The missionaries and cannibals microplanner example

```
* ; The Missionaries and Cannibals Problem
* ;
* ;The problem is that three missionaries and three cannibals are on the left bank of a river. None of them can swim, but
* ; they have a boat that holds at most two people although it could be rowed by one person alone. The missionaries
* ; and cannibals want to get across the river together, but if the cannibals ever outnumber the missionaries on either
* ; bank, their old habits will overcome them, and they will eat the missionaries. How can all six people get across the
* ; river safely?
* ;
* ;This is done as a state space search problem in Microplanner. To represent a state we use an assertion of the form
* ; (MC (i j) side) where side is either RIGHT or LEFT and the assertion represents the state in which there are i
* ; missionaries and j cannibals on the side-th bank of the river and the boat is also on the side-th bank. The start state
* ; will be (MC (3 3) LEFT), and the goal state will be (MC (3 3) RIGHT). The operators will be boat crews. Note that each
* ; operator is symmetric - if an operator will take state si into state sj, it will also take state sj into state si. We rely on
* ; this symmetry in the program. The global MAX holds the maximum number of people in each group.
* ;
* ;
* (THINIT)
>      MICRO-PLANNER
>      >>>
>      TOP LEVEL
* (SETQ MAX 3)
>      3
* ;
* ; MC is a list of the number of missionaries and the number of cannibals on one bank. This function returns NIL if
* ; either bank has at least one missionary and more cannibals than missionaries. Otherwise it returns T.
* ;
* (DEFPROP SAFE EXPR
*    (LAMBDA (MC)
*       (OR (ZEROP (CAR MC)) (EQ (CAR MC) MAX) (EQ (CAR MC) (CADR MC)))))
>   SAFE
* ;
* ; X and Y are each lists of two numbers. This function returns T if neither element of X is greater than the respective
* ; element of Y.
* ;
* ;
* (DEFPROP LES EXPR
*    (LAMBDA (X Y)
*       (AND (NOT (GREATERP (CAR X) (CAR Y)))
*            (NOT (GREATERP (CADR X)  (CADR Y))))))
>   LES
* ;
* ;
* ; If X is the number of missionaries and cannibals on one bank and DX is the number of missionaries and cannibals
* ; that leave that bank, MOVE returns the number of missionaries and cannibals that result on the other bank.
* ;
* (DEFPROP MOVE EXPR
*    (LAMBDA (X DX)
*       (LIST (PLUS MAX (MINUS (CAR X)) (CAR DX))
*             (PLUS MAX (MINUS (CADR X)) (CADR DX)))))
>   MOVE
* ;
* ;
* ; If SIDE is the name of one bank, OTHER returns the name of the other bank.
* ;
* (DEFPROP OTHER EXPR
*    (LAMBDA (SIDE)
*       (COND ((EQ SIDE 'LEFT) 'RIGHT)
*             (T 'LEFT))))
>   OTHER
```

```
* ;
* ; Prints a report of a state and the operator that is applied to that state.
* ;
* (DEFPROP REPORT EXPR
*    (LAMBDA (MC DMDC B)
*       (PRINT (LIST 'HAVE (CAR (MOVE MC DMDC)) 'MISSIONARIES
*                    (CADR (MOVE MC DMDC)) 'CANNIBALS 'AND 'BOAT
*                    'ON (OTHER B)  'BANK))
*       (PRINT (LIST (CAR DMDC) 'MISSIONARIES 'AND
*                    (CADR DMDC) 'CANNIBALS 'CROSS))))
>   REPORT
* ;
* ;
* (THASSERT (CREW (0 1)))
>   ((CREW (0 1)))
* (THASSERT (CREW (0 2)))
>   ((CREW (0 2)))
* (THASSERT (CREW (1 0)))
>   ((CREW (1 0)))
* (THASSERT (CREW (2 0)))
>   ((CREW (2 0)))
* (THASSERT (CREW (1 1)))
>   ((CREW (1 1)))
* (THASSERT (MC (3 3) LEFT))
>   ((MC (3 3) LEFT))
* ;
* ;
* ; CROSS is a consequent theorem that finds a way to get (CAR $?MC) missionaries, (CADR $?MC) cannibals, and the
* ; boat to the $?B bank of the river.
* ;
* ; CROSS carries out the following operations:
* ;  1. Assert the state being looked for to prevent an infinite recursive loop.
* ;  2. Make sure the state being looked for is safe (no eating).
* ;  3. Find an inverse operator (a crew). Remember, each operator is its own inverse.
* ;  4. Make sure this inverse operator applies.
* ;  5. Find a way to get to the resulting state.
* ;  6. Use the automatic popping of the recursive stack to print the solution branch from start state to goal state.
* ;  7. Erase what was asserted in step 1, so that no garbage is left over.
* ;
* ; Note that $E in CROSS returns the Microplanner value of the next S-expression, i.e., $E (CAR '(A B C)) would be
* ; the Microplanner atom A.
* ;
* (THCONSE CROSS (MC B DMDC) (MC $?MC $?B)
*    (THASSERT (LOOK $?MC $?B))
*    (SAFE $?MC)
*    (THGOAL (CREW $?DMDC))
*    (LES $?DMDC $?MC)
*    (THGOAL (MC $E (MOVE $?MC $?DMDC) $E (OTHER $?B)) (THUSE CROSS))
*    (REPORT $?MC $?DMDC $?B)
*    (THERASE (LOOK $?MC $?B)))
>  (CROSS DEFINED AND ASSERTED)
>   CROSS
*
* (THDUMP)
>   (THDATA)
>   ((MC (3 3) LEFT))
>   (CROSS)
>   ((CREW (1 1)))
>   ((CREW (2 0)))
>   ((CREW (1 0)))
>   ((CREW (0 2)))
>   ((CREW (0 1)))
>   NIL
>   NIL
```

*****
*** (THGOAL (MC (3 3) RIGHT) (THUSE CROSS))**
> (HAVE 3 MISSIONARIES 3 CANNIBALS AND BOAT ON LEFT BANK)
> (1 MISSIONARIES AND 1 CANNIBALS CROSS)
> (HAVE 1 MISSIONARIES 1 CANNIBALS AND BOAT ON RIGHT BANK)
> (1 MISSIONARIES AND 0 CANNIBALS CROSS)
> (HAVE 3 MISSIONARIES 2 CANNIBALS AND BOAT ON LEFT BANK)
> (0 MISSIONARIES AND 2 CANNIBALS CROSS)
> (HAVE 0 MISSIONARIES 3 CANNIBALS AND BOAT ON RIGHT BANK)
> (0 MISSIONARIES AND 1 CANNIBALS CROSS)
> (HAVE 3 MISSIONARIES 1 CANNIBALS AND BOAT ON LEFT BANK)
> (2 MISSIONARIES AND 0 CANNIBALS CROSS)
> (HAVE 2 MISSIONARIES 2 CANNIBALS AND BOAT ON RIGHT BANK)
> (1 MISSIONARIES AND 1 CANNIBALS CROSS)
> (HAVE 2 MISSIONARIES 2 CANNIBALS AND BOAT ON LEFT BANK)
> (2 MISSIONARIES AND 0 CANNIBALS CROSS)
> (HAVE 3 MISSIONARIES 1 CANNIBALS AND BOAT ON RIGHT BANK)
> (0 MISSIONARIES AND 1 CANNIBALS CROSS)
> (HAVE 0 MISSIONARIES 3 CANNIBALS AND BOAT ON LEFT BANK)
> (0 MISSIONARIES AND 2 CANNIBALS CROSS)
> (HAVE 3 MISSIONARIES 2 CANNIBALS AND BOAT ON RIGHT BANK)
> (1 MISSIONARIES AND 0 CANNIBALS CROSS)
> (HAVE 1 MISSIONARIES 1 CANNIBALS AND BOAT ON LEFT BANK)
> (1 MISSIONARIES AND 1 CANNIBALS CROSS)
> (MC (3 3) RIGHT)
*****
*** (THDUMP)**
> (THDATA)
> ((MC (3 3) LEFT))
> (CROSS)
> ((CREW (1 1)))
> ((CREW (2 0)))
> ((CREW (1 0)))
> ((CREW (0 2)))
> ((CREW (0 1)))
> NIL
> NIL
0 (MTS)