Choosing Objects and Relations

How to represent: “Pen #7 is red.”

Choosing Objects and Relations

How to represent: "Pen #7 is red."

$red(pen_7)$. It's easy to ask "What's red?"

Can't ask "what is the color of pen_7?"

Choosing Objects and Relations

How to represent: "Pen #7 is red."

$red(pen_7)$. It's easy to ask "What's red?"

Can't ask "what is the color of pen_7?"

$color(pen_7, red)$. It's easy to ask "What's red?"

It's easy to ask "What is the color of pen_7?"

Can't ask "What property of pen_7 has value *red*?"

Choosing Objects and Relations

How to represent: "Pen #7 is red."

$red(pen_7)$. It's easy to ask "What's red?"

Can't ask "what is the color of pen_7?"

$color(pen_7, red)$. It's easy to ask "What's red?"

It's easy to ask "What is the color of pen_7?"

Can't ask "What property of pen_7 has value *red*?"

$prop(pen_7, color, red)$. It's easy to ask all these questions.

Choosing Objects and Relations

How to represent: "Pen #7 is red."

$red(pen_7)$. It's easy to ask "What's red?"

Can't ask "what is the color of pen_7?"

$color(pen_7, red)$. It's easy to ask "What's red?"

It's easy to ask "What is the color of pen_7?"

Can't ask "What property of pen_7 has value *red*?"

$prop(pen_7, color, red)$. It's easy to ask all these questions.

prop(*Object*, *Attribute*, *Value*) is the only relation needed:

object-attribute-value representation

Universality of *prop*

To represent “a is a parcel”

- *prop*(*a*, *is_a*, *parcel*), where *is_a* is a special attribute
- *prop*(*a*, *parcel*, *true*), where *parcel* is a Boolean attribute

Reification

- To represent *scheduled*(*cs*422, 2, 1030, *cc*208). “section 2 of course *cs*422 is scheduled at 10:30 in room *cc*208.”
- Let *b*123 name the booking:

  prop(*b*123, *course*, *cs*422).

  prop(*b*123, *section*, 2).

  prop(*b*123, *time*, 1030).

  prop(*b*123, *room*, *cc*208).
- We have reified the booking.
- Reify means: to make into an object.

Semantics Networks

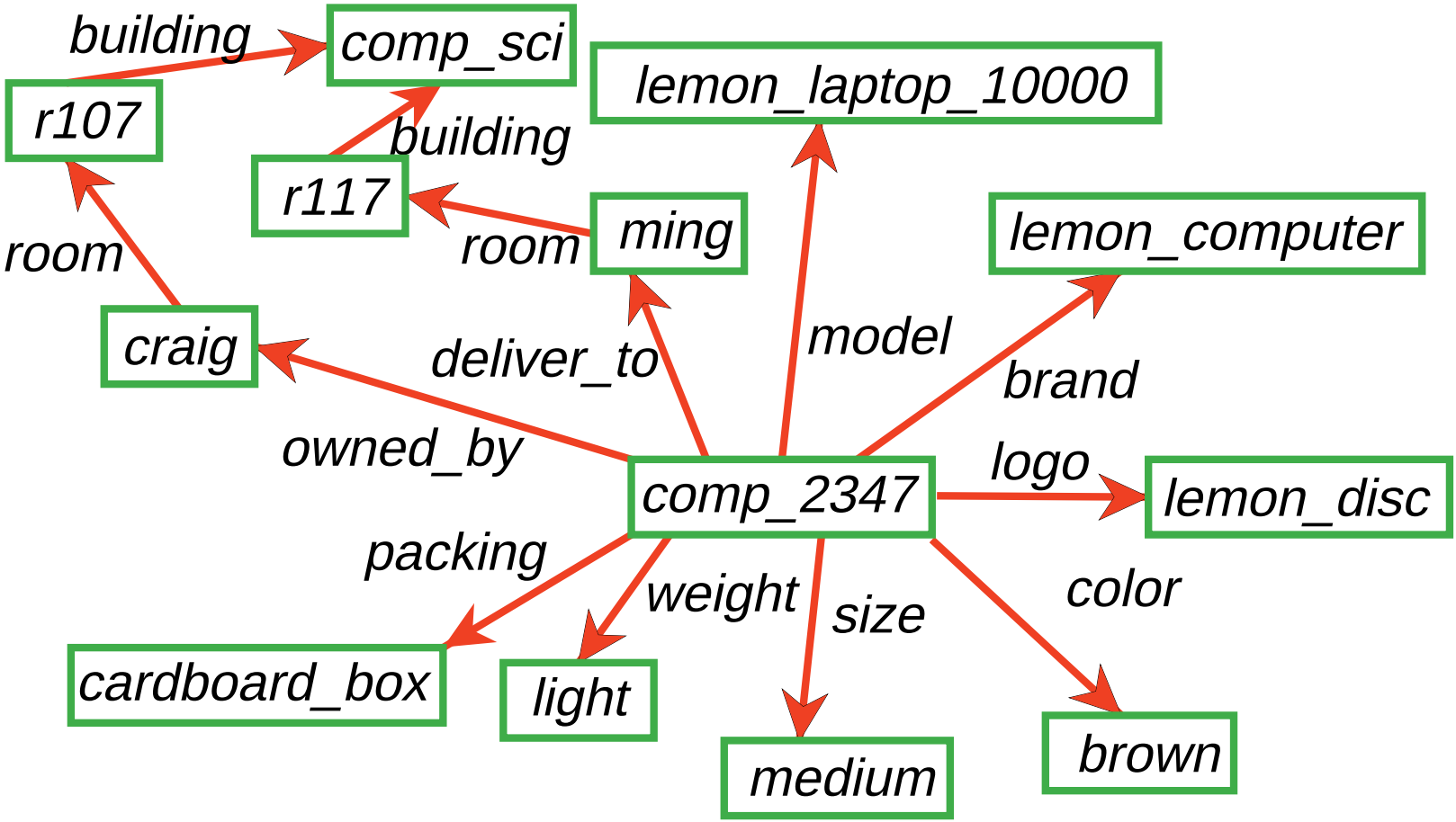
When you only have one relation, *prop*, it can be omitted without loss of information.

Write

$$prop(Obj, Att, Value)$$

as

An Example Semantic Network



Equivalent Logic Program

prop(*comp_*2347, *owned_by*, *craig*).

prop(*comp_*2347, *deliver_to*, *ming*).

prop(*comp_*2347, *model*, *lemon_laptop_*10000).

prop(*comp_*2347, *brand*, *lemon_computer*).

prop(*comp_*2347, *logo*, *lemon_disc*).

prop(*comp_*2347, *color*, *brown*).

prop(*craig*, *room*, *r*107).

prop(*r*107, *building*, *comp_sci*).

⋮

Frames

The properties and values for a single object can be grouped together into a frame.

We can write this as a list of *attribute* = *value* or *slot* = *filler*.

$$
\begin{aligned}
&[owned\_by = craig,\\
&deliver\_to = ming,\\
&model = lemon\_laptop\_10000,\\
&brand = lemon\_computer,\\
&logo = lemon\_disc,\\
&color = brown,\\
&\ldots]
\end{aligned}
$$

Primitive versus Derived Relations

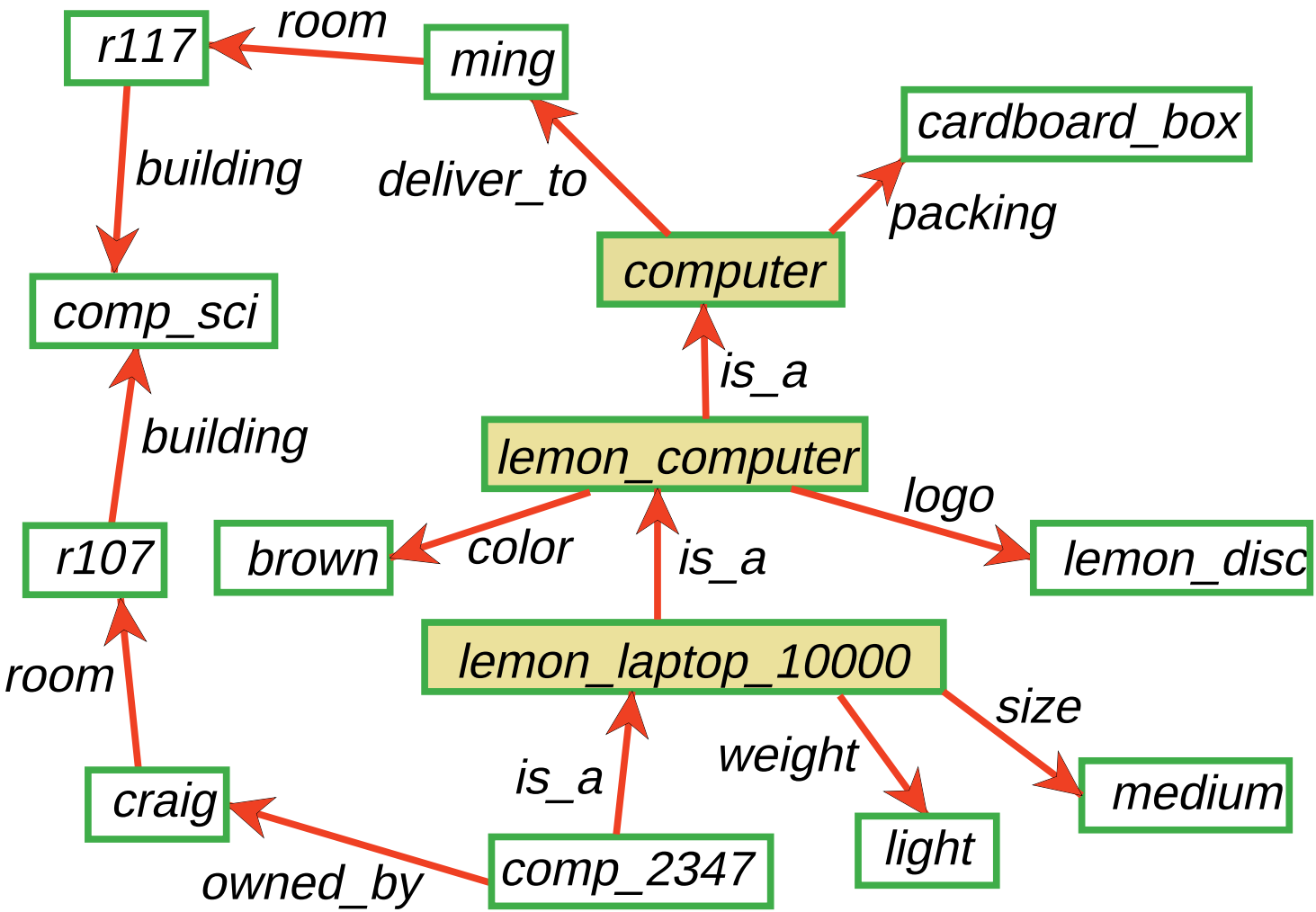
Primitive knowledge is that which is defined explicitly by facts.

Derived knowledge is knowledge defined by rules.

Example: All lemon laptops may have $size = medium$. Associate this property with the class, not the individual.

Allow a special attribute *is_a* between an individual and a class or between two classes that allows for property inheritance.

A Structured Semantic Network



Logic of Property Inheritance

An arc $\xrightarrow{p} n$ from a class c means every individual in the class has value n of attribute p:

$$prop(Obj, p, n) \leftarrow$$
$$prop(Obj, is\_a, c).$$

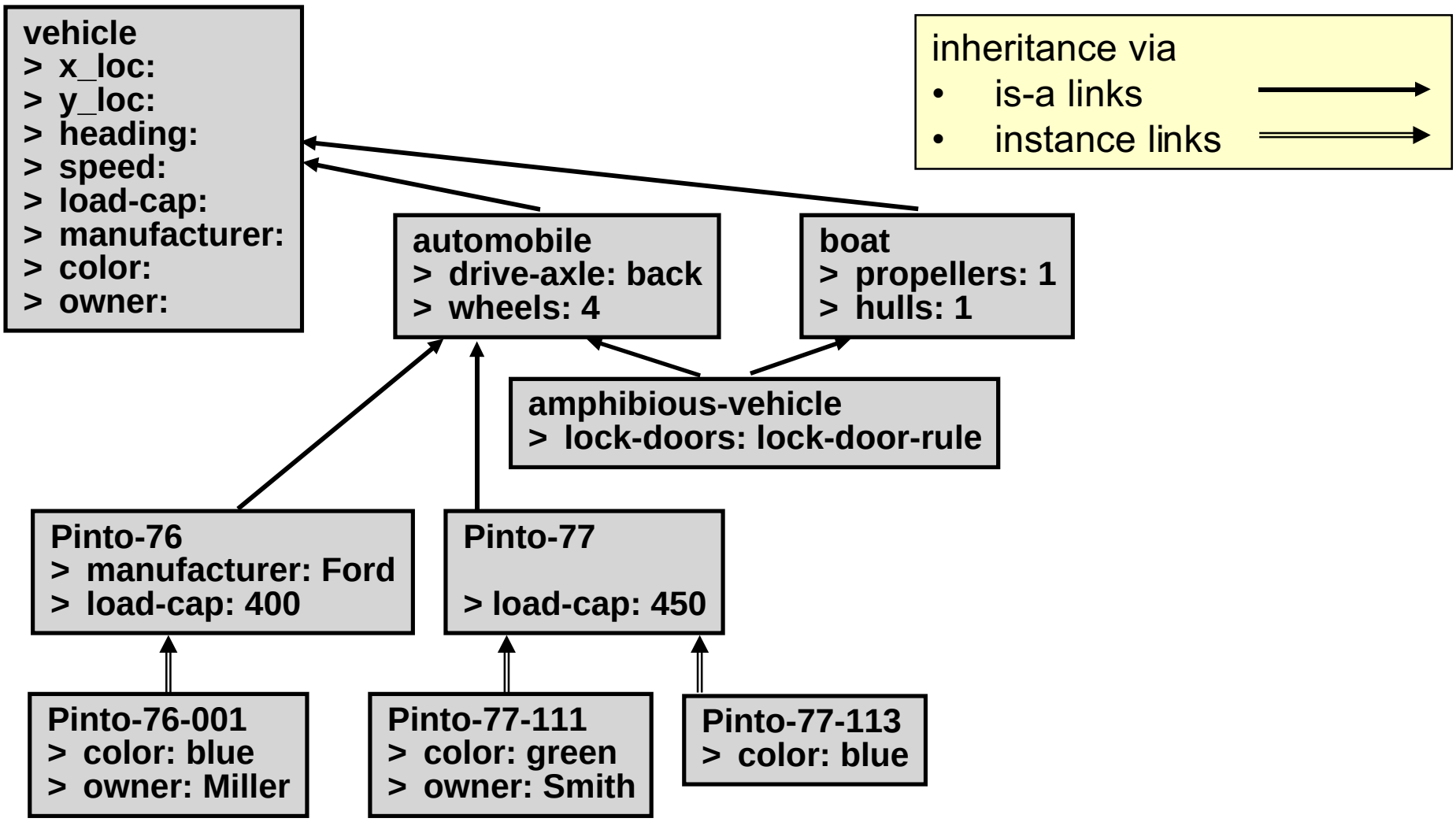
Example:

$$prop(X, weight, light) \leftarrow$$
$$prop(X, is\_a, lemon\_laptop\_10000).$$
$$prop(X, is\_a, lemon\_computer) \leftarrow$$
$$prop(X, is\_a, lemon\_laptop\_10000).$$

Multiple Inheritance

- An individual is usually a member of more than one class. For example, the same person may be a mother, a teacher, a football coach,….
- The individual can inherit the properties of all of the classes it is a member of: multiple inheritance.
- If there are default values, we can have a problem when an individual inherits conflicting defaults from the different classes: multiple inheritance problem.

Knowledge base with frames



Choosing Primitive and Derived Relations

- Associate an attribute value with the most general class with that attribute value.
- Don’t associate contingent properties of a class with the class. For example, if all of current computers just happen to be brown.
- Axiomatize in the causal direction. You want knowledge that is stable as the world changes.