Programming Languages I:G221.2110
Solution to HW 4

B. Mishra

October 23, 1995

Disclaimer *Beware, the programs shown here have not been compiled, debugged, and run. So there might be errors. Nonetheless, you should get the general idea of how the solutions go.*

Problem 1. The program POWERSET

```
(DEFUN POWERSET (L)
   (COND ((NULL L) '(()))
   (T (LET ((PS (POWERSET (CDR L))))
      (APPEND PS
         (MAPCAR '#(LAMBDA (S) (CONS (CAR L) S)) PS))))))
```

The power set of a set is a set of all subsets of the original set. Observe that we can partition a power set into two classes, where the first class contains all the elements of the power set such that each element is a subset of the underlying set that does not contain the first element of the underlying set; the second class contains the elements of the power set containing the first element of the underlying set. Notice that these two conditions are mutually exclusive and exhaust the power set. Moreover, the first class is the power set of the underlying set with the first element removed.

The program POEWERSET makes use of these facts. It return a set containing the empty set if the underlying set is empty; otherwise, it builds the power set by taking the union of these two classes: The first class is the power set of "(CDR L)," and the second class is the class obtained by inserting "(CAR L)" into each subset of the power set of "(CDR L)."

Problem 2. The program ANCESTOR

```
(DEFUN ANCESTORS (X)
   (COND ((NULL X) NIL)
         (T (CONS (LIST X 0)
                  (MAPCAR '#(LAMBDA (S) (LIST (CAR S) (PLUS 1 (CADR S))))
                          (APPEND
                           (ANCESTOR (GET X 'FATHER))
                           (ANCESTOR (GET X 'MOTHER))))))))
```

This is a simple recursive program that finds ancestors for both parents, increments the distance by one for each of the ancestors and inserts a list containing X and 0 (X's distance from itself) in front.

(b) The program INSTALL-CHILDREN

```
(DEFUN INSTALL-CHILDREN (C)
   (COND ((NULL C) T)
         (T (LET ((F (GET C 'FATHER))
                  (M (GET C 'MOTHER)))
              (UPDATE-PARENT C F)
              (UPDATE-PARENT C M)
             T))))
```

```
(DEFUN UPDATE-PARENT (CHILD PARENT)
   (COND ((NOT (NULL PARENT))
          (LET ((P-CHILDREN (GET PARENT 'CHILDREN)))
             (COND ((NOT (MEMBER CHILD P-CHILDREN))
                    (PUTPROP PARENT (CONS CHILD P-CHILDREN) 'CHILDREN))))
          (INSTALL-CHILDREN PARENT))))
```

The program `INSTALL-CHILDREN` makes change to the database starting with the member `C`. If `C` is non-null then, `INSTALL-CHILDREN` updates the database appropriately by applying the auxiliary function `UPDATE-PARENT` on both of the parents.

The program `UPDATE-PARENT` on the other hand, takes a child and a parent and if the parent is not `NIL`, then adds the child to the children property of the parent (subject to the condition that child does not appear already) and calls `INSTALL-CHILDREN` on the parent.

(c) The program `COUSIN-TEST`

```
(DEFUN ANCESTOR-INTERSECTION (L1 L2)
   (COND ((NULL L1) NIL)
         ((NULL L2) NIL)
         (T (LET ((PAIR (ASSOC (CAAR L1) L2)))
              (COND ((NULL PAIR) (ANCESTOR-INTERSECTION (CDR L1) L2))
                    (T (CONS (LIST (CAAR L1) (CADAR L1) (CADR PAIR))
                             (ANCESTOR-INTERSECTION (CDR L1) L2))))))))

(DEFUN MOST-RECENT-COMMON-ANCESTOR (L)
   (COND ((NULL L) NIL)
         ((NULL (CDR L)) (CAR L))
         (T (LET ((S1 (CAR L))
                  (S2 (MOST-RECENT-COMMON-ANCESTOR (CDR L))))
               (COND ((GREATERP (PLUS (CADR S1) (CADDR S1))
                                (PLUS (CADR S2) (CADDR S2)))
                      S2)
                     (T S1))))))

(DEFUN COUSIN-TEST (A1 A2)
   (LET ((ANCESTOR (MOST-RECENT-COMMON-ANCESTOR
                    (ANCESTOR-INTERSECTION
                      (ANCESTOR A1)
                      (ANCESTOR A2)))))
     (COND ((NULL ANCESTOR) NIL)
           (T (LIST
               (DIFFERENCE
                (MIN (CADR ANCESTOR) (CADDR ANCESTOR))
                1)
               (ABS (DIFFERENCE (CADR ANCESTOR) (CADDR ANCESTOR))))))))
```

The program `COUSIN-TEST` tests if its two arguments `A1` and `A2` are cousins, and if they are, it calculates their *cousin number* and *removedness number* as defined.

In order to do this, it makes use of two auxiliary functions:

1. `ANCESTOR-INTERSECTION`, which for each person that is an ancestor of both given persons (`L1` and `L2`), adds two distances to that common ancestor, and returns a list of all such common ancestors.

2. `MOST-RECENT-COMMON-ANCESTOR`, which chooses the *most common ancestor* out of all the common ancestors given by the `ANCESTOR-INTERSECTION`. For this it uses the criterion that the common ancestor with the least combined distance to its descendants is the most recent one.