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.”

$\text{red}(\text{pen}_7)$. It’s easy to ask “What’s red?”

Can’t ask “what is the color of $\text{pen}_7$?”

$\text{color}(\text{pen}_7, \text{red})$. It’s easy to ask “What’s red?”

It’s easy to ask “What is the color of $\text{pen}_7$?”

Can’t ask “What property of $\text{pen}_7$ has value $\text{red}$?”
Choosing Objects and Relations

How to represent: “Pen #7 is red.”

\( \text{red}(pen_7) \). It’s easy to ask “What’s red?”
Can’t ask “what is the color of \( pen_7 \)?”

\( \text{color}(pen_7, \text{red}) \). 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 \( \text{red} \)?”

\( \text{prop}(pen_7, \text{color}, \text{red}) \). It’s easy to ask all these questions.
Choosing Objects and Relations

How to represent: “Pen #7 is red.”

\[ \text{red}(pen_7). \] It’s easy to ask “What’s red?”
Can’t ask “what is the color of \( pen_7 \)?”

\[ \text{color}(pen_7, \text{red}). \] 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 \text{red}?”

\[ \text{prop}(pen_7, \text{color}, \text{red}). \] It’s easy to ask all these questions.

\[ \text{prop} \left( \text{Object}, \text{Attribute}, \text{Value} \right) \] is the only relation needed:
\[ \text{object-attribute-value representation} \]
Universality of *prop*

To represent “a is a parcel”

1. $\text{prop}(a, \text{is}_a, \text{parcel})$, where $\text{is}_a$ is a special attribute

2. $\text{prop}(a, \text{parcel}, \text{true})$, where $\text{parcel}$ is a Boolean attribute
To represent \( \text{scheduled}(cs422, 2, 1030, cc208) \). “section 2 of course \( cs422 \) is scheduled at 10:30 in room \( cc208 \).”

Let \( b123 \) name the booking:

\[
\begin{align*}
\text{prop} & (b123, \text{course}, cs422). \\
\text{prop} & (b123, \text{section}, 2). \\
\text{prop} & (b123, \text{time}, 1030). \\
\text{prop} & (b123, \text{room}, cc208).
\end{align*}
\]

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

\[ Obj \rightarrow_{Att} Val \]
An Example Semantic Network

- building: comp_sci
- room: r107
- room: r117
- owned_by: craig
- deliver_to: ming
- building: room
- model: lemon_laptop_10000
- brand: lemon_computer
- logo: lemon_disc
- packing: cardboard_box
- weight: light
- size: medium
- color: brown

© David Poole, Alan Mackworth, Russell哥, and Oxford University Press 1998-2002
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).

::
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.

```plaintext
[owned_by = craig,
 deliver_to = ming,
 model = lemon_laptop_10000,
 brand = lemon_computer, 
 logo = lemon_disc, 
 color = brown, 
 ⋮]
```
**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 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

comp_2347

owned_by
craig

room
r107

building

comp_sci

ming

result_of

deliver_to

building

r117

room

cardboard_box

packing

computer

is_a

lemon_computer

logo

brown

color

is_a

lweight

size

medium

lemon_laptop_10000

light

is_a

is_a

comp_2347

craig

owned_by

David Poole, Alan Mackworth, Randy Goebel and Oxford University Press 1998-2002
Logic of Property Inheritance

An arc $p \rightarrow n$ 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.
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.