

Choosing Objects and Relations

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

Choosing Objects and Relations

How to represent: “Pen #7 is red.”

- `red(pen7)`. It's easy to ask “What's red?”
Can't ask “what is the color of `pen7`?”

Choosing Objects and Relations

How to represent: “Pen #7 is red.”

- `red(pen7)`. It's easy to ask “What's red?”
Can't ask “what is the color of *pen₇*?”
- `color(pen7, red)`. It's easy to ask “What's red?”
It's easy to ask “What is the color of *pen₇*?”
Can't ask “What property of *pen₇* has value *red*?”

Choosing Objects and Relations

How to represent: “Pen #7 is red.”

- `red(pen7)`. It's easy to ask “What's red?”
Can't ask “what is the color of *pen₇*?”
- `color(pen7, red)`. It's easy to ask “What's red?”
It's easy to ask “What is the color of *pen₇*?”
Can't ask “What property of *pen₇* has value *red*?”
- `prop(pen7, color, red)`. It's easy to ask all these questions.

Choosing Objects and Relations

How to represent: “Pen #7 is red.”

- `red(pen7)`. It's easy to ask “What's red?”
Can't ask “what is the color of *pen₇*?”
- `color(pen7, red)`. It's easy to ask “What's red?”
It's easy to ask “What is the color of *pen₇*?”
Can't ask “What property of *pen₇* has value *red*?”
- `prop(pen7, color, red)`. It's easy to ask all these questions.

`prop(Object, Property, Value)` is the only relation needed:

`object-property-value representation`

To represent “a is a parcel”

- $prop(a, type, parcel)$, where *type* is a special property
- $prop(a, parcel, true)$, where *parcel* is a Boolean property

- To represent *scheduled(cs422, 2, 1030, cc208)*. “section 2 of course *cs422* is scheduled at 10:30 in room *cc208*.”
- Let *b123* name the booking:
 - prop(b123, course, cs422)*.
 - prop(b123, section, 2)*.
 - prop(b123, time, 1030)*.
 - prop(b123, room, cc208)*.
- 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.

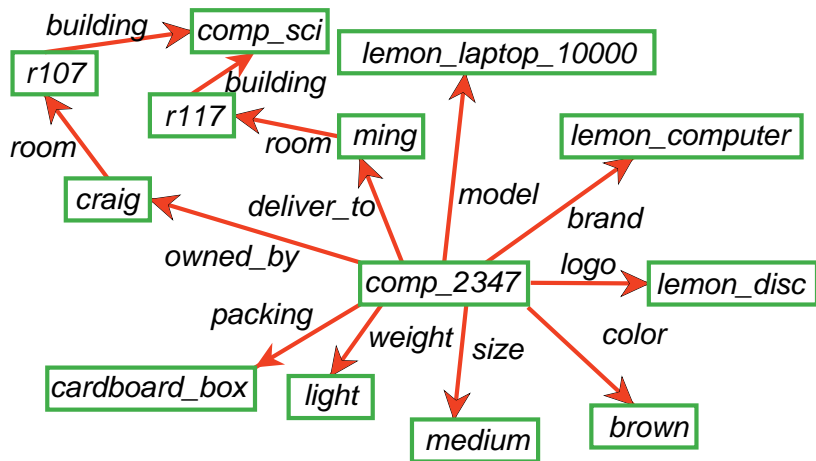
Write

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, r107).  
prop(r107, building, comp_sci).  
  
⋮
```

Turtle: a simple language of triples

A triple is written as

Subject Verb Object.

A comma can group objects with the same subject and verb.

$S V O_1, O_2.$ is an abbreviation for $\begin{matrix} S V O_1. \\ S V O_2. \end{matrix}$

A semi-colon can group verb-object pairs for the same subject.

$S V_1 O_1; V_2 O_2.$ is an abbreviation for $\begin{matrix} S V_1 O_1. \\ S V_2 O_2. \end{matrix}$

Square brackets can be used to define an individual that is not given an identifier. It can then be used as the object of a triple.

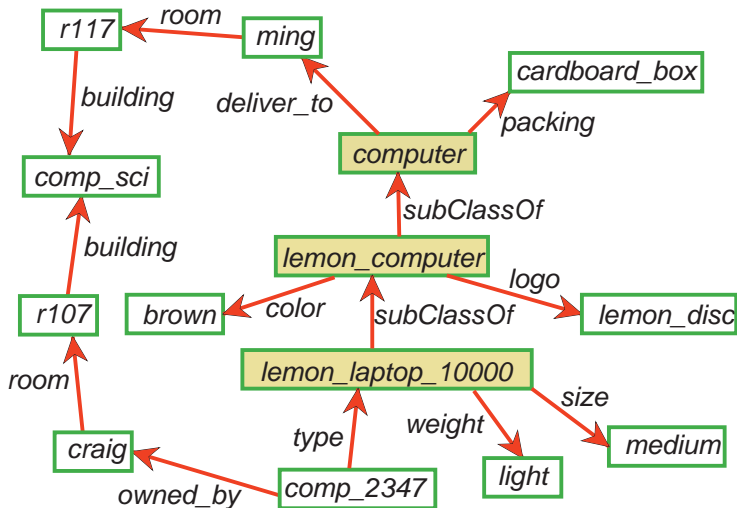
Turtle Example

```
<comp_3645> <#owned_by> <#fran>;  
            <#color> <#green>, <#yellow>;  
            <#managed_by> [ <#occupation> <#sys_admin>;  
                           <#serves_building> <#comp_sci>].
```

Primitive versus Derived Properties

- **Primitive knowledge** is that which is defined explicitly by facts.
- **Derived knowledge** is knowledge defined by rules.
- a **class** is a set of individuals that are grouped together as they have similar properties.
- **Example:** All lemon laptops may have have *size = medium*. Associate this property with the class, not the individual.
- Allow a special property **type** between an individual and a class.
- Use a special property **subClassOf** between two classes that allows for **property inheritance**.

A Structured Semantic Network



Logic of Property

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

$$\begin{aligned} \text{prop}(\text{Obj}, p, n) \leftarrow \\ \text{prop}(\text{Obj}, \text{type}, c). \end{aligned}$$

Example:

$$\begin{aligned} \text{prop}(X, \text{weight}, \text{light}) \leftarrow \\ \text{prop}(X, \text{type}, \text{lemon_laptop_10000}). \\ \text{prop}(X, \text{packing}, \text{cardboard_box}) \leftarrow \\ \text{prop}(X, \text{type}, \text{computer}). \end{aligned}$$

You can do inheritance through the subclass relationship:

$$\begin{aligned} \text{prop}(X, \text{type}, T) \leftarrow \\ \text{prop}(S, \text{subClassOf}, T) \wedge \\ \text{prop}(X, \text{type}, S). \end{aligned}$$

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.

Choosing Primitive and Derived Properties

- Associate an property value with the most general class with that property value.
- Don't associate contingent properties of a class with the class. For example, if all of current computers just happen to be brown.