COMP318

Ontologies and Semantic Web

OWL - Part 1

Dr Valentina Tamma
V.Tamma@liverpool.ac.uk

Where were we

- RDF data modelling language
- RDFS Schema language
- SPARQL query language

Brief recap of RDF/RDFS

- RDF/RDFS describe subject predicate object triples and basic subsumption relations.
  - Very efficient deduction/querying
    - SPARQL based on simple entailment, but other entailment regimes possible
    - answers to SPARQL queries are well defined
  - Describe the model behind the data (RDFS) by:
    - Declare the “types” and properties/ relationships of the things we want to make assertions about

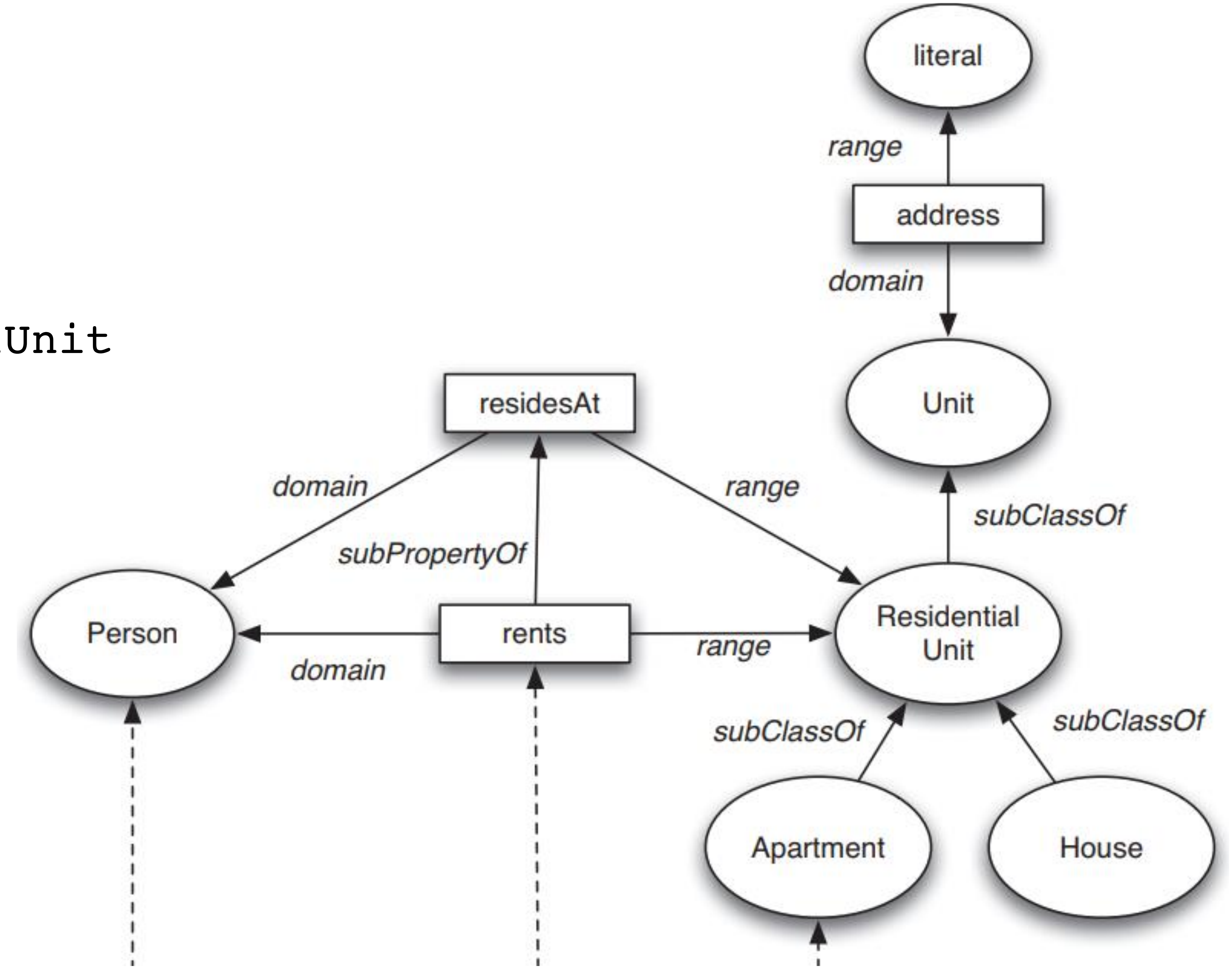
- Has a formal semantics
  - Allowing to infer new assertions implicitly stated from a set of given facts
    - Entailment is a mathematically defined relationship between RDF(S) graphs
  - RDFS uses three types of inference rules
    - type propagation,
    - property propagation, and
    - domain and range reasoning.

Brief recap of RDF/RDFS

- RDF/RDFS describe subject predicate object triples and basic subsumption relations.
    - Very poor expressivity...
        - Describe data in terms of triples (RDF)
            - Allowing to infer new assertions implicitly stated from a set of given facts
    - But sometimes we need to express more advanced notions, e. g.:
        - A person has only one birth date
        - No person can be male and female at the same time
    - The RDFS entailment rules are incomplete

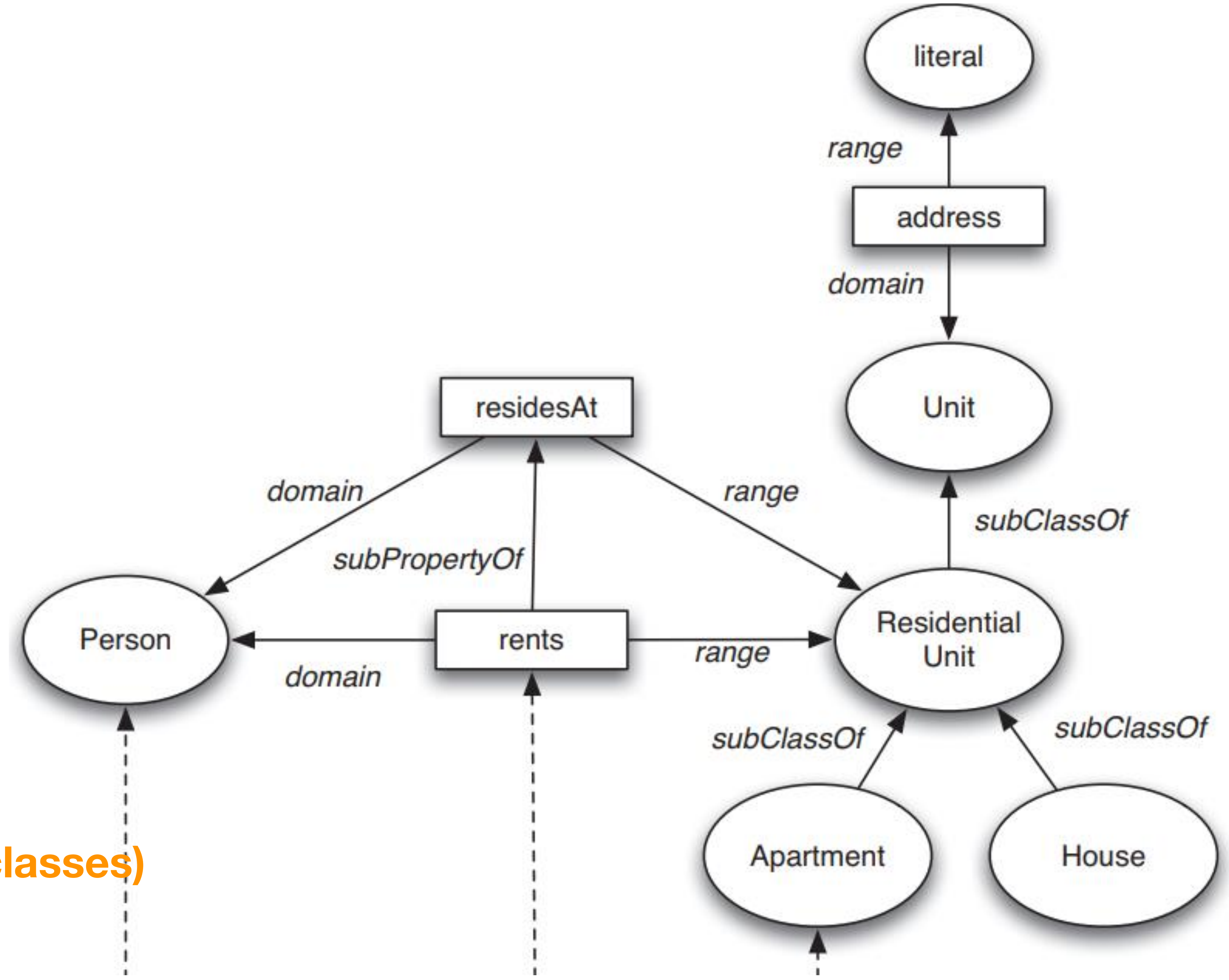
What can you represent with RDFS

- ***RDFS provides:***
  - Classes
    - `:Apartment rdf:type rdfs:Class`
  - Class hierarchies
    - `:Apartment rdfs:subClassOf :ResidentialUnit`
  - Properties
    - `:rents rdf:type rdf:Property`
  - Property hierarchies
    - `:rents rdfs:subPropertyOf :residesAt`
  - Domain and range declarations
    - `:rents rdfs:domain :Person`
    - `:rents rdfs:range :ResidentialUnit`
      - They infer information rather than checking data



What can't you represent in RDFS

- ***RDFS does NOT provide:***
  - Disjointness of Classes
    - `Apartment` *and* `House` *are disjoint*
      - *they cannot have any shared instaces*
  - Property characteristics (inverse, transitive, ...)
    - `:Person :rents :Apartment` *and* `:Apartment :isRentedBy :Person` *are not explicitly related*
  - Local scope of properties
    - A class `:ApartmentTenant` is defined by a property `:rents` whose values (range) is restricted to all values of the class `:Apartment`
      - *only people who rent an apartment can be an apartment tenant*
      - but other `:Person` can rent a house
  - Complex concept definitions (Boolean combination of classes)
    - `:ResidentialUnit = :Apartment ∪ :House`
    - `:Apartment = :ResidentialUnit ∩ ¬(:House)`



What can't you represent in RDFS

- ***RDFS does NOT provide:***
  - Cardinality restrictions
    - `:ResidentialUnit` *may have at most 1 address*
    - `:TwoBedroomApartment` *must have exactly two tenants*
  - A way to distinguish between classes and instances, no clear ontology/data boundary:
    - `:Building rdf:type rdfs:Class`
    - `:AmsterdamBuilding rdf:type :Building`
    - `:BaronWayApartment rdf:type :AmsterdamBuilding`
      - `:BaronWayApartment` is an instance of an instance (`:AmsterdamBuilding`)
  - A way to distinguish between language constructors and ontology vocabulary:
    - `rdf:type rdfs:range rdfs:Class`
    - `rdfs:Property rdfs:type rdfs:Class`
    - `rdf:type rdfs:subPropertyOf rdfs:subClassOf`
- Reasoning for these non-standard semantics

What can you infer in RDFS

Schema

```
ex:Lecturer
rdfs:subClassOf
ex: AcademicStaff
```

RDFS Entailment ⇒

Inferred Fact

```
staffUniv:john_smith
rdf:type
ex: AcademicStaff
```

Assertion

```
staffUniv:john_smith
rdf:type
ex: Lecturer
```

What can't you infer in RDFS

- However, not all types of inferences are possible in RDFS

Schema

```
:wife_of rdfs:subPropertyOf married_to.
:married_to rdfs:domain :Spouse;
            rdfs:range  :Spouse.
:wife_of rdfs:domain :Wife;
         rdfs:range  :Husband.
```

Assertion

```
:juliet :wife_of :romeo.
```

Facts Inferred

```
:juliet rdf:type   :Wife;
        rdf:type   :Spouse;
        married_to :romeo;
:romeo rdf:type :Spouse;
       rdf:type :Husband.
```

What can't you infer in RDFS

- What about if we want to model symmetry, *i.e.* `:x :married_to :y` implies `:y :married_to :x`?

Facts Inferred

```
:juliet rdf:type   :Wife;
        rdf:type   :Spouse;
        married_to :romeo;
:romeo rdf:type :Spouse;
       rdf:type :Husband.
```

Schema

```
:wife_of rdfs:subPropertyOf married_to.
:married_to rdfs:domain :Spouse;
            rdfs:range  :Spouse.
:wife_of rdfs:domain :Wife;
         rdfs:range  :Husband.
:husband_of rdfs:domain :Husband;
            rdfs:range  :Wife.
```

Facts *NOT Inferred*

```
:romeo :married to :juliet.
:romeo :husband_of :juliet.
```

Assertion

```
:juliet :wife_of :romeo.
```

Too much representational freedom is not good!

- We might want to be able to detect what might seem inconsistent facts, but RDFS is not able to constrain models through consistency and axioms:

Schema

```
:wife_of rdfs:subPropertyOf married_to.
:married_to rdfs:domain :Spouse;
            rdfs:range  :Spouse.
:wife_of rdfs:domain :Wife;
         rdfs:range  :Husband.
:husband_of rdfs:domain :Husband;
            rdfs:range  :Wife.
```

Assertions

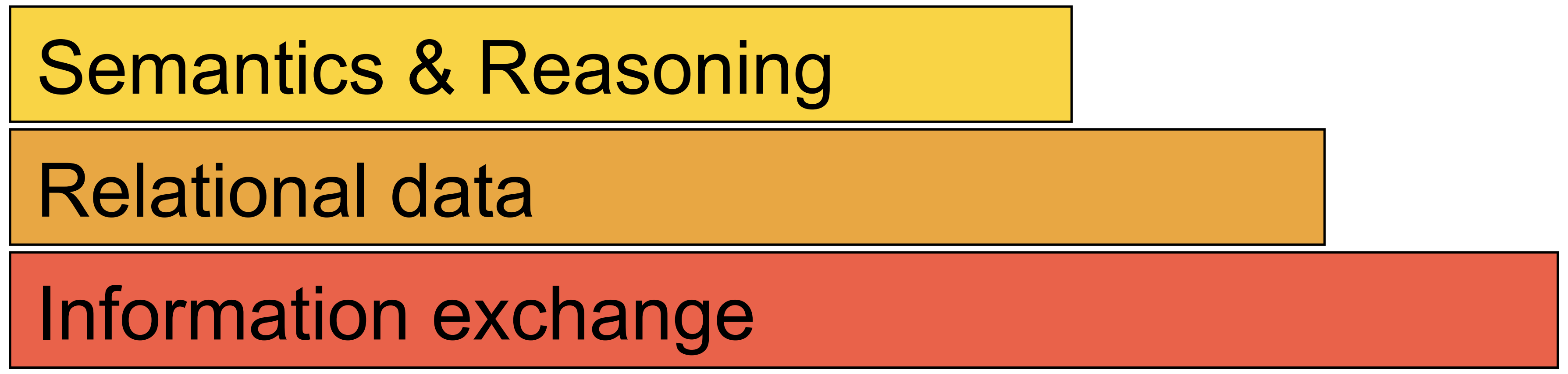
```
:romeo rdf:type :Husband.
:romeo :wife_of : juliet.
```

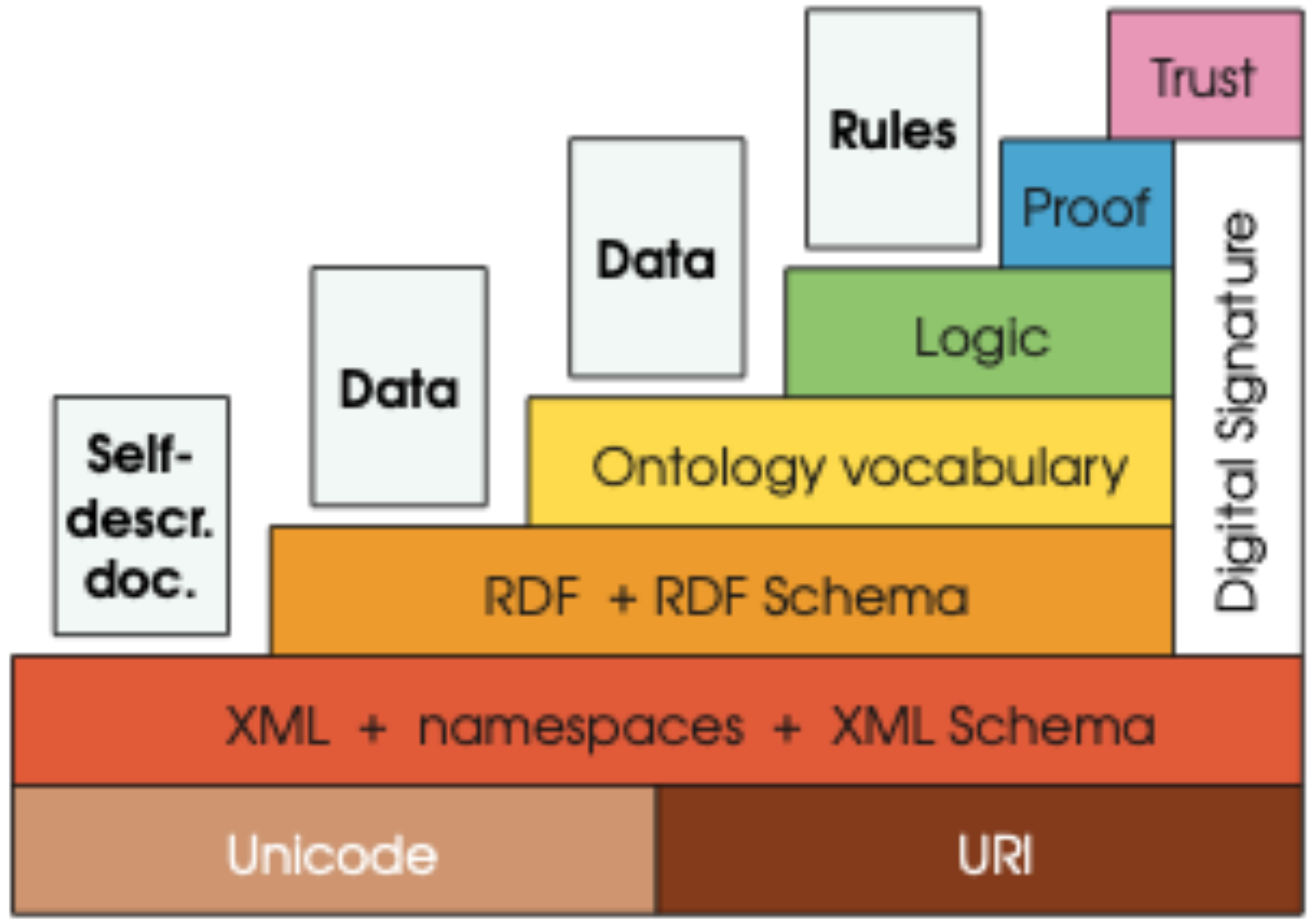
Facts ***INCORRECTLY Inferred***

```
:romeo rdf:type :Wife.
```

There is no contradiction, and the mis-modelling is not diagnosed automatically!

Layering of SW languages





T. Berners-Lee

Semantic + Web = Semantic Web

Represent Web content in a form that is more easily machine-processable:

describe meta-data about resources on the Web

i.e. descriptions about the data being represented, the model and constraints used to represent them.

Use intelligent techniques to take advantage of these representations:

process meta-data in a way that is similar to human reasoning and inference

thus information gathering can be done by a machine in a similar way to how humans currently gather information on the web..

Conflicting aims

- Ontologies provide the structured vocabulary for describing meta-data
  - Languages to represent ontologies need to be as expressive as possible whilst permitting automated deduction:
    - To describe meta-data, we want a (logic-based) language that is as expressive as possible.
    - To simulate human deduction in an efficient way, we want a logic that permits efficient automated deduction.
    - The logic of choice is a compromise between expressiveness and complexity of deduction.

Wish list for ontology languages....

- Compromise between expressivity and scalability
  - Well defined syntax
    - necessary for automatic machine-processing of information;
  - Formal semantics
    - describe the meaning of knowledge precisely, it allows computers to reason precisely about knowledge
      - **class membership:** if $x \in A$ and $A \subseteq B$ then we can infer that $x \in B$
      - **consistency:** $x \in A$, $A \subseteq B \cap C$, $A \subseteq D$ and $B \cap D = \varnothing$ then we have an inconsistency!
      - **classification:** if some property-value pairs are a sufficient condition for membership in A, and x satisfies them, then infer $x \in A$
- Efficient reasoning
  - derivations can be computed mechanically
    - consistency, classification, detection of unintended relationships between classes;
- Sufficient expressive power
  - Compatibility with RDF and RDFS;
- Open world assumption:
  - The absence of a particular statement means, in principle, that the statement has not been made explicitly yet.
  - Whether the statement is true or not, and whether it is believed that it is (or would be) true or not is irrelevant.
  - Thus, from the absence of a statement alone, a deductive reasoner cannot infer that the statement is false.

Requirements for Ontology Languages

- The main requirements are:
  - a well-defined syntax
  - a formal semantics
  - convenience of expression
  - efficient (complex) reasoning support
  - sufficient expressive power

RDFS

OWL

COMP318

Ontologies and Semantic Web

End of OWL - Part 1

Dr Valentina Tamma
V.Tamma@liverpool.ac.uk