IAI : Semantic Networks and Frames

Components of a Semantic Network

We can define a ***Semantic Network*** by specifying its fundamental components:

Lexical part	nodes – denoting objects links – denoting relations between objects labels – denoting particular objects and relations
Structural part	the links and nodes form directed graphs the labels are placed on the links and nodes
Semantic part	meanings are associated with the link and node labels (the details will depend on the application domain)
Procedural part	constructors allow creation of new links and nodes destructors allow the deletion of links and nodes writers allow the creation and alteration of labels readers can extract answers to questions

Clearly we are left with plenty of flexibility in creating these representations.

Semantic Networks as Knowledge Representations

Using Semantic Networks for representing knowledge has particular advantages:

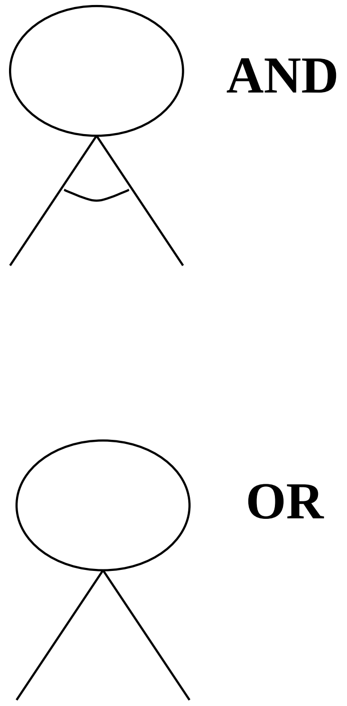
1. They allow us to structure the knowledge to reflect the structure of that part of the world which is being represented.
2. The semantics, i.e. real world meanings, are clearly identifiable.
3. There are very powerful representational possibilities as a result of "is a" and "is a part of" inheritance hierarchies.
4. They can accommodate a hierarchy of default values (for example, we can assume the height of an adult male to be 178cm, but if we know he is a baseball player we should take it to be 195cm).
5. They can be used to represent events and natural language sentences.

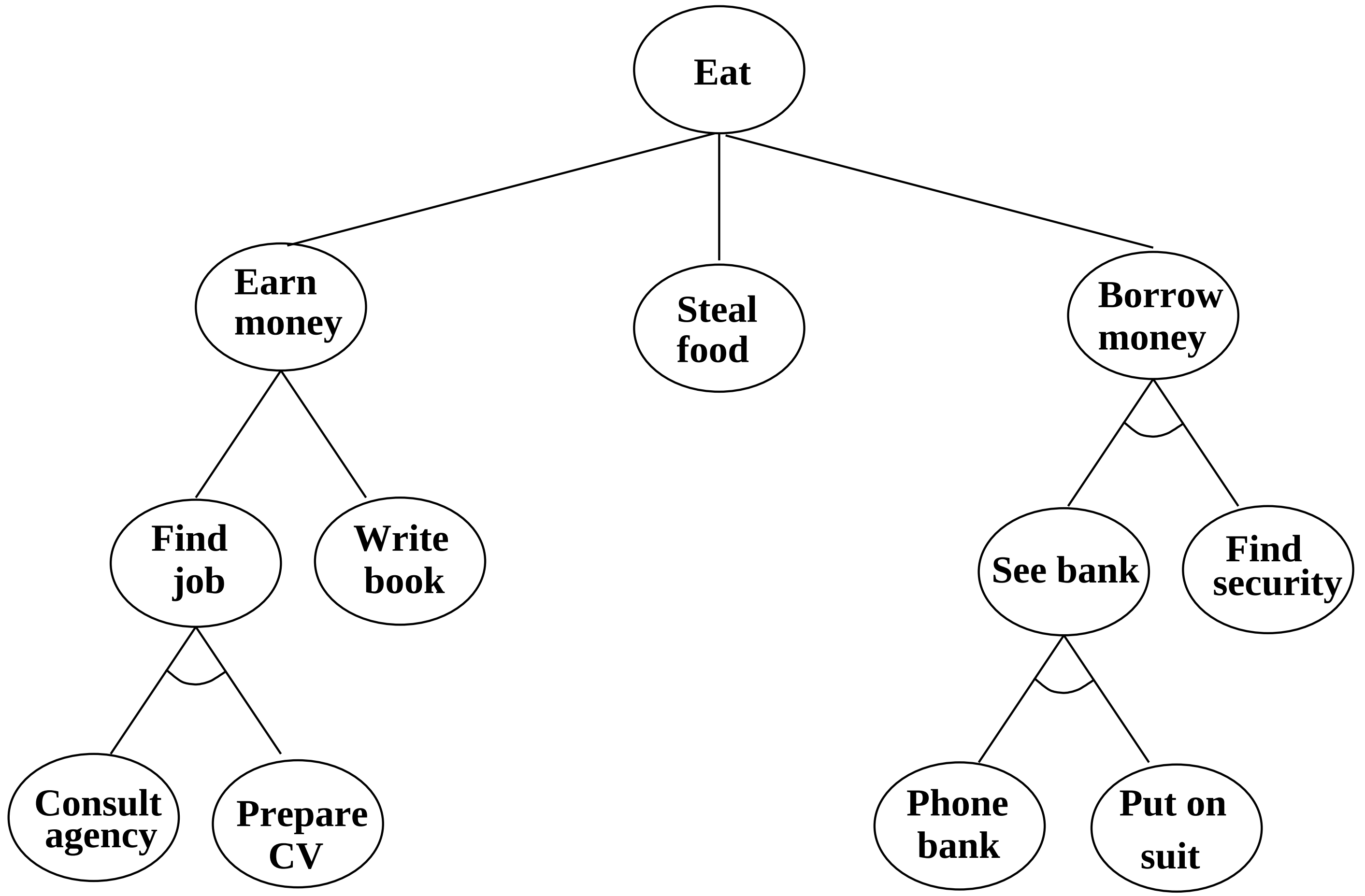
Clearly, the notion of a semantic network is extremely general. However, that can be a problem, unless we are clear about the syntax and semantics in each case.

AND / OR **Trees**

One particularly simple form of semantic network is an ***AND/OR Tree***. For example:

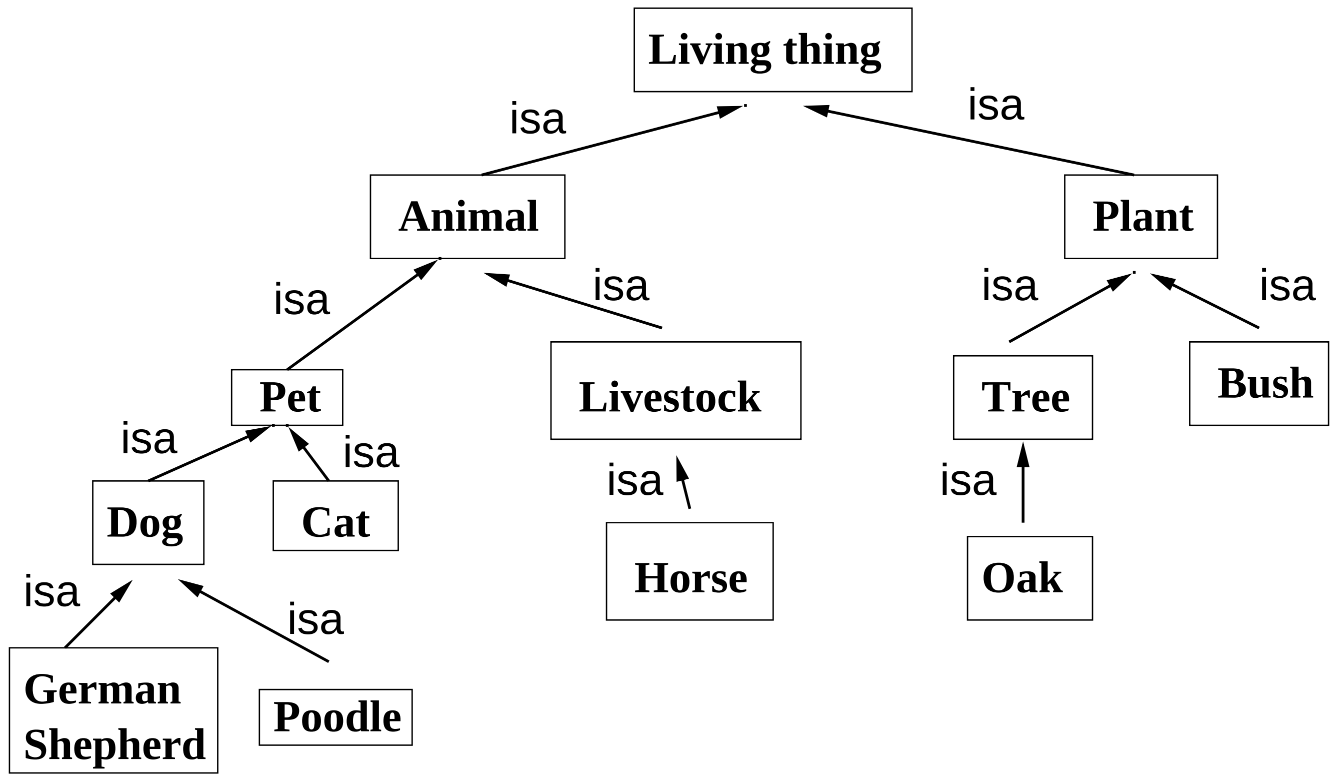
Two node types:





An *IS-A* Hierarchy

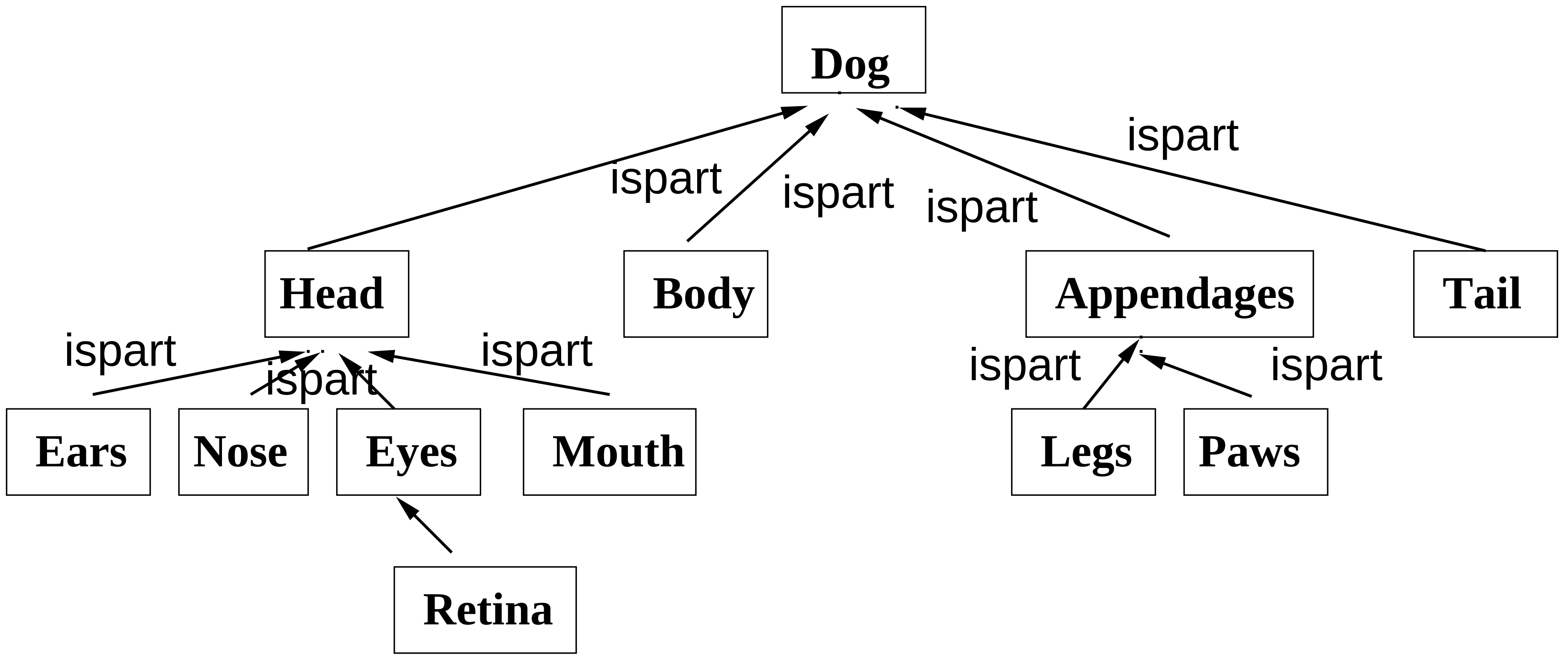
Another simple form of semantic network is an ***is-a hierarchy***. For example:



In set-theory terms, ***is-a*** corresponds to the sub-set relation $\subseteq$, and ***instance*** corresponds to the membership relation $\in$.

An *IS-PART* Hierarchy

If necessary, we can take the hierarchy all the way down to the molecular or atomic level with an ***is-part hierarchy***. For example:

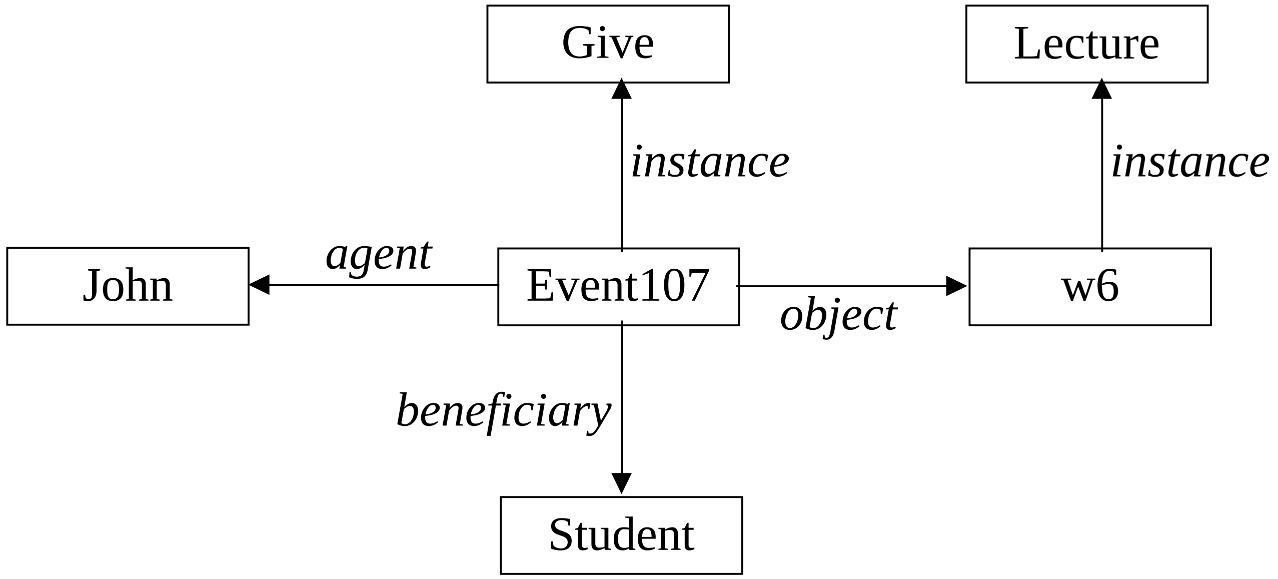


Naturally, where we choose to stop the hierarchy depends on what we want to represent.

Representing Events and Language

Semantic networks are also very good at representing events, and simple declarative sentences, by basing them round an “event node”. For example:

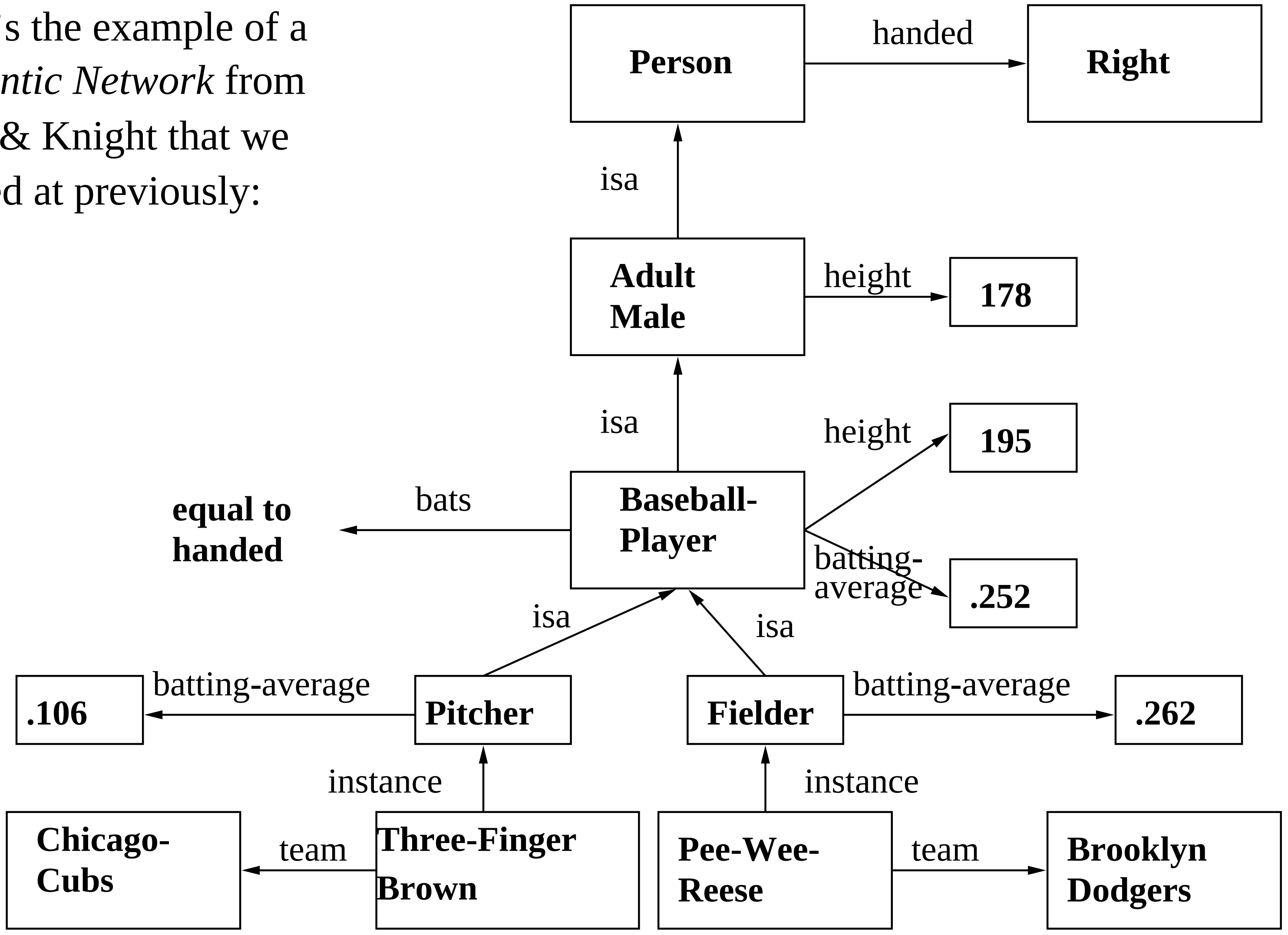
“John gave lecture w6 to his students”



In fact, several of the earliest semantic networks were English-understanding programs.

A Typical Mixed-Type Semantic Network

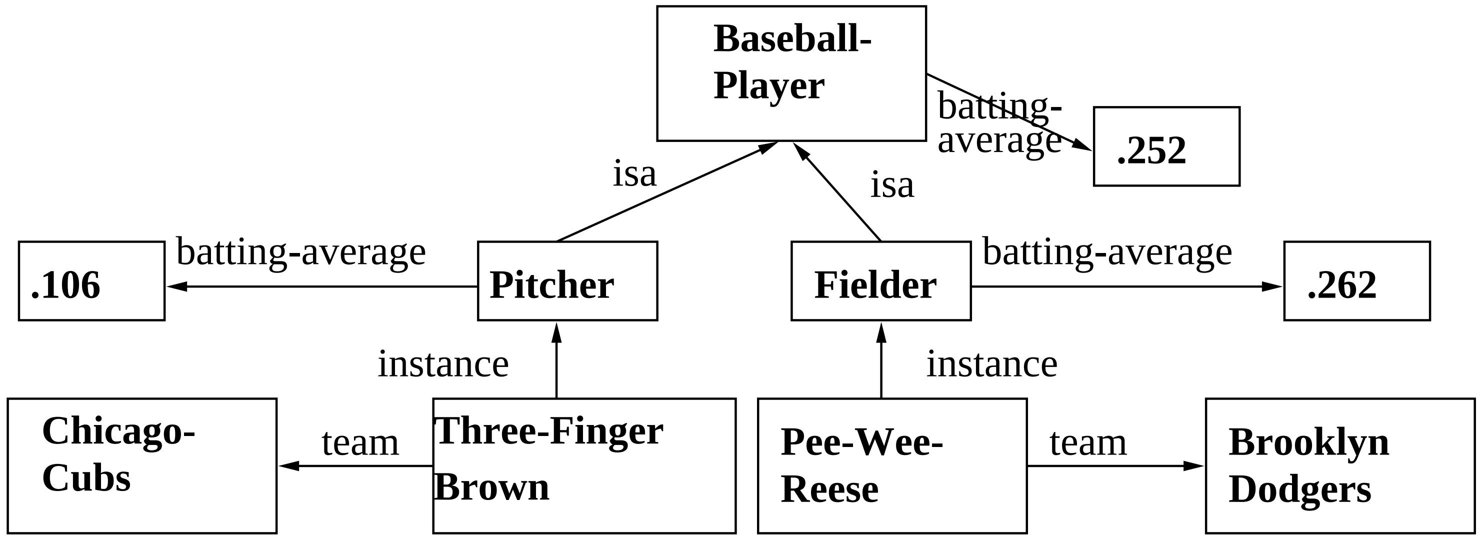
Here's the example of a *Semantic Network* from Rich & Knight that we looked at previously:



Intersection Search

One of the earliest ways that semantic networks were used was to find relationships between objects by spreading ***activation*** from each of two nodes and seeing where the activations met. This process is called ***intersection search***.

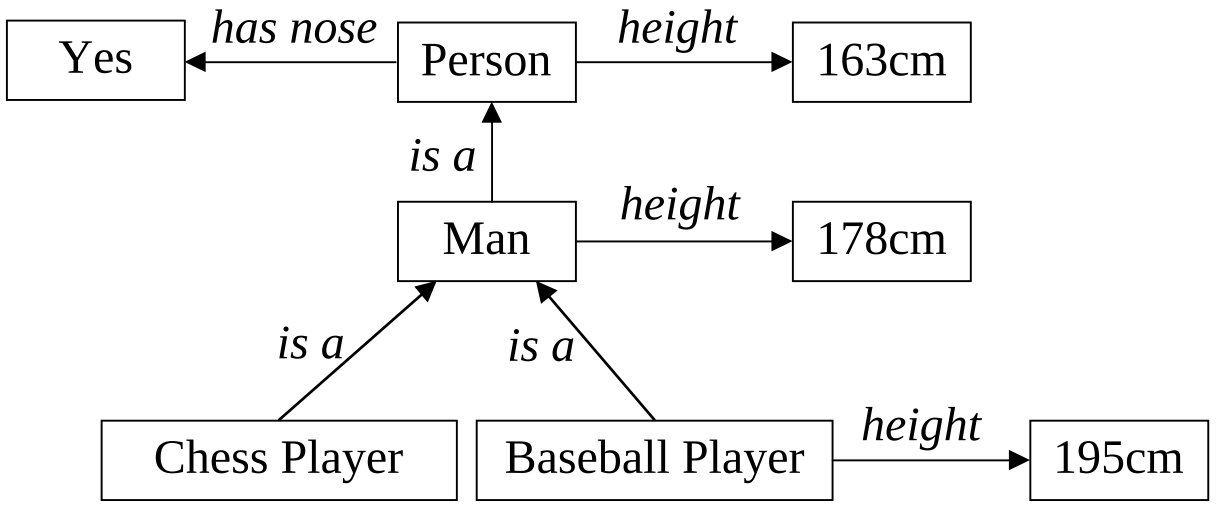
Question: "What is the relation between Chicago cubs and Brooklyn Dodgers?"



Answer: "They are both teams of baseball players."

Inheritance and Defaults

Two important features of semantic networks are the ideas of ***default*** (or typical) values and ***inheritance***. Consider the following section of a semantic network:

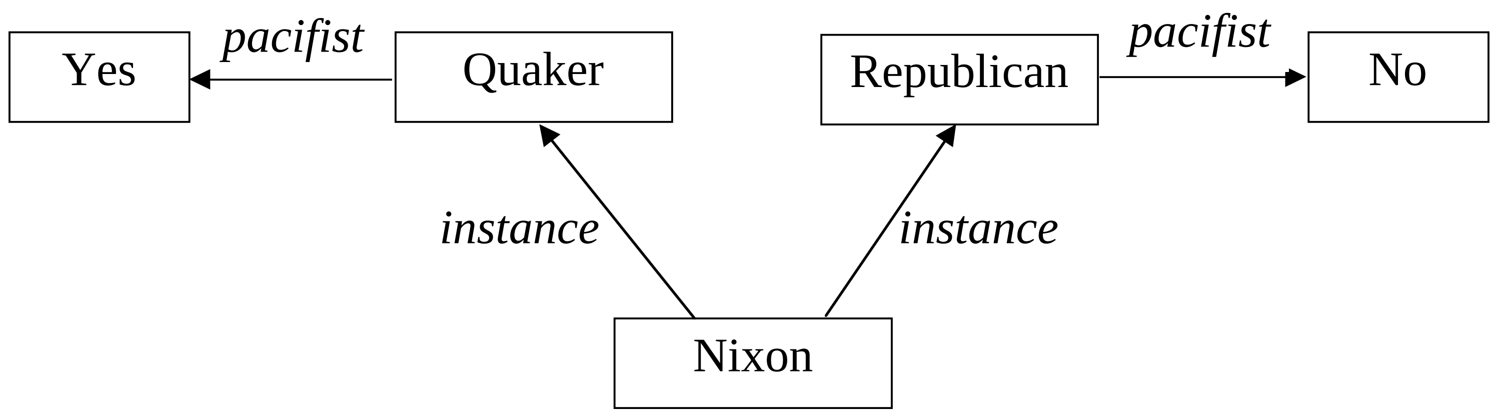


We can assign expected/default values of parameters (e.g. height, has nose) and inherit them from higher up the hierarchy. This is more efficient than listing all the details at each level. We can also ***over-ride*** the defaults. For example, baseball players are taller than average, so their default height over-rides the default height for men.

Multiple Inheritance

With simple trees, inheritance is straight-forward. However, when multiple inheritance is allowed, problems can occur. For example, consider this famous example:

Question: "Is Nixon a pacifist?"

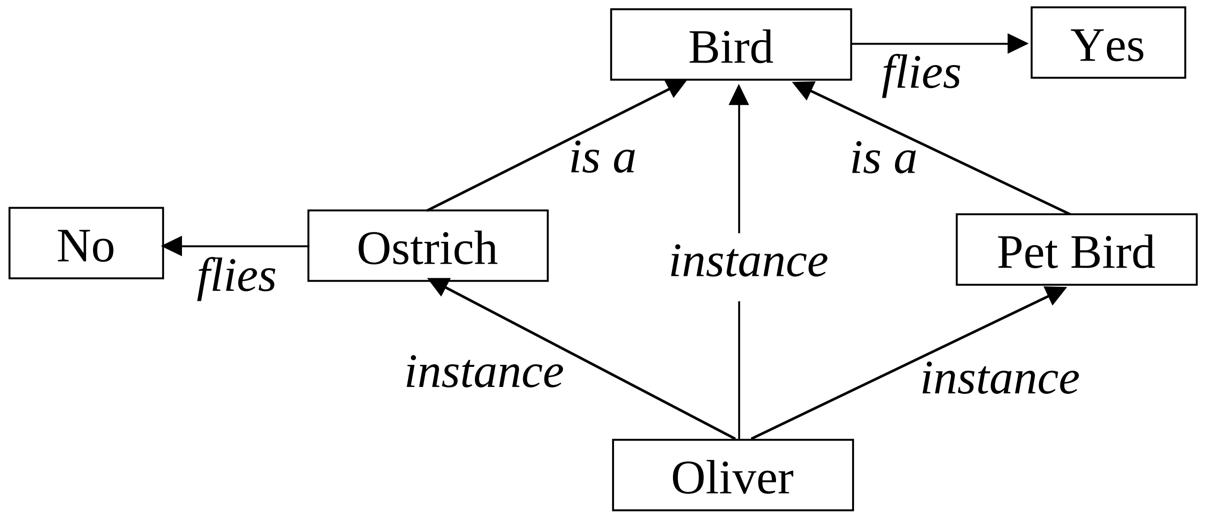


Conflicts like this are common is the real world. It is important that the inheritance algorithm reports the conflict, rather than just traversing the tree and reporting the first answer it finds. In practice, we aim to build semantic networks in which all such conflicts are either over-ridden, or resolved appropriately.

Tangled Hierarchies

Hierarchies that are not simple trees are called ***tangled hierarchies***. These allow another type of inheritance conflict. For example:

Question: “Can Oliver fly?”



A better solution than having a specific “flies no” for all individual instances of an ostrich, would be to have an algorithm for traversing the algorithm which guarantees that specific knowledge will always dominate over general knowledge. How?

Inferential Distance

Note that simply counting nodes as a measure of distance will not generally give the required results. Why?

Instead, we can base our inheritance algorithm on the ***inferential distance***, which can be used to define the concept of "closer" as follows:

> "*Node1* is closer to *Node2* than *Node3* if and only if *Node1* has an inference path through *Node2* to *Node3*, i.e. *Node2* is in between *Node1* and *Node3*."

Closer nodes in this sense will be more specific than further nodes, and so we should inherit any defaults from them.

Notice that inferential distance only defines a ***partial ordering*** – so it won't be any help with the Nixon example.

In general, the ***inferential engine*** will be composed of many procedural rules like this to define how the semantic network should be processed.

The Relation between Semantic Networks and Frames

The idea of ***semantic networks*** started out as a natural way to represent labelled connections between entities. But, as the representations are expected to support increasingly large ranges of problem solving tasks, the representation schemes necessarily become increasingly complex.

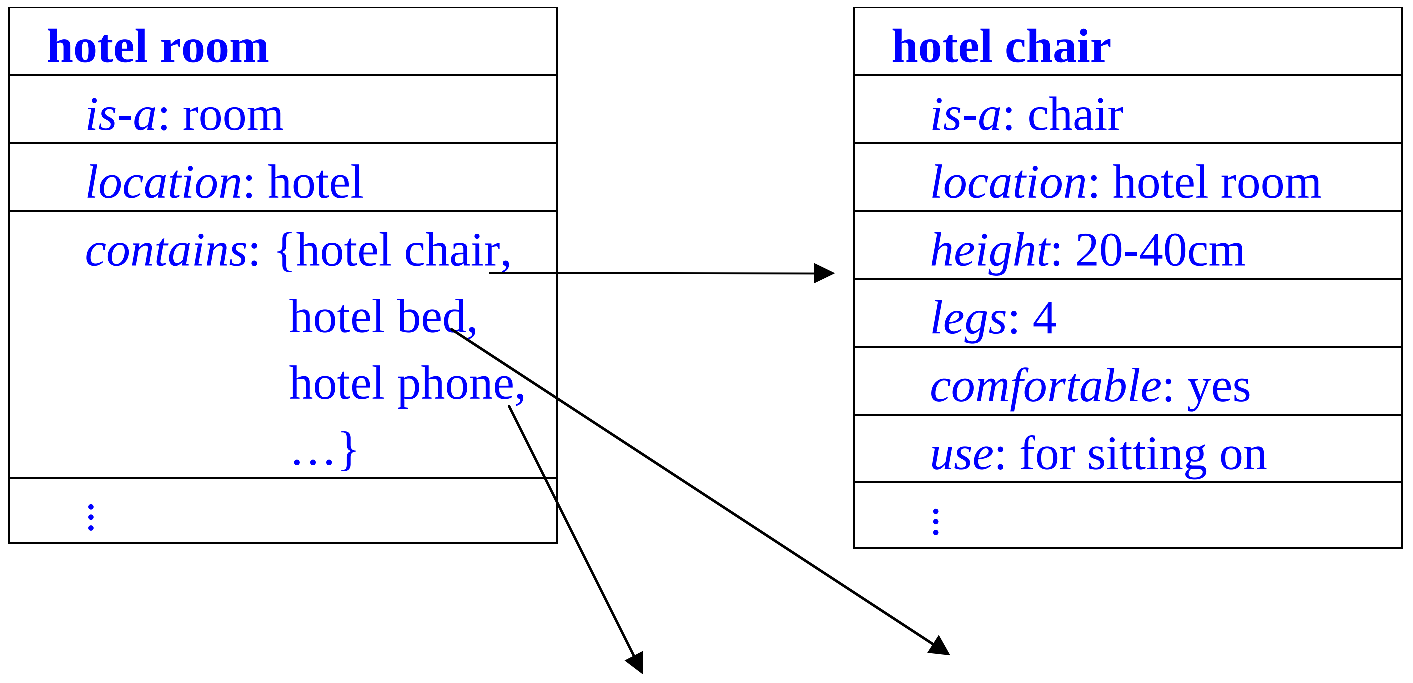
In particular, it becomes necessary to assign more structure to nodes, as well as to links. For example, in many cases we need node labels that can be computed, rather than being fixed in advance. It is natural to use database ideas to keep track of everything, and the nodes and their relations begin to look more like ***frames***.

In the literature, the distinction between frames and semantic networks is actually rather blurred. However, the more structure a system has, the more likely it is to be termed a frame system rather than a semantic network.

For our purposes, we shall use the ***practical distinction*** that semantic networks look like networks, and frames look like frames.

Frame Based Systems – The Basic Idea

A ***frame*** consists of a selection of slots which can be filled by values, or procedures for calculating values, or pointers to other frames. For example:



A complete frame based representation will consist of a whole hierarchy or network of frames connected together by appropriate links/pointers.

Frames as a Knowledge Representation

The simplest type of frame is just a data structure with similar properties and possibilities for knowledge representation as a semantic network, with the same ideas of ***inheritance*** and ***default values***.

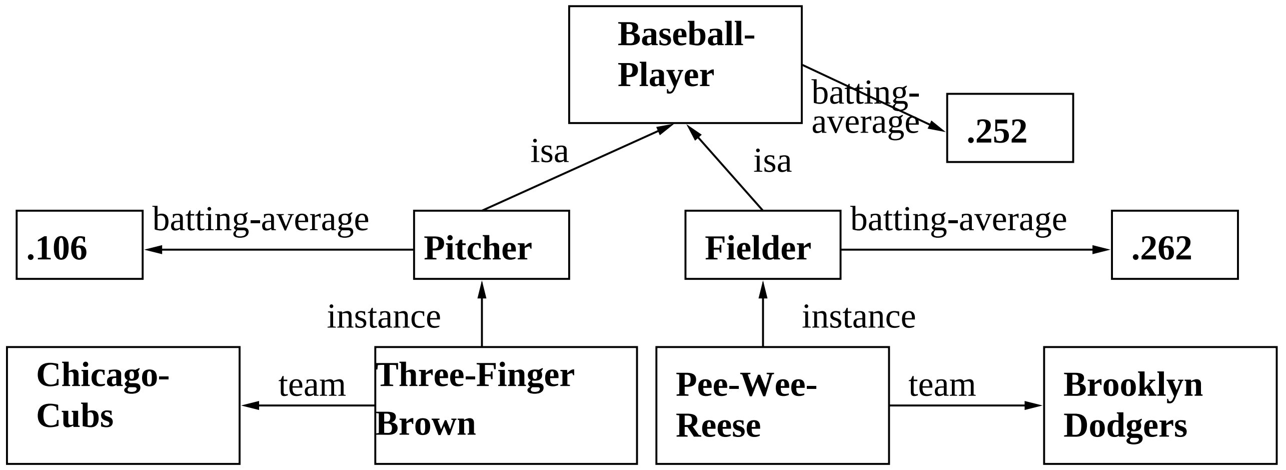
Frames become much more powerful when their slots can also contain instructions (procedures) for computing things from information in other slots or in other frames.

The ***original idea*** of frames was due to Minsky (1975) who defined them as “data-structures for representing stereotyped situations”, such as going into a hotel room.

This type of frames are now generally referred to as ***Scripts***. Attached to each frame will then be several kinds of information. Some information can be about how to use the frame. Some can be about what one can expect to happen next, or what one should do next. Some can be about what to do if our expectations are not confirmed. Then, when one encounters a new situation, one can select from memory an appropriate frame and this can be adapted to fit reality by changing particular details as necessary.

Converting between Semantic Networks and Frames

It is easy to construct frames for each node of a semantic net by reading off the links, e.g.



$\Rightarrow$

Baseball Player
is-a: Adult Male
batting average: .252
bats: equal to handed
team:
⋮

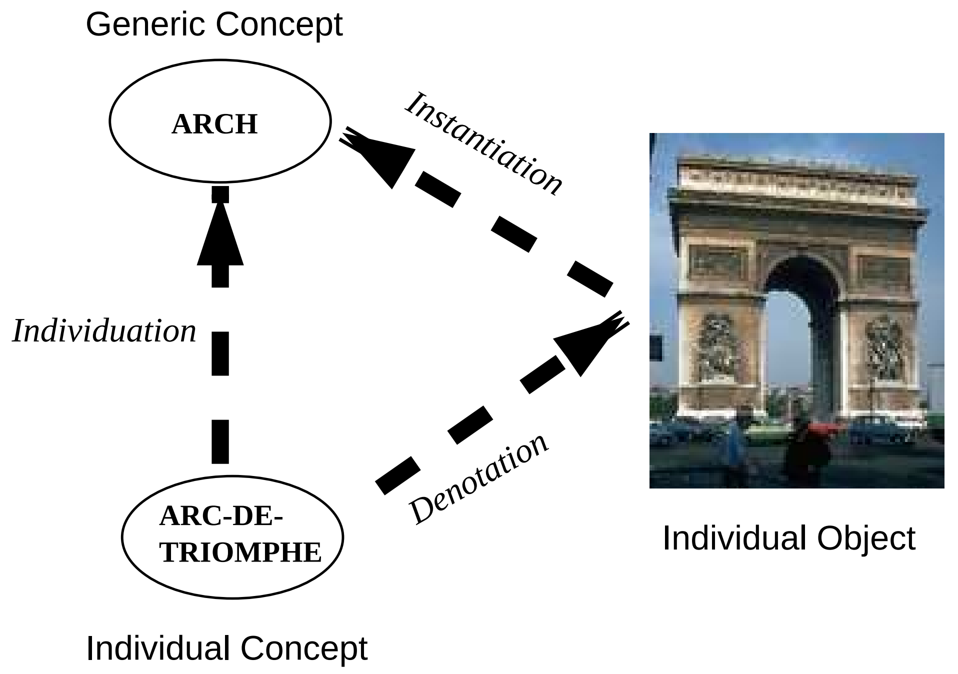
Fielder
is-a: Baseball player
batting average: .262

Pee-Wee-Reese
instance: Baseball player
team: Brooklyn Dodgers

• • •

Set Theory as a Basis For Frame Systems

The relationship between real world instances, the representation of instances, and the representation of sets/classes of instances, is already quite familiar, e.g.



Clearly *is-a* corresponds to subset ⊆, and *instance* corresponds to element ∈. Then set theory concepts such as ***transitivity***, ***intersection***, etc. apply automatically to our frames.

Slots as Fully-Fledged Objects

We have seen that frame based representations can be made much more powerful by allowing the slot fillers to become more than simple values. This includes being frames in their own right, with a full range of hierarchical arrangements, inheritance, etc. The main filler properties we generally want to represent are:

1. Details about whether the slot is single or multi-valued.
2. Constraints on the ranges of values or types of values.
3. Simple default values for the attribute.
4. Rules for inheriting values for the attribute.
5. Rules for computing values separately from inheritance.
6. The classes/frames to which it can be attached.
7. Inverse attributes.

Naturally, the frame system interpreter must know how to process such frames.

Overview and Reading

1. We began by looking at various common styles of semantic networks: *AND/OR* trees, *IS-A* and *IS-PART* hierarchies, and event/language networks.
2. The important procedural concepts of *Intersection Search*, *Inheritance*, *Defaults*, and *Inferential Conflict* were then covered.
3. We then looked at the relation between Semantic Networks and Frames, and how to convert from one to the other.
4. We ended by considering how slots can be promoted to fully-fledged objects with extremely general filler properties.

Reading

1. Rich & Knight: Chapters 9, 10
2. Winston: Chapter 2, 10
3. Jackson: Chapter 6
4. Russell & Norvig: Section10.6