Representations

There are many issues to consider when choosing a representation. For example, how to depict a definition. What about time? Causality? Uncertainty? (For example, logic does not retain causality.) Further, one must worry about metaknowledge; or common-sense knowledge. And the distinction between the abstract properties and the set of all objects. In this chapter we discuss some common/early representations.

2.1 Classification and Formalization

There are at least four categories of knowledge representation schemes, though there is some overlap.

1. *Logical representation.* e.g. first-order logic as embodied in Prolog.
2. *Procedural representation.* e.g. rule base of an expert system, or production system (discussed below)
3. *Network representation.* i.e. semantic networks, decision trees, neural networks
4. *Structured representation.* e.g. slotted representations such as scripts and frames (discussed below)

According to Finlay and Dix, we should judge a representation scheme by four properties:

- *Expressiveness.* Can it clearly and completely represent the necessary data?
- *Effectiveness.* Can one use the data for computation and inference?
- *Efficiency.* Can one gather and harness the data easily?
- *Explicitness.* Does it provide for clear explicit explanations?

More formally, a representation has four parts:

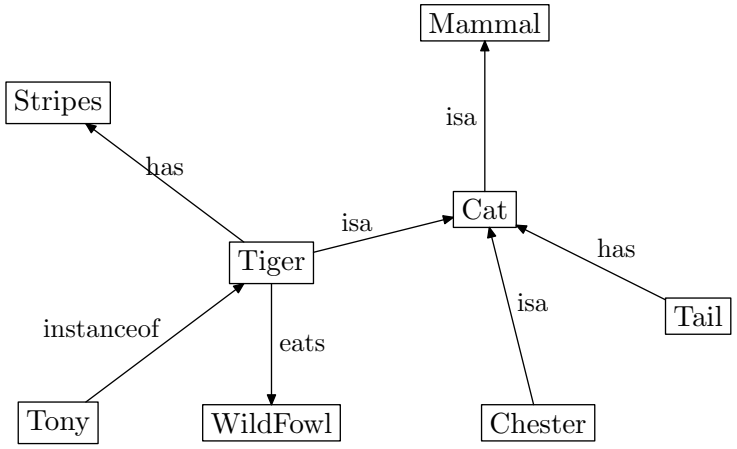
- the ***lexical*** part describes vocabulary of allowable symbols
- the ***structural*** part describes constraints on the combination of symbols (syntax)
- the ***procedural*** part describes access procedures such as constructors, readers, writers or erasers
- the ***semantic*** part associates a meaning with the description

One thing to note with logic is that the inference is completely syntactic—it ignores the meaning of the symbols and simply manipulates them. Another representation scheme is objects. The ***frame problem*** is that when representing something that changes over time that almost all the data stays the same. So should one copy all data or only keep track of that data that has changed?

2.2 Semantic Nets

Semantic nets arose from models of human information storage and management. We have already seen search trees, game trees and state space graphs. There are several more.

A typical semantic network can be used to store relational information. Common binary relations are **isa**, **instance-of**, and **partof**. Semantic networks are especially good at representing property inheritance.

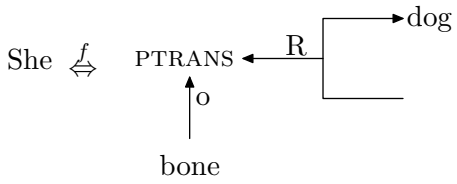


The four parts of a semantic net are:

lexical: nodes, links, labels;
structural: each link connects tail node to head node;
semantic: nodes & links denote application-specific entities;
procedural: can construct a node, construct a link, list from node its out-links, list from link its tail node, list from link its label, etc.

EXAMPLE 2.1. *Schank developed* ***conceptual dependency*** *representations to represent general knowledge. This attempts to break all knowledge down to a small set of primitives. For example, move, swipe, transfer and throw are all examples of transfer of position. They can all be represented by the abstract action* PTRANS. *The advantage is that built into* PTRANS *is the concept of time, of there being an old and a new position, etc: we do not have to re-specify all this each time.*

This represents actions, objects, modifiers of actions, and modifiers of objects. So for example, the following encodes the information: "She will fetch the dog a bone."



Another class of network representations are ***value propagation nets***. Neural networks fall into this category.

2.3 Production Scheme

The production scheme was originally developed by Newell & Simon for control. In this there is:

1. set of production rules
2. working memory
3. the recognize-act control cycle: match, select, execute

There is also a mechanism for conflict resolution. This can be in the rule structure, or explicitly added. A rule base for an expert system can be thought of as a special case. A production scheme is also related to a general grammar (remember context-free grammars?). Production schemes provide modularity and can be grown.

EXAMPLE 2.2. *The following is an example production scheme for sorting a string of a's, b's and c's into alphabetical order:*

$ba \rightarrow ab$
$ca \rightarrow ac$
$cb \rightarrow bc$

When a rule is fired, the substring on the left is replaced by the string on the right.

EXAMPLE 2.3. *(From Rich and Knight.) The Water and Jug problem can also be pursued via production rules. In this problem, one has two jugs, a 4-liter and a 3-liter jug, each with no markings and wishes to obtain exactly 2 liters. If (x, y) means that there are x liters in the 4-liter jug and y liters in the 3-liter jug, then the possible moves are:*

$(x, y) \to (4, y)$	*fill the big jug*
$(x, y) \to (x, 3)$	*fill the small jug*
$(x, y) \to (0, y)$	*empty big jug on ground*
$(x, y) \to (x, 0)$	*empty small jug on ground*
$(x, y) \to (4, 4 - x + y)$	*fill big jug from small jug*
$(x, y) \to (3 - y + x, 3)$	*fill small jug from big jug*
$(x, y) \to (0, x + y)$	*empty big jug into small jug*
$(x, y) \to (x + y, 0)$	*empty small jug into big jug*

Of course, one must add limitations on x and y in some rules.

2.4 Structured Representations: Frames and Scripts

"In structured representations information is organized into more complex knowledge structures. *Slots* in the structure represent attributes into which values can be placed. These values are either specific to a particular instance or default values, which represent stereotypical information. Structured representations can capture complex situations or objects, for example eating a meal in a restaurant or the context of a hotel room. Such structures can be linked together as networks, giving property inheritance. *Frames* and *scripts* are the most common types of structured representations." (Finlay & Dix)

"Here is the essence of the frame theory: When one encounters a new situation (or makes a substantial change in one's view of a problem) one selects from memory a structure called a "frame." This is a remembered framework to be adapted to fit reality by changing details as necessary (Minsky 1975)".

In some respects, ***frames*** are just semantic nets: nodes become frames and links become slots. The following picture gives the frame structure corresponding to the earlier semantic net. However, there are two main extensions: (a) emphasis on default knowledge; and (b) emphasis on procedural knowledge that can be placed in the slots.

Mammal	
Cat	
isa:	mammal
has:	tail

Tiger	
isa:	cat
eats:	wildfowl
name:	none

Tony	
instanceof:	Tiger
name:	Tony
father:	Kellogs

Slots can have procedures: examples are **if-needed** or **if-added** (sometimes called daemon procedures). The default knowledge is linked to inheritance of default properties from membership in superclass (object-oriented).

A ***script*** is designed for an expectation of stereotypical sequence of events. It consists of

- entry conditions
- results
- props: slots for objects
- roles: slots for agents
- scenes: sequence of events

EXAMPLE 2.4.

Fast food encounter

roles:	*C=customer*
	S=server
	F=food
Scene 1:	*C enters diner*
	C joins line
Scene 2:	*C chooses F*
	If no suitable then C exits
Scene 3:	*S takes order for F*
	S takes money for F
	S fetches F
Scene 4:	*C eats food*
	C exits

How to use a script? First activate by finding the best-fit. For deductions, go backwards or forwards assuming the sequence of events is a causal chain (though an antecedent is not necessarily a cause). For example, if we know that the server provides the food, then the customer must have provided the money.