**Knowledge Based System Course**

|  |  |
| --- | --- |
| Lab Objective: | * To introduce CLIPS as Expert system shell * understand the fundamental workings of an ES shell   + knowledge representation   + reasoning * apply rule-based techniques to simple examples |
| Topics: | * Usage of Basic commands such as assert, reset, load, clear, facts….. * Defining ordered and non-ordered facts * deftemplate, deffacts constructs |
| Contents | 1. **Basic elements of an Expert System**  * Fact-list : Global memory for data * Knowledge-base : Contains all the rules * Inference Engine : Controls overall execution  1. **A program in CLIPS consists of**  * Facts * Rules  1. Commands can be entered directly to CLIPS; this mode is called the top level.   Examples: *clips>(+ 3 4)*  *clips>(exit)*  A symbol surrounded by parentheses is considered to be a command or function call.  The input (+ 3 4) is a call to the + function.  The input (exit) invokes the exit command.  Note that clips work in a prefix notation where operators come before operands.  Each CLIPS command must have a matching number of left and right parentheses.   1. Clips is case-sensitive. 2. **A Fact:**  * 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.   Example:  (person  (hair-color black)  (name "John Q. Public")  (eye-color blue)  (age 23))  The symbol person is the fact’s relation name and the fact contains four slots:  name, age, eye-color, and hair-color.   * The order in which slots are specified is irrelevant. * Before facts can be created, CLIPS must be informed of the list of valid slots for a given relation name.   The general format of a deftemplate is:  (deftemplate <relation-name> [<optional-comment>]  <slot-definition>\*)  The syntax description <slot-definition> is defined as:  (slot <slot-name>) | (multislot <slot-name>)  The deftemplate for the person fact:  (deftemplate person "An example deftemplate"  (slot name)  (slot age)  (slot eye-color)  (slot hair-color))  What if I would like the name slot to have its value without the double quotation like this (name John Q. Public), then the name slot must be defined as (multislot name).   * Ordered facts:   Facts with a relation name that does not have a corresponding deftemplate are called ordered facts.  Example:  (number-list 7 9 3 4 20)  This fact is different from the fact:  (number-list 9 7 3 4 20)   1. **assert:**   It inserts a fact or more into the working memory.  Example:  clips>(assert (number-list 7 9 3 4 20))  <Fact-0>  clips> (assert (person  (hair-color black)  (name "John Q. Public")  (eye-color blue)  (age 23))  Missing Function Declaration for hair-color.  clips> (deftemplate person "An example deftemplate"  (slot name)  (slot age)  (slot eye-color)  (slot hair-color))  clips> (assert (person  (hair-color black)  (name "John Q. Public")  (eye-color blue)  (age 23)))  <Fact-1>  Now try this:  clips> (facts)   1. **Write a file that contains the following text:**   (deftemplate person "An example deftemplate"  (slot name)  (slot age)  (slot eye-color)  (slot hair-color))  a) And we issue the commands  clips> (clear)  clips> (load “cfile”)  clips> (assert (person  (hair-color black)  (name "John Q. Public")  (eye-color blue)  (age 23)))  clips> (facts)  b) Suppose that we want to add assert to the file, then we should transform it to a deffacts construct.  Groups of facts that represent initial knowledge can be defined using the deffacts construct.  Add the following to the file:  (deffacts people "Some people we know"  (person (name "John Q. Public") (age 24)  (eye-color blue) (hair-color black))  (person (name "Jack S. Public") (age 24)  (eye-color blue) (hair-color black))  (person (name "Jane Q. Public") (age 36)  (eye-color green) (hair-color red)))    The general format of a deffacts is:  (deffacts <deffacts name> [<optional comment>]  <facts>\* )  Now we enter:    clips> (clear)  clips> (load “cfile”)  clips> (reset)  clips> (facts)  The facts in a deffacts statement are asserted using the CLIPS reset command.  The reset command removes all facts from the fact list and then asserts the facts from existing deffacts statement. |