Knowledge-based systems for bioinformatics – 1MB602

Professor Jan Komorowski

Exam 2005-11-23

The exam consists of three parts: Programming in Scheme, Programming in Prolog, and Foundations of Knowledge Based Systems.

Grading:

- 3: 50p – 69p
- 4: 70p – 84p
- 5: 85p – 100p.

Good luck!

PART I – SCHEME (25p total)

TASK 1 (5p): Consider the procedure `foo` below:

```
(define (foo op lst)

  (define (fi op lst)
    (op (car lst)
        (fee op (cdr lst))))

  (define (fee op lst)
    (if (null? lst)
        ()
        (op (car lst)
            (fee op (cdr lst)))))

  (fi (lambda (x y) (cons (+ 1 x) y)) (fee op lst)))
```

What are the results of the following two calls?

```
1. (foo (lambda (x y) (cons (* x x) y)) (list 1 2 3 4))

2. (foo (lambda (x y)
          (if (null? y)
              (list x)
              (append (cons x (list (car y))) (cdr y))))
        (list 1 2 3 4 5 6))
```

TASK 2 (5p):
a) (3p) Define the procedure `my-reverse` using the, in the course discussed, `accumulate` procedure given below:

```
(define (accumulate op init lst)
     (if (null? lst)
         init
         (op (car lst)
             (accumulate op init (cdr lst)))))
```

`my-reverse` takes a list as an argument and returns the list reversed as a result, e.g.:

```
(my-reverse (list 1 2 3 4 5))
;Value 1: (5 4 3 2 1)
(my-reverse (list (list 1 2) 3 (list 4 5))
;Value 2: ((4 5) 3 (1 2))
```

b) (2p) Modify `my-reverse` (call the new procedure `my-reverse2`) so that it also reverses all sub-lists of the list (still using the `accumulate` procedure), e.g.:

```
(my-reverse2 (list (list 1 2) 3 (list 4 5)))
;Value 3: ((5 4) 3 (2 1))
(my-reverse2 (list (list 1 (list 2 4 5)) 3 4 5))
;Value 4: (5 4 3 ((5 4 2) 1))
(my-reverse2 (list 1 2 3 4 5))
;Value 5: (5 4 3 2 1)
```

TASK 3 (15p): Define a **max-priority queue** using **message-driven programming**, i.e. with a dispatch procedure and local state. A priority queue is a data structure for maintaining a set *S* of elements, each associated with a value called a key. A max-priority queue supports the following three operations:

- INSERT(*S*,x,k): insert the element *x* with key *k* into the set *S*.
- MAXIMUM(*S*): returns the element of *S* with the largest key.
- EXTRACT-MAX(*S*): removes and returns the element of *S* with the largest key.

Your code should support the following scenario:

```
(define S (make-max-priority-queue))
(insert S ‘a 3)
(insert S ‘b 4)
(insert S ‘c 2)
(insert S ‘d 1)
(maximum S)
;Value 3: b
(extract-max S)
;Value 4: b
(maximum S)
;Value 5: a
```

PART II – PROLOG (20p total)

TASK 4 (10p):
a) (5p) Implement the predicate `select(Element, List, ListWithoutElement)` which is true if `Element` belongs to `List`, and `ListWithoutElement` is the `List` but where the `Element` has been removed, e.g.:

```
?- select(X,[a,b,c],Rest).

X = a
Rest = [b, c] ;

X = b
Rest = [a, c] ;

X = c
Rest = [a, b] ;

No
```

b) (5p) Implement the predicate `permutation(List, List2)` which is true if `List2` is a permutation of `List`, e.g.:

```
?- permutation([a,b,c],P).

P = [a, b, c] ;

P = [a, c, b] ;

P = [b, a, c] ;

P = [b, c, a] ;

P = [c, a, b] ;

P = [c, b, a] ;

No
```

TASK 5 (10p): Given the predicate `append/3` in Prolog:

```
append([], Ys, Ys).
append([X|Xs], Ys, [X|Zs]):- append(Xs, Ys, Zs).
```

a) (5p) Give the SLD-tree for the following query:
`?- append(X, Y, [jacke,robin,jan]).`

b) (5p) Modify the predicate using a cut, so that only one answer is given to the query above. Show how the SLD-tree is modified after adding the cut.

PART III – FOUNDATIONS OF KNOWLEDGE BASED SYSTEMS (55p total)

TASK 6 (10p): Show that all connectives ($\neg, \wedge, \vee, \leftrightarrow$) in propositional logic can be written using only the connective $\rightarrow$ (implication) and the sentence $\perp$ (the sentence which is always false).

Hint: Truth tables.

TASK 7 (5p): Consider the following two sentences:

1. "There exists no man that does not obey his stomach's demands"
2. "Items that are boxes can be put on top other free boxes or directly on the floor"

a) (3p) Translate the two sentences into First Order Logic (FOL).

b) (2p) Translate the FOL clauses to Prolog clauses.

TASK 8 (5p): The binding site for the transcription factor MIG1 is a DNA sequence which can be described as (A|T)(A|T)N(C|G)(T|C)GGGG (note: N denotes an arbitrary nucleotide base, and (X|Y) denotes a position which can be either X or Y). Describe a Non-Deterministic Finite Automaton (NDFA), which can read a DNA sequence and accept the string if it contains any binding sites for MIG1. For example, the string

*CCCACATGCATATACTTTTAACCA**ATCGCGGGG**GGGGTATTACGTATV*

has one MIG1 binding site. The task is only to describe the NDFA, not to implement it in any programming language.

TASK 9 (15p): In Natural Language Processing, a grammar can be represented as a set of production rules. Define a grammar for recognizing numbers written in English. The grammar should be able to recognize all numbers between (and including) 0-999:

zero,
one,
two,
three,
four,
.
.
.
nine hundred and ninety eight,
nine hundred and ninety nine

Note that the trivial solution, with 1000 different rules will not give you any points!

TASK 9 (20p):

a) (5p) Describe three ways to improve the Hill Climbing algorithm to avoid local optima.

b) (3p) Discuss the pros and cons of Breadth First Search (BFS) and Depth First Search (DFS)? When would you prefer one over the other?

c) (2p) What is meant by an admissible heuristics?

d) (10p) Consider the following graph where the numbers by the edges define the cost of following that edge and the numbers in parentheses in the nodes give the estimated cost of going from that node to the goal. The initial state is S and the goal state is G. Using Depth First Search (DFS), Breadth First Search (BFS), Iterative Deepening Search (IDS) using DFS, and the A* Search algorithms show which path every algorithm finds and in which order the nodes are visited.