Chapter 5
Structured Knowledge Representations

Approaches to knowledge representation:

- **logical**: use formal logic to represent knowledge
  e.g., state spaces, Prolog databases
- **procedural**: knowledge as a set of instructions for solving a problem
  e.g., production systems, expert systems (next week)
- **associationist**: knowledge as objects/concepts and their associations
  e.g., semantic nets, conceptual dependencies
- **structured**: extend networks to complex data structures with slots/fillers
  e.g., scripts, frames

Issues in Knowledge Representation:

Below are listed issues that should be raised when using a knowledge representation technique:

Important Attributes

-- Are there any attributes that occur in many different types of problem?

There are two *instance* and *isa* and each is important because each supports property inheritance.

Relationships

-- What about the relationship between the attributes of an object, such as, inverses, existence, techniques for reasoning about values and single valued attributes. We can consider an example of an inverse in

band(Amrit, Nepathya)

This can be treated as Amrit plays in the band *Nepathya* or Amrit's band is *Nepathya.*

Another representation is *band = Nepathya*

band-members = Amrit, Subas, …

Granularity

-- At what level should the knowledge be represented and what are the primitives. Choosing the Granularity of Representation Primitives are fundamental concepts such as holding, seeing, playing and as English is a very rich language with over half a million words it is clear we will find difficulty in deciding upon which words to choose as our primitives in a series of situations.

If *Tom feeds a dog* then it could become:

feeds(tom, dog)

If *Tom gives the dog a bone* like:

gives(tom, dog, bone) Are these the same?

In any sense does giving an object food constitute feeding?

If *give*(*x*, *food*) $\rightarrow$ *feed(x)* then we are making progress.

But we need to add certain inferential rules.

Clearly the separate levels of understanding require different levels of primitives and these need many rules to link together apparently similar primitives.

Obviously there is a potential storage problem and the underlying question must be what level of comprehension is needed.

Practical Aspects for 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.

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

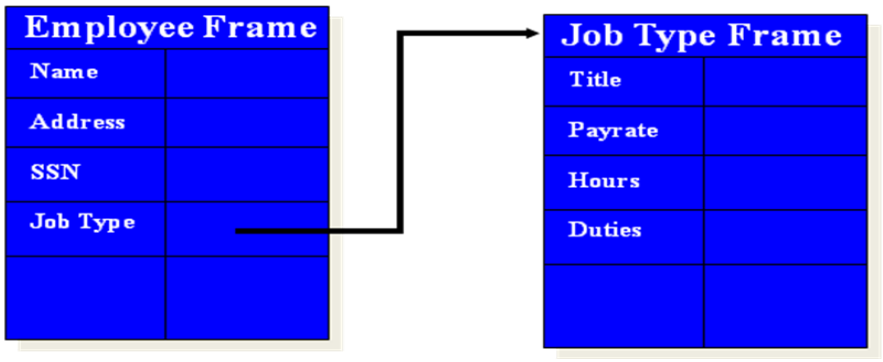
- complex
- reasoning (inferencing) is difficult
- explanation is difficult, expressive limitation

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.)

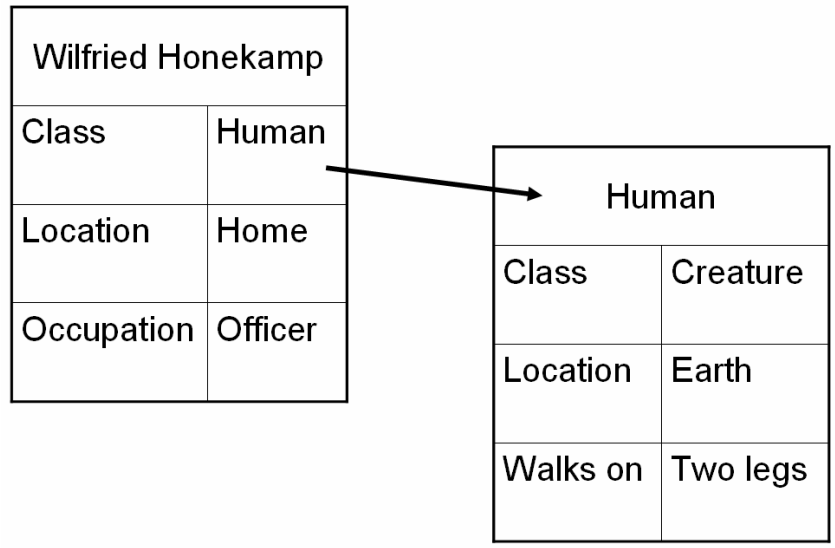


Fig: Two frames describing a human being

Object-based Approach: Semantic Network

Semantic Networks and Frames are called *Network Representations* or *Associative Representations*

Intuition: Knowledge is not a large collection of small pieces of knowledge but larger pieces that are highly interconnected. Logics and Rule-Based Systems seem to not have this property.

The meaning of concepts emerges from how it is connected to other concepts.

Semantic networks can
- show natural relationships between objects/concepts
- be used to represent declarative/descriptive knowledge

Knowledge is represented as a collection of concepts, represented by nodes. Thus, semantic networks are constructed using ***nodes*** linked by directional lines called ***arcs.***

A node can represent a fact description
- physical object
- concept
- event

An arc (or link) represents relationships between nodes. There are some 'standard' relationship types
- 'Is-a' (instance relationship): represent class/instance relationships
- 'Has-a' (part-subpart relationship): identify property relationships

In other words, a semantic network is often used as a form of knowledge representation. It is a directed graph consisting of vertices which represent concepts and edges which represent semantic relations between the concepts.

What does "Semantic" means?

Semantic networks are mainly used as an aid to analysis to visually represent parts of the problem domain.

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.

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:

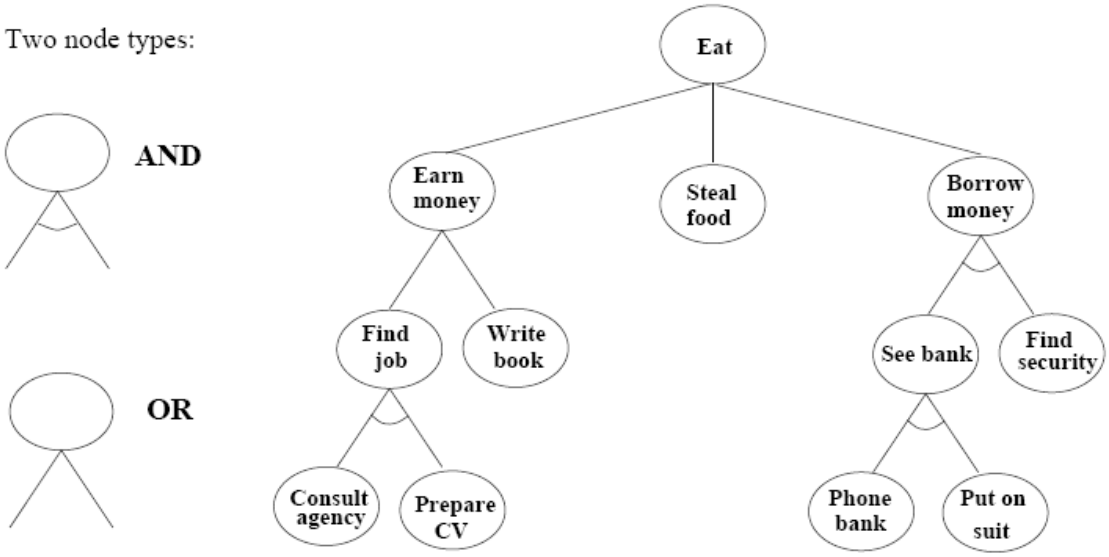
meanings are associated with the link and node labels (the details will depend on the application domain)

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

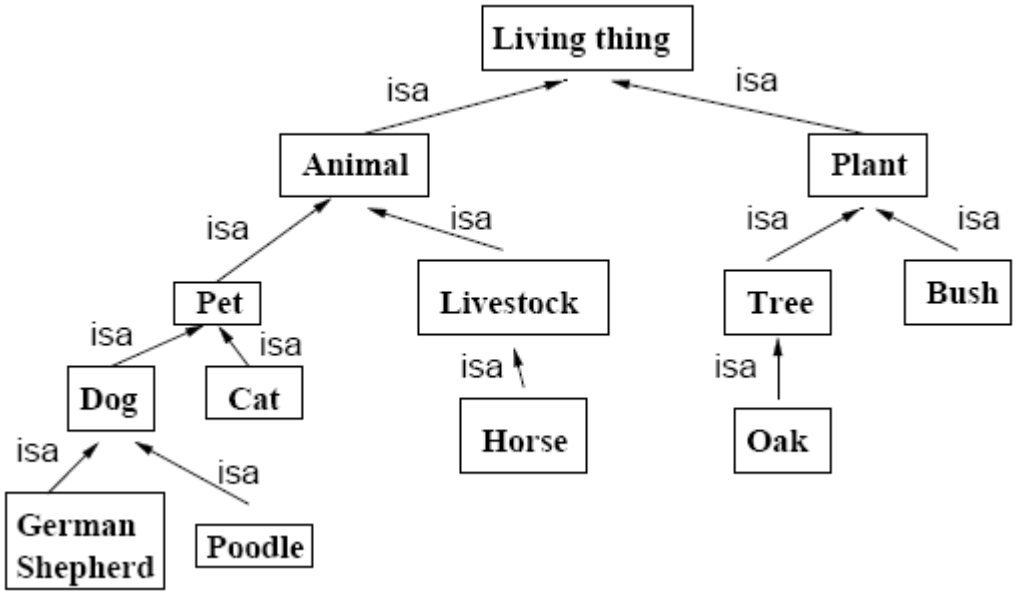
AND / OR Trees in Semantic net:

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



An Is-A Hierarchy:

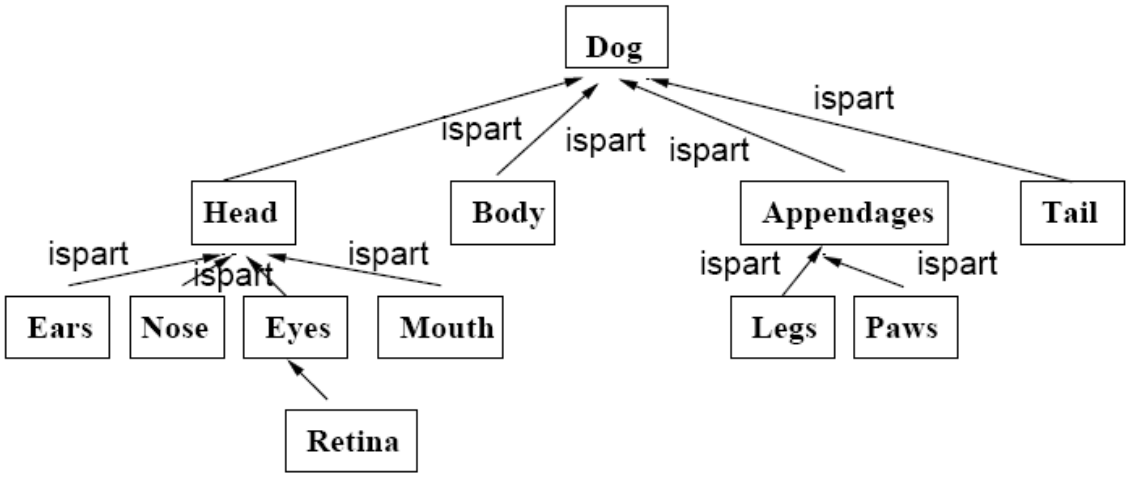
Another simple form of semantic network is an ***is-a hierarchy***. For example:



In set-theory terms, ***is-a*** corresponds to the sub-set relation Í, and ***instance*** corresponds to the membership relation Î.

Is-Part Hierarchy:

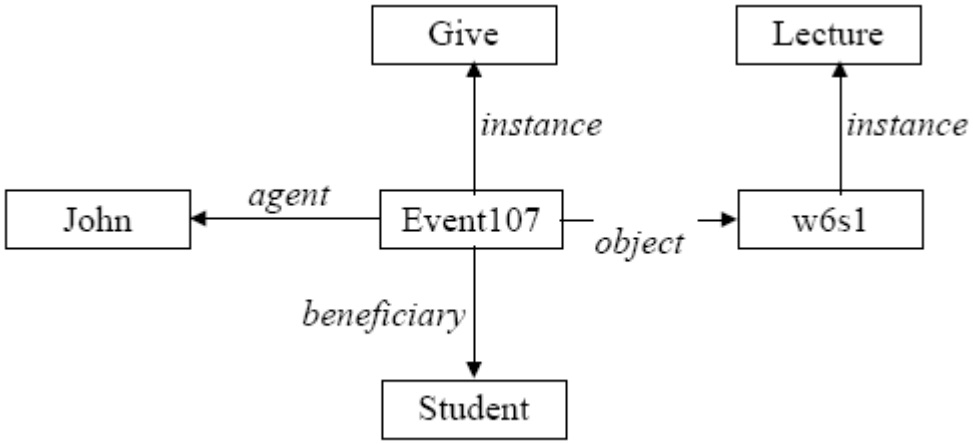
If necessary, we can take the hierarchy all the way down to the molecular or atomic level with an ***is-part hierarchy***. For example:



Representing Events and Language:

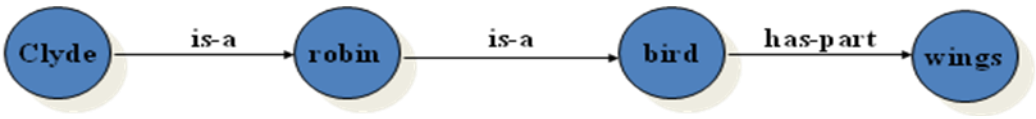
Semantic networks are also very good at representing events, and simple declarative sentences, by basing them round an "event node". For example:

"John gave lecture w6s1 to his students"



One feature of a semantic net is the ability to use the net to deduce new facts

- Begin with the fact that - all robins are birds
- If Clyde is the name of a particular pet robin then
- By following the is-a links it is easy to deduce that Clyde is a bird



Deduce: Robins have wings and Clyde has wings

More Examples:

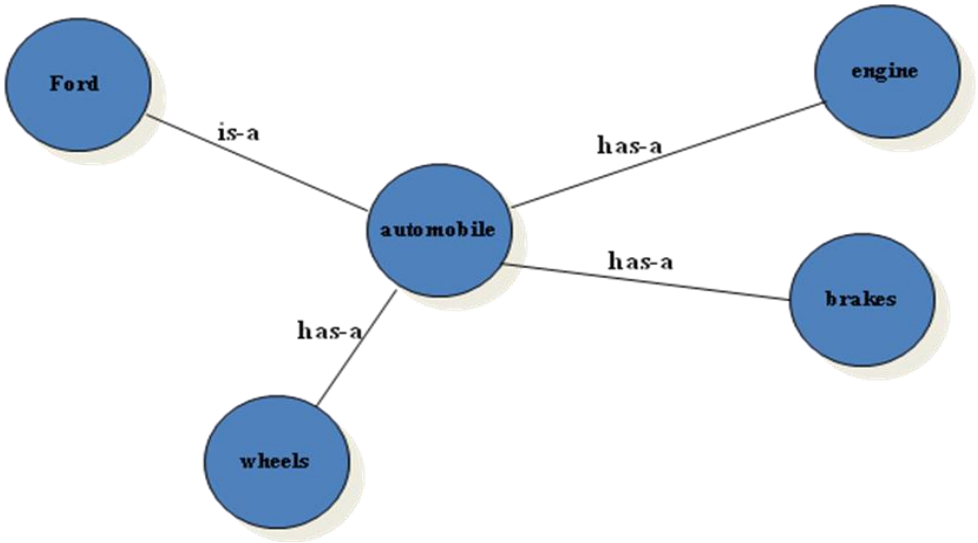


Fig: Automobile Semantic Net

Disadvantages of a semantic network

- incomplete (no explicit operational/procedural knowledge)
- no interpretation standard
- lack of standards, ambiguity in node/link descriptions
- not temporal (i.e. doesn't represent time or sequence)

Advantages of a semantic network

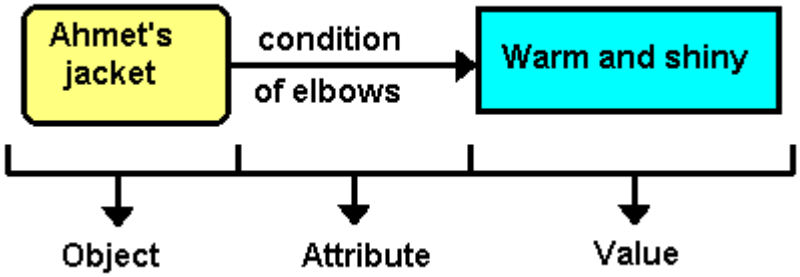
- Explicit and easy to understand.
- The net is its own index – quick inference possible.
- Supports default reasoning in finite time.
- Focus on bigger units of knowledge and interconnectedness.

O-A-V Triple:

Object – Attribute – Value (OAV) provides a particularly convenient way in which to represent certain facts and heuristic rules within KB. Each OAV triplet is present with specific entity or object and a set of attributes with their values associated to every object.

Objects may be physical or conceptual.

Attributes are general characteristics or properties associated with objects. For example, interest rate is an attribute for a bank loan.



O-A-V and Semantic Networks

O-A-V is a specialized case of the semantic-net approach.

object ==> attribute "has a"
attribute ?== value "is a"

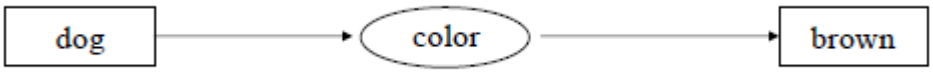
The object ==> attribute link is a "has-a" link.
A bank **has a** rate of interest.

The attribute ==>value links is a "is-a" link.

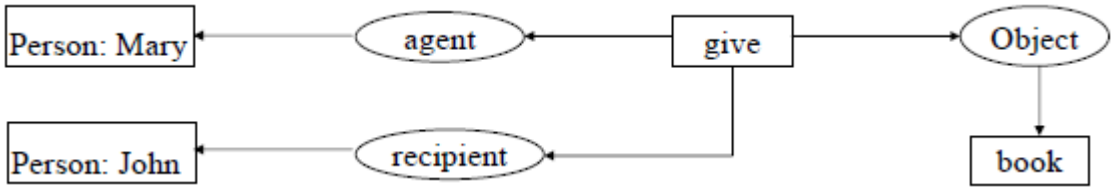
Conceptual Graph (source : www.conceptual graph.org)

The nodes of graph are either concepts or conceptual relations (no leveled arcs). Conceptual relation nodes indicate a relation involving one or more concepts. A relation of arity n is represented by a conceptual relation node having n arcs. The types and individual labels are separated by colon.
Examples:

1. Dog has brown color.



2. Mary give Jhon the book



Conceptual Dependencies:

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

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.

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

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.
- o→indicates the object case relation
- R→ indicates the recipient case relation.

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 in the various sources on CD, a typical set is the following, which are taken from the Schanks conceptual dependency theory.

- ATRANS – 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)

In CD theory, there are four conceptual categories:

ACTs : Actions (one of the above premitives)
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

To represent natural sentence, CD theory uses following rules:

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.

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.