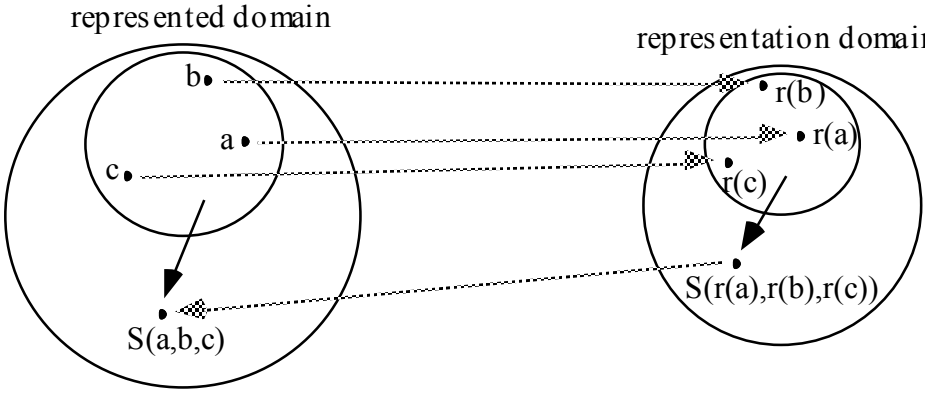
Chapter : Structured Knowledge Representation

What is a representation?

A representation consists of two sets and a mapping between them. The elements of each set are objects, relations, classes, laws, actions. The first set is called the represented domain, and the second one is called the representation domain.



This mapping allows one to reason about the represented domain by performing reasoning processes in the representation domain and transferring the conclusions back into the represented domain (by using reverse mapping between the two sets).

The processes that manipulate the objects and the relationships in the representation domain are also part of the representation.

Knowledge representation is concerned with encoding the human knowledge into the form that is efficiently manipulated by the computer. Following are the kinds of knowledge that is represented in AI systems.

- ⇒ **Objects**: Guitar has Strings
- ⇒ **Events**: Ram played Guitar
- ⇒ **Performance**: Dirt cleaned by vacuum cleaner
- ⇒ **Meta-Knowledge**: Renuka knows that she can read street signs along the way to find where it is
- ⇒ **Facts**: Truths about real worldand what we represent

General features of a representation

- ⇒ **Representational adequacy**: The ability to represent all of the kinds of knowledge that are needed in a certain domain.

- ⇒ **Inferential adequacy***:* The ability to represent all of the kinds of inferential procedures (procedures that manipulate the representational structures in such a way as to derive new structures corresponding to new knowledge inferred from old).

⇒ **Inferential efficiency***:* The ability to represent efficient inference procedures (for instance, by incorporating into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions).

⇒ **Acquisitional efficiency***:* The ability to acquire new information easily.

Knowledge Representation schemes can be classified as below:

⇒ Logical representation Scheme
⇒ Procedural representation Scheme
⇒ Network representation Scheme
⇒ Structured representation Scheme

Logical representation Scheme

It uses expression in the form of formal notation to represent the knowledge base. Propositional logic and predicate logic are examples of the logical representation Scheme. Give example your self.

Procedural representation Scheme

These representation schemes constitute rule based expert systems.

If type of light is sunlight then light is bright
If type of light is bulb then light is dim
If location is outdoor then light is sunlight
If location is indoor then light is bulb

Network representation Scheme

Network representation schemes represents knowledge in the form of a graph in which the nodes represent objects, situations, or events, and the arcs represent the relationships between them. Semantic networks, conceptual dependencies and conceptual graphs are examples of network representation.(Give example of semantic net yourself)

Structured representation Scheme

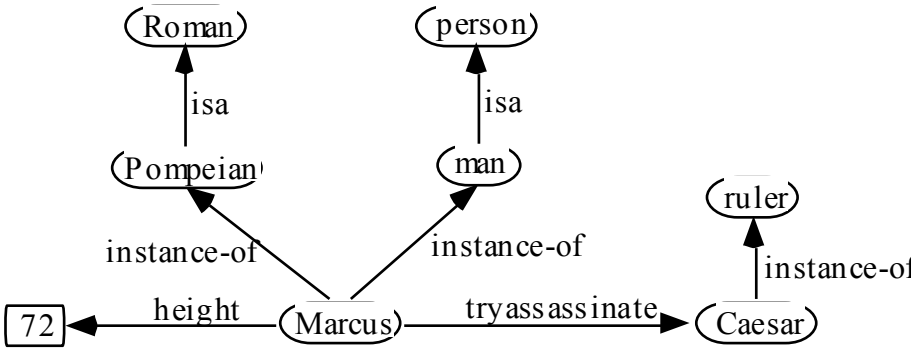
This is the extension of network representation schemes where nodes are complex data structures consisting of named slots with attached values. Scripts, frames and objects are examples of structured representation scheme.

Representing knowledge in semantic networks

The underlying idea of semantic networks is to represent knowledge in the form of a graph in which the nodes represent objects, situations, or events, and the arcs represent the relationships between them.

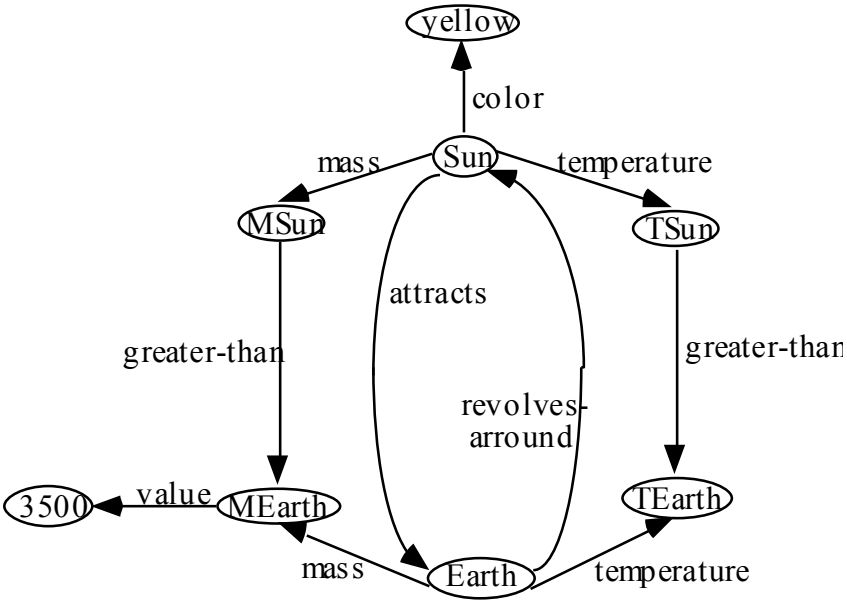
Semantic networks with binary relations

The following is an example of a semantic network with binary relations:



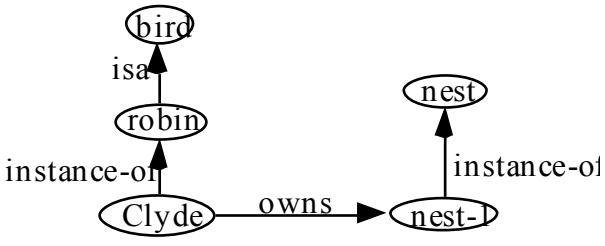
Some nodes represent individual objects (Marcus, Caesar) or constants (72).Other nodes represent classes of individuals (Pompeian, Roman, man, person, ruler). The relationship "instance-of" asserts that an individual is an instance (element) of a class. The relationship "isa" asserts that a class, (e.g. "man"), is a subclass of another class ("person"). "Man isa person" means that any element of man is also an element of person. That is, if someone is a man then he is a person. The relationship "tryassassinate", from Marcus to Caesar, states that Marcus tried to assassinate Caesar. The relationship "height", from Marcus to 72, states that the height of Marcus is 72. Marcus and Caesar are called instances because they represent individual objects.

The following is another example of a semantic network representing knowledge about our solar system. One may notice that this network states that the temperature of the Sun is greater than the temperature of the Earth, without indicating the values of these temperatures:



Representing non-binary predicates

Suppose we wish to represent the fact that Clyde, which is a robin, owns a nest. This may be encoded as follows:



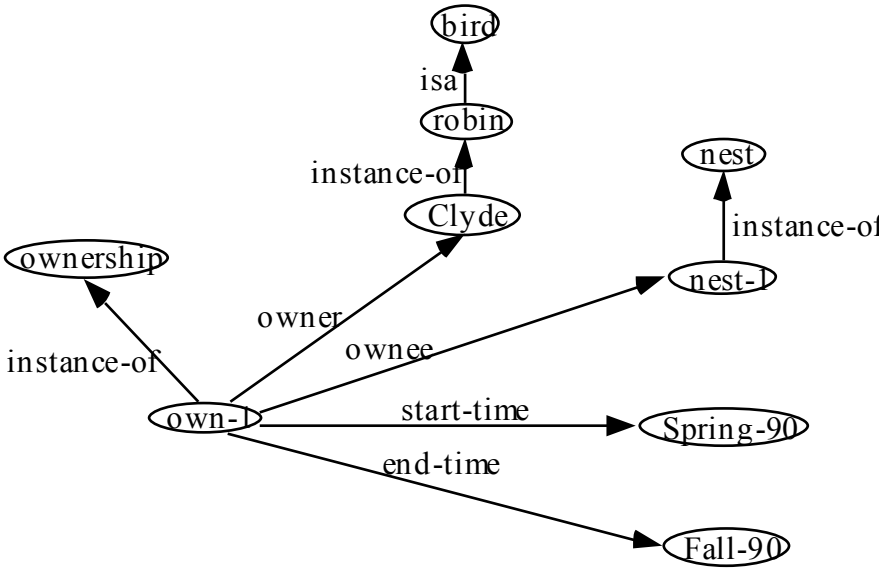
.

"nest-1" is the nest Clyde owns. It is an instance of "nest" which represents the class of all nests. Suppose one wants to encode the additional information that Clyde owned nest-1 from Spring 90 to Fall 90. This is impossible to do in the above network because the ownership situation is encoded as a link, and links, by their nature, can encode only binary relations.

The solution is to represent this ownership situation as a node characterized by four properties: the owner (Clyde), the object owned (nest-1), the time when the ownership started (Spring 90), and the time when the ownership ended (Fall 90). In this way one may represent the equivalent of a four-place predicate:

ownership(owner, ownee, start-time, end-time)

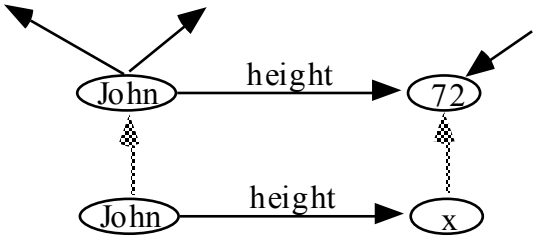
The corresponding semantic network is the following one:



Therefore, the semantic networks are not limited to representing only binary relationships

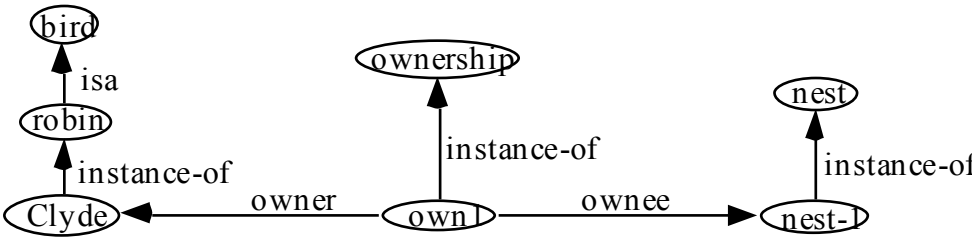
Network matching

A network fragment is constructed, representing a sought-for object or a query, and then matched against the network database to see if such an object exists. Variable nodes in the fragment are bound in the matching process to the values with which they match perfectly.

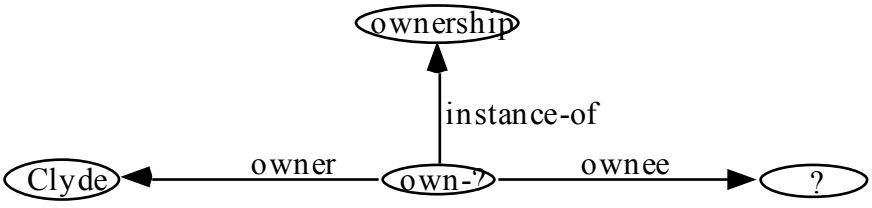


Question:
What is the height of John
Answer:
The height of John is 72.

As another example, let us suppose that we use the network given below as a database and suppose we wish to answer the question "What does Clyde own?".



We might construct the network in above figure which represents an instance of ownership in which Clyde is the owner. This fragment is then matched against the network database looking for an own node that has an owner link to Clyde. When it is found, the node that the ownee link points to is bound in the partial match and is the answer to the question.

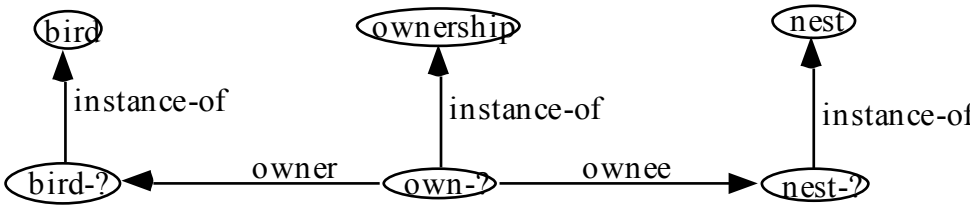


The matcher can make inferences during the matching process to create a network structure that is not explicitly present in the network.

For example, suppose we wish to answer the question

"Is there a bird who owns a nest?"

We could translate that question into the following network fragment:



Here, bird-?, nest-?, and own-? nodes represent the yet to be determined bird-owning-nest relation.

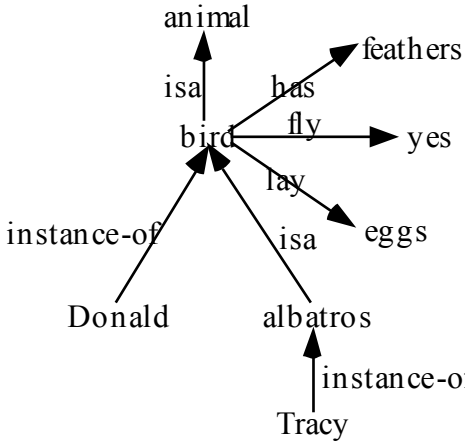
Notice that the query network fragment does not match the knowledge base exactly. The deduction procedure would have to construct an "instance-of" link from Clyde to bird to make the match possible. The matcher would bind bird-? to the node Clyde, own-? to own-1, and nest-? to nest-1, and the answer to the question would be "Yes, Clyde".

Exercises

1. Represent the following sentences into a semantic network.

Birds are animals.
Birds have feathers, fly and lay eggs.
Albatros is a bird.
Donald is a bird.
Tracy is an albatros.

Solution:



2. Represent the following sentences into a semantic network:

Puss is a calico.

Herb is a tuna.

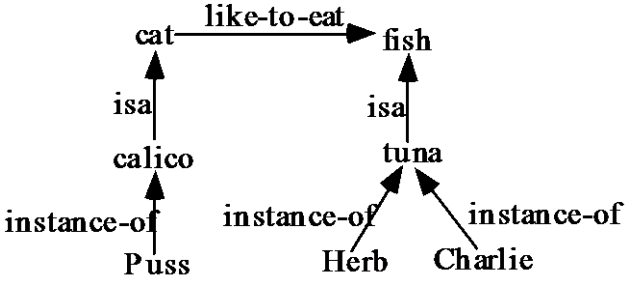
Charlie is a tuna.

All tunas are fishes.

All calicos are cats.

All cats like to eat all kinds of fishes.

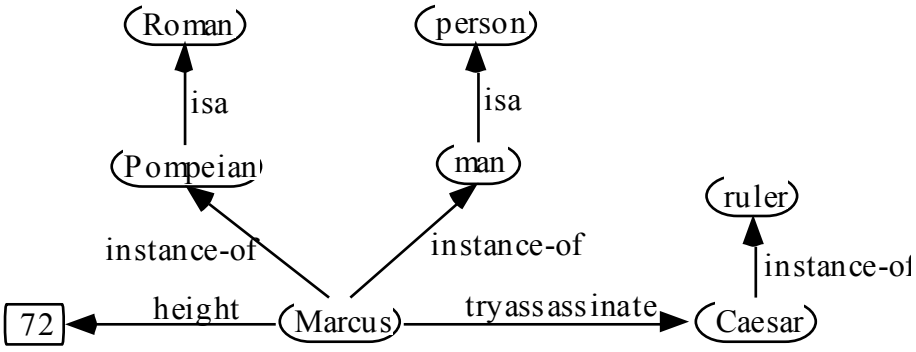
Solution:



3. Represent the following sentence with a semantic network:

John lived in Fairfax from May 1985 till February 1990

4. Consider the following network fragment:



Explain how a semantic network system would answer the questions:

What is the height of Marcus?

Is there a person who tried to assassinate Caesar?

Object-based Approach: Frames

With this approach, knowledge may be represented in a data structure called a **frame**. A *frame* is a data structure containing typical knowledge about a concept or object (Marvin Minsky (mid 1970s)). A frame represents knowledge about real world things (or entities).

Each frame has **a name and slots**. **Slots** are the properties of the entity that has the name, and they have **values** or pointer to other frames (a table like data structure). A particular value may be:

- a default value
- an inherited value from a higher frame
- a procedure, called a daemon, to find a value
- a specific value, which might represent an exception.

When the slots of a frame are all filled, the frame is said to be **instantiated**, it now represents a specific entity of the type defined by the unfilled frame. Empty frames are sometimes called object prototypes

The idea of frame hierarchies is very similar to the idea of class hierarchies found in object-orientated programming. Frames are an application of the object-oriented approach to knowledge-based systems.

Disadvantages

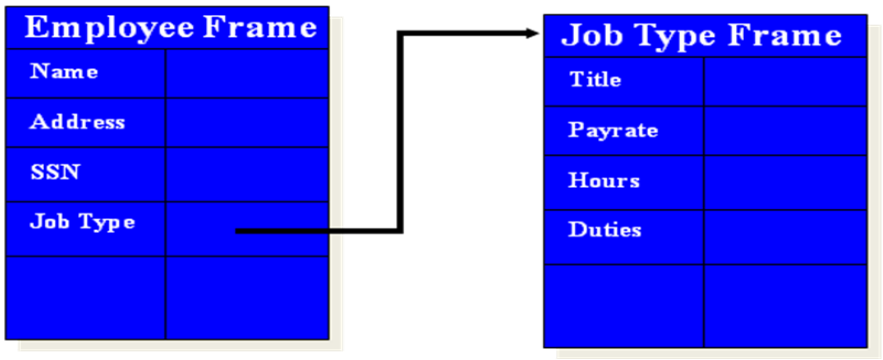
- complex
- reasoning (inferencing) is difficult
- explanation is difficult, expressive limitation

Advantages

- knowledge domain can be naturally structured [a similar motivation as for the O-O approach].
- easy to include the idea of default values, detect missing values, include specialised procedures and to add further slots to the frames

Examples:

(1.)



(2.)

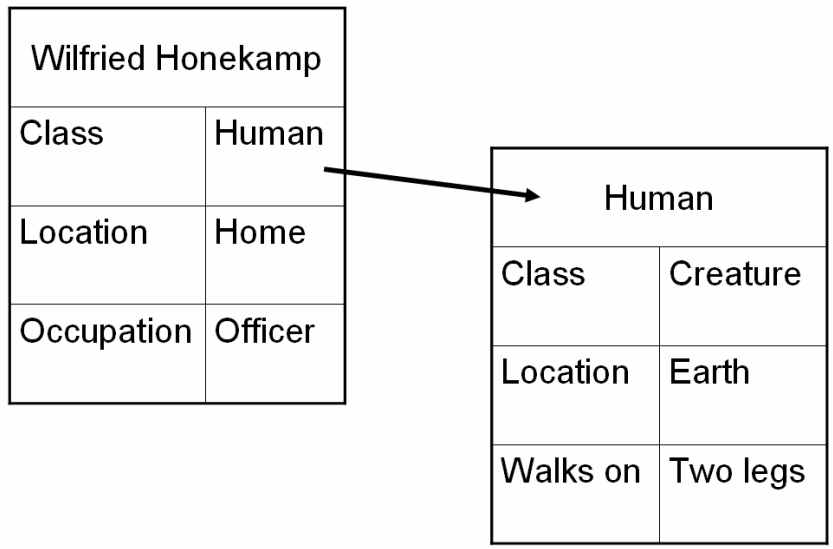


Fig: Two frames describing a human being

Conceptual Dependencies:

Conceptual Dependency originally developed to represent knowledge acquired from natural language input.

The goals of this theory are:

- To help in the drawing of inference from sentences.
- To be independent of the words used in the original input

- That is to say: *For any 2 (or more) sentences that are identical in meaning there should be only one representation of that meaning.*

CD provides:

- a structure into which nodes representing information can be placed
- a specific set of primitives
- at a given level of granularity.

Sentences are represented as a series of diagrams depicting actions using both abstract and real physical situations.

- The agent and the objects are represented
- The actions are built up from a set of primitive acts which can be modified by tense.

dfd

Conceptual Dependency (CD) is a theory of how to represent the kind of knowledge about the events that is typically contained in natural language processing sentences. The goal is to represent the knowledge in a way that

(i) Facilitates drawing the inferences from the natural language sentences.
(ii) Is independent of the language in which natural language sentences were originally stated.

Because of the two concerns mentioned, the CD representation of a sentence is built not out of primitives corresponding to the words in the sentence, but rather out of conceptual primitives that can be combined to form the semantic meanings of the words in any particular language. The conceptual dependency theory was first developed by Schank in 1973 and was further more developed in 1975 by the same author. It has been implemented in a variety of programs that read and understand natural language text processing. Unlike semantic nets which provide only a structure into which nodes representing information and a specific set of primitives, at a particular level of granularity, out of which representations of particular pieces of information can be constructed.

In CD's the symbols have the following meanings.

Arrows indicate the decision of dependency.

Double arrow indicates two way link between actor and action

P indicates past tense.

ATRANS is one of the primitive symbol acts used by the CD theory. It indicates transfer of possession

O indicates the object case relation

R indicates the recipient case relation.

As a simple example of the way knowledge is represented in conceptual dependency theory, the event represented by the sentence.

'I gave the man a book'

would be represented in conceptual dependency theory, the event represented by the sentence as shown.

In Conceptual dependency, the representations of actions are built from a set of primitive acts. Although there are slight differences in the exact set of primitives actions provided inn the various sources on CD, a typical set is the following, which are taken from the Schanks conceptual dependency theory.

- RANS – transfer of an abstract relationship (give)
- PTRANS – transfer of the physical location of an object (go)
- PROPEL – application of physical force to an object (push)
- MOVE – movement of a body part by ots owner (kick)
- GRASP – grasping of an object by an actor (clutch)
- INGEST – ingestion of an object by an animal (eat)
- EXPEL – Expulsion of something from the body of an animal (cry)
- MTRANS – Transfer of mental information (tell)
- MBUILD – Building new information out of old (decide)
- SPEAK – production of sounds (say)
- ATTEND – Focusing of a sense organ toward a stimulus (listen)

Conceptual Dependency the symbols have the following meanings.

• Single arrows (<—) indicate direction of dependency.
• Double arrows (o) indicates two way link between action and actor.
• P indicates past tenst.
• Primitives (ATRANS, PTRANS, PROPEL) indicates transfer of possession.
• O indicates object relation.
• R indicates recipient relation.

Conceptual Dependency introduced several interesting ideas:

(i) An internal representation that is language free, using primitive ACTs instead.
(ii) A small number of primitive ACTs rather than thousands.
(iii) Different ways of saying the same thing can map to the same internal representation.

There are some problems too:

(i) the expansion of some verbs into primitive ACT structures can be complex
(ii) Graph matching in NP-hard.

Conceptual dependency theory offers a set of primitives conceptualizations from which the world of meaning is build.
These are equal and independent, They are :

ACTs : Actions
PPs : Object (picture procedure)
AAs : Modifiers of actions (action aiders)
PAs : Modifiers of objects i.e., PPS (picture aiders)
AA : Action aiders are properties or attributes of primitive actions.
PP : Picture produces are actors or physical objects that perform different acts or produces

Scripts

A *script* is a structure that prescribes a set of circumstances which could be expected to follow on from one another. Scripts are used for representing knowledge about common sequences of events.

It is similar to a thought sequence or a chain of situations which could be anticipated.

It could be considered to consist of a number of slots or frames but with more specialised roles.

Scripts are beneficial because:

- Events tend to occur in known runs or patterns.
- Causal relationships between events exist.
- Entry conditions exist which allow an event to take place
- Prerequisites exist upon events taking place. *E.g.* when a student progresses through a degree scheme or when a purchaser buys a house.

The components of a script include:

Entry Conditions
-- these must be satisfied before events in the script can occur.
Results
-- Conditions that will be true after events in script occur.
Props
-- Slots representing objects involved in events.
Roles
-- Persons involved in the events.

Track

-- Variations on the script. Different tracks may share components of the same script.

Scenes

-- The sequence of *events* that occur. *Events* are represented in *conceptual dependency* form.

The classic example is the restaurant script:

Scene: A restaurant with an entrance and tables.
Actors: The diners, servers, chef.
Props: The table setting, menu, table, chair.
Acts: Entry, Seating, Ordering a meal, Serving a meal, Eating the meal, requesting the check, paying, leaving.

Advantages of Scripts:

- Ability to predict events.
- A single coherent interpretation may be build up from a collection of observations.

Disadvantages:

- Less general than frames.
- May not be suitable to represent all kinds of knowledge.

Uncertain Knowledge

Let action A_t = leave for airport t minutes before flight

Will A_t get me there on time?

Problems:

1. Partial observability (road state, other drivers' plans, etc.)
2. Noisy sensors (radio traffic reports)
3. Uncertainty in action outcomes (flat tyre, etc.)
4. Complexity of modeling and predicting traffic

Hence a purely logical approach either

1. Risks falsehood: “A25 will get me there on time” or

2. Leads to conclusions that are too weak for decision making: “A25 will get me there on time if there's no accident on the bridge and it doesn't rain and my tires remain intact etc etc.”

A1440 might reasonably be said to get me there on time but I'd have to stay overnight in the airport…

Handling Uncertainty

Instead of providing all condition it can express with degree of beliefs in the relevant sentences.

Example:

Say we have a rule

if toothache then problem is cavity

But not all patients have toothaches because of cavities (although perhaps most do)

So we could set up rules like

if toothache and not(gum disease) and not(filling) and .....then problem is cavity

This gets very complicated! a better method would be to say

if toothache then problem is cavity with probability 0.8

Given the available evidence,

A_{25} *will get me there on time with probability 0.04*

A most important tool for dealing with degree of beliefs is probability theory, which assigns to each sentence a numerical degree of belief between 0 & 1.

Probability Basics:

Probability of getting 1 when a die is rolled

$P(1) = 1/6$

Probability of getting a number < 4 when a die is rolled

$P(< 4) = P(1) + P(2) + P(3) = 1/6 + 1/6 + 1/6 = 1/2$

Proposition = Set of atomic events in which it is true.

$(a \vee b) = (\neg a \wedge b) \vee (a \wedge \neg b) \vee (a \wedge b)$

$P(a \vee b) = P(\neg a \wedge b) \vee P(a \wedge \neg b) \vee P(a \wedge b)$

$(a \wedge b) =$ a v b – (

The **prior or unconditional** probability associated with a proposition is the degree of belief accorded to it in the absence of any other information.

Example:

P(Weather = sunny) = 0.72

P(Cavity = true) = 0.1 or P(cavity) = 0.1

Probability distribution gives values for all possible assignments:

P(Weather) = (0.72, 0.1, 0.08, 0.1)

The conditional probability "P(a|b)" is the probability of "a" given that all we know is "b".

e.g., *P(cavity|toothache) = 0.8* means if a

Patients have toothache and no other information is yet available, then the probability of patient's having the cavity is 0.8

$P(a|b) = P(a \wedge b)/ P(b)$

$P(a \wedge b) = P(a|b)\, P(b)$

We consider the following domain consisting of three Boolean variables: Toothache, Cavity, and Catch (the dentist's nasty steel probe catches in my tooth).

The full joint distribution is the following 2x2x2 table:

	toothache		¬toothache	
	catch	¬catch	catch	¬catch
Cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

P(cavity v toothache) = P(cavity, toothache, catch) + P(cavity, toothache, ¬catch) + P(cavity, ¬toothache, catch) + P(cavity, ¬toothache, ¬catch) + P(¬cavity, toothache, catch) + P(¬cavity, toothache, ¬catch)

= 0.108+0.012+0.072+0.008+0.016+0.064=0.28

P(cavity) = 0.108 +0.012 +0.072 + 0.008 = 0.2

P(¬Toothache) = 0.072 + 0.008 + 0.144 + 0.576 = 0.8

P(Cavity, ¬Toothache) = 0.072 + 0.008 = 0.08

P(¬cavity | Toothache) = P(¬cavity ^ Toothache)/ P(Toothache)

= (0.016 + 0.064)/(0.108 + 0.012 + 0.016 + 0.064)

= 0.4

Bayes Rule

P(b|a) = P(a|b) * P(b) / P(a)