Ontologies and Knowledge-based Systems

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

How to represent: “Pen #7 is red.”
Choosing Individuals and Relations

How to represent: “Pen #7 is red.”

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

\textcolor{red}{prop(Individual, Property, Value)} is the only relation needed:
called individual-property-value representation or triple representation.

©D. Poole and A. Mackworth 2010  Artificial Intelligence, Lecture 13.1, Page 3
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 Individuals and Relations

How to represent: “Pen #7 is red.”

- \textit{red(}pen_{7}\textit{)}: It’s easy to ask “What’s red?”
  Can’t ask “what is the color of \textit{pen}{}_{7}\textit{?”}

- \textit{color(}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}\textit{?”}
  Can’t ask “What property of \textit{pen}{}_{7}\textit{ has value \textit{red}?”

- \textit{prop(}pen_{7}, \textit{color, red)}: It’s easy to ask all these questions.
Choosing Individuals and Relations

How to represent: “Pen #7 is red.”

- \textcolor{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?”
- \textcolor{green}{\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}?”
- \textcolor{blue}{\textit{prop}(\textit{pen}_7, \textit{color}, \textit{red})}: It’s easy to ask all these questions.

\textit{prop(Individual, Property, Value)} is the only relation needed:
called \textcolor{yellow}{individual-property-value representation}
or \textcolor{yellow}{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 \( \text{scheduled}(cs422, 2, 1030, cc208) \). “section 2 of course cs422 is scheduled at 10:30 in room cc208.”
Reification

- To represent \( \text{scheduled}(\text{cs422}, 2, 1030, \text{cc208}) \). “section 2 of course cs422 is scheduled at 10:30 in room cc208.”
- Let \( b123 \) name the booking:
  - \( \text{prop}(b123, \text{course}, \text{cs422}) \).
  - \( \text{prop}(b123, \text{section}, 2) \).
  - \( \text{prop}(b123, \text{time}, 1030) \).
  - \( \text{prop}(b123, \text{room}, \text{cc208}) \).
- We have \textbf{reified} the booking.
- Reify means: to make into an individual.
- What if we want to add the year?
Semantics Networks

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

Logic:

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

triple:

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

simple sentence:

*Individual Property Value.*

graphically:

```
Obj       Prop
\arrow{->}  \arrow{->}
   Val       Obj
```
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).
  ;
A triple is written as

\[ \text{Subject Verb Object}. \]

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

\[ S\ V\ O_1, O_2. \] is an abbreviation for \[ S\ V\ O_1. S\ V\ O_2. \]

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 \[ S\ V_1\ O_1. S\ V_2\ O_2. \]

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

\(
\langle \text{comp\_3645} \rangle \langle \#\text{owned\_by} \rangle \langle \#\text{fran} \rangle ;
\langle \#\text{color} \rangle \langle \#\text{green} \rangle , \langle \#\text{yellow} \rangle ;
\langle \#\text{managed\_by} \rangle [ \langle \#\text{occupation} \rangle \langle \#\text{sys\_admin} \rangle ;
\langle \#\text{serves\_building} \rangle \langle \#\text{comp\_sci} \rangle ].
\)
**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 computers may have $color = brown$. 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**.
Logic of Property

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 $p$:

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

Example:

$$\text{prop}(X, weight, light) \leftarrow \text{prop}(X, type, \text{lemon\_laptop\_10000}).$$

$$\text{prop}(X, packing, \text{cardboard\_box}) \leftarrow \text{prop}(X, type, \text{computer}).$$
You can do inheritance through the subclass relationship:

\[
\text{prop}(X, \text{type}, T) \leftarrow \\
\text{prop}(S, \text{subClassOf}, T) \wedge \\
\text{prop}(X, \text{type}, S).
\]
Multiple Inheritance

- 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**.
Choose 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.