

Production Systems

- Knowledge representation using logic can be seen as a procedural representation
- The steps to solve a problem are described as a chain of deductions
- This formalism is based on two elements:
 - **Facts:** Propositions or predicates
 - **Rules:** Conditional expressions where the consequent is usually an atomic predicate or an action
- Analogy with space state search
 - Facts = state of the problem
 - Rules = Search operators

Production systems

A production system is composed by:

- **Fact base:** Predicates that describe a problem
- **Knowledge base (or rule base):** Rules that describe the reasoning mechanisms that can be used to solve problems in a domain
- **Inference engine:** Applies the rules and obtain the chain of deductions that solve the problem

Inference Engine

- Role
 - Deduce new facts, apply actions to solve the problem (goal) from a set of initial facts and eventually by interaction with the user or the environment
- Components
 - Rule interpreter + control strategy
- Phases
 - Detection (filter): **Activable Rules**
 - Obtain a set of rule instantiations (**rule conflict set**)
 - Selection: **Rule to execute**
 - Conflict resolution: Select the rule instantiation to execute
 - Execution:
 - Execution of the rule

Detection

- Compute the set of possible instantiations of the rules (**Conflict set**)
- The rule interpreter computes all the possible instantiations with the facts in the current state of the problem (matching)
- A rule could be instantiated more than one time (rules with variables)

Selection

- This phase selects the *best* instantiation
- The rules are selected using a control strategy (*Conflict resolution strategy*)
 - Fixed strategy
 - Fixed dynamic strategy
 - Strategy guided by metarules
- Possible criteria:
 - First rule in order
 - The rule more/less times executed
 - The more specific/general rule
 - The rule with higher certainty degree
 - The instantiation that uses the facts:
 - with more priority,
 - that are older in the fact base (the oldest instantiation),
 - that are newer in the fact base (the more recent instantiation),
 - ...
- It is possible to use a combination of strategies

Execution

- Rule execution $\Rightarrow$
 - Update the fact base (if forward chaining reasoning is used)
 - New computations, new actions applied, query the user
 - New subgoals (if backward chaining is used)
- Instantiation propagation to the consequent
- Propagation of certainty.
- The deduction process ends when:
 - The conclusion (goal) is found $\Rightarrow$ success
 - There is no more rules to apply $\Rightarrow$ success? / fail?

Types of reasoning

- Deductive, forward chaining (FC), data driven reasoning
 - Evidences, symptoms, facts $\Rightarrow$ conclusions, hypothesis
- Inductive, backward chaining (BC), goal driven reasoning
 - Conclusions, hypothesis $\Rightarrow$ evidences, symptoms, facts
- Hybrid chaining

Data driven reasoning

- Based on Modus Ponens rule: $A, A \rightarrow B \vdash B$
- The base of facts (BF) is initialized with the facts that describe the problem

Procedure: Forward chaining

Input: Facts base, Rules base, Goals

Alternative $\leftarrow$ true

```
while  $\exists g (g \in Goals \wedge g \notin Facts\_Base) \wedge alternative$  do  
    Conflict_Set  $\leftarrow$  Interpreter.Satisfactible_Conditions(Facts_Base,  
    Rules_Base)  
    if Conflict_Set  $\neq \emptyset$  then  
        | Rule  $\leftarrow$  Control_Strategy.Conflict_Resolution(Conflict_Set)  
        | Interpreter.Apply(Facts_Base, Rule)  
    else  
        | Alternative  $\leftarrow$  false
```

Data driven reasoning

- Problems

- No goal oriented
- Combinatorial explosion, facts not related to the goal

- Advantages

- Intuitive to understand
- It is easier to formalize the knowledge this way, it is the more natural representation for many problems
- Can be used in an exploratory way

Goal driven reasoning

- Induction based. The goal is used as starting point and the chain of deductions that leads to it is obtained backwards
- Each iteration leads to new subgoals: hypothesis to validate

Procedure: Backward chaining

Input: Facts base, Rules base, Goals

No_alternative $\leftarrow$ false

while $Goals \neq \emptyset \wedge \neg No_alternative$ **do**

 Goal $\leftarrow$ Control_Strategy.Select_Goal(Goals)

 Goals.Delete(Goal)

 Conflict_Set $\leftarrow$ Interpreter.Satisfiable_goal(goal, Rules_Base)

if $Conflict_Set \neq \emptyset$ **then**

 Rule $\leftarrow$ Control_Strategy.Conflict_Resolution(Conflict_Set)

 Goals.Add(Rule.Extract_antecedent_as_goals())

else

 No_alternative $\leftarrow$ true

Goal driven reasoning

- Problems are solved by problem decomposition
- The resolution process is a search of an and/or tree
- Problems
 - The solution has to be known beforehand
- Advantages
 - Only the subgoals needed to solve the problem are considered

Hybrid reasoning

- Some parts of the chain of reasoning from the facts to the goal are constructed either deductively or inductively
- Bidirectional search
- The change from one strategy to another is done by using meta-rules
- The main advantage is to reduce the combinatorial explosion of forward chaining
- This strategy increases the efficiency of backward chaining when there is not a clear goal

Rules are used in many places

The use of rules as programming formalism is extended

- As a mechanism of transformation, compilation, translation, ...
 - Programming language compilers (LEX, YACC)
 - In the Web: XLTS (Extensible Stylesheet Language Transformations)
 - Task automation: Make, ANT, ...
- As a formalism to represent **business rules** in programs
 - Inference engines as a part of software development: Interpreted rules instead of code
 - Many commercial software development environments include rules: SAP, IBM, Oracle, Microsoft, ...
 - Are used in modern software development paradigms (Service Oriented Architectures)