AITA : Frame Based Systems

The Evolution of Semantic Networks into Frames

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

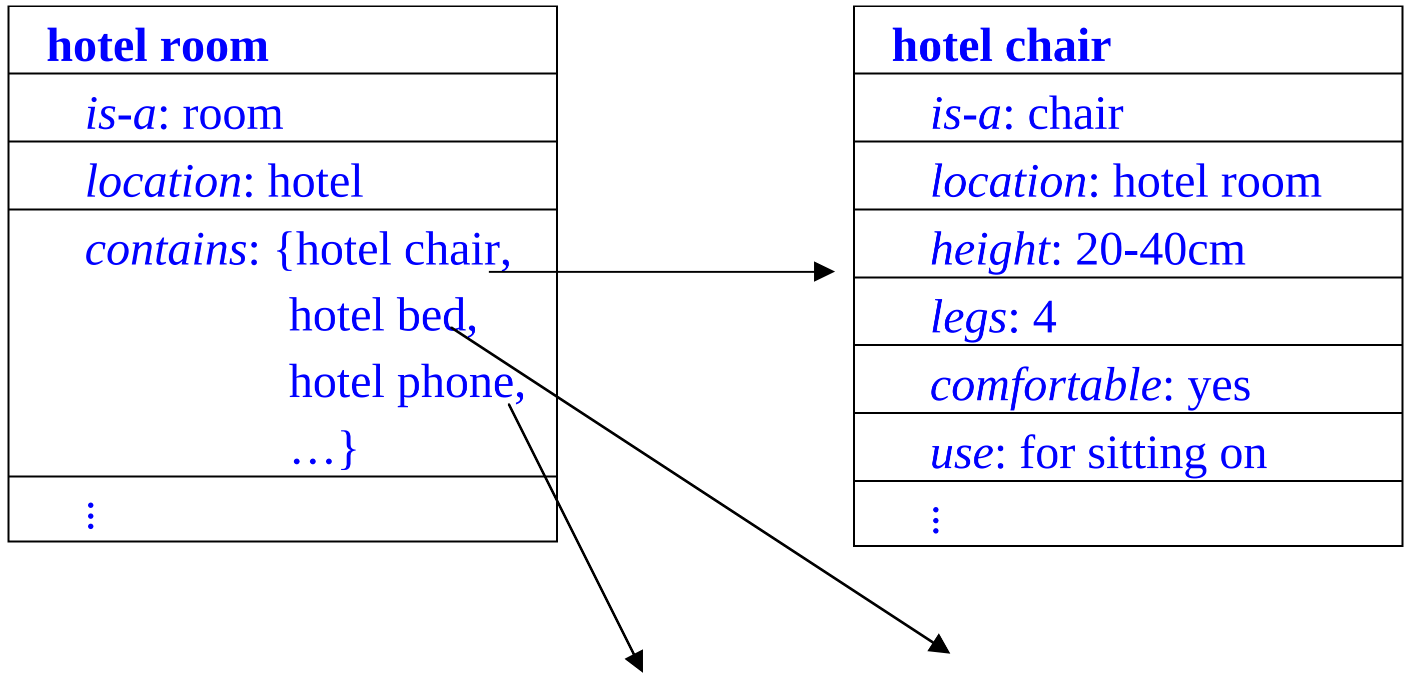
In particular, it became necessary to assign more structure to nodes, as well as to links. For example, in many cases we need node labels that can be computed, rather than being fixed in advance. It was natural to use database ideas to keep track of everything, and the nodes and their relations began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

In the literature, the distinction between frames and semantic networks is 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 of frames connected together by a network of 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***.

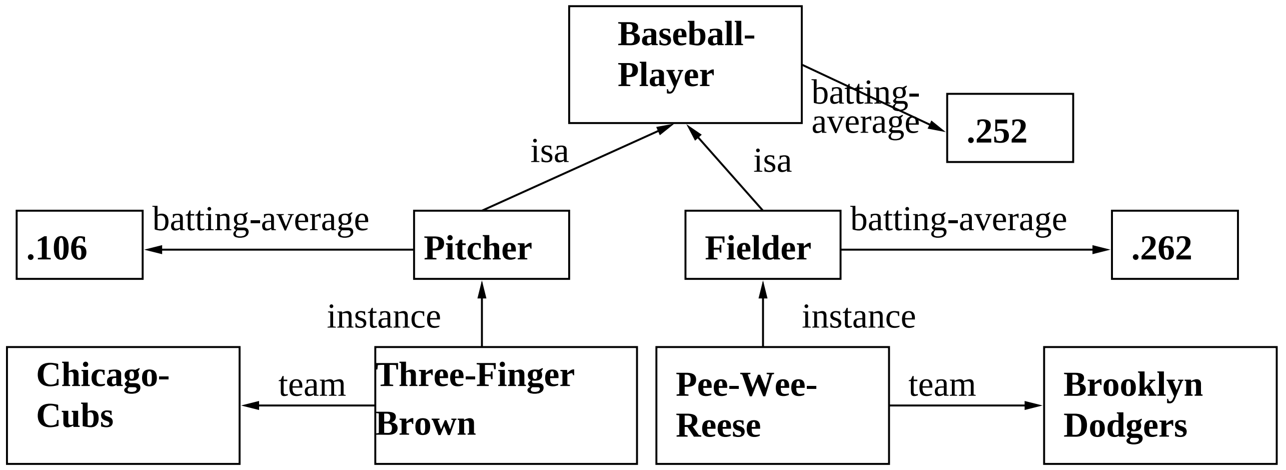
Frames become much more powerful when their slots can also contain instructions (procedures) for computing things from information in other slots or in other frames.

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 Semantic Networks to 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:
⋮

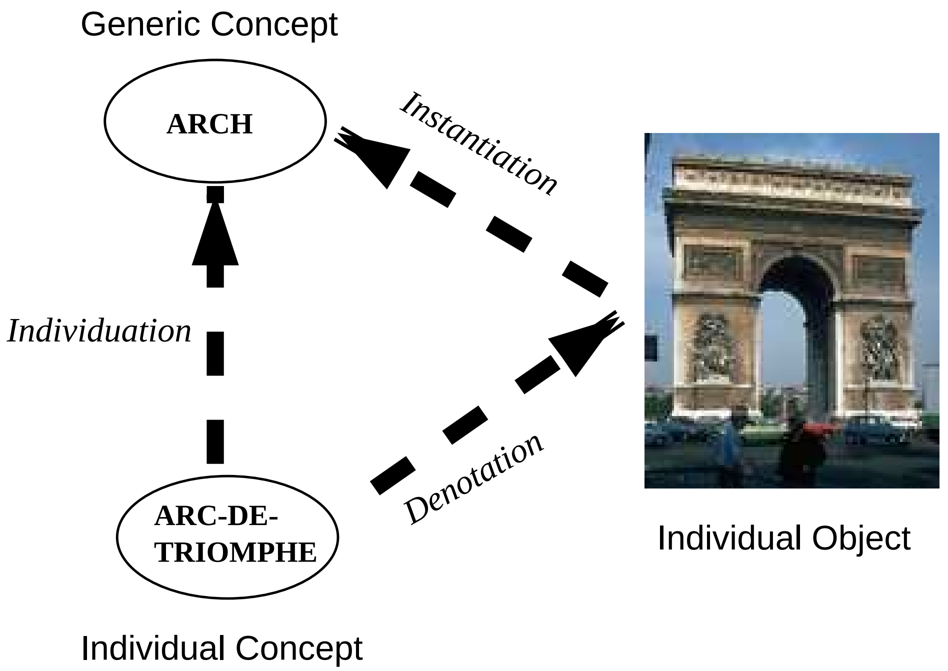
Fielder
is-a: Baseball player
batting average: .262

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.

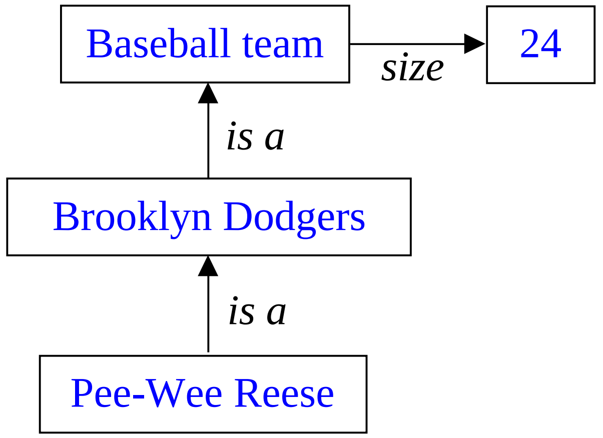


Clearly *is-a* corresponds to subset $\subseteq$, and *instance* corresponds to element $\in$. Then set theory concepts such as ***transitivity***, ***intersection***, etc. apply automatically to our frames.

Problems with Sets That Are Also Instances

Consider the frame that we might create for the Brooklyn Dodgers baseball team:

Brooklyn Dodgers
instance: baseball team
team size: 24
manager: Leo-Durocher
players: {Pee-Wee-Reese, …}
⋮



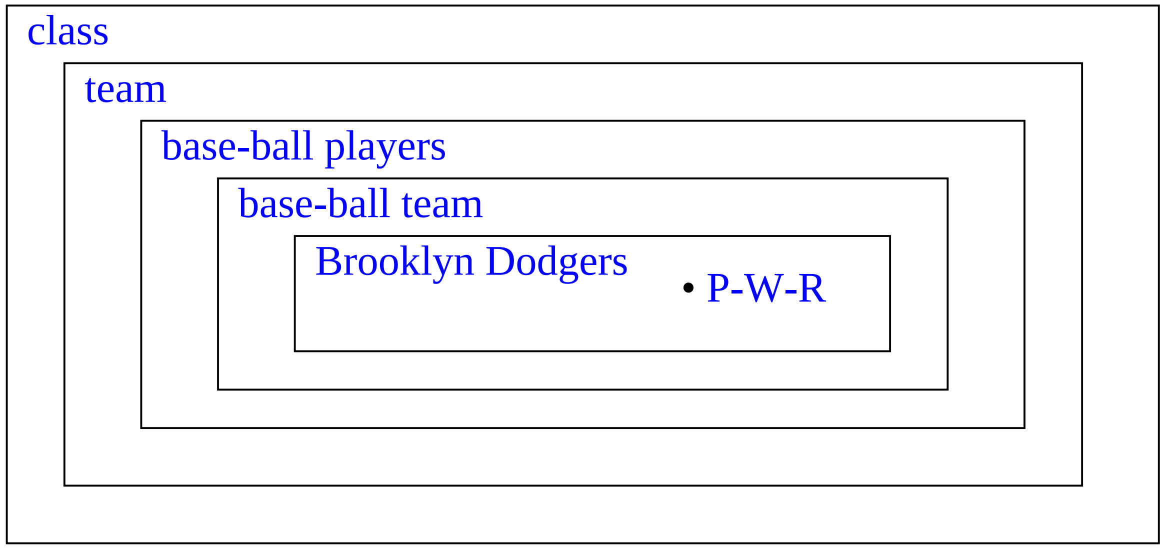
This is obviously an ***instance*** of a baseball team, but it is also a ***set/class*** of players of which Pee-Wee-Reese is an instance.

We need to be very careful about how we set up the hierarchy because we clearly don't want Pee-Wee-Reese to inherit the general properties of baseball teams (e.g. size 24), but we do want Brooklyn Dodgers to inherit the properties that baseball teams have.

Meta-Classes

Our Brooklyn Dodgers problem is quite general. From a set of instances we want to use inheritance to infer the instance properties from the general knowledge (default properties) of the set. But the class is also an entity in its own right, and may possess properties that belong to the class as a whole rather than to the individual instances.

It is useful to make a distinction between ***regular classes*** whose instances are individual entities, and ***meta-classes*** which are classes whose instances are other classes, e.g.



Representing Meta-Classes in Frames

We can see that each frame that corresponds to a set/class needs to contain attributes about the set itself, as well as attributes to be inherited by each element of the set. We distinguish them by prefixing the latter with an asterisk (*). For our previous example:

Team
instance: Class
isa: Class
cardinality: 4563
**team-size*:

Baseball Team
instance: Class
isa: Team
cardinality: 274
**team-size*: 24
**manager*:

Brooklyn Dodgers
instance: Baseball Team
isa: Baseball Player
manager: Leo-Durocher
**uniform-colour*: blue

Pee-Wee-Reese
instance: Brooklyn Dodgers
instance: Fielder
handed: left

So Brooklyn Dodgers inherits team size 24, but Pee-Wee Reese doesn't.

Slots as Full-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 type 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.

Scripts

Scripts are Minsky's original idea of a frame based structure that describes stereotyped sequences of events in a particular context.

The slots in such frames will contain several different kinds of information, some of which may be rather complex. Typically they will contain:

1. Information about how to use the frame.
2. A specification for the language/notation used in the frame.
3. Details about the 'props' and 'roles' that may be encountered.
4. Instructions about what one can expect to happen next, or what one should do next.
5. Indications about what to do if our expectations are not confirmed.
6. Any other information/instructions that might be appropriate.

The idea is that, when someone or something (e.g. our AI agent) encounters a new situation, they can select from memory an appropriate frame, and this can be adapted to fit reality by filling in the details as necessary.

Primitive Acts

In formulating scripts it is sensible to build them within a framework of ***agents*** manipulating ***props*** using a well defined set of ***primitive acts***, such as:

MOVE	Movement of a body part by its owner (e.g. kick)
PROPEL	Application of physical force to an object (e.g. push)
GRASP	Grasping of an object by an agent (e.g. clutch)
INGEST	Ingestion of an object by an animal (e.g. eat)
ATTEND	Focussing of a sensor towards a stimulus (e.g. listen)
MTRANS	Transfer of mental information (e.g. tell)
SPEAK	Production of sounds (e.g. say)
PTRANS	Transfer of the physical location of an object (e.g. go)
ATRANS	Transfer of an abstract relationship (e.g. give)
MBUILD	Building of new information out of old (e.g. decide)

From these we can construct arbitrarily complex scripts.

Components of a Script

Looking at some typical scripts we can identify six important components:

Entry conditions	Conditions that must be satisfied before the events described in the script can occur.
Roles	Agents involved in the events described in the script (which may be explicitly or implicitly declared).
Props	Objects involved in the events described in the script (which may be explicitly or implicitly declared).
Scenes	All the actual sequences of events that are represented in the script.
Track	The specific route through the possible sequences of events that arises when the script is processed.
Results	Conditions that will be true after the events described in the script have occurred.

It is clear that scripts provide an extremely powerful representational structure.

Overview and Reading

1. We started by looking at the evolution of semantic networks into frames, and how we can convert from one to the other.
2. We then considered the basic components of frame based representations, and noted how set theory provides a convenient basis for their processing.
3. We saw how problems can arise when sets are also instances, and how these problems can be avoided with the concept of meta-classes.
4. We considered how slots can be promoted to fully-fledged objects with extremely general filler properties.
5. We ended with a discussion of frame based scripts.

Reading

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