**Knowledge Based System Course**

|  |  |
| --- | --- |
| Lab Objective: | Apply rule-based techniques to simple examples |
| Topics: | * Defining data types for slot values in deftemplate, Defining Rules, Agenda. * Commands: agenda, run, watch, list-deffacts, list-defrules, list-deftemplates, ppdefrule, ppdeftemplate, ppdeffacts, retract, refresh. |
| Contents | 1. **DataTypes of slots** **in deftemplate:**  * Using Deftemplate for a Hiring Expert System where a candidate is defined by:   Name, age, graduated from, Computer Skills, English proficiency.  (deftemplate candidate  (slot name)  (slot age)  (slot uni)  (slot computer-skills)  (slot E-lang)  )   * Slots could be assigned to a datatype and allowed values.   (deftemplate candidate  (slot name  **(type STRING)**)  (slot age  **(type NUMBER)**)  (slot uni)  (slot computer-skills  **(type SYMBOL)**  **(allowed-symbols ex good poor))**  (slot E-lang))  CLIPS> (assert (candidate  (name **3**)  (age 23)  (uni computer-science)  (computer-skills ex)  (E-lang poor)  )  )  [CSTRNCHK1] A literal slot value found in the assert command does not match the allowed types for slot name.  CLIPS> (assert (candidate  (name "Hala")  (age 23)  (uni computer-science)  (computer-skills **ec**)  (E-lang poor)  )  )  [CSTRNCHK1] A literal slot value found in the assert command does not match the allowed values for slot computer-skills.  CLIPS>   1. **The Components of a Rule:**  * The pseudocode for one of the possible rules:   IF the emergency is a fire  THEN the response is to activate the sprinkler system   * The rule expressed in CLIPS is:   (defrule fire-emergency “An example rule"  (emergency (type fire))  =>  (assert (response (action activate-sprinkler-system))))   * The general format of a rule is:   (defrule rule name> [<comment>]  <patterns>\* ;Left-Hand Side (LHS) of the rule  =>  <actions>\* ) ;Right-Hand Side (RHS) of the rule  **Example:**  **Write a file that contains the following text:**  (deffacts initial-colors (colors red blue))  (defrule rule-1 (colors red white blue) => (assert (rule 1 fires)) )  (defrule rule-2 (colors red blue) => (assert (rule 2 fires)) )  (defrule rule-3 (colors blue red) => (assert (rule 3 fires)) )  (defrule rule-4 (colors white red) => (assert (rule 4 fires)) )  a) And we issue the commands:  clips> (load “cfile.clp”) clips> (reset) clips> (facts)  What is in the memory?  b) Suppose that we now:  clips> (run)  What is in the memory?   1. **Note that:**  * CLIPS attempts to match the patterns of rules against facts in the fact list, if all the patterns of a rule match facts, the rule is activated and put on the agenda, the collection of activated rules. * A program normally ceases execution when there are no rules on the agenda to fire.  1. **Displaying the Agenda:**   The agenda has the list of rules that are activated (have all of their pattern matched) and ready to execute. It can be displayed using the command (agenda).  Example: (Use the previous file)  clips> (clear)  clips> (load “cfile.clp”)  clips> (facts)  clips> (agenda) ;what is the output?  clips> (reset)  clips> (facts)  clips> (agenda) ;what is the output?  clips> (run)  clips> (facts)  clips> (agenda) ;what is the output?   1. **Watch Command:**   In the previous example we called the commands (facts) and (agenda) many times to check them, instead we can use (watch facts) and (watch activations) such that any changes in facts or agenda will be displayed automatically.  Repeat the previous but with watch:  clips> (clear)  clips> (load “cfile.clp”)  clips> (watch facts)  clips> (watch activations)  clips> (reset)  clips> (run)  To undo watch, use (unwatch <watched\_item>)   1. **Variables:**  * We want to define a rule that prints good IT candidates (a good IT candidate must be graduated from computer-science with excellent computer skills). Variable names must be preceded with a “?” and names are case-sensitive.   (defrule IT  (candidate (name ?name-var) (age ?age-var) (uni computer-science) (computer-skills ex))  =>  (printout t "IT candidates: " ?name " is " ?age " years old, computer science graduate with excellent computer skills" crlf))  Run rules on the following facts:  clips> (deffacts candidates (candidate  (name "Ahmed")  (age 23)  (uni computer-science)  (computer-skills ex)  (E-lang poor))  (candidate  (name "Ali")  (age 42)  (uni commerce)  (computer-skills good)  (E-lang good))  (candidate  (name "Amira")  (age 31)  (uni computer-science)  (computer-skills ex)  (E-lang ex))  )  clips> (reset)  clips> (facts)  clips> (run)  IT candidates: Amira is 31 years old, computer science graduate with excellent computer skills  IT candidates: Ahmed is 23 years old, computer science graduate with excellent computer skills   * (retract fact-index) to remove a fact:   clips> (retract 1)  clips> (run)  Due to the refraction property rules are activated only once, to re-run a rule on the same set of facts use *(refresh rule-name)*  clips> (refresh IT)  clips> (run)  IT candidates: Amira is 31 years old, computer science graduate with excellent computer skills   1. **Commands for Manipulating Constructs:**  * Displaying the List of Members of a Specified Construct   (list-defrules) ;displays the names only  (list-deftemplates) ;displays the names only  (list-deffacts) ;displays the names only   * Displaying the Text Representation of a Specified Construct Member   (ppdefrule<defrule-name>) ;displays the definition  (ppdeftemplate<deftemplate-name>) ;displays the definition  (ppdeffacts<deffacts-name>) ;displays the definition |