2- Procedural Representation

- The most common technique for representing knowledge in computers is procedural knowledge.

- Procedural code not only encodes facts (constants or bound variables) but also defines the sequence of operations for using and manipulating those facts.

- Thus, program code is a perfectly natural way of encoding procedural knowledge. Whether data structures or objects are used to model the problem, the program is essentially one big knowledge representation.

- Programs written in scripting languages such as Visual Basic Script, JavaScript, and LotusScript are examples of a procedural knowledge representation.

Weakness of Procedural Code

In procedural code, the knowledge and the manipulation of that knowledge are inextricably linked.

Declarative

This weakness is overcome by the most popular knowledge representation approach, called declarative.

In declarative knowledge representation, a user explains the simple facts, rules, and relationships.
3- Network Representation

3-1 Semantic Nets

Semantic net at the time of its origin were used mainly in understanding natural languages, where the semantics (meaning) of the associated words in sentence was extracted by employing such nets.

For example the sentence: "the man hit the boy"

```
                 Sentence
                  |
                 /|
               noun phrase
               /  |
          article(The, a, ..) noun (N)
          /    |
     The    man

                  /|
               verb phrase
               /  |
         verb (V) noun phrase(NP)
         /    |
      hit    the
         /    |
   the     boy
```

Generally, semantic nets found a winder application in reasoning of knowledge based systems. A semantic net consists of two elementary tuples: events denoted by nodes and relationship between events denoted by links.

Semantic nets are used to define the meaning of a concept by its relationships to other concepts. A graph data structure is used, with nodes used to hold concepts, and links with natural language labels used to show the relationships. Frames and semantic nets are very closely related to predicate logic.
3-2 Conceptual Graph:

A conceptual graph is a directed bipartite graph. "Concepts" (objects, abstract objects, values) are represented by a node which is called a conceptual node. This node is drawn as a square or a rectangle. "Relations" are also represented by nodes which are called conceptual relation nodes. These nodes are drawn as an ellipse. No two nodes of the same type are connected by an arc. A conceptual node can only be linked to a conceptual relation node and vice versa.

Example:

Birds(fly)

Each Childs have a parent
A frame is a collection of attributes which defines the state of an object and its relationship to other frames (objects).

In AI, frames are called "slot-and-filler" data representations. The slots are the data values, and the fillers are attached procedures which are called before, during (to compute the value of), or after the slot's value is changed. Frames are often linked into a hierarchy to represent has-part and isa relationships.
In the figure:
- Each frame has a set of slots
- The automobile frame has three slots.
- The automobile has doors, a motor, and four wheels. It is a subset of vehicles.
- A sports car is a subset or type of automobile. It has two doors and is small. The sports car inherits several attributes from automobile.
- Corvette is an instance of a sports car, and each instance has a unique license number.

3-4 Petri Nets

What are Petri Nets?
Petri Nets are a graphical formalism for systems specification.

Petri Nets are formed from finite sets of:

**Places**

**Transitions**

**Arrows** connecting either places to transitions or transitions to places

- A Petri Net (PN) is given a state by marking its places.
- Marking of a PN consists of assigning a nonnegative integer to each place.
  - Graphically, tokens are inserted in places of a PN
- **Input place** - arrow goes from the place to the transition
- **Output place** - arrow goes from the transition to the place

- A transition may have one or more **Input** and **Output places**
- A transition is enabled if there is at least one token in each of its input places.
- An Enabled transition may fire:
  - one token is removed from each input place and one token is inserted in each output place of the transition
A Petri Net as a four-tuple \((P,T,I,O)\), where

- \(P\) is a set of *places*
- \(T\) is a set of *transitions*
- \(I\) is an input function:
  - *for places leading into a transition*
- \(O\) is an output function
  - *for places leading out of a transition*

Example:

- **t1** associations:
  - \(P_2 > P_1\) and \(P_4 := P_2 + P_1\)

- **t2** associations:
  - \(P_3 > P_2\)
  - \(P_4 := P_3 + P_2\)
  - \(P_5 := P_2 + P_3\)

4- Production Systems:

Is one of the oldest techniques of knowledge representation. Production system consist of a *set of production rules*, a *working memory*, and a *recognize-act control cycle*. 

Assist lec. : *Ahmed M. Al-Saleh*
1- The Set of Production Rules:

These are often simply called productions. A production is a condition, action pair and defines a single chunk of problem-solving knowledge. The condition part of a rule is a pattern that determines what the rule may be applied to a problem instance. The action part defines the associated problem-solving step.

Example:

if (person age above -21) & (person wife nil) & (person gender male) then (person eligible for marriage)

2- Working Memory:

The working memory (WM) generally holds data either in the form of clauses or object-attribute-value (OAV) triplet form. The variables in the condition part are matched against the data items of the WM. In cases all the variable instantiation of the condition parts of a rule are consistent, then the rule is fired and the new consequents are added to the WM and some old data of WM, which are no longer needed, are deleted from the WM to minimize the search time required for matching the condition parts of a rule with the data in WM.

Example for Production System:
The 8-Puzzle
Control Recognize:

1- Try each production in order.
2- Do not allow loops.
Stop when goal is found.