

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."

- $\text{red}(\text{pen}_7)$. It's easy to ask "What's red?"
Can't ask "what is the color of pen_7 ?"

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

Choosing Individuals 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 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 pen_7 ? ”
Can't ask “What property of pen_7 has value red ? ”
- $\text{prop}(\text{pen}_7, \text{color}, \text{red})$. It's easy to ask all these questions.

Choosing Individuals 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 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 pen_7 ?"
Can't ask "What property of pen_7 has value red?"
- $\text{prop}(\text{pen}_7, \text{color}, \text{red})$. It's easy to ask all these questions.

$\text{prop}(\text{Individual}, \text{Property}, \text{Value})$ is the only relation needed:

called individual-property-value representation

or 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

Reification

- To represent $scheduled(cs422, 2, 1030, cc208)$. “section 2 of course *cs422* is scheduled at 10:30 in room *cc208*.”

Reification

- 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 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:

$$prop(Individual, Property, Value)$$

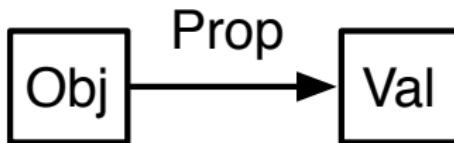
triple:

$$\langle Individual, Property, Value \rangle$$

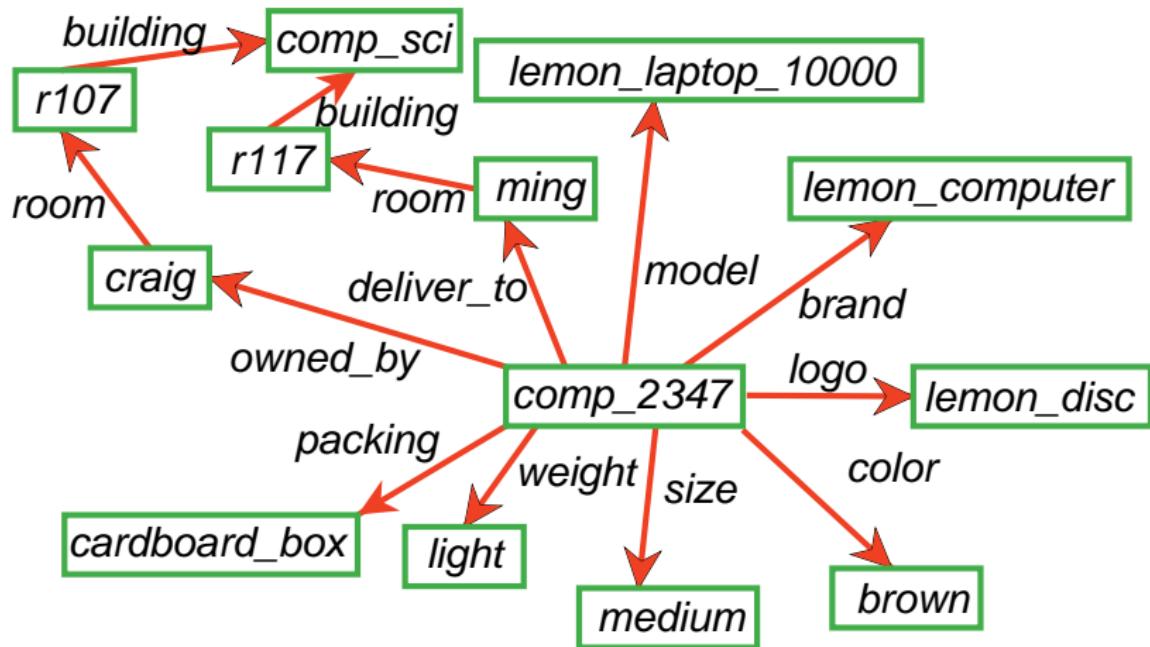
simple sentence:

Individual Property Value.

graphically:



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 $\begin{array}{c} S \ V \ O_1. \\ S \ V \ O_2. \end{array}$

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 $\begin{array}{c} S \ V_1 \ O_1. \\ S \ V_2 \ O_2. \end{array}$

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.

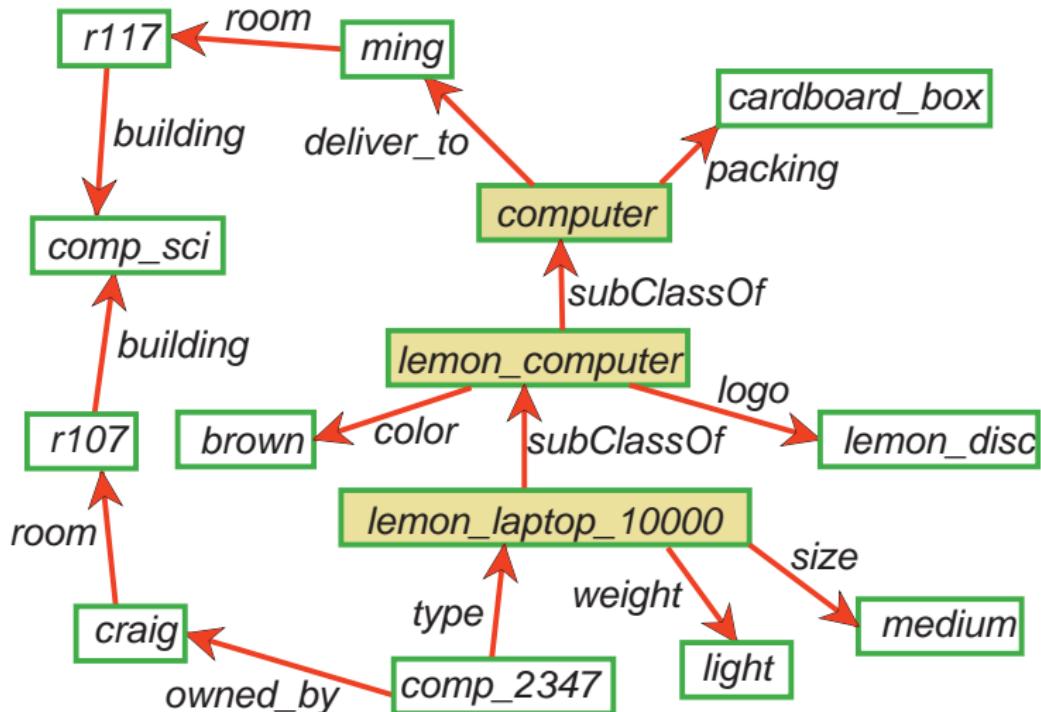
Turtle Example

```
<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 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 .

A Structured Semantic Network



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 :

```
prop(Obj, p, v) ←  
    prop(Obj, type, c).
```

Example:

```
prop(X, weight, light) ←  
    prop(X, type, lemon_laptop_10000).  
prop(X, packing, cardboard_box) ←  
    prop(X, type, computer).
```

Logic of Property Inheritance

You can do inheritance through the subclass relationship:

```
prop(X, type, T) ←  
    prop(S, subClassOf, T) ∧  
    prop(X, 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.

Choosing Primitive and Derived Properties

- Associate a 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.