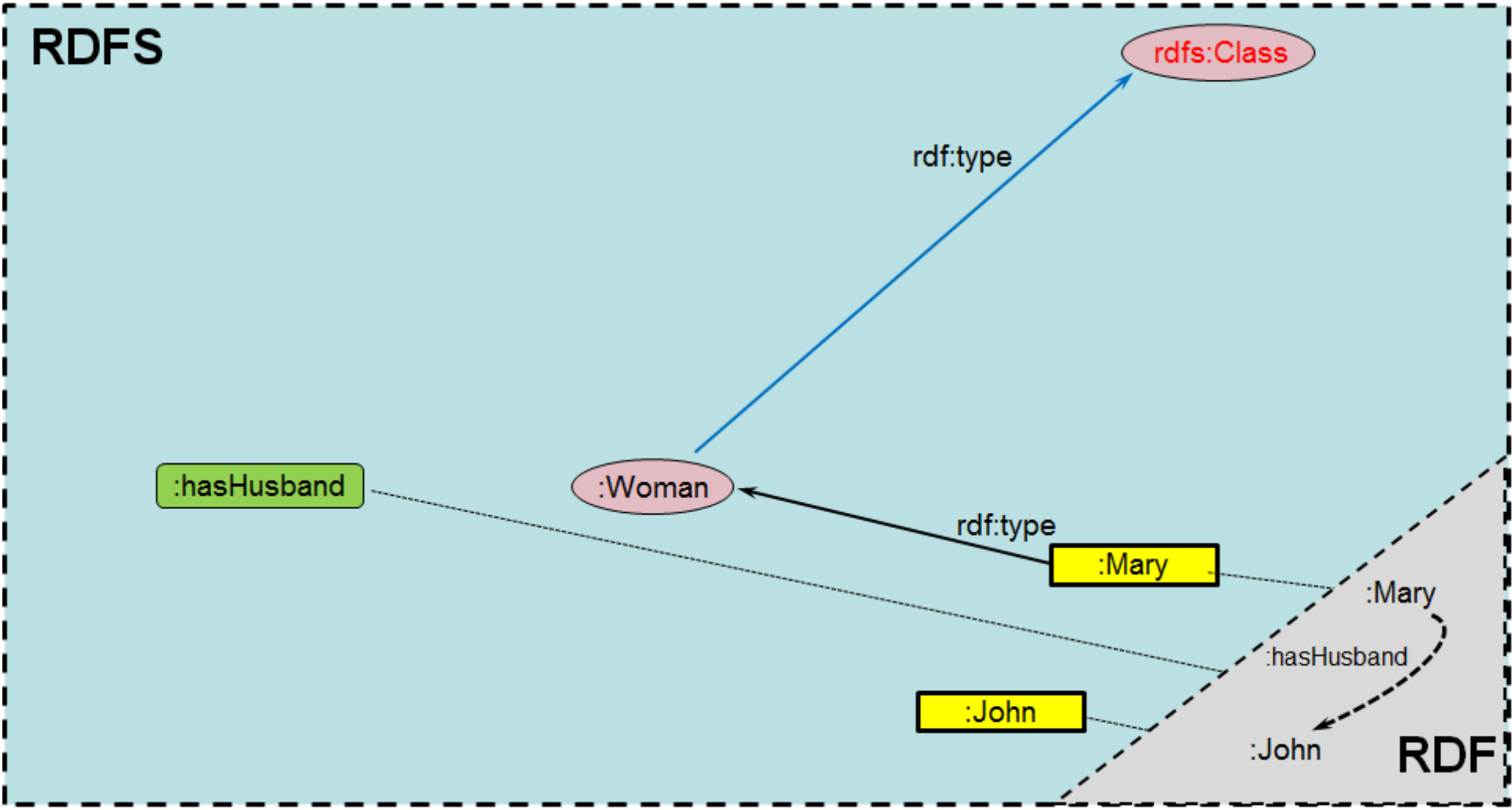
RDFS example

Now the question is, can we tell more about RDF statement components (subject, predicate, object) with the RDFS ?



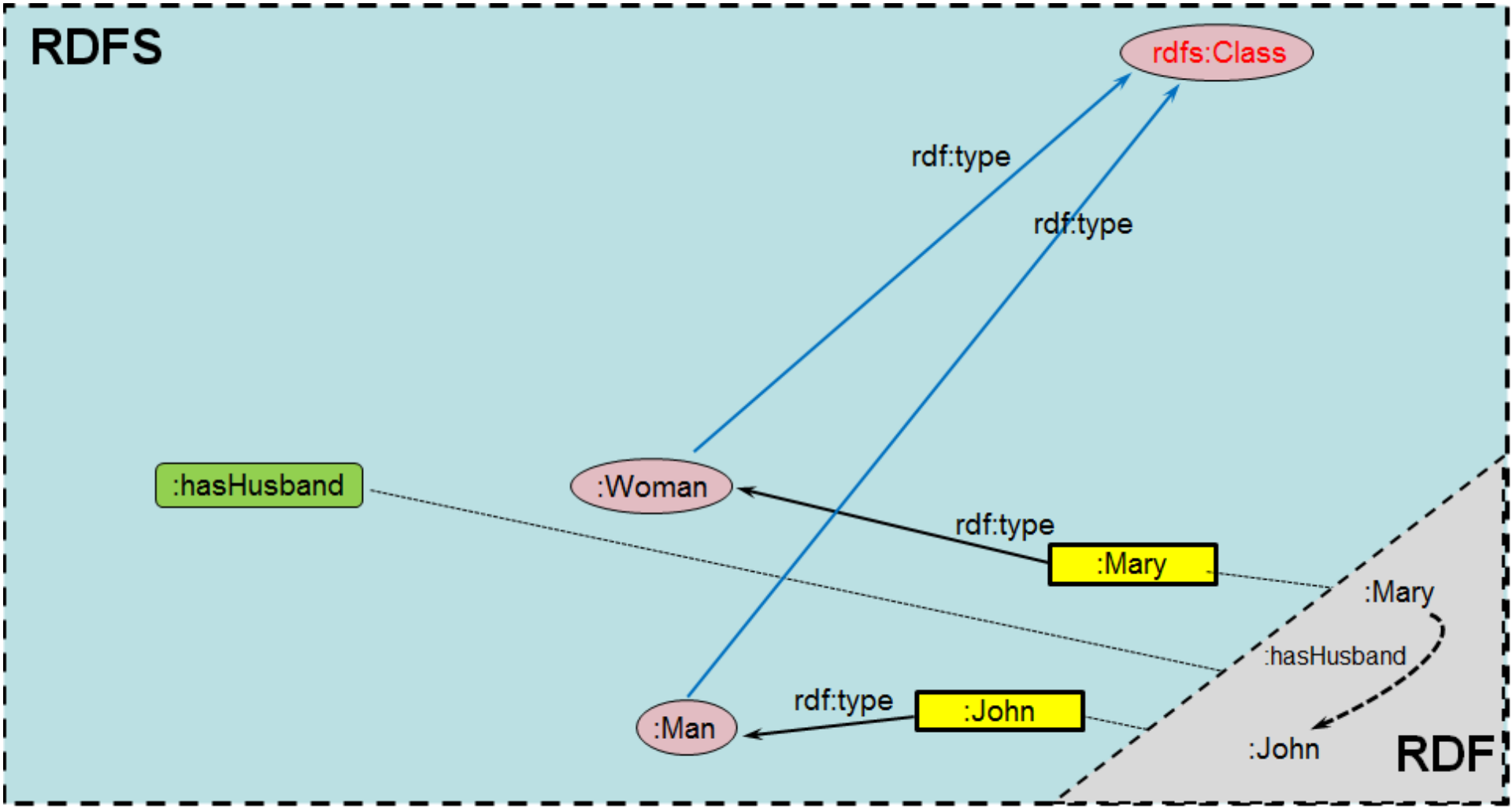
RDFS example

First, we say explicitly that :Mary is a :Woman and :Woman is a Class



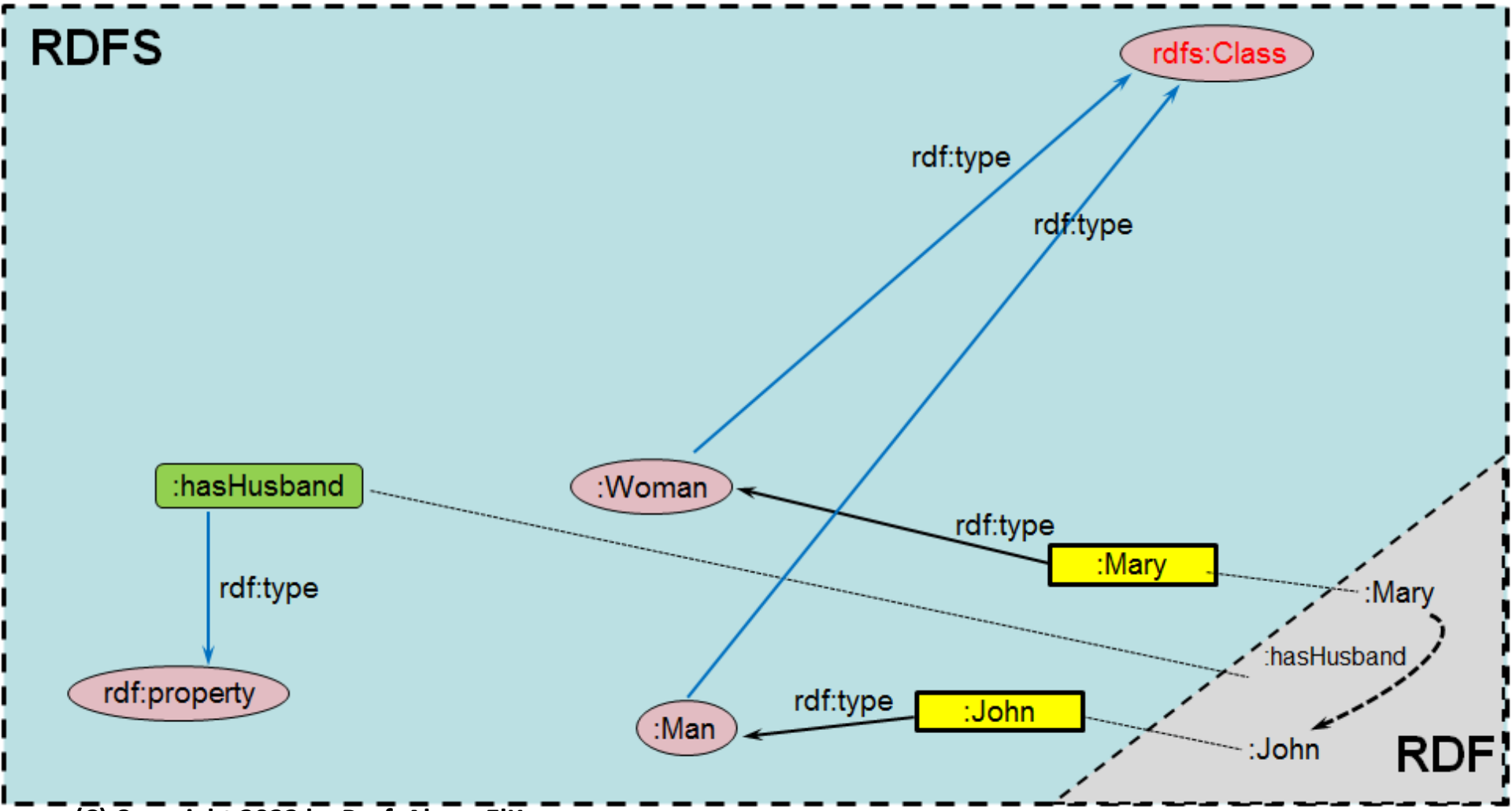
RDFS example

Then we say explicitly that :John is a :Man and :Man is a rdfs:Class



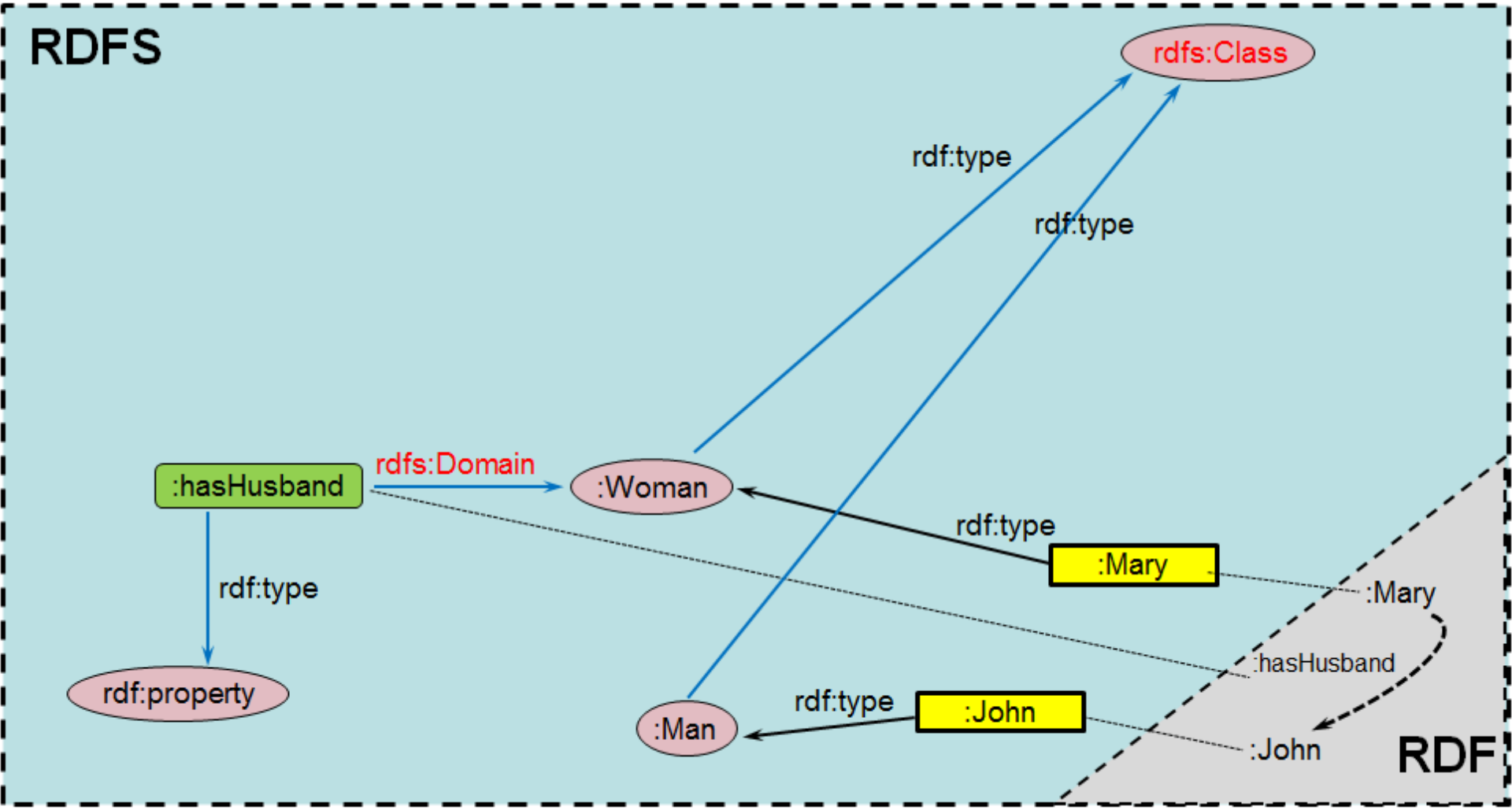
RDFS example

After that, we say explicitly that :hasHusband is a rdf:property



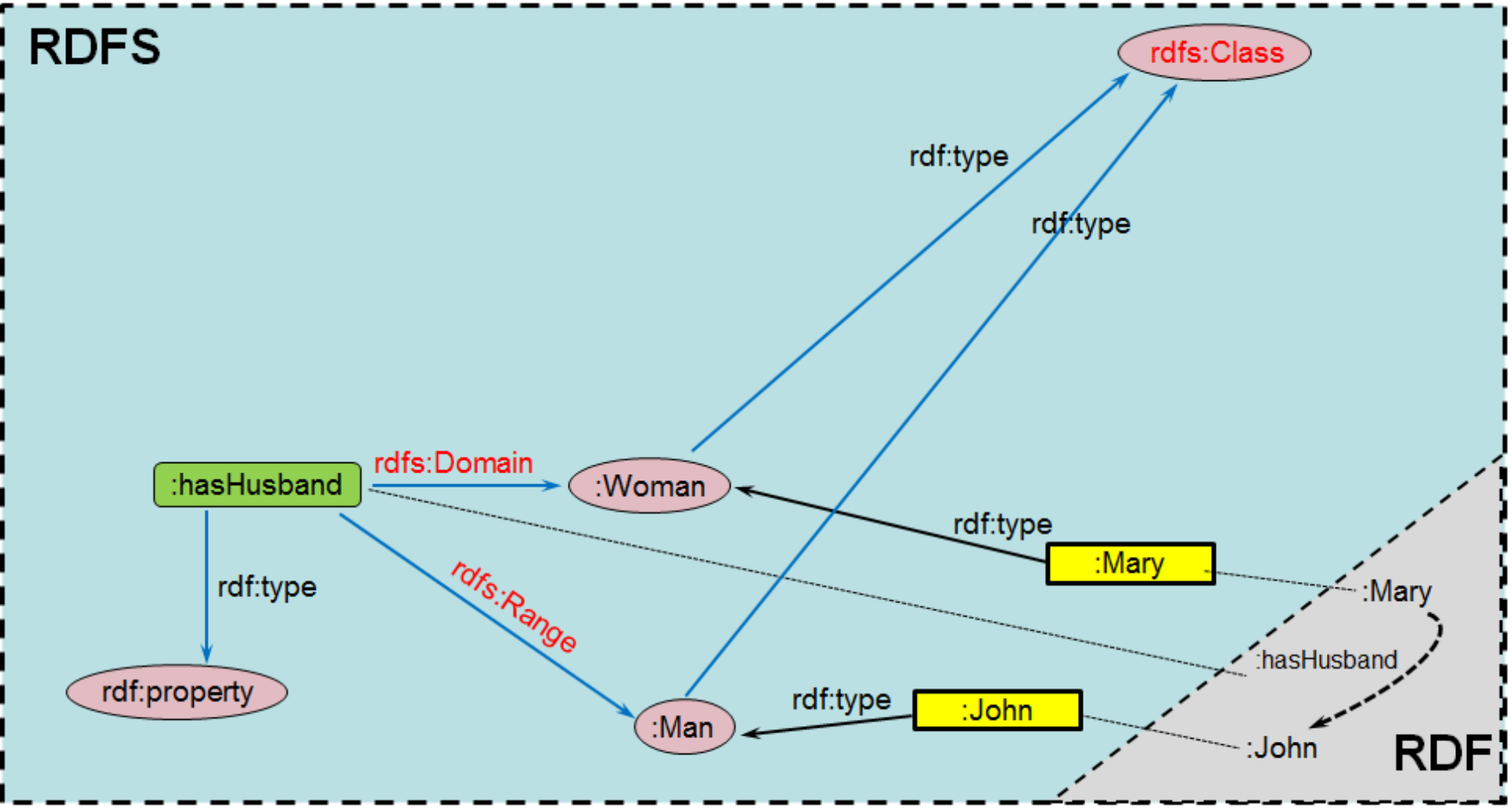
RDFS example

Then we say that class :Woman is the domain for the:hasHusband property



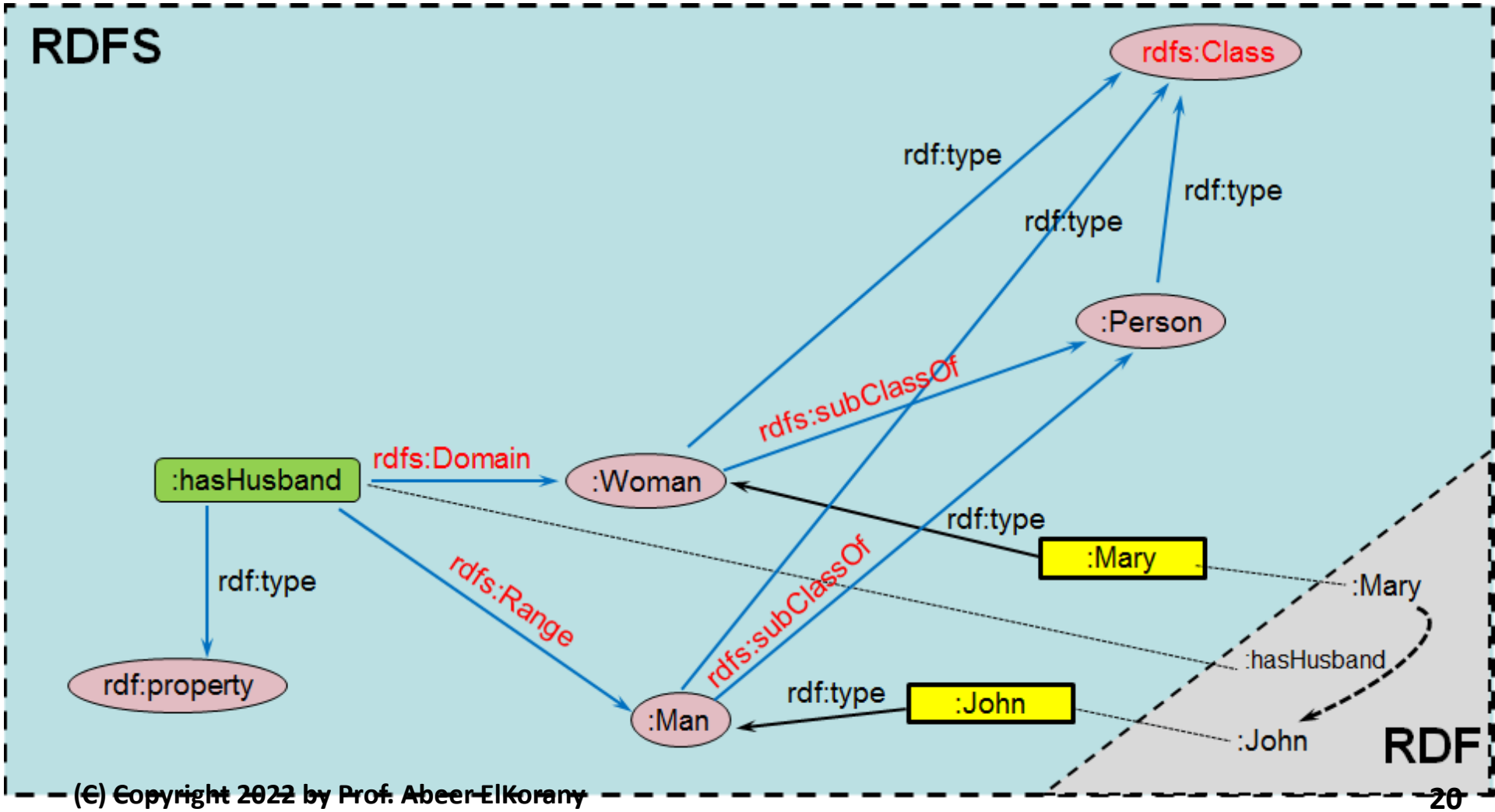
RDFS example

Then we say that class :Man is the range for the:hasHusband property



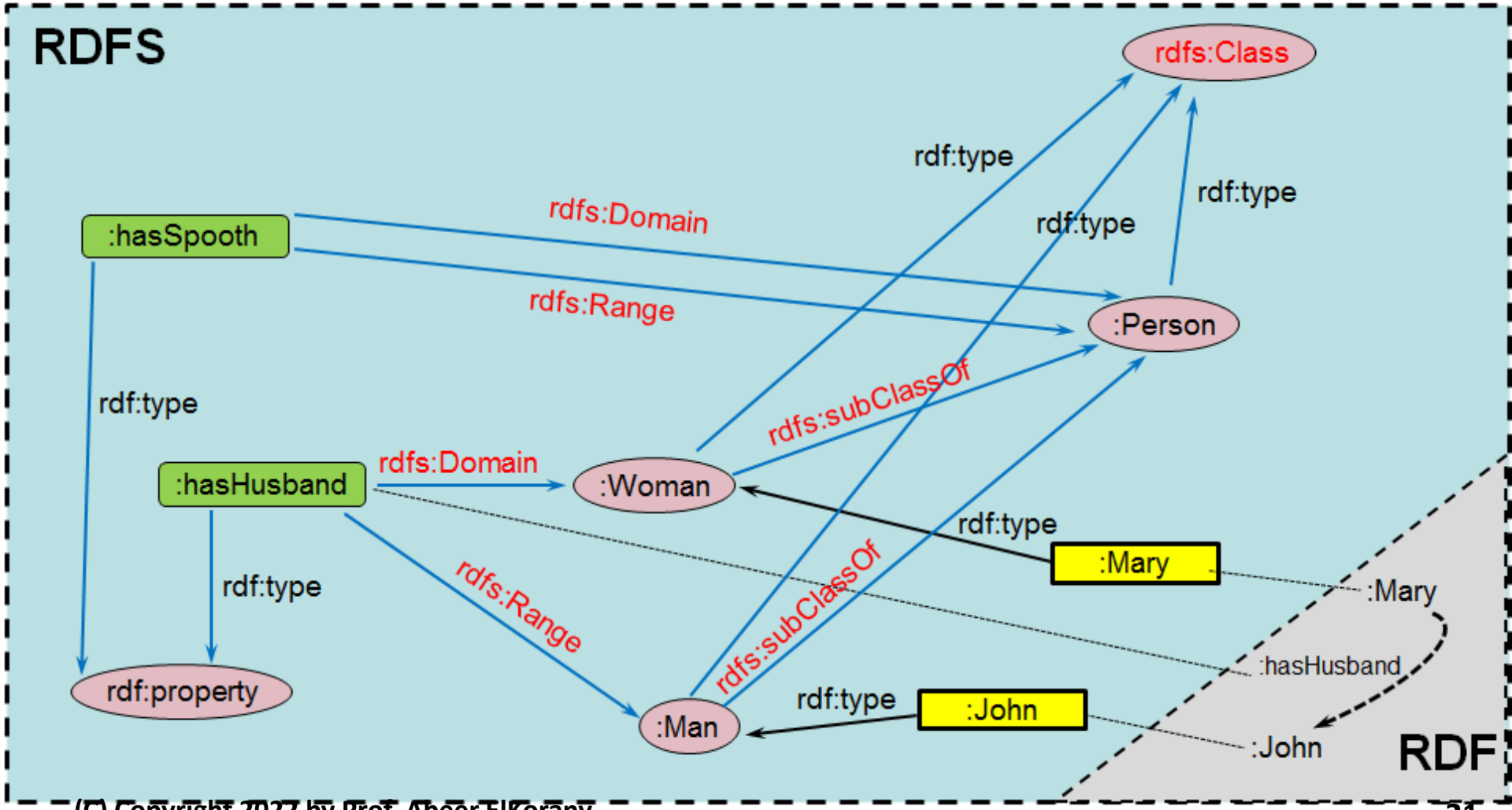
RDFS example

We can also add that both classes :Woman and :Man are subclasses of the class :Person



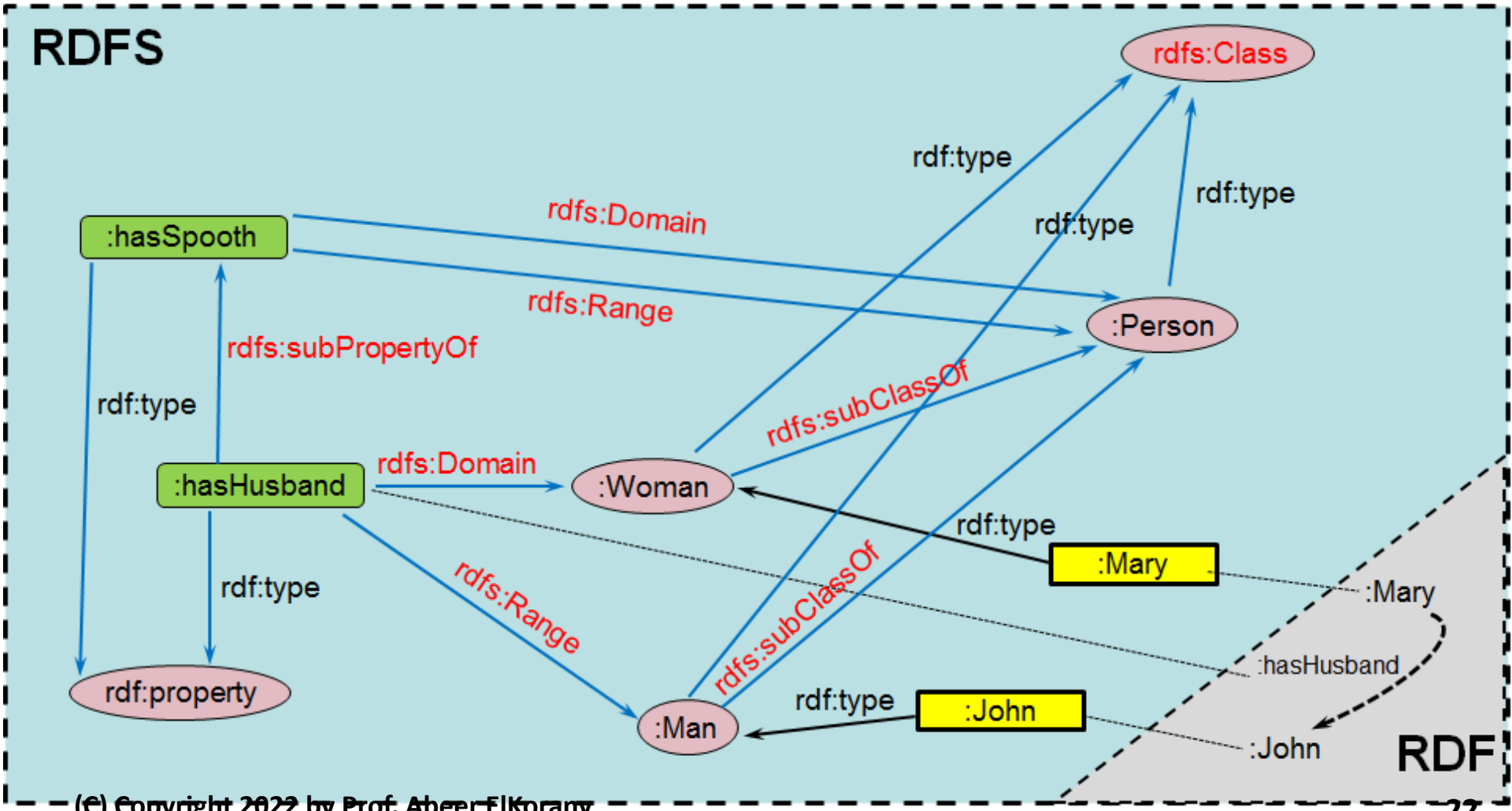
RDFS example

We can also introduce new (more general one) property :hasSpooth, which domain and range both belong to class :Person



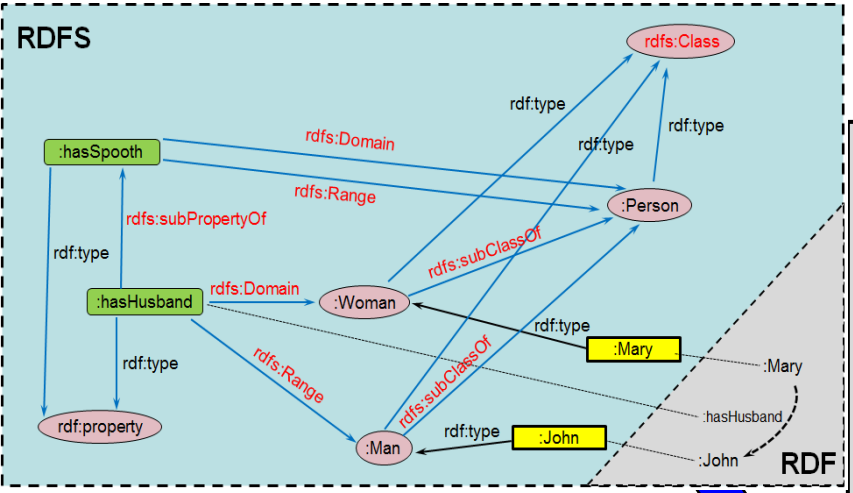
RDFS example

… and we can specify that the :hasHusband property is a subproperty of the :hasSpooth property



RDFS example

All the RDF and RDFS statements can be easily written within one "story" using RDF (N3) notation:




```
@prefix   rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix   rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix        :  <http://www.cs.jyu.fi/ai/vagan/ontologies/LifeInFinland.owl#> .

:Person          a                   rdfs:Class .
:Woman           a                   rdfs:Class .
:Man             a                   rdfs:Class .
:Woman           rdfs:subClassOf     :Person .
:Man             rdfs:subClassOf     :Person .
:hasHusband      a                   rdf:Property
:hasSpooth       a                   rdf:Property .
:hasHusband      rdfs:subPropertyOf  :hasSpooth .
:hasSpooth       rdfs:domain         :Person .
:hasSpooth       rdfs:range          :Person .
:hasHusband      rdfs:domain         :Woman .
:hasHusband      rdfs:range          :Man .
:John            a                   :Man .
:Mary            a                   :Woman .
:Mary            :hasHusband         :John .
```

Subclasses

- In order to express that every textbook is a book, e.g., that every instance of the class ex:Textbook is "automatically" an instance of the class ex:Book

Use rdfs:subClassOf property: ex:Textbook rdfs:subClassOf ex:Book .

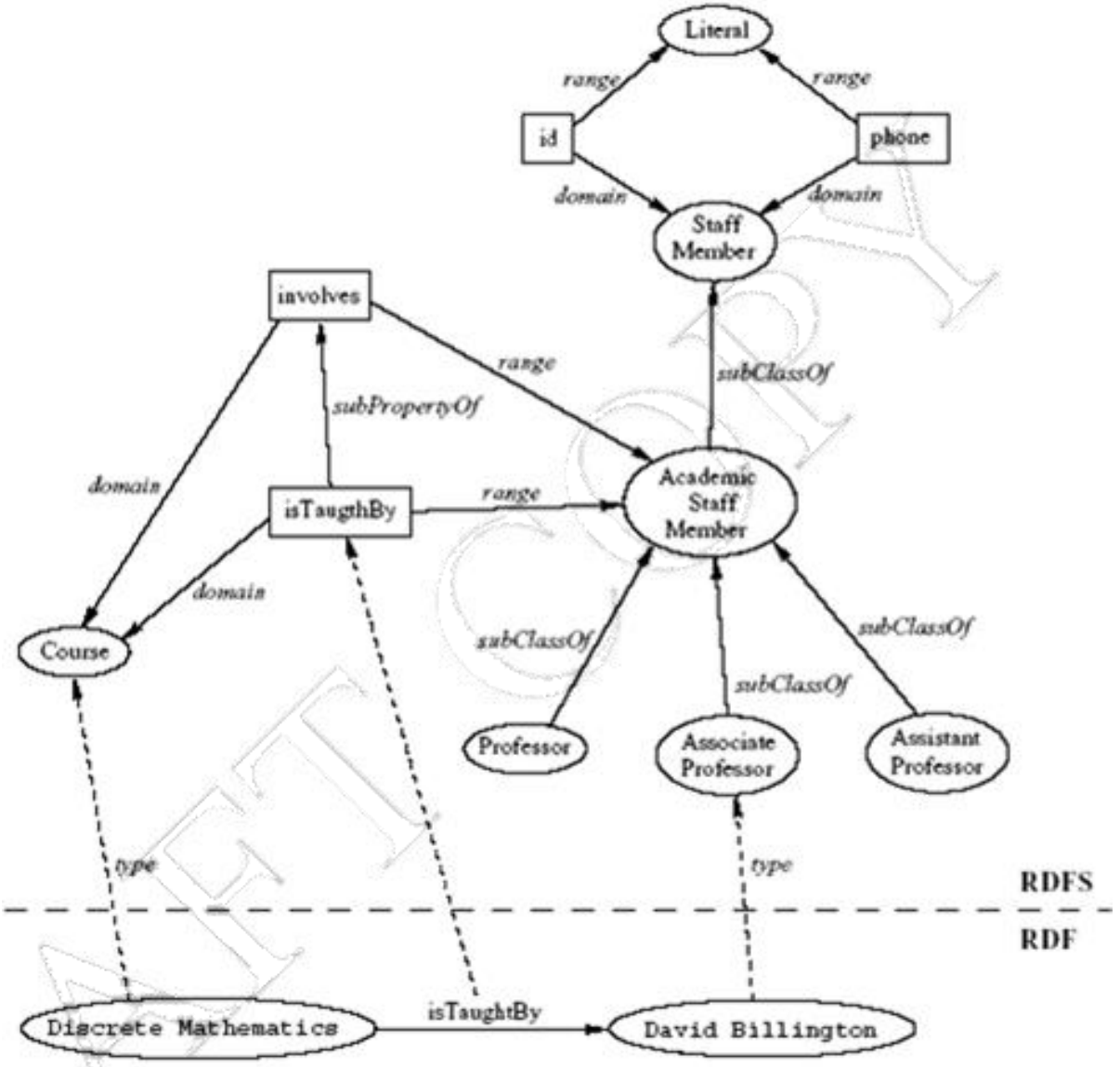
- rdfs:subClassOf is defined to be transitive and reflexive

- rule of thumb:

rdf:type means ∈ is an instance of

rdfs:subClassOf means ⊆ is a subclass of

RDF Layer vs RDF Schema Layer



Property Core elements

- rdfs:range
  - rdfs:range is an instance of rdf:Property that is used to state that the values of a property are instances of one or more classes.
- rdfs:domain
  - rdfs:domain is an instance of rdf:Property that is used to state that any resource that has a given property is an instance of one or more classes.

- More detailed syntax definitions are available at:
  - http://www.w3.org/TR/rdf-schema/
- You need to be familiar with them but don't need to remember all of them.

Example

```
<rdfs:Class rdf:ID=”staffMember">
</rdfs:Class>

<rdfs:Class rdf:ID=”lecturer">
      <rdfs:subClassOf rdf:resource=”#staffMember"/>
</rdfs:Class>

<rdf:Description rdf:ID="DavidBillingtonURI">
      <rdf:type rdf:resource=”#lecturer"/>
      <uni:name>David Billington</uni:name>
</rdf:Description>

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

Property Hierarchies

- The rdfs:subPropertyOf property may be used to state that one property is a subproperty of another.
- Hierarchical relationships for properties
  - E.g., “is taught by” is a **subproperty** of “involves”.
  - If a course C is involves an academic staff member A, then C also is taught by 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.

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>
```

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>
```

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**)

Limitations of RDFS

- **Local scope of properties**
  - **rdfs:range** defines the range of a property (e.g. eats) for **all classes**
  - In RDF Schema we cannot declare range restrictions that apply to some classes only
  - E.g. we cannot say that cows eat only plants, while other animals may eat meat, too
  - E.g. we cannot say:
    - Animals eat things
    - Herbivores are animals that eat only plants

Limitations of RDFS

- **Disjointness of classes**
  - Sometimes we wish to say that classes are disjoint (e.g. male and female).

- **Boolean combinations of classes**
  - Sometimes we wish to build new classes by combining other classes using union, intersection, and complement.
  - E.g. person is the union of the classes male and female.

Limitations of RDFS

- **Cardinality restrictions**
  - E.g. a person has exactly two parents, a course is taught by at least one lecturer.

- **Special characteristics of properties**
  - Transitive property (like “greater than”).
  - A property is the inverse of another property (like “eats” and “is eaten by”).

Summary

- RDF provides a foundation for representing and processing metadata
- RDF has a graph-based data model
- RDF has an XML-based syntax to support syntactic interoperability
  - XML and RDF complement each other because RDF supports semantic interoperability
- RDF has a decentralized philosophy and allows incremental building of knowledge, and its sharing and reuse

Summary (2)

- RDF is domain-independent
  - RDF Schema provides a mechanism for describing specific domains
- RDF Schema is a primitive ontology language
  - It offers certain modelling primitives with fixed meaning
- Key concepts of RDF Schema are class, subclass relations, property, subproperty relations, and domain and range restrictions
- There exist query languages for RDF and RDFS, including SPARQL

Further Reading

- I encourage you to look at the complete specification of RDF and RDFS.
  - RDF primer: http://www.w3.org/TR/rdf-syntax/
  - RDFS introduction http://www.w3.org/TR/rdf-schema/

- W3C formal definitions
  - http://www.w3.org/RDF/
  - http://www.w3.org/TR/rdf-schema/