Ontologies and Knowledge-based Systems

- Is there a flexible way to represent relations?
- How can knowledge bases be made to interoperate semantically?
Choosing Individuals and Relations

How to represent: “Pen #7 is red.”
How to represent: “Pen #7 is red.”

- \textit{red}(pen_7). It’s easy to ask “What’s red?”
  Can’t ask “what is the color of \textit{pen}_7?”
Choosing Individuals and Relations

How to represent: “Pen #7 is red.”

- \textit{red}(\textit{pen}_7). It’s easy to ask “What’s red?”
  Can’t ask “what is the color of \textit{pen}_7?”

- \textit{color}(\textit{pen}_7, \textit{red}). It’s easy to ask “What’s red?”
  It’s easy to ask “What is the color of \textit{pen}_7?”
  Can’t ask “What property of \textit{pen}_7 has value \textit{red}?”
Choosing Individuals and Relations

How to represent: “Pen #7 is red.”

- \textit{red}(\textit{pen}_7). It’s easy to ask “What’s red?”
  Can’t ask “what is the color of \textit{pen}_7?”

- \textit{color}(\textit{pen}_7, \textit{red}). It’s easy to ask “What’s red?”
  It’s easy to ask “What is the color of \textit{pen}_7?”
  Can’t ask “What property of \textit{pen}_7 has value \textit{red}?”

- \textit{prop}(\textit{pen}_7, \textit{color}, \textit{red}). It’s easy to ask all these questions.
Choosing Individuals and Relations

How to represent: “Pen 7 is red.”

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

- \textit{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?”

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

\textit{prop}(\textit{Individual}, \textit{Property}, \textit{Value}) is the only relation needed:
called \textit{individual-property-value representation}
or \textit{triple representation}
Universality of $prop$

To represent “$a$ is a parcel”
Universality of *prop*

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 \textit{scheduled}(cs422, 2, 1030, cc208). “section 2 of course cs422 is scheduled at 10:30 in room cc208.”
To represent \textit{scheduled}(cs422, 2, 1030, cc208). “section 2 of course cs422 is scheduled at 10:30 in room cc208.”

Let \( b123 \) name the booking:

\begin{itemize}
\item \( prop(b123, \text{course}, \text{cs422}) \).
\item \( prop(b123, \text{section}, 2) \).
\item \( prop(b123, \text{time}, 1030) \).
\item \( prop(b123, \text{room}, \text{cc208}) \).
\end{itemize}

We have reified the booking.

Reify means: to make into an individual.

What if we want to add the year?
When you only have one relation, prop, it can be omitted without loss of information.

Logic:

\[ \text{prop}(\text{Individual}, \text{Property}, \text{Value}) \]

triple:

\[ \langle \text{Individual}, \text{Property}, \text{Value} \rangle \]

simple sentence:

\[ \text{Individual Property Value} \]

graphically:
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).

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

$$prop(Obj, p, v) \leftarrow$$
$$prop(Obj, type, c).$$

Example:

$$prop(X, weight, light) \leftarrow$$
$$prop(X, type, lemon\_laptop\_10000).$$
$$prop(X, packing, cardboard\_box) \leftarrow$$
$$prop(X, type, computer).$$
You can do inheritance through the subclass relationship:

\[
\text{prop}(X, \text{type}, T) \leftarrow \\
\text{prop}(S, \text{subClassOf}, T) \land \\
\text{prop}(X, \text{type}, S).
\]
An individual is usually a member of more than one class. For example, the same person may be a wine expert, a teacher, a football coach, . . . .

The individual can inherit the properties of all of the classes it is a member of: multiple inheritance.

With default values, what is 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.