Exam
TDDA23 - Artificial Intelligence and Lisp
15 August 2005

Points: The exam consists of eight exercises where a full and correct solution is worth between two and four points for each of them. The maximum possible number of points is 25 points, and the requirement for passing the exam is 11 points.

The solutions for exercises 3-8 are defined according to the course lecture notes that have been posted on the course webpage since the end of 2004.

Auxiliary items: English dictionary (Tillåtna hjälpmedel: Engelsk-svenskt och svensk-engelskt lexikon)

Directions: Each solution for an exercise shall be on a separate sheet of paper, with your name and exercise number on it in the upper RIGHT corner; not in the upper left corner (because your solutions will be stapled together there). Write only on one side of the paper. Answers can be written in Swedish or English.

Jour: Erik Sandewall, mobiltelefon 0708 - 28 14 08.

1. In one application one wishes to represent sets of integers as lists where the integers are sorted in the order of increasing value, and the same integer can not occur more than once. Define the functions union-x and isect-x that calculate the union and the intersection, respectively, of two sets that are represented in this way, and where the values are also represented as lists of the members in increasing order. (3 points)

2.a Define the Lisp function define-fun which can be used for defining functions, like defun, but where define-fun is an ordinary function that evaluates its arguments. For example, (define-fun 'foo '(a b) '(list a b a)) has the same effect and the same value as (defun foo (a b)(list a b a)). One may assume that define-fun always has three arguments, not more than three. (1 point)

2.b Use define-fun for defining a memoization operator called memoizze, where (memoize 'foo 'foomem) defines foomem as a memoized counterpart of foo, when foo is a function of one argument that is defined separately. A memoized function is a function that 'learns' argument-value combinations, in the sense that it keeps a list of argument-value pairs stored somewhere, and each time it is called with an argument it first checks whether the corresponding value has been stored. If so, it uses it, otherwise it calls the counterpart (in this case: foo), obtains the value, accumulats the new argument-value pair to the store, and returns the value. At any later time that the memoized function is called with this argument it reuses the value that has already been calculated and stored.

It may be assumed that the arguments of foo and foomem are symbols. The value may be any valid construct in Lisp (symbol, list, string, number, etc).

Write a definition of memoize, and explain also in words how the stored argument-value pairs are represented. (3 points)

3.a Define what it means for a plan to be consistent.
3.b Define a linearization of a plan.
3.c Is it true that a linearization of a consistent plan is always consistent? Give a reason or a counterexample.

(1 point for each of a, b, c).

4. The Yale shooting example is often used to illustrate a problem that can arise when reasoning about actions.

4.a Explain the example in words (1 point) or using formally expressed states (2 points)

4.b Explain what is the general problem that the Yale shooting problem illustrates. (2 points)

5. What things (phenomena) are expressed in second-order logic and in modal logic? Explain each of them in general terms and with an example. (3 points)

6. A person with light skin may obtain darker skin color, i.e. may be tanned by several means, such as exposure to the sun, exposure to a UV lamp, and application of a bronzing liquid. This can be expressed in a logic of actions and change using the following axioms:

H(t, lightskin(p)) & D(t, t+1, insun(p)) -> H(t+1, tanned(p))
H(t, lightskin(p)) & D(t, t+1, expose-uv-lamp(p)) ->H(t+1, tanned(p))
H(t, lightskin(p)) & D(t, t+1, apply-bronzing-liquid(p)) ->H(t+1, tanned(p))

6.a Write the reverse frame axiom(s) that correspond to these three action laws. (2 points)

6.b The reverse frame axiom(s) say something more besides what the given axioms say. What? (1 point)

7. Express one of the three action laws from example 6 in the situation calculus. (2 points). (No extra points for doing more than one of the axioms).

8. Describe an architecture with buffered output for a robotic dialogue system. Draw a diagram showing the successive steps for input to the system and output from it, and explain the use of prototext and the buffer. (3 points).

Solutions

1.
```
(defun union-x (a b)(cond
    ((null a) b)
    ((null b) a)
    ((equal (car a)(car b))
     (cons (car a)(union-x (cdr a)(cdr b))) )
    ((< (car a)(car b))
     (cons (car a)(union-x (cdr a) b)) )
    (t (cons (car b)(union-x a (cdr b)))) ))

(defun isect-x (a b)(cond
    ((null a) nil)
    ((null b) nil)
    ((equal (car a)(car b))
     (cons (car a)(isect-x (cdr a)(cdr b))) )
    ((< (car a)(car b))
     (isect-x (cdr a) b) )
    (t (isect-x a (cdr b))) ))
```

2.a.
```
(defun define-fun (f a b)(eval (list 'defun f a b)))
```

b.
```
(defun memoize (f fm)
  (define-fun fm '(x) #'(lambda (y)
      (let ((j (get y f)))
          (if j (car j)
              (car (setf (get y f)
                         (list (funcall f y))
                   ))) ))))
```

Representation: if the value of (f x) is a then (a) is kept as the value of (get x f). This representation is chosen in order to distinguish the case where the value is nil from the case where no value is known.