

APPENDIX F. TOWERS OF HANOI BENCHMARK

F.1 Towers of Hanoi in CLIPS

```
; Towers of Hanoi in CLIPS

(defrule move-disk
  ?old-list <- (goals disk ?number ?from ?to $?rest)
=>
  (retract ?old-list)
;   (printout t "move disk from " ?from " to " ?to crlf)
  (assert (goals ?rest)))

(defrule move-single-disk-tower
  ?old-list <- (goals tower 1 ?from ?to $?rest)
=>
  (retract ?old-list)
  (assert (goals disk 1 ?from ?to ?rest)))

(defrule move-tower
  ?old-list <- (goals tower ?number&~1 ?from ?to $?rest)
=>
  (retract ?old-list)
  (assert (goals tower =(- ?number 1) ?from =(- 7 ?from ?to)
            tower 1 ?from ?to
            tower =(- ?number 1) =(- 7 ?from ?to) ?to ?rest)))

(defrule completed
  ?old-list <- (goals done)
=>
  (retract ?old-list))
```

F.2 Towers of Hanoi in TEXMEX 3

```
( ===== TOWERS OF HANOI IN TEXMEX3 ===== )
(          ADAPTED FOR DISTRIBUTED INFERENCING   5 DEC 95          )
( translated from Towers of Hanoi in FORPS,                    )
( by Christopher J. Matheus,                                    )
( in "The Internals of FORPS: A FORth-based Production System" )
( Journal of Forth Application and Research, Vol. 4, No. 1      )

1 constant GoalKeeper          \ keeper of the goals
2 constant DiskExpert          \ knows how to move disks
3 constant TowerExpert         \ knows how to move towers

      1 CONSTANT HOME          2 CONSTANT GOAL      ( tower numbers)
      0 CONSTANT TOWER        $80 CONSTANT DISK    ( goals)
VARIABLE SOURCE      VARIABLE TARGET      VARIABLE SPARE
VARIABLE #DISKS     VARIABLE DISK?       VARIABLE STARTED

: SPARE! ( - ) SOURCE @ TARGET @ OR 7 XOR SPARE ! ;

( Constants, variables, and GOALSTACK words, from FORPS.)
( Modified to change cell size from 4 to 2 bytes.)
( Modified again to pack goal into a single cell.)
(   f f f f t t t t d n n n n n n n          )
(   -from-- --to---   ---#disks----   d=1 if disk, 0 if tower)

10 CONSTANT MAX-#DISKS
CREATE GOALSTACK MAX-#DISKS 1- DUP * 2 + CELLS ALLOT
VARIABLE GS.PTR GOALSTACK GS.PTR !

: >GSTACK ( n a - a ) DUP ROT SWAP ! 2 + ;
: GSTACK> ( a - a n ) 2 - DUP @ ;
: >GOAL ( goal -- )
  GS.PTR @ ! 2 GS.PTR +! ;
: GOAL> ( -- goal )
  -2 GS.PTR +! GS.PTR @ @ ;
: @GOAL GS.PTR @ 2- @ ;
variable curgoal          \ to signal when problem is solved
: .goals
  cr gs.ptr @ goalstack do i @ 5 u.r 2 +loop ;

: PACKGOAL ( action #disks from to - goal time )
  $100 * SWAP $1000 * + + + \ pack into one cell
  FOREVER ;
: UNPACKGOAL ( goal -- )
  DUP $80 AND DISK? !
```

```

DUP $7F AND #DISKS !
FLIP $FF AND $10 /MOD SOURCE ! TARGET ! SPARE! ;

: maingoal ( #disks -- )
  #disks !
  goalstack gs.ptr !
  0 >goal
  tower #disks @ home goal packgoal drop >goal ;

\ : ADDGOAL ( action #disks from to - )
\   GS.PTR @ >GSTACK >GSTACK >GSTACK >GSTACK GS.PTR !
\ : REMOVEGOAL ( - ) GS.PTR @ GSTACK> DROP GSTACK> #DISKS !
\   GSTACK> SOURCE ! GSTACK> TARGET ! SPARE! GS.PTR ! ;

: .PEG ( n ) DUP 1 = IF ." A" DROP ELSE 2 = IF ." B" ELSE
  ." C" THEN THEN ;

\ Knowledge Base =====
( The logical expression IS-GOAL must be converted to a rule)
( for TEXMEX to deduce its consequences. )
( The "truth value" returned by IS-GOAL is the current goal,)
( rather than a boolean flag. )
( IS-#-OF-DISKS is similarly modified. )

\ Rules owned by Goalkeeper.
FACT ADDGOAL \ These three words have propound actions
  ADDGOAL EVALUATOR-IS (import)
  Goalkeeper ADDGOAL ]Rule .Owner !
FACT REMOVEGOAL \ which manage the goal stack.
  REMOVEGOAL EVALUATOR-IS (import)
  Goalkeeper REMOVEGOAL ]Rule .Owner !
FACT NEWGOAL \ The TELL message causes the action to happen.
  NEWGOAL EVALUATOR-IS (import)
  Goalkeeper NEWGOAL ]Rule .Owner !
FACT CURRENT-GOAL \ returns a cell value, the current goal.
  CURRENT-GOAL EVALUATOR-IS (import)
  Goalkeeper CURRENT-GOAL ]Rule .Owner !

\ Rules owned by TowerExpert.
FACT MOVE-TOWER
  MOVE-TOWER EVALUATOR-IS (import)
  TowerExpert MOVE-TOWER ]Rule .Owner !
FACT MOVE-SINGLE-DISK-TOWER
  MOVE-SINGLE-DISK-TOWER EVALUATOR-IS (import)
  TowerExpert MOVE-SINGLE-DISK-TOWER ]Rule .Owner !

\ Rules owned by DiskExpert.
FACT MOVE-SINGLE-DISK
  MOVE-SINGLE-DISK EVALUATOR-IS (import)
  DiskExpert MOVE-SINGLE-DISK ]Rule .Owner !

```

```

\ (import) evaluators will be overridden below for each cpu.

( The following is a direct translation of the FORPS rules.)

: setgoals ( goal time -- )
  DROP UNPACKGOAL ;

: is-tower? ( -- flag time )
  DISK? @ 0= FOREVER ;

: 1-disk? ( -- flag time )
  #DISKS @ 1 = FOREVER ;

: many-disks? ( -- flag time )
  #DISKS @ 1 > FOREVER ;

TowerExpert ME = #IF

  : move-tower-action
    NULL REMOVEGOAL Goalkeeper TellOnce
    TOWER #DISKS @ 1- SPARE @ TARGET @ PACKGOAL
      ADDGOAL Goalkeeper TellOnce
    DISK #DISKS @ SOURCE @ TARGET @ PACKGOAL
      ADDGOAL Goalkeeper TellOnce
    TOWER #DISKS @ 1- SOURCE @ SPARE @ PACKGOAL
      ADDGOAL Goalkeeper TellOnce
    -1 FOREVER NEWGOAL Goalkeeper TellOnce
  ;

  MOVE-TOWER RULE: CURRENT-GOAL DOFORTH setgoals
    DOFORTH is-tower?
    DOFORTH many-disks?
    AND
  CONCLUDES DOFORTH move-tower-action
;RULE

  : move-1tower-action
    NULL REMOVEGOAL Goalkeeper TellOnce
    DISK 1 SOURCE @ TARGET @ PACKGOAL
      ADDGOAL Goalkeeper TellOnce
    -1 FOREVER NEWGOAL Goalkeeper TellOnce
  ;

  MOVE-SINGLE-DISK-TOWER RULE: CURRENT-GOAL DOFORTH setgoals
    DOFORTH is-tower?
    DOFORTH 1-disk?
    AND
  CONCLUDES DOFORTH move-1tower-action
;RULE

```

```

#THEN

DiskExpert ME = #IF

    : move-disk-action
      NULL REMOVEGOAL Goalkeeper TellOnce
      \   TARGET @ SOURCE @
      \   CR ." *** Move disk on peg " .PEG ." to peg " .PEG
      -1 FOREVER NEWGOAL Goalkeeper TellOnce
      ;

MOVE-SINGLE-DISK RULE: CURRENT-GOAL DOFORTH setgoals
                      DOFORTH is-tower? NOT
CONCLUDES DOFORTH move-disk-action
;RULE

#THEN

Goalkeeper ME = #IF

    : fetch-goal ( -- goal time) GS.PTR @ 2- @ FOREVER
      over curgoal ! ; \ to signal when problem is solved
CURRENT-GOAL RULE: NEWGOAL DOFORTH 2drop DOFORTH fetch-goal
;RULE \ ^--for propagation
0 CURRENT-GOAL ]Rule .Export ! ( changes are broadcast)

    : push-goal ( goal time -- ) DROP >GOAL ;
ADDGOAL EVALUATOR-IS NULL
ADDGOAL proponent-IS push-goal

    : pop-goal ( fact -- ) 2DROP GOAL> DROP ;
REMOVEGOAL EVALUATOR-IS NULL
REMOVEGOAL proponent-IS pop-goal

NEWGOAL RULE: -1 LITERAL NOW ;RULE ( for AND)

#THEN

\ Exercise routine =====

( Read the 18.2 Hz BIOS clock)
CODE BIOSCLK ( - d)
    MOV AH, # 0 INT # $1A PUSH DX PUSH CX NEXT
END-CODE

( The high level word is modified somewhat from FORPS.)
( The system is initiated with the command n DISKS )
VARIABLE #CYCLES 1 #CYCLES !

ALSO ITL ALSO FORTH

```

```

: DISKS ( n - )

FWD-CHAIN ON  BWD-CHAIN OFF
BIOSCLK DROP ( n time - )
#CYCLES @ 0 DO

    OVER maingoal                \ set goal = n disks
    @goal curgoal !              \ zeroed when problem solved
    -1 forever newgoal assert    \ start inferencing
    ( will propagate changes until goal=0.)

    BEGIN
        key? if abort then      \ does ?incoming & ?outgoing
        curgoal @ 0= UNTIL

    LOOP
    BIOSCLK DROP SWAP - SPACE U. ." ticks"
    DROP ;

( same, using backward chaining )

: BDISKS ( n - )  FWD-CHAIN OFF

    BIOSCLK DROP ( n time - )
    \ #CYCLES @ 0 DO

    OVER #DISKS !

    GOALSTACK GS.PTR !
    0 >GOAL
    TOWER #DISKS @ HOME GOAL PACKGOAL DROP >GOAL

    BEGIN
        MOVE-TOWER ]Rule EVALUATE
        MOVE-SINGLE-DISK-TOWER ]Rule EVALUATE
        MOVE-SINGLE-DISK ]Rule EVALUATE
    GS.PTR @ 2 - @
    0= UNTIL      ( until bottom of rule-stack reached)

    \ LOOP
    BIOSCLK DROP SWAP - SPACE U. ." ticks"
    DROP ;

PREVIOUS PREVIOUS
bwd-chain off fwd-chain on

```