DeepTelos

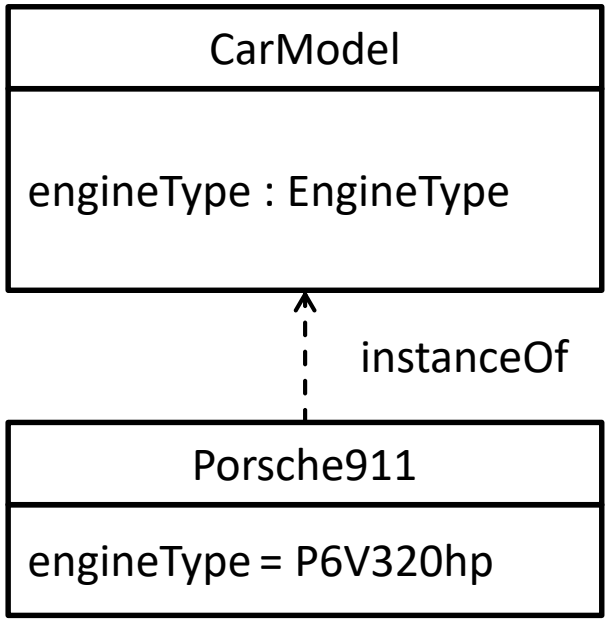
Multi-level Modeling with Most General Instances

Manfred Jeusfeld
University of Skövde, Sweden
Bernd Neumayr
Johannes Kepler University Linz
University of Oxford
FREQUENTIS AG

ER 2016, Gifu

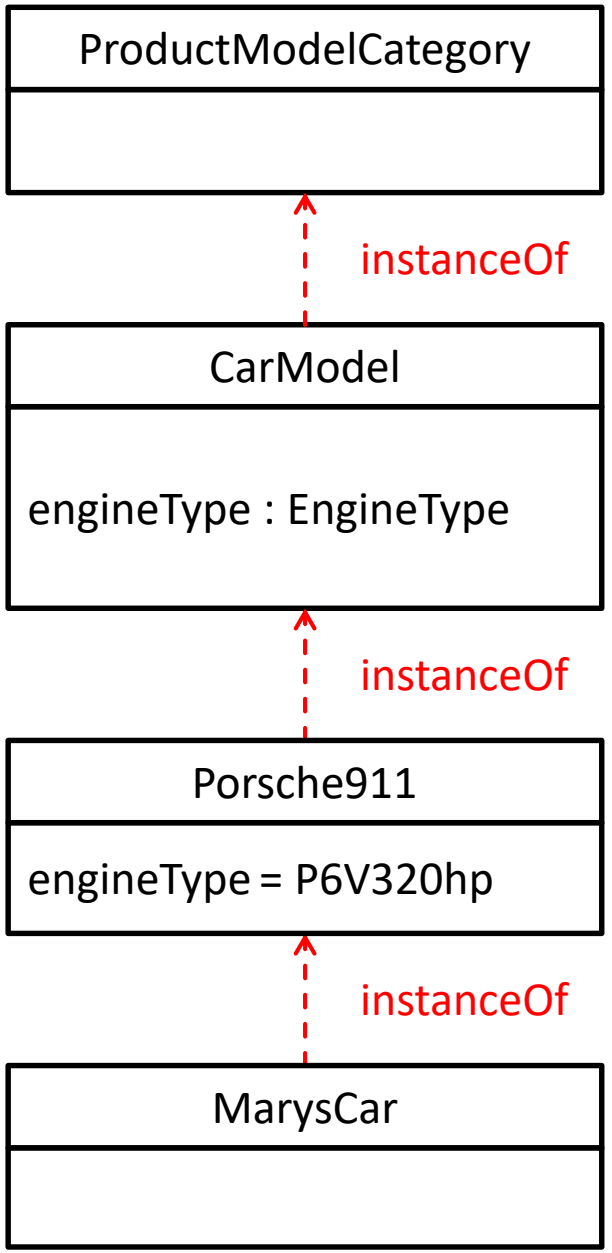
Multi-level Modeling

- extends object-oriented modeling



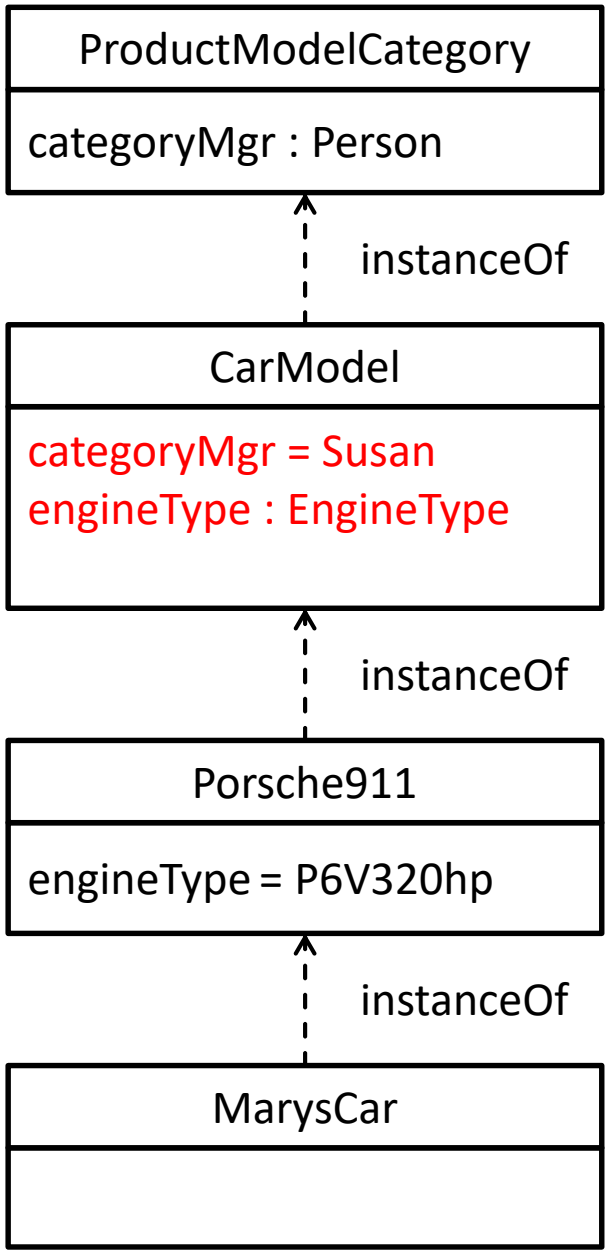
Multi-level Modeling

- extends object-oriented modeling with
  - unbounded levels of instantiation



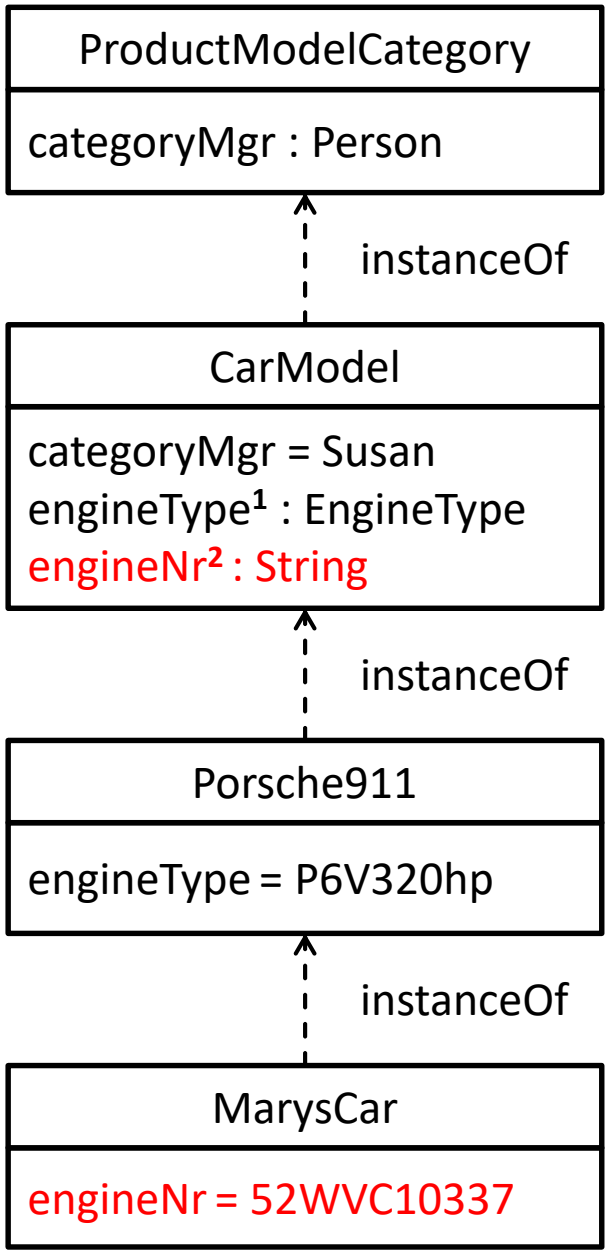
Multi-level Modeling

- extends object-oriented modeling with
  - unbounded levels of instantiation
  - clabjects combining class and object facets



Multi-level Modeling

- extends object-oriented modeling with
  - unbounded levels of instantiation
  - clabjects combining class and object facets
  - deep characterization



Motivation for DeepTelos

- Existing multi-level modeling approaches are arguably
  - firmly rooted in two-level modeling
  - overly complex and rather inflexible

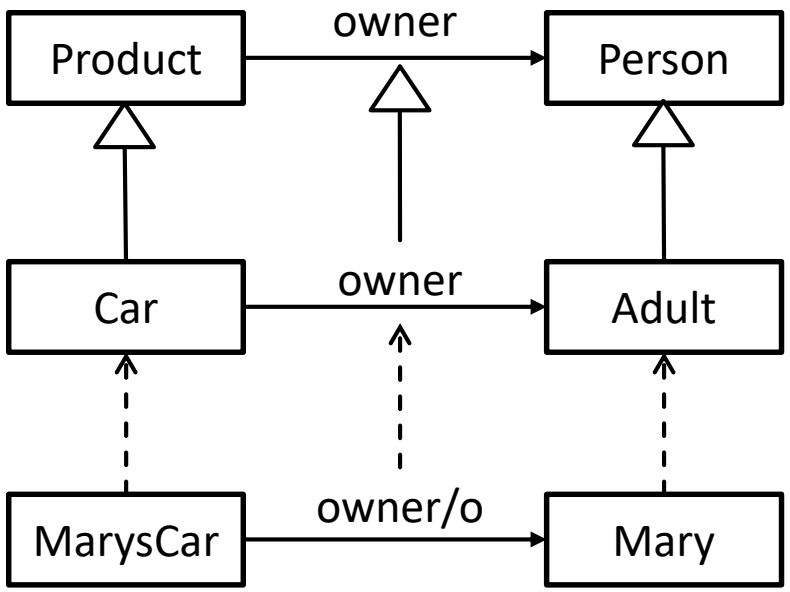
- DeepTelos is based on Telos[1] and its implementation ConceptBase[2] and inherits their
  - natural metamodeling facilities
  - uniform treatment of objects, classes, metaclasses, ...
  - uniform treatment of entities, attributes, and relationships
  - and remains simple and flexible

[1] John Mylopoulos, Alexander Borgida, Matthias Jarke, Manolis Koubarakis: Telos: Representing Knowledge About Information Systems. ACM Transactions on Information Systems 8(4): 325-362 (1990)

[2] Matthias Jarke, Rainer Gallersdörfer, Manfred A. Jeusfeld, Martin Staudt: ConceptBase - A Deductive Object Base for Meta Data Management. J. Intell. Inf. Syst. 4(2): 167-192 (1995)

Prerequisites:
Basics of Telos/ConceptBase

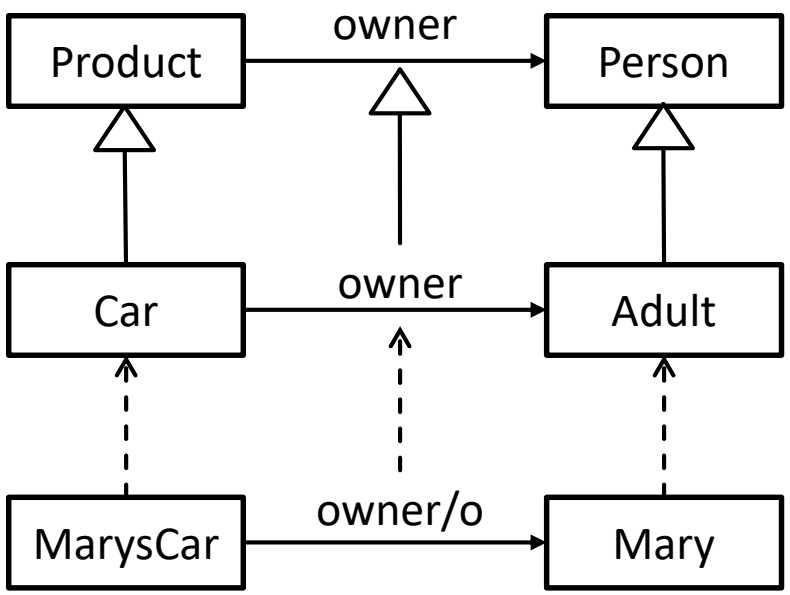
An example Telos model (without metamodeling)



Everything is represented as *Proposition*

P(<ID>,<source>,<label>,<target>)

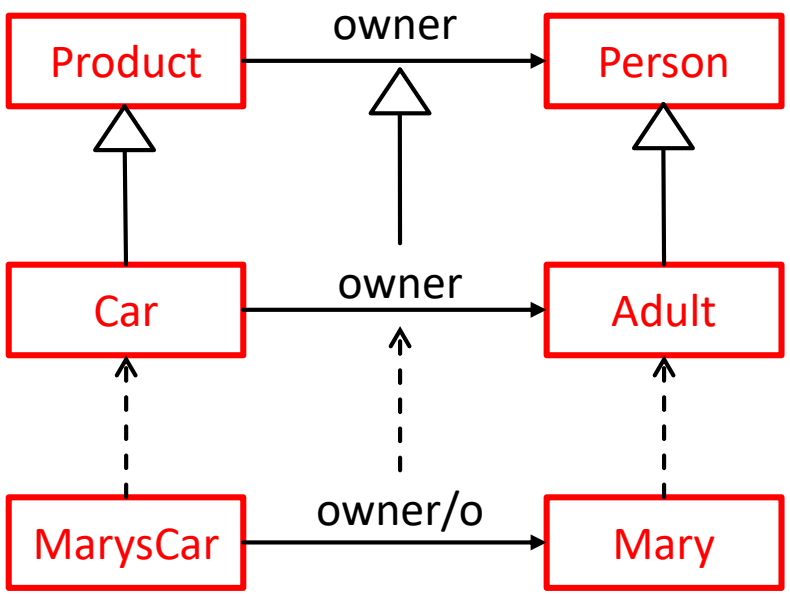
P(o0,o0,Product,o0).
P(o1,o1,Person,o1).
P(o2,o0,owner,o1).
P(o3,o3,Car,o3).
P(o4,o3,isa,o0).
P(o5,o5,Adult,o5).
P(o6,o5,isa,o1).
P(o7,o3,owner,o5).
P(o8,o7,isa,o2).
P(o9,o9,MarysCar,o9).
P(o10,o9,in,o3).
P(o11,o11,Mary,o11).
P(o12,o11,in,o5).
P(o13,o9,o,o11).
p(o14,o13,in,o7).



Four kinds of propositions

P(<ID>,<source>,<label>,<target>)

P(o0,o0,Product,o0).
P(o1,o1,Person,o1).
P(o2,o0,owner,o1).
P(o3,o3,Car,o3).
P(o4,o3,isa,o0).
P(o5,o5,Adult,o5).
P(o6,o5,isa,o1).
P(o7,o3,owner,o5).
P(o8,o7,isa,o2).
P(o9,o9,MarysCar,o9).
P(o10,o9,in,o3).
P(o11,o11,Mary,o11).
P(o12,o11,in,o5).
P(o13,o9,o,o11).
p(o14,o13,in,o7).

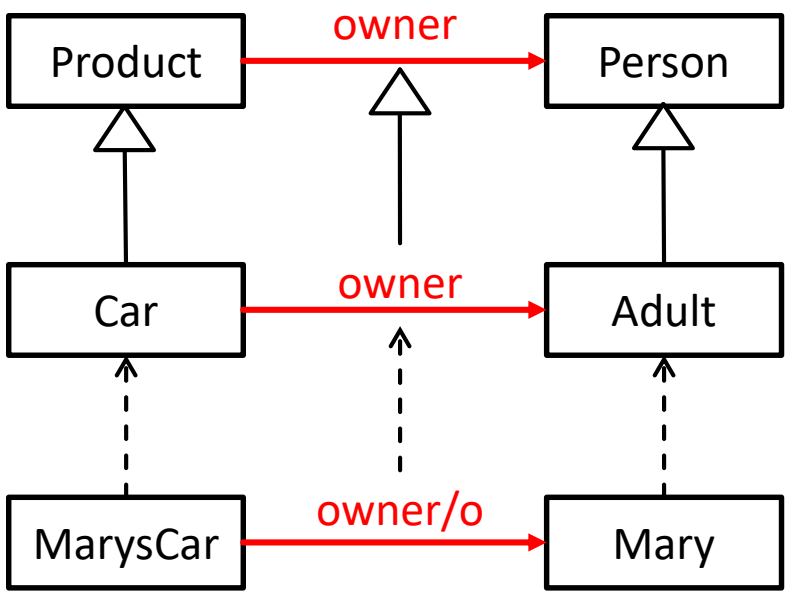


Individuals (Individual objects, Classes)

Four kinds of propositions

P(<ID>,<source>,<label>,<target>)

P(o0,o0,Product,o0).
P(o1,o1,Person,o1).
P(o2,o0,owner,o1).
P(o3,o3,Car,o3).
P(o4,o3,isa,o0).
P(o5,o5,Adult,o5).
P(o6,o5,isa,o1).
P(o7,o3,owner,o5).
P(o8,o7,isa,o2).
P(o9,o9,MarysCar,o9).
P(o10,o9,in,o3).
P(o11,o11,Mary,o11).
P(o12,o11,in,o5).
P(o13,o9,o,o11).
p(o14,o13,in,o7).

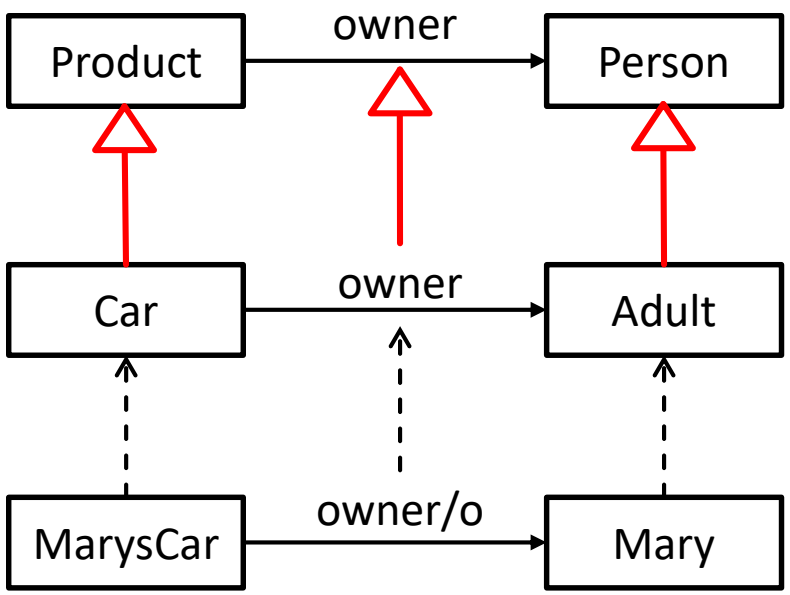


Attribute classes and attribute instances

Four kinds of propositions

P(<ID>,<source>,<label>,<target>)

P(o0,o0,Product,o0).
P(o1,o1,Person,o1).
P(o2,o0,owner,o1).
P(o3,o3,Car,o3).
P(o4,o3,isa,o0).
P(o5,o5,Adult,o5).
P(o6,o5,isa,o1).
P(o7,o3,owner,o5).
P(o8,o7,isa,o2).
P(o9,o9,MarysCar,o9).
P(o10,o9,in,o3).
P(o11,o11,Mary,o11).
P(o12,o11,in,o5).
P(o13,o9,o,o11).
p(o14,o13,in,o7).

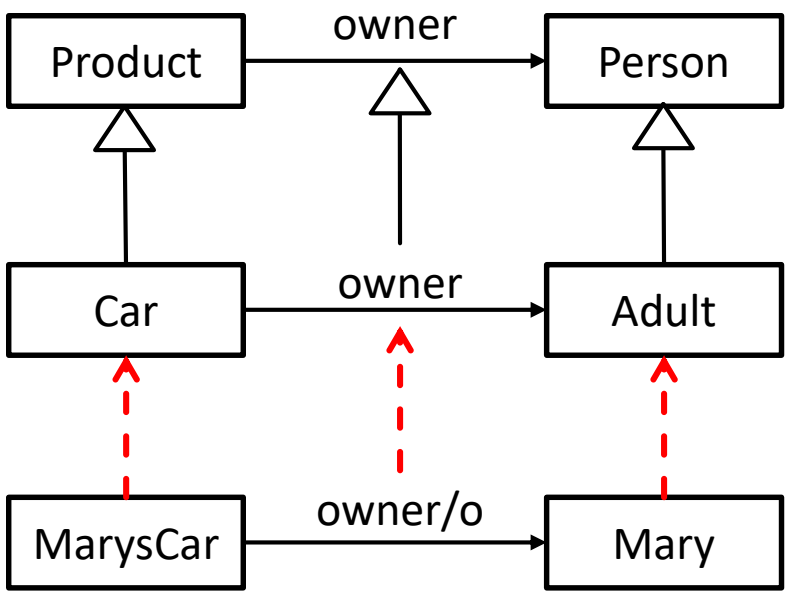


Specializations
(Multiple Specialization)

Four kinds of propositions

P(<ID>,<source>,<label>,<target>)

P(o0,o0,Product,o0).
P(o1,o1,Person,o1).
P(o2,o0,owner,o1).
P(o3,o3,Car,o3).
P(o4,o3,isa,o0).
P(o5,o5,Adult,o5).
P(o6,o5,isa,o1).
P(o7,o3,owner,o5).
P(o8,o7,isa,o2).
P(o9,o9,MarysCar,o9).
P(o10,o9,in,o3).
P(o11,o11,Mary,o11).
P(o12,o11,in,o5).
P(o13,o9,o,o11).
p(o14,o13,in,o7).

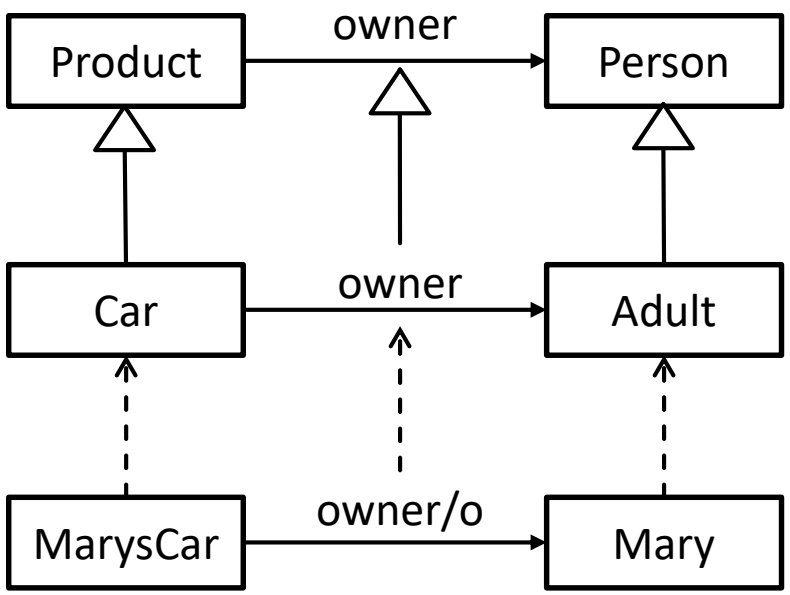


Instantiations
(Multiple Instantiation)

Names and IDs

P(<ID>,<source>,<label>,<target>)

P(o0,o0,Product,o0).
P(o1,o1,Person,o1).
P(o2,o0,owner,o1).
P(o3,o3,Car,o3).
P(o4,o3,isa,o0).
P(o5,o5,Adult,o5).
P(o6,o5,isa,o1).
P(o7,o3,owner,o5).
P(o8,o7,isa,o2).
P(o9,o9,MarysCar,o9).
P(o10,o9,in,o3).
P(o11,o11,Mary,o11).
P(o12,o11,in,o5).
P(o13,o9,o,o11).
p(o14,o13,in,o7).

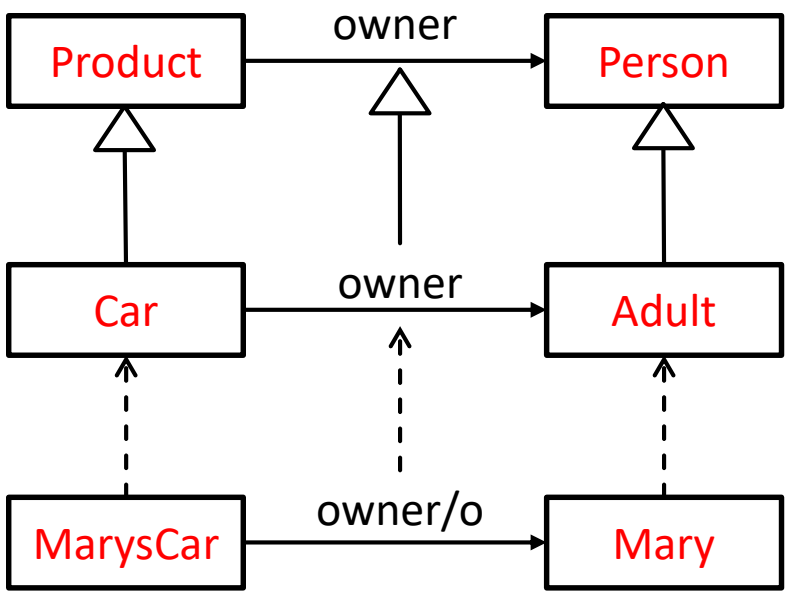


Object identifiers are unique.

Names and IDs

P(<ID>,<source>,<label>,<target>)

P(o0,o0,Product,o0).
P(o1,o1,Person,o1).
P(o2,o0,owner,o1).
P(o3,o3,Car,o3).
P(o4,o3,isa,o0).
P(o5,o5,Adult,o5).
P(o6,o5,isa,o1).
P(o7,o3,owner,o5).
P(o8,o7,isa,o2).
P(o9,o9,MarysCar,o9).
P(o10,o9,in,o3).
P(o11,o11,Mary,o11).
P(o12,o11,in,o5).
P(o13,o9,o,o11).
p(o14,o13,in,o7).

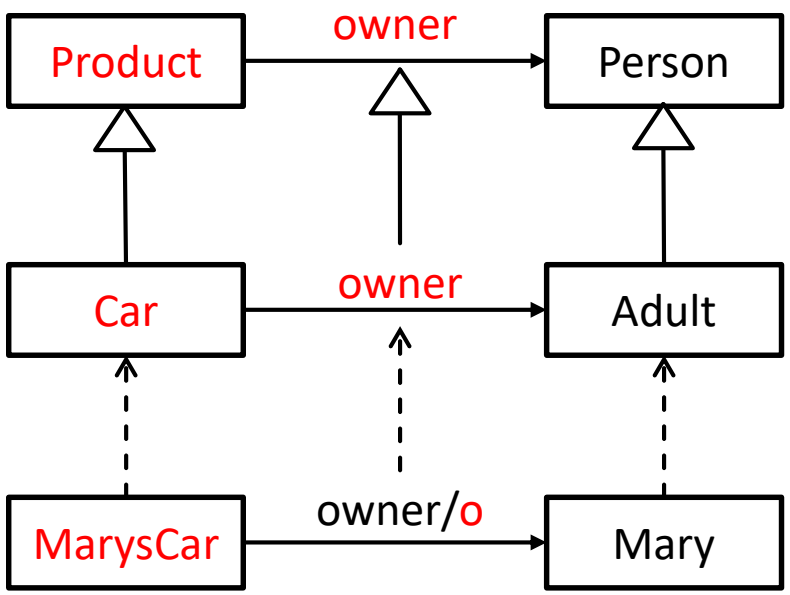


Names of individual are unique

Names and IDs

P(<ID>,<source>,<label>,<target>)

P(o0,o0,Product,o0).
P(o1,o1,Person,o1).
P(o2,o0,owner,o1).
P(o3,o3,Car,o3).
P(o4,o3,isa,o0).
P(o5,o5,Adult,o5).
P(o6,o5,isa,o1).
P(o7,o3,owner,o5).
P(o8,o7,isa,o2).
P(o9,o9,MarysCar,o9).
P(o10,o9,in,o3).
P(o11,o11,Mary,o11).
P(o12,o11,in,o5).
P(o13,o9,o,o11).
p(o14,o13,in,o7).



Names of attributes are unique in conjunction with the source object.

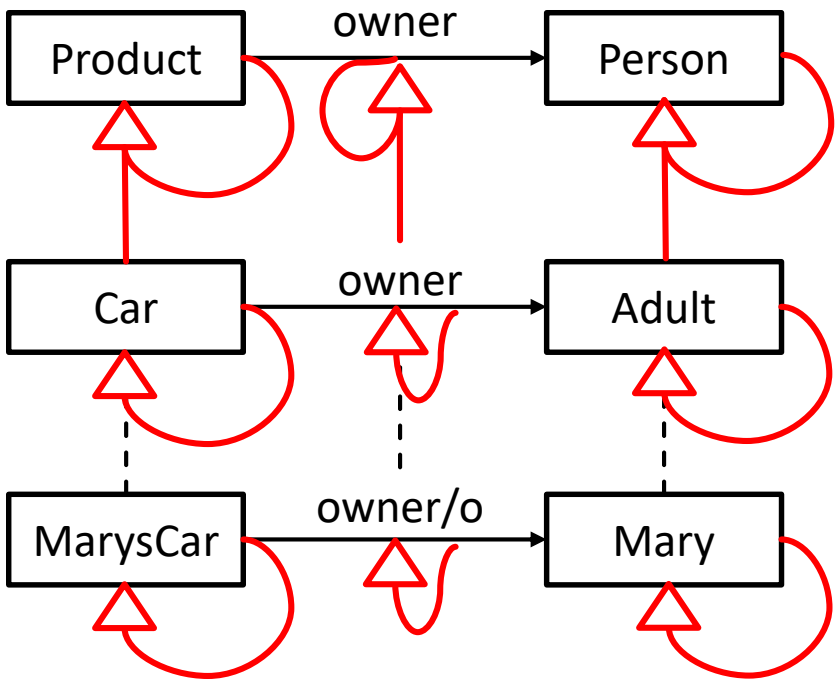
Derived Specialization Predicate

$$\forall o, x, c : P(o, c, isa, d) \Rightarrow Isa(c, d)$$

$$\forall c, d, e : Isa(c, d), Isa(d, e) \Rightarrow Isa(c, e)$$

$$\forall c : In(c, \#Obj) \Rightarrow Isa(c, c)$$

= Reflexive-transitive closure of specialization propositions

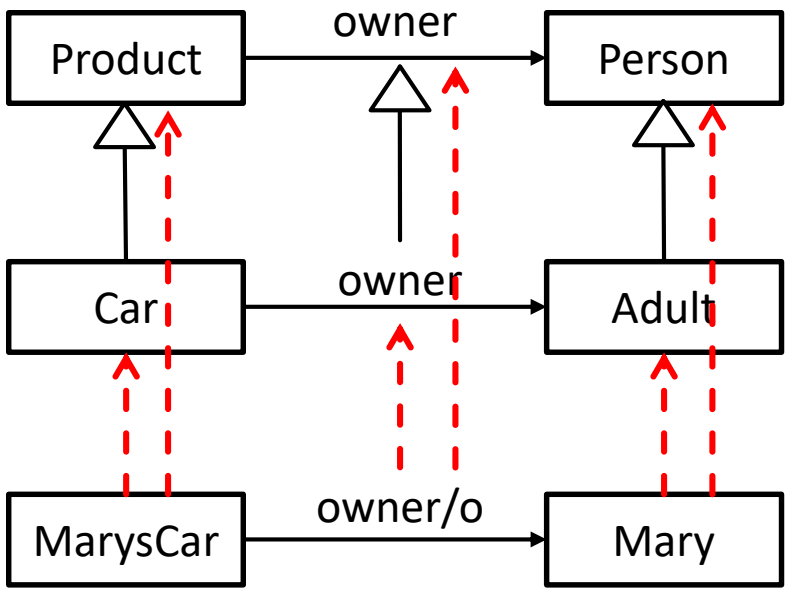


Derived Instantiation Predicate

$$\forall o, x, c : P(o, x, in, c) \Rightarrow In(x, c)$$

$$\forall x, c, d: In(x, c) \wedge Isa(c, d) \Rightarrow In(x, d)$$

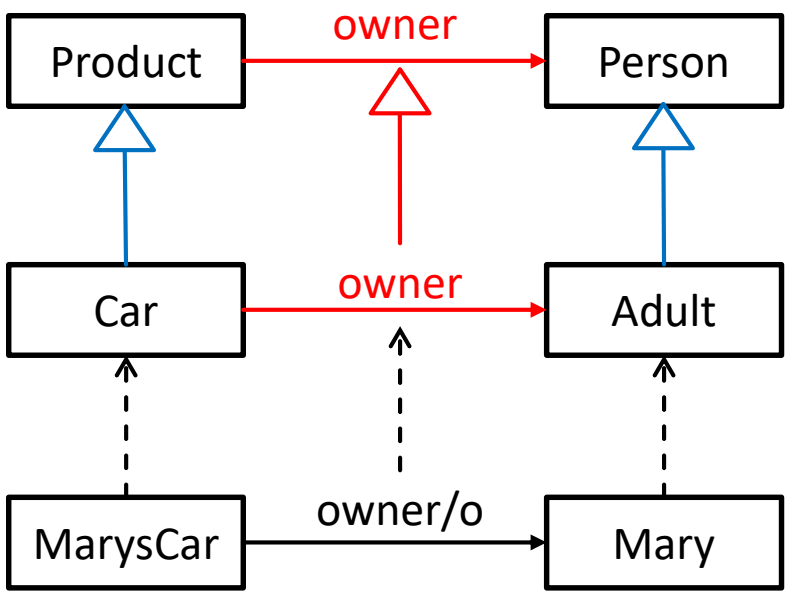
Class membership of objects is inherited upwardly to the superclasses.



Constraint on Attribute Specialization

Specialization of relationships, requires that source and target are specialized (with specialization being reflexive/transitive)

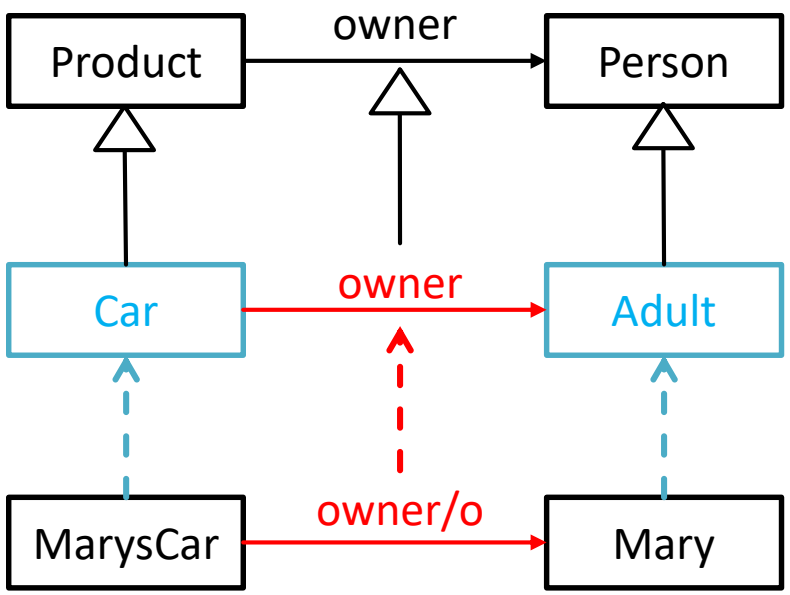
$$\forall s, s', a, a', m, m', t, t':$$
$$Isa(a', a) \wedge P(a, s, m, t) \wedge P(a', s', m', t')$$
$$\Rightarrow Isa(s', s) \wedge Isa(t', t).$$



Constraint on Attribute Instantiation

Instantiation of relationships, requires that source and target are instantiated

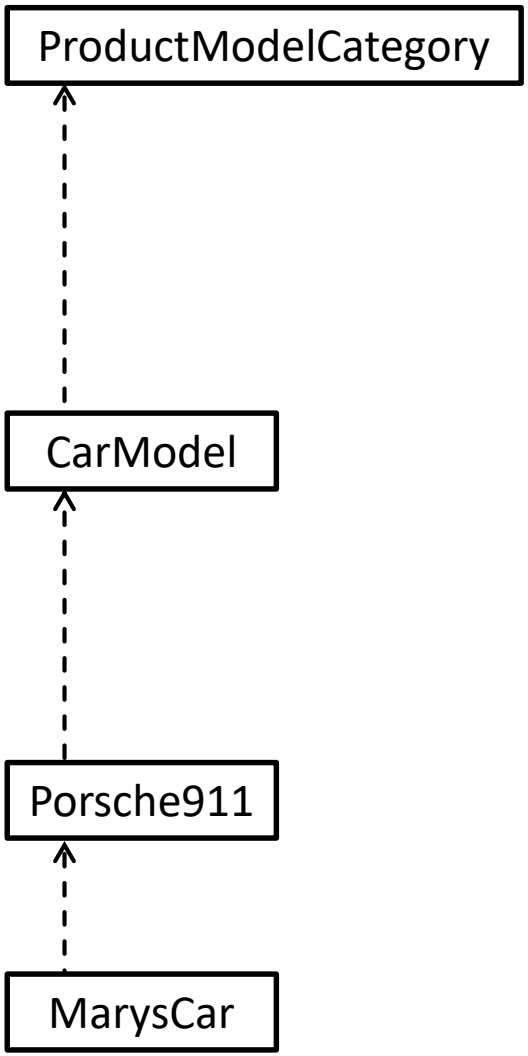
$$\forall o, x, n, y, p: P(o, x, n, y) \wedge In(o, p) \Rightarrow$$
$$\exists c, m, d: P(p, c, m, d) \wedge In(x, c) \wedge In(y, d)$$



Metamodeling in Telos/ConceptBase (without extension)

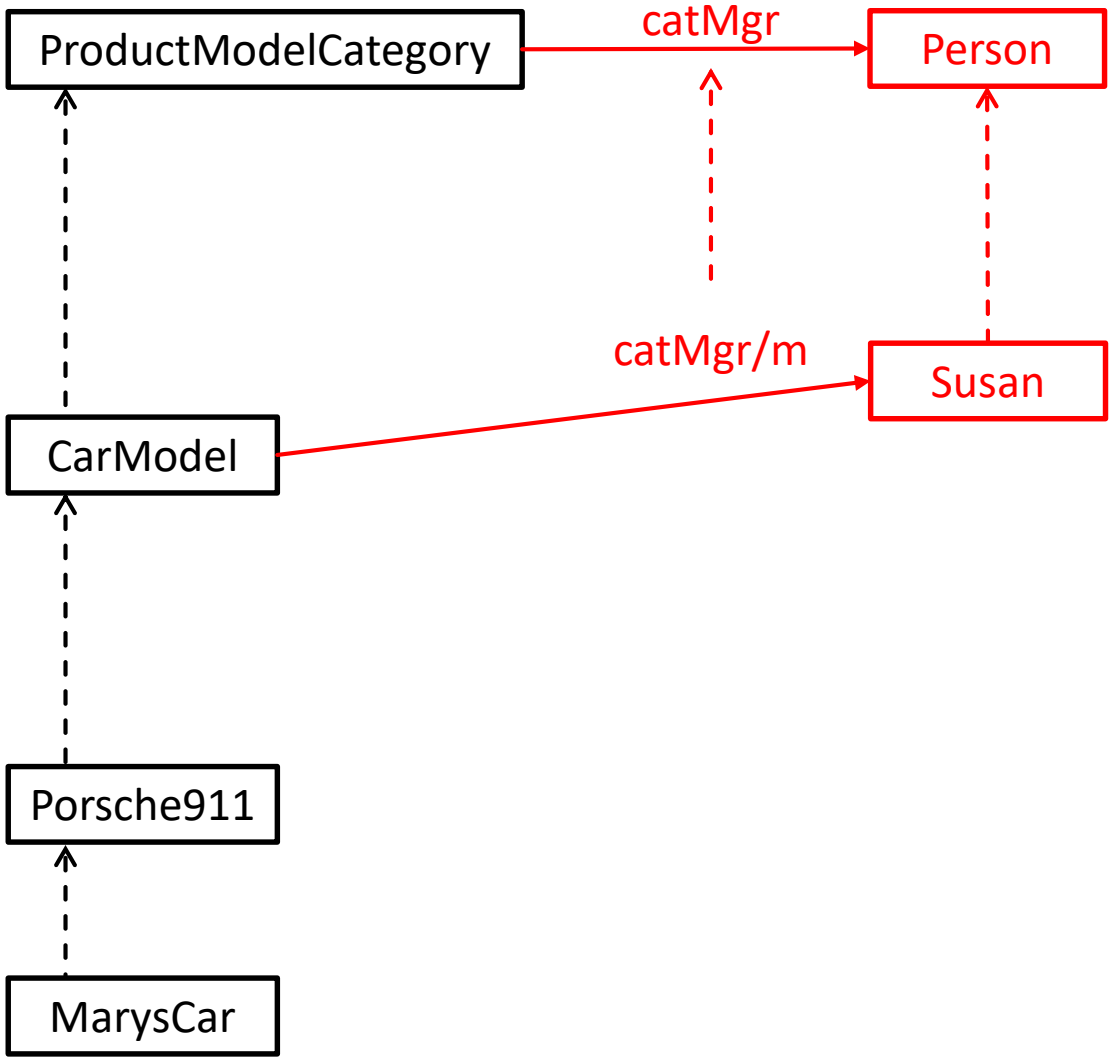
Unbounded Classification Hierarchies

Individuals act as classes,
metaclasses,
metametaclasses ...



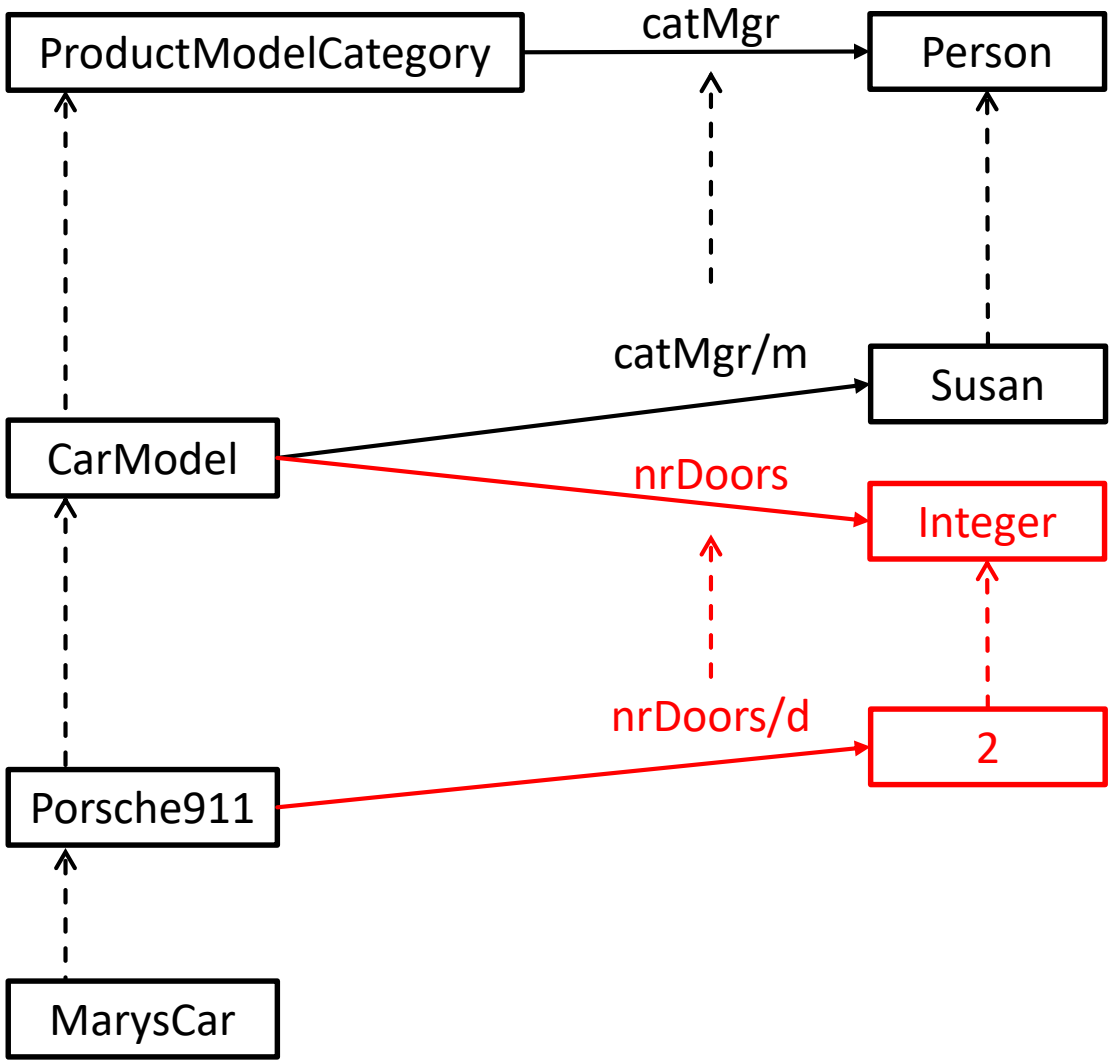
Clabjects

Individuals in multi-level hierarchies introduce attribute classes and instantiate attributes



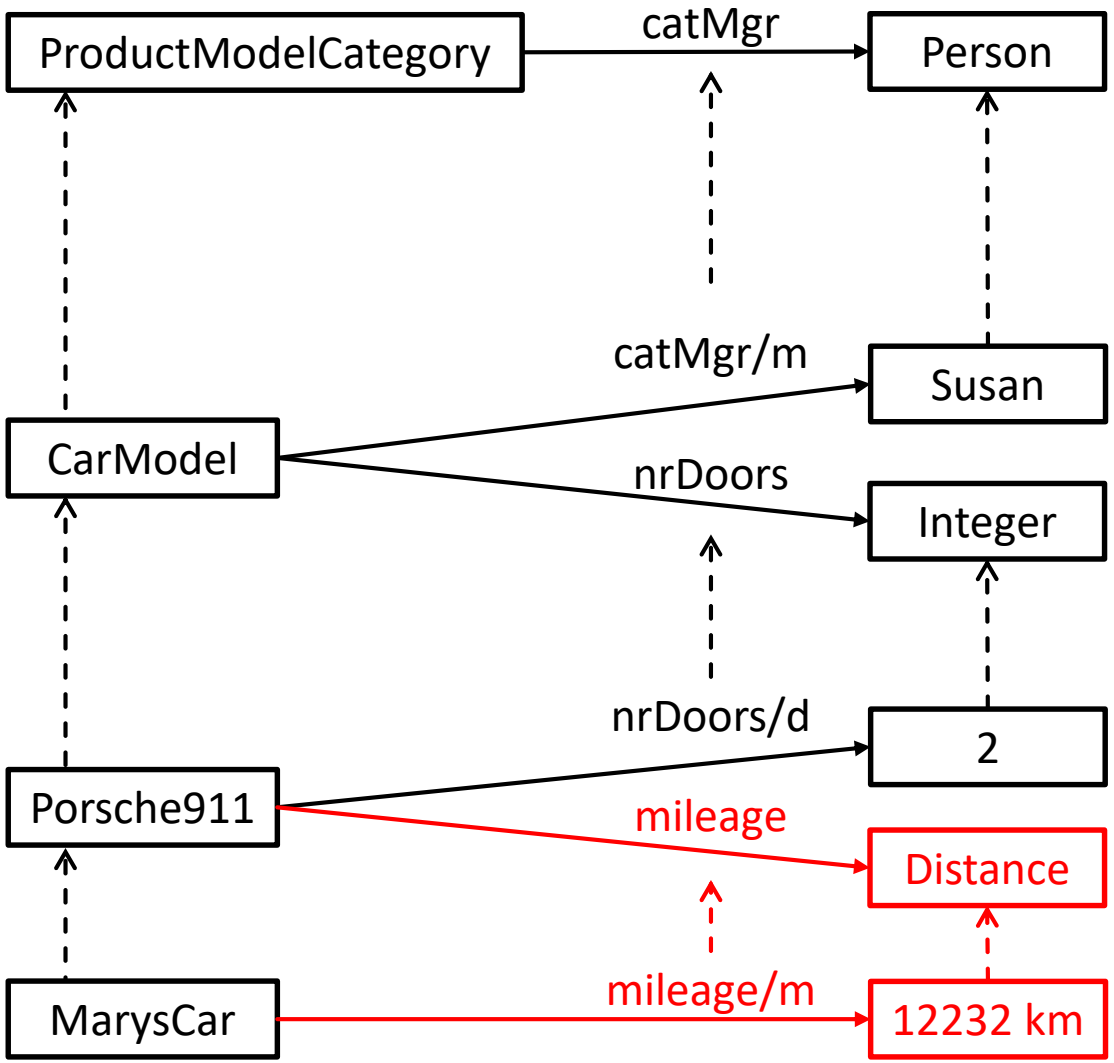
Clabjects

Individuals in multi-level hierarchies introduce attribute classes and instantiate attributes



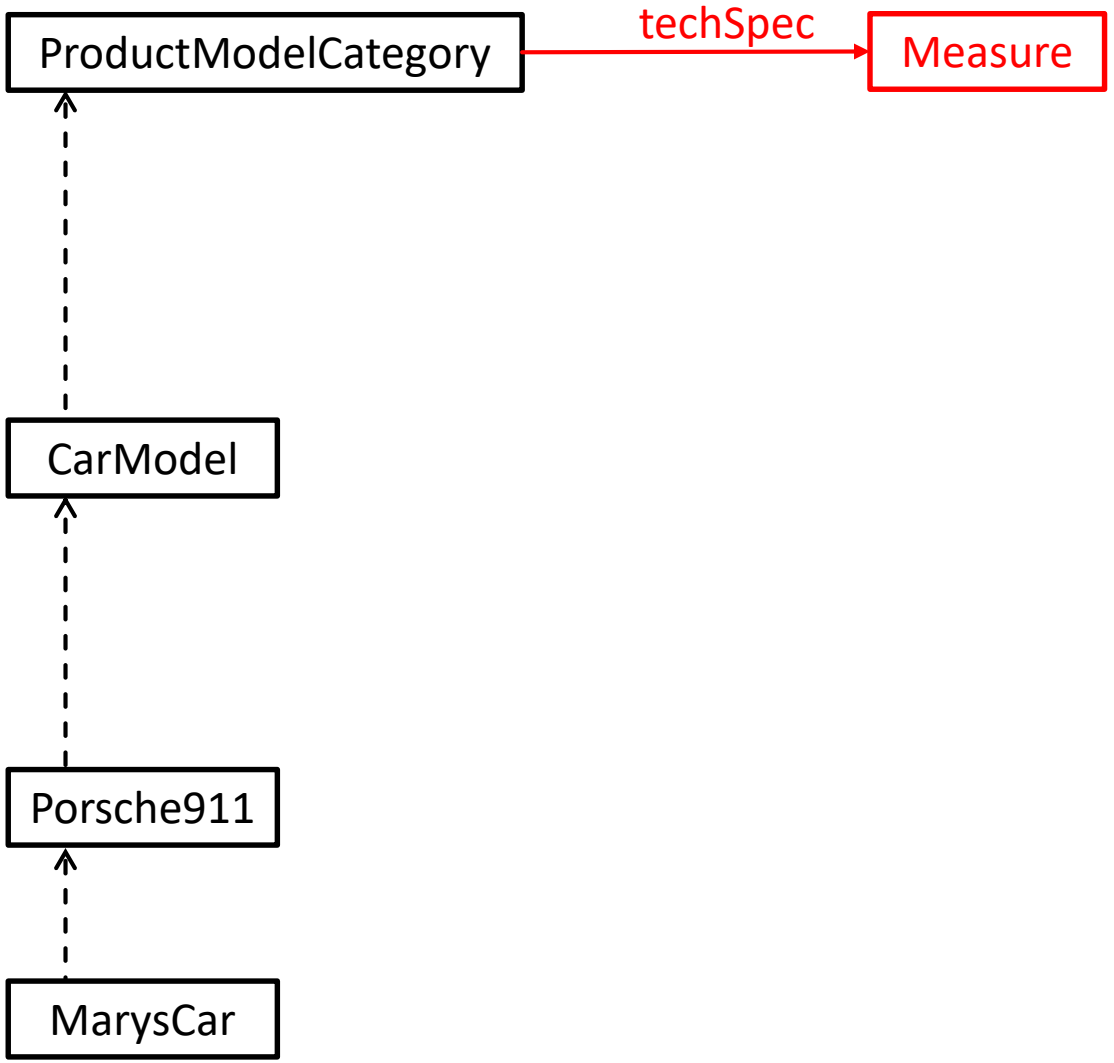
Clabjects

Individuals in multi-level hierarchies introduce attribute classes and instantiate attributes



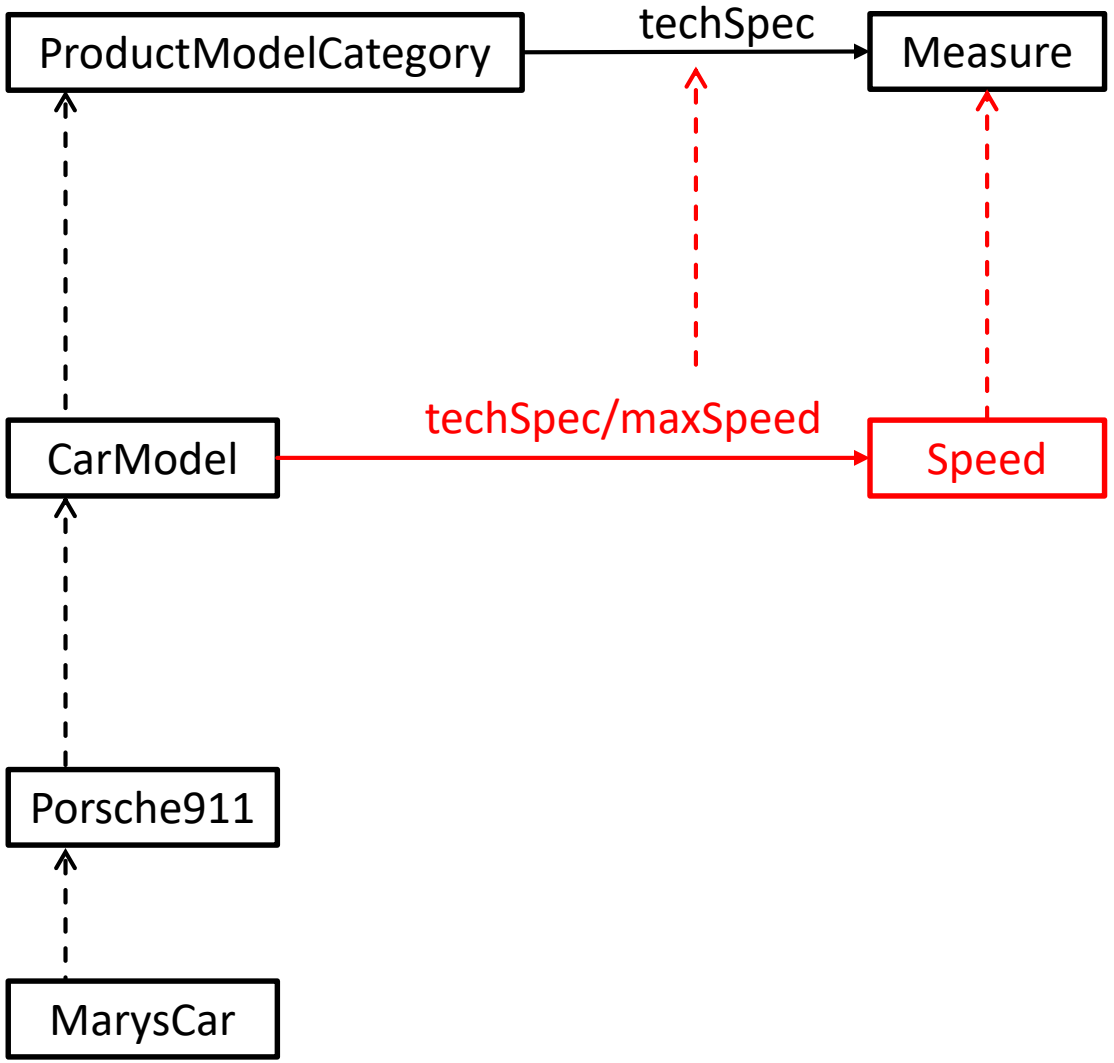
Attribute Metaclasses

Attribute metaclasses have metaclasses as target.



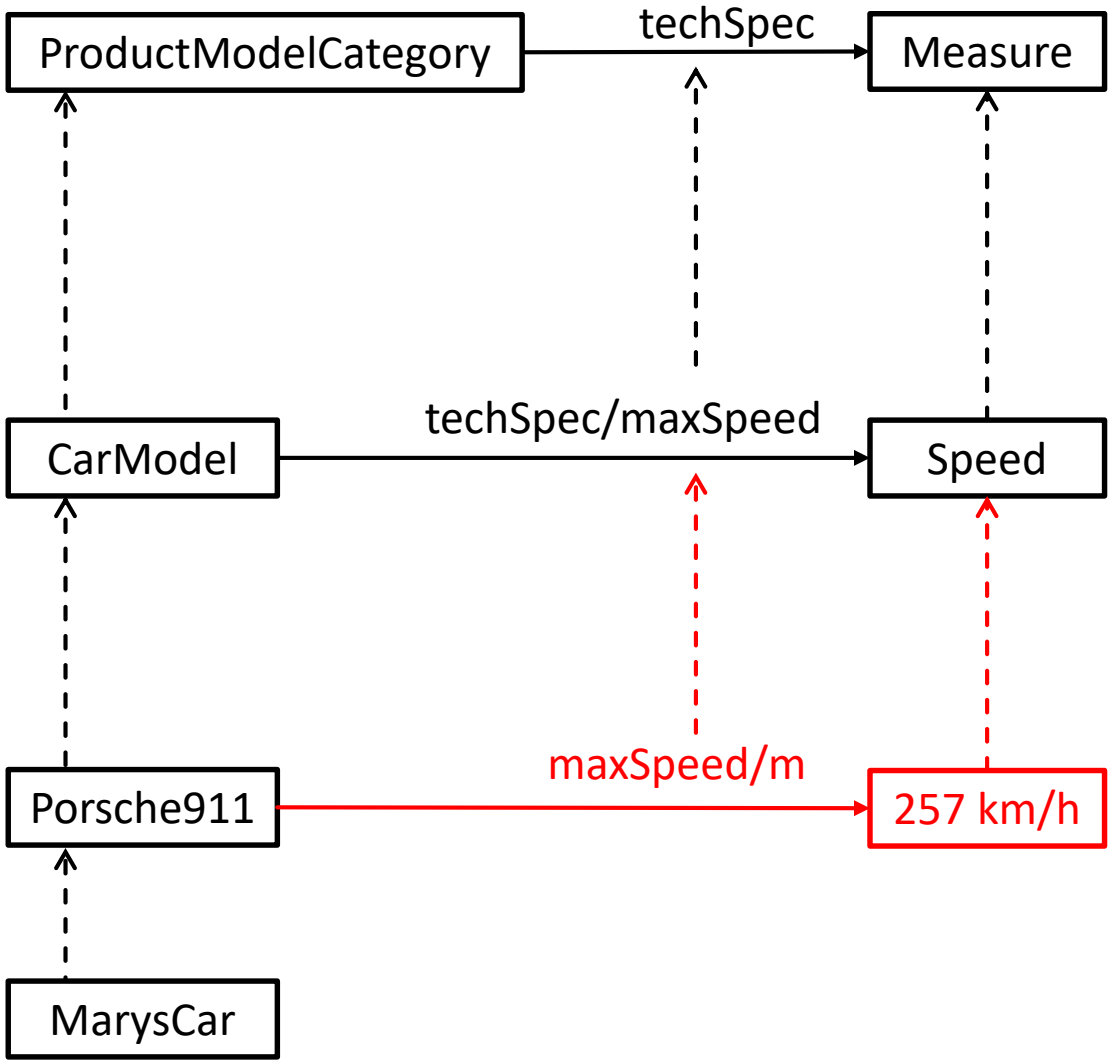
Attribute Metaclasses

Attribute metaclasses are instantiated by attribute classes



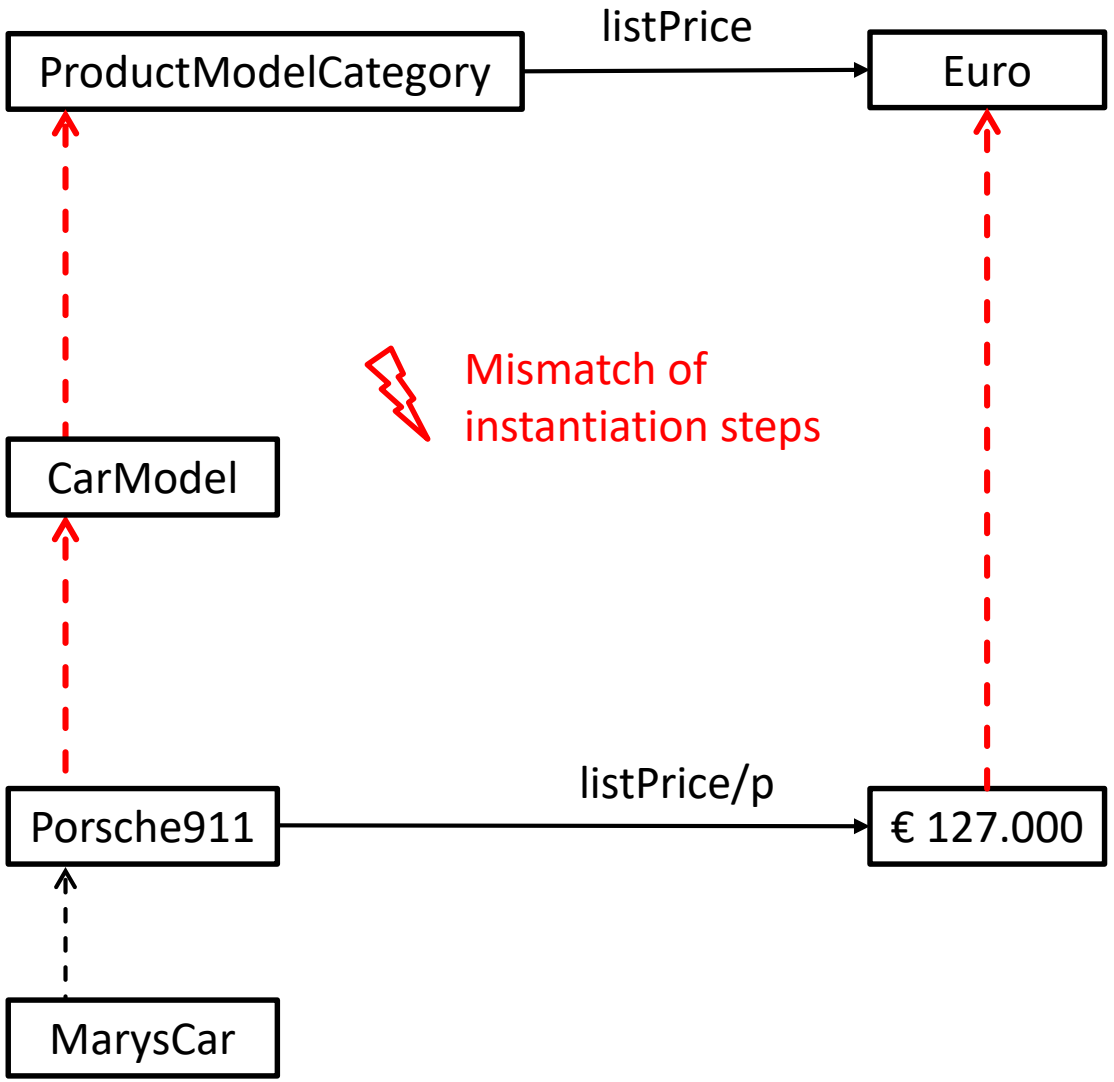
Attribute Metaclasses

... which in turn are instantiated by attribute values



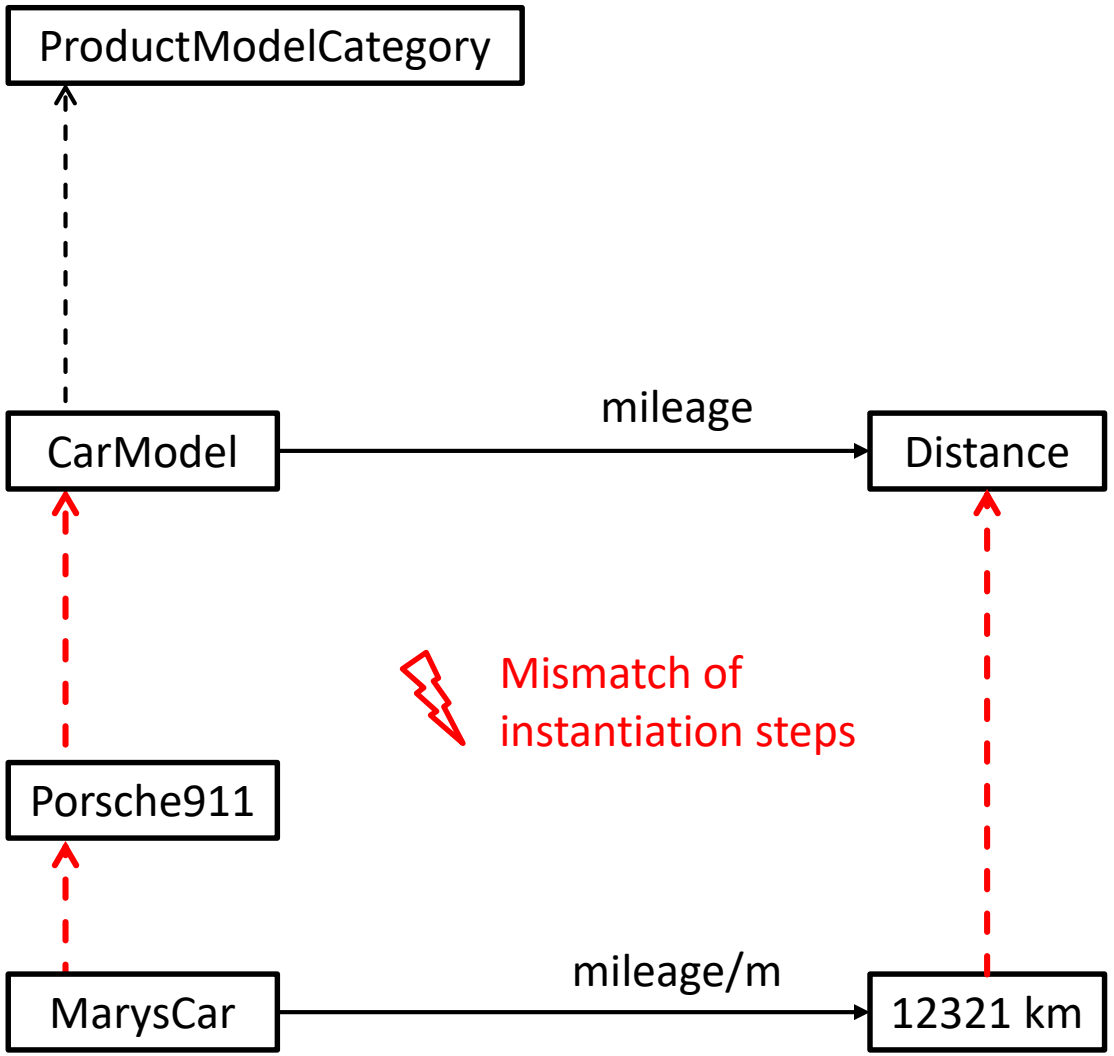
What is missing?

How to specify an attribute class **listPrice** that is instantiated by instance-instances of ProductModelCategory?



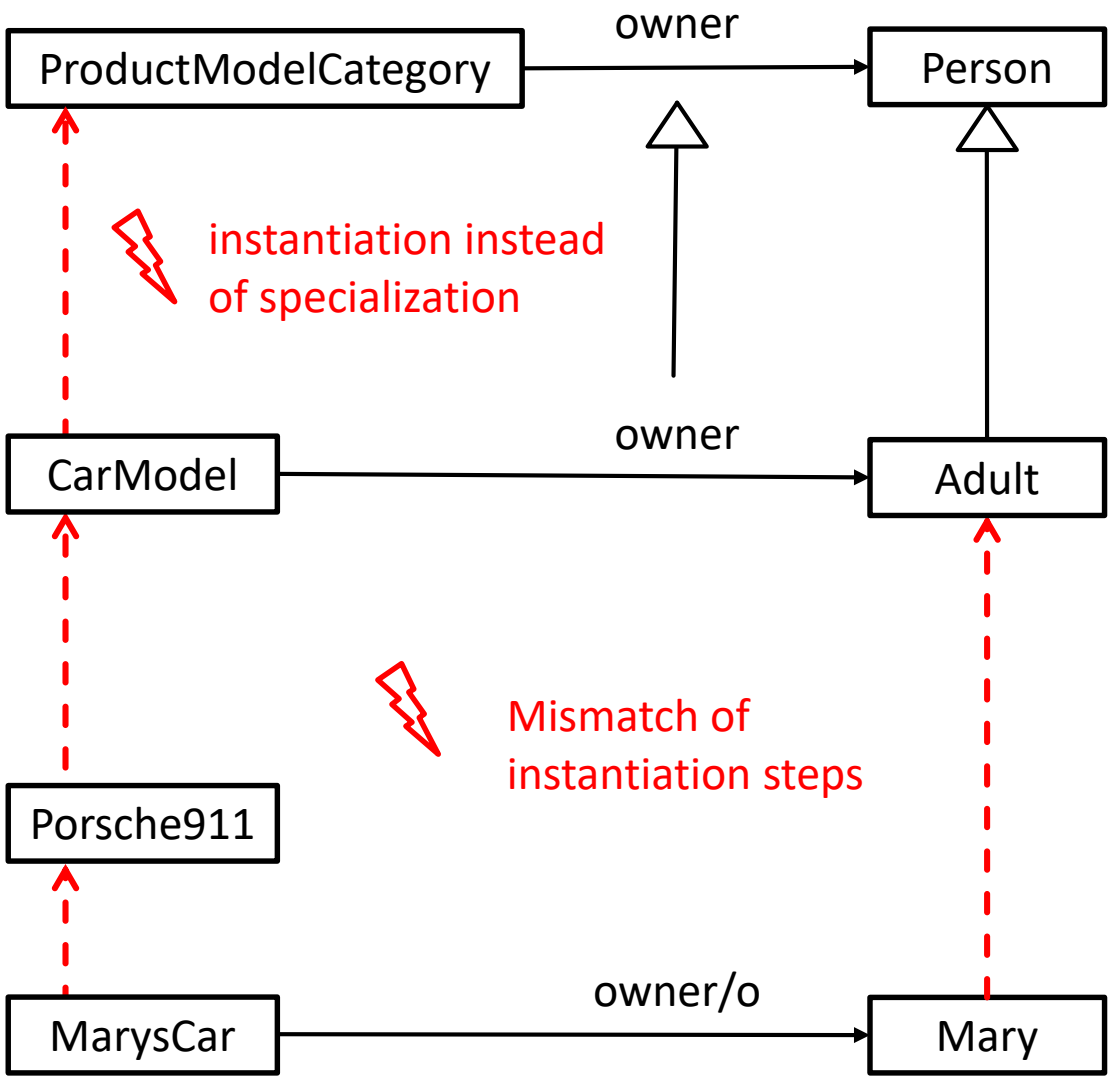
What is missing?

How to specify an attribute class **mileage** that is instantiated by instance-instances of CarModel?



What is missing?

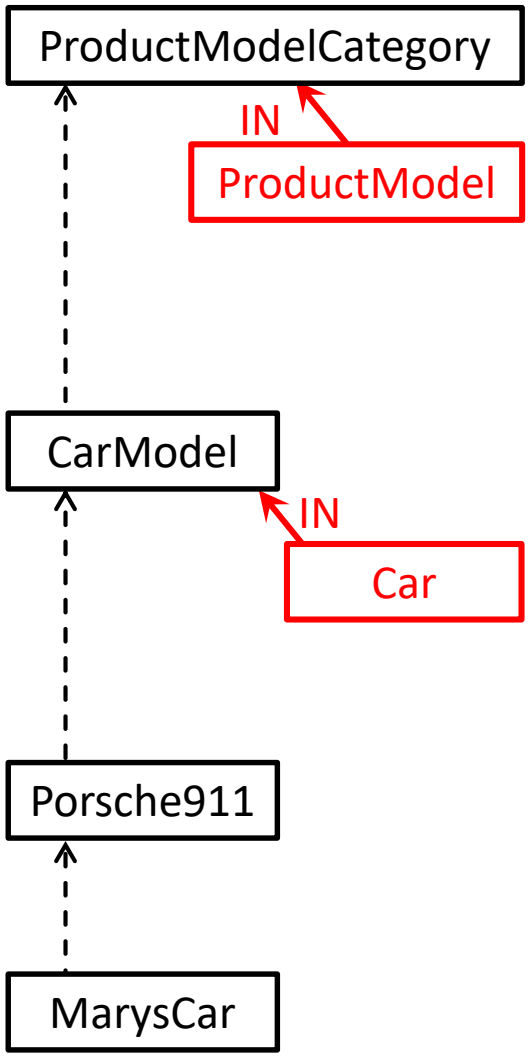
How to specify an attribute class **owner** that is instantiated by instance-instance-instances of ProductModelCategory and which is specialized for instance-instances of CarModel?



Extending Telos for Deep Characterization

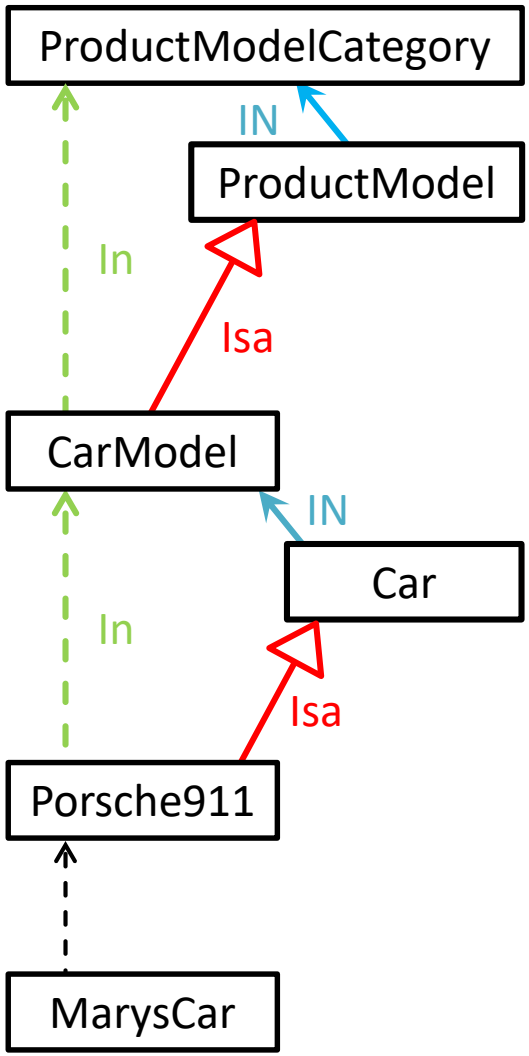
Most General Instances (MGIs)

A metaclass may have a class as most general instance.



Derived Subclasses of MGIs

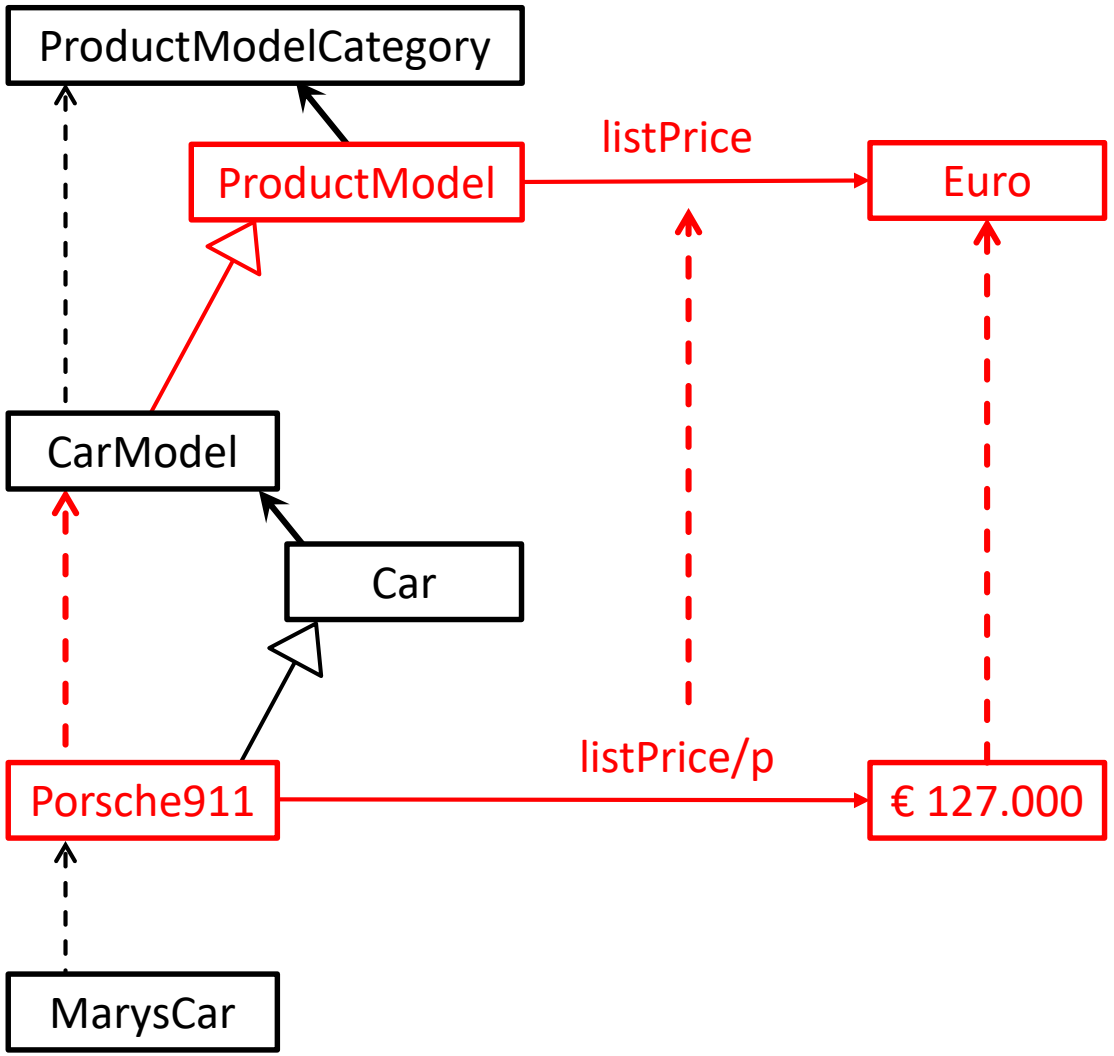
All instances of the metaclass are (by inference) specializations of the most general instance of the metaclass



$$\forall x, c, m : In(x, c) \wedge IN(m, c) \Rightarrow Isa(x, m)$$

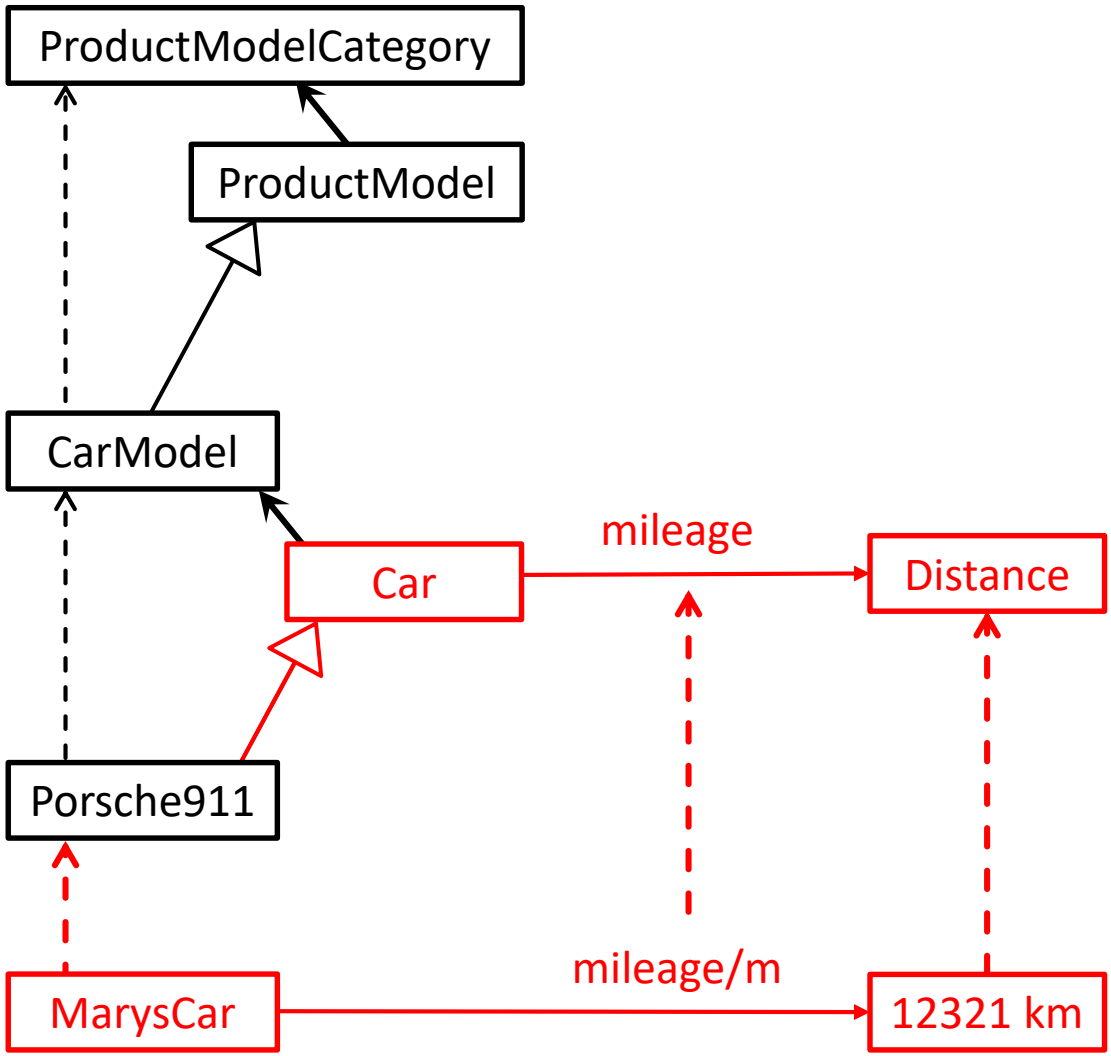
Deep Characterization

Attribute class **listPrice** is specified with the most-general instance of ProductModelCategory and is instantiated by instance-instances of ProductModelCategory



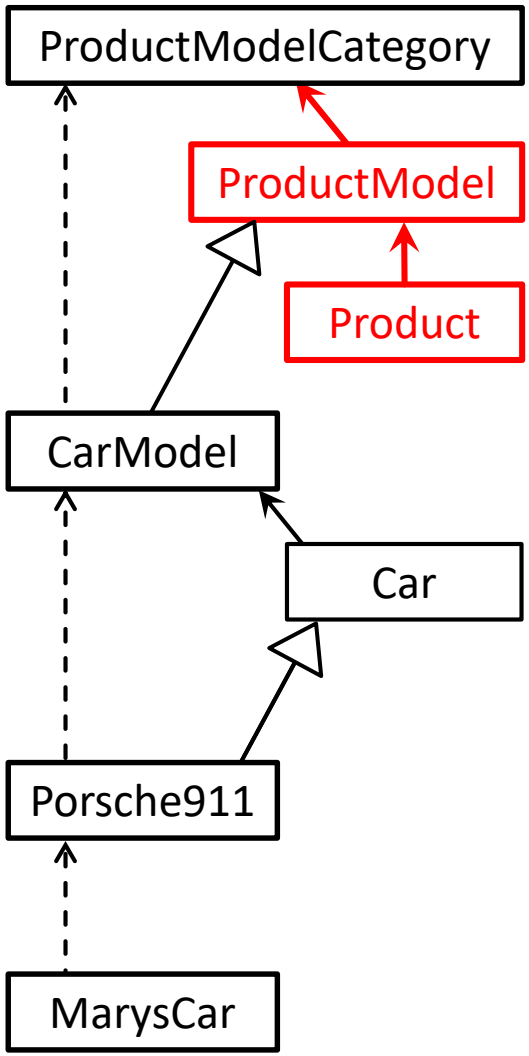
Deep Characterization

Attribute class **mileage** is specified with Car (the most-general instance of CarModel) and instantiated by instance-instances of CarModel.



Chains of MGIs

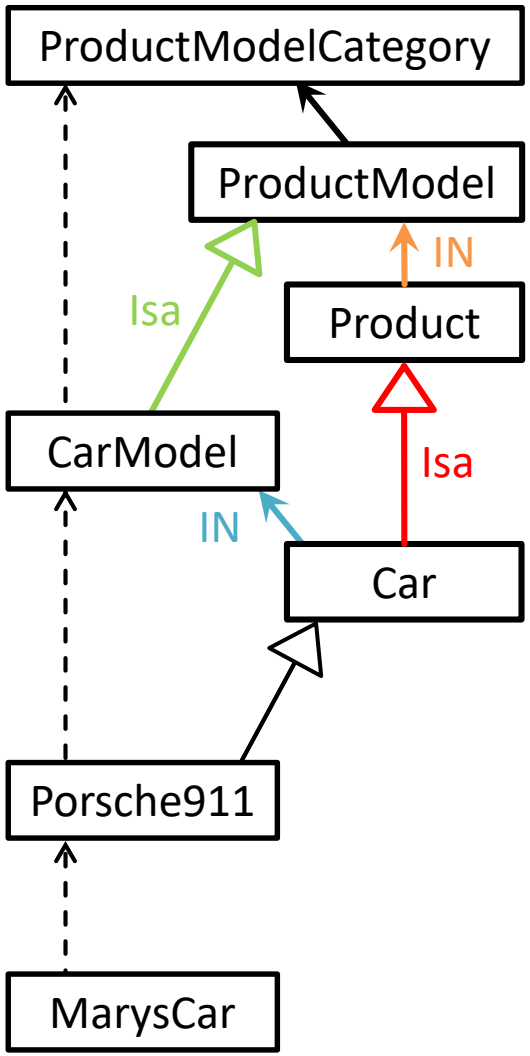
A most-general instance ***d*** may in turn have a most-general instance.



Chains of MGIs and Derived Subclasses

A most-general instance ***d*** may in turn have a most-general instance.

Most-general instances of specializiations of ***d*** are (by inference) specializations of the most-general instance of ***d***.



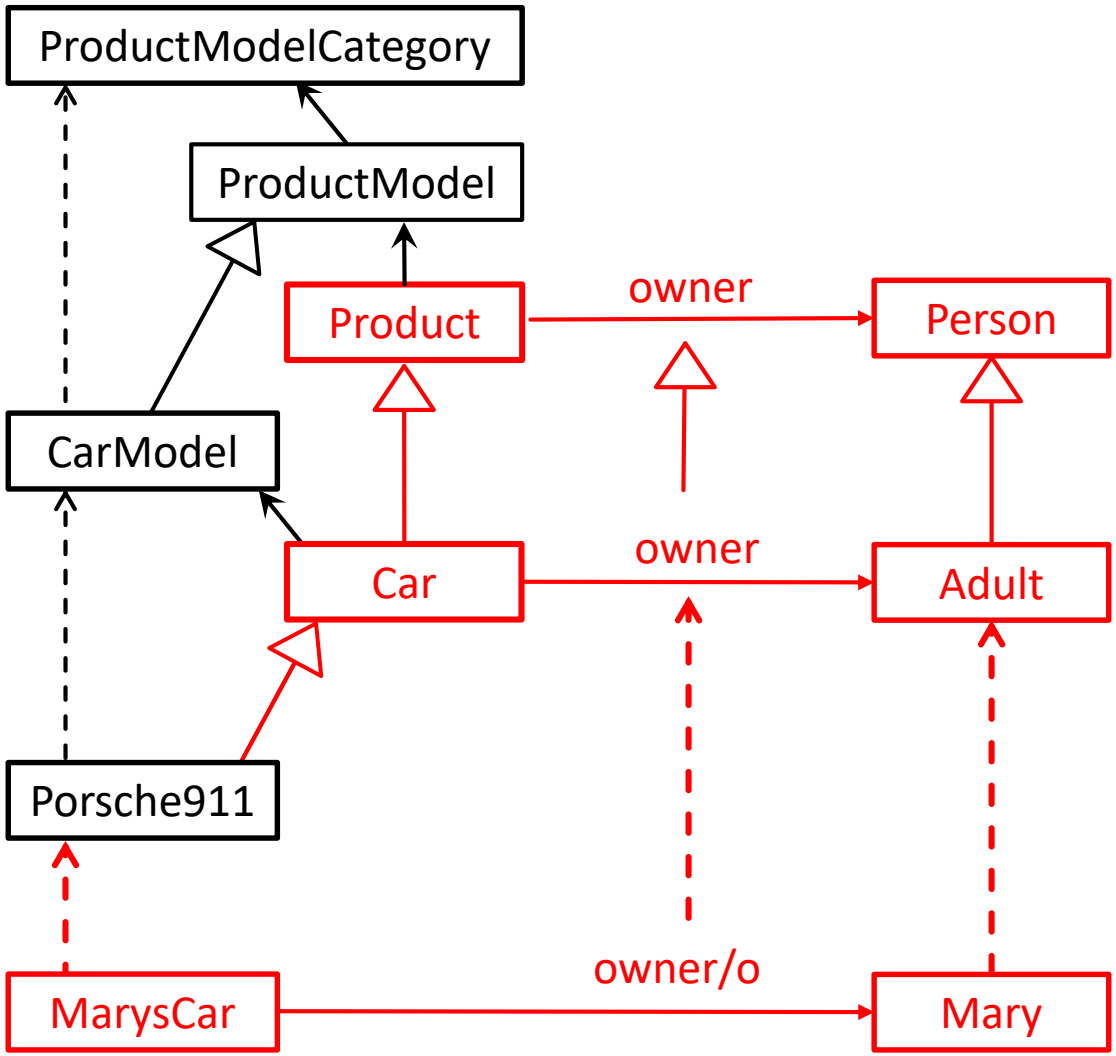
$$\forall x, c, m : IN(m, c) \wedge IN(n, d) \wedge Isa(c, d) \Rightarrow Isa(x, m)$$

Deep Characterization and Specialization

Attribute class ***owner*** is specified with the most general instance of the most general instance of ProductModelCategory.

Attribute class owner is specialized with Car, the most general instance of CarModel.

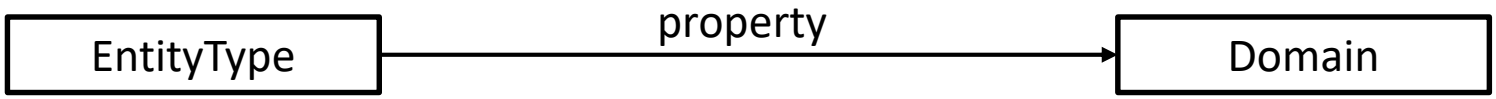
Attribute class owner is instantiated by instance-instance-instances of ProductModelCategory.



Linguistic Metamodeling in DeepTelos

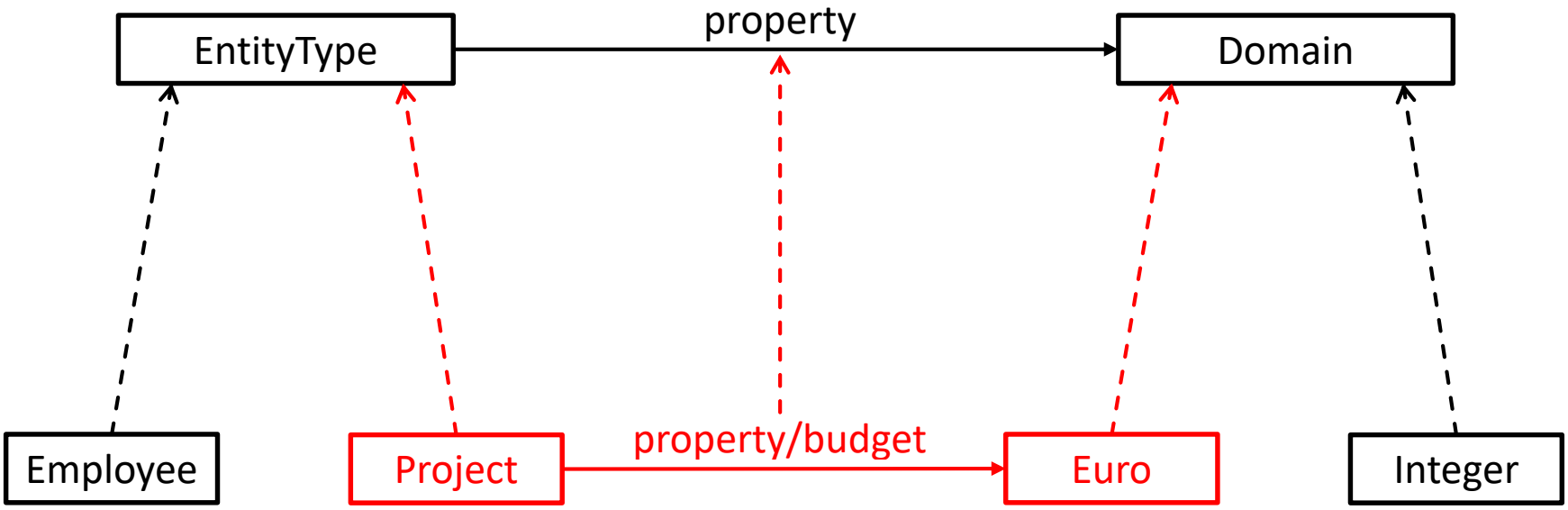
Linguistic Metamodeling

Modeling language constructs are modeled as metaclasses.



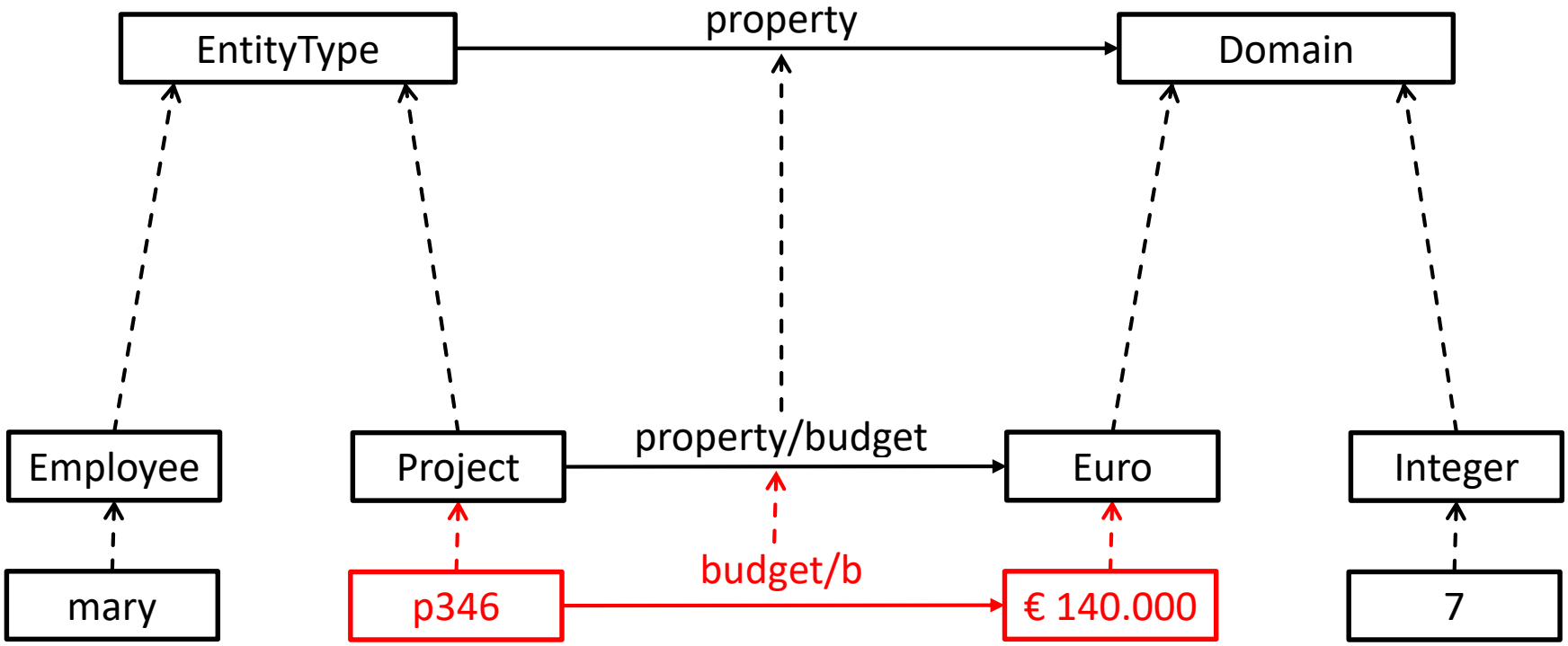
Linguistic Metamodeling

Modeling language constructs are modeled as metaclasses.



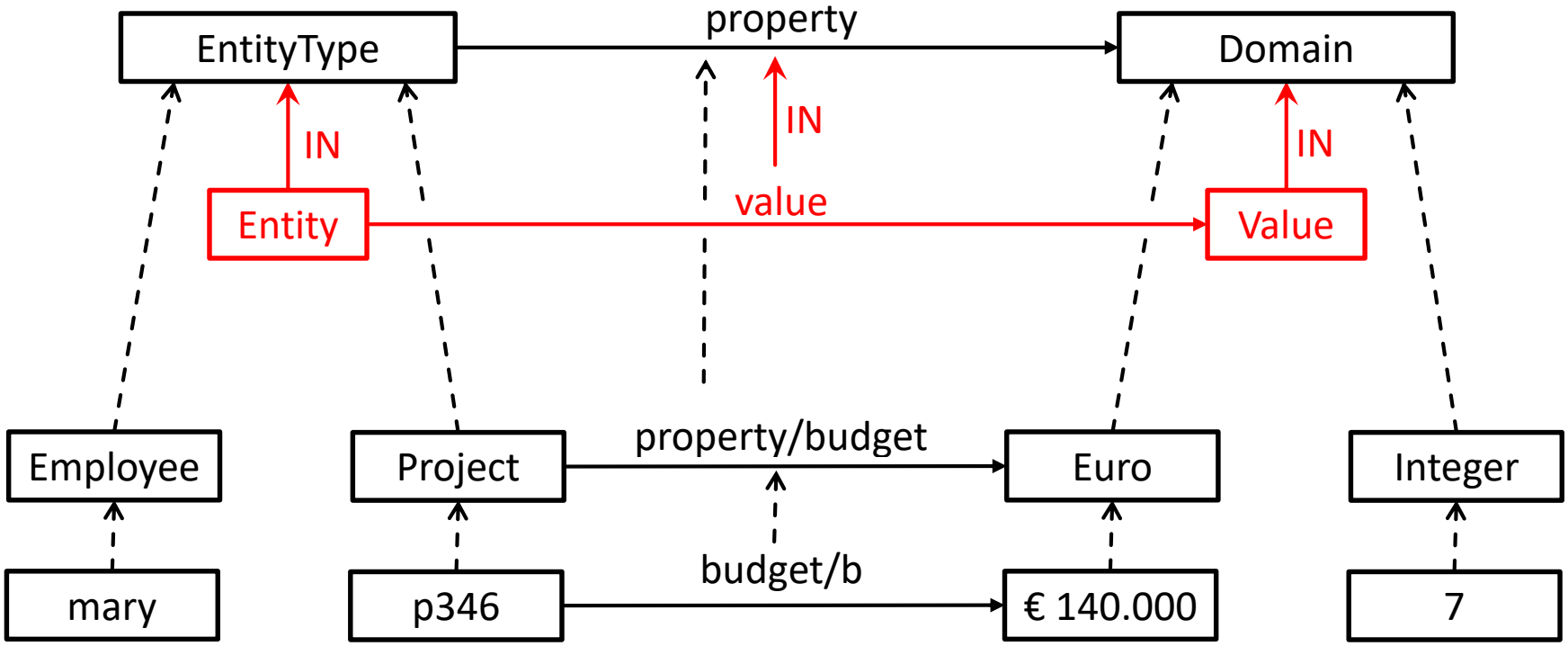
Linguistic Metamodeling

Modeling language constructs are modeled as metaclasses.



Linguistic Metamodeling with Most General Instances

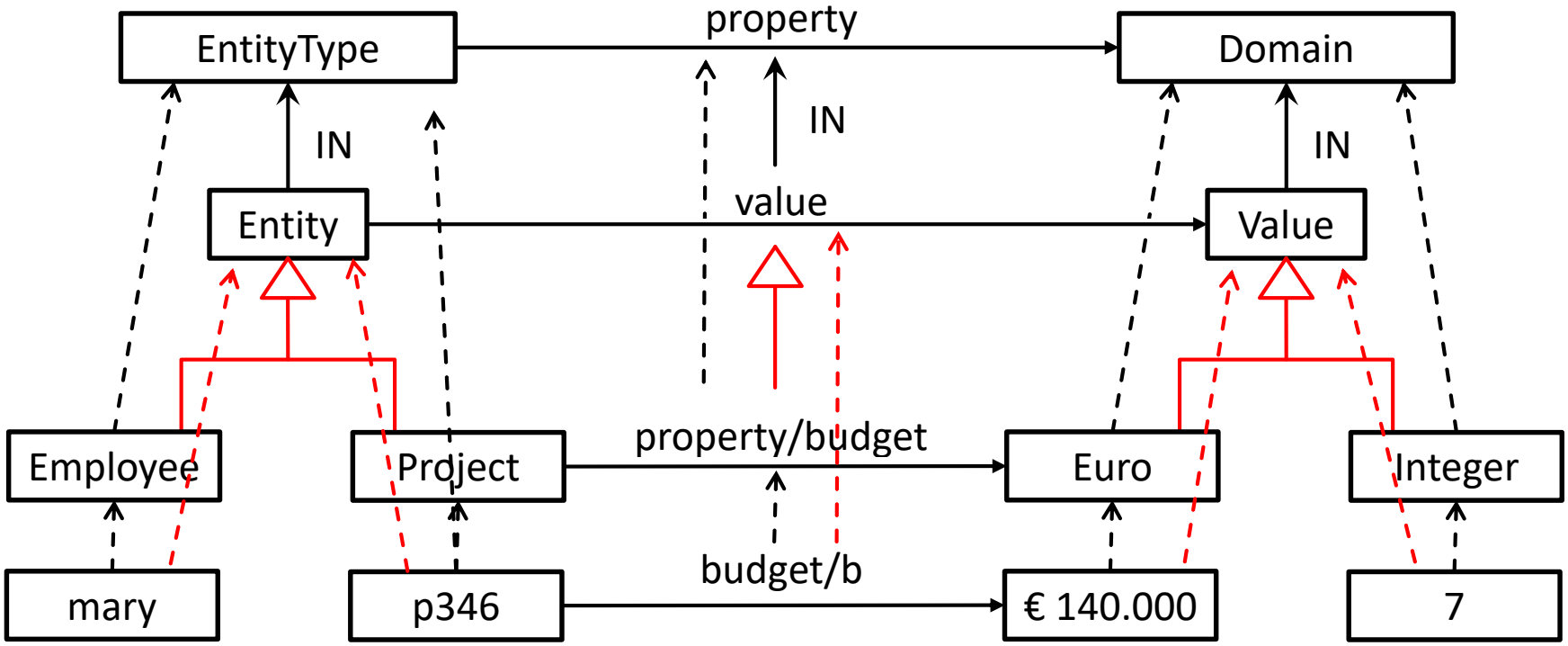
Modeling constructs (i.e., metaclasses) get most-general instances which act as their proxies on the class level.



Linguistic Metamodeling with Most General Instances

Derived specialization and instantiation relationships facilitate schemaless querying.

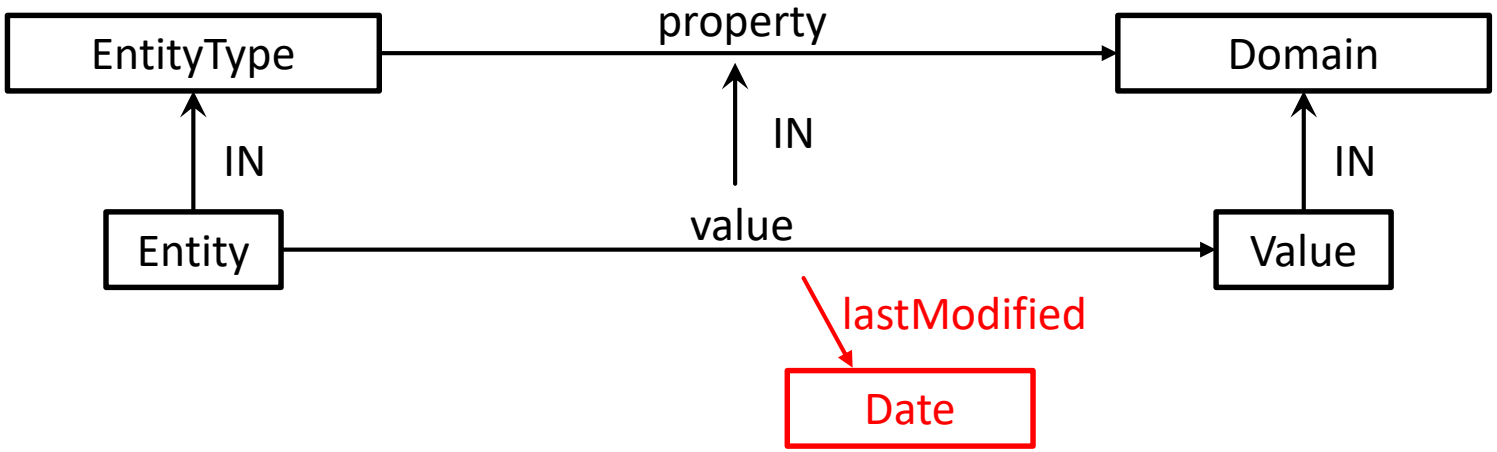
E.g., what are the property values of p346?



Linguistic Metamodeling with Most General Instances

... and generic (schema-independent) extensions.

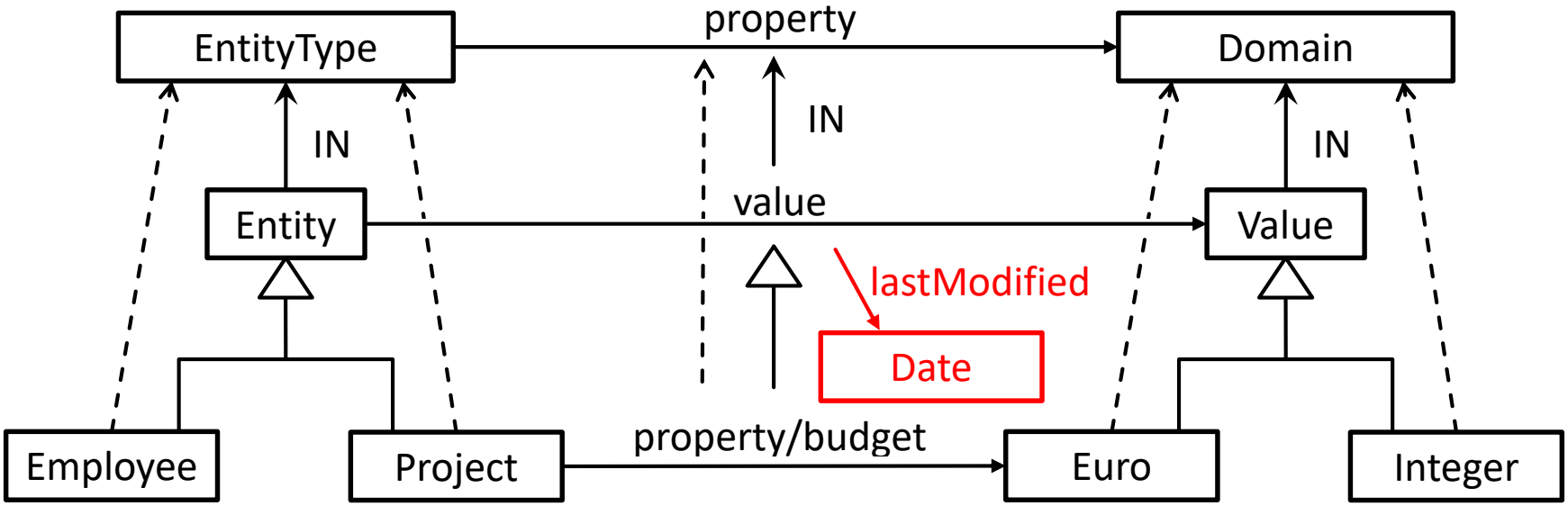
E.g., property values have a lastModified attribute.



Linguistic Metamodeling with Most General Instances

... and generic (schema-independent) extensions.

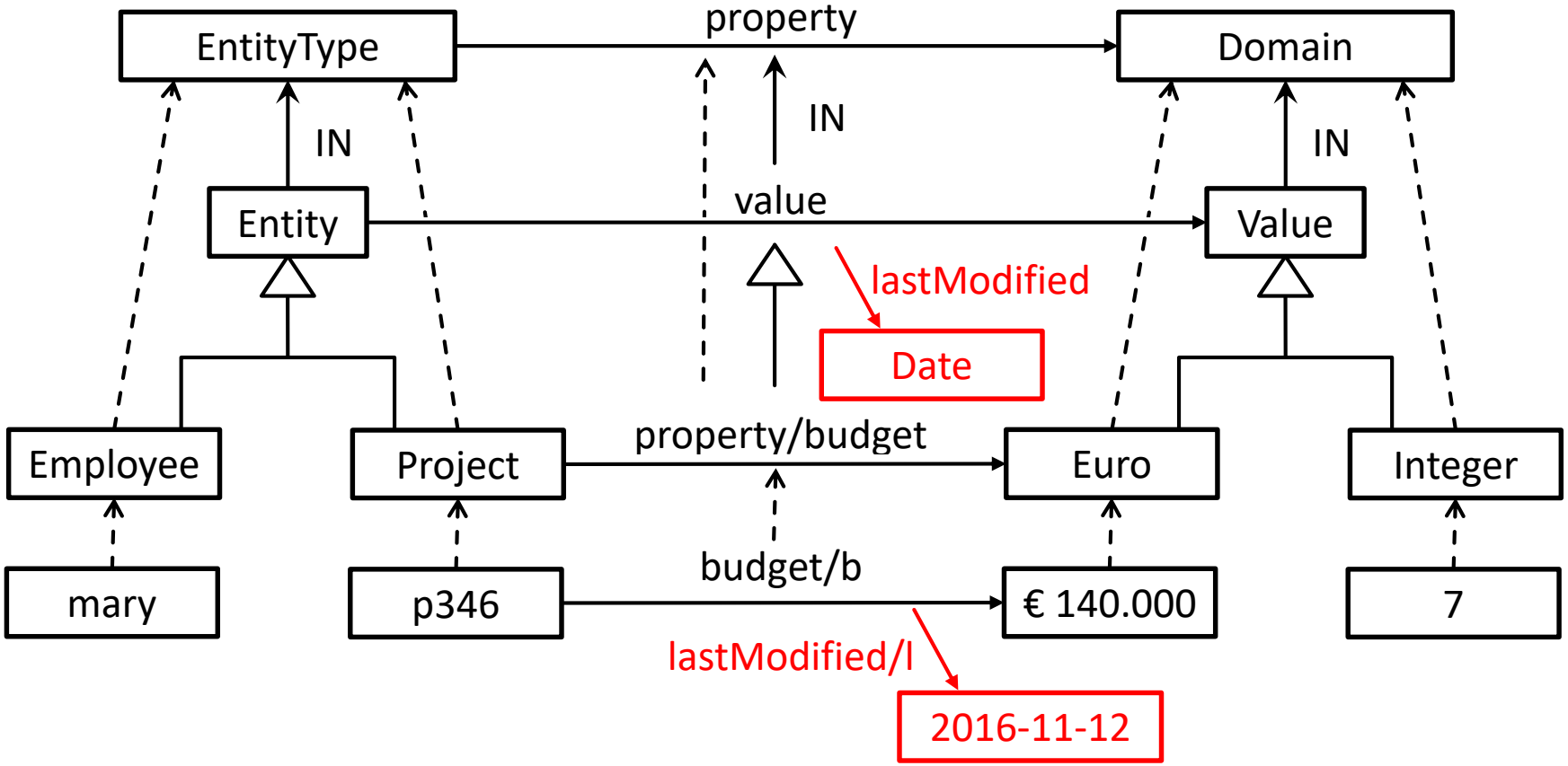
E.g., property values have a lastModified attribute.



Linguistic Metamodeling with Most General Instances

... and generic (schema-independent) extensions.

E.g., property values have a lastModified attribute.



Implementation, Related Work, and Conclusion

Implementation

- DeepTelos is implemented in ConceptBase
- The implementation together with further examples is available under an open license at

http://conceptbase.cc/deeptelos

- We invite everyone to use DeepTelos for own experiments, developments and further extensions

Related Work

- **Powertypes** and **powertype pattern**: Most general instances (MGIs) are inverse of power types. In contrast to adding a bit of metamodeling to two-level modeling (powertype pattern), DeepTelos adds deep characterization to a fully-fledged metamodeling language and system.
- **VODAK** pioneered linguistic metamodeling with deep characterization (with a construct similar to MGIs) but limited to three levels.
- **Deep Instantiation/Modeling (DI)** use potencies for deep characterization. Current approaches make a strong distinction between attributes (with deep characterization based on potencies) and relationships (restricted to strict metamodeling, restricted even more than Telos without MGIs).
- **Dual Deep Instantiation/Modeling (DDI)** overcomes limitations of DI by source and target potencies, to specify the number of instantiation steps separately for the source and target of a relationship. Even more flexible than DeepTelos but with added complexity.
- Ontological foundations: **UFO-MLT**

Conclusion and Future Work

- We introduced DeepTelos
  - a lightweight extension of the metamodeling system Telos/ConceptBase
  - deep characterization of meta^n-classes by (chains of) most general instances
- What sets DeepTelos apart are strengths inherited from Telos and ConceptBase:
  - simplicity and conceptual clarity
  - formal semantics expressed and implemented in Datalog
  - rich query and query optimization facilities

- Future work includes
  - comparing DeepTelos with other multilevel modeling approaches
  - investigating further axioms, possibly based on UFO-MLT

Thank you for your attention!

http://conceptbase.cc/deeptelos