February 17, 2003

NAME:

CSC/CPE 481 Midterm Exam Winter 2003

Instructor: Franz J. Kurfess

Task I – Multiple Choice Questions
Mark the correct answers (only one per question).

a) Which of the following is *not* a main component of a typical expert system?

- ☐ knowledge base
- ☐ inference engine
- [✓] data base
- ☐ user interface 3

b) Which of the following is the best description of *knowledge acquisition*?

- ☐ the storage of knowledge in a format suitable for processing by computers
- [✓] the transfer of knowledge from humans to computers
- ☐ a computer-based mechanism for the generation of new conclusions from existing knowledge
- ☐ a description of the reasons why a particular solution was generated 3

c) Which of the following operations constitute the ”inference engine cycle”?

- ☐ knowledge acquisition, knowledge representation, and reasoning
- ☐ forward and backward chaining
- [✓] conflict resolution, execution, and match
- ☐ the RETE algorithm 3

d) What does CLIPS stand for?

- [✓] C Language Implementation Production System
- ☐ it is a combination of C (its implementation language) and Lisp (its appearance)
- ☐ Common Lisp Implementation Production System
- ☐ it is based on the initials of its developers 3

e) Which of the following CLIPS commands creates an *instance* of a fact?

- ☐ `facts`
- ☐ `deftemplate`
- [✓] `assert`
- ☐ `retract` 3

f) Which of the following CLIPS commands lists the instances of facts currently known to CLIPS?

- [✓] `facts`
- ☐ `deftemplate`
- ☐ `assert`
- ☐ `retract` 3

g) What is the role of the *antecedent* in a CLIPS rule?

- ☐ it contains the `defrule` keyword, the name of the rule, and an optional comment string
- [✓] it specifies the patterns that are to be matched against the facts
- ☐ it separates the antecedent and the consequent of the rule
- ☐ it contains the actions to be performed when the rule fires 3

h) Which of the following statements is the best description of *a procedural knowledge*?

- ☐ knowledge that is available prior to perception through senses
- ☐ knowledge that is verifiable through sensory perception
- [✓] knowledge that indicates how to perform some activity
- ☐ knowledge that is difficult to express through language 3

i) What does it mean that a logical sentence is *valid*?

- [✓] the sentence is true under all possible interpretations in all possible worlds
- ☐ the sentence is true under all possible interpretations in some possible worlds
- ☐ the sentence is true if there exists a true interpretation in some possible world
- ☐ the sentence is syntactically correct 3

j) What is *conflict resolution* in rule-based systems?

- ☐ If there is no rule that possibly matches the currently active facts, conflicting variable bindings may be eliminated through conflict resolution.
- [✓] If there are several rules that possibly match the currently active facts, one of them must be selected.
- ☐ Inconsistent rules in the knowledge base are modified or eliminated through conflict resolution.
- ☐ Conflict resolution is a more efficient variant of the resolution proof method for logic. 3

30

Task II – Short Questions

1. Describe the main differences between *natural language* and *rules* for the representation of knowledge based on the criteria below. What are the respective advantages and problems? 15

Name:

Aspect	*Natural Language*	*Rules*
Expressiveness	• very high expressiveness, especially with spoken language • flexible syntax allows multiple statements with identical or very similar interpretations • uncertainty, approximative statements can be made very easily, and with sophisticated distinctions (choice of words, sentence construction, punctutation marks) • used by humans for communication and representation of knowledge • effective in conveying knowledge • context is very important, otherwise ambiguous • interpretation is very critical to the user of the language •	• somewhat limited expressiveness, but not seriously so; equivalent to Horn clauses • much more cumbersome to use for humans • uncertainty, approximative statements can be difficult • mostly used for KR in computers • context is of limited importance • very low ambiguity • interpretation is critical for human users, but not so much for the manipulations performed by the computer •
Comprehensibility	• easy to understand for humans • difficult for computers • • • •	• mediocre • some statements must be made in an awkward manner • relevant knowledge may be distributed over several rules • large sets of rules become difficult to handle • •
Computational Complexity	• very high • flexible syntax • ambiguities must be resolved • human communication often deviates from syntactical rules • •	• low to medium for small sets of rules • reasonably efficient with some limitations (Horn clauses) • Rete algorithm increases efficiency substantially • • •
Advantages	• natural for humans • very expressive • • • •	• reasonable compromise between comprehensibility for humans, expressiveness, and computational efficiency • reasonably similar to the way humans express knowledge • some variants have strong formal foundations (e.g. Prolog) • •

NAME:

2. What is the role of *pattern matching* in Clips?. 7

- matches the antecedents of a rule with facts from working memory
- greatly increases the flexibility of rules; lifts rules from propositional logic to predicate logic
-
- guarantees the compatibility of variables across rules
- propagates values of variables between rules (variable binding)
- after rules are identified that are compatible with facts in working memory, one rule is selected, and its variables are attempted to be made consistent with the facts; if this is not possible, the rule is discarded
- the RETE algorithm makes pattern matching very efficient
- CLIPS specifically uses variables and wildcards
- one-to-one matching (only one slot) vs. one-to-many matching (multiple slots)
-

3. What limitations and inconveniences have you encountered in Clips? Describe their nature for at least three, and what you did to overcome or work around them. 8

   (a) Clips limitation:
   Complexity of Rules
   - Nature:

     Especially the LHS of rules can become quite complicated
   - Work-around:

     Break complex rules up into sets of smaller rules, possibly using modularity

   (b) Clips limitation:
   Size of Rule Base
   - Nature:

     Even for relatively simple domains the number of rules can become difficult to manage
   - Work-around:

     Try to alleviate it through careful design, modularity, good documentation

   (c) Clips limitation:
   Execution Sequence
   - Nature:

     It is difficult to control the sequence in which rules are executed
   - Work-around:

     In the first place, rules should be formulated in such a way that only as many rules as necessary apply to a specific set of facts. If this is done, salience can be used to enforce an ordering on rules

   (d) Clips limitation:
   No Nested Rules
   - Nature:

     rules within rules are not possible

- Work-around:

  Rules have to be designed accordingly. Modularity, auxiliary facts help to some degree.

(e) CLIPS limitation:
Arithmetic Calculations

- Nature:

  The reverse Polish notation for arithmetic expressions is cumbersome.

- Work-around:

  Use function calls for more complex calculations.

(f) CLIPS limitation:
Loose coupling between rules and objects.

- Nature:

  Rules and objects in Cool can communicate only through message passing.

- Work-around:

  Don't design the system in an object-oriented way. Use JESS instead.

(g) CLIPS limitation:
Debugging

- Nature:

  CLIPS programs can be hard to debug.

- Work-around:

  Use single-step execution, breakpoints, watch agenda, facts

(h) CLIPS limitation:
Abundant Use of Parentheses

- Nature:

  Parentheses are used to structure parts of programs.

- Work-around:

  Live with it ...

(i) CLIPS limitation:
Cryptic Error Messages

- Nature:

  The error messages for syntax problems are sometimes not very helpful.
- Work-around:

  Don't make syntax errors ...

(j) CLIPS limitation:
Basic Evaluation Mode is Forward Chaining

- Nature:

  Asking a query isn't straightforward.
- Work-around:

  Write a simple backward reasoner.

(k) CLIPS limitation:
Sting Comparison

- Nature:

  Testing strings for (in-)equality isn't straightforward.
- Work-around:

  Use `test, string-compare`

(l) CLIPS limitation:

- Nature:

- Work-around:

(m) Clips limitation:

- Nature:
- Work-around:

Task III – Clips **Program: Building Towers**

In this task, you need to trace the evaluation of a short Clips program. This program is a variation of the "blocks world" program discussed in class. The basic task is to arrange blocks into a stack according to their size. Complete the form on the next page, based on the program printed below by listing the asserted facts (indicated by `==>`), the retracted facts (indicated by `<==`), the rules on the agenda, and the actions performed in each step. The program should finish in step 8. The number of lines in each step does not necessarily indicate the number of entries for that step (some lines may be empty). 40

- see spreadsheet grading criteria: - correct firing of rules - selection of activated rules on the agenda - assertion and retraction of facts - output roughly five points per step

```
;; Stacking Blocks Example
;; from Peter Jackson, "Introduction to Expert Systems", 3d ed., p. 89-90
;; modified by Franz J. Kurfess, 02-04-2003

;; Templates: a block has color, size, location

(deftemplate block
  (field color (type SYMBOL))
  (field size (type INTEGER))
  (field place (type SYMBOL) (default heap))
)

(deftemplate on
  (field upper (type SYMBOL))
  (field lower (type SYMBOL))
  (field place (type SYMBOL) (default heap))
)

(deftemplate goal
  (field task (type SYMBOL))
)

;; Initialization

(deffacts initial-blocks
  (block (color red) (size 10))
  (block (color yellow) (size 20))
  (block (color blue) (size 30))
)

;; Rules

(defrule begin
  (initial-fact)
  =>
  (assert (goal (task find)))
)
```

```
;; pick up the largest block on the heap
(defrule pick-up
  ?my-goal <- (goal (task find))
  ?my-block <- (block (size ?S1) (place heap))
  (not (block (color ?C2) (size ?S2&:(> ?S2 ?S1)) (place heap)))
  =>
  (modify ?my-block (place hand))
  (modify ?my-goal (task build))
   (printout t ?my-block " is in location hand "  crlf)
   (printout t "Task: build." crlf)
)

;; first block is the foundation of the tower
(defrule place-first
  ?my-goal <- (goal (task build))
  ?my-block <- (block (place hand))
  (not (block  (place tower)))
  =>
  (modify ?my-block (place tower))
  (modify ?my-goal (task find))
   (printout t ?my-block " is in location tower "  crlf)
   (printout t "Task: find." crlf)
)

;; subsequent blocks go on top
(defrule put-down
  ?my-goal <- (goal (task build))
  ?my-block <- (block (color ?C0) (place hand))
  (block (color ?C1) (place tower))
  (not (on (upper ?C2) (lower ?C1)  (place tower)))
  =>
  (modify ?my-block (place tower))
  (assert (on (upper ?C0) (lower ?C1)  (place tower)))
  (modify ?my-goal (task find))
   (printout t "adding " ?my-block " to location tower "  crlf)
   (printout t "Task: find." crlf)
)

;; stop, all blocks on tower
(defrule stop
  ?my-goal <- (goal (task find))
  (not (block  (place heap)))
  =>
  (retract ?my-goal)
   (printout t "Finished!" crlf)
)
```

40

Total Points: 100