Module 8

Other representation formalisms

8.1 Instructional Objective

- The students should understand the syntax and semantic of semantic networks
- Students should learn about different constructs and relations supported by semantic networks
- Students should be able to design semantic nets to represent real life problems
- The student should be familiar with reasoning techniques in semantic nets
- Students should be familiar with syntax and semantic of frames
- Students should be able to obtain frame representation of a real life problem
- Inferencing mechanism in frames should be understood.

At the end of this lesson the student should be able to do the following:

- Represent a real life problem in terms of semantic networks and frames
- Apply inferencing mechanism on this knowledge-base.

Lesson 19

Semantic nets

8. 2 Knowledge Representation Formalisms

Some of the abstract knowledge representation mechanisms are the following:

Simple relational knowledge

The simplest way of storing facts is to use a relational method where each fact about a set of objects is set out systematically in columns. This representation gives little opportunity for inference, but it can be used as the knowledge basis for inference engines.

- Simple way to store facts.
- Each fact about a set of objects is set out systematically in columns.
- Little opportunity for inference.
- Knowledge basis for inference engines.

Musician	Style	Instrument	Age
Miles Davis	Jazz	Trumpet	deceased
John Zorn	Avant Garde	Saxophone	35
Frank Zappa	Rock	Guitar	deceased
John Mclaughlin	Jazz	Guitar	47

We can ask things like:

- Who is dead?
- Who plays Jazz/Trumpet *etc.*?

This sort of representation is popular in database systems.

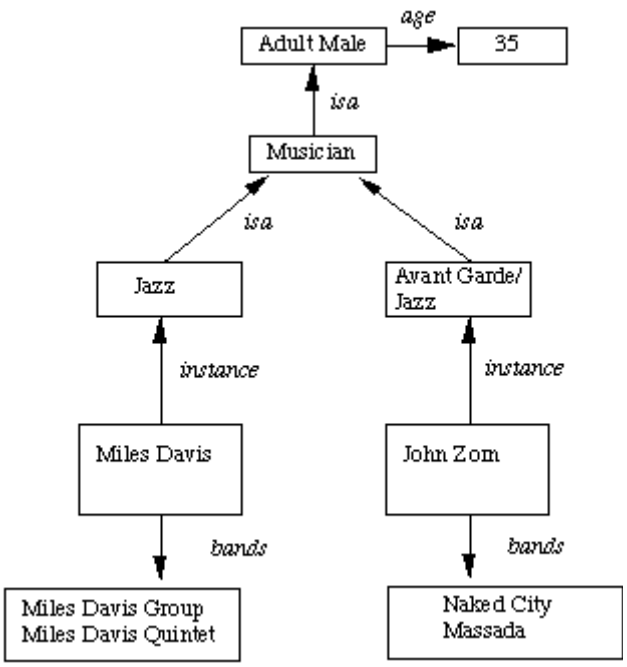
Inheritable knowledge

Relational knowledge is made up of objects consisting of

- attributes
- corresponding associated values.

We extend the base more by allowing inference mechanisms:

- Property inheritance
    - elements inherit values from being members of a class.
    - data must be organised into a hierarchy of classes.



- Boxed nodes -- objects and values of attributes of objects.
- Values can be objects with attributes and so on.
- Arrows -- point from object to its value.
- This structure is known as a slot and filler structure, semantic network or a collection of frames.

The algorithm to retrieve a value for an attribute of an instance object:

1. Find the object in the knowledge base
2. If there is a value for the attribute report it
3. Otherwise look for a value of instance if none fail
4. Otherwise go to that node and find a value for the attribute and then report it
5. Otherwise search through using *isa* until a value is found for the attribute.

Inferential Knowledge

Represent knowledge as *formal logic:*

All dogs have tails ∀x: *dog(x)* → *hasatail(x)* Advantages:

- A set of strict rules.
  - Can be used to derive more facts.
  - Truths of new statements can be verified.
  - Guaranteed correctness.
- Many inference procedures available to in implement standard rules of logic.
- Popular in AI systems. *e.g* Automated theorem proving.

Procedural knowledge

Basic idea:

- Knowledge encoded in some procedures
  - small programs that know how to do specific things, how to proceed.
  - *e.g* a parser in a natural language understander has the knowledge that a *noun phrase* may contain articles, adjectives and nouns. It is represented by calls to routines that know how to process articles, adjectives and nouns.

Advantages:

- *Heuristic* or domain specific knowledge can be represented.
- *Extended logical inferences*, such as default reasoning facilitated.
- *Side effects* of actions may be modelled. Some rules may become false in time. Keeping track of this in large systems may be tricky.

Disadvantages:

- Completeness -- not all cases may be represented.
- Consistency -- not all deductions may be correct.

  e.g If we know that *Fred is a bird* we might deduce that *Fred can fly*. Later we might discover that *Fred is an emu*.

- Modularity is sacrificed. Changes in knowledge base might have far-reaching effects.
- Cumbersome control information.

The following properties should be possessed by a knowledge representation system.

Representational Adequacy
-- the ability to represent the required knowledge;

Inferential Adequacy
- the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;

Inferential Efficiency
- the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;

Acquisitional Efficiency
- the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

To date no single system optimises all of the above. We will discuss in this module two formalisms, namely, semantic networks and frames, which trades off representational adequacy for inferential and acquisitional efficiency.

5.3 Semantic Networks

A semantic network is often used as a form of knowledge representation. It is a directed graph consisting of vertices which represent concepts and edges which represent semantic relations between the concepts.
The following semantic relations are commonly represented in a semantic net.

Meronymy (A is part of B)

Holonymy (B has A as a part of itself)

Hyponymy (or *troponymy*) (A is subordinate of B; A is kind of B)

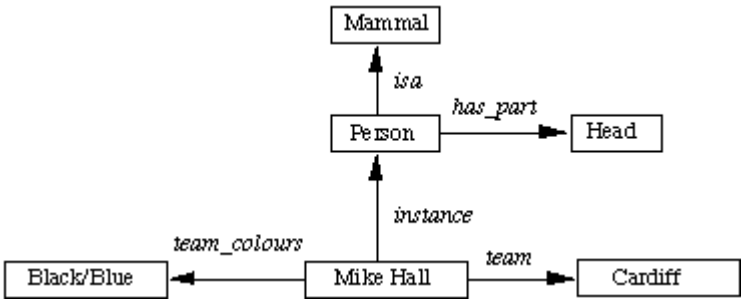
Hypernymy (A is superordinate of B)

Synonymy (A denotes the same as B)

Antonymy (A denotes the opposite of B)

Example

The physical attributes of a person can be represented as in the following figure using a semantic net.

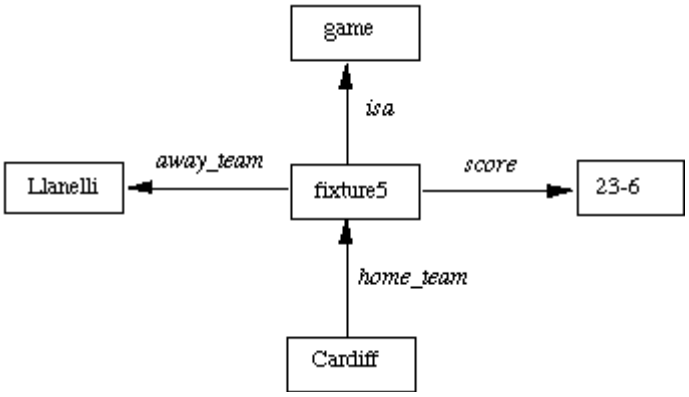


These values can also be represented in logic as: *isa(person, mammal), instance(Mike-Hall, person) team(Mike-Hall, Cardiff)*

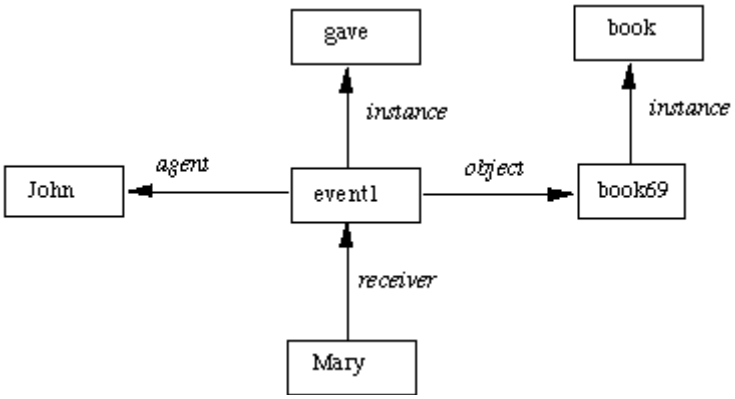
We know that conventional predicates such as *lecturer(dave)* can be written as *instance (dave, lecturer)* Recall that *isa* and *instance* represent inheritance and are popular in many knowledge representation schemes. But we have a problem: *How we can have more than 2 place predicates in semantic nets? E.g. score(Cardiff, Llanelli, 23-6)*

Solution:

- Create new nodes to represent new objects either contained or alluded to in the knowledge, *game* and *fixture* in the current example.
- Relate information to nodes and fill up slots.



As a more complex example consider the sentence: *John gave Mary the book*. Here we have several aspects of an event.



5.3.1 Inference in a Semantic Net

Basic inference mechanism: *follow links between nodes.*

Two methods to do this:

Intersection search

-- the notion that *spreading activation* out of two nodes and finding their intersection finds relationships among objects. This is achieved by assigning a special tag to each visited node.

Many advantages including entity-based organisation and fast parallel implementation. However very structured questions need highly structured networks.

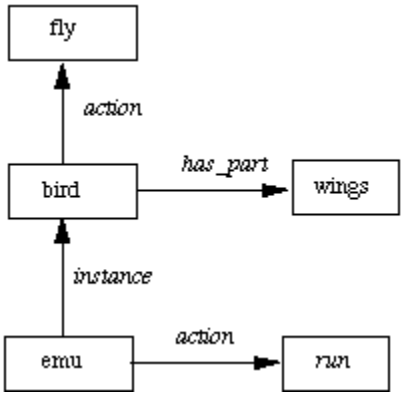
Inheritance

-- the *isa* and *instance* representation provide a mechanism to implement this.

Inheritance also provides a means of dealing with *default reasoning. E.g.* we could represent:

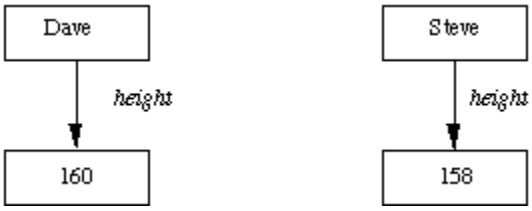
- Emus are birds.
- Typically birds fly and have wings.
- Emus run.

in the following Semantic net:

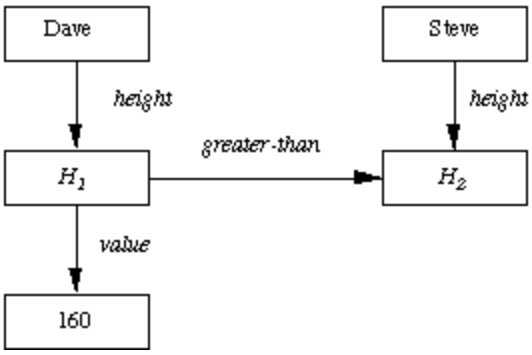


In making certain inferences we will also need to *distinguish between the link that defines a new entity and holds its value and the other kind of link that relates two existing entities*. Consider the example shown where the height of two people is depicted and we also wish to compare them.

We need extra nodes for the concept as well as its value.



Special procedures are needed to process these nodes, but without this distinction the analysis would be very limited.



Extending Semantic Nets

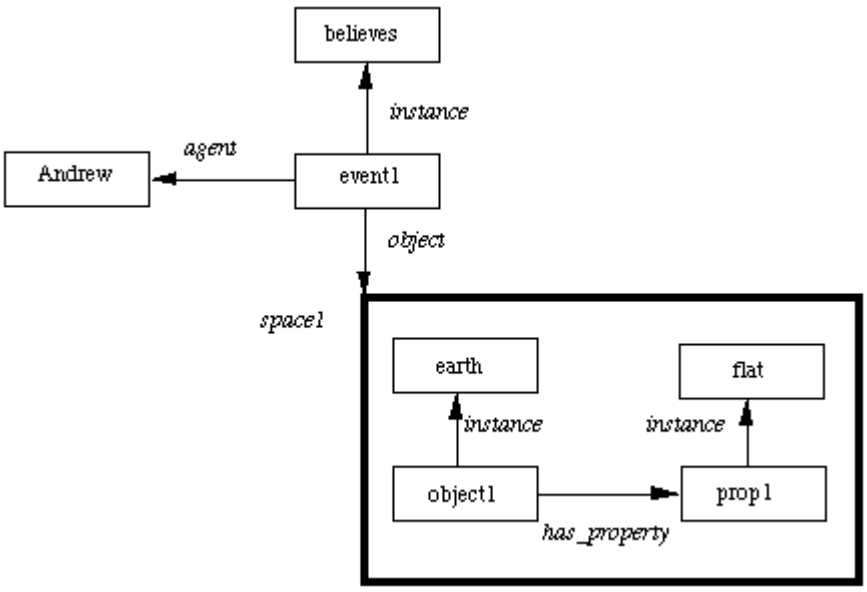
Here we will consider some extensions to Semantic nets that overcome a few problems or extend their expression of knowledge.

Partitioned Networks *Partitioned* Semantic Networks allow for:

- propositions to be made without commitment to truth.
- expressions to be quantified.

Basic idea: *Break network into **spaces** which consist of groups of nodes and arcs and regard each **space** as a node.*

Consider the following: *Andrew believes that the earth is flat.* We can encode the proposition *the earth is flat* in a *space* and within it have nodes and arcs the represent the fact (next figure). We can the have nodes and arcs to link this *space* the the rest of the network to represent Andrew's belief.



Now consider the quantified expression: *Every parent loves their child* To represent this we:

- Create a *general statement*, GS, special class.
- Make node g an instance of GS.
- Every element will have at least 2 attributes:
  - a *form* that states which relation is being asserted.
  - one or more *forall* (∀) or *exists* (∃) connections -- these represent universally quantifiable variables in such statements *e.g. x, y* in ∀*x* : *parent(x)* → ∃*y* : *child(y)* ∧ *loves(x,y)*

Here we have to construct two *spaces one* for each *x,y*. **NOTE:** We can express ∃ variables as *existentially qualified* variables and express the event of *love* having an agent *p* and receiver *b* for every parent *p* which could simplify the network.

Also If we change the sentence to *Every parent loves child* then the node of the object being acted on (*the child*) lies outside the form of the general statement. Thus it is not viewed as an existentially qualified variable whose value may depend on the agent. So we could construct a partitioned network as in the next figure.