Semantic Networks and Frames

Reference: Bratko ed. 3, section 15.7 / page 372-

Aim:
To describe semantic networks and frames. Semantic nets are a simple way of representing the relationships between entitities and concepts. Frames can do the things that semantics networks do, but take a more object-oriented type approach. Frames allow procedures called demons to be attached to their slots greatly increasing the power of this knowledge representation method.
Keywords: ako, default facet in a frame, demon, if_added demon, if_removed demon, if_replaced demon, if_needed demon, if_new demon, range demon, help demon, cache facet, multi-valued facet, facet, frame, generic frame, inheritance, instance frame, isa, procedural attachment, semantic network, slot
Plan:
• commonsense reasoning using defaults and inheritance • semantic network examples • quantification and semantic networks • frames, slots, procedural attachment, demons • `cylinder` example • `person`, `measure`, `ph_frame` example

Logic and Rules

So far we have been dealing with rule-based systems. These have the following attributes:

- Symbolic representation
- Inference mechanism allows conclusions to be drawn from facts and rules.
- Certainty factors allow rules to deal with uncertainty. [These will be covered later.]

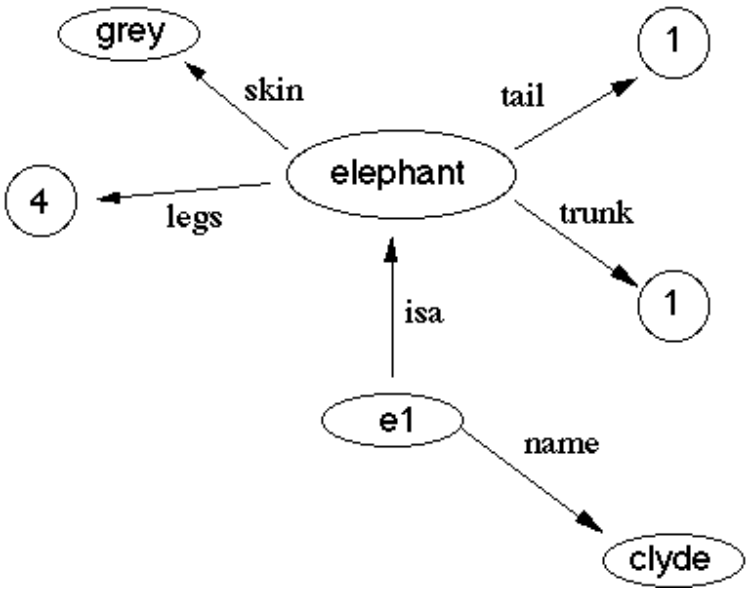
Common Sense Reasoning

However, rules are cumbersome as a way of encoding relational knowledge and knowledge about objects.

- People have vast background knowledge to cope with everyday situations.
- We don't have to be told everything explicitly because we can call on the background knowledge.
- We use `default' knowledge to handle situations where knowledge is incomplete.
- This is called common sense reasoning.

Defaults and Inheritance

- Defaults and inheritance are ways of achieving some commonsense:
- Inheritance is a way of reasoning by default, that is, when information is missing, fall back to defaults.
- Semantic networks represent inheritance.

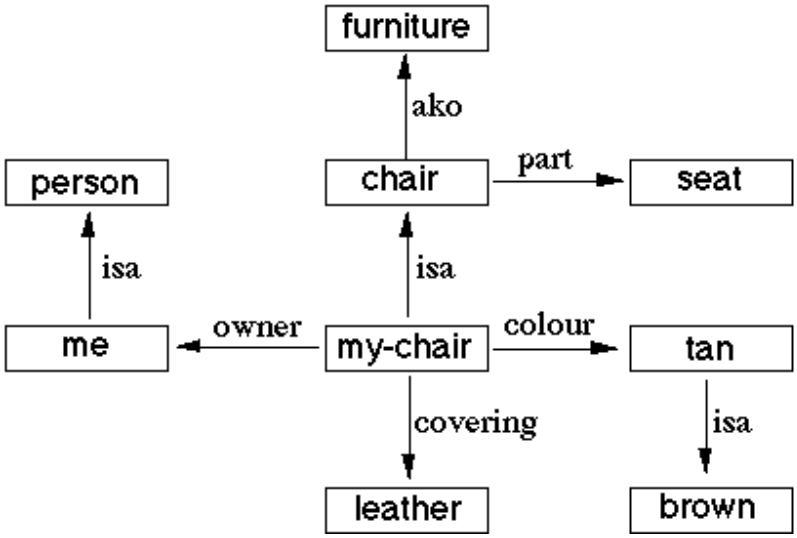


Using Inheritance

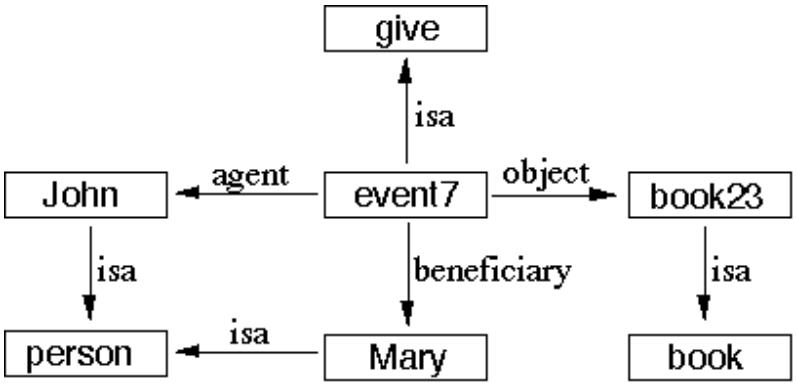
- To find the value of a property of *e1*, first look at *e1*.
- If the property is not attached to that node, "climb" the *isa* link to the node's parent and search there.
  - *isa* signifies set membership: $\in$
  - *ako* signifies the subset relation: $\subseteq$
- Repeat, using *isa*/*ako* links, until the property is found or the inheritance hierarchy is exhausted.
- Sets of things in a semantic network are termed **types**.
- Individual objects in a semantic network are termed **instances**.
- Here is Prolog code for doing inheritance with semantic nets.

Examples of Semantic Networks

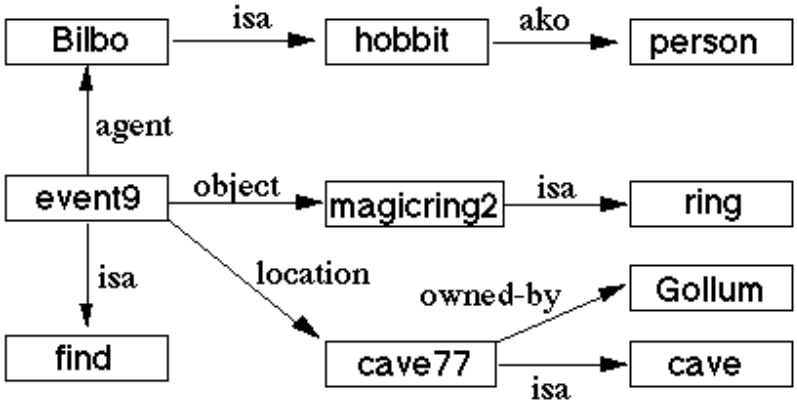
State: I own a tan leather chair.



Event: John gives the book to Mary.



Complex event:Bilbo finds the magic ring in Gollum's cave.



Frames

- Frames implement semantic networks.

- They add **procedural attachment**.
- A frame has **slots** and slots have **values**.
- A frame may be **generic**, i.e. it describes a class of objects.
- A frame may be an **instance**, i.e. it describes a particular object.
- Frames can inherit properties from generic frames.

Demons

Demons are attached to slots to cause side effects when the slot is accessed.

if_added
: demons are triggered when a new value is put into a slot.

if_removed
: demons are triggered when a value is removed from a slot.

if_replaced
: is triggered when a slot value is replaced.

if_needed
: demons are triggered when there is no value present in an instance frame and a value must be computed from a generic frame.

More Demons

if_new
: is triggered when a new frame is created.

range
: is triggered when a new value is added. The value must satisfy the range

help
: is triggered when the range demon is triggered and returns false.

Demon-related facets

cache
: means that when a value is computed it is stored in the instance frame. condition.

multi_valued
: means that the slot may contain more than one value.

A Simple Frame Example

Documentation for the frame language used in this example is available (but the system only works in iProlog: it's pretty unusual for a Prolog system to support frames directly).

```
cylinder ako object with
  height:
    range      number(new value) and new value > 0
    help       print("Height must be a positive number")
    if_needed  ask
    if_removed remove volume from this cylinder
    cache      yes;
  radius:
    range      number(new value) and new value > 0
```

```
    help       print("Radius must be a positive number")
    if_needed  ask
    if_removed remove cross_section from this cylinder
    cache      yes;
  cross_section:
    if_needed  pi * radius of this cylinder ^ 2
    if_removed remove volume from this cylinder
    cache      yes;
  volume:
    if_needed  cross_section of this cylinder *
                 height of this cylinder
    cache      yes!
```

A More Complicated Example

```
person ako object with
  name:
    range      atom(new value)
    help       print("The name should be a string.")
    if_new     ask
    cache      yes;
  sex:
    range      new value in [male, female]
    help       print("Sex can only be male or female, not ", new value)
    if_needed  ask
    if_replaced print("Are you sure you want a sex change?")
    if_removed print("Are you sure you want the sex removed?")
    cache      yes;
  year_of_birth:
    range      year of current_date - 120 .. year of current_date
    help       print("Invalid year of birth.")
    if_needed  ask("Year of birth")
    cache      yes;
  age:
    cache      yes
    if_needed  year of current_date - year_of_birth of this person;
  parents:
    multivalued yes
    range      new value must_be_a person
    help       print("The value in a parents slot must be a person.");
  height:
    range      10..220
    help       if new value < 10 then
                  print(new value, "cm is too short."),
               if new value > 220 then
                  print(new value, "cm is too tall."),
               print("The height should be between 10 and 220cm.")
    if_needed  ask("What is the height of ", name of this person);
  weight:
    range      1..150
    if_needed  ask("What is the weight of ", name of this person)
```

```
    cache      yes
    if_added   if new value > 100 then
                  print("Your ", this slot, " is too high!");
  occupation:
    range      atom(new value)
    help       print("The occupation should be a string.")
    if_needed  ask("What is the occupation")
    cache      yes
    if_removed print("I used to be a ", old value, ".")!

measure ako object with
  current_value:
    range      allowable_low of this measure .. allowable_high of this measure
    help       print("The patient is dead!")
    if_new     ask(prompt of this measure)
    cache      yes
    if_added
        if      new value < expected_low of this measure then
                replace interpretation of this measure by low
        else if new value > expected_high of this measure then
                replace interpretation of this measure by high
        else    replace interpretation of this measure by normal
    if_replaced replace last_value of this measure by old value!
```

Comment: The first reference to interpretation of this measure has the effect of dynamically adding a slot to the measure frame. Similarly, references to low , normal , and high dynamically add these as possible values.
end of comment

```
ph_frame ako measure with
    prompt:             default "ph level";
    allowable_low:      default 6;
    allowable_high:     default 8;
    expected_high:      default 7.6;
    expected_low:       default 6.5!

'HCO3_frame' ako measure with
    prompt:             default "HCO3 level";
    allowable_low:      default 6;
    allowable_high:     default 8;
    expected_high:      default 7.6;
    expected_low:       default 6.5!

paCO2_frame ako measure with
    prompt:             default "paCO2 level";
    allowable_low:      default 6;
    allowable_high:     default 8;
    expected_high:      default 7.6;
    expected_low:       default 6.5!

patient ako person with
  ph:
    cache               yes
```

```
    if_needed   make [ph_frame];
  'HCO3':
    cache               yes
    if_needed   make ['HCO3_frame'];
  paCO2:
    cache               yes
    if_needed   make [paCO2_frame];
  diagnosis:
    multivalued yes
    default     [];
  investigation:
    if_new
        if      interpretation of ph of this patient = low
        then    add acidosis to diagnosis of this patient,

        if      interpretation of ph of this patient = high
        then    add alkalosis to diagnosis of this patient,

        if      interpretation of paCO2 of this patient = low
        then    add hypocarbic to diagnosis of this patient,

        if      interpretation of paCO2 of this patient = high
        then    add hypercarbic to diagnosis of this patient,

        if      acidosis in diagnosis of this patient
        and     interpretation of 'HCO3' of this patient = low
        then    add primary_metabolic_acidosis
                        to diagnosis of this patient!

current_date isa object with
    year:       2005;
    month:      4;
    day:        1!
```

Summary: Semantic Networks and Frames

- Semantic networks start out by encoding relationships like **isa** and **ako** but can represent quite complicated information on this simple basis.
- Frames organize knowledge around concepts considered to be of interest (like person and patient in the example code).
- A frame can be a **generic** frame (or template) or an **instance** frame.
- Frames also allow **procedural attachmen**t - that is, **demons** can be attached to slots so that the mere fact of creating a frame or accessing a slot can cause significant computation to be performed.

---

based on an earlier version by Claude Sammut.