Annex G

Complex Properties

With applications towards:

- CEN TC442 Product/Property Data Templates (PDTs)
- bSI IFC/PSETS/bSDD

Existing Semantic Resource: OPM

- Ontology for Property Management (OPM)
  - Based on paper “OPM, An Ontology for describing properties that evolve over time”, Mads Holten Rasmussen (DTU), Maxime Lefrançois (Uni Lyon), Mathias Bonduel (KU Leuven), Christian Anker Hviid (DTU) & Jan Karlshøj (DTU)

- Context:
  - World Wide Web Consortium (W3C)
    - Linked Building Data (LBD) Community/Working Group (CG/WG)

- Dependencies (currently) on other resources
  - CDT/UCUM, SEAS, schema.org ontologies

Two property forms

1. Numerical datatype properties like ‘height’
2. Lexical datatype properties like ‘materialType’

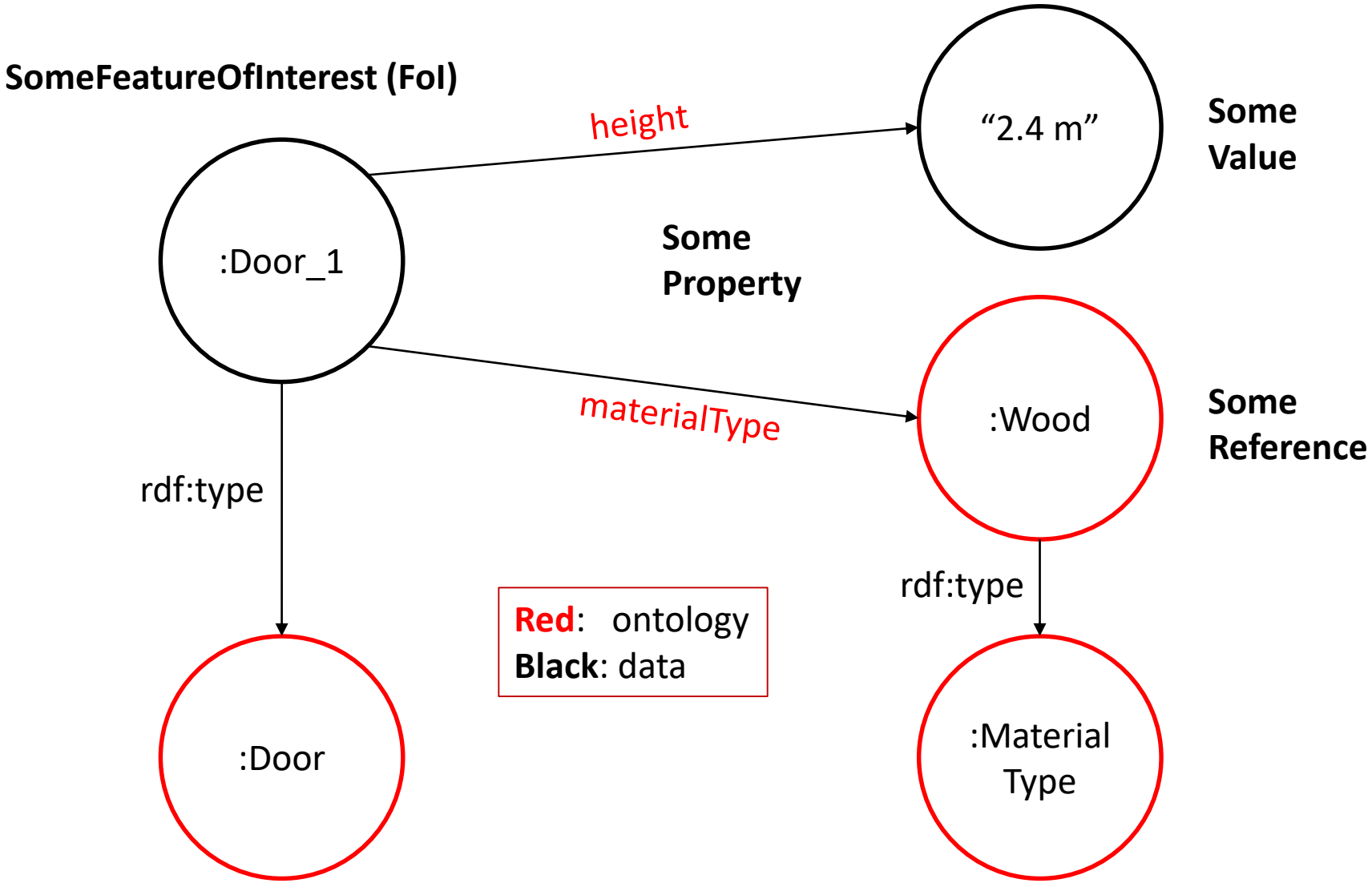
- In short: Quantities & Qualities

- i.e. ‘height’ having a value scaled according to some unit of measure, related to some base or derived quantity kind,
- i.e. ‘materialType’ having a reference to some allowed (reference) instance as enumeration item

OPM Complexity Levels

- Level L1: simplest, no objectification
  - Used by this standard for simple properties
- Level L2: more complex: one time objectification
  - Good for explicit units, values (not WKT) or other metadata on property/value assignment
  - Used by this standard for complex properties
- Level L3: even more complex: double objectification
  - Good for separate metadata on property assignment and metadata on value assignment
- You can choose, or you can combine
- L1 can be derived from L2 can be derived from L3, with data loss (when not combined)
- Several options to keep combined levels ‘in sync’ (i.e. via standard SPARQL Query)
- Values in all cases represented as Well Known Text (WKT) strings

Level L1 - graphically



Level L1 – in OWL/Turtle

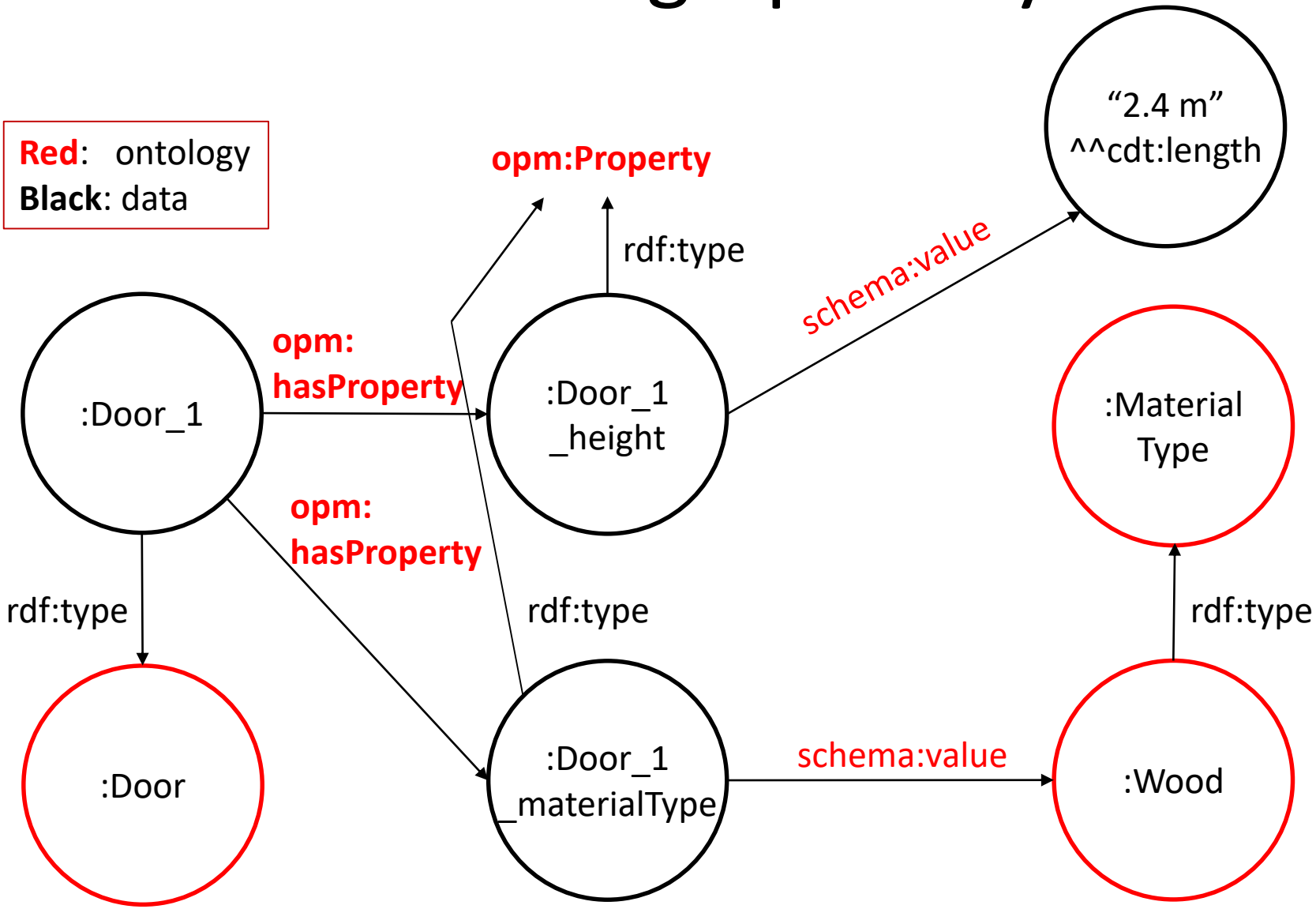
Ontology

```
:Door rdf:type owl:Class .
:MaterialType rdf:type owl:Class .
:Wood rdf:type :MaterialType .
:height rdf:type owl:DatatypeProperty
        rdfs:range cdt:length .
:materialType rdf:type owl:ObjectProperty ;
        rdfs:range :MaterialType .
```

Data

```
:Door_1 rdf:type :Door ;
        :height “2.40 m”^^cdt:length ;
        :materialType :Wood .
```

Level L2 – graphically



Level L2 – in OWL/Turtle

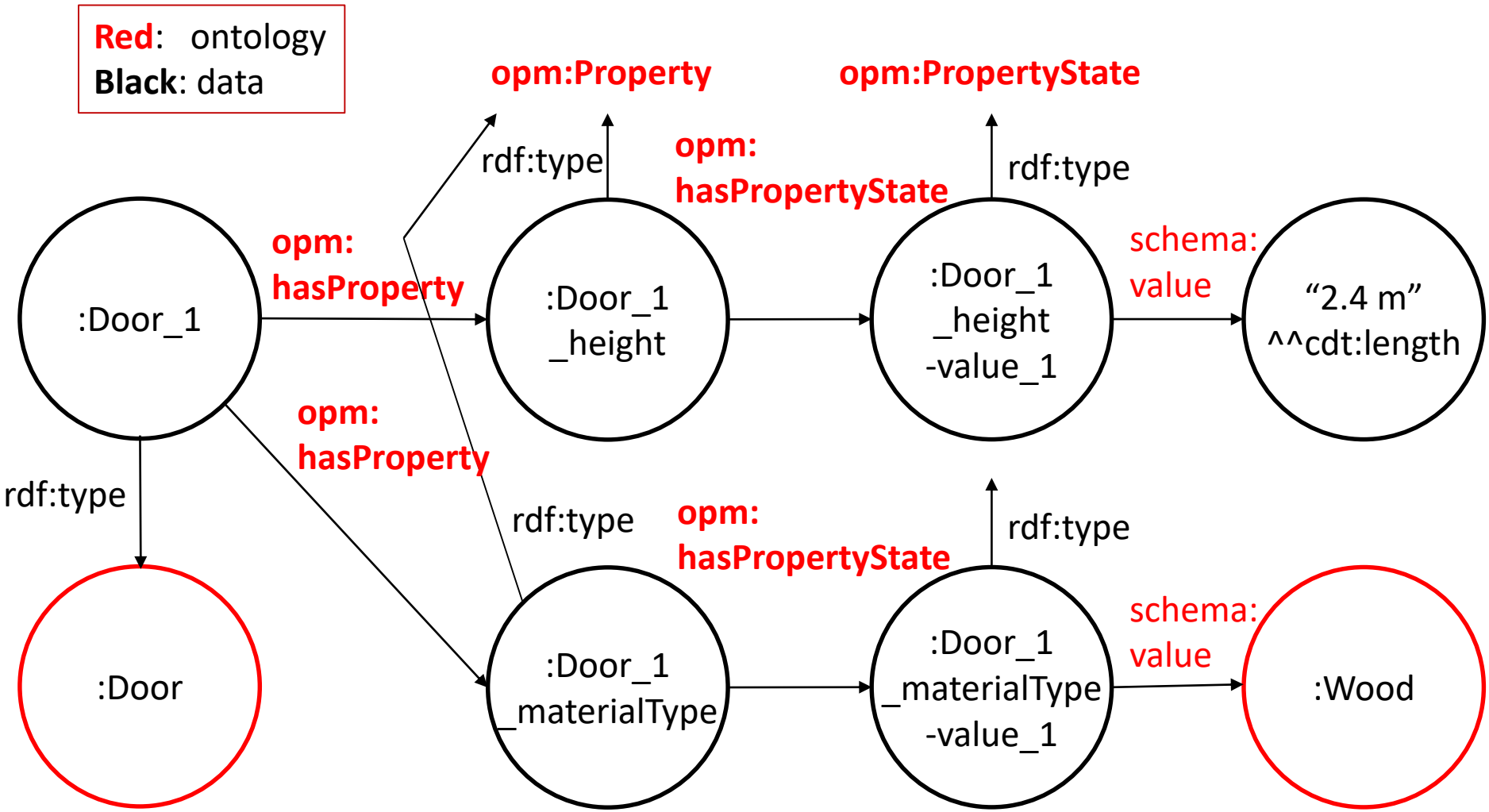
Ontology

```
:Door rdf:type owl:Class .
:MaterialType rdf:type owl:Class .
:Wood rdf:type :MaterialType .
opm:Property rdf:type owl:Class .
opm:hasProperty rdf:type owl:Class ;
        rdfs:range opm:Property .
schema:value rdf:type rdf:Property .
```

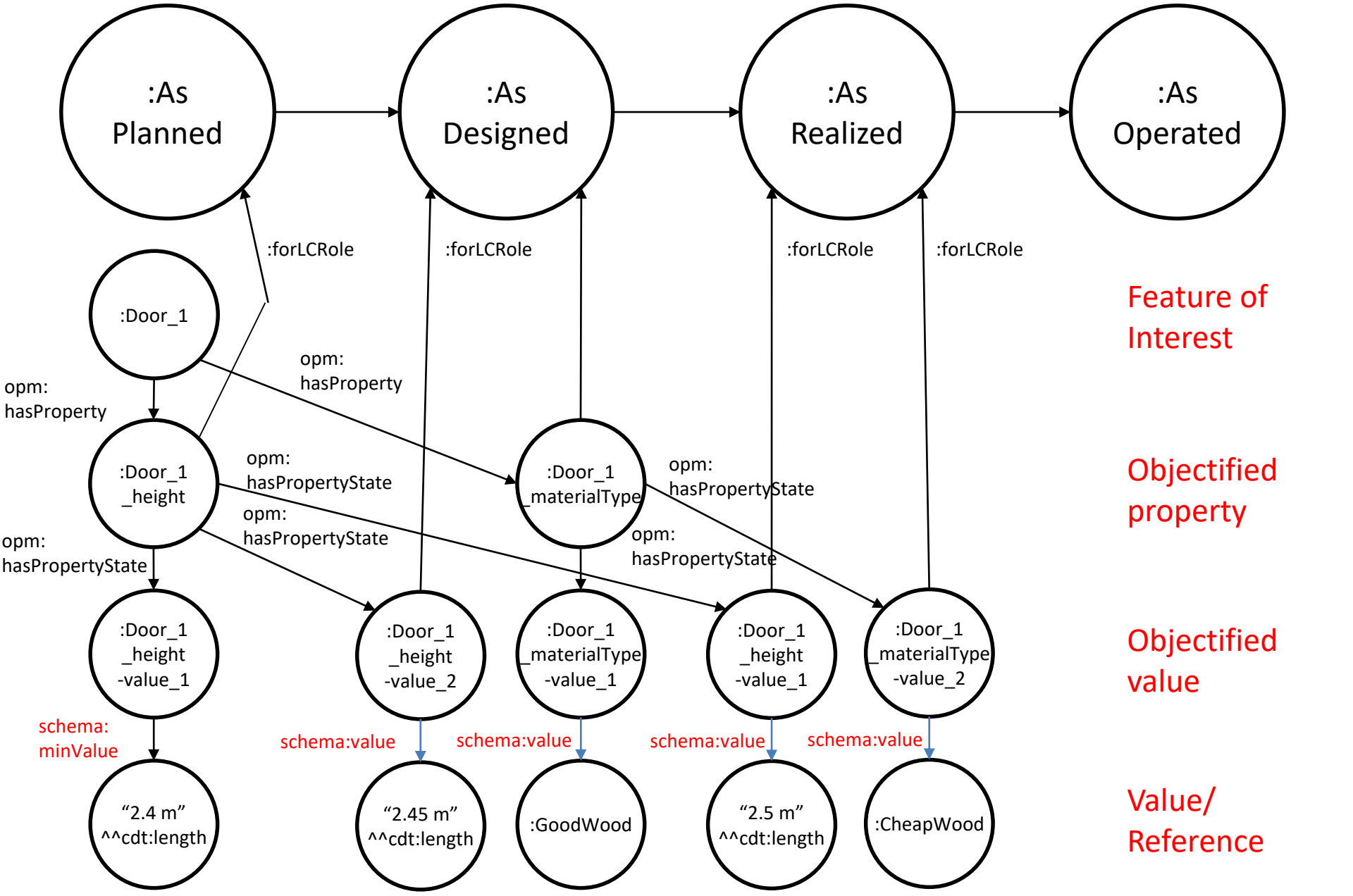
Data

```
:Door_1 rdf:type :Door;
        opm:hasProperty :Door_1_height ;
        opm:hasProperty :Door_1_materialType .
:Door_1_height rdf:type opm:Property ;
        schema:value “2.40 m”^^cdt:length .
:Door_1_materialType rdf:type opm:Property ;
        schema:value :Wood .
```

Level L3 - graphically



Example application for L3 Simple Systems Engineering



Level L3 – in OWL/Turtle /1

Ontology

```
:Door rdf:type owl:Class .
:MaterialType rdf:type owl:Class .
:Wood rdf:type :MaterialType .
opm:Property rdf:type owl:Class .
opm:hasProperty rdf:type owl:Class ;
        rdfs:range opm:Property .
opm:PropertyState rdf:type owl:Class .
opm:hasPropertyState rdf:type owl:ObjectProperty ;
        rdfs:range opm:PropertyState .
schema:value rdf:type rdf:Property .
```

Level L3 – in OWL/Turtle /2

Data

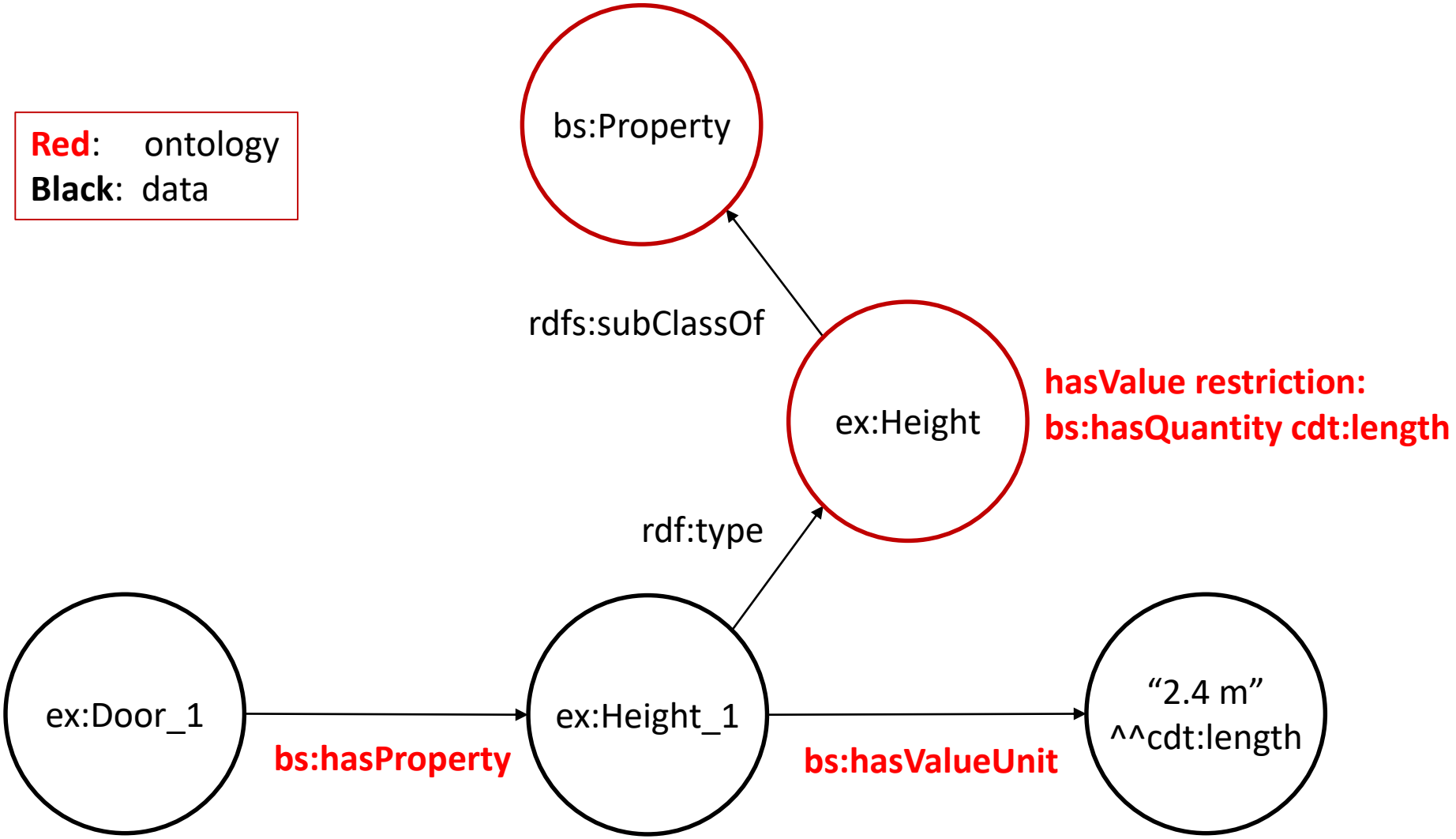
```
:Door_1 rdf:type :Door ;
        opm:hasProperty :Door_1_height ;
        opm:hasProperty :Door_1_materialType .
:Door_1_height rdf:type opm:Property ;
        opm:hasPropertyState :Door_1_height-value_1 .
:Door_1_materialType rdf:type opm:Property ;
        opm:hasPropertyState :Door_1_material-value_1 .
:Door_1_height-value_1 rdf:type opm:PropertyState ;
        schema:value “2.40 m”^^cdt:length .
:Door_1_materialType-value_1 rdf:type opm:PropertyState ;
        schema:value :Wood .
```

Choices for the CEN SMLS standard

- Try to stay Level 1 for simplicity
- If really needed go primary for Level 2
  - And derive Level 1 as secondary combination
- Extension needed for modelling:
  - Property Types
  - 3 Options for Level 2
    1. Via subclass of Property (attribute values become restrictions)
       - asserted attribute values become restrictions
       - Property Type groupis via superclasses
    2. Via "punning": meta-class with Subclasses of Property as instances (attributes for meta-instances) > too complex, not worked out
    3. Via separate, related PropertyType, worked out below
       - + :hasQuantity (rdfs:Datatype)
       - Property Type Groups via membership relation

- Explicit unit and value (as alternative for WKT option) when Level 2:
  - hasValueUnit (WKT of datatype cdt:ucum quantity like cdt:length)
  - hasValue (xsd:decimal)
  - hasUnit (xsd:string)
- Prefix bs (from basic semantics)

Option 1 - Example



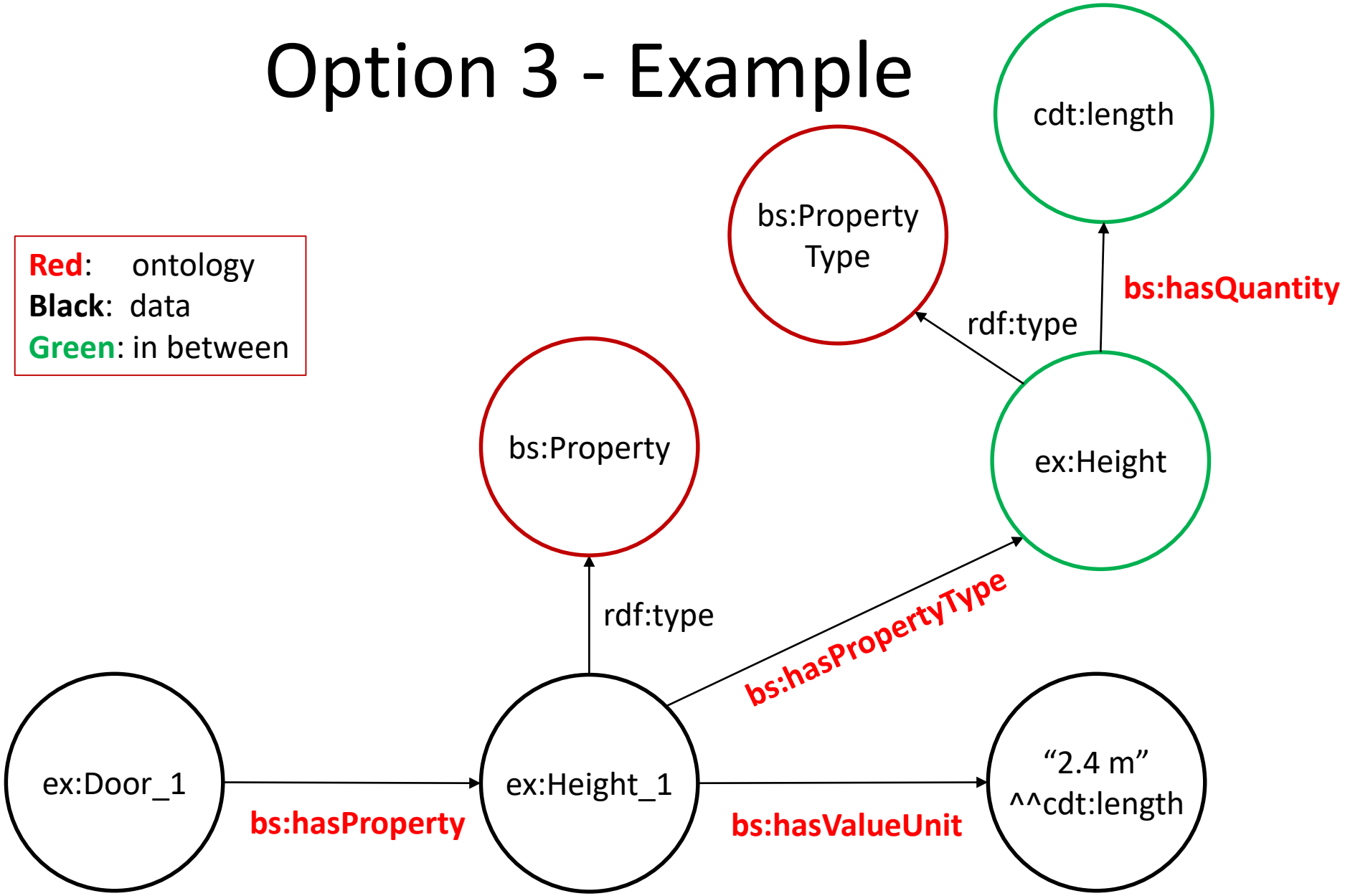
Option 1: Example Data

```
ex:Height rdfs:subClassOf bs:Property ;
        rdfs:subClassOf [ rdf:type owl:Restriction ;
        owl:hasValue “cdt:length” ;
        owl:onProperty bs:hasQuantity ; ] .

ex:Door_1 rdf:type ex :Door ;
        bs:hasProperty ex:Height_1 .

ex:Height_1 rdf:type ex:Height ;
        bs:hasValueUnit “2.40 m”^^cdt:length .
```

Option 1: Grouping

To be done!

Option 3 - Example



Option 3: Example Data

```
ex:Height rdf:type bs:PropertyType ;
        bs:hasQuantity cdt:length .

ex:Door_1 rdf:type ex :Door ;
        bs:hasProperty ex:Height_1 .

ex:Height_1 rdf:type bs:Property ;
        bs:hasPropertyType ex:Height ;
        bs:hasValueUnit “2.40 m”^^cdt:length .
```

Option 3: Property Type Grouping

- Added
  - PropertyTypeGroup class
  - memberOfPropertyTypeGroup object property
  - hasPropertyTypeGroup object property

- Property grouping implicit via type-level

Option3: Extended Example Data

```
ex:Height rdf:type bs:PropertyType ;
        bs:hasQuantity cdt:length .
ex:ClearOpeningHeight rdf:type bs:PropertyType ;
        rdfs:subClassOf ex:Height ;
        bs:isMemberOfPropertyTypeGroup ex:WindowGeometricProperties .
ex:WindowGeometricProperties rdf:type bs:PropertyTypeGroup .

ex:Door_1 rdf:type ex :Door ;
        rdfs:subClassOf bs:PhysicalObject ;
        bs:hasProperty ex:ClearOpeningHeight_1 .
ex:Height_1 rdf:type bs:Property ;
        bs:hasPropertyType ex:ClearOpeningHeight ;
        bs:hasValueUnit “2.40 m”^^cdt:length ;
        bs:hasValue "2.40"^^xsd:decimal ;
        bs:hasUnit "m"^^xsd:string .
```

Choice

- Best approach to be discussed and decided at W3C LBD
- White paper to be written over Summer 2019
- Harmonized way for
  - W3C LBD
  - CEN TC442
  - bSI LDWG
- Combined with ‘Product’ modelling patterns?