CPE/CSC 481 - W05

Knowledge-Based Systems

Final Exam

Instructor: Franz J. Kurfess

This is the final exam for CPE/CSC 481-W05 *Knowledge-Based Systems*. It is a take-home exam, and you may use textbooks, course notes, or other material, but you must develop and formulate the text and code for your answers yourself. You are allowed to and should use a computer for the programming questions. Copy and paste the code to the exam or attach it on a separate sheet, and also submit an electronic version via `handin`. Please provide references if you use material other than the textbooks or course notes. You are not allowed to discuss the questions and answers with other students or anybody else.

If you need clarifications about questions, you can contact me via email at `fkurfess@calpoly.edu`, or see me during my office hours on Tue, 2:10-4:00 p.m.; Thu, 2:10-5:00 p.m.. The deadline for the exam is Wednesday, March 16, 2004, at 4:00 p.m.. You must submit a printed and signed copy of the exam, which you can either leave in the drop box in front of the CSC department office (room 14-254), or give it to me on Wednesday, March 16, 2004 during my office hours.

The number of points is indicated on the margin, with subtotals for the different tasks.

Student Name: **Date:**

Signature:

Task I – Multiple Choice Questions

Mark the correct answers (only one per question).

1. What is *conflict resolution* in rule-based systems? 3

   - ☐ If there are several rules that possibly match the currently active facts, one of them must be selected.
   - ☐ If there is no rule that possibly matches the currently active facts, conflicting variable bindings may be eliminated through conflict resolution.
   - ☐ 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.

2. What does *shallow reasoning* (as opposed to deep reasoning) mean? 3

   - ☐ A solution is only found "deep down" in the search tree.
   - ☐ It is a particularly efficient reasoning method based on depth-first search.
   - ☐ A causal chain of cause and effect can be established across the application of different rules.
   - ☐ The knowledge is encoded as heuristics in very few rules.

3. What is the historical background for *fuzzy logic*? 3

   - ☐ It was developed by Thomas Bayes, an eighteenth-century British clergyman.
   - ☐ It was initially developed in the 1960s by Lotfi Zadeh as an attempt to formalize common-sense reasoning
   - ☐ The initial development was an ad-hoc attempt at dealing with uncertainty in expert system.
   - ☐ It is a relatively recent development that subsumes some of the other approaches to uncertainty as special cases.

4. In which of the following approaches to uncertainty is *certainty* defined as an interval describing a range of belief? 3

   - ☐ certainty factors
   - ☐ fuzzy logic
   - ☐ Bayesian reasoning
   - ☐ Dempster-Shafer theory

5. What is the most appropriate description of *fuzzy logic* with respect to uncertainty in knowledge-based systems? 3

   - ☐ it expresses a measure of belief in a hypothesis, as supported by available evidence
   - ☐ under certain conditions, it allows the calculation of the probability of a cause given some symptoms for the cause
   - ☐ it is a mathematical theory of evidence based on intervals indicating the range of belief in a statement
   - ☐ it uses linguistic variables to describe concepts with vague values

6. Which of the following terms are examples of *fuzzy quantifiers*? 3

   - ☐ somewhat, fairly, quite, very, extremely
   - ☐ most, many, usually, often, rarely
   - ☐ tall, large, high,
   - ☐ size, height, width, depth

7. Which of the following aspects has the strongest negative aspect on the performance of a rule-based system that uses the Rete algorithm for pattern matching? 3

   - ☐ the number of rules
   - ☐ the number of facts
   - ☐ the set of facts kept in working memory changes frequently
   - ☐ the number of conditions in the left-hand side of the rules

8. What is the primary role of *pattern nodes* in the Rete network? 3

   - ☐ they are responsible for matching facts with patterns in a rule
   - ☐ they make sure that variable bindings are consistent across multiple patterns
   - ☐ they represent a successful pattern match
   - ☐ they are used to distinguish between variables with the same name in different rules

Task II – Short Questions

1. Explain the differences between *field constraints* and *conditional elements* in CLIPS. List the main instances for both, and address the conceptual issues, usage and performance aspects. You can use code samples to illustrate your arguments. 10

   - Field constraints:

   - Conditional elements:

   - Conceptual differences:

   - Usage:

   - Performance:

2. The Rete algorithm is a core element of most rule-based expert systems. Describe its basic idea, and explain why it is critical for performance. 10

   - Basic idea:

   - Performance:

3. The rule given below is not formulated very efficiently. Identify problematic aspects, and rewrite the rule to make it more efficient. 10

```
(defrule bad-rule
  (items (x ?x1) (y ?y1) (z ?z1))
  (items (x ?x2) (y ?y2) (z ?z2))
  (items (x ?x3) (y ?y3) (z ?z3))
  (test (and (or  (eq ?x3 green)
                  (eq ?x3 red))
             (eq ?z2 ?y3)
     (> ?yq ?xq)
     (< ?z1 ?x1)
     (neq ?z3 ?x1)))
  =>)
```

Problematic aspects:

More efficient formulation:

Task III – CLIPS Program: Permutations

Write a CLIPS or JESS program that generates all the permutations of a base fact. For example, `(base-fact red green blue)` should generate

```
(red green blue) (red blue green)
(green blue red) (green red blue)
(blue red green) (blue green red)
```

Complete the program fragment below by specifying rules that compute and then print the permutations. If possible, use the rule headers given, but you can also structure your program differently. Give a brief explanation, and demonstrate your program with the sample facts listed below. Submit your program in a file named `Permutations.clp` via handin to `grade480` under the directory `Final-Exam`.

a) Program to generate permutations. 11

```
(deftemplate permutation
   (multislot values)
   (multislot rest))

(deffacts initial-state ; replace base facts as appropriate
   (base-fact red green blue))

(defrule start-permuations


...


)

(defrule continue-permutations


...


)

(defrule print-permutation


...


)
```

b) Explanation of the program. Describe the main ideas behind the overall program, and the individual rules (you can also include comments in the source code for the individual rules). What are the effects of using a base fact with a large number of elements, in particular on computation time and space requirements? 5

-
-
-

c) Test run. Execute your program for the test cases below, and describe relevant aspects such as the number of solutions generated, the number of facts asserted and retracted, and the number of rules on the agenda. Copy and paste the output, or append a printout.

(a) `(base-fact red green blue yellow)`

(b) `(base-fact 1 2 3 4 5 6)` 5

Subtotal Task 3: 21

Task IV – CLIPS Program: Gizmos and Chassis

Your task here is to complete a CLIPS/JESS program that checks the validity of a system configuration. It makes sure that a chassis has enough bays and power to accommodate a number of gizmos with different characteristics. It should show the essential information about the chassis and gizmos, and print out warnings if the configuration is not viable. A configuration is viable if all of the following constraints are satisfied:

- the number of gizmos is less than or equal to the number of bays in the chassis,
- the total power consumed by the gizmos is less than or equal to the power provided in the chassis,
- the overall cost of the system is less than or equal to the cost limit.

Below is a partial program. You need to complete it and answer the questions listed after the program.

```
(deftemplate chassis
   (slot name)
   (slot gizmo-bays)
   (slot power-provided)
   (slot price))

(deftemplate gizmo
   (slot name)
   (slot power-used)
   (slot price))

(deffacts chassis-list
   (chassis (name C100)
            (gizmo-bays 1)
            (power-provided 4)
            (price 2000.00))
   (chassis (name C200)
            (gizmo-bays 2)
            (power-provided 5)
            (price 2500.00))
   (chassis (name C300)
            (gizmo-bays 3)
            (power-provided 7)
            (price 3000.00))
   (chassis (name C400)
            (gizmo-bays 2)
            (power-provided 8)
            (price 3000.00))
   (chassis (name C500)
            (gizmo-bays 4)
            (power-provided 9)
            (price 3500.00)))

(deffacts gizmo-list
   (gizmo (name Zaptron)
```

```
       (power-used 2)
       (price 100.00))
 (gizmo (name Yatmizer)
       (power-used 6)
       (price 800.00))
 (gizmo (name Phenerator)
       (power-used 1)
       (price 300.00))
 (gizmo (name Malcifier)
       (power-used 3)
       (price 200.00))
 (gizmo (name Zeta-shield)
       (power-used 4)
       (price 150.00))
 (gizmo (name Warnosynchronizer)
       (power-used 2)
       (price 50.00))
 (gizmo (name Dynoseparator)
       (power-used 3)
       (price 400.00)))
```

a) *Basic Program*: Complete the above program with the following functionality: 15

- print out the selected chassis and its essential information;
- print out the current gizmo;
- if the configuration is viable, print out the information about the selected chassis and gizmos;
- if it is not viable, print out the violated constraint(s) such as number of bays, power or overall cost.

Submit the program via `handin` as `Gizmos-Basic.clp`

b) *Explanation of the main elements*: Explain how your program works. You can provide additional details as comments in the code. If you do that, please indicate it in your answer. Submit this program via `handin` as `Gizmos-Single-Constraint.clp` 5

c) *Best partial configuration, multiple constraint*: For this part, your program should be able to deal with multiple unsatisfied constraints. One problem here is to balance violations of such constraints. You can select your own strategy to resolve this, but please explain it. Submit this program via `handin` as `Gizmos-Multiple-Constraints.clp` 5

A sample output for a configuration and some test cases are given here. I may test your programs against additional configurations.

```
CLIPS> (run)
Chassis selected is C100 with a cost of 2000.0,
 1 bays and 4 units of power.
Current gizmo is Yatmizer.
==============================
Total number of gizmos is 1
Total power required is 6
Total price is 2800.0
==============================
The selected gizmos require 6 power,
but chassis C100 only provides 4 power.
CLIPS>

; Test cases
;
 (deffacts initial-state
   (chassis-selected C100) ; bays 1  power 4  price 2000
   (gizmos-selected Phenerator Dynoseparator))  ;  power 1+3 price 300+400

; (deffacts initial-state
;   (chassis-selected C100) ; bays 1  power 4  price 2000
;   (gizmos-selected Yatmizer))   ;  power 6 price 800

; (deffacts initial-state
;   (chassis-selected C200) ; bays 2  power 5  price 2500
;   (gizmos-selected Phenerator Dynoseparator))   ;  power 1+3 price 300+400

; (deffacts initial-state
;   (chassis-selected C200) ; bays 2  power 5  price 2500
;   (gizmos-selected Zeta-shield Dynoseparator))   ;  power 4+3 price 150+400

; (deffacts initial-state
;   (chassis-selected C300) ; bays 3  power 7 price 3500
;   (gizmos-selected Zaptron Yatmizer Malcifier))   ;  power 2+6+3 price 100+800+200

; (deffacts initial-state
;   (chassis-selected C400) ; bays 2  power 8 price 3000
;   (gizmos-selected Zaptron Yatmizer Malcifier))   ;  power 2+6+3 price 100+800+200

; (deffacts initial-state
;   (chassis-selected C500) ; bays 4  power 9 price 3500
;   (gizmos-selected Zaptron Malcifier Zeta-shield))   ; power 2+3+4 price 100+200+150

; (deffacts initial-state
;   (chassis-selected C500) ; bays 4  power 9 price 3500
;   (gizmos-selected Zaptron Yatmizer Malcifier))   ;  power 2+6+3 price 100+800+200
```

Subtotal Task 4: 25

Total Points: 100