AITA : Semantic Networks

1. Practical Aspects for Good Representations
2. Components of a Good Representation
3. Components of a Semantic Network
4. What Does "Semantics" Really Mean?
5. AND/OR Trees
6. IS-A and IS-PART Hierarchies
7. Representing Events and Language
8. Intersection Search
9. Inheritance and Defaults
10. Tangled Hierarchies and Inferential Distance

Practical Aspects for Good Representations

We have already looked at the general requirements for knowledge representation. Now consider the practical aspects of formulating good representations:

1. They must be ***computable*** – to be created with standard computing procedures.
2. They should make the important ***objects*** and ***relations*** explicit – so it is easy to see what is going on.
3. They need to ***bring together*** the objects and relations – so everything you need can be seen at once.
4. They should ***suppress irrelevant detail*** – so that rarely used details can be kept out of sight, but are still available when needed.
5. They should be ***transparent*** – so you can easily understand what is being said.
6. They need to be ***concise*** and ***fast*** – so information is stored and retrieved rapidly.
7. They should expose any natural ***constraints*** – so it is easy to express how one object or relation influences another.
8. They must be ***complete*** – so they can represent everything that needs representing.

Components of a Good Representation

For analysis purposes it is useful to be able to break any knowledge representation down into their four fundamental components:

1. The ***lexical*** part – that determines which symbols or words are used in the representation's ***vocabulary***.

2. The ***structural*** or ***syntactic*** part – that describes the ***constraints*** on how the symbols can be arranged, i.e. a grammar.

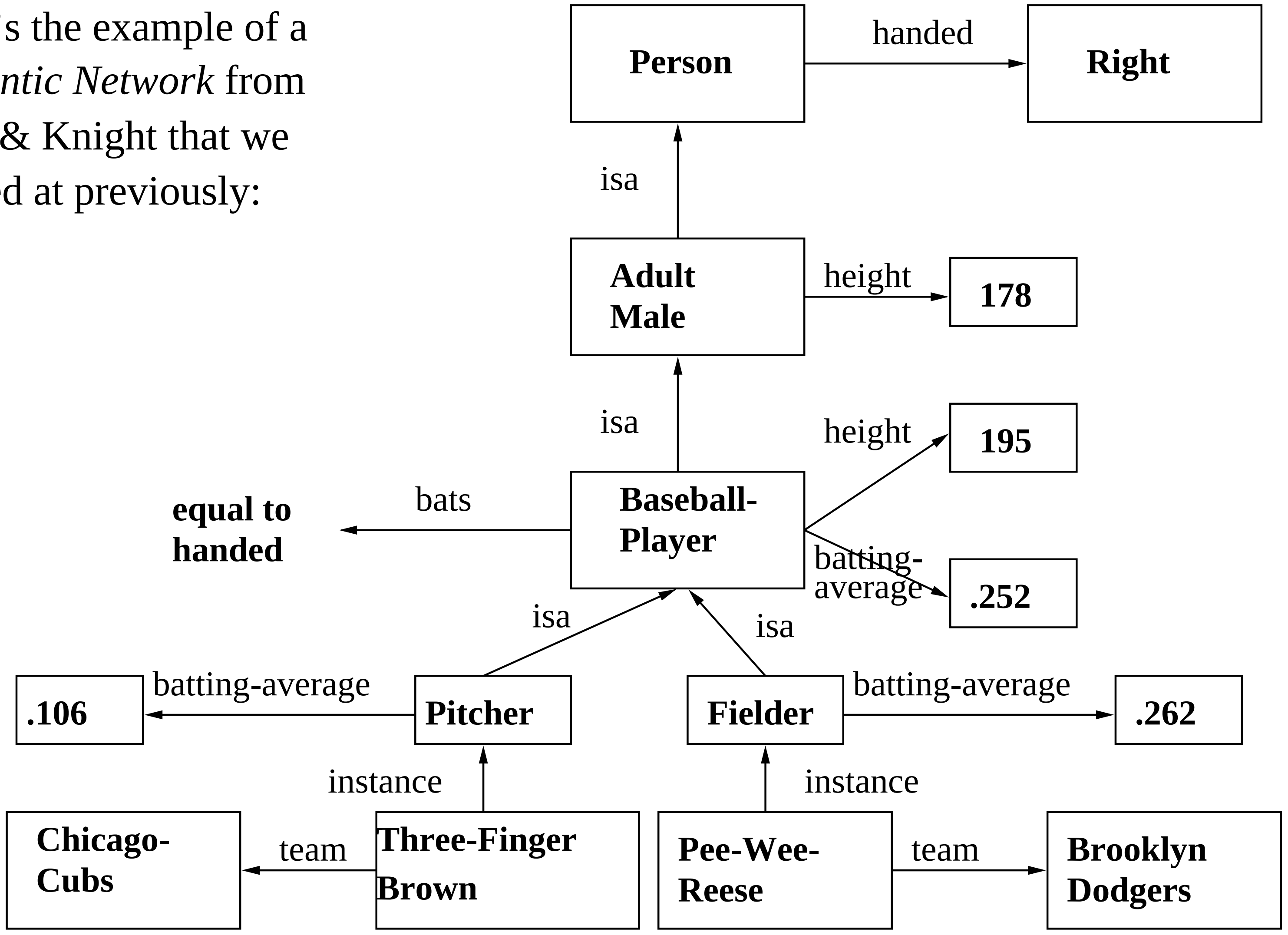
3. The ***semantic*** part – that establishes a way of associating ***real world meanings*** with the representations.

4. The ***procedural*** part – that specifies the access procedures that enables a way of ***creating*** and ***modifying*** representations and ***answering questions*** using them, i.e. how we generate and compute things with the representation.

We saw these in the brief overviews of different representations last lecture.

A Simple Semantic Network

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



Components of a Semantic Network

The fundamental components of semantic networks are straightforward to identify:

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.

What does “Semantics” Really Mean?

For our purposes, ***“semantics”*** just means ***meanings***. But for many people (particularly philosophers), it is important to be more precise. The three main practical approaches to semantics for semantic networks, and AI in general, are:

Descriptive Semantics – The formulation of explanations of what the labels in our representations mean in terms of things we understand intuitively.

Procedural Semantics – The formulation of a set of procedures/programs that operate on the labels in the representation, and then the idea that the meanings are defined by what the programs do.

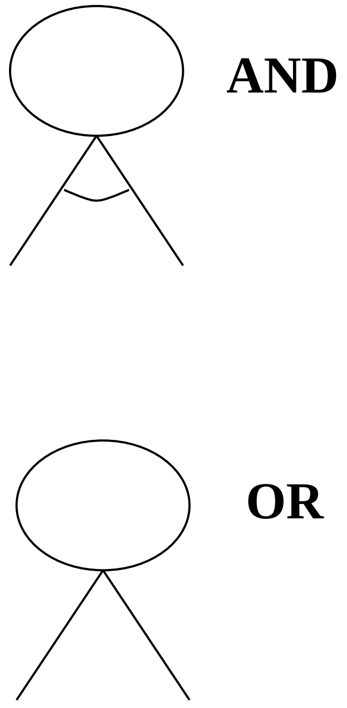
Equivalence Semantics – The relation of the meanings in the representation to those in some other representation that already has an accepted semantics.

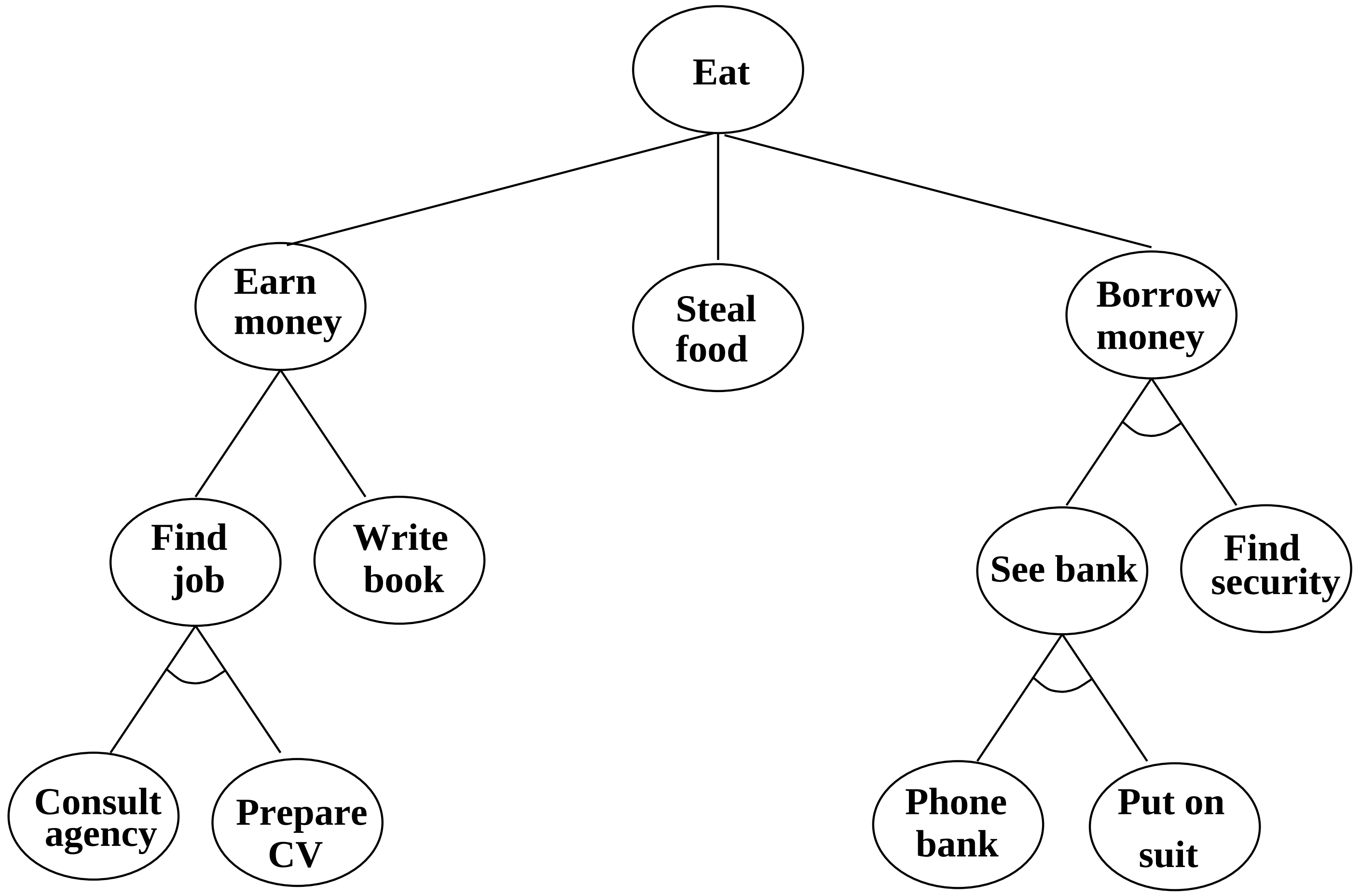
The extent to which one needs to worry about these distinctions depends on what you are trying to do. Often, simple intuitive ideas about meanings are sufficient.

AND / OR **Trees**

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

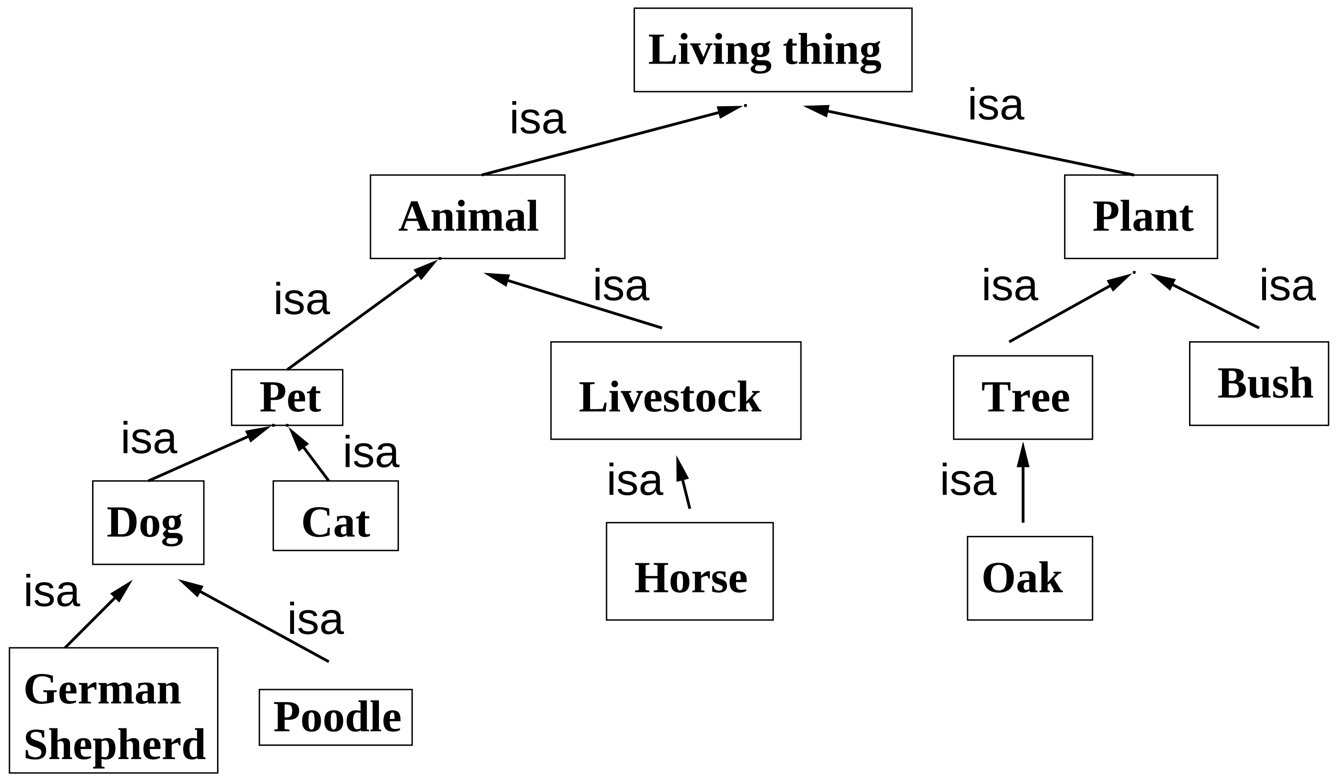
Two node types:





An *IS-A* Hierarchy

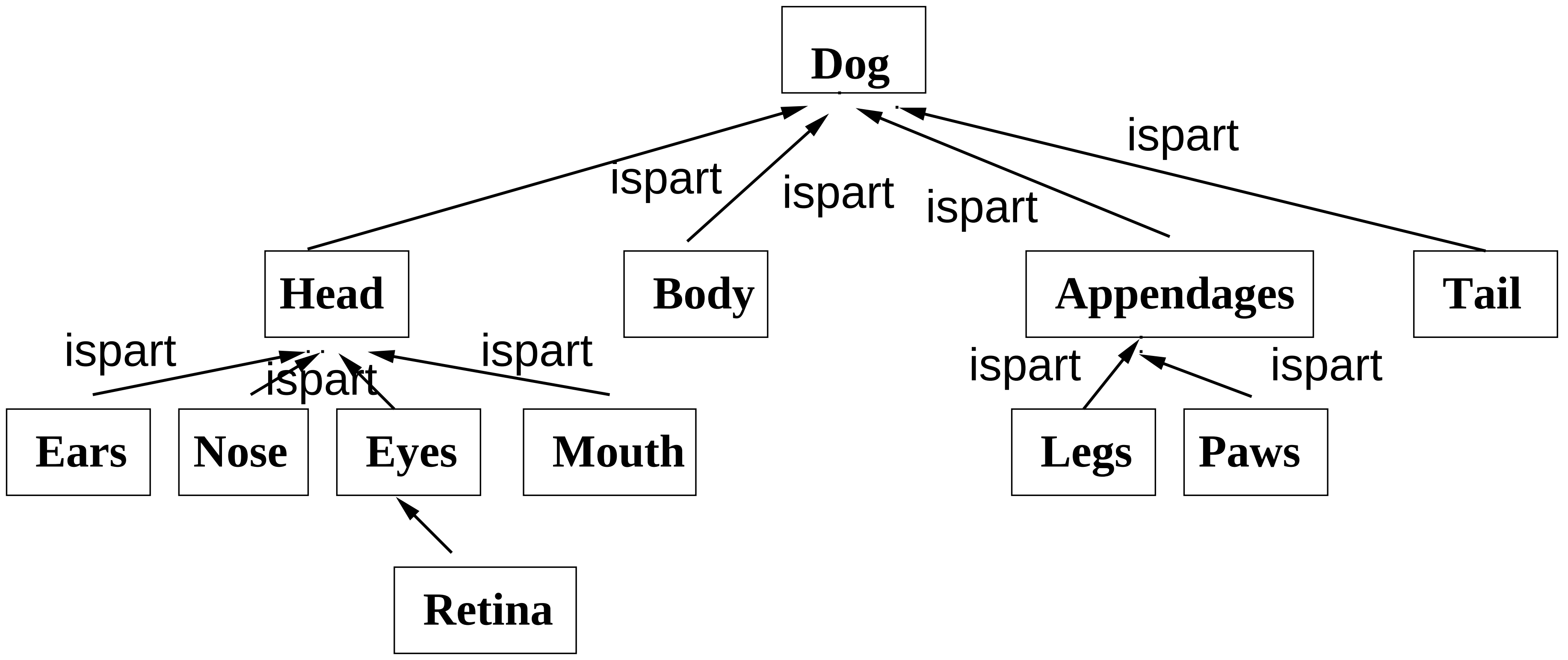
Another simple form of semantic network is an ***is-a hierarchy***. For example:



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:

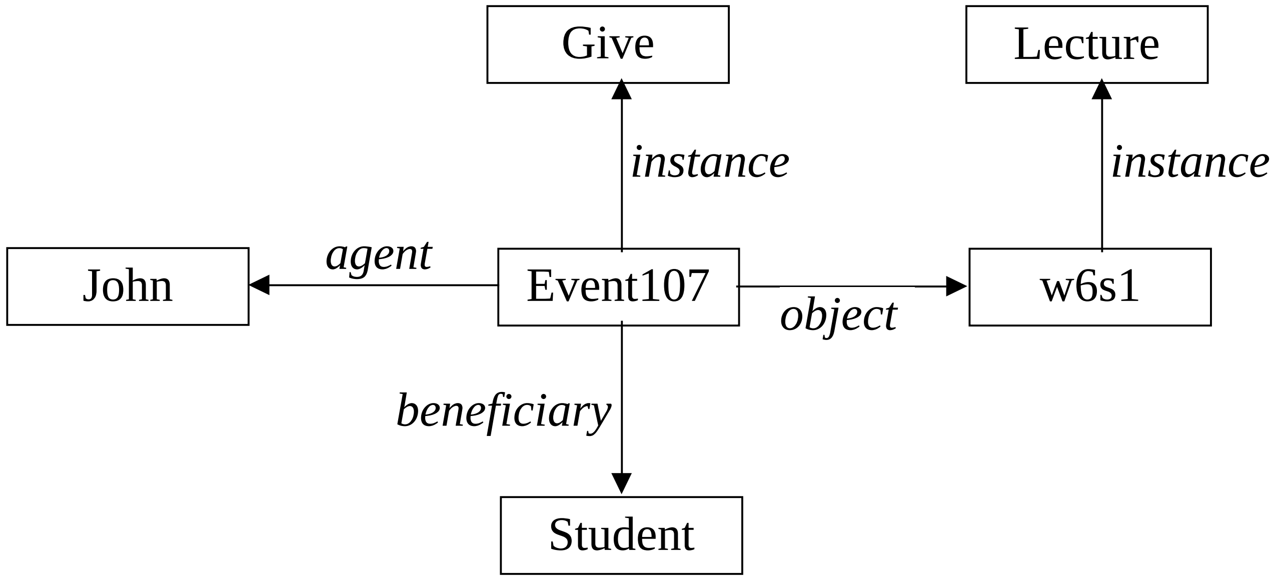


Naturally, where we choose to stop the hierarchy depends on what we want to represent.

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 w6s1 to his students"

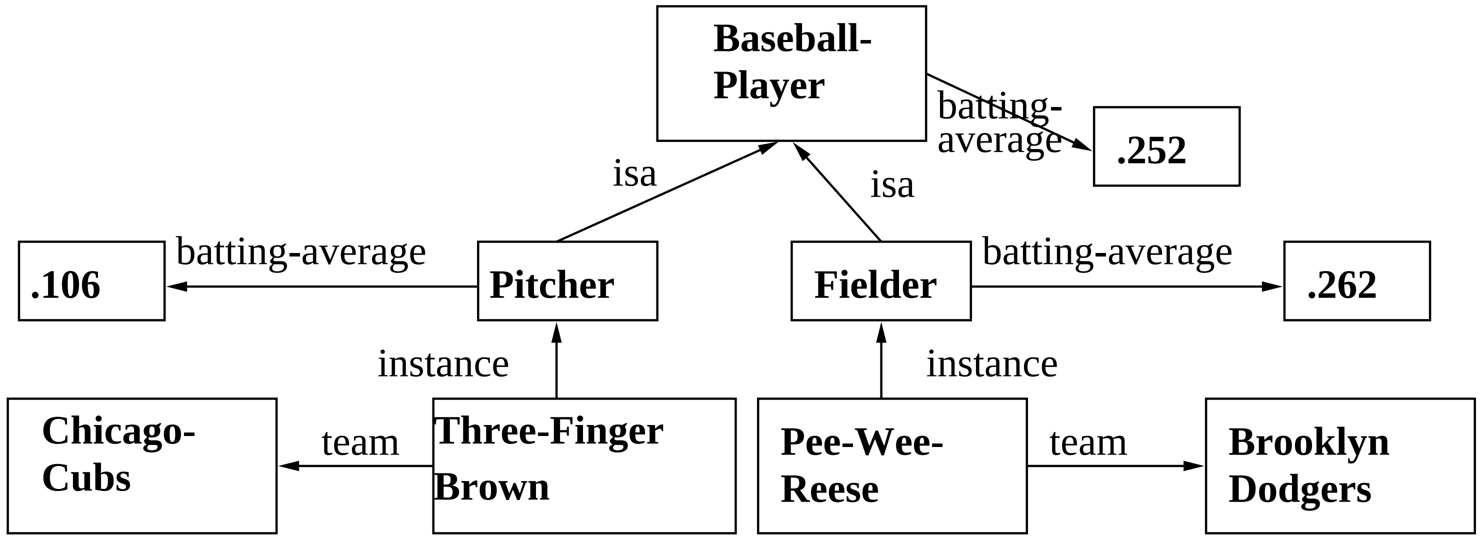


In fact, several of the earliest semantic networks were English-understanding programs.

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***.

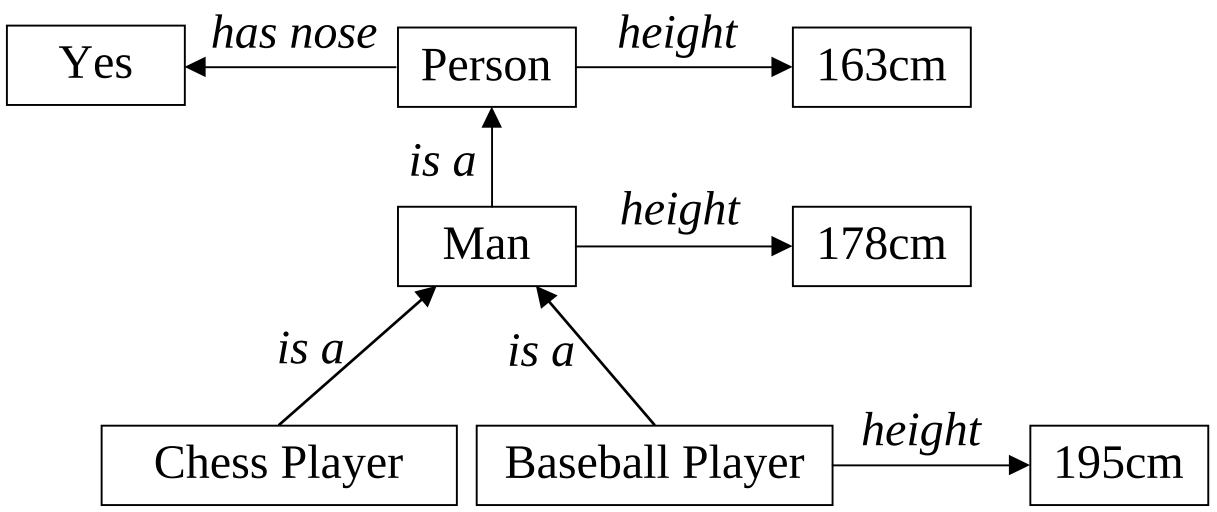
Question: "What is the relation between Chicago cubs and Brooklyn Dodgers?"



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:

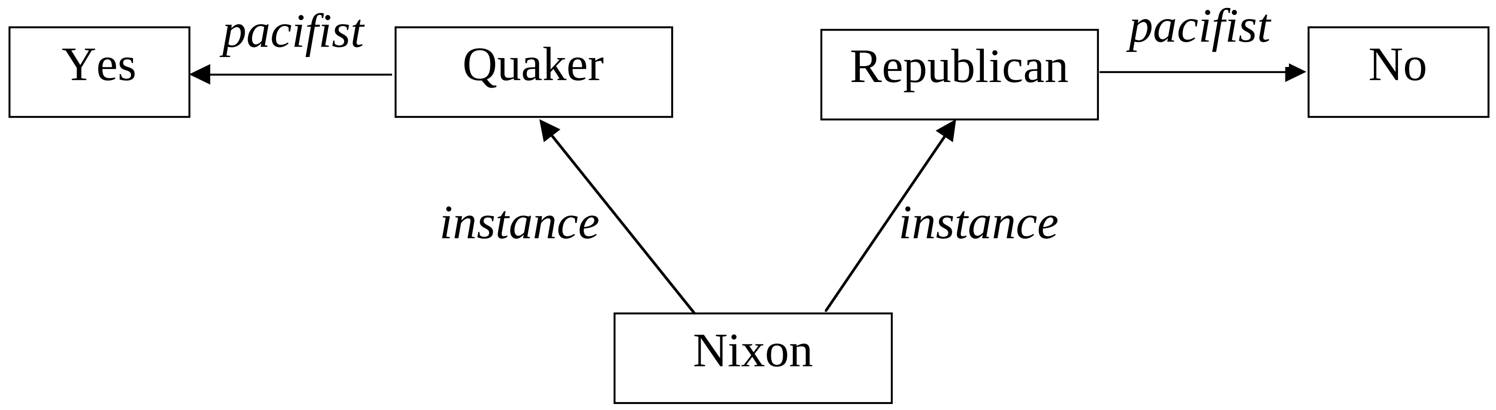


We can assign expected/default values of parameters (e.g. height, has nose) and inherit them from higher up the hierarchy. This is more efficient than listing all the details at each level. We can also ***over-ride*** the defaults. For example, baseball players are taller than average, so their default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

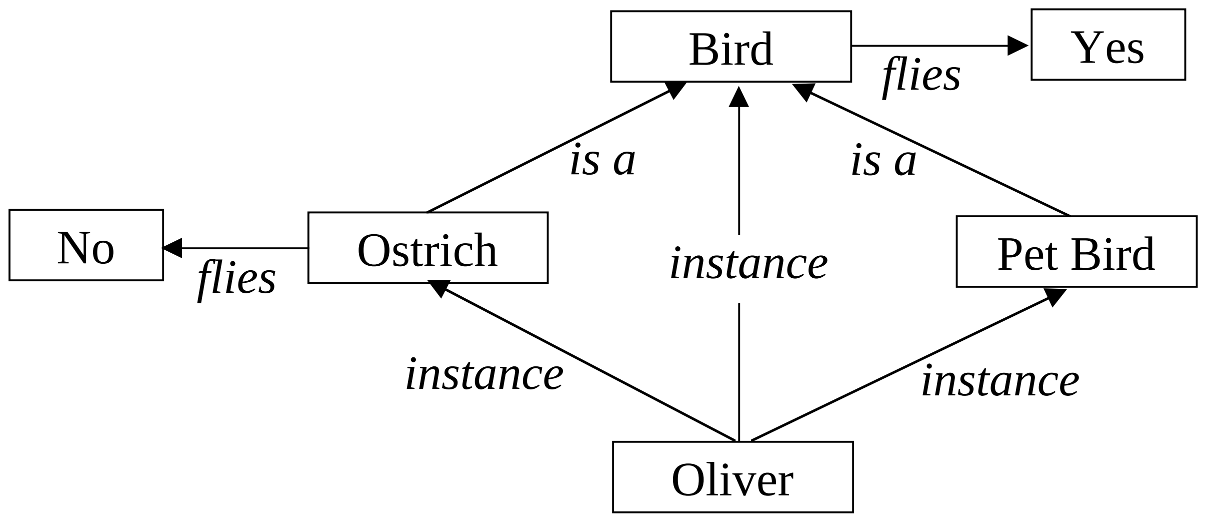


Conflicts like this are common is the real world. It is important that the inheritance algorithm reports the conflict, rather than just traversing the tree and reporting the first answer it finds. In practice, we aim to build semantic networks in which all such conflicts are either over-ridden, or resolved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

Overview and Reading

1. We began by looking at the components of good representations in general, and then at those of semantic networks.
2. We then looked at various common styles of semantic networks: *AND/OR* trees, *IS-A* and *IS-PART* hierarchies, and event/language networks.
3. The important procedural concepts of *Intersection Search*, *Inheritance*, and *Defaults* were then covered.
4. We ended by considering how some common inferential conflicts could be resolved.

Reading

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