Object-based representation

Luger, Part III, 6.0, 6.1, 6.2.2-6.2.4, 6.4 (skim)

Objects

Two basic forms of Structured Objects
- Semantic Nets
- Frames

Semantic Nets (Associative Nets)
- Components
  - Nodes - represent concepts (e.g. objects, events)
  - Links - represent relations between nodes

Example1: John took a big car
\( \exists x \exists t \; [\text{take}(\text{John}, x, t) \land \text{car}(x) \land \text{past}(t)] \)

Diagram:
- Nodes: John, Past, car, size, big
- Links: take, agent, time, object
More examples

Example 2:

```
A
 B
```

Example 3: Mary is younger than Jill

```
Mary
   age
      A1
Jill
   age
      A2
```

Example 4: Every dog has a tail

```
∀
```

Semantic Nets

- Correspond to human information storage & management
- Meaning of a concept = how it is connected to other concepts
**Semantic Nets**

- Correspond to human information storage & management
- Meaning of a concept = how it is connected to other concepts
- Can be extended to represent quantifiers (by labeled links) and procedure (by procedural attachment)
- Links
  - provide structure of which other knowledge can be inferred
  - usually represent binary relations but can represent non-binary relations
  - important common links
    - is-a: Class inclusion
    - instances: Class membership
    - has-part: Part-whole relation

```
Mammal          Ear
    │           └── infras
    ├── Type relations
      │        └── Elephant
      │                └── Dumbo
```

**Inheritance of Properties**

“instances” – inherit property of class members to an individual member
“isa” – inherit property of class to subclass

**Basic Algorithm:**

- Given a tree hierarchy
- **Input:** an instance object O, attribute A
- **Output:** O.A which is a value of A of object O

1. Find O in KB
2. If there is a value for A, report it, else if there is no “instance”, fail else move to the node destination of “instance”, X.
3. If X has A’ s value, report it, else Repeat Move to the node destination of “isa”. If there is a value there for A, report it

**Until** answer is reported or no more destination of “isa”
**Reasoning in Semantic Nets**

- Deduce new facts by traversing links in the network
  - Intersection Search
  - Inheritance property

**Example:** The system can infer that

- “Dumbo is a mammal” - by *intersection search*
- “Dumbo has ears” - by *inheritance property*

**Semantic Nets → Frame Systems**

- Frame systems try to improve problems of Semantic Nets
  - Use *slot structures* to sharpen type-token distinctions
  - Use *small number of links* to improve semantics
  - Use *default value* and *procedural attachment* to improve efficiency in processing information stored
  - Frames write general knowledge about class individuals in one place
    - structured, shared, centrally located knowledge
    - easier to build/update/modify/retrieve
  - Frame systems are generalized semantic nets
**Frame systems**

- Frames represent “typical” concepts
  - Frame Structure
    
    ![Frame Structure Diagram](image)

- Frame system is a collection of frames connecting one frame as an attribute value of another frame
- Slot values can be empty initially or can be
  - constant value
  - frame
  - procedure (executed when the slot value is activated)

---

**Example**

A frame system describing a typical hotel room

- **hotel-room**
  - is-a: room
  - location: hotel
  - contains: (hotel-chair, hotel-phone, hotel-bed)

- **hotel-chair**
  - is-a: chair
  - height: 20-40 cm
  - no-legs: 4
  - use: sitting

- **hotel-phone**
  - is-a: phone
  - use: (calling, room-service)
  - billing: through room

- **hotel-bed**
  - is-a: bed
  - use: sleeping
  - size: king
  - part: (mattress, frame)
Procedural Attachment

● Creates “demons” – a procedure invoked as a side effect of some other actions (not directly called) E.g., when a certain slot value is changed, system performs type checks or run consistency tests

● Typical demons, e.g.,
  ■ if-created: executed when an instance is created
  ■ if-needed: can override the slot value
  ■ if-read, if-written: a change in a slot’s value automatically reflected in another

Frame systems (contd)

Example: Mammals have ears. Elephants are mammals. They are grey. Dumbo is an elephant and has very large ear.

- Add “Elephant’s weight (lbs) is about 10 times the cube of its height (ft). Dumbo is 3 ft tall”
Tangled hierarchies

When hierarchies are not tree and no cycle (DAG)

Does Opus fly?
- System reports Ambiguity

Better representation
- System answers NO

Multiple inheritance is powerful but gives responsibility for designer to ensure consistency

Tangled hierarchies (contd.)

When hierarchies are not tree and no cycle (DAG)

Q1: Does Opus fly?
- Using the shortest inferential distance
  Opus $\Rightarrow$ Penguin
  Opus $\Rightarrow$ Penguin $\Rightarrow$ Bird
  Thus, Penguin is closer to Opus than Bird
  Answer: No (inherited from Penguin)

Q2: What does Opus eat?
- Penguin and Pet-Bird are of the same inferential distance to Opus.

Remedy: specify order of class precedence
Tangled hierarchies (contd.)

When hierarchies are not tree and no cycle (DAG)

Q1: How do we represent the fact that Opus eats fish but lives in a pet-house?

How many of these exceptions can we add?
→ Default vs. Exceptions

Semantic Nets and Frame systems

+ Easier to understand than logic
+ Easier to visualize than large set of rules
+ Fast attribute value retrieval
+ Eliminate storage of repetitive information by using inheritance property
  - Lack “standard” for naming and clear semantics
  - Poor at representing rule-like assertions
  - Hard to manage for very large set of knowledge
Other Variations

- Associative nets
  - associate concepts to human response
- Semantic Nets – too unconstrained
- Conceptual Dependency
  - offers primitive at computational cost
- Conceptual Graph
  - balance between the above two
- Script – represent sequence of events
Other Variations

● Associative networks

Other Variations (contd.)

● Associative networks
  ■ human response times to queries about objects are proportional to associated objects structured in memory
  ■ define meaning of an object in terms of a network of association with other objects

● Semantic networks
  ■ used in NLP to find meaning of text
  ■ for multiple meanings choose the shortest intersection path
  ■ Problems – too unconstrained, burden users
Other Variations (contd.)

- Conceptual dependency (Schank)
  - Offers primitives – more standard
  - Problems: computational infeasibility

Other Variations (contd.)

- Scripts (Schank)
  - Organize knowledge base in terms of sequence of events in certain situations, e.g., a restaurant script
  - Components: entry conditions, results, props, roles, scenes
  - Problems – matching based on keywords is not robust, inflexible; not possible to anticipate all situations that can break a script
Other Variations (contd.)

- **Frames (Minsky)**
  - adds power to semantic networks – allow complex objects to be represented as a single frame
  - sometimes known as schemas or class/objects
  - Frame systems led to OO systems
- **Conceptual Graph (Sowa)**
  - extended semantic networks
  - equivalent expressive power to logic
Declarative KRs and languages

Syntactic Systems
- Non-monotonic System
  - Predicate Logic
- Production Rules
  - CLIPS
  - OPS5
  - CYC

Semantic Systems
- Slot-and-Filler
  - Strong
    - Frame
    - Conceptual Dependency
    - Script
  - Weak
    - Semantic Nets
    - Conceptual Graph

Declarative KR

Summary of KR Themes

<table>
<thead>
<tr>
<th>AI Approaches</th>
<th>Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>Weak problem solving</td>
</tr>
<tr>
<td></td>
<td>Logic-based</td>
</tr>
<tr>
<td>Rules</td>
<td>Strong problem solving</td>
</tr>
<tr>
<td></td>
<td>Rule-based</td>
</tr>
<tr>
<td>Objects</td>
<td>Distributed and</td>
</tr>
<tr>
<td></td>
<td>Emergent problem solving</td>
</tr>
<tr>
<td></td>
<td>Object-based</td>
</tr>
<tr>
<td></td>
<td>Agent-based</td>
</tr>
</tbody>
</table>

Objects vs. Agents -- autonomy
(See more in Luger, Part III)
Summary of KR Themes (contd.)

● Weak problem solving
  ■ syntax-based strategies intended to use in general architectures for a wide variety of applications,
  ■ e.g., GPS (Newell & Simon)
    Automated reasoning, theorem proving
  ■ issues – no single heuristic can be used for all problem domain

Summary of KR Themes (contd.)

● Strong problem solving
  ■ uses explicit knowledge of a particular problem domain
  ■ e.g., expert systems
  ■ issues – acquisition and organization of large body of knowledge bases
Distributed/Emergent problem solving

- Three characteristics:
  - Situated (interact in its active environments, e.g., internet, teamwork)
  - Autonomous (interact on its own without intervention from other agents)
  - Flexible (both responsive and proactive)

- e.g., subsumption architecture in robotics, intelligent agents (with “social” characteristic as well) in various applications