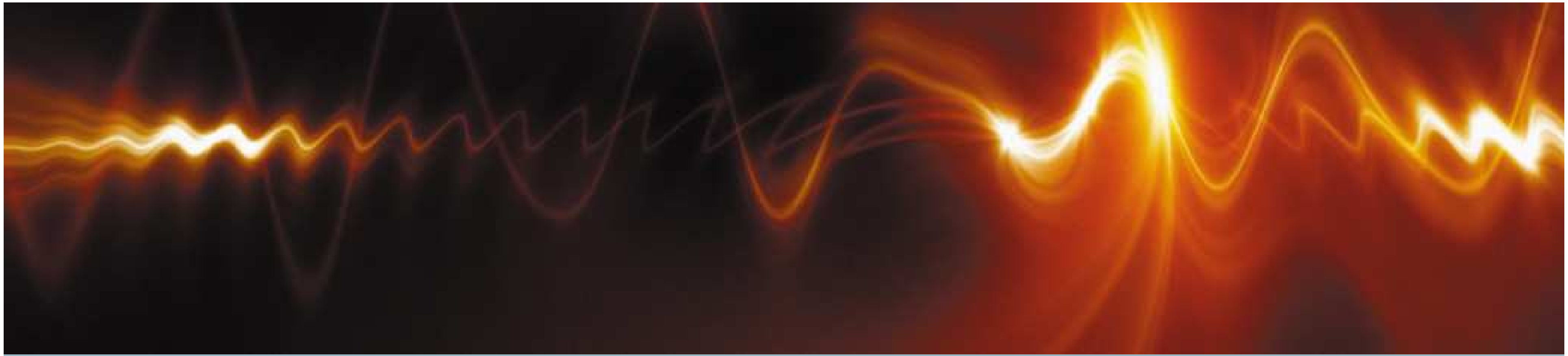


Some thoughts on requirements for languages in engineering

Requirements for Languages for modelling big systems

World Ontology Summit, 2012-03-22

David Leal

CAESAR Systems

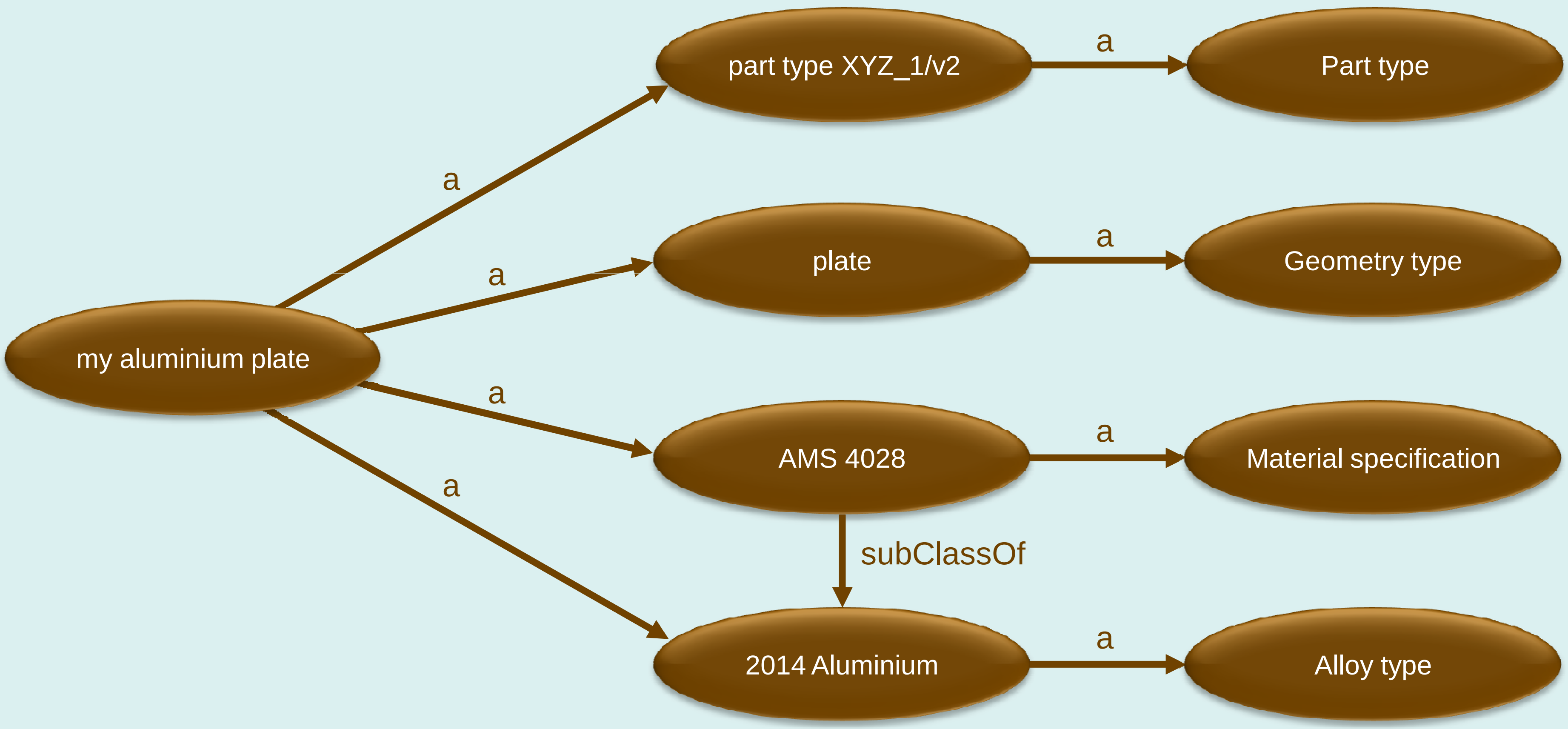
Topics

1. Need for classes, classes of class, etc.
   - Not a problem for practical queries
   - Inferencing has to accommodate this
2. Need to treat class level information and instance level information in analogous ways
   - Existing engineering practice does this for good reasons
   - A design is created without knowing whether one will be built or lots
3. What about variables?
   - Parameterised designs and optimization within design spaces are important

1. Need for classes and classes of class

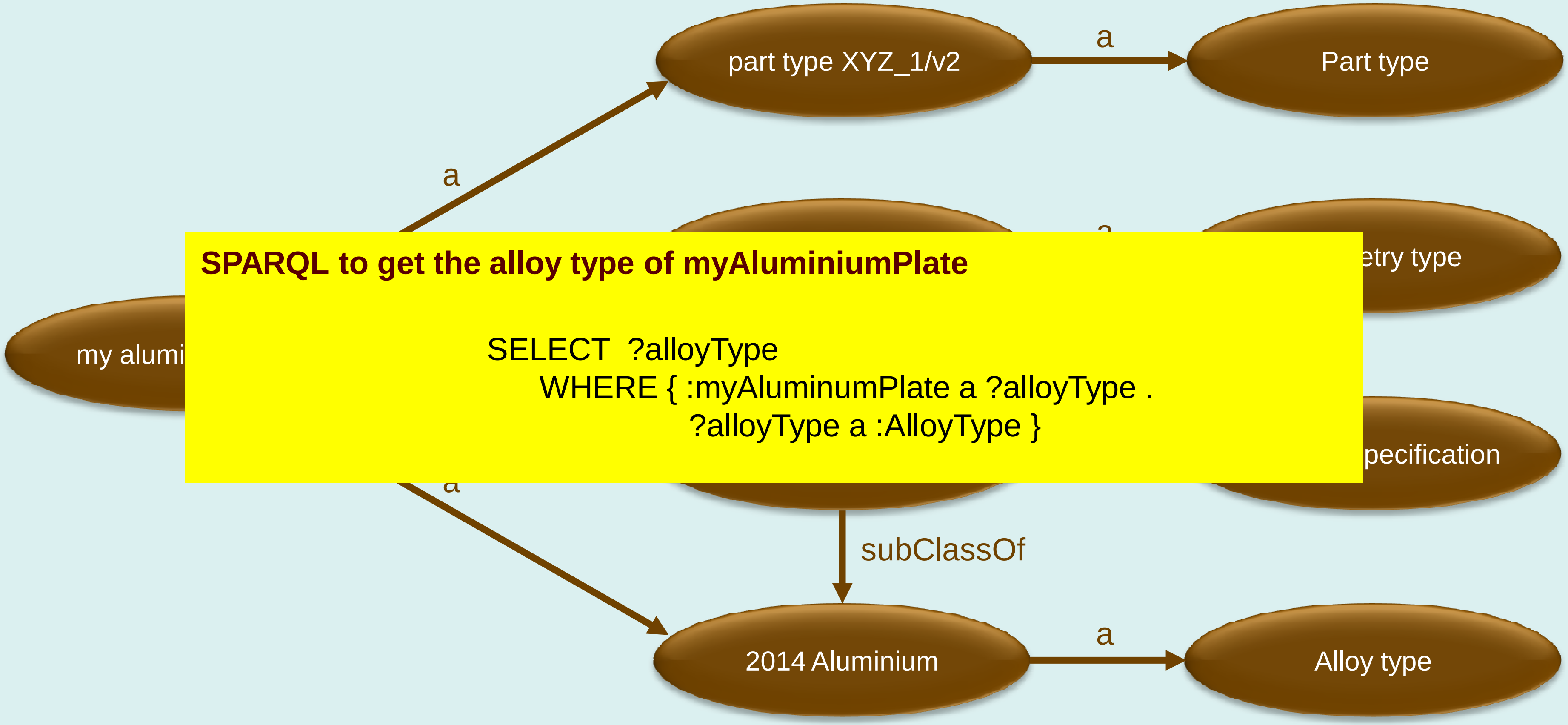
Things are multiply classified

- A query "**what type of thing is this**" will return lots of stuff.
- Therefore it is important to classify the classes, so that you can select what you want.



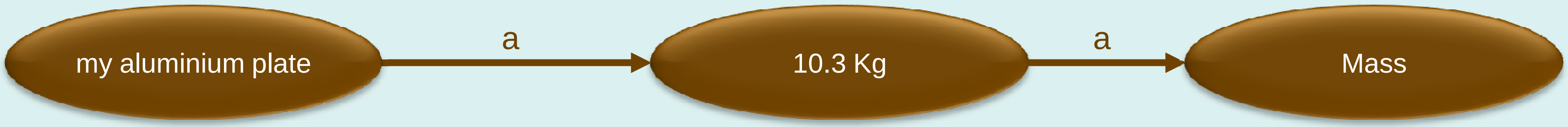
Things are multiply classified

- A query "**what type of thing is this**" will return lots of stuff.
- Therefore it is important to classify the classes, so that you can select what you want.



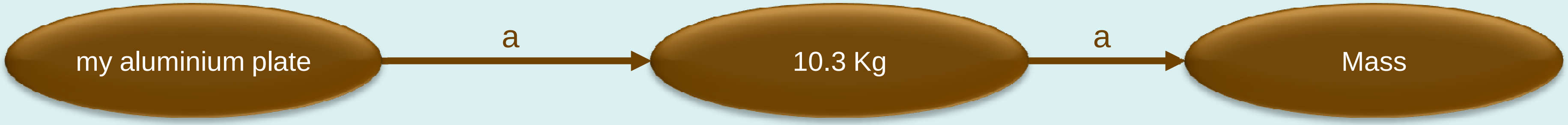
Things are multiply classified

- Sometimes the relationship with a quantity is just classification too.



Things are multiply classified

- Sometimes the relationship with a quantity is just classification too.

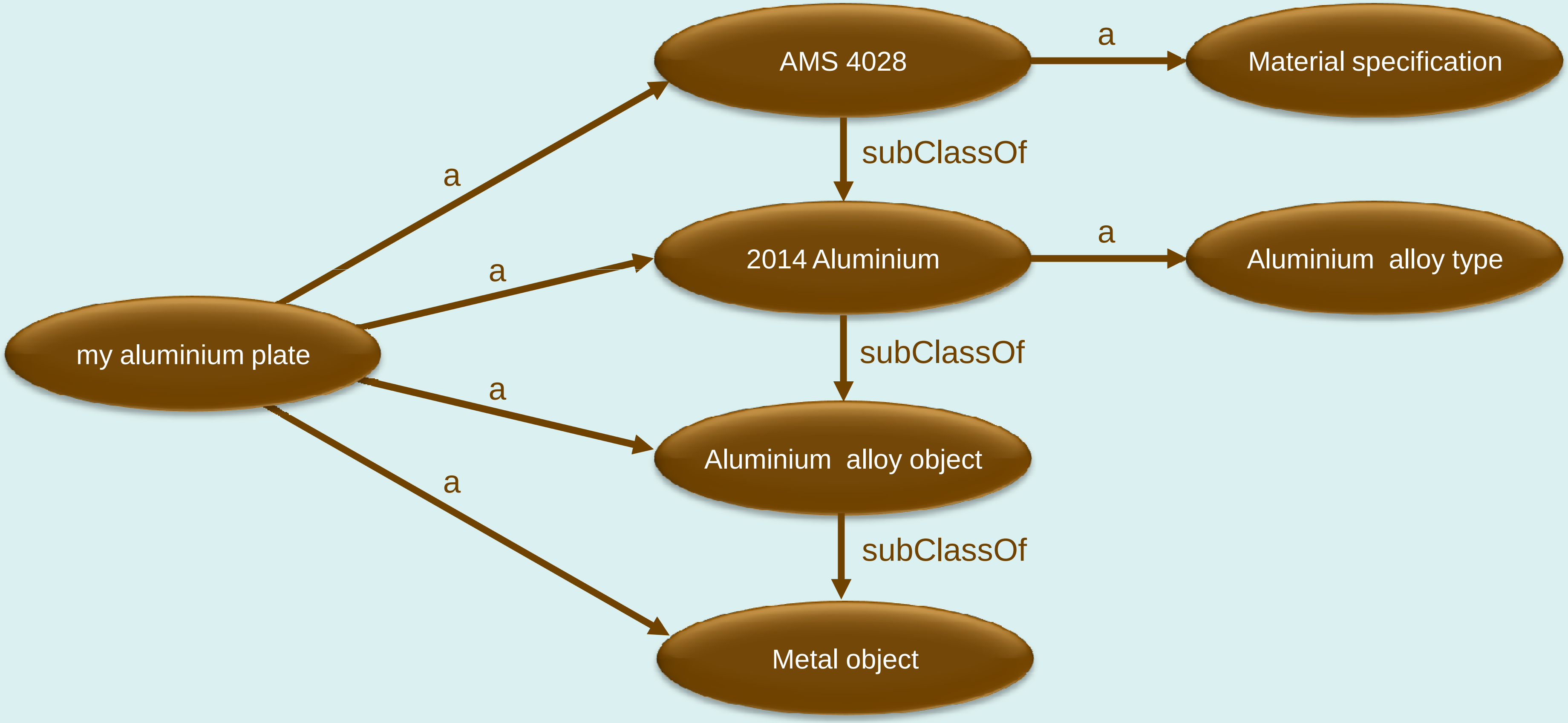


SPARQL to get the mass of myAluminiumPlate

```
SELECT  ?mass
     WHERE { :myAluminumPlate a ?mass .
                    ?mass a :Mass }
```

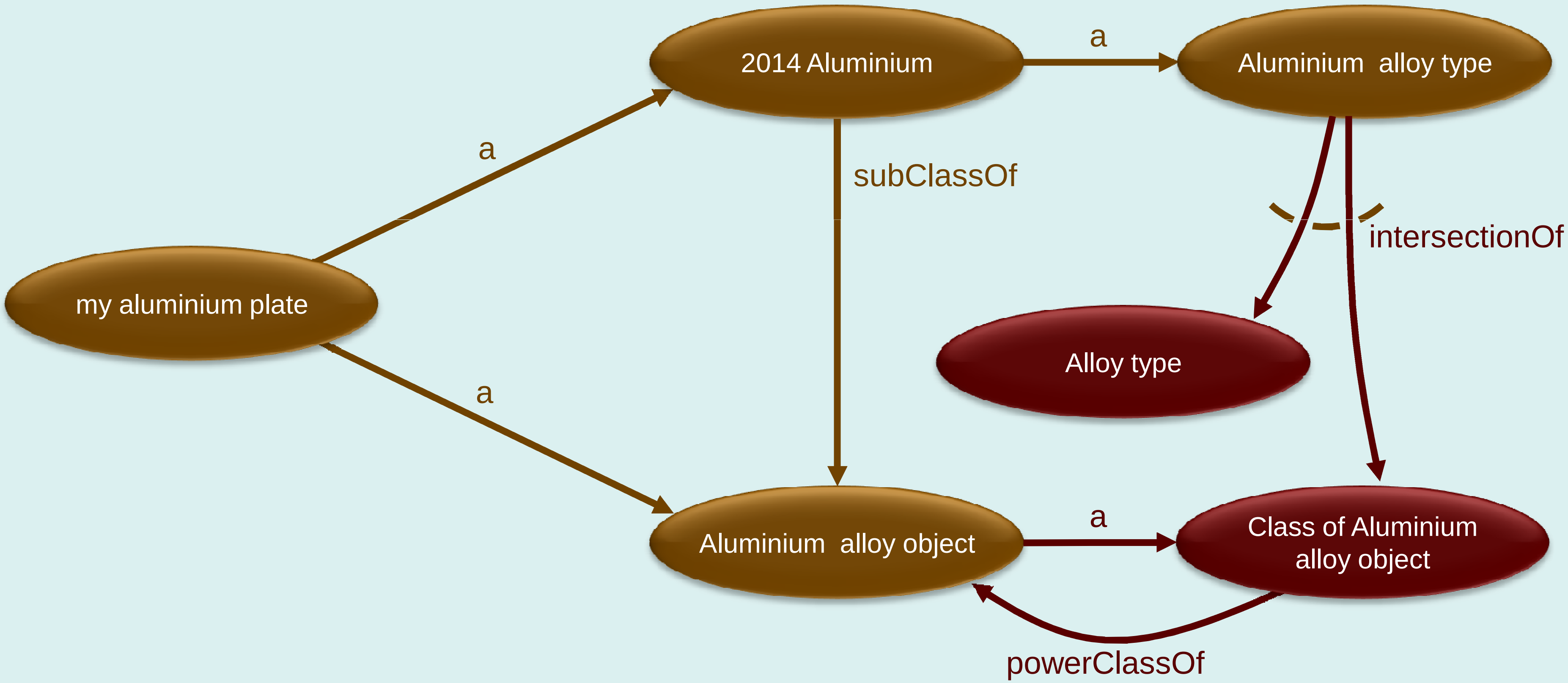
Things are multiply classified

- There are so many classes as different meta-levels, that it is difficult to keep track.



Things are multiply classified

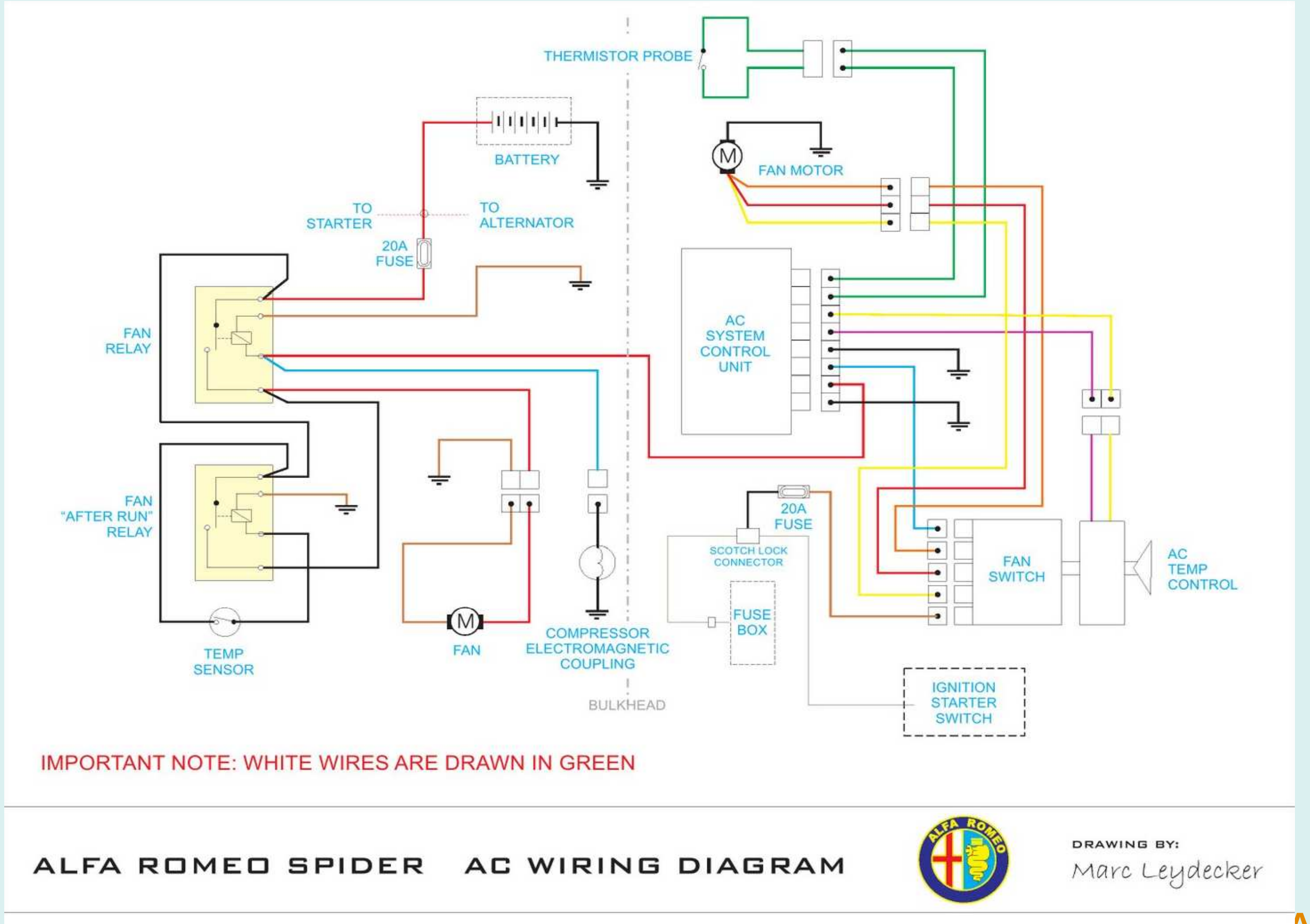
- There are so many classes as different meta-levels, that it is difficult to keep track.
- The use of power classes helps to organise the data.



2. Need to treat class level and individual level information in analogous ways

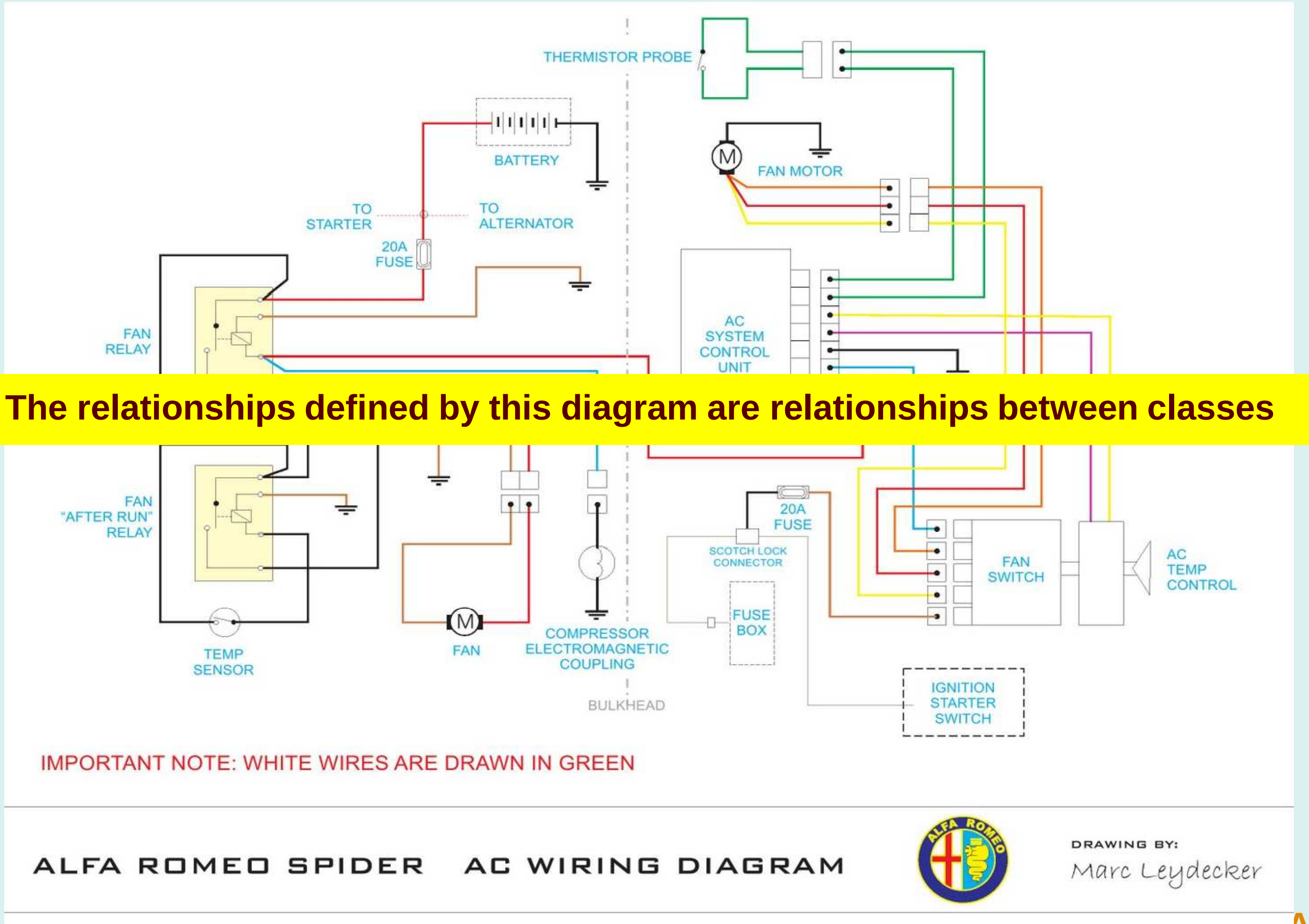
A car wiring diagram

- Each symbol on the diagram represents a class of component.
- But when working on myCar, I assume that each symbol represents an individual component of myCar – the ambiguity is useful.



A car wiring diagram

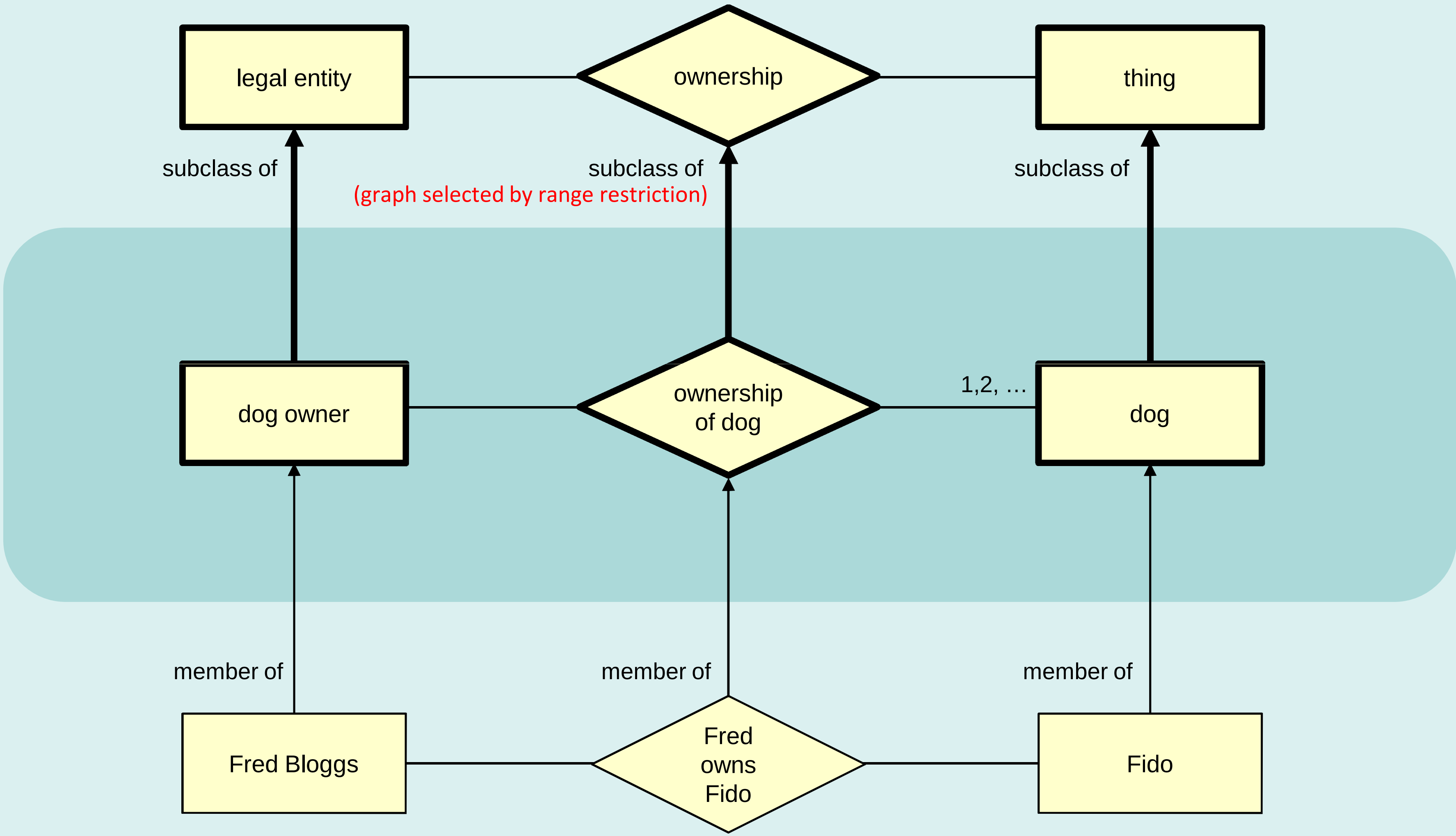
- Each symbol on the diagram represents a class of component.
- But when working on myCar, I assume that each symbol represents an individual component of myCar – the ambiguity is useful.



The relationships defined by this diagram are relationships between classes

Digression on notation - dog owner example

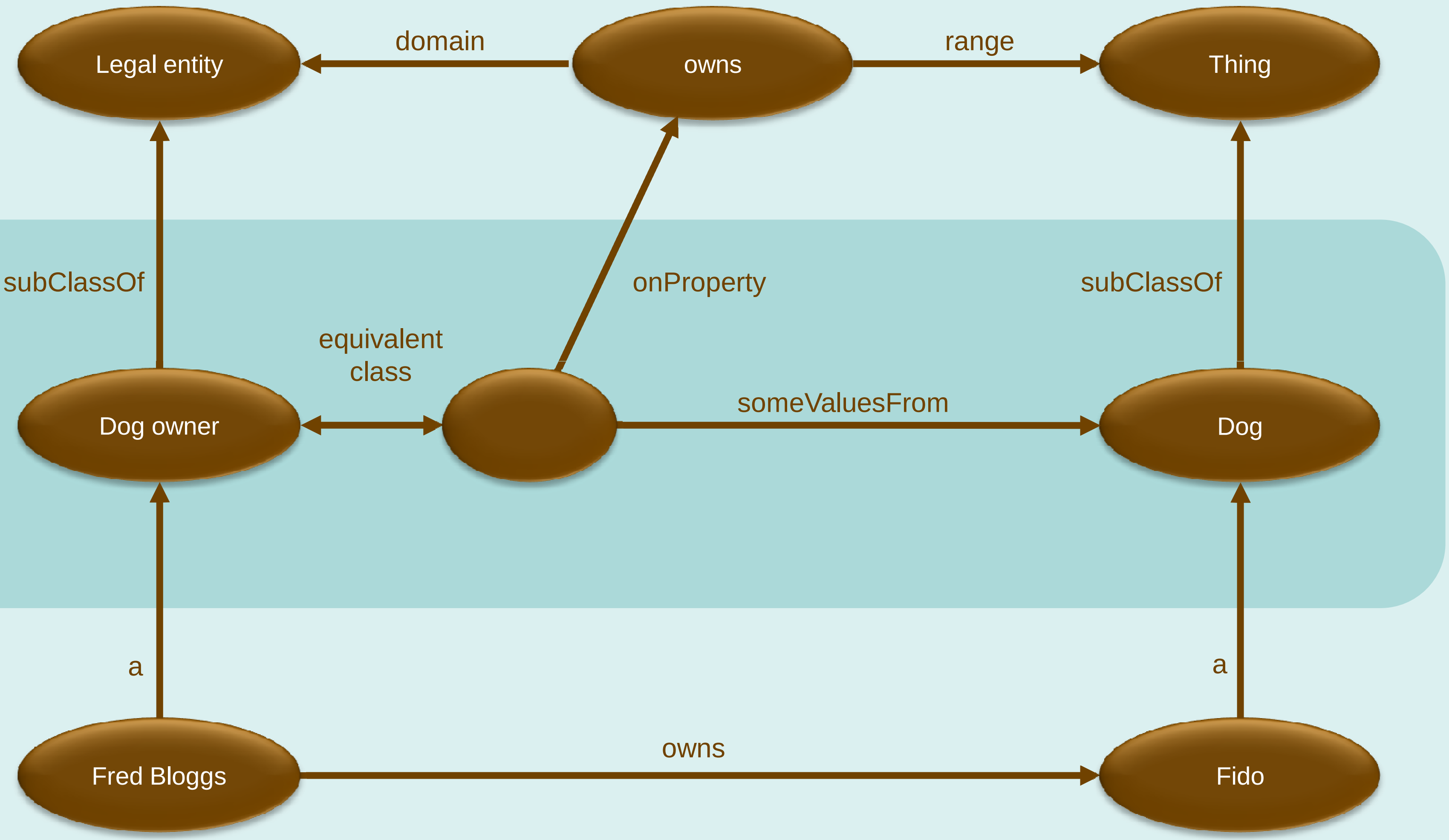
- Define a specialised class and a specialised (class of) relationship



ISO 15926-2 notation

Digression on notation - dog owner example

- Define a specialised class (but not a specialised relationship)



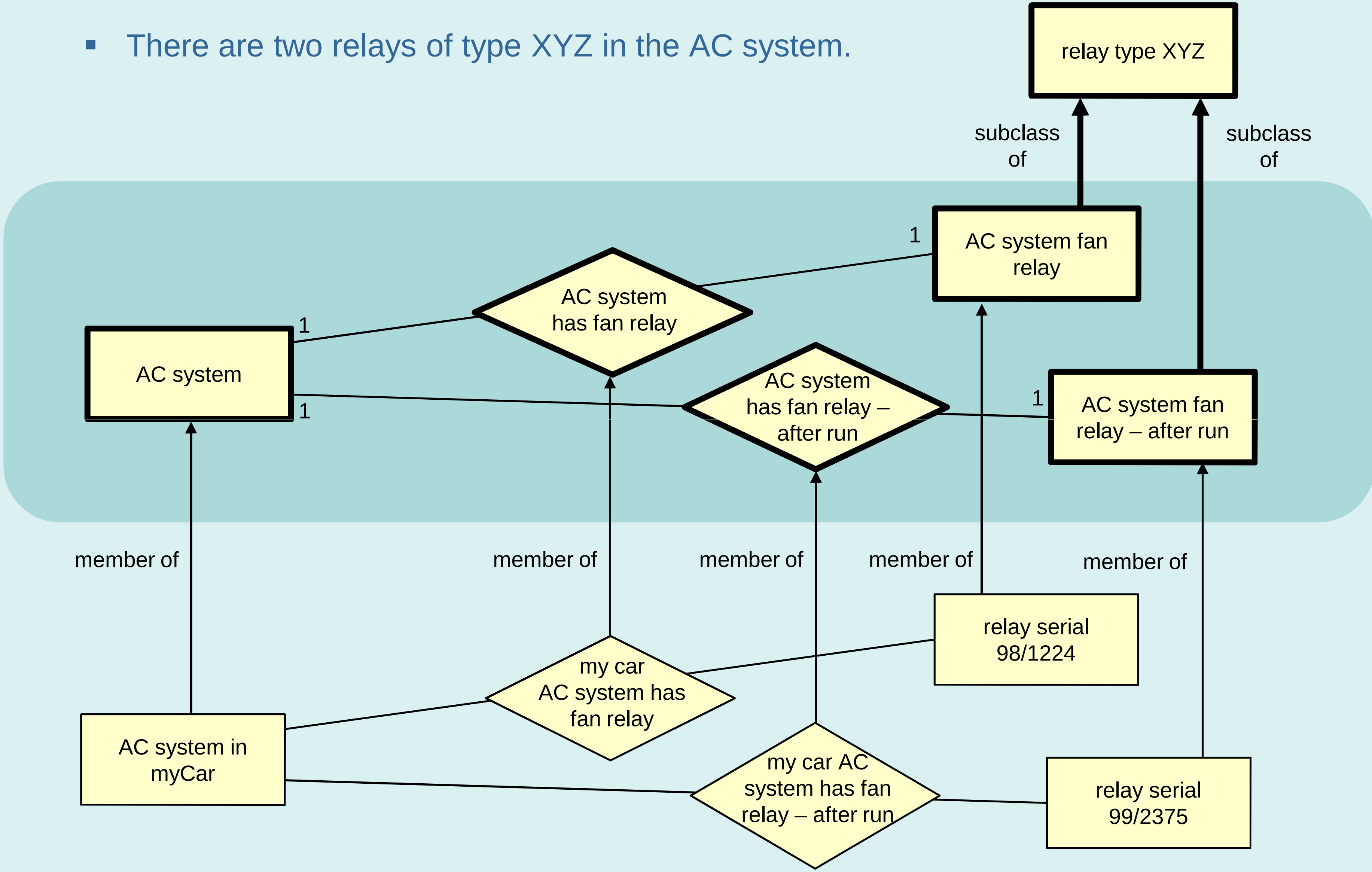
RDF/OWL notation

Digression on notation - dog owner example

- ISO 15926 and OWL are equivalent
  - Actually a small upgrade to ISO 15926 is required to specify how the specialised (class of) relationship is created
- ISO 15926 defines a specialised (class of) relationship, but OWL does not.
- The specialised relationships are useful, because they give an analogous relationships at the class and instance levels.

A car wiring diagram

- There are two relays of type XYZ in the AC system.



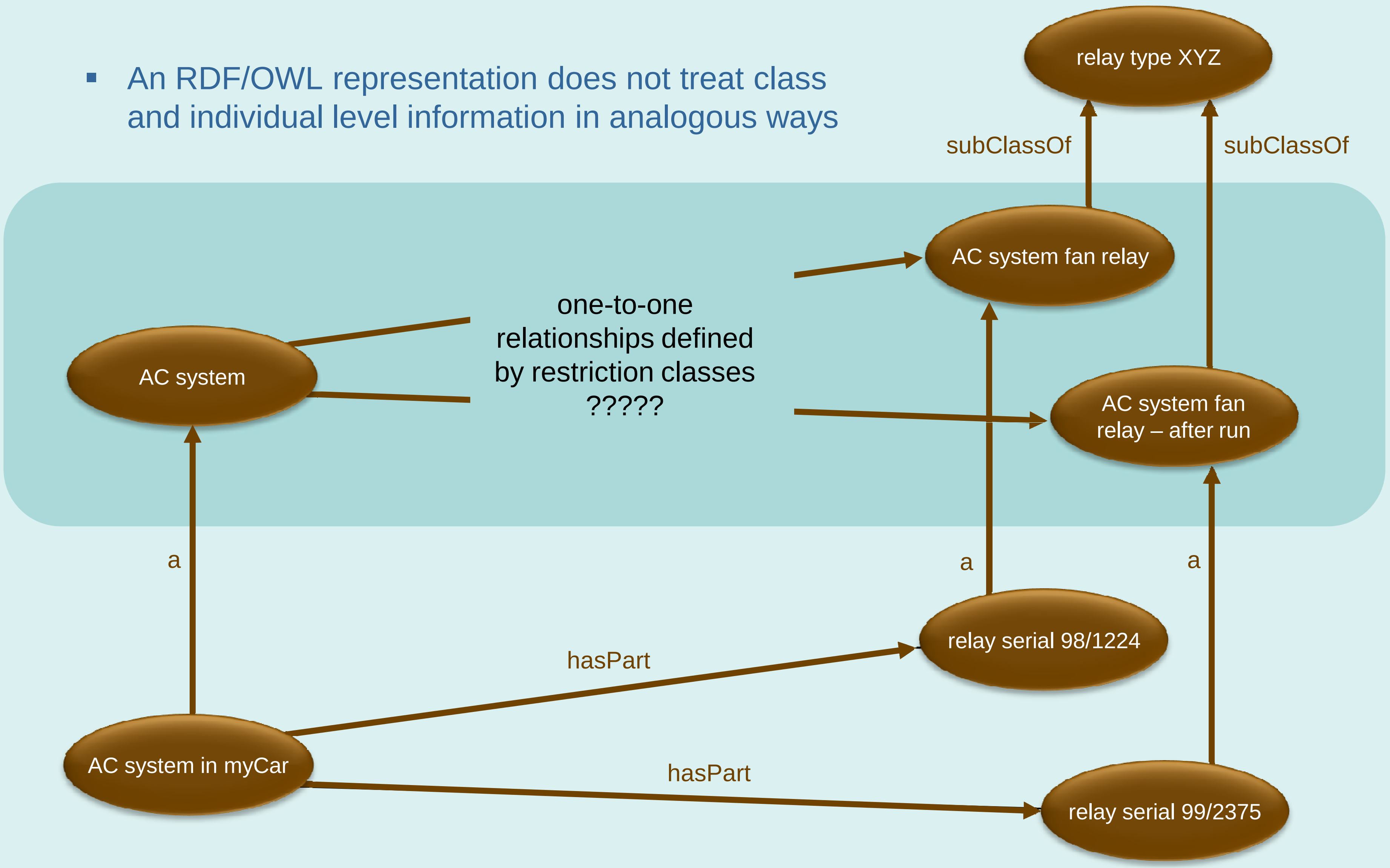
ISO 15926-2 notation

A car wiring diagram - terminology

- *AC system fan relay* and *AC system fan relay – after run* are two “**design occurrences**” of *relay type XYZ*.
- *AC system fan relay* is the “**role**” of *relay serial 98/1224* in the AC system of *myCar*.
- The relationships “**design occurrence**” and “**role**” are very important to engineering, but there is no established terminology and the relationships are not usually defined in ontologies.

A car wiring diagram - notation

- An RDF/OWL representation does not treat class and individual level information in analogous ways



RDF/OWL notation

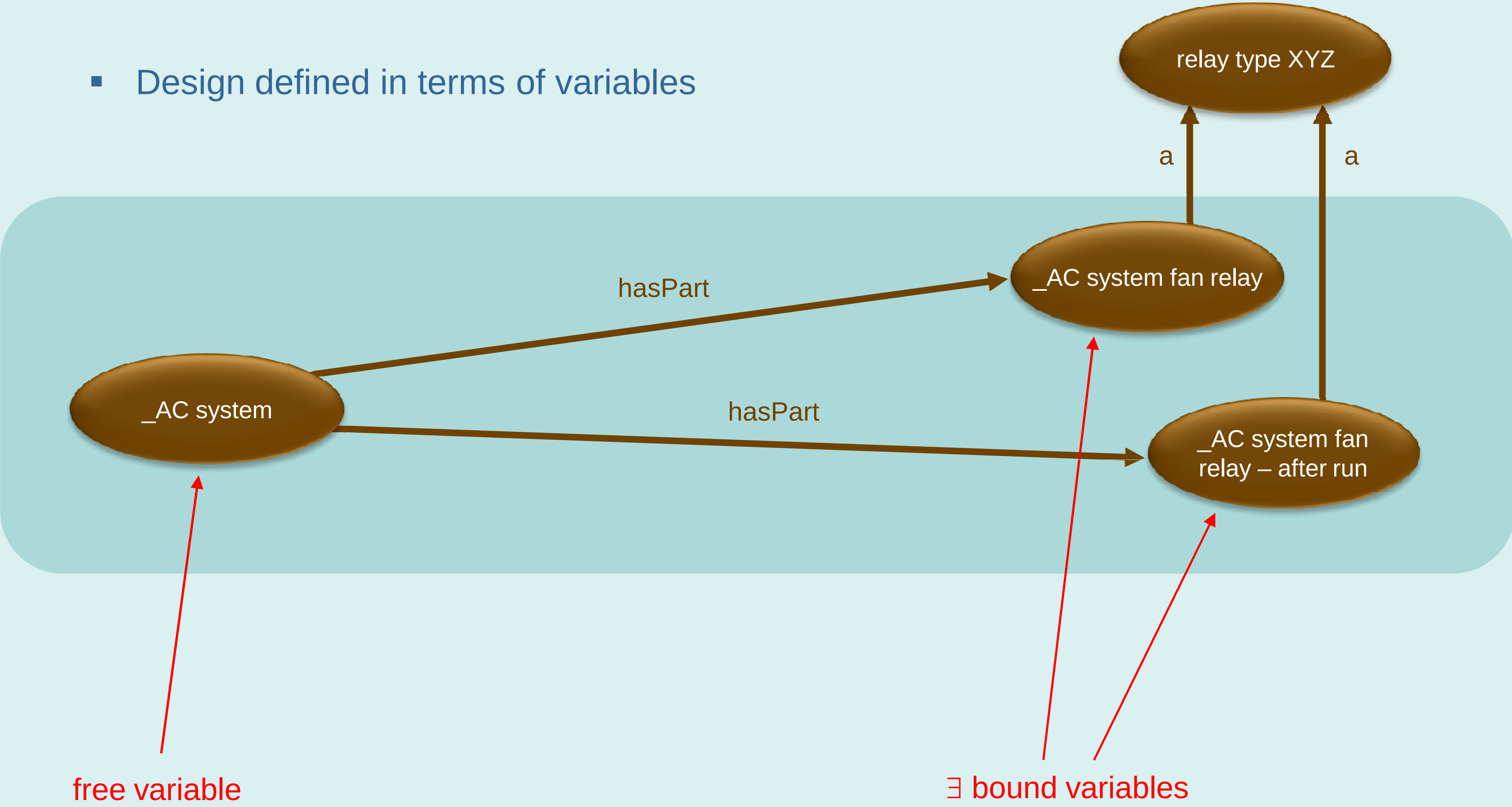
3. What about variables?

Why variables?

- Designers define design spaces
  - An optimal (or perhaps manufacturable) design is then found within the space
- A design space is a class that contains individual designs as members.
  - A design space is defined by “ranges” of variables. (A “range” can be a finite set of choices.)
- A specific design within a design space can also be expressed in terms of variables, where an instance of the design is a binding of the variables to individuals.
  - OK – it sound odd – but bear with me, and look again at the car wiring diagram.

A car wiring diagram

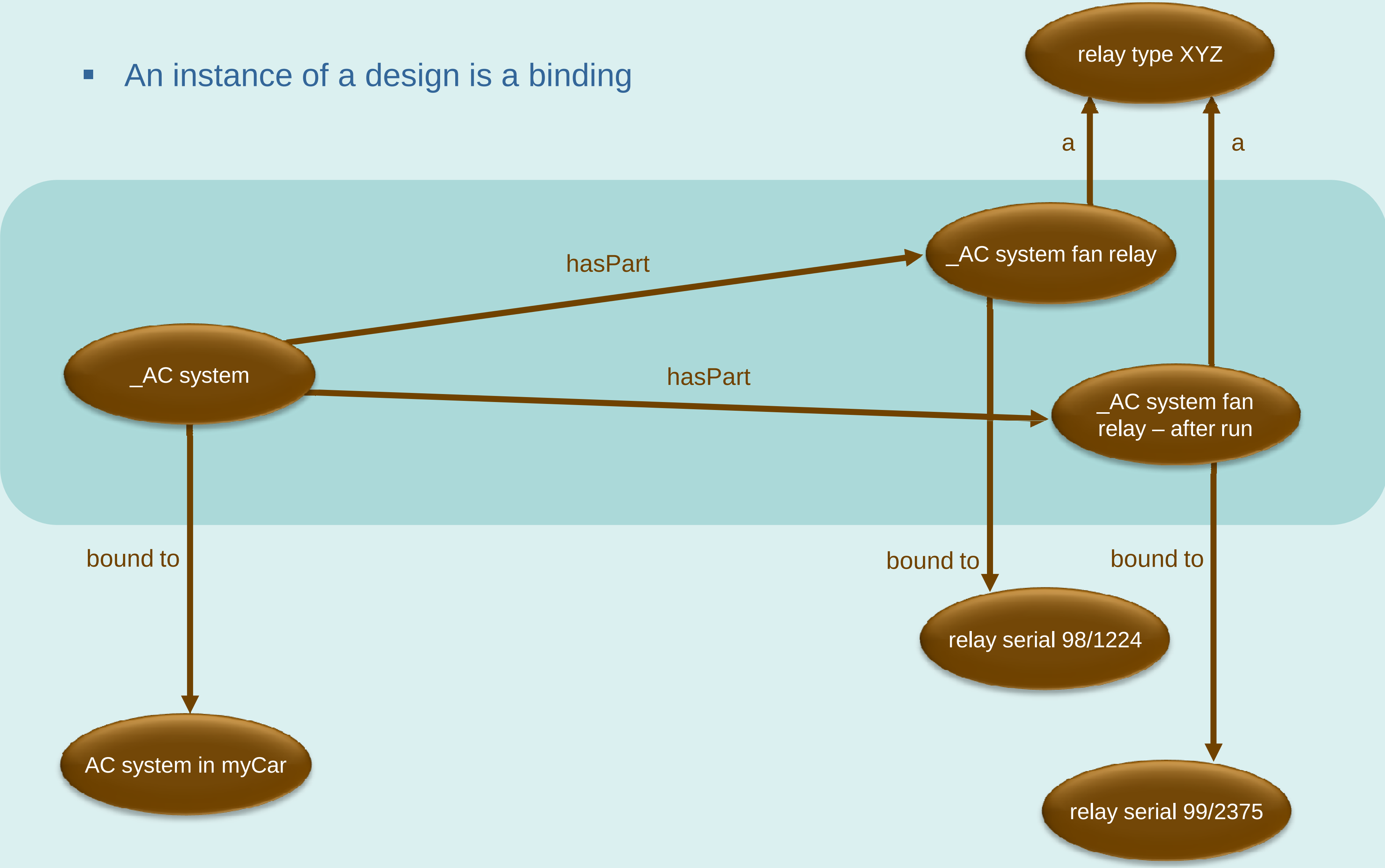
- Design defined in terms of variables



RDF/OWL notation

A car wiring diagram

- An instance of a design is a binding



RDF/OWL notation

Where do we go with variables?

- A mathematical definition of a design uses variables
  - I believe that any expression involving variables can ***with sufficient effort*** be expressed in terms of mappings between classes
  - We use variables, because they make life easier.
- Heretofore, attempts to record a design as a formal set of statements have not made use of variables
  - These attempts have not been successful, because the complexity of the information in a design makes it difficult.
  - Recording a design space is even more difficult.
- Some research is needed in this area.

End