Expert Systems - Introduction

Expert system is one representative tool of Artificial Intelligence. Its development was performed mainly in the 70/80s (the first one was born in 1965). The objective of this tool is to follow the same kind of reasoning then a Human (syllogisms and deductions) in order to:

- Classify
- Decide and Diagnose

Several examples of Expert Systems can be found:

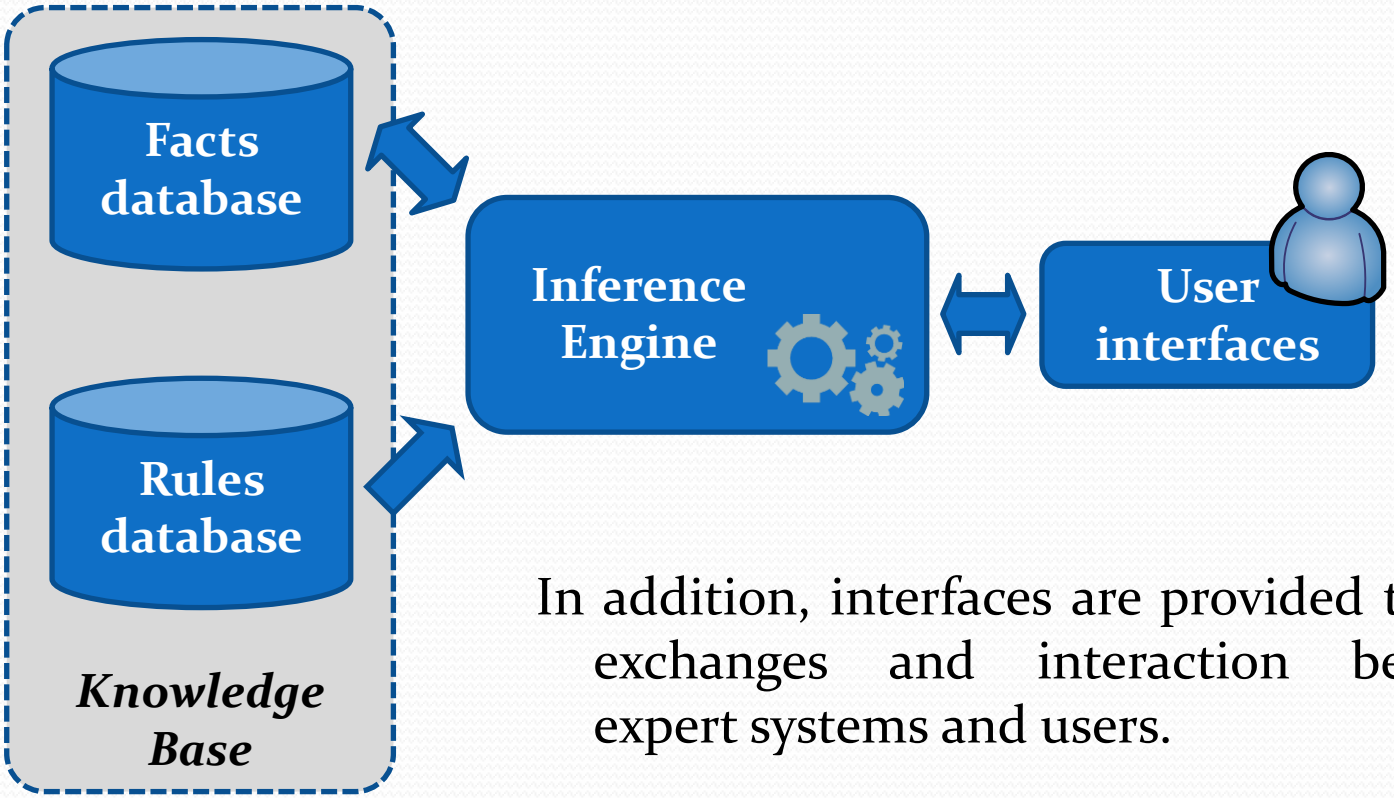
- Dendral (1967) : Molecular Physics identification from mass spectrometry **[Kindsay 67]**
- MyCin (1973) : Diagnostic of blood diseases
- Sachem (1990) : Control of furnaces
- GARI (1981) : Generation of manufacturing plans **[Descotte 81]**
- PROPEL : Machining plans generation **[Brissaud 92]**
- PART : Machining plans generation **[Van Houten 91]**

In order to work, this kind of tool needs rules and knowledge supplied by experts of the domain where the system is used.

Expert Systems - Architecture

Expert systems are composed of two main parts :

- The memories composed of rules + facts (the problem to solve and intermediate results)
- The Inference Engine, the core of the AI tool, which tries to fire rules according to the facts available



In addition, interfaces are provided to ease exchanges and interaction between expert systems and users.

Expert Systems - Knowledge Base - Facts

The Fact base is composed of several facts which are:

- Variable (during the reasoning facts are created, other are removed)
- Deducted from other facts by the inference engine
- Asked to the user if the expert system needs help

The types of facts can be generally:

- Boolean facts (True/False)
- Symbols (Strings)
- Real numbers

These kind of facts are needed to perform manufacturing generation

Expert Systems - Knowledge Base - Rules

The rules formalizes Knowledge needed to perform reasoning. These rules are described with the *If <Assumptions> Then <Conclusions>* structure, where the conclusion of using a rule is the creation or the modification of facts.

It is possible to express these rules:

"If A AND B Then C"

"If A OR B Then C AND D"

It is not possible to express this rule:

*"If A Then (B **OR** C)"*

Example of rule : If (Bore_Diameter > 25 AND Bore_Deepth > 200) Then Mfg_Operation = Foring AND Tool = Sand256B

The rules base does not evolve during the reasoning.

It is possible to characterize these rules with a **confidence degree**: consequently when two rules can potentially be fired, the system will firstly use the one with the best confidence degree.

The different kinds of rules

The power of an expert systems depends on the type of rules it can handle:

- **Order 0**: The Expert System can only handle Boolean relationships.

  Example: *If X = True And B = False Then…*

- **Order 0+**: In addition to Boolean relations, the system can handle comparisons operators (such as >, <, =, >= or =<).

  Example: *If X > 20 And B = False Then…*

- **Order 1**: The rules can be expressed with quantifier operators (∃ : *existential* quantifier, ∀ : *universal* quantifier) of predicate logic.

  Example: *If ∃ X And X ≠ "value" And Operation(X) <20 Then…*

Some rules called **Meta-rules** are used to describe to the system how it has to fire rules (mainly selection criteria).

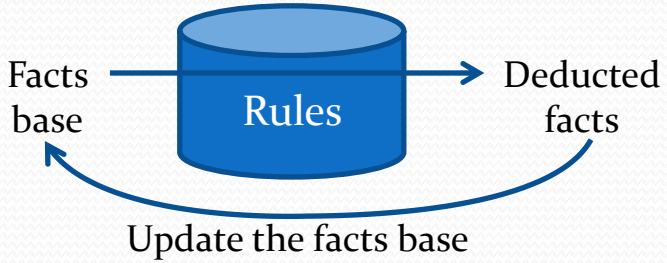
Expert Systems - Inference Engine

In order to manage these facts and rules, the inference engine follows a three step algorithm:

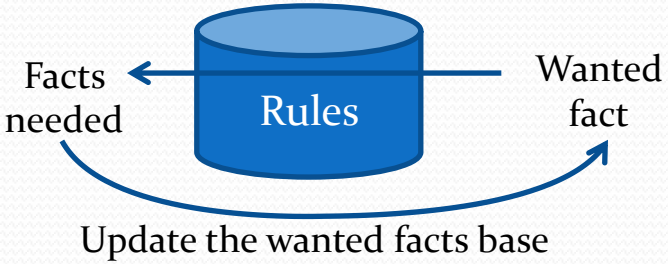
1. Filter: Find all rules triggered by the current facts base
2. Selection: Select the best rule to carry out
3. Activate: Fire the rule and update the facts base

Three type of reasoning (called chaining) are available:

- Forward Chaining: Deduct from facts and rules news facts

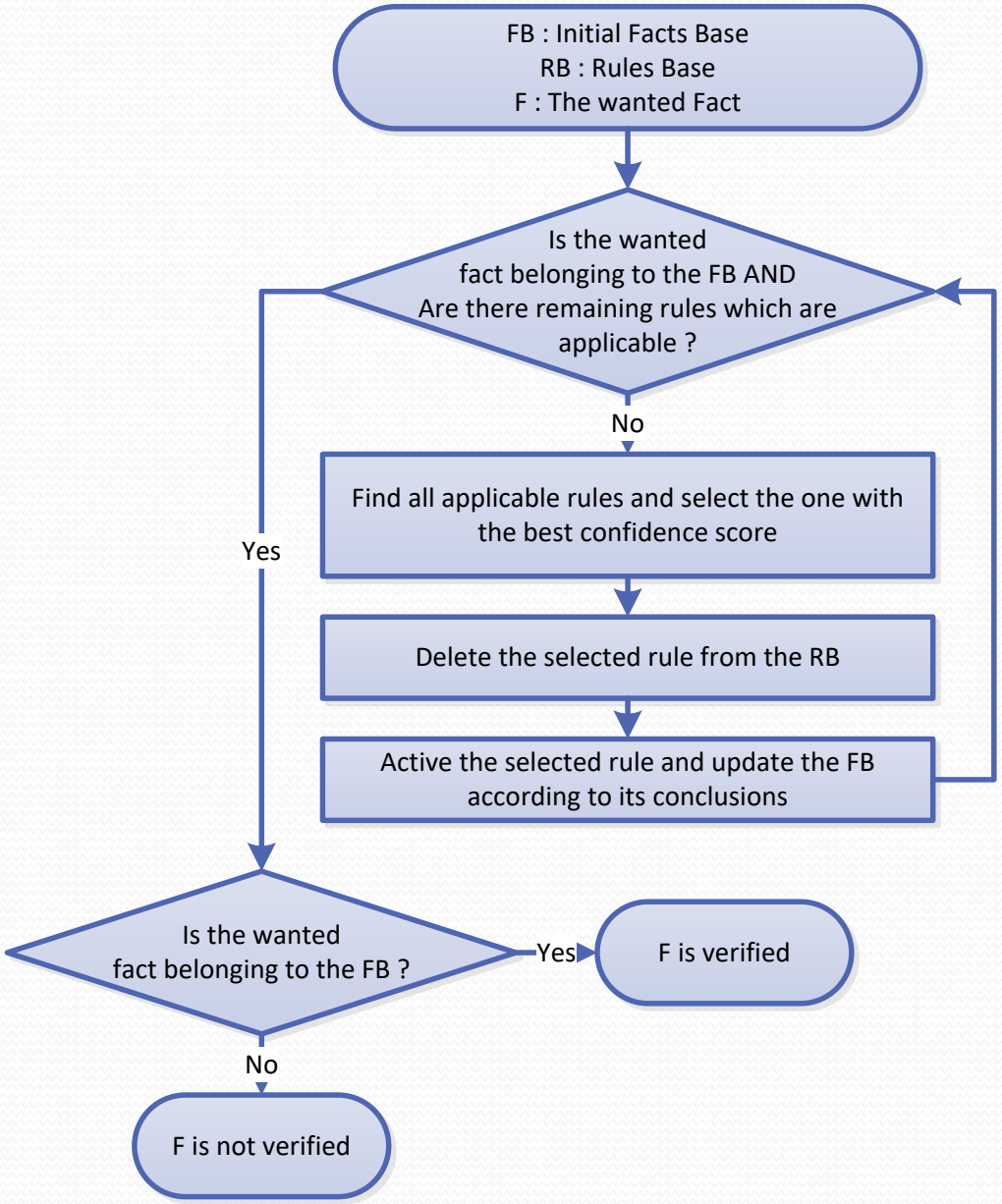


- Backward Chaining: Prove the validity of a wanted fact



- (Mixed Chaining)

Expert System - Forward Chaining - Algorithm



Expert System - Forward Chaining - Exercise

Consider the following case:

- Initial Facts: FB={B,C}
- Rules:
  - If B And D And E Then F
  - If G And D Then A
  - If C And F Then A
  - If B Then X
  - If D Then E
  - If X And A Then H
  - If C Then D
  - If X And C Then A
  - If X And B Then D

Questions : Is the Fact "H" verified ?

Are there only one solution?

Solution

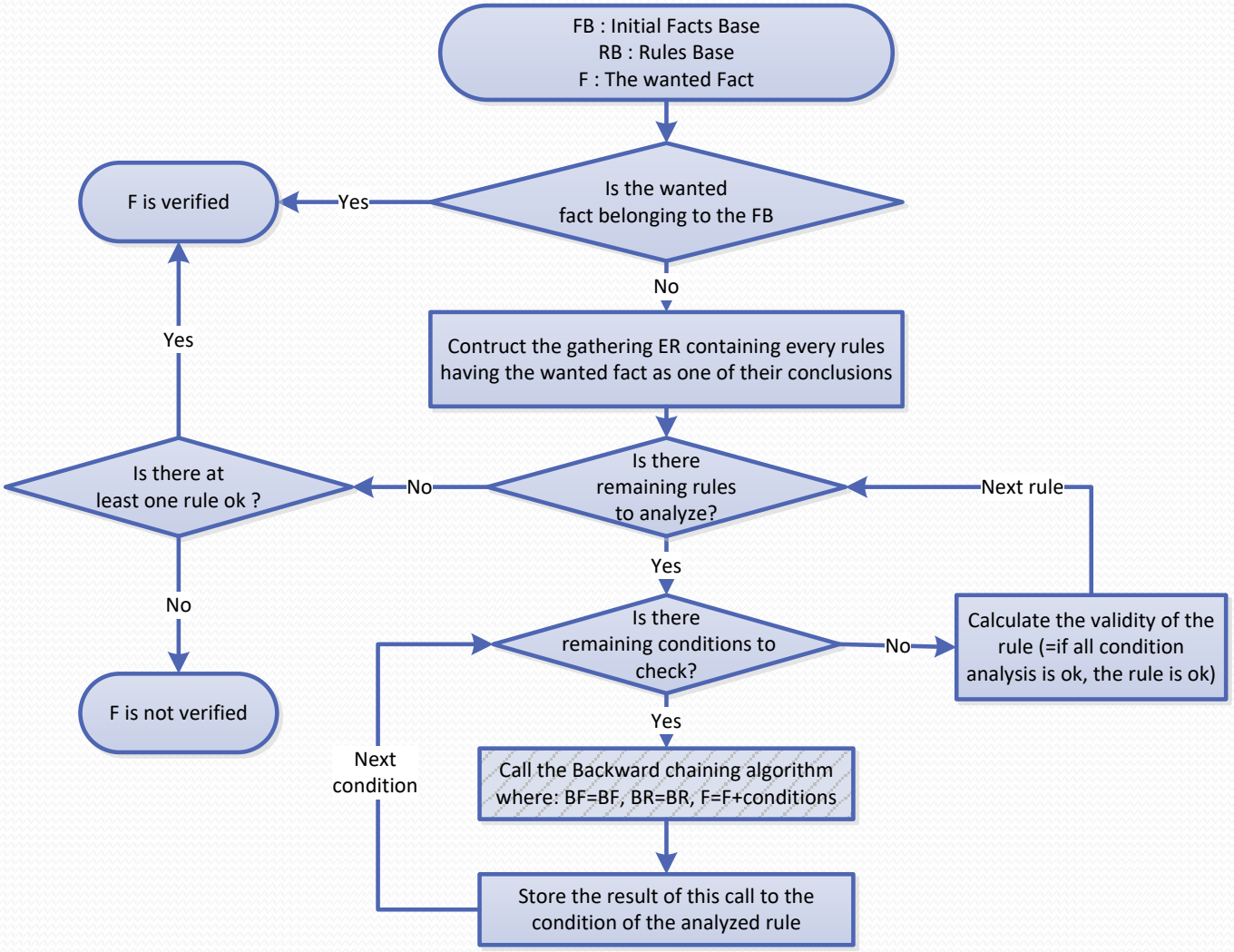
Forward Chaining - Brief Synthesis

As a brief conclusion of this algorithm of Forward chaining:

- This algorithm is interesting since:
  - It is very easy to understand, and so, to code
  - It always stops – no infinite loop - (when all rules are fired or when the Facts base is empty)
- However, the process time can become very long… (depending on the size of the rules and facts bases)
- It is not necessarily the fastest way that the algorithm uses to perform the fact validation

Expert System - Backward Chaining

The Forward chaining is a **recursive** Algorithm (that's to say that the algorithm can call the same one with different inputs (a easier issue to solve)).



Expert System - Backward Chaining - Example

Consider the following case:

- Initial Facts: FB={A,F}
- Rules:
  - If A And B And C Then G
  - If B And D Then E
  - If F And G Then E
  - If A Then H
  - If B Then C
  - If E And H Then I
  - If F Then B
  - If H And F Then E

Questions : How to verify Fact "I" ?

Are there only one way to verify this fact?

Solution

Backward Chaining - Brief Synthesis

As a brief conclusion of this algorithm of Forward chaining:

- This algorithm is interesting since:
  - It doest not need to explore the whole facts base
  - It can be associated with a user-interface asking to user some questions where some facts doesn't belong to rules conditions (like D in the previous example)
- However, this algorithm is more difficult to implement and code.
- Do to its recursive design, it is possible to explore the same part of the tree (like, for instance for the fact B in the previous example)
- It is not necessarily the fastest way that the algorithm uses to perform the back tracking (in the previous example using Rule 8 is an easier way to prove the fact E).

Expert System – Exercice #2

We want to build an Expert System to scan professional social networks (such as *LinkedIn.com* or *Viadeo.com*) in order to find interesting profiles for our company.

R1 IF Responsibilities AND Easy speaking AND Speak Dutch THEN Dynamic
R2 IF Easy speaking AND Speak English THEN Adaptable
R3 IF Slav AND Dynamic THEN Adaptable
R4 IF Responsibilities THEN Leadership
R5 IF Easy speaking THEN Speak Dutch
R6 IF Adaptable AND Leadership THEN Interesting Profile
R7 IF Slav THEN Easy speaking
R8 IF Leadership AND Slav THEN Adaptable

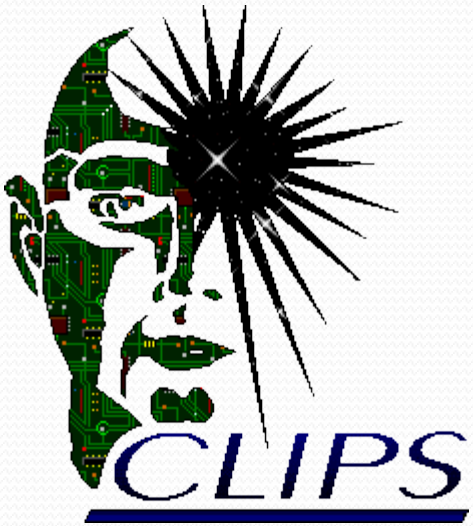
Use the forward chaining to identify what are the deducted facts if the initial fact is only: "*Slav*" (an Indo-European ethno-linguistic group).

Expert System – Exercice #2

We want to understand how the fact "*Interesting Profile*" can be reached with backward chaining approach. To do so, build completely the deduction tree leading to this fact. What are non-demonstrable facts which are needed to validate the fact "*Interesting Profile*"? Are there useless facts and/or rules for this chaining?

Example of an Expert Systems - CLIPS

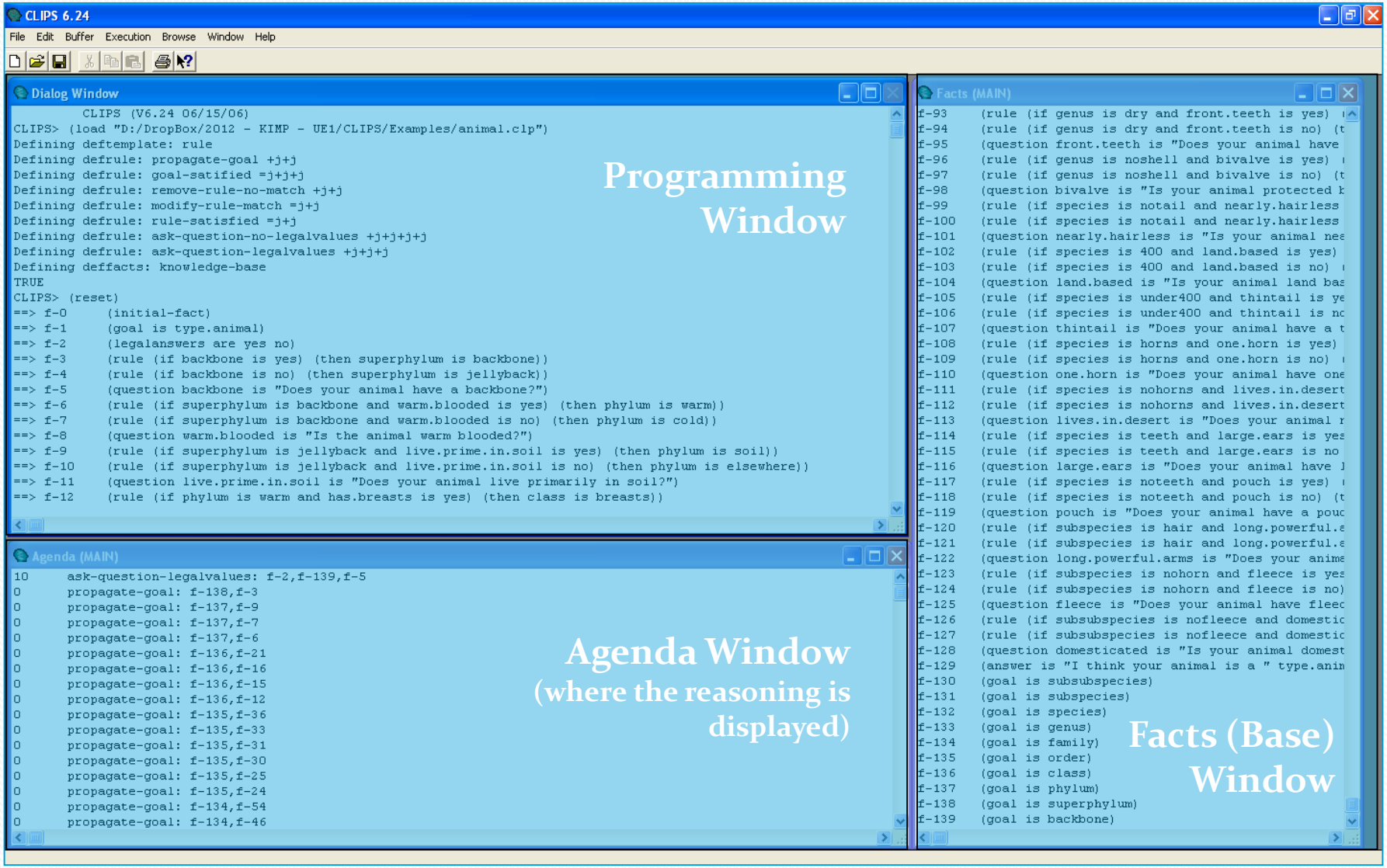
Presentation of CLIPS Tool:



- CLIPS is a **free** tool (public domain) for building expert systems by proposing a framework where it is possible to:
  - Express rules
  - Describe facts
  - Run all type of chaining (improved version of those seen previously in this course)
- It was firstly developed by the NASA in the 80s and is still updated (the last version, 6.24 was released in April 2008)
- CLIPS proposed several tools to ease the debugging of rules translation into LISP language, to check consistency of rules bases…
- It provides connectors to integrate CLIPS routines into bigger programs

For these reasons, this tool is selected to illustrate several cases of Expert Systems use in Manufacturing plans generation.

CLIPS - User Interface



CLIPS - Facts

Facts are easily handled in CLIPS:

- They can be easy ones:

  (assert *OneFact***)**

- Or structured and so more detailed and powerful ones:
  - By using structure

    (deftemplate *NameOfTemplate "Comment about the Structure"*
    (slot *Attribute1*
    (Type *TypeoftheAttribute***)**
    (default *ValuebyDefault***))**
    (slot *Attribute2*
    …
    (default *ValuebyDefault***)))**

    Name of template

    Attribute's definition

  - By using classes (same definition than the template but having inheritance operator (=is-a))
- They can be stored in a kind of database called **deffacts**

CLIPS - Facts - Examples

- Simple facts

  (assert Cost_Limit 5600**)**

  (assert Bore Diameter 25 Length 45**)**

- Templates
  - Template Definition

    (deftemplate Threaded_Bore *"Definition of Threaded Bore geometrical feature"*

    (slot *Name* **(type** *Symbol***) (default** *none***))**

    (slot *Length* **(type** *Float***) (default** 0.0**))**

    (slot *Diameter* **(type** *Float***) (default** 0.0**))**

    (slot *Thread* **(type** *Integer***) (default** 0**)))**
  - Template Use

    (assert (Threaded_Bore (Name f12) (Length 45) (Diameter 25) (Thread 1)**))**

CLIPS - Rules

The rules in CLIPS are:

- Written in a kind of LISP Language
- Follow this format:

```
(defrule Rule_Name "Optional Comments"      } Name of rule
    (declare (salience Integer))            } Salience definition
    (condition_ 1)                          }
    ...                                     } Assumptions (If)
    (condition_n)                           }
=>
    (action_1)                              }
    ...                                     } Conclusions (Then)
    (action_m))                             }
```

- Can be prioritized with the **salience** parameter (from -5000 to 5000)
- Can use **variable** to make filters (simple or complex ones) or just to make links between data from several facts in a rule
  - Single value: **?**VariableName
  - Multiple values: **$?**

CLIPS - Rules - Examples

Several examples of rules:

- *Simple:*

```
(defrule Rule_1 "Rule 1"
    (B)
    (D)
    (E)
=>
    (assert(F)) )
```

- *More complex:*

```
(defrule VolumeOk "Calculate de Volume to Machine"
    (Threaded_Bore (Diameter ?Dia) (Length ?Len) (Thread o) (Name ?Nam))
    (test (< (* ?Dia ?Len) 2500))
=>
    (printout t "Ok volume machinable for the feature " ?Nam crlf) )
```

Using the template previously defined

Variable definition

Displaying the result of the activation of the rule

CLIPS - Handling facts

Facts can be: **asserted** (like previously illustrated), **retracted** or **modified**. To retract or modify a particular fact (meeting the criteria of a rule), it is mandatory to store it previously in one variable:

```
(defrule VolumeOk "Calculate de Volume to Machine"
   ?feature <- (Threaded_Bore (Diameter ?Dia) (Length ?Len) (Thread o)
   (Name ?Nam))
    (test (< (* ?Dia ?Len) 2500))
=>
    (printout t "Ok volume machinable for the feature " ?Nam crlf)
    (retract ?feature)
)
```

The fact is stored in the variable

The fact is deleted

CLIPS - Handling facts

Facts can be: **asserted** (like previously illustrated), **retracted** or **modified**. To retract or modify a particular fact (meeting the criteria of a rule), it is mandatory to store it previously in one variable:

```
(defrule VolumeOk "Calculate de Volume to Machine"
   ?feature <- (Threaded_Bore (Diameter ?Dia) (Length ?Len) (Thread o)
   (Name ?Nam))
    (test (< (* ?Dia ?Len) 2500))
=>
    (printout t "Ok volume machinable for the feature " ?Nam crlf)
    (modify ?feature (Length 0.0) (Diameter 0.0))
)
```

The fact is stored in the variable

The modifications concern slots of the fact

CLIPS - Operators

Several operators are available in CLIPS. They used the **prefix form**. In the prefix form of CLIPS, the function precedes the arguments, and parentheses must surround the numeric expression : **(+ 2 3)**

They can be combined, for instance 2*(3+7): **(* 2 (+ 3 7))**

Several classic operators are available : **+, -, *, /, ** ** (^{2}), **sqrt, max, min, abs**... Several extended math functions are available(trig, hyperbolic...)

It is possible to store the result of any operation (simple or complex ones) into a variable through the binding instruction: **(bind ?result (+ 2 3))**. It is useful when this result is needed for rule criteria.

You can build your own functions (with the instruction **deffunction**)

CLIPS - DEMOs and Works

Several demos are proposed to illustrate how does CLIPS work:

- **Example 1:** Description of the problem use to illustrate the forward chaining
  Aim: To see how facts and rules work in CLIPS

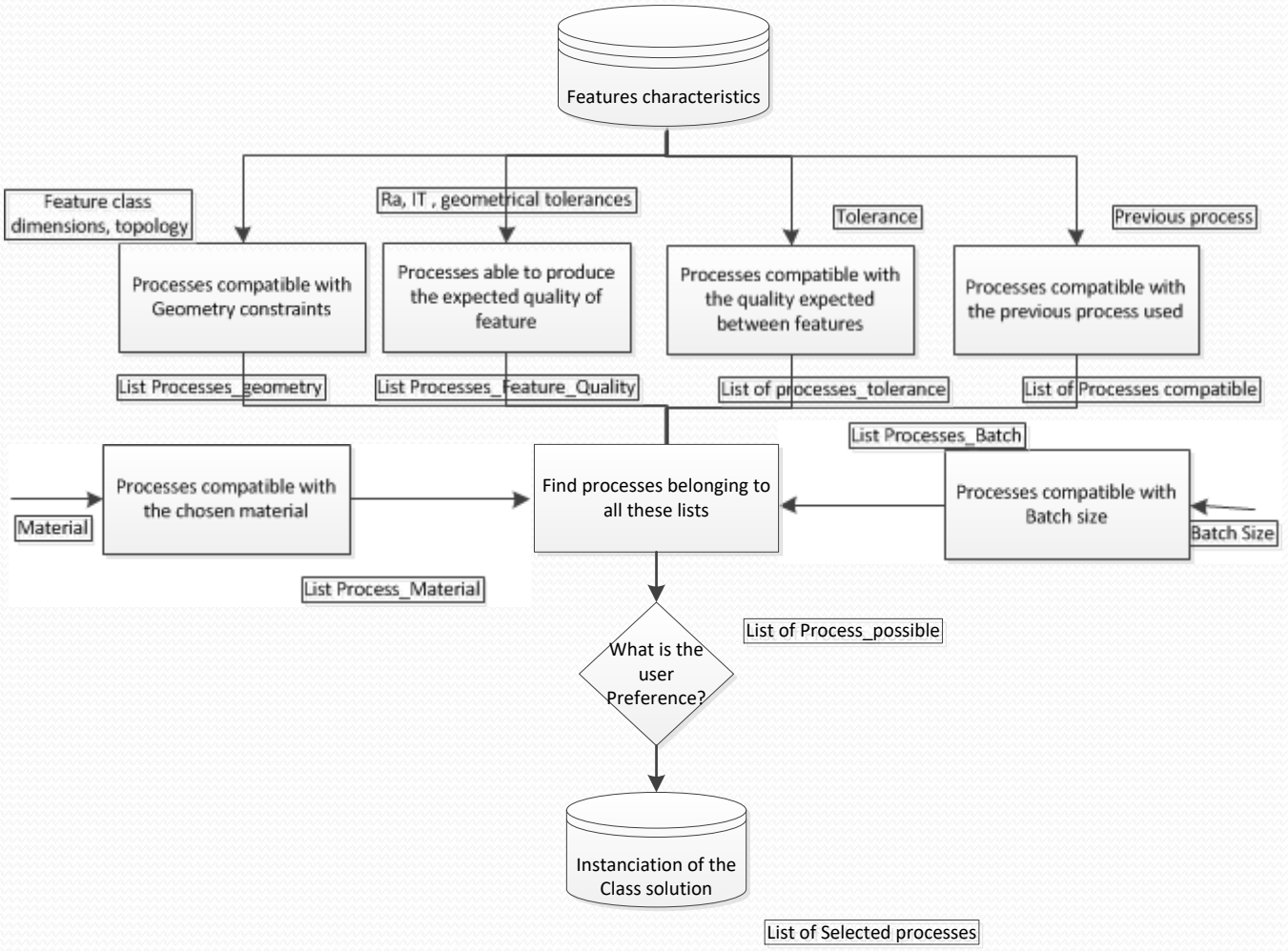
- **Example 2:** A small example of templates and variable use in rules
  Aim: To see how variables can be used

- **Example 3:** Using Expert System to generate conceptual process planning with interactions with the user

- **Example 4:** Using Expert System to generate the process plan of prismatic part with only axial features (hole, drilling…) in order to assess the machining cost

CLIPS - Conceptual Process planning

[Houin 05] proposed to use expert system to follow the Ashby's approach coded in its tool CES. Expert System is used as a constraint program which is able to select processes compatible with several constraints are user preferences:



Expert Systems - Drawbacks

The drawbacks of this approach depends mainly on the drawbacks of rules based systems:

- The difficulty to identify and express rules:
  - It is difficult for expert to **identify and express how they work** and what is the different steps of their reasoning.
  - Experts of the domain handled by the Expert System (and so who are the most suitable to express rules) are not computer scientist: it is then difficult to **translate these rules into logical language** which are used to code rules (especially LISP which needs to be prepared to use it).
  - The previous drawback can be solve thanks to the service of a Knowledge engineer. However adding an intermediate in the extraction flow decreases the quality of the Knowledge extracted.
  - How expert systems can manage rules based on **fuzzy expression** of Knowledge? Example: *If the hole is **large** then…* What *large* means, how to express it into rules?

- The difficulty to manage these rules:
  - Difficulty of managing the **consistency** of hundreds of rules
  - Difficulty to perform the **maintenance** of rules base (to modify, add and remove rules)
  - Some experts have the black box effect when using Expert Systems

CLIPS Lab - PAG V1.0

The aim of this little Lab is to develop a simple implementation of PAG approach with CLIPS system:

- Defining concepts (Feature, Operation, Tool)
- Defining rules (Cutting Tool Chart)
- Running inferences

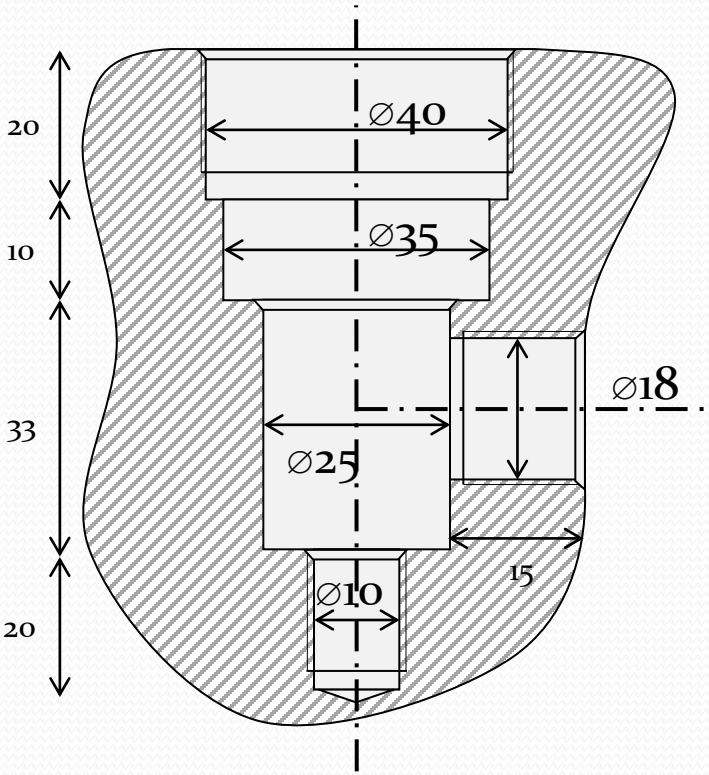
Concepts definition - Feature

Three Concepts have to be defined by using CLIPS template with their descriptive parameters:

- Machining Feature
  - Name (SYMBOL)
  - Type (SYMBOL)
  - Diameter (FLOAT)
  - Depth (FLOAT)
  - Tapping (TRUE or FALSE)
  - Type of Bottom (SYMBOL). Two values are considered : “Square” and “Conical”
  - Chamfer (TRUE or FALSE)

Instantiations - Features

The first concept to instantiate is the Feature one. Add to fact base (by using a `Initial-Fact` Rule) the features describing the part below.



Manufacturing Rules

- By using the `DefRule` pattern in CLIPS, express the rule above, synthetized into a cutting tool chart

Type of Feature	Constraints on its parameters	Machining Operation	Feature resulting
Non through bore	Tapping = False Bottom Conical Length/Diameter ≥2	Deep Drilling	Raw material (no feature remains)
	Tapping = False Bottom Square Diameter >20	Countersinking	Bottom conical Diameter = 15
	Tapping = False Bottom Conical	Drilling	Raw material (no feature remains)
Through bore	Tapping = False Diameter >10	Drilling	The same having it diameter divided by 2
	Tapping = False Diameter <15	Drilling	Raw material (no feature remains)
Both	Tapping = True	Tapping	The same having no tapping
	Chamfer= True	Chamfer	The same having no chamfer

- After a first run, by modifying the salience parameter, modify the behavior of the program to force him to machine first the tapping and chamfer

Tool and Operation - Template & Rule

- Tool
  - Name (SYMBOLS)
  - Type (SYMBOLS). Three main types are considered : “Drill”, “Milling cutter” and “Turning tool”
  - The diameter (FLOAT)
  - The cutting length (FLOAT)
- Machining Operation
  - Feature Name (SYMBOLS)
  - Operation Name (SYMBOLS)
- Modify manufacturing rules to manage compatibility between tools and features (regarding their diameters and length)

Rules Improvements

Of course this first try can be strongly improved by:

- Preventing the system from generating dead ends
- Taking into account the relationships linking features
- Taking into account machines and their capabilities
- Taking into account deviations of resources
- Handling the whole PAG network, not only one branch
- …