6. Rule based expert systems

The production system

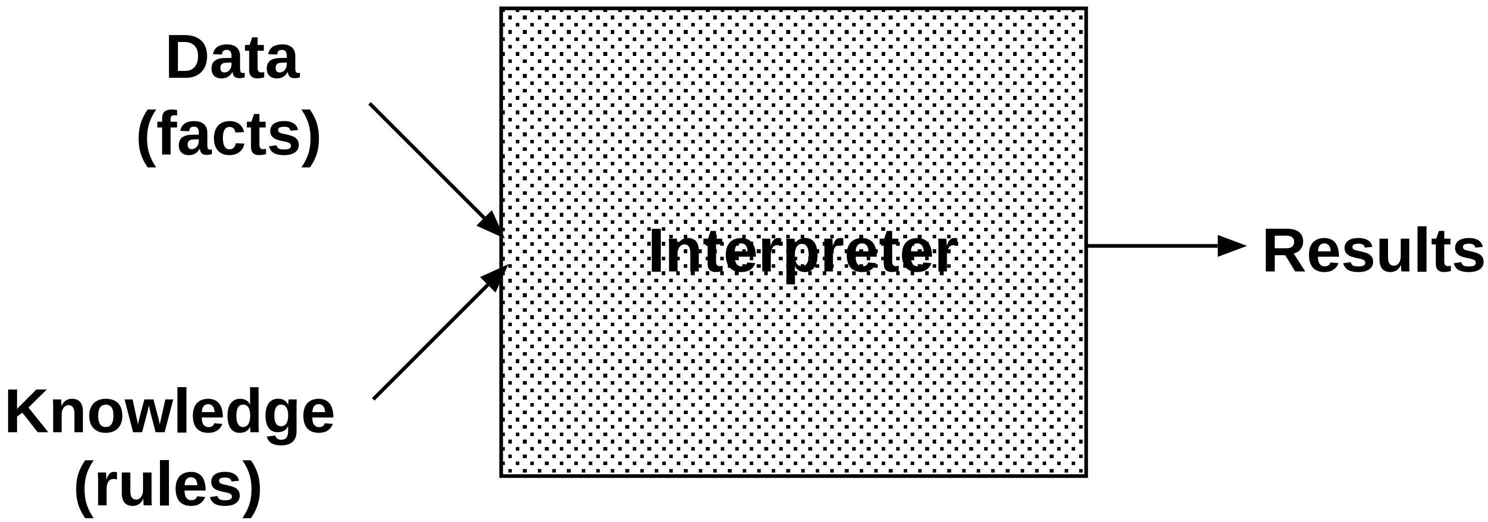

Figure 1: Architecture of a production system

Production rules

Format :

$$\textbf{IF}\ \texttt{Condition(s)}\ \textbf{THEN}\ \texttt{ACTION}\quad[\texttt{weight}]$$

A production rule is :

- Atomic.
- Independent.
- Declarative
- Natural.
- Evolutive.

The interpreter

Goal : Given a set of facts, proof a preposition or deduce (infer) new information.

- Forward chaining : data-driven strategy.
- Backward chaining : goal-driven strategy.
- Forward-Backward chaining.

The production system

Data-driven strategy : the recognize-act cycle

1. Detect the subset of *enabled* production rules (conflict set) by matching the facts describing the actual state against the **conditions** of the production rules. If there is no *enabled* rules then **exit**.
2. Conflict resolution : select one of the productions in the conflict set using one of the following ways :
   - Arbitrary choice.
   - Choose the most specific rule (containing the largest number of conditions).
   - Choose the least recently used rule.
   - Choose a rule where the condition is a new fact.
   - Choose the rule with the highest priority (weight).
   - Use Meta-Rules.
3. Fire the selected rule : the action of the selected rule is performed, changing the contents of the actual state.
4. If the goal is not reached Goto 1.

The production system

Example 1 : The 3 x 3 knight's tour problem

Goal : Determine whether a path exists from square 1 to square 2.

Production rules : Move rules where the **condition** of each rule specifies the square the piece must be on to make the move and the **action** the square to which it can move.

Strategy used : Forward chaining.

Conflict resolution strategy : Select and fire the first move rule encountered in the conflict set that does not lead to a repeated state.

1	2	3
4	5	6
7	8	9

Example 1 : The 3 x 3 knight's tour problem

RULE #	CONDITION		ACTION
1	knight on square 1	⟶	move knight to square 8
2	knight on square 1	⟶	move knight to square 6
3	knight on square 2	⟶	move knight to square 9
4	knight on square 2	⟶	move knight to square 7
5	knight on square 3	⟶	move knight to square 4
6	knight on square 3	⟶	move knight to square 8
7	knight on square 4	⟶	move knight to square 9
8	knight on square 4	⟶	move knight to square 3
9	knight on square 6	⟶	move knight to square 1
10	knight on square 6	⟶	move knight to square 7
11	knight on square 7	⟶	move knight to square 2
12	knight on square 7	⟶	move knight to square 6
13	knight on square 8	⟶	move knight to square 3
14	knight on square 8	⟶	move knight to square 1
15	knight on square 9	⟶	move knight to square 2
16	knight on square 9	⟶	move knight to square 4

Example 1 : The 3 x 3 knight's tour problem

Iteration #	Current square	Conflict set	Fire rule
0	**1**	1,2	1
1	8	13,14	13
2	3	5,6	5
3	4	7,8	7
4	9	15,16	15
5	**2**		**Halt**

The production system

- Goal-driven strategy :

  1. Consider the goal as the initial state.
  2. Detect the subset of *enabled* production rules (conflict set) by matching the facts describing the actual state (goal) against the **actions** of the production rules.
  3. When the action of a rule is matched, the condition(s) are added to the actual state.
  4. The process continues until a fact is found, usually in the problem initial description or, as is often the case in expert systems, by directly asking the user for specific information.
  5. The search stops when the condition(s) of all the productions fired in the backward fashion are found to be true.
  6. These conditions and the chain of rule firings leading to the original goal form a proof of its truth through successive inferences.

Example 2 : Diagnosing automotive problems

Rule 1 : **IF**

the engine is getting gas, and

the engine will turn over,

THEN

the problem is spark plugs

Rule 2 : **IF**

the engine does not turn over, and

the lights do not come on

THEN

the problem is battery or cables

Rule 3 : **IF**

the engine does not turn over, and

the lights do come on

THEN

the problem is the starter motor

Rule 4 : **IF**

there is gas in the fuel tank, and

there is gas in the carburator

THEN

the engine is getting gas

The production system

For any fact F : -1 $\leq$ Weight(F) $\leq$ +1.

- Weight(F) = +1 : Sure that F is true.
- Weight(F) = -1 : Sure that F is false.
- Weight(F) = 0 : F is unknown.

The production system

Rules for propagating the different weights :

- R1 : **If** A[C1] **and** B[C2] **Then** C [C3]
  - WC1 = Weight(C) = min(weight(A),weight(B)) $*$ weight(R1) = min(C1,C2) $*$ C3.
- R2 : **If** A[C1] **or** B[C2] **Then** C [C3]
  - WC2= Weight(C) = max(weight(A),weight(B)) $*$ weight(R2) = max(C1,C2) $*$ C3.
- Using the two rules :

$$Weight(C) = \begin{cases} WC1 + WC2 - WC1 * WC2 & \textbf{if } WC1, WC2 \geq 0 \\ WC1 + WC2 + WC1 * WC2 & \textbf{if } WC1, WC2 \leq 0 \\ (WC1 + WC2)(1 - min(|WC1|, |WC2|) & \textbf{if } WC1 * WC2 \leq 0 \end{cases}$$