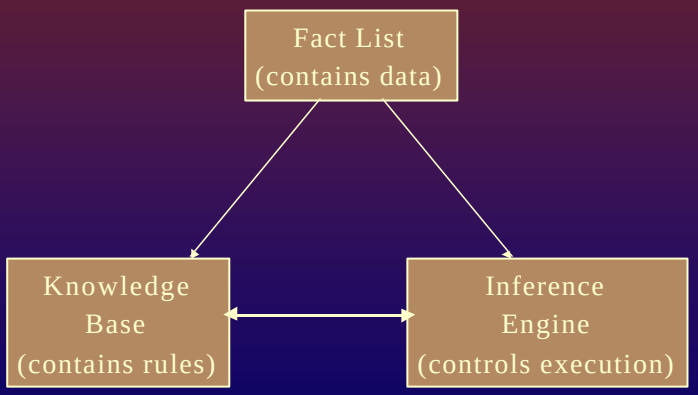
# Introduction to CLIPS




# Overview

- CLIPS is a programming language that provides support for rule-based, object-oriented and procedural programming.
- The search in the Inference Engine uses forward chaining and rule prioritization.
- Looks like LISP (List Processing) with object features.

## Object Features

- CLIPS Object-Oriented Language (COOL) is a hybrid of features found in Common Lisp Object and SmallTalk.
- Example: object template or frame for a bearing
  - (deftemplate bearing
    (slot type) (slot size) (slot load) (slot lubrication) (slot max_temperature). . .)

## Car Frame or Template Example

| Slots | Fillers |
|---|---|
| Name | car name |
| Type | sedan, sports, station_wagon . . |
| Manufacturer | GM, Ford, Chrysler, Toyota . . . |
| Owner | Name of owner |
| Wheels | 4, 6 |
| Transmission | manual, automatic |
| Engine | gasoline, diesel, methanol |
| Condition | lemon, OK, peach |
| Under-warranty | no, yes |

## Car Instance Example

| Slots | Fillers |
|---|---|
| Name | Alice's car |
| Type | station_wagon |
| Manufacturer | GM |
| Owner | Alice M. Agogino |
| Wheels | 4 |
| Transmission | manual |
| Engine | gasoline |
| Condition | OK |
| Under-warranty | yes |

## Notation

- Everything in parentheses is to be entered exactly are shown within the parentheses. (exit) or (+ 6 3 2)
- Square brackets indicate that the contents are optional. (example [1])
- Pointed brackets indicate that a replacement is to be made with a value of the type specified within the brackets. (example <integer>)

## Elementary Math Operators

- Arithmetic
  - Addition (+ 6 3 2)
  - Subtraction (- 6 3 2)
  - Multiplication (* 6 3 2)
  - Division (/ 6 3 2)
- Logical Arithmetic
  - (> 6 3 2)
  - (< 6 3 2)

## Notation wild cards

- A * following a description indicates that the description can be replaced with zero or one or more occurrences of the value, separated by spaces. <integer *>
- A + following a description indicates that the description can be replaced with one or more occurrences of the value, separated by spaces. <integer *>
- <integer>+ is equivalent to <integer> <integer> *
- A vertical bar | indicates a choice among one or more of the items separated by the bars.
- A | B | C means (A, B or C).

## Fields

- Tolkens represent groups of characters that have special meaning in CLIPS.
- Fields or CLIPS primitive data types are groups of tokens.
- Seven types of Fields
  - Float
  - Integer
  - Symbol
  - String
  - External address
  - Instance name
  - Instance address

## Facts

- A “chunk” of information in CLIPS is called a **fact**.
- Facts consist of a **relation name** followed by zero or more **slots** and their associated values.

## Deftemplate Construct

- A template for a fact needs to be defined in order to determine the name and number and types of slots.
- (deftemplate <relation-name> [<optional-comment>] <slot-definition>*]
- (deftemplate apple “facts about the color of apples” (slot color))

## Multifield Slots

- Use multislot in cases where you want an undefined zero or more “arity” to the slot.
- (deftemplate apple “facts about the color of apples” (multislot color))
- (assert (apple (color red)))
- (assert (apple (color green)))
- (assert (apple (color red green)))

## Adding and Removing Facts

- New facts are added with the **assert** command: (assert <fact>+)
- Facts can be listed with: (facts)
- Facts are numbered sequentially starting with 1. (fact 0 is baseline).
- Remove facts with: (retract <i>+ )
- Facts can also be duplicated with slot modifications (duplicate 0 (color brown)) .

## Deffacts Construct

- Useful for automatically asserting a set of facts.
- Useful for defining initial knowledge.
- Assumes template of first item and single slots by default.
- (deffacts apple "apple color facts"
  (apple (color red))
  (apple (color green)))

## Watch Command

- The watch command is for debugging programs.
- (watch <watch-item>)
- <watch-item>):
  - Facts or Rules
  - Activations
  - Statistics & Compilations
  - Focus or all

## Rules

- Rules are in the following form:

  (defrule <rule name> [<comment>]
  <patterns>* ; condition or left-hand side (LHS)
  => ; implies
  <action>* ;action, consequence or right-hand side (RHS)
- (<patterns> => <action>*) or

  (IF <patterns>* <action>*)

## Manipulating Constructs

- Listing members of a construct:
  - (list-defrules)
  - (list-deftemplates)
  - (list-deffacts)
- Display text of a construct with pp (pretty print command):
  - (ppdefrules <defrule-name>)
  - (ppdeftemplates <deftemplates-name>)
  - (ppdeffacts <deffacts-name>)

## Deleting Constructs

- Delete a member by undefining:
  - (undefrule <defrule-name>)
  - (undeftemplates <deftemplates-name>)
  - (undeffacts <deffacts-name>)
- Delete all with **clear** command: (clear)

## Load, Save, Print, Break commands

- **Load and Save files:**
  - (load <file-name>)
  - (save <file-name>)
- **printout** command:
  - (printout <logical-name> <print-items>*)
  - Default is usually the terminal.
- The **set-break** command allows the execution to be halted before a rule is fired: (set-break <rule-name>)

## Agenda and Execution

- A CLIPS program can be made to run with the **run** command.
  - (run [<limit>])
  - Where the optional <limit> is the maximum number of rules to be fired.
- Rules which can be activated are put on the **agenda** list.
- The rule with the highest **salience** or priority on the agenda is fired.

# Pattern Matching: Rules and Facts