December 9, 1978 5:58PM in <VANNELLE>WW.SAV/8112

1. ITOSPACE
   $INITIALIZE
   $TEST
2. ADDATTRIBUTE
   ADDOBJECT
   ANNOUNCE TASK
3. ATTRIBUTE
   AWARD
4. BID
   CHECKBIDS
5. CHECK:ELIGIBILITY
6. CILP
7. CILPARSE
8. CNET
9. DEFINE:OBJECT
   DELETE:OBJECT
10. DELETE:PSEUDO:CONTRACT
    DIRECTED:AWARD
    DISPLAY:OBJECT
11. DISPLAY:EVENT
12. DISPLAY:EVENTS:AT:TIME
    DISPLAY:MESSAGE
13. DISPLAY:NODE
14. DISPLAY:PARAMETERS
15. DISPLAY:RECORDS
16. DISPLAY:STATISTICS
17. EXTEND:BOARD
18. FINAL:REPORT
   FIND:SUBCONTRACT
   GENERATE:SUBTASK
20. GET:TASK:ANNOUNCEMENT
21. GOOD:BOARD
22. INITIALIZE
24. INITIALIZE
25. INSTALL:EVENT
26. INSTALL:INTERNAL:EVENT
27. INTERIM:REPORT
28. MAKE:BID
29. NEW:BOARD
30. NEXT:CONTRACT
31. NODE:SEARCH
32. OBJECTP
   OUTSTANDING:SUBCONTRACTS
   PARSE:NODE:ABSTRACTION
33. PARSE:TASK:ABSTRACTION
   PROCESS:ACKNOWLEDGEMENT
34. PROCESS:BID
35. PROCESS:CONTRACT
36. PROCESS:DIRECTED:AWARD
37. PROCESS:DISPLAY:EVENT
38. PROCESS:FINAL:REPORT
39. PROCESS:INFORMATION
40. PROCESS:INTERNAL:EVENT
41. PROCESS:MESSAGE
42. PROCESS:NODE:AVAILABILITY:ANNOUNCEMENT
43. PROCESS:REQUEST
44. PROCESS:TASK:ANNOUNCEMENT
45. PROCESS:TERMINATION
46. QARANK
47. QDISPLAY
48. QFINALIZE
49. QINITIALIZE
50. QRECEIVE
51. RESIMULATE
52. RESUME:OBJECT
53. SIMULATE
54. STORE:OBJECT
55. STORE:TASK:OBJECT
56. TERMINATE
57. TERMINATE:SUBCONTRACTS
58. UPDATE:ACTIVE:TASK:ANNOUNCEMENTS
59. UPDATE:OBJECT
60. UPDATE:TASK:TIME
61. VALUEP

Note: The text layer for this file was generated by OCR. Expect errors.
Fsns on CNET:

$INITIALIZE
$TEST
ADATTRIBUTE
ADOOBJECT
ANNOUNCE TASK
ATTRIBUTEP
AWARD
BID
CHECK1BIOS
CHECKELIGIBILITY
CILP
CILPARSE
CNET
CNET*
DEFINE OBJECT
DELETE OBJECT
DELETE PSEUDO CONTRACT
DIRECTED AWARD
DISPLAY CONTRACT
DISPLAY EVENT
DISPLAY EVENTS AT TIME
DISPLAY MESSAGE
DISPLAY NODE
DISPLAY PARAMETERS
DISPLAY RECORDS
DISPLAY STATISTICS
EXTEND BOARD
FINAL REPORT
FIND SUBCONTRACT
GENERATE SUBTASK
GET TASK ANNOUNCEMENT
GOOD BOARD
INITCIL
INITIALIZE
INSTALL DISPLAY EVENT
INSTALL EVENT
INSTALL INTERNAL EVENT
INTERIM REPORT
MAKE BID
NEW BOARD
NEXT CONTRACT
NEXT EVENT
NODE SEARCH
OBJECT
OUTSTANDING SUBCONTRACTS
PARSE NODE ABSTRACTION
PARSERTASK ABSTRACTION
PROCESS ACKNOWLEDGEMENT
PROCESS ANNOUNCED AWARD
PROCESS BID
PROCESS CONTRACT
PROCESS DIRECTED AWARD
PROCESS DISPLAY EVENT
PROCESS FINAL REPORT
PROCESS INFORMATION
PROCESS INTERIM REPORT
PROCESS INTERNAL EVENT
PROCESS MESSAGE
PROCESS NODE AVAILABILITY ANNOUNCEMENT
PROCESS REQUEST
PROCESS TASK ANNOUNCEMENT
PROCESS TERMINATION
QANNOUNCE
QARANK
QBRANK
QDISPLAY
QFINALIZE
QINITIALIZE
QRECEIVE
QGET PARAMETERS
QRANDOMCOMPARE
QRANDOMDISPLAY
QRANDOMCOPY
QREANNOUNCE TASK
QRELEASE TASK
QRESIMULATE
QRESUME TASK
QRETRIEVE OBJECT
QSAME STATUS CHECK
QSDISPLAY
QSEND MESSAGE
QSET PARAMETERS
QSIMULATE
QSTORE OBJECT
QSTORE TASK OBJECT
QSTORE TASK TIME
QTERMINATE
QUPDATE ACTIVE TASK ANNOUNCEMENTS
QUPDATE OBJECT
QUPDATE TASK OBJECT
QUPDATE TASK TIME
QVALUEP
rgs: 11-Oct-78 21:06 (CNET)

(ITOSPACE)

(ITOSPACE
  (LAMBDA (x)
    (MKSTRING (PACK (SUBST " " "!" (UNPACK x))))
  )
)

Called by: DISPLAY!EVENT DISPLAY!MESSAGE

Explanation: Replaces "!" with " " in atom names for cleaner output.

rgs: 10-Sep-78 08:46 (CNET)

($INITIALIZE
  (LAMBDA (xnetsize restartflag)
    (PROG NIL
      (UPDATE!TASK!TIME 1)
      (CNET* 'SDISPLAY (LIST xspecification))
      (UPDATE!TASK!TIME 1)
      (CNET* 'GENERATE!SUBTASK (LIST (LIST '$TEST "This is T1")
        (LIST '$TEST "This is T2"))
      (TERMINATE)
    )))
)

Calls: CNET* TERMINATE UPDATE!TASK!TIME

Explanation: A sample initial applications function. Such a function is called to initialize nodes in the net with applications-specific information for the simulation. The arguments are "xnetsize", the number of processor nodes in the distributed architecture, "restartflag", a flag that is T if at least one simulation has already been performed, and "olduserparamflag", a flag that can be set to T by the user during interaction with CNET if the current user parameters are to be used as defaults during acquisition of new parameters. All I/O should be done directly and required CNET functions should be used directly without going through CNETs since this function is handled in a special manner, and not through the generator structure. The initial applications function returns a list of two-element lists of the form "(type specification)", where "type" is the type of task, and "specification" is the task specification. The returned N tasks in the list are assigned as top-level contracts to the first N processor nodes. See QINITIALIZE as an example for the N Queens problem.

rgs: 11-Oct-78 21:04 (CNET)

($TEST
  (LAMBDA (xnnode xname xspecification xcontract)
    (PROG NIL
      (UPDATE!TASK!TIME 1)
      (CNET* 'SDISPLAY (LIST xspecification))
      (UPDATE!TASK!TIME 1)
      (CNET* 'GENERATE!SUBTASK (LIST (LIST '$TEST "This is subtask 1")
        (LIST '$TEST "This is subtask 2")))
      (TERMINATE))
)

Calls: CNET* TERMINATE UPDATE!TASK!TIME

Explanation: A sample task execution procedure. Such a function is called to execute a task. The arguments are "xnnode", the name of the node in which the task is being executed, "xname", the name of the contract for the task, "xspecification", the task specification, and "xcontract", the complete contract record. Such functions are implemented as generators and must access all CNET functions through CNETs (they are suspended each time a call to CNETs is made—for quasi parallelism).

Two special functions are available, SUSPEND, which moves the contract to the suspended state, and TERMINATE, which moves the contract to the terminated state.

No value is returned. See CNET*, SUSPEND, and TERMINATE for details on how they are called. See EXTEND!BOARD as an example for the N Queens problem.

-2-
ADDATTRIBUTE

LAMBDA N
  (for I from 1 to N do (COND
    ((NOT (MEMBER (U-CASE (ARG N I))
      (GETPROP '#ATTRIBUTE 'POSSIBLEVALUES))
     (ADDPROP '#ATTRIBUTE 'POSSIBLEVALUES (U-CASE (ARG N I)))))

Called by: DEFINEOBJECT

Explanation: Installs all of its arguments as 'attributes' in the common internode language. All attributes are first converted to upper case. Checks are made for duplication.

ADDOBJECT

LAMBDA N
  (for I from 1 to N do (COND
    ((NOT (MEMBER (U-CASE (ARG N I))
      (GETPROP '#OBJECT 'POSSIBLEVALUES))
     (ADDPROP '#OBJECT 'POSSIBLEVALUES (U-CASE (ARG N I)))))

Called by: DEFINEOBJECT

Explanation: Installs all of its arguments as 'objects' in the common internode language. All objects are first converted to upper case. Checks are made for duplication.

ANNOUNCE!TASK

LAMBDA (xpnode xname)
  (PROG (temp)
    (setq temp (GET!TASK!ANNOUNCEMENT xpnode xname))
    (cond
      ((equal (car temp) 'DIRECTED!AWARD)
       (DIRECTED!AWARD xpnode xname (cadr temp) (caddr temp) (cadddr temp) (car (caddddr temp))
       (create TASK!ANNOUNCEMENT NAME ← xname ELIGIBILITY!SPECIFICATION ← (cadr temp)
       TASK!ABSTRACTION ← (caddr temp)
       BID!SPECIFICATION ← (cadddr temp)
       EXPIRATION!TIME ← (car (caddddr temp))
       (INSTALL!INTERNAL!EVENT (+ time (car (caddddr temp)))
       xpnode xname 'BID!CHECK)))

Calls: DIRECTED!AWARD GET!TASK!ANNOUNCEMENT INSTALL!INTERNAL!EVENT SENDMESSAGE

Called by: GENERATE!SUBTASK REANNOUNCE!TASK

Freevars: ta time

Explanation: Sends either a directed award or a task announcement message for the contract with name "xname" that has been generated by node "xpnode". It also places a "bid!check" event on the event list to check for bids at the end of the expiration time. Uses GET!TASK!ANNOUNCEMENT to generate the task-dependent information for the directed award or task announcement.
ATTRIBUTEP

LAMBDA (xobject xattribute)
(COND
  ((OBJECTP xobject)
   (COND
     ((MEMBER xattribute (RECORDFIELDNAMES (REC LOOK xobject)))
      T)
     (T (WRITE "CIL error: " xattribute " is not a valid attribute of " xobject)
      NIL))

Calls: OBJECTP

Explanation: Returns T if "xattribute" is a valid attribute of the object "xobject". If "xobject" is not a valid object or "xattribute" is not a valid attribute of "xobject" then WRITES an error message and returns NIL.

AWARD

LAMBDA (xpnode xname xaddressee)
(PROG (sc)
  (SETQ sc (NODE!SEARCH xpnode xname 'ANNOUNCED T T))
  (SENDMESSAGE xpnode (IPLUS time tpb tsaw) xaddressee
    (create ANNOUNCED!AWARD NAME ← xname TASK!SPECIFICATION ←(fetch (TASK SPECIFICATION)
      of (RETRIEVE!OBJECT xpnode
        'TRSK
        (fetch (SUBCONTRACT TASK)
          of (CAR sc)
        (e note the name of the contractor in the subcontract record)

  (replace (SUBCONTRACT CONTRACTOR) of (CAR sc) with xaddressee))

Calls: NODE!SEARCH RETRIEVE!OBJECT SENDMESSAGE

Called by: CHECK!BIDS PROCESS!BID

Freevars: time tpb tsaw

Explanation: Sends an award message to "xaddressee" from "xpnode" for contract "xname". Updates the appropriate subcontract record.
Calls: RETRIEVE!OBJECT SENDMESSAGE
Called by: MAKE!BID
Freevars: NET tb time
Explanation: Makes a bid on contract with name "xcontract" to manager "xmanager" from node "xpnode". The bid specification "xbid!specification" and the task type "xtype" are used to access the appropriate bid construction procedure and construct the bid.

rgs: 12-Sep-78 01:06 [CNET]  
(BID  
(LAMBDA (xpnode xcontract xmanager xtype xbid!specification)  
(PROG (xpnode xbidconsproc xnode!abstraction)  
 (SETQ xpnode (ELT NET xpnode))  
 (SETQ xbidconsproc (fetch (TASK!TEMPLATE BID!CONSTRUCTION!PROCEDURE) of (RETRIEVE!OBJECT xpnode 'TASK!TEMPLATE xtype)))  
 (ICOND  
 (xbidconsproc (SETQ xnode!abstraction (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE xbidconsproc))  
 (LIST xpnode xbid!specification)  
 (SENDMESSAGE xpnode (IPLUS time tb)  
 xmanager  
 (create BID NAME ← xcontract NODE!ABSTRACTION ← xnode!abstraction))  
 (RETURN T))  
 (RETURN NIL))  
 (RETURN T))
Calls: RETRIEVE!OBJECT SENDMESSAGE
Called by: MAKE!BID
Freevars: NET tb time
Explanation: Makes a bid on contract with name "xcontract" to manager "xmanager" from node "xpnode". The bid specification "xbid!specification" and the task type "xtype" are used to access the appropriate bid construction procedure and construct the bid.

rgs: 27-Oct-78 10:50 [CNET]  
(CHECK!BIDS  
(LAMBDA (xpnode xname)  
(PROG (xpnode! sc active!bids xawproc)  
 (SETQ xpnode! (ELT NET xpnode))  
 (SETQ xbidconsproc (fetch (TASK!TEMPLATE BID!CONSTRUCTION!PROCEDURE) of (RETRIEVE!OBJECT xpnode 'TASK!TEMPLATE xtype)))  
 (COND  
 (xbidconsproc (SETQ xnode!abstraction (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE xbidconsproc))  
 (LIST xpnode xbid!specification)  
 (SENDMESSAGE xpnode (IPLUS time tb)  
 xmanager  
 (create BID NAME ← xcontract NODE!ABSTRACTION ← xnode!abstraction))  
 (RETURN T))  
 (RETURN NIL))  
 (RETURN T))
Calls: AWARD NODE!SEARCH REANNOUNCE!TASK RETRIEVE!OBJECT
Called by: PROCESS!INTERNAL!EVENT
Explanation: Checks the bids on the contract with name "xname" in node "xpnode" at the end of the expiration time. If the contract has been awarded then returns T. If the contract has not been awarded then calls the award procedure for the task. If no award procedure exists, then awards the contract to the first bidder in the active!bids list. If no bids have been received, then reannounces the contract.
(CHECK ELIGIBILITY
  (LAMBDA (xpnode elspec)
    (PROG (pelspec)
      (SETQ pelspec (for x in elspec always (CILPARSE x ELSPECGRAMMAR))))
    (RETURN pelspec)))

Calls: CILPARSE
Called by: MAKE!BID PROCESS!DIRECTED!AWARD PROCESS!TASK!ANNOUNCEMENT
Freevars: ELSPECGRAMMAR

Explanation: Checks to see if node "xpnode" meets the eligibility specification "elspec", and, if so returns T. ANDS a series of statements written according to "elspecgramar".

(CILP
  (LAMBDA (word)
    (COND
      (ISOME CILCLASSES (FUNCTION (LAMBDA (class)
          (FMEMB word (GETPROP class 'POSSIBLEVALUES)
            T)
        (T NIL))
      Freevars: CILCLASSES

rgs: 27-Oct-78 10:23

rgs: 17-Oct-78 22:04

-6-
(DEFUN CILPARSE (
  (phrase grammar)
  (PROG (match)
    (SETQ match (INTERPRET
                  phrase (COND
                            (grammar grammar)
                            (T CILGRAMMAR)))
               CILCLASSES NIL T)))

(COND
  ((AND (fetch (INTERPRETATION MATCH) of match)
         (NOT (fetch (INTERPRETATION REMAININGPHRASE) of match)))
     (RETURN match))
  ((fetch (INTERPRETATION MATCH) of match)
     (WRITE "Parse succeeded, but words remain in phrase")
     (WRITE "Results: " (fetch (INTERPRETATION RESULTS) of match))
     (WRITE "Remaining words: " (fetch (INTERPRETATION REMAININGPHRASE) of match))
     (WRITE "Bindings: " (fetch (INTERPRETATION BOUNDCLASSES) of match))
     (RETURN match))
  (T (COND
     ((fetch (INTERPRETATION RESULTS) of match)
      (e remove duplicate templates)
      (replace (INTERPRETATION RESULTS) of match with (INTERSECTION (fetch (INTERPRETATION RESULTS) of match)
                                             (fetch (INTERPRETATION RESULTS) of match)))
     (WRITE "Parse failed: " (LENGTH (fetch (INTERPRETATION RESULTS) of match))
               " template(s) partially matched")
     (for x in (fetch (INTERPRETATION RESULTS) of match)
       do (WRITE
            (WRITE "Partially matched template:
            "
            (fetch (FAILURE TEMPLATE) of x))
            (WRITE "Semantic Predicate and Action Functions:
            "
            (fetch (FAILURE FUNCTIONS) of x))
            (WRITE "Unmatched portion of template:
            "
            (fetch (FAILURE REMTEMPLATE) of x))
            (WRITE "Unmatched portion of phrase:
            "
            (fetch (FAILURE REMPHRASE) of x))
            (WRITE "Bindings:
            "
            (for y in (fetch (FAILURE FBINDINGS) of x) do (WRITE "y"))
            (COND
             ((NOT (fetch (FAILURE REMPHRASE) of x))
              (WRITE "Semantic Predicate Failed")
              (T (WRITE "Parse failed: No templates matched")
              (RETURN NIL))))

-----

Called by: CHECKELIGIBILITY

Freevars: CILCLASSES CILGRAMMAR

Explanation: Parses "phrase" using CILGRAMMAR and CILCLASSES.
CNET

(CNET
  (LAMBDA (restartflag oldcnetparamflag olduserparamflag)
    (PROG (temp)
      (TTYOUT "----- CONTRACT NET Simulation -----")
      (TERPRI)
      (TERPRI)
      (COND
        (NOT restartflag)
        (SET!PARAMETERS (NOT oldcnetparamflag)
          (SIMULATE restartflag olduserparamflag)
          (while (IGREATERP (SETQ temp (RESIMULATE)) 0)
            do (COND
              ((EQ temp 1)
                (SIMULATE T T))
              (T (SET!PARAMETERS)
                (SIMULATE NIL T))
        )))
    Calls: RESIMULATE SET!PARAMETERS SIMULATE

Explanation: The top-level function in the CNET system. Starts a contract net simulation. "restartflag" is T if new parameters are not to be requested. "oldcnetparamflag" is T if the current cnet parameters are to be used as defaults when new cnet parameters are requested. "olduserparamflag" is T if the current user parameters are to be used as defaults when new user parameters are requested.

rgs: 7-Sep-78 05:33 [CNET]  

(CNET#
  (LAMBDA (xfunction xarguments)
    (** contract net system call -
      used by a user program to access contract net system functions -
      a user task processor which is implemented as a generator through the possibilities list construct
      is suspended when such a call is made -
      to give quasi-parallelism) **

    (PROG NIL
      (COND
        ((EQ 'RELEASE (AU-REVOIR NIL))
         (ADIEU)
        ))
      (APPLY xfunction (APPEND (LIST xpnode xname) xarguments))
    ))

Called by: *TEST EXTEND!BOARD QRECEIVE
Freevars: xname xpnode

Explanation: Used by a user function to access CNET functions. This is the only mode of access to CNET functions that is to be used by the user functions that actually execute tasks. Every time CNET# is called, the calling function is suspended so as to simulate parallelism. "xfunction" is the name of the CNET function to be applied. "xarguments" is the list of arguments for the function.

DEFINE OBJECT

(LAMBDA N
  (PROG (temp)
    (SETQ temp (CONS 'TYPERECORD (CONS (U-CASE (ARG N 1))
      (LIST (FOR I FROM 2 TO N COLLECT (U-CASE (ARG N I)
      (EVAL temp)
      (ADDOBJECT (ARG N 1)))
      (APPLY 'ADDATTRIBUTE FOR I FROM 2 TO N COLLECT (ARG N I)))

Calls: ADDATTRIBUTE ADDOBJECT

Explanation: Makes a TYPERECORD declaration using the first argument as the type of record. Calls ADDOBJECT and ADDATTRIBUTE to add the type of record to the list of objects defined in the common internode language, and the rest of the arguments to the list of attributes.

DELETE OBJECT

(LAMBDA (xpnode xobject xkey)
  (PROG (xpnode! kb index otherindex otherindexl xinstance)
    (SETQ xpnode! (ELT NET xpnode))
    (SETQ kb (fetch (PNODE KNOWLEDGE!BASE) OF xpnode!))
    (COND
      [(MEMBER xobject (RECORDFIELDNAMES 'KNOWLEDGE!BASE))
        (SETQ index (RECORDACCESS xobject kb (RECLOOK 'KNOWLEDGE!BASE))
        (SETQ xinstance (CAR (SOME index (FUNCTION (LAMBDA (x)
          (EQUAL (CADR x)
            xkey))))
        (COND
          [(xinstance (SETQ index (REMOVE xinstance index)))
            (RECORDACCESS xobject kb (RECLOOK 'KNOWLEDGE!BASE)
              'replace index)
          (T (SETQ index (fetch (KNOWLEDGE!BASE OTHER) OF kb)))
          (SETQ otherindexl (CAR (SOME (CDR otherindex)
            (FUNCTION (LAMBDA (x)
              (EQUAL (CADR x)
                xobject))))
          (COND
            [(xinstance (SETQ otherindexl (REMOVE xinstance otherindexl))
              (REPLACE (KNOWLEDGE!BASE OTHER) OF kb WITH index)
            (RETURN xinstance)]
        )
    )

Called by: DELETE!PSEUDO!CONTRACT PROCESS!TERMINATION UPDATE!NODE

Freevars: NET

Explanation: Removes an object from the knowledge base of node "xpnode". "xobject" is the 'type' of object, and "xkey" is the key that specifies the object. All objects are represented as record structures and the key must be the first field in the record structure for the object to be deleted.
DELETE!PSEUDO!CONTRACT

(LAMBDA (xpnode xname)
  (PROG NIL
    (COND
      ((EQUAL (fetch (CONTRACT STATE) of (RETRIEVE!OBJECT xpnode 'CONTRACT xname)) 'PSEUDO)
        (DELETE!OBJECT xpnode 'CONTRACT xname)
        (RETURN T))
      (T (RETURN))
    ))
  )
)

Calls: DELETE!OBJECT RETRIEVE!OBJECT
Called by: PROCESS!INTERNAL!EVENT

Explanation: Removes the pseudo-contract with name "xname" from node "xpnode". If the contract was not awarded, then the pseudo-contract that was set up when a bid was made still has state 'pseudo'.

DIRECTED!AWARD

(LAMBDA (xpnode xname xaddressee xes xta xts)
  (PROG (sc)
    (SETQ sc (CAR (NODE!SEARCH xpnode xname 'ANNOUNCED T)))
    (SENDMESSAGE xpnode (IPLUS time tdaw) xaddressee
      (create DIRECTED!AWARD NAME ← xname ELIGIBILITY!SPECIFICATION ← xes TASK!ABSTRACTION ← xta TASK!SPECIFICATION ← xts))
      (note the name of the prospective contractor in the subcontract record - bound to a "e" to indicate that acknowledgement has not yet been received)
    (replace (SUBCONTRACT CONTRACTOR) of sc with (CONS xaddressee 'e))
  )
)

Calls: NODE!SEARCH SENDMESSAGE
Called by: ANNOUNCE!TASK
Freevars: tdaw time

Explanation: Sends a directed award message to "xaddressee" from "xpnode" for the contract with name "xname". Uses "xes" as eligibility specification, "xabs" as task abstraction, and "xts" as task specification.

DISPLAY!CONTRACT

(LAMBDA (c)
  (PROG NIL
    (DISPLAY "$name: " (fetch (CONTRACT NAME) of c))
    (DISPLAY "$manager: " (fetch (CONTRACT MANAGER) of c))
    (DISPLAY "$subcontract/subcontractor: " (for x in (CDR (fetch (CONTRACT SUBCONTRACTS) of c)) collect (LIST (fetch (SUBCONTRACT NAME) of x) (fetch (SUBCONTRACT CONTRACTOR) of x))))
    )
)

Explanation: Displays the name, manager, subcontract names and subcontracts for the contract record "c".
(DISPLAY!EVENT
  (LAMBDA (e forceflag)
    (PROG NIL
      (SELECTQ (CAR (fetch (EVENT DATA) of e))
        (DISPLAY!EVENT (COND
            ((OR forceflag (AND (EQ (fetch (DISPLAY!EVENT TYPE) of (fetch (EVENT DATA) of e)) 'SIMULATION) (fetch (DISPLAY!EVENT events!flag)))))
          (e two types of display!event - task and simulation - task display!events are always processed whereas simulation display!events are only processed when display!display!events!flag is set)
          (PROCESS!DISPLAY!EVENT (fetch (EVENT DATA) of e))
          (INTERNAL!EVENT (COND
              ((OR forceflag display!internal!events!flag)
                (DISPLAY)
                (DISPLAY "node: " (fetch (INTERNAL!EVENT PNODE) of (fetch (EVENT DATA) of e)))
                (DISPLAY "contract: " (fetch (INTERNAL!EVENT NAME) of (fetch (EVENT DATA) of e)))
                (DISPLAY "internal event: " (!TOSPACE (fetch (INTERNAL!EVENT TYPE) of (fetch (EVENT DATA) of e)))
                (DISPLAY)
                (MESSAGE (COND
                    ((OR forceflag display!messages!flag)
                      (DISPLAY!MESSAGE (fetch (EVENT DATA) of e)))
                    (DISPLAY)
                    (DISPLAY!MESSAGE (fetch (EVENT DATA) of e)))
                    NIL)
            )
        )
      )
    )
  )
)

Calls: !TOSPACE DISPLAY!MESSAGE PROCESS!DISPLAY!EVENT
Called by: DISPLAY!EVENTS!AT!TIME SIMULATE
Freevars: display!display!events!flag display!internal!events!flag display!messages!flag
Explanation: Displays the particulars of the event "e". If "forceflag" is T then the normal display flags are overridden, and the event is always displayed.
(DISPLAY!EVENTS!AT!TIME
(LAMBDA (t forceflag)
  (PROG (e)
    (SETQ e eventlist)
    (while e do (COND
      ((ILESSP t (fetch (EVENT TIME) of e))
        (COND
          ((fetch (EVENT LLINK) of e)
            (SETQ e (fetch (EVENT LLINK) of e)))
          (T (GO $$OUT))
          ((IGREATERP t (fetch (EVENT TIME) of e))
            (COND
              ((fetch (EVENT RLINK) of e)
                (SETQ e (fetch (EVENT RLINK) of e)))
              (T (GO $$OUT))
              (T (GO $$OUT))))))))

Calls: DISPLAY!EVENT
Freevars: eventlist
Explanation: Displays the particulars of all events scheduled for time "t". If "forceflag" is T then the normal display flags are overridden, and the events are always displayed.

(DISPLAY!MESSAGE
(LAMBDA (m)
  (PROG NIL
    (DISPLAY)
    (DISPLAY "To: " (fetch (MESSAGE ADDRESSEE) of m))
    (DISPLAY "From: " (fetch (MESSAGE ORIGINATOR) of m))
    (DISPLAY "Type: " (!TOSPACE (CAR (fetch (MESSAGE TEXT) of m)))
    (DISPLAY "Contract: " (CADR (fetch (MESSAGE TEXT) of m)))
    (DISPLAY)))

Calls: !TOSPACE
Called by: DISPLAY!EVENT
Explanation: Displays the address, originator, type, and contract name for message "m".
(DISPLAY NODE)

(LAMBDA (xpnode forceflag)
  (PROG (xpnode!)
    (SETQ xpnode! (ELT NET xpnode))
    (COND
      ((OR forceflag (EQUAL (fetch (PNODE STATUS) of xpnode!) "Busy")
        (fetch (PNODE ANNOUNCED) of xpnode!))
            (DISPLAY)
        (DISPLAY "Node " xpnode)
      [DISPLAY "Executing: " (LIST (fetch (CONTRACT NAME) of (CAR (fetch (PNODE EXECUTING) of xpnode!))
          of (CAR x))
      [DISPLAY "Announced: " (for x in (fetch (PNODE ANNOUNCED) of xpnode!) collect (fetch (SUBCONTRACT NAME) of (CAR x))
      [DISPLAY "Suspended: " (for x in (fetch (PNODE SUSPENDED) of xpnode!) collect (fetch (CONTRACT NAME) of (CAR x))
      [DISPLAY "Terminated: " (for x in (fetch (PNODE TERMINATED) of xpnode!) collect (fetch (CONTRACT NAME) of x))
    (DISPLAY))

----------
Called by: SIMULATE
Freevars: NET
Explanation: Displays the names of the contracts and subcontracts in the contract processing states of node "xpnode". If "forceflag" is T then the names are always displayed. Otherwise, they are only displayed if the node is "Busy".
DISPLAYPARAMETERS
(LAMBDA NIL
(DISPLAY)
(DISPLAY " CONTRACT NET Simulation Parameters
")
(DISPLAY " Number of Processor Nodes in Net: " netsize)
(DISPLAY " Applications time unit expansion: " gain)
(DISPLAY " Contracts held in terminated state: " ntermcs)
(DISPLAY " CONTRACT NET Delay Parameters
")
(DISPLAY " Time to make a task announcement: " ta)
(DISPLAY " Time before a task is reannounced: " tra)
(DISPLAY " Time to process a task announcement: " tpa)
(DISPLAY " Time to make a node availability announcement: " tna)
(DISPLAY " Time to process a node availability announcement: " tpna)
(DISPLAY " Time to make a bid: " tb)
(DISPLAY " Time to process a bid: " tpb)
(DISPLAY " Time to make an announced award: " tsaw)
(DISPLAY " Time to process an announced award: " tpsaw)
(DISPLAY " Time to make a directed award: " tdaw)
(DISPLAY " Time to process a directed award: " tpdaw)
(DISPLAY " Time to acknowledge a directed award: " tack)
(DISPLAY " Time to process an acknowledgement: " tpack)
(DISPLAY " Time to make a report: " tr2)
(DISPLAY " Time to process a report: " tpr)
(DISPLAY " Time to generate a termination: " tt)
(DISPLAY " Time to process a termination: " tpt)
(DISPLAY " Time to generate a request: " treq)
(DISPLAY " Time to process a request: " tpreq)
(DISPLAY " Time to generate an information message: " ti)
(DISPLAY " Time to process an information message: " tpi))

Called by: SIMULATE

Freevars: gain netsize ntermcs ta tbd tda tdi tna tpa tpack tpb tpdaw tpi tpn a tpa treq tpsaw tpt tr2 tna treq tmsaw tt

Explanation: Displays the CONTRACT NET simulation parameters.
(DISPLAY!RECORDS
 (LAMBDA NIL
 (PRGG NIL
 (TYPE-RECORD PNODE (UTILIZATION STATUS EXECUTING READY ANNOUNCED SUSPENDED TERMINATED ACTIVE!TASK!ANNOUNCEMENTS KNOWLEDGE!BASE TASKCOUNTER))
 (TYPE-RECORD KNOWLEDGE!BASE (CONTRACT TASK!TEMPLATE TASK NODE PROCEDURE DEVICE POSITION OTHER))
 (TYPE-RECORD NODE (NAME DEVICE POSITION))
 (TYPE-RECORD DEVICE (NAME TYPE NUMBER))
 (TYPE-RECORD POSITION (NAME AREA LAT LONG))
 (TYPE-RECORD CONTRACT (NAME MANAGER REPORT!RECIPIENTS RELATED!CONTRACTORS TASK RESULTS SUBCONTRACTS STATE))
 (TYPE-RECORD SUBCONTRACT (NAME CONTRACT TASK RESULTS PREDECESSORS SUCCESSORS))
 (TYPE-RECORD TASK!TEMPLATE (TYPE ANNOUNCEMENT!PROCEDURE ANNOUNCEMENT!RANKING!PROCEDURE BID!CONSTRUCTION!PROCEDURE BID!RANKING!PROCEDURE AWARD!PROCEDURE REFUSAL!PROCEDURE REFUSAL!PROCESSING!PROCEDURE REPORT!ACCEPTANCE!PROCEDURE TERMINATION!PROCEDURE INFORMATION!ACCEPTANCE!PROCEDURE EXECUTION!PROCEDURE TASKS))
 (TYPE-RECORD TASK (NAME TYPE ANNOUNCEMENT!PROCEDURE ANNOUNCEMENT!RANKING!PROCEDURE BID!CONSTRUCTION!PROCEDURE BID!RANKING!PROCEDURE AWARD!PROCEDURE REFUSAL!PROCEDURE REFUSAL!PROCESSING!PROCEDURE REPORT!ACCEPTANCE!PROCEDURE TERMINATION!PROCEDURE INFORMATION!ACCEPTANCE!PROCEDURE EXECUTION!PROCEDURE SPECIFICATION))
 (TYPE-RECORD PROCEDURE (NAME CODE))
 (TYPE-RECORD EVENT (TIME DATA LLINK RLINK))
 (TYPE-RECORD INTERNAL!EVENT (PNODE NAME TYPE DATA))
 (TYPE-RECORD DISPLAY!EVENT (PNODE TYPE DATA))
 (TYPE-RECORD MESSAGE (TIME ADDRESSEE ORIGINATOR TEXT))
 (TYPE-RECORD TASK!ANNOUNCEMENT (NAME ELIGIBILITY!SPECIFICATION TASK!ABSTRACTION BID!SPECIFICATION EXPIRATION!TIME))
 (TYPE-RECORD ACTIVE!TASK!ANNOUNCEMENT MANAGER CONTRACT TYPE ABSTRACTION BID!SPECIFICATION TIME EXPIRATION!TIME))
 (TYPE-RECORD ACTIVE!BID (CONTRACTOR ABSTRACTION TIME))
 (TYPE-RECORD NODE!AVAILABILITY!ANNOUNCEMENT (NODE!ABSTRACTION ELIGIBILITY!SPECIFICATION EXPIRATION!TIME))
 (TYPE-RECORD BID (NAME NODE!ABSTRACTION))
 (TYPE-RECORD ANNOUNCED!AWARD (NAME TASK!SPECIFICATION))
 (TYPE-RECORD DIRECTED!AWARD (NAME ELIGIBILITY!SPECIFICATION TASK!ABSTRACTION TASK!SPECIFICATION))
 (TYPE-RECORD ACKNOWLEDGEMENT (NAME RESPONSE REFUSAL!JUSTIFICATION))
 (TYPE-RECORD INTERIM!REPORT (NAME RESULT!DESCRIPTION))
 (TYPE-RECORD FINAL!REPORT (NAME RESULT!DESCRIPTION))
 (TYPE-RECORD TERMINATION (NAME))
 (TYPE-RECORD REQUEST (NAME REQUEST!SPECIFICATION))
 (TYPE-RECORD INFORMATION (NAME INFORMATION!SPECIFICATION))
 (TYPE-RECORD BOARD (COLUMN Q A B C QUEENS)))

Explanation: Included so that CONTRACT NET record definitions appear in a file generated with "listfn's."
(DISPLAY STATISTICS
(LAMBDA NIL
(PROG (k ptu ptu2 temp)
  (DISPLAY)
  (DISPLAY "Time Units to Completion: " rtime)
  (DISPLAY)
  (DISPLAY "Communications Traffic Summary")
  (DISPLAY "-----------------------")
  (DISPLAY)
  (DISPLAY "Number of messages: " messagecounter)
  (DISPLAY "Number of broadcast messages: " bdcstcounter)
  (DISPLAY "Number of task announcements: " tacounter)
  (DISPLAY "Number of bids: " bidcounter)
  (DISPLAY "Number of announced awards: " aacounter)
  (DISPLAY "Number of directed awards: " dacounter)
  (DISPLAY "Number of acceptances: " acccounter)
  (DISPLAY "Number of refusals: " rcounter)
  (DISPLAY "Number of interim reports: " ircounter)
  (DISPLAY "Number of final reports: " frcounter)
  (DISPLAY "Number of terminations: " tecounter)
  (DISPLAY "Number of node availability announcements: " nacounter)
  (DISPLAY "Number of requests: " rqcounter)
  (DISPLAY "Number of information messages: " imcounter)
  (DISPLAY)
  (DISPLAY "Number of events: " eventcounter)
  (DISPLAY)
  (DISPLAY "Number of task re-announcements: " tracounter)
  (DISPLAY)
  (DISPLAY "Processor Node Utilization Statistics")
  (DISPLAY "--------------------------")
  (DISPLAY)
  (DISPLAY " Node Utilization")
  (SETQ ptu 0)
  (SETQ ptu2 0)
  (SETQ k 0)
  (for xpnode from 1 to netsize do
    (COND ((IGREATERP (SETQ temp (IDIFFERENCE (ELT utilization xpnode) 1)) 0)
      (SETQ k (ADD1 k))
      (SETQ ptu (IPLUS ptu temp))
      (SETQ ptu2 (IPLUS ptu2 (ITIMES temp temp))))
    (DISPLAY " " xpnode " " Mean Processor Node Utilization: " (FQUOTIENT ptu (ITIMES k rtime)))
    (DISPLAY " " xpnode " " Standard Deviation: " (FQUOTIENT (SQRT (FQUOTIENT (FDIFFERENCE ptu2 (FQUOTIENT (ITIMES ptu ptu) k)) (SUB1 k))) rtimex)
  ))
  (DISPLAY))
)

Called by: SIMULATE

Freevars: aacounter acccounter bdcstcounter bidcounter dacounter eventcounter frcounter ircounter imcounter nacounter netsize rcounter tecounter tracounter utilizat

Explanation: Displays processor node utilization statistics for the simulation.
\begin{verbatim}
EXTEND!BOARD
LAMBDA (xpnode xname xspecification xcontract)
  (PRG (xboard nextrow xrow subtaskflag solutions)
  (SETQ nextrow 1)
  (do (SETQ xboard (NEW!BOARD (fetch (BOARD COLUMN) of xspecification)
    (fetch (BOARD Q) of xspecification)
    (fetch (BOARD A) of xspecification)
    (fetch (BOARD B) of xspecification)
    (fetch (BOARD C) of xspecification))
    (SETA (fetch (BOARD Q) of xboard)
      (fetch (BOARD COLUMN) of xboard)
      nextrow)
  (UPDATE!TASK!TIME tgenerate)
  (COND
    ((GOOD!BOARD xboard)
      (SETQ subtaskflag T)
      (* a valid board has been generated -
        start up another processor to extend it)
      (* first check to see if it is a
        solution to the problem -
        if so, then report success)
    (COND
      ((EQUAL (fetch (BOARD COLUMN) of xboard)
        qsize)
        (* augment the time by the time it takes
to decide that a complete board has been
generated)
        (* report success)
    (replace (CONTRACT RESULTS) of xcontract with (LIST (LIST 'SUCCESS 'BOARD 'Q (fetch (BOARD Q) of xboard))))
    (TERM))
    (SETQ xrow (ELT (fetch (BOARD Q) of xboard)
      (fetch (BOARD COLUMN) of xboard)))
    (SETA (fetch (BOARD A) of xboard)
      xrow T)
    (SETA (fetch (BOARD B) of xboard)
      (IPLUS (fetch (BOARD COLUMN) of xboard)
        xrow)
      T)
    (SETA (fetch (BOARD C) of xboard)
      (IDIFFERENCE (IPLUS qsize xrow)
        (fetch (BOARD COLUMN) of xboard))
      T)
    (replace (BOARD COLUMN) of xboard with (ADD1 (fetch (BOARD COLUMN) of xboard)))
    (* augment the time by the time it takes
to package a subtask)
  (UPDATE!TASK!TIME tsesubtask)
  (CNETe 'SDISPLAY (LIST CONS "Generated Board-->")
  (QDISPLAY (fetch (BOARD Q) of xboard)
  (CNETe 'GENERATE!SUBTASK (LIST (LIST 'EXTEND!BOARD xboard)))
  (SETQ nextrow (ADD1 nextrow)) until (IGREATERP nextrow qsize))
  (COND
    (subtaskflag (COND
      ((fetch (CONTRACT RESULTS) of xcontract)
        (RECEIVE xpnode xname xcontract))
      (T (SUSPEND)
        (RECEIVE xpnode xname xcontract))
    (T)
    (UPDATE!TASK!TIME tgfailure)
    (replace (CONTRACT RESULTS) of xcontract with '((FAILURE))
    (CNETe 'FINAL!REPORT (LIST (LIST (LIST 'FAILURE)
      (TERMINATE))))
\end{verbatim}
Freevars: qsize tqfailure tqgenerate tqsubtask tqsuccess

Explanation: The task execution function for the N Queens 'extend!board' task. "xpnode" is the node in which the function is being executed, "xname" is the name of the contract, "xspecification" is the 'task!specification', and "xcontract" is the contract record.

Generates subtasks by adding 1 new queen to the existing board to each possible row for the next column. Accepts reports and passes them on according to the report strategy set up in QSET!PARAMETERS. Updates task time for a realistic of concurrency.

edit: 18-Sep-78 07:36 [CNET]

(FINAL!REPORT
  (LAMBDA (xpnode xname xrslt xaddressee)
    (COND
      ((SAME!STATUS!CHECK xpnode xname)
       (PROG (xpnode! xcontract xreport)
         (SETO xpnode! (ELT NET xpnode))
         (SETO xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT xname))
         (COND
           ((NOT xaddressee)
            (SETQ xaddressee (fetch (CONTRACT REPORT!RECIPIENTS) of xcontract))
            (to default addressee is the list of report recipients for the contract))
         )
         (SETO xreport (create FINAL!REPORT NAME ← xname RESULT!DESCRIPTION ← xrslt))
         (SENDMESSAGE xpnode (IPLUS time tr2)
           xaddressee xreport)
      )))
    )
  )

Calls: RETRIEVE!OBJECT SAME!STATUS!CHECK SENDMESSAGE

Freevars: NET time tr2

Explanation: Sends a final report from "xpnode" to "xaddressee" for the contract with name "xname". If "xaddressee" is NIL, then the report is sent to the report recipients for the contract. The text of the report is "xrslt".

rgs: 27-Sep-78 20:42 [CNET]

(FIND!SUBCONTRACT
  (LAMBDA (xpnode xname)
    (PROG NIL
      (RETURN (CAR (SOME (CDR (fetch (CONTRACT SUBCONTRACTS) of (RETRIEVE!OBJECT xpnode 'CONTRACT (CDR xname) )
        (FUNCTION LAMBDA (x)
          (EQUAL (fetch (SUBCONTRACT NAME) of x)
            xname)))))
    )
  ))

Calls: RETRIEVE!OBJECT

Called by: GENERATE!SUBTASK PROCESS!FINAL!REPORT TERMINATE!SUBCONTRACTS

Explanation: Returns the subcontract with name "xname" in the node "xpnode". The function searches the subcontracts slot of the contract with name (CDR xname).
(GENERATE!SUBTASK (xnode xname xsubtask xpredecessors))

(COND
  ((SAME!STATUS!CHECK xnode xname)
   (PROG (xpnodealc scl sc sname xsubtaskname)
     (SETQ xpnode! (ELT NET xpnode))
     [SETQ scl (fetch (CONTRACT SUBCONTRACTS) of (CAR (fetch (PNODE EXECUTING) of xpnode!))]
     (COND
      [scl (FRPLACA scl (ADD1 (CAR scl))]
      [T (SETQ scl (LIST 1))]
     (SETQ xsubtaskname (STORE!TASK!OBJECT xpnode (CAR xsubtask) (CADR xsubtask)))
     (COND
      [xpredecessors (SETQ xpredecessors (for x in xpredecessors collect x when (PROG (tmpsc)
        (SETQ tmpsc (FIND!SUBCONTRACT xpnode x))
        (COND
         (tmpsc (replace (SUBCONTRACT SUCCESSORS) of tmpsc
                with (CONS (fetch (SUBCONTRACT NAME) of sc)
                (fetch (SUBCONTRACT SUCCESSORS) of tmpsc)))
         (RETURN T))
         (T (RETURN))
        ]
        (SETQ sc (create SUBCONTRACT NAME ←[CONS (CAR scl) (fetch (CONTRACT NAME) of (CAR (fetch (PNODE EXECUTING) of xpnode!)) CONTRACTOR ← 0 TASK ← xsubtaskname PREDECESSORS ← xpredecessors))]
        (FRPLACD scl (CONS sc (CDR scl)))
        (replace (CONTRACT SUBCONTRACTS) of (CAR (fetch (PNODE EXECUTING) of xpnode!)) with scl)
      (e find any outstanding (i.e., not yet completed) subcontracts in the announced state that correspond to members of the predecessors specified - mark the new subcontract as a successor - the actual outstanding subcontracts become the actual predecessors bound to the new subcontract in the announced state)
      (replace (PNODE ANNOUNCED) of xpnode! with (CONS (CONS sc NIL) (fetch (PNODE ANNOUNCED) of xpnode!)))
      (e if there are no predecessors outstanding, then take the necessary steps to announce the subcontract)
    (COND
      ((NOT xpredecessors)
       (ANNOUNCE!TASK xpnode (fetch (SUBCONTRACT NAME) of sc))
      )
    ))
  ))
(Calls: ANNOUNCE!TASK FIND!SUBCONTRACT SAME!STATUS!CHECK STORE!TASK!OBJECT)
(Freevars: NET)
Explanation: Called through CNET by a user task execution function. Generates a subtask of the contract with name "xname" in node "xpnode". The subtask is "xsubtask". The names of any subtasks that must be completed before this subtask can be announced are given by "predecessors".

A task object is created for the task, and a subcontract record is added to the subcontract list for the contract (the name is formed by consing the index of the subcontract with the name of the contract from which it is generated [i.e., a count (index) is kept of how many subcontracts have been generated from a contract]. Subcontracts are not currently stored as separate objects.

Finally the subtask is announced.

rgs: 10-Oct-78 22:32 [CNET]

(GET!TASK!ANNOUNCEMENT
LAMBDA (xpnode xname)
(PROG (sc xsubtaskname xannproc)
 (SETQ sc (CAR (NODE!SEARCH xpnode xname 'ANNOUNCED NIL T)))
 (SETQ xsubtaskname (fetch (SUBCONTRACT TASK) of sc))
 (e create a task announcement for the subtask)

 (e if a task!announcement!procedure is available, then use it to create the addressee, eligibility!specification, task!abstraction, bid!specification, and expiration!time - otherwise default to broadcast, NIL for the eligibility!specification, task!abstraction, and bid!specification, and default the expiration!time)

 (SETQ xannproc (fetch (TASK ANNOUNCEMENT!PROCEDURE) of (RETRIEVE!OBJECT xpnode 'TASK xsubtaskname)))
 (COND
 (xannproc (RETURN (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE xannproc))
 (LIST xpnode (fetch (TASK SPECIFICATION) of (RETRIEVE!OBJECT xpnode 'TASK xsubtaskname))
 (T (RETURN (LIST "o" NIL (LIST (fetch (TASK TYPE) of (RETRIEVE!OBJECT xpnode 'TASK xsubtaskname))
 NIL tra))

 Calls: NODE!SEARCH RETRIEVE!OBJECT
 Called by: ANNOUNCE!TASK MAKE!BID
 Freesvars: tra

Explanation: Forms the essence of a task announcement for the (sub)contract with name "xname" in node "xpnode". If an 'announcement!procedure' exists for the task of the contract, then it is used to return a list of 'addressee', 'eligibility!specification', 'task!abstraction', 'bid!specification' and 'expiration!time'. Otherwise default values are used - "o" (for broadcast), NIL, the task 'type' (as a list), NIL, and tra (the default expiration time).
GOODIBOARD

(LAMBDA (xboard)
  (PROG (c r)
    (SETQ c (fetch (BOARD COLUMN) of xboard))
    (SETQ r (ELT (fetch (BOARD Q) of xboard)
                 c))
    (RETURN (NOT (OR (ELT (fetch (BOARD A) of xboard) r)
                      (ELT (fetch (BOARD B) of xboard)
                           (IPLUS c r))
                      (ELT (fetch (BOARD C) of xboard)
                           (IDIFFERENCE (IPLUS qsize r)
                                       c)))))

Called by: EXTENDIBOARD

Freevars: qsize

Explanation: Returns T if the board "xboard" is a plausible partial solution for the N Queens problem.
; INITCIL
(INITCIL ILAMOA NIL (PROG NIL)
  (SETQ CILGRAMMAR (((#verb!phrase #noun!phrase #object!phrase)
                        (T (LIST #verb!phrase #noun!phrase #object!phrase)))
                       ((#noun!phrase)
                        (T #noun!phrase))
                       ((#verb!phrase #noun!phrase)
                        (T (LIST #verb!phrase #noun!phrase)))
                       ((#verb!phrase)
                        (T #verb!phrase))
                       )

(SETQ ELSPECGRAMMAR (((#OBJECT #attval))
                      (for x in #attval always (ATTRIBUTEP #OBJECT (CAR x)))
                      (PROG (object)
                            (SETQ object (RETRIEVEOBJECT xo node #OBJECT (CADAR #attval))
                                 (CAAR #attval)))
                      (RETURN (EQUAL (CADAR #attval)
                                      (RECORDACCESS (CADAR #attval))
                                      xobject
                                      (RECLOOK #OBJECT)))

(SETQ TABSGRAMMAR (((#OBJECT (#attval))
                       (for x in #attval always (ATTRIBUTEP #OBJECT (CAR x)))
                       (CONS #OBJECT #attval))
                       )

(SETQ BSPECGRAMMAR (((#VERB)
                        (T (LIST #VERB)))
                       ((#OBJECT #attl)
                        (T (COND
                             (for x in #attl always (ATTRIBUTEP #OBJECT x))
                             (CONS #OBJECT #attl)))
                       )

(SETQ RUGRAMMAR (((#OBJECT (#attval))
                       (for x in #attval ALWAYS (ATTRIBUTEP #OBJECT (CAR x)))
                       (CONS #OBJECT #attval))
                       )

(SETQ IFSGRAMMAR (((#OBJECT (#attval))
                   (for x in #attval ALWAYS (ATTRIBUTEP #OBJECT (CAR x)))
                   (CONS #OBJECT #attval))
                   )

(PUTPROP '#OBJECT 'POSSIBLEVALUES (COPY 'BUSY EVERY OWN))

(PUTPROP '#OBJECT 'POSSIBLEVALUES (COPY 'ANNOUNCEMENT 'PROCEDURE 'AREA 'AWARD 'PROCEDURE 'BID 'CONSTRUCTION 'PROCEDURE 'BID 'RANKING 'PROCEDURE 'CODE 'DEVICE 'EXECUTION 'PROCEDURE 'INFORMATION 'PROCEDURE 'ACCEPTANCE 'PROCEDURE 'LAT 'LONG 'MANAGER 'NAME 'NUMBER 'POSITION 'PREDECESSOR 'REFUSAL 'PROCEDURE 'REFUSAL 'PROCESSING 'PROCEDURE 'RELATED 'CONTRACTOR 'REPORT 'RECIPIENT 'RESULT 'SUBCONTRACT 'SUCCESSOR 'SPECIFICATION 'TYPE)

(PUTPROP '#OBJECT 'POSSIBLEVALUES (COPY 'BID 'CONSTRUCTION 'PROCEDURE 'BID 'RANKING 'PROCEDURE 'CODE 'DEVICE 'EXECUTION 'PROCEDURE 'INFORMATION 'PROCEDURE 'ACCEPTANCE 'PROCEDURE 'LAT 'LONG 'MANAGER 'NAME 'NUMBER 'POSITION 'PREDECESSOR 'REFUSAL 'PROCEDURE 'REFUSAL 'PROCESSING 'PROCEDURE 'RELATED 'CONTRACTOR 'REPORT 'RECIPIENT 'RESULT 'SUBCONTRACT 'SUCCESSOR 'SPECIFICATION 'TYPE)

(PUTPROP '#AUXILIARY 'POSSIBLEVALUES 'MUST)

(PUTPROP '#CONNECTIVE 'POSSIBLEVALUES 'AND NOT OR)

(PUTPROP '#OBJECT 'POSSIBLEVALUES (COPY 'CONTACT 'DEVICE 'NODE 'POSITION 'PROCEDURE 'TASK 'TASKTEMPLATE))

(PUTPROP '#PREPOSITION 'POSSIBLEVALUES 'TO FROM WITH)

(PUTPROP '#VALUE 'PREDICATE 'VALUE)

(PUTPROP '#VERB 'POSSIBLEVALUES 'ACKNOWLEDGE BID CHANGE CONFIRM HAVE REQUIRE RESPOND SEND SUSPEND)

(PUTPROP '#att1 'GRAMMARS '(((#ATTRIBUTE #att1))
                           (T (CONS #ATTRIBUTE #att1)))
                           )

(PUTPROP '#att2 'GRAMMARS '(((#att1 #obval #att2))
                           (T (APPEND #att1 #obval)
                               (LIST #att2)))
                           )

(PUTPROP '#attval 'GRAMMARS '(((#ATTRIBUTE #VALUE #attval))
                             (T (CONS (LIST #ATTRIBUTE #VALUE)
                                     #attval)))
                             )

(PUTPROP '#ATTRIBUTE 'VALUE)

(PUTPROP '#noun!phrase 'GRAMMARS '(((#np2 #CONNECTIVE #np3)

--22--
Called by: INITIALIZE

Freevars: BSPECGRAMMAR CILGRAMMAR ELSPECGRAMMAR INFOGRAMMAR REPGRAMMAR REGRAMMAR TABSGRAMMAR

Explanation: Initializes the common internode language.
(INITIALIZE (LAMBDA NIL (PROG NIL
(INITCIL)
(SETQ eventlist (create EVENT TIME ← infinity DATA ←"header")
(SETQQ time -1)
(SETQQ task!time 0)
(SETQQ rtime 0)
(SETQQ NET (ARRAY netsize 0 NIL))
(for xpnode from 1 to netsize do (SETA NET xpnode (create PNODE UTILIZATION ← 0 STATUS ←"Idle" KNOWLEDGE!BASE ←(create KNOWLEDGE!BASE) TASKCOUNTER ← 0))
(STORE!OBJECT xpnode 'NODE (create NODE NAME ←'SELF)))

(INSTALL!DISPLAY!EVENT (LAMBDA (xtime xpnode xtype xdata) (* rgs: "7-Sep-78 05:43")
(PROG (d)
(SETQ d (create DISPLAY!EVENT PNODE ← xpnode TYPE ← xtype DATA ← xdata))
(INSTALL!EVENT xtime d))

Explanation: Installs a display event in the event list at time "xtime". The event is placed by node "xpnode". It is of type "xtype" and the data of the event is given by "xdata". There are two types of display events: 'simulation' and 'task'. Both are used to display text of some sort in the execution trace of the simulation. Simulation display events originate in contract net functions, and are used to display messages about the status of contract execution. Task display events originate in user functions and can be used to display any string at the correct simulation time, with an indicator as to the originator of the event.
INSTALL!EVENT

(LAMBDA (xtime xdata)
  (PROG (e ev)
    (SETQ ev (create EVENT TIME ← xtime DATA ← xdata))
    (SETQ e eventlist)
    (while T do (COND
      ((ILESSP xtime (fetch (EVENT TIME) of e))
        (COND
          ((fetch (EVENT LLINK) of e)
            (SETQ e (fetch (EVENT LLINK) of e)))
          (T (replace (EVENT LLINK) of e with ev)
            (GO $$OUT)
          )
        )
      )
      (T (COND
        ((fetch (EVENT RLINK) of e)
          (SETQ e (fetch (EVENT RLINK) of e)))
        (T (replace (EVENT RLINK) of e with ev)
          (GO $$OUT)
        )
      )
    )
  )
)

Called by: INSTALL!DISPLAY!EVENT INSTALL!INTERNAL!EVENT SENDMESSAGE

Freevars: eventlist

Explanation: Installs an event in the event list at time "xtime". "xdata" is the data of the event. The event list is currently stored as a binary tree. There are three types of event: messages, internal events, and display events.

INSTALL!INTERNAL!EVENT

(LAMBDA (xtime xpnode xname xtype xdata)
  (PROG (i)
    (SETQ i (create INTERNAL!EVENT PNODE ← xpnode NAME xname TYPE ← xtype DATA ← xdata))
    (INSTALL!EVENT xtime i)
)

Calls: INSTALL!EVENT

Called by: ANNOUNCE! TASK MAKE! BID NEXT! CONTRACT PROCESS! ANNOUNCED! AWARD PROCESS! CONTRACT PROCESS! DIRECTED! AWARD SIMULATE UPDATE! NODE

Explanation: Installs an internal event in the event list at time "xtime". The node involved is "xpnode", and the name of the contract involved is "xname". "xtype" is the type of internal event and "xdata" is the data. There are currently four types of internal events: 'contract!processing' and 'node!update', that are used to perform the necessary bookkeeping for task execution; 'bid!check', that is used to assess bids and take action (if necessary) at the end of the expiration time for a task announcement; and, 'pseudo!contract', that is used to eliminate (if necessary) the temporary information stored by a node in anticipation of the receipt of a contract on which a bid has been made.
(INTERIMREPORT
 (LAMBDA (xpnode xname xrslt xaddressee)
   (COND
    ((SAME!STATUS!CHECK xpnode xname)
     (PROG (xpnode! xcontract xreport)
       (SETQ xpnode! (ELT NET xpnode))
       (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT xname))
       (COND
        ((NOT xaddressee)
         (SETQ xaddressee (fetch (CONTRACT REPORT!RECIPIENTS) of xcontract)))
         ((default addresssee is the list of report recipients for the contract)
          ))
       (SETQ xreport (create INTERIMREPORT NAME ← xname RESULT!DESCRIPTION ← xrslt))
       (SENDMESSAGE xpnode (IPLUS time tr2)
                      xaddressee xreport))
   )
  )
)

Calls: RETRIEVE!OBJECT SAME!STATUS!CHECK SENDMESSAGE

Freevars: NET time tr2

Explanation: Sends an interim report from "xpnode" to "xaddressee" for the contract with name "xname". If "xaddressee" is NIL, then the report is sent to the report!recipients for the contract. The text of the report is "xrslt".
MAKE!BID

ILAMBDA (xpnode)
(PROG (xpnode! active pc oldest temp xtype $$es $$bs $$expt)
  (SETQ xpnode! (ELT NET xpnode))
  (SETQ active (fetch (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode!))

  (SETQ oldest (CAR active))
  (COND
    (oldest (for x in (CDR active) do (COND
      ((ILESSP (fetch (ACTIVE!TASK!ANNOUNCEMENT TIME) of x)
        (fetch (ACTIVE!TASK!ANNOUNCEMENT TIME) of oldest))
        (SETQ oldest x))

      (BID xpnode (fetch (ACTIVE!TASK!ANNOUNCEMENT CONTRACT) of oldest)
        (fetch (ACTIVE!TASK!ANNOUNCEMENT MANAGER) of oldest)
        (fetch (ACTIVE!TASK!ANNOUNCEMENT TYPE) of oldest)
        (fetch (ACTIVE!TASK!ANNOUNCEMENT BID!SPECIFICATION) of oldest))
      (replace (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode! with (REMOVE oldest active)))

    (* store a "pseudo-contract" in the knowledge base so that an award message can be handled without
     need for retransmission of task type and so on - then set up a "pseudo-contract" internal event to remove the pseudo-contract after the expiration
     time plus the time it takes to get an award has passed - that is, if the node is not awarded the contract)

    (STORE!OBJECT xpnode 'CONTRACT pc)
    INSTALL!INTERNAL!EVENT (IPLUS time (fetch (ACTIVE!TASK!ANNOUNCEMENT EXPIRATION!TIME) of oldest)
                             tpb tsaw tpsaw)
                             xpnode
                             (fetch (CONTRACT NAME) of pc)
                             'PSEUDO!CONTRACT)
    (T)
    (* note that the "task" slot is filled
     in with a task!template "type" until the
     award is received)
    (SETQ pc (create CONTRACT NAME ←(fetch (ACTIVE!TASK!ANNOUNCEMENT CONTRACT) of oldest)
              MANAGER ←(fetch (ACTIVE!TASK!ANNOUNCEMENT MANAGER) of oldest)
              REPORT!RECIPIENTS ←LIST (fetch (ACTIVE!TASK!ANNOUNCEMENT MANAGER) of oldest))
              STATE ←'PSEUDO))
    (STORE!OBJECT xpnode 'CONTRACT pc)
    (INSTALL!INTERNAL!EVENT (IPLUS time (fetch (ACTIVE!TASK!ANNOUNCEMENT EXPIRATION!TIME) of oldest)
                             tpb tsaw tpsaw)
                             xpnode
                             (fetch (CONTRACT NAME) of pc)
                             'PSEUDO!CONTRACT))
    (T)
    (* look on the announced list for
     outstanding subcontracts and make a bid
     on the oldest one for which the node
     meets the eligibility specification)
    (SETQ xan (REVERSE (fetch (PNODE ANNOUNCED) of xpnode!)))
    (COND
      (xan ISETQ temp!CAAR (SOME xan (FUNCTION ILAMBDA x)
                               (PROG (temp)
                                 ISETQ templ!GET!TASK!ANNOUNCEMENT xpnode
                                 (fetch (SUBCONTRACT NAME)
                                      of (CAR x))
                                 (COND
                                  (NOT (EQUAL (CAR templ) 'DIRECTED!AWARD))
                                  (SETQ $$es (CARD temp!))
                                  (COND
                                    (AND NOT (fetch (SUBCONTRACT PREDECESSORS)
                                              of (CAR x))
                                              OR (NOT $$es)
                                              (CHECK!ELIGIBILITY xpnode $$es)))
                                    (SETQ $$bs (CAADDR templ!))
                                    (SETQ $$expt (CAR (CADDR temp!)))
                                    (RETURN T))
                                  (T (RETURN))
                                  (T (RETURN))
                                  (RETURN))
                                (T (RETURN))
                                (T (RETURN))
                                (T (RETURN))
                                (T (RETURN))
                                (T (RETURN)))
    (COND
      (temp ISETQ xtype (fetch (TASK TYPE) of (RETRIEVE!OBJECT xpnode 'TASK
                               (fetch (SUBCONTRACT TASK) of temp))}

-27-
(BID xpnode (fetch (SUBCONTRACT NAME) of temp)
  xpnode xtype $$bs)
(SETQ pc (create CONTRACT NAME ← (fetch (SUBCONTRACT NAME) of temp)
  MANAGER ← xpnode REPORTRECIPIENTS ← (LIST xpnode)
  TASK ← xtype STATE ← 'PSEUDO)
(STORE!OBJECT xpnode 'CONTRACT pc)
(INSTALL!INTERNAL!EVENT (PLUS time $$expt tpb tsaw tpsaw)
  xpnode
  (fetch (SUBCONTRACT NAME) of temp)
  'PSEUDO!CONTRACT))

Calls: BID CHECK!ELIGIBILITY GET!TASK!ANNOUNCEMENT INSTALL!INTERNAL!EVENT RETRIEVE!OBJECT STORE!OBJECT
Called by: NEXT!CONTRACT PROCESS!TASK!ANNOUNCEMENT
Freevars: NET time tpb tsaw tpsaw
Explanation: Makes a bid on an appropriate contract by node "xpnode".
  First looks at the "active!task!announcements" list. If only one task 'type' exists then bid on it.
  Otherwise bid on the oldest announcement. If no active task announcements exist then check the 'announced'
  state, and bid on the oldest subcontract for which the node meets the eligibility specification (and for
  which there are no predecessors).

rgs: 16-Oct-78 21:55 [CNET]

(NEW!BOARD
  (LAMBDA (xcol xq xa xb xc) (* rgs: "16-Oct-78 21:55")
    (PROG (xcolumn xQ xA xB xC)
      (SETQ xcolumn (COND
          (xcol xcol)
          (T 1))
      )
      (SETQ xQ (COND
          (xq (COPYALL xq))
          (T (ARRAY qsize qsize))
      )
      )
      (SETQ xA (COND
          (xa (COPYALL xa))
          (T (ARRAY qsize NIL NIL))
      )
      )
      (SETQ xB (COND
          (xb (COPYALL xb))
          (T (ARRAY (LSH qsize 1) NIL NIL))
      )
      )
      (SETQ xC (COND
          (xc (COPYALL xc))
          (T (ARRAY (SUB1 (LSH qsize 1)) NIL NIL))
      )
      )
      (RETURN (create BOARD COLUMN ← xcolumn Q ← xQ A ← xA B ← xB C ← xC))
    )
  )
)

Called by: EXTEND!BOARD QINITIALIZE
Freevars: qsize
Explanation: Generates a new board for the N Queens problem. "xcol" is the column in which the next queen is to be
  placed (1 if "xcol" is NIL). "xq" is the array of current row indices in which queens have been placed (all
  NIL if "xq" is NIL). "xa", "xb", and "xc" are the arrays associated with Floyd's solution of the problem
  [JACM 14:4 Oct. '67, pp. 636-644] (all NIL if corresponding arguments are NIL).
(NEXT!CONTRACT)

(LAMBDA (xpnode xtype)
  (PROG (xpnode rc)
    (SETQ xpnode! (ELT NET xpnode))
    (COND
      [(fetch (PNODE READY) of xpnode!)
       (SETQ rc (CAR (fetch (PNODE READY) of xpnode!)))
       (replace (PNODE EXECUTING) of xpnode! with (LIST (CAR rc)))
       (replace (PNODE READY) of xpnode! with (CDR (fetch (PNODE READY) of xpnode!))
       (T (replace (PNODE EXECUTING) of xpnode! with NIL)))
    (COND
      [(fetch (PNODE EXECUTING) of xpnode!)
       (replace (PNODE STATUS) of xpnode! with "Busy")
       (replace (CONTRACT STATE) of (CAR (fetch (PNODE EXECUTING) of xpnode!)) with 'EXECUTING)
       (COND
        [(CDR rc)
         (INSTALL!DISPLAY!EVENT (IPLUS time (COND
           ([EQUAL xtype 'REPORT)
            tpr)
            (T tt))
          xpnode
          (fetch (CONTRACT NAME) of (CAR rc))
          'SIMULATION
          (APPEND ' (Resumed Processing Contract)
            (fetch (CONTRACT NAME) of (CAR rc))
            (INSTALL!INTERNAL!EVENT (IPLUS time (COND
              ([EQUAL xtype 'REPORT)
               tpr)
               (T tt))
            xpnode
            (fetch (CONTRACT NAME) of (CAR rc))
            'NODE!UPDATE
            (LIST (CDR rc))
            (T (INSTALL!DISPLAY!EVENT (IPLUS time (COND
              ([EQUAL xtype 'REPORT)
               tpr)
               (T tt))
            xpnode
            (fetch (CONTRACT NAME) of (CAR rc))
            'SIMULATION
            (APPEND ' (Started Processing Contract)
              (fetch (CONTRACT NAME) of (CAR rc))
              (INSTALL!INTERNAL!EVENT (IPLUS time (COND
                ([EQUAL xtype 'REPORT)
                 tpr)
                 (T tt))
              xpnode
              (fetch (CONTRACT NAME) of (CAR rc))
              'CONTRACT!PROCESSING
              (UPDATE!ACTIVE!TASK!ANNOUNCEMENTS xpnode)
              (MAKE!BID xpnode))
            (T (replace (PNODE STATUS) of xpnode! with "Idle")
              (UPDATE!ACTIVE!TASK!ANNOUNCEMENTS xpnode)
              (MAKE!BID xpnode))
          )
          )
      )
      )
    )
  )))

Calls: INSTALL!DISPLAY!EVENT INSTALL!INTERNAL!EVENT MAKE!BID UPDATE!ACTIVE!TASK!ANNOUNCEMENTS

Called by: PROCESS!FINAL!REPORT PROCESS!INTERIM!REPORT UPDATE!NODE

Freevars: NET time tpr tt

Explanation: Tries to install a new contract in the 'executing' state of node "xpnode". If a contract exists in the
'ready' state, then it is installed, and an appropriate event (either 'contract!processing' or
'node!update') is installed on the event list to start processing. Otherwise an attempt is made to bid on
an active task announcement or outstanding subcontract. There are two types of call to this function,
specified by "xtype": 'report' and 'termination'. The type is used to update simulation time in the correct
manner.
(LAMBDA NIL
  (PROG (e1 e2)
     (SETQ e1 event list)
     (SETQ e2 (fetch (EVENT LLINK) of e1))
     (while (fetch (EVENT LLINK) of e2) do ((SETQ e1 e2)
       (SETQ e2 (fetch (EVENT LLINK) of e2)
       (replace (EVENT LLINK) of e1 with (fetch (EVENT RLINK) of e2))
       (replace (EVENT LLINK) of e2 with NIL)
       (replace (EVENT RLINK) of e2 with NIL)
       (RETURN e2))
     )))

Called by: SIMULATE
Freevars: event list
Explanation: Returns the next event to be processed from the event list.
(PROG (xpnode c)
  (SETQ xpnode (ELT NET xpnode))
  (SELECTQ xstate
    (EXECUTING (GO LEX))
    (READY (GO LRD))
    (ANNOUNCED (GO LAN))
    (SUSPENDED (GO LSU))
    (TERMINATED (GO LTR))
    NIL)
   (LEX ISETQ c (CAR (SOME (fetch (PNODE EXECUTING) of xpnode))
   (FUNCTION (LAMBDA (x)
     (EQUAL xname (fetch (CONTRACT NAME) of x)
     (COND
      (c (COND
        ((AND xdeleteflag (NOT (MEMBER 'EXECUTING xconditions)))
         (replace (PNODE EXECUTING) of xpnode! with (REMOVE c (fetch (PNODE EXECUTING) of xpnode!)
         (RETURN c))))
      (COND
       (xstate (RETURN NIL))))
   (LRD ISETQ c (CAR (SOME (fetch (PNODE READY) of xpnode))
   (FUNCTION (LAMBDA (x)
     (EQUAL xname (fetch (CONTRACT NAME) of (CAR x)
     (COND
      (c (COND
        ((AND xdeleteflag (NOT (MEMBER 'READY xconditions)))
         (replace (PNODE READY) of xpnode! with (REMOVE c (fetch (PNODE READY) of xpnode!)
         (RETURN c))))
      (COND
       (xstate (RETURN NIL))))
   (LAN ISETQ c (CAR (SOME (fetch (PNODE ANNOUNCED) of xpnode))
   (FUNCTION (LAMBDA (x)
     (EQUAL xname (fetch (SUBCONTRACT NAME) of (CAR x)
     (COND
      (c (COND
        ((AND xdeleteflag (NOT (MEMBER 'ANNOUNCED xconditions)))
         (replace (PNODE ANNOUNCED) of xpnode! with (REMOVE c (fetch (PNODE ANNOUNCED) of xpnode!)
         (RETURN c))))
      (COND
       (xstate (RETURN NIL))))
   (LSU ISETQ c (CAR (SOME (fetch (PNODE SUSPENDED) of xpnode))
   (FUNCTION (LAMBDA (x)
     (EQUAL xname (fetch (CONTRACT NAME) of (CAR x)
     (COND
      (c (COND
        ((AND xdeleteflag (NOT (MEMBER 'SUSPENDED xconditions)))
         (replace (PNODE SUSPENDED) of xpnode! with (REMOVE c (fetch (PNODE SUSPENDED) of xpnode!)
         (RETURN c))))
      (COND
       (xstate (RETURN NIL))))
   (LTR ISETQ c (CAR (SOME (fetch (PNODE TERMINATED) of xpnode))
   (FUNCTION (LAMBDA (x)
     (EQUAL xname (fetch (CONTRACT NAME) of x)
     (COND
      (c (COND

-31-
((AND xdeleteflag (NOT (MEMBER "TERMINATED" xconditions)))
  (replace (PNODE TERMINATED) of xpnode! with (REMOVE c (fetch (PNODE TERMINATED) of xpnode!)
  (RETURN c))

Called by: AWARD CHECK!BIDS DIRECTED!AWARD GET!TASK!ANNOUNCEMENT PROCESS!BID PROCESS!FINAL REPORT
PROCESS!INTERIM REPORT TERMINATE!SUBCONTRACTS

Freevars: NET

Explanation: Searches the contract processing states of "xpnode" for the contract with name "xname", and returns
the contract record, if found. "xstate" can specify the state to be searched. If "xdeleteflag" is T then
the contract is removed from the processing state in which it is found. "xconditions" is a list of states
from which the contract should not be deleted. It overrides "xdeleteflag".

rgs: 17-Oct-78 00:10 [CNET]

OBJECTP
(LAMBDA (xobject)
  (COND
   ((RECLOOK xobject)
    T)
   (T (WRITE "CIL error: " xobject " is not a valid object")
    NIL))

Called by: ATTRIBUTE

Explanation: Returns T if "xobject" is a valid object; else WRITEs an error message and returns NIL.

rgs: 18-Sep-78 00:45 [CNET]

OUTSTANDING!SUBCONTRACTS
(LAMBDA (xcontract)
  (CAR (fetch (CONTRACT SUBCONTRACTS) of xcontract))

Called by: QRECEIVE

Explanation: Returns the number of subcontracts of "xcontract" that have not yet been completed.

rgs: 10-Sep-78 17:17 [CNET]

PARSE!NODE!ABSTRACTION
(LAMBDA (xabstraction)
  xabstraction)

Called by: PROCESS!BID

Explanation:
Called by: PROCESS!DIRECTED!AWARD PROCESS!TASK!ANNOUNCEMENT

Explanation:

rgs: 7-Sep-78 08:49 [CNET]

(PROCESS!ACKNOWLEDGEMENT
 (LAMBDA (xpnode xmessage) NIL))

Called by: PROCESS!MESSAGE

Explanation:

rgs: 1-Oct-78 17:41 [CNET]

(PROCESS!ANNOUNCED!AWARD
 (LAMBDA (xpnode xmessage)
 (PROG (xpnode! xsa xcontract xtaskname temp)
 (SETQ xpnode! (ELT NET xpnode))
 (SETQ xsa (fetch (MESSAGE TEXT) of xmessage))
 (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT (fetch (ANNOUNCED!AWARD NAME) of xsa)))
 (SETQ xtaskname (STORE!TASK!OBJECT xpnode (fetch (CONTRACT TASK) of xcontract))
 (fetch (ANNOUNCED!AWARD TASK!SPECIFICATION) of xsa))))
 (COND
 (EQUAL (fetch (PNODE STATUS) of xpnode!) "Busy")
 (replace (CONTRACT TASK) of xcontract with xtaskname)
 (replace (CONTRACT STATE) of xcontract with 'READY)
 (READY!CONTRACT xpnode xcontract))
 (T (replace (CONTRACT STATE) of xcontract with 'EXECUTING)
 (replace (PNODE EXECUTING) of xpnode! with (LIST xcontract))
 (replace (PNODE STATUS) of xpnode! with "Busy")
 (INSTALL!INTERNAL!EVENT (IPLUS time tpsaw) xpnode
 (fetch (CONTRACT NAME) of xcontract)
 'CONTRACT!PROCESSING))

Called by: INSTALL!INTERNAL!EVENT READY!CONTRACT RETRIEVE!OBJECT STORE!TASK!OBJECT

Called by: PROCESS!MESSAGE

Freevars: NET time tpsaw

Explanation: Performs the necessary bookkeeping to handle the receipt of an announced award by node "xpnode". 
"xmessage" is the message. If the node is "Idle", then an event is placed on the event list to begin 
processing on the new contract. Otherwise the contract is placed in the 'ready' state.
(PROCESSIBILITY)

(LAMBDA (xpnode xmessage)
  (PROG (xpnode! xbid xnode!abstraction xbidl xbidrankproc sc active!bids)
    (setq xpnode! (elt net xpnode))
    ;; make sure that the contract has not yet been awarded by searching for it in the announced state - remember that it is bound to the active!bids
    (setq xbid (fetch (MESSAGE TEXT) of xmessage))
    (setq sc (NODESEARCH xpnode (fetch (BID NAME) of xbid)
      'ANNOUNCED NIL T))
    (cond
      (sc (setq xnode!abstraction (PREAD!NODE!ABSTRACTION (fetch (BID NODE!ABSTRACTION) of xbid)))
        (setq xbidl (create ACTIVE!BID CONTRACTOR ←(fetch (MESSAGE ORIGINATOR) of xmessage)
          ABSTRACTION ← xnode!abstraction TIME ←(fetch (MESSAGE TIME) of xmessage)))
        (setq xbidrankproc (fetch (TASK BID!RANKING!PROCEDURE) of (RETRIEVE!OBJECT xpnode 'TASK
          (fetch (SUBCONTRACT TASK) of (CAR sc)
            (if there is a bid ranking procedure then use it, else cons the new bid to the old active!bids list)
            (cond
              (xbidrankproc (setq active!bids (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode
                'PROCEDURE xbidrankproc))))
                (list xbidl (cadr sc)
                  ;; the bid ranking procedure returns (action list). if the action is (QUOTE satisfactory,) then award the contract to the new bidder
                  (cond
                    ((equal (car active!bids) 'SATISFACTORY)
                      (AWARD xpnode (fetch (SUBCONTRACT NAME) of (car sc))
                        (fetch (ACTIVE!BID CONTRACTOR) of (cdr active!bids))
                        (t (frplaca (cdr sc) active!bids))))
                      ;; update the active!bids list
                      (t (setq active!bids (cons xbidl (cdr sc)))))
                      ;; update the active!bids list
                      (t (setq active!bids (cons xbidl (cdr sc)))))
        (t (setq active!bids (cons xbidl (cdr sc)))))))
    (cond
      (t (setq active!bids (cons xbidl (cdr sc))))
    (cond
      (equal (car active!bids) 'SATISFACTORY)
        (AWARD xpnode (fetch (SUBCONTRACT NAME) of (car sc))
          (fetch (ACTIVE!BID CONTRACTOR) of (cdr active!bids)))
        (t (frplaca (cdr sc) active!bids)))))))

Calls: AWARD NODE!SEARCH PARSE!NODE!ABSTRACTION RETRIEVE!OBJECT

Called by: PROCESS!MESSAGE

Freevars: NET

Explanation: Performs the necessary bookkeeping to handle the receipt of a bid by node "xpnode". "xmessage" is the message. If the contract has not already been awarded then it is ranked relative to other bids using the bid!ranking!procedure for the task. If no procedure exists, then the bid is consed to the old active!bid list for the contract.

If a bid!ranking!procedure exists, and it returns 'satisfactory as the first element of its list of values, then the contract is awarded to the contractor named in the active!bid record that is the second element of the list.
PROCESS\ CONTRACT

\LAMBDA (xpnode xname)

(e to get the processing on the task associated with a contract started, set up a possibilities list
and use TRYNEXT to get a value -
the value is only used for messages to the update!node function which determines whether a contract
should be suspended, terminated, or resumed)

(PROG (xpnode! xcontract taskprocesspointer temp)
 (SETQ xpnode! (ELT NET xpnode))
 (COND
  ((EQUAL (fetch (PNODE STATUS) of xpnode!) "Idle")
   (RETURN))
  (T (SETQ xcontract (CAR (fetch (PNODE EXECUTING) of xpnode))))
  (INSTALL\DISPLAY\EVENT time xpnode 'SIMULATION
   (APPEND '((Started Processing Contract)
     xname))
  (SETQ taskprocesspointer (POSSIBILITIES
    (APPLY (fetch (PROCEDURE CODE)
      of (RETRIEVE\OBJECT xpnode 'PROCEDURE
        (fetch (TASK EXECUTION\PROCEDURE)
          of (RETRIEVE\OBJECT xpnode 'TASK
            (fetch (CONTRACT TASK)
              of xcontract))))
      xname)
    LIST xpnode (fetch (CONTRACT NAME) of xcontract)
    (fetch (TASK SPECIFICATION)
      of (RETRIEVE\OBJECT xpnode 'TASK
        (fetch (CONTRACT TASK)
          of xcontract))))
  )
  (SETQ temp (RESUME\TASK taskprocesspointer))
  (INSTALL\INTERNAL\EVENT (IPLUS time task!time)
    xpnode
    (fetch (CONTRACT NAME) of xcontract)
    'NODE\UPDATE
    (CONS taskprocesspointer temp))
  (RETURN T))

Calls: INSTALL\DISPLAY\EVENT INSTALL\INTERNAL\EVENT RESUME\TASK RETRIEVE\OBJECT

Called by: PROCESS\INTERNAL\EVENT

Freevars: NET task!time time

Explanation: Starts the processing of the contract named "xname" in node "xpnode". Sets up the task execution
function as a generator via POSSIBILITIES. Then installs a 'node!update' event to continue, after
performing a RESUME\TASK.
(PROCESS!DIRECTED!AWARD)

(LAMBDA (xpnode xmessage)
  (PROG (xabs xpnode! xcontract xtaskname xtt xrefproc xrefjust)
  (SETQ xpnode! (LET NET xpnode))
  (SETQ xda (fetch (MESSAGE TEXT) of xmessage))
  (SETQ xabs (PARSE!TASK!ABSTRACTION (fetch (DIRECTED!AWARD TASK!ABSTRACTION) of xda)))
  (SETQ xtt (RETRIEVE!OBJECT xpnode 'TASK!TEMPLATE (CAR xabs)))
  (COND
   ((AND (CHECK!ELIGIBILITY xpnode (fetch (DIRECTED!AWARD ELIGIBILITY!SPECIFICATION) of xda))
       xtt)
    (SETQ xtaskname (STORE!TASK!OBJECT xpnode (CAR xabs)
       (fetch (DIRECTED!AWARD TASK!SPECIFICATION) of xda)))
    (SETQ xcontract (create CONTRACT NAME ←(fetch (DIRECTED!AWARD NAME) of xda)
       MANAGER ←(fetch (MESSAGE ORIGINATOR) of xmessage)
       REPORT!RECIPIENTS ←(LIST (fetch (MESSAGE ORIGINATOR) of xmessage))
       TASK ← xtaskname))
    (STORE!OBJECT xpnode 'CONTRACT xcontract)
    (COND
     ((EQUAL (fetch (PNODE STATUS) of xpnode!)
        "Busy")
      (replace (CONTRACT STATE) of xcontract with 'READY)
      (READY!CONTRACT xpnode xcontract))
     (T (replace (CONTRACT STATE) of xcontract with 'EXECUTING)
      (replace (PNODE EXECUTING) of xpnode! with (LIST xcontract))
      (replace (PNODE STATUS) of xpnode! with "Busy")
      (INSTALL!INTERNAL!EVENT (IPLUS time tpdaw)
       xpnode
       (fetch (CONTRACT NAME) of xcontract)
       'CONTRACT!PROCESSING)
    )
  )
  (SENDMESSAGE xpnode (IPLUS time tpdaw tack)
   (fetch (CONTRACT MANAGER) of xcontract)
   (create ACKNOWLEDGEMENT NAME ←(fetch (DIRECTED!AWARD NAME) of xda)
    RESPONSE ←'ACCEPTANCE)))
  (T (SETQ xrefproc COND
       (xtt (fetch (TASK!TEMPLATE REFUSAL!PROCEDURE) of xtt))
       (SETQ xrefjust COND
       (xrefproc (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE xrefproc))
          (LIST xpnode xda))))
       (T NIL))
   (SENDMESSAGE xpnode (IPLUS time tpdaw tack)
    (fetch (MESSAGE ORIGINATOR) of xmessage)
    (create ACKNOWLEDGEMENT NAME ←(fetch (DIRECTED!AWARD NAME) of xda)
     RESPONSE ←'REFUSAL
     REFUSAL!JUSTIFICATION ← xrefjust))
  )
  
Calls: CHECK!ELIGIBILITY INSTALL!INTERNAL!EVENT PARSE!TASK!ABSTRACTION READY!CONTRACT RETRIEVE!OBJECT
SENDMESSAGE STORE!OBJECT STORE!TASK!OBJECT

Called by: PROCESS!MESSAGE

Freevars: NET tack time tnack tpdaw

Explanation: Performs the necessary bookkeeping to handle the receipt of a directed award by node "xpnode".
"xmessage" is the message. If the node meets the eligibility specification for the task and has a task
template for task type mentioned in the task abstraction, then the contract is accepted and affirmatively
acknowledged. Otherwise the contract is refused and a negative acknowledgement is sent to the originator.
The 'refusal!justification' is obtained from the procedure for the task, or set to NIL, if no procedure
exists.

If the node is "Idle", then an event is placed on the event list to begin processing on the new
contract. Otherwise the contract is placed in the 'ready' state.

---36---
rgs: 7-Sep-78 05:47 [CNET]

(PROCESS!DISPLAY!EVENT
 (LAMBDA (d)
  (DISPLAY "From: " (fetch (DISPLAY!EVENT PNODE) of d))
  (DISPLAY)
  (DISPLAY (fetch (DISPLAY!EVENT DATA) of d))
  (DISPLAY))

-----
Called by: DISPLAY!EVENT SIMULATE

Explanation: Displays the data of display event "d" with an indication about its originator.
(PROCESS!FINAL!REPORT

(LAMBDA (xpnode xmessage)
  (PROG (xpnode! pname xcontract xsubcontract xstate xrepaccproc sc nsc xreport tmpsc
  (SETQ xpnode! (ELT NET xpnode))
  (SETQ xreport (fetch (MESSAGE TEXT) of xmessage))
  (SETQ pname (CDR (fetch (FINAL!REPORT NAME) of xreport)))
  (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT pname))
  (COND
    ((NOT (EQUAL xstate 'TERMINATED))
      (SETQ nsc (SUB1 (CAR (fetch (CONTRACT SUBCONTRACTS) of xcontract)))
      (SETQ tmpsc (CAR (SOME (CDR (fetch (CONTRACT SUBCONTRACTS) of xcontract))
        (FUNCTION (LAMBDA (x)
          (EQUAL (fetch (SUBCONTRACT NAME) of x)
            (fetch (FINAL!REPORT NAME) of xreport)))
        for x in (fetch (CONTRACT SUBCONTRACTS) of xcontract)
        do (SETQ tmpsc1 (FIND!SUBCONTRACT xpnode x))
        (replace (SUBCONTRACT PREDECESSORS) of tmpsc1
          with (REMOVE (fetch (SUBCONTRACT NAME) of x)
            (fetch (SUBCONTRACT PREDECESSORS) of tmpsc1)))
      (for x in (fetch (SUBCONTRACT SUCCESSORS) of tmpsc)
        do (SETQ tmpsc1 (FIND!SUBCONTRACT xpnode x))
        (replace (SUBCONTRACT SUCCESSORS) of tmpsc1
          with (REMOVE (fetch (SUBCONTRACT NAME) of x)
            (fetch (SUBCONTRACT SUCCESSORS) of tmpsc1)))
      (RPLACA (fetch (CONTRACT SUBCONTRACTS) of xcontract)
        nsc)
      (RPLACD (fetch (CONTRACT SUBCONTRACTS) of xcontract)
        (REMOVE tmpsc (CDR (fetch (CONTRACT SUBCONTRACTS) of xcontract)))
    (COND
      ((EQUAL nsc 0)
        (replace (CONTRACT SUBCONTRACTS) of xcontract with NIL)))
      (SETQ xrepaccproc (fetch (TASK REPORT!ACCEPTANCE!PROCEDURE) of (RETRIEVE!OBJECT xpnode
        'TASK xcontract))))
  (COND
    (xrepaccproc (replace (CONTRACT RESULTS) of xcontract
      with (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode
        'PROCEDURE xrepaccproc))
      (LIST (fetch (FINAL!REPORT RESULT!DESCRIPTION) of xreport)
        (fetch (CONTRACT RESULTS) of xcontract)))
    (T (replace (CONTRACT RESULTS) of xcontract
      with (COND
        ((fetch (CONTRACT RESULTS) of xcontract)
          (APPEND (fetch (FINAL!REPORT RESULT!DESCRIPTION) of xreport)
            (fetch (CONTRACT RESULTS) of xcontract))))
        (T (fetch (FINAL!REPORT RESULT!DESCRIPTION) of xreport)))
    (COND
      ((EQUAL xstate 'SUSPENDED)
        (SETQ sc (NODE!SEARCH xpnode (fetch (CONTRACT NAME) of xcontract)
          'SUSPENDED T))
        (replace (PNODE READY) of xpnode with (SORT (CONS sc (fetch (PNODE READY) of xpnode))
          'READYCOMPARE))
        (replace (CONTRACT STATE) of (CAR sc) with 'READY)
        (COND
          ((EQUAL (fetch (PNODE STATUS) of xpnode)
            'Idle')
            (NEXT!CONTRACT xpnode 'REPORT))))

----------

Calls: FIND!SUBCONTRACT NEXT!CONTRACT NODE!SEARCH READYCOMPARE RETRIEVE!OBJECT
Called by: PROCESS!MESSAGE
Freevars: NET

-38-
Explanation: Performs the necessary bookkeeping to handle the receipt of a final report by node "xpnode". "xmessage" is the message. If the contract for which the report is intended has not been terminated, then the appropriate subcontract is deleted from the list of subcontracts for the contract. Predecessors and successors are updated. The 'report acceptance procedure' for the contract is used to update the 'results' slot. If no procedure exists then the new result is appended to the previous results.

If the contract is currently in the 'suspended' state, then it is moved to the 'ready' state. The status of the node is checked, and another contract executed if the node is "Idle".

rgs: 11-Oct-78 00:06 (CNET)

(PROCESS!INFORMATION
 (LAMBDA (xpnode xmessage)
 (PROG (xpnode! pname xcontract xstate xinfoaccproc xinfo)
 (SETQ xpnode! (ELT NET xpnode))
 (SETQ xinfo (fetch (MESSAGE TEXT) of xmessage))
 (SETQ pname (fetch (INFORMATION NAME) of xinfo))
 (SETQ xcontract (RETRIEVE OBJECT xpnode 'CONTRACT pname))
 (COND
 (xcontract (SETQ xstate (fetch (CONTRACT STATE) of xcontract))
 (COND
 (NOT (EQUAL xstate 'TERMINATED))
 (SETQ xinfoaccproc (fetch (TASK INFORMATION ACCEPTANCE PROCEDURE) of (RETRIEVE OBJECT xpnode 'TASK xcontract)))
 (COND
 (xinfoaccproc (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE OBJECT xpnode 'PROCEDURE xinfoaccproc))
 (LIST (fetch (INFORMATION INFORMATION SPECIFICATION) of xinfo)
 (T (for x in (fetch (INFORMATION INFORMATION SPECIFICATION) of xinfo)
 do (STORE OBJECT xpnode (CAR x)
 (EVAL (CONS 'create
 (PROG (z)
 (SETQ z (CAR x))
 (for y in (CDR x)
 do (SETQ z (APPEND z
 (CONS (CAR y)
 (CONS → (CDR y)
 (RETURN z))

-----------
 Calls: RETRIEVE OBJECT STORE OBJECT
 Called by: PROCESS MESSAGE
 Freevars: NET
 Explanation:
(PROCESS!INTERIM!REPORT
(LAMBDA (xpnode xmessage) (e rgs: "23-Sep-78 16:45")
(PROG (xpnode! pname xcontract xsubcontract xstate xrepaccproc sc xreport)
  (SETQ xpnode! (ELT NET xpnode))
  (SETQ xreport (fetch (MESSAGE TEXT) of xmessage))
  (SETQ pname (CDR (fetch (INTERIM!REPORT NAME) of xreport)))
  (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT pname))
  (COND
    ((NOT (EQUAL xstate 'TERMINATED))
     (replace (SUBCONTRACT RESULTS) of ICAR (SOME (CDR (fetch (CONTRACT SUBCONTRACTS)
       of xcontract))
       (FUNCTION (LAMBDA (x)
         (EQUAL (fetch (SUBCONTRACT NAME) of x)
         (fetch (INTERIM!REPORT NAME)
         of xreport)'
       with (fetch (INTERIM!REPORT RESULT!DESCRIPTION) of xreport)))
     (SETQ xrepaccproc (fetch (TASK REPORT!ACCEPTANCE!PROCEDURE) of (RETRIEVE!OBJECT xpnode
       'TASK xcontract)))
    (COND
      (xrepaccproc (replace (CONTRACT RESULTS) of xcontract
        with (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode
          'PROCEDURE xrepaccproc))
        (LIST (fetch (INTERIM!REPORT RESULT!DESCRIPTION) of xreport)
        (fetch (CONTRACT RESULTS) of xcontract)
      (T (replace (CONTRACT RESULTS) of xcontract
        with (COND
          (((fetch (CONTRACT RESULTS) of xcontract)
            (APPEND (fetch (INTERIM!REPORT RESULT!DESCRIPTION) of xreport)
            (fetch (CONTRACT RESULTS) of xcontract)))
          (T (fetch (INTERIM!REPORT RESULT!DESCRIPTION) of xreport)
        (COND
          ((EQUAL xstate 'SUSPENDED)
            (SETQ sc (NODE!SEARCH xpnode (fetch (CONTRACT NAME) of xcontract)
              'SUSPENDED T))
            (replace (PNODE READY) of xpnode! with (SORT (CONS sc (fetch (PNODE READY) of xpnode!))
              'READYCOMPARE))
          (replace (CONTRACT STATE) of (CAR sc) with 'READY)
          (COND
            ((EQUAL (fetch (PNODE STATUS) of xpnode!)
              "Idle")
            (NEXT!CONTRACT xpnode 'REPORT))
          (COND
            ((EQUAL (fetch (PNODE STATUS) of xpnode!)
              "Idle")
            (NEXT!CONTRACT xpnode 'REPORT)))
    (COND
      (EQUAL xstate 'SUSPENDED)
      (SETQ sc (NODE!SEARCH xpnode (fetch (CONTRACT NAME) of xcontract)
        'SUSPENDED T))
      (replace (PNODE READY) of xpnode! with (SORT (CONS sc (fetch (PNODE READY) of xpnode!))
        'READYCOMPARE))
      (replace (CONTRACT STATE) of (CAR sc) with 'READY)
      (COND
        ((EQUAL (fetch (PNODE STATUS) of xpnode!)
          "Idle")
        (NEXT!CONTRACT xpnode 'REPORT))
    Calls: NEXT!CONTRACT NODE!SEARCH READYCOMPARE RETRIEVE!OBJECT
    Called by: PROCESS!MESSAGE
    Freevars: NET
    Explanation: Performs the necessary bookkeeping to handle the receipt of an interim report by node "xpnode". "xmessage" is the message. If the contract for which the report is intended has not been terminated, then the 'results' slot of the appropriate subcontract is updated with the new result. If no procedure exists then the new result is appended to the previous results. If the contract is currently in the 'suspended' state, then it is moved to the 'ready' state. The status of the node is checked, and another contract executed if the node is "Idle".)
PROCESS!INTERNAL!EVENT

(LAMBDA (e) (PROG NIL
  (RETURN (SELECTQ (fetch (INTERNAL!EVENT TYPE) of e)
    (CONTRACT!PROCESSING (PROCESS!CONTRACT (fetch (INTERNAL!EVENT PNODE) of e)
      (fetch (INTERNAL!EVENT NAME) of e)
      (fetch (INTERNAL!EVENT DATA) of e)))
    (NODE!UPDATE (UPDATE!NODE (fetch (INTERNAL!EVENT PNODE) of e)
      (fetch (INTERNAL!EVENT NAME) of e)
      (fetch (INTERNAL!EVENT DATA) of e)))
    (BID!CHECK (CHECK!BIDS (fetch (INTERNAL!EVENT PNODE) of e)
      (fetch (INTERNAL!EVENT NAME) of e)))
    (PSEUDO!CONTRACT (DELETE!PSEUDO!CONTRACT (fetch (INTERNAL!EVENT PNODE) of e)
      (fetch (INTERNAL!EVENT NAME) of e)))
    NIL)))

Calls: CHECK!BIDS DELETE!PSEUDO!CONTRACT PROCESS!CONTRACT UPDATE!NODE

Called by: SIMULATE

Explanation: Routes internal event "e" to the appropriate function.
There are currently four types of internal event: 'contract!processing' and 'node!update', that are used
to perform the necessary bookkeeping for task execution; 'bid!check', that is used to assess bids and take
action (if necessary) at the end of the expiration time for a task announcement; and, 'pseudo!contract',
that is used to eliminate (if necessary) the temporary information stored by a node in anticipation of the
receipt of a contract on which a bid has been made.

PROCESS!MESSAGE

(LAMBDA (xpnode xaddressee xmessage) (PROG NIL
  (COND
    ((OR (EQUAL xaddressee "*")
      (EQUAL xpnode xaddressee)
      (MEMBER xpnode xaddressee))
      (SELECTQ (CAR (fetch (MESSAGE TEXT) of xmessage))
        (TASK!ANNOUNCEMENT (PROCESS!TASK!ANNOUNCEMENT xpnode xmessage))
        (BID (PROCESS!BID xpnode xmessage))
        (ANNOUNCED!AWARD (PROCESS!ANNOUNCED!AWARD xpnode xmessage))
        (DIRECTED!AWARD (PROCESS!DIRECTED!AWARD xpnode xmessage))
        (ACKNOWLEDGEMENT (PROCESS!ACKNOWLEDGEMENT xpnode xmessage))
        (TERMINATION (PROCESS!TERMINATION xpnode xmessage))
        (REQUEST (PROCESS!REQUEST xpnode xmessage))
        (INFORMATION (PROCESS!INFORMATION xpnode xmessage))
        (NODE!AVAILABILITY!ANNOUNCEMENT (PROCESS!NODE!AVAILABILITY!ANNOUNCEMENT xpnode xmessage))
        NIL))

Calls: PROCESS!ACKNOWLEDGEMENT PROCESS!ANNOUNCED!AWARD PROCESS!BID PROCESS!DIRECTED!AWARD PROCESS!FINAL!REPORT
PROCESS!INFORMATION PROCESS!INTERIM!REPORT PROCESS!NODE!AVAILABILITY!ANNOUNCEMENT PROCESS!REQUEST
PROCESS!TASK!ANNOUNCEMENT PROCESS!TERMINATION

Called by: SIMULATE

Freevars: XMESSAGE XPNODE

Explanation: Routes the message "xmessage" to node "xpnode" if it is one of the addressees (or the message is a
broadcast). The addressee is "xaddressee".
The message is routed to the appropriate function. There is a function to receive each of the messages
of the contract net protocol.
(PROCESS!NODE!AVAILABILITY!ANNOUNCEMENT

(LAMBDA (xpnode xmessage) NIL)

Called by: PROCESS!MESSAGE

Explanation:

rgs: 10-Oct-78 23:27 [CNET]

(PROCESS!REQUEST

(LAMBDA (xpnode xmessage)
  (PROG (xpnode! pname xrequest xreqspec xobject xinfo)
    (SETQ xpnode! (ELT NET xpnode))
    (SETQ xrequest (fetch (MESSAGE TEXT) of xmessage))
    (SETQ pname (fetch (REQUEST NAME) of xrequest))
    (SETQ xreqspec (fetch (REQUEST REQUEST!SPECIFICATION) of xrequest))
    (SETQ xinfo (for x in xreqspec collect xobject when (SETQ xobject (RETRIEVE!OBJECT xpnode (CAR x) (CADDR x) (CADR x))
     (COND
       (xinfo (SENDMESSAGE xpnode time (fetch (MESSAGE ORIGINATOR) of xmessage)
       (create INFORMATION NAME ← pname INFORMATION!SPECIFICATION ← xinfo))
     )

Calls: RETRIEVE!OBJECT SENDMESSAGE

Called by: PROCESS!MESSAGE

Freevars: NET time

Explanation:
PROCESS!TASK!ANNOUNCEMENT

(SETQ active (fetch (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xnode))
(SETQ xta (fetch (MESSAGE TEXT) of xmessage))
(UPDATE!ACTIVE!TASK!ANNOUNCEMENTS xnode)
(SETQ xabs (PARSE!TASK!ABSTRACTION (fetch (TASK!ANNOUNCEMENT TASK!ABSTRACTION) of xta)))
(SETQ xtt (RETRIEVE!OBJECT xnode 'TASK!TEMPLATE (CAR xabs)))

COND

(AND xtt OR (NOT (fetch (TASK!ANNOUNCEMENT ELIGIBILITY!SPECIFICATION) of xta))
(CHECK!ELIGIBILITY xnode (fetch (TASK!ANNOUNCEMENT ELIGIBILITY!SPECIFICATION) of xta)
(SAMETYPE (CAR (SOME active (FUNCTION (LAMBDA (x)
(EQUAL (fetch (ACTIVE!TASK!ANNOUNCEMENT TYPE) of x)
(CAR xabs)
(COND

(* if there is an active task announcement of the same type as the new task announcement then get
the announcement!ranking!procedure and apply it to determine if the old active task announcement
should be replaced -
if there is no ranking procedure then keep the old active task announcement)

(SETQ xarankproc (fetch (TASK!TEMPLATE ANNOUNCEMENT!RANKING!PROCEDURE) of xtt))
(COND

(* the announcement!ranking!procedure is passed the parsed abstraction for the new task announcement
and the parsed task abstraction for the old active task announcement of the same type -
it returns 1 if the new announcement is "better", 0 if the two are equally important, and -1 if the
old active task announcement is "better" -
the current default if they are equally important is to retain the old active task announcement)

[SETQ rank (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xnode 'PROCEDURE xarankproc))

(List (CADR xabs)
(fetch (ACTIVE!TASK!ANNOUNCEMENT ABSTRACTION) of sametype)

(COND

(EQUAL rank 1)

(SETQ xtal (create ACTIVE!TASK!ANNOUNCEMENT MANAGER -(fetch (MESSAGE ORIGINATOR) of xmessage)
CONTRACT -(fetch (TASK!ANNOUNCEMENT NAME) of xta)
TYPE -(CAR xabs)
ABSTRACTION -(CADR xabs)
BID!SPECIFICATION -(fetch (TASK!ANNOUNCEMENT BID!SPECIFICATION) of xta)
TIME -(fetch (MESSAGE TIME) of xmessage)
EXPIRATION!TIME -(fetch (TASK!ANNOUNCEMENT EXPIRATION!TIME) of xta)))

(replace (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xnode!
with (SUBST xtal sametype active)
with (CONS xtal (fetch (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xnode))
(SET xtal (create ACTIVE!TASK!ANNOUNCEMENT MANAGER -(fetch (MESSAGE ORIGINATOR) of xmessage)
CONTRACT -(fetch (TASK!ANNOUNCEMENT NAME) of xta)
TYPE -(CAR xabs)
ABSTRACTION -(CADR xabs)
BID!SPECIFICATION -(fetch (TASK!ANNOUNCEMENT BID!SPECIFICATION) of xta)
TIME -(fetch (MESSAGE TIME) of xmessage)
EXPIRATION!TIME -(fetch (TASK!ANNOUNCEMENT EXPIRATION!TIME) of xta)))

(replace (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xnode!
with (CONS xtal (fetch (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xnode))

-43-
(COND
  ((EQUAL (fetch (PNODE STATUS) of xpnode!) "Idle")
   (MAKE!BID xpnode)))

**Calls:** CHECK!ELIGIBILITY MAKE!BID PARSE!TASK!ABSTRACTION RETRIEVE!OBJECT UPDATE!ACTIVE!TASK!ANNOUNCEMENTS

**Called by:** PROCESS!MESSAGE

**Freevars:** NET

**Explanation:** Performs the necessary bookkeeping to handle the receipt of a task announcement by node "xpnode". "xmessage" is the message. If the node meets the eligibility specification of the task then the task announcement is ranked relative to other currently active task announcements. If there are other announcements of the same type (as specified by the task abstraction), then the 'announcement!ranking!procedure' is used to rank them. If there is no procedure, then the old announcement is kept, and the new one is discarded. If there are no other announcements of the same type then the new announcement is consed to the list of other announcements.

The announcement!ranking!procedure returns +1, 0, or -1. +1 indicates that the new announcement should be kept, -1 that the old announcement should be kept, and 0 if the two are equally good (the current default is to keep the old one in this case).

If the node is "Idle", then a bid is made on the current best task announcement.
PROCESS!TERMINATION

(LAMBDA (xpnode xmessage)
  (PROG (xpnode! xterm xcontract xtn xtt xtermproc)
    (SETQ xpnode! (ELT NET xpnode))
    (SETQ xterm (fetch (MESSAGE TEXT) of xmessage))
    (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT (fetch (TERMINATION NAME) of xterm)))
    (COND
      ((NOT (EQUAL (fetch (CONTRACT STATE) of xcontract) 'TERMINATED))
       (REPLACE (PNODE TERMINATED) of xpnode! with (CONS xcontract (fetch (PNODE TERMINATED) of xpnode!)))
       (REPLACE (CONTRACT STATE) of xcontract with 'TERMINATED)
       (COND
         ((ILESSP ntermcs (LENGTH (fetch (PNODE TERMINATED) of xpnode!)))
           (REVERSE (fetch (PNODE TERMINATED) of xpnode!))
           (SETQ xtn (fetch (CONTRACT TASK) of (CAR (fetch (PNODE TERMINATED) of xpnode!)))
           (SETQ xta (RETRIEVE!OBJECT xpnode 'TASK xtn))
           (SETQ xtermproc (fetch (TASK TERMINATION!PROCEDURE)
             of (RETRIEVE!OBJECT xpnode 'TASK (fetch (CONTRACT TASK) of (CAR (fetch (PNODE TERMINATED) of xpnode!)))))
           (COND
             (xtermproc (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE xtermproc))
               (LIST xpnode (CAR (fetch (PNODE TERMINATED) of xpnode!)))
               (T (SETQ xtn (RETRIEVE!OBJECT xpnode 'TASK!TEMPLATE (fetch (TASK TYPE) of xtn))
                (REPLACE (TASK!TEMPLATE TASKS) of xtn with (REMOVE xtn (fetch (TASK!TEMPLATE TASKS) of xtn)))
                (DELETE!OBJECT xpnode 'TASK xtn)
                (DELETE!OBJECT xpnode 'CONTRACT (fetch (CONTRACT NAME) of (CAR (fetch (PNODE TERMINATED) of xpnode!))))
                (INSTALL!DISPLAY!EVENT (IPLUS time tpt)
                  xpnode 'SIMULATION
                  (APPEND '(Terminated Contract)
                    (fetch (TERMINATION NAME) of xterm)))
                (e now terminate the subcontracts)
                (TERMINATE!SUBCONTRACTS xpnode xcontract tpt)
               )))
         ))
      )))
    )))
  )))
)

QANNOUNCE

(LAMBDA (xpnode xspecification)
  (LAMBDA (xpnode xspecification xspecification)
    (LIST "x" NIL (LIST 'EXTEND!BOARD (SUB1 (fetch (BOARD COLUMN) of xspecification)))
    NIL xtral)
)

Explanation: Performs the necessary bookkeeping to handle the receipt of a termination by node "xpnode". "xmessage" is the message. If the contract named in the message has not already been terminated then it is placed in the terminated state, and all of its outstanding subcontracts are terminated.

If the terminated state contains more than 'ntermcs' contracts then the eldest contract is discarded, after presenting it to the 'termination!procedure' for its task (if such a procedure exists).
QARANK

(LAMBDA (xabsl xabs2)

(PROG (rn)
   (RETURN (SELECTQ qsearch!strategy
     (0 (COND
         ((IGREATERP xabsl xabs2) 1)
         ((IGREATERP xabs2 xabsl) -1)
         (T 0)))
     (1 (COND
         ((IGREATERP xabs2 xabsl) 1)
         ((IGREATERP xabsl xabs2) -1)
         (T 0)))
     (2 (SETQ rn (RAND -1.0 1.0))
         (COND
             ((FGREATERP rn 0.0) 1)
             ((MINUSP rn) -1)
             (T 0)))
     NIL))

Freevars: qsearch!strategy

Explanation: Orders two 'extend board' task abstractions, "xabsl" (the abstraction for the 'new' announcement) and "xabs2" (the abstraction for the current best announcement). Returns +1, -1, or 0 according to the search strategy determined in QSET!PARAMETERS.

QBRANK

(LAMBDA (newbid oldbids)
   (LIST 'SATISFACTORY newbid))

Explanation: Handles bids received by a node. "newbid" is the 'new' active!bid, and "oldbids" is the list of previously received active!bids. QBRANK always returns a list of 'satisfactory and the new active!bid.

QDISPLAY

(LAMBDA (xboard)
   (CONS "Queen-rows:" (for i from 1 to qsize collect (ELT xboard i) when (IGREATERP (ELT xboard i) 0))

Called by: EXTEND!BOARD QFINALIZE

Freevars: qsize

Explanation: Returns a list of rows in which queens have been placed on the board "xboard".
(QFINALIZE)
  (LAMBDA (xnode xname xrslt)
    (PROG (solutions msg)
      (setq solutions (for x in xrslt collect x when (equal (car x) 'SUCCESS))
      (cond
        (solutions (setq msg (list "Solutions Found:
          (for x in solutions do (setq msg (append msg (QDISPLAY (CDADDR x)))
        (t (QDISPLAY "No Solutions Found"))))
      Calls: QDISPLAY
      Explanation: The 'final' function for the N Queens problem. Displays the solutions found (or that no solutions have been found).
      "xnode" is the node that sent the top-level report. "xname" is the name of the contract. "xrslt" is the text of the report.

rgs: 16-Oct-78 21:56
(QINITIALIZE)
  (LAMBDA (xnetsize restartflag olduserparamflag)
    (PROG (xprocedure xannproc xarankproc xbrankproc xtask!template)
      (cond
        (not restartflag)
        (setq parameters (not olduserparamflag)
          (setq xprocedure (create procedure name ← 'EXTEND!BOARD
          code ← 'EXTEND!BOARD))
          (setq xannproc (create procedure name ← 'QANNOUNCE
          code ← 'QANNOUNCE))
          (setq xarankproc (create procedure name ← 'QARANK
          code ← 'QARANK))
          (setq xbrankproc (create procedure name ← 'QBRANK
          code ← 'QBRANK))
          (setq xtask!template (create task!template type ← 'EXTEND!BOARD
          announcement!procedure ← 'QANNOUNCE
          announcement!ranking!procedure ← 'QARANK
          bid!ranking!procedure ← 'QBRANK
          execution!procedure ← 'EXTEND!BOARD))
        (for x from 1 to xnetsize do (store!object x 'procedure xprocedure)
          (store!object x 'procedure xannproc)
          (store!object x 'procedure xarankproc)
          (store!object x 'procedure xbrankproc)
          (store!object x 'task!template (copyall xtask!template)))
      (return (list (list 'EXTEND!BOARD (new!board)
      Calls: NEW!BOARD QSET!PARAMETERS STORE!OBJECT
      Explanation: The 'initial' function for the N Queens problem. Initializes the knowledge bases of the nodes in the net with the required task!templates and procedures. Returns a list of the top-level task type and the initial board (no queens placed).
      "xnetsize" is the number of nodes in the net. "restartflag" is T if new parameters are not to be requested. "olduserparamflag" is T if the current user parameters are to be used as defaults when new user parameters are requested.
(QRECEIVE (xpnode xname xcontract) (solutions)
  (COND ((EQUAL qreport!strategy 0)
            (while (AND (OUTSTANDING!SUBCONTRACTS xcontract)
                       (ILESSP (LENGTH (SETQ solutions (for x in (fetch (CONTRACT RESULTS) of xcontract)
                              collect x when (EQUAL (CAR x) 'SUCCESS)
                              qnsol)))
                   do (CNET* 'INTERIM!REPORT (LIST (LIST (CAR (fetch (CONTRACT RESULTS) of xcontract)
                                                    SUSPEND))
                        (CNET* 'FINAL!REPORT (LIST (LIST (CAR (fetch (CONTRACT RESULTS) of xcontract)
                                                   (TERMINATE)))
                        (T (while (OUTSTANDING!SUBCONTRACTS xcontract) do (SUSPEND))
                          (SETQ solutions (for x in (fetch (CONTRACT RESULTS) of xcontract) collect x when (EQUAL (CAR x) 'SUCCESS)
                          (COND (solutions (CNET* 'FINAL!REPORT (LIST solutions)))
                                 (T (CNET* 'FINAL!REPORT (LIST (LIST (CAR (fetch (CONTRACT RESULTS) of xcontract)
                                                      (TERMINATE))))

----------
Calls:  CNET* OUTSTANDING!SUBCONTRACTS SUSPEND TERMINATE
Called by:  EXTEND!BOARD
Freevars:  qnsol qreport!strategy
Explanation:  Actually decides what to do upon receipt of a report for the N Queens problem.  "xpnode" is the node receiving the report, "xname" is the name of the contract, "xcontract" is the contract record.

----------
(QSET!PARAMETERS (cleanstart) (NIL)
  (COND (cleanstart (PROG NIL
                     (COND (cleanstart (PROG NIL
                                (SETQQ qsize 5)
                                (SETQQ qnsol 1)
                                (SETQQ qsearch!strategy 0)
                                (SETQQ qreport!strategy 0)
                                (SETQQ tqgenerate 1)
                                (SETQQ tqsubtask 1)
                                (SETQQ tqsuccess 1)
                                (SETQQ tqfailure 1)
                                (TTYOUT))
                                (SETQ qsize (ASKFORNUMBER "Number of Queens" qsize 'QSIZE 0))
                                (SETQ qsearch!strategy (ASKFORNUMBER "Search Strategy" qsearch!strategy 'QSEARCH -1 3))
                                (SETQ qreport!strategy (ASKFORNUMBER "Report Strategy" qreport!strategy 'QREPORT -1 2))
                                (COND ((EQUAL qreport!strategy 0)
                                        (SETQ qnsol (ASKFORNUMBER "Number of solutions" qnsol 'QNSOL 0)
                                        (TTYOUT)))

----------
Called by:  QINITIALIZE
Freevars:  qnsol qreport!strategy qsearch!strategy qsize tqfailure tqgenerate tqsubtask tqsuccess
Explanation:  Asks user for parameters for the N Queens problem. Sets global variables. Same style as SET!PARAMETERS.
If "cleanstart" is T then the settings built into the function are used as defaults for the questions.
RANDOMCOMPARE

(RANDOMCOMPARE
 ILAMBDA (a b)
 (COND
  ((FGREATERP (RAND -1.0 1.0) 0.0)
   T)
  (T NIL))
)

Called by: SIMULATE

Explanation: Orders two items "a", and "b" according to a random number between -1 and +1.

READY!CONTRACT

(REady CONTRACT
 ILAMBDA (xpnode xcontract xpointer)
 (PROG (xpnode!)
   (SETQ xpnode! (ELT NET xpnode))
   (RETURN (replace (PNODE READY) of xpnode with (COND
    ((fetch (PNODE READY) of xpnode!)
     (NCONC (fetch (PNODE READY) of xpnode!))
    (T LIST (CONS xcontract xpointer))))
     (T LIST (CONS xcontract xpointer)))
   )
)

Called by: PROCESS!ANNOUNCED!AWARD PROCESS!DIRECTED!AWARD

Freevars: NET

Explanation: The contract with name "xname" is placed at the end of the list of contracts in the 'ready' state of node "xpnode". "xpointer" can be a pointer to the task execution procedure, if READY!CONTRACT is called to ready a suspended contract.

READYCOMPARE

(READYCOMPARE
 ILAMBDA (a b)
 (COND
  ((AND (CDR a) (CDR b))
   (COND
    ((ILESSP (LENGTH (fetch (CONTRACT NAME) of (CAR a))) (LENGTH (fetch (CONTRACT NAME) of (CAR b))
      T)
    (T NIL))
    ((AND (CDR a) (NOT (CDR b)))
     T)
    ((AND (NOT (CDR a)) (CDR b))
     NIL)
    (T T))
  )
)

Called by: PROCESS!FINAL!REPORT PROCESS!INTERIM!REPORT

Explanation: Orders two contracts in the ready state, "a", and "b". The ordering is such that resumed contracts have priority over newly acquired contracts, and the older resumed contracts have priority over the newer ones.
REANNOUNCE!TASK

(REANNOUNCE!TASK
  (LAMBDA (xpnode xname)
    (PROG NIL
      (SETQ tracounter (ADD1 tracounter))
      (ANNOUNCE!TASK xpnode xname)))

Calls: ANNOUNCE!TASK
Called by: CHECK!BIDS
Freevars: tracounter

RELEASE!TASK

(RELEASE!TASK
  (LAMBDA (taskprocesspointer)
    (TRYNEXT taskprocesspointer NIL 'RELEASE)))

Called by: UPDATE!NODE

Explanation: Used to release the pointer to a task execution function. It does this by calling up the function with the pointer "taskprocesspointer" with TRYNEXT, and passing the keyword 'RELEASE. The function that always catches this keyword is CNET*, and it performs an 'ADIEU. This function is used to release the pointer to a task when the associated contract is terminated by the manager.

RESIMULATE

(RESIMULATE
  (LAMBDA NIL
    (PROG NIL
      (TTYOUT)
      (SETQ resimulateflag (ASKFORYESNO "Another task" resimulateflag 'RESTART))
      (COND
        ((NOT resimulateflag)
          (COND
            (fileflag (CLOSEF cnetfile)))
            (RETURN 0))
        (T (SETQ sameparameterflag (ASKFORYESNO "Same parameters" sameparameterflag 'RESTARTPARAMS))
          (COND
            (NOT sameparameterflag)
              (COND
                (fileflag (CLOSEF cnetfile)))
                (RETURN 2))
            (T (RETURN 1)))))

Called by: CNET
Freevars: cnetfile fileflag resimulateflag sameparameterflag

Explanation: Asks the user questions about doing another simulation. Returns 0 if the user doesn't want another simulation to be done. Returns 1 if another simulation is to be done with the same parameters. Returns 2 if another simulation is to be done with different parameters.
(RESUME! TASK
  (LAMBDA (taskprocesspointer)
    (SETQ task!time 0)
    (TRYNEXT taskprocesspointer))

Called by: PROCESS! CONTRACT UPDATE! NODE

Freevars: task!time

Explanation: Resumes the task with pointer "taskprocesspointer" with TRYNEXT. Also initializes the 'task!time'.

(RETRIEVE! OBJECT
  (LAMBDA (xpnode xobject xkey xslot everyflag)
    (PROG (xpnode! kb index otherindex)
      (SETQ xpnode! (ELT NET xpnode))
      (SETQ kb (fetch (PNODE KNOWLEDGE! BASE) of xpnode!))
      (COND
        [(MEMBER xobject (RECORDFIELDNAMES 'KNOWLEDGE! BASE))
         (SETQ index (RECORDACCESS xobject kb (RECLOOK 'KNOWLEDGE! BASE))
          (RETURN (COND
            [(xkey (CAR (SOME index (FUNCTION (LAMBDA (x)
              (EQUAL x xkey)
            ) (x))))
             (T (RECORDACCESS xslot x (RELOCK xobject) xkey))
             (T (CAR index))
             (T (SETQ index (fetch (KNOWLEDGE! BASE OTHER) of kb))
              (SETQ otherindex (CDAR (SOME index (FUNCTION (LAMBDA (x)
                (EQUAL (CAR x) xkey)
              ) (x))))
              (RETURN (COND
                [(xkey (CAR (SOME otherindex (FUNCTION (LAMBDA (x)
                  (EQUAL (x)))
                ) (x))))
                (T (RECORDACCESS xslot x (RELOCK xobject) xkey))
                (T (CAR otherindex))
                (T (CAR otherindex)))])))]]
      ))


Freevars: NET

Explanation: Returns the record for the object of type "xobject" in node "xpnode". "xkey" is the key used to find the object. "xsplot" is the name of the slot to which the key belongs. If "xsplot" is NIL then the first slot for the object is used.
(SAME!STATUS!CHECK
  (LAMBDA (xpnode xname)
    (if (SAME!STATUS!CHECK xpnode xname)
      (INSTALL!DISPLAY!EVENT time xpnode 'TASK xdata)
    ))

RGS: "7-Sep-78 06:02"

CALLED BY: FINAL!REPORT GENERATE!SUBTASK INTERIM!REPORT SDISPLAY UPDATE!NODE

FREEVARS: NET

EXPLANATION: Returns T if "xpnode" is "Busy", and the contract with name "xname" is being executed.

(SENDMESSAGE
  (LAMBDA (xoriginator xtime xaddressee xtext)
    (PROG (m)
      (SETQ m (create MESSAGE TIME ← xtime ADDRESSSEE ← xaddressee ORIGINATOR ← xoriginator TEXT ← xtext))
      (INSTALL!EVENT xtime m))
  ))

RGS: "12-Sep-78 01:02"

CALLED BY: ANNOUNCE!TASK AWARD BID DIRECTED!AWARD FINAL!REPORT INTERIM!REPORT PROCESS!DIRECTED!AWARD PROCESS!REQUEST TERMINATE!SUBCONTRACTS

EXPLANATION: Sends a message from "xoriginator" to "xaddressee" at time "xtime". "xtext" is the text of the message. The message is sent by placing a message event on the event list.
(SETPARAMETERS
(LAMBDA (cleanstart)
(PROG NIL
(COND
(cleanstart (PROG NIL
(SETQ netsize 10)
(SETQ gain 1000)
(SETQ nterms 10)
(SETQ dpflag T)
(SETQ delayfile cnet.delay)
(SETQ ta 1)
(SETQ tra 1000)
(SETQ tpa 1)
(SETQ tna 1)
(SETQ tpna 1)
(SETQ tb 1)
(SETQ tpb 1)
(SETQ tsaw 1)
(SETQ tspaw 1)
(SETQ tdaw 1)
(SETQ tdpaw 1)
(SETQ tack 1)
(SETQ track 1)
(SETQ tr2 1)
(SETQ tpr 1)
(SETQ tt 1)
(SETQ tpt 1)
(SETQ treq 1)
(SETQ tpreq 1)
(SETQ ti 1)
(SETQ tpi 1)
(SETQ display!parameter!flag T)
(SETQ display!statistics!flag T)
(SETQ display!banners!flag T)
(SETQ display!time!flag T)
(SETQ display!messages!flag NIL)
(SETQ display!internal!events!flag NIL)
(SETQ display!display!flag NIL)
(SETQ display!nodelflag NIL)
(SETQ fileflag NIL)
(SETQ cnetfile cnet.results)
(SETQ termflag T)
(SETQ resimulateflag T)
(SETQ sameparameterflag T)
(SETQ initial!function $INITIALIZE$
(SETQ final!function $FINALIZE$
(SETQ randstart (RANDSET T)

(TTYOUT)
(SETQ netsize (ASKFORNUMBER "Nodes" netsize 'NETSIZE 0))
(SETQ nodelist (for i from 1 to netsize collect i))
(SETQ gain (ASKFORNUMBER "Task time expansion factor" gain 'GAIN 0))
(SETQ nterms (ASKFORNUMBER "Terminated contracts" nterms 'TERMCS -1))
(SETQ dpflag (ASKFORSNO "Default delay parameters" dpflag 'DELAY))
(ICOND
(NOT dpflag)
(SETQ dpflag (ASKFORSNO "Read parameters from a file" dpflag 'DELAYFILE))
(ICOND
(dpflag (SETQ delayfile (ASKFORFILENAME 'DELAYFILE))
(INPUT (INFLE delayfile))
(SETQ ta (REDO delayfile))
(SETQ tra (REDO delayfile))
(SETQ tpa (REDO delayfile))
(SETQ tna (REDO delayfile))
(SETQ tpna (REDO delayfile))
(SETQ tb (REDO delayfile))
(SETQ tpb (REDO delayfile))

-53-
(SETQ tsaw (READ delayfile))
(SETQ tpsaw (READ delayfile))
(SETQ tdaw (READ delayfile))
(SETQ tpdaw (READ delayfile))
(SETQ tack (READ delayfile))
(SETQ tpack (READ delayfile))
(SETQ tr2 (READ delayfile))
(SETQ tpr (READ delayfile))
(SETQ tt (READ delayfile))
(SETQ tpt (READ delayfile))
(SETQ treq (READ delayfile))
(SETQ tpreq (READ delayfile))
(SETQ ti (READ delayfile))
(SETQ tpi (READ delayfile))
(CLOSEF delayfile))

(T (SETQ ta (ASKFORNUMBER "ta" ta 'TA 0))
(SETQ tra (ASKFORNUMBER "tra" tra 'TRA 0))
(SETQ tpa (ASKFORNUMBER "tpa" tpa 'TPA 0))
(SETQ tna (ASKFORNUMBER "tna" tna 'TNA 0))
(SETQ tpna (ASKFORNUMBER "tpna" tpna 'TPNA 0))
(SETQ tb (ASKFORNUMBER "tb" tb 'TB 0))
(SETQ tpb (ASKFORNUMBER "tpb" tpb 'TPB 0))
(SETQ tsaw (ASKFORNUMBER "tsaw" tsaw 'TSAW 0))
(SETQ tpsaw (ASKFORNUMBER "tpsaw" tpsaw 'TPSAW 0))
(SETQ tdaw (ASKFORNUMBER "tdaw" tdaw 'TDAW 0))
(SETQ tpdaw (ASKFORNUMBER "tpdaw" tpdaw 'TPDAW 0))
(SETQ tack (ASKFORNUMBER "tack" tack 'TACK 0))
(SETQ tpack (ASKFORNUMBER "tpack" tpack 'TPACK 0))
(SETQ tr2 (ASKFORNUMBER "tr2" tr2 'TR2 0))
(SETQ tpr (ASKFORNUMBER "tpr" tpr 'TPR 0))
(SETQ tt (ASKFORNUMBER "tt" tt 'TT 0))
(SETQ tpt (ASKFORNUMBER "tpt" tpt 'TPT 0))
(SETQ treq (ASKFORNUMBER "treq" treq 'TREQ 0))
(SETQ tpreq (ASKFORNUMBER "tpreq" tpreq 'TPREQ 0))
(SETQ ti (ASKFORNUMBER "ti" ti 'TI 0))
(SETQ tpi (ASKFORNUMBER "tpi" tpi 'TP1 0))

(SETQ display!parameter!flag (ASKFORYESNO "Display Parameters" display!parameter!flag 'DPARAM))
(SETQ display!statistics!flag (ASKFORYESNO "Display statistics" display!statistics!flag 'DSTAT))
(SETQ display!banners!flag (ASKFORYESNO "Display banners" display!banners!flag 'DBAN))
(SETQ display!time!flag (ASKFORYESNO "Display time" display!time!flag 'DTIME))
(SETQ display!messages!flag (ASKFORYESNO "Display messages" display!messages!flag 'DMESS))
(SETQ display!internal!events!flag (ASKFORYESNO "Display internal events" display!internal!events!flag 'DINT))

(SETQ display!display!events!flag (ASKFORYESNO "Display display events" display!display!events!flag 'DO DISPLAY))
(SETQ display!node!events!flag (ASKFORYESNO "Display nodes" display!node!events!flag 'DNODE))
(SETQ display!events!flag (OR display!statistics!flag display!banners!flag display!time!flag display!messages!flag display!internal!events!flag))
(SETQ file!flag (ASKFORYESNO "Diagnostic information to file" file!flag 'DFILE))
(SETQ initial!function (ASKFORYESNO "Initial Applications Function" initial!function 'INITIALIZE))
(SETQ final!function (ASKFORYESNO "Final Applications Function" final!function 'FINALIZE))
(COND
  (file!flag (SETQ cnetfile (ASKFORYESNO "OUTPUT cnetfile")))
  (FILEOUT cnetfile)))

(RANDSET randstart)

(TTYOUT))

Called by: CNET

Freevars: cnetfile delayfile displaybanners!flag display!display!events!flag display!events!flag display!internal!events!flag display!messages!flag display!node!flag display!parameter!flag display!statistics!flag display!time!flag dpf!flag dpflag file!flag final!function gain initial!function netsize model!list ntermcs randstart resimulate!flag same!parameter!flag ta tack tb tdaw termflag ti tna tpa tpack tpb tpdaw tpi tpn ta tpr tpsaw tpt tr2 tra treq tpsaw tpt tr2 tra treq tsaw tt
Explanation: Asks the user for parameter settings for the simulation. Sets global variables. All questions give a prompt, have a default, and respond to "?" with a help message. If "cleanstart" is T then the settings built into the function are used as defaults for the questions.
(SIMULATE)

(LAMBDA (restartflag olduserparamflag)

(PRDC (xnode xnode! eventflag ev evdata xaddressee delta newtimeflag initial!tasks xcontract)

(COND ((display!parameter!flag (DISPLAY)

(DISPLAY) (DISPLAY)$PARAMETERS) (DISPLAY))

(COND ((display!banners!flag (DISPLAY)

(DISPLAY) "Start of Simulation" (DISPLAY))

(INITIALIZE)

(SETQ initial!tasks (APPLY initial!function (LIST netsize restartflag olduserparamflag))))

(SETQ xnode 0)

(do (SETQ xnode! (ADD1 xnode))

(SETQ xnode! (ELT NET xnode))

(SETQ xcontract (create CONTRACT NAME ←(LIST xnode)

MANAGER ← 0 REPORT!RECIPIENTS ←(LIST 0)

TASK ←(STORE!TASK!OBJECT xnode! (CAAR initial!tasks)

(CDAR initial!tasks)

(STORE!OBJECT xnode! 'CONTRACT xcontract)

(replace (PNODE EXECUTING) of xnode! with (LIST xcontract))

(replace (PNODE STATUS) of xnode! with "Busy")

(INSTALL!INTERNAL!EVENT 0 xnode! xcontract!PROCESSING)

(SETQ initial!tasks (CDR initial!tasks)) until (NULL initial!tasks)

(SETQ eventflag NIL)

(SETQ ev (NEXT!EVENT))

(SETQ eventcounter (ADD1 eventcounter))

(COND ((IGREATERP (fetch (EVENT TIME) of ev)

time)

(SETQ delta (DIFFERENCE (fetch (EVENT TIME) of ev)

(time))

(for xnode from 1 to netsize do (COND

((EQUAL (fetch (PNODE STATUS) of (ELT NET xnode))

"Busy")

(SETA utilization xnode (IPLUS (ELT utilization xnode)

delta)

(SETQ time (fetch (EVENT TIME) of ev))

(SETQ newtimeflag T))

(T (SETQ newtimeflag NIL))))

(COND (newtimeflag (COND

(display!time!flag (DISPLAY)

(DISPLAY) "Time: " time)

(DISPLAY)))

(COND (display!node!flag (DISPLAY)

(DISPLAY) "-- Node Status --")

(DISPLAY)

(for xnode from 1 to netsize do (DISPLAY!NODE xnode)

(COND ((display!events!flag (DISPLAY!EVENT ev)))

(SELECTQ (CAR (fetch (EVENT DATA) of ev))

(DISPLAY!EVENT (COND

((EQ (fetch (DISPLAY!EVENT TYPE) of (fetch (EVENT DATA) of ev))

'TASK)

(PROCESS!DISPLAY!EVENT (fetch (EVENT DATA) of ev)

[INTERNAL!EVENT (SETQ eventflag (PROCESS)INTERNAL!EVENT (fetch (EVENT DATA) of ev)

MESSAGE (SETQ eventflag T)

(SETQ evdata (fetch (EVENT DATA) of ev))

(SETQ xaddressee (fetch (MESSAGE ADDRESEEE) of evdata))

))

-56-
(COND
  ((EQUAL xaddressee '(0))
   (APPLY final!function (LIST (fetch (MESSAGE ORIGINATOR) of evdata)
     (fetch (FINAL!REPORT NAME)
       (fetch (MESSAGE TEXT) of evdata)
       (fetch (FINAL!REPORT RESULT!DESCRIPTION)
         (fetch (MESSAGE TEXT) of evdata)
     )
   )
   (T (SETQ messagecounter (ADD1 messagecounter))
     (if (EQUAL evdata:MESSAGE.ADDRESSEE "*"
       then (SETQ bdcstcounter (ADD1 bdcstcounter))
     (SETQ nodelist (SORT nodelist 'RANDOMCOMPARE)
       (SELECTQ (CAR evdata:MESSAGE.TEXT)
         TASK!ANNOUNCEMENT (SETQ tacounter (ADD1 tacounter))
       (BID (SETQ bidcounter (ADD1 bidcounter)))
       (ANNOUNCED!AWARD (SETQ aacounter (ADD1 aacounter)))
       (DIRECTED!AWARD (SETQ dacounter (ADD1 dacounter)))
       (ACKNOWLEDGEMENT (if (EQUAL evdata:MESSAGE.TEXT:ACKNOWLEDGEMENT.RESPONSE
         'ACCEPTANCE)
           then (SETQ acccounter (ADD1 acccounter))
         else (SETQ recounter (ADD1 recounter))
       (INTERIM!REPORT (SETQ ircounter (ADD1 ircounter)))
       (FINAL!REPORT (SETQ frcounter (ADD1 frcounter)))
       (TERMINATION (SETQ tecounter (ADD1 tecounter)))
       (NGCE!AVAILABILITY!ANNOUNCEMENT (SETQ nacounter (ADD1 nacounter)))
       (REQUEST (SETQ rqcounter (ADD1 rqcounter)))
       (INFORMATION (SETQ imcounter (ADD1 imcounter)))
       NIL)
     (for xpnode in nodelist do (PROCESS!MESSAGE xpnode xaddressee evdata)
       NIL)
   (COND
     (eventflag (SETQ rtime time))
     until (NOT (fetch (EVENT LLINK) of eventlist)))
   (COND
     (display!banners!flag (DISPLAY))
     (DISPLAY)
     (DISPLAY "::::::::::::::::::::::::::: End of Simulation :::::::::::::::::::::::::::")
     (DISPLAY)))
   (COND
     (display!statistics!flag (DISPLAY!STATISTICS)))
   (DISPLAY)

Calls: DISPLAY!EVENT DISPLAY!NODE DISPLAY!PARAMETERS DISPLAY!STATISTICS INITIALIZE INSTALL!INTERNAL!EVENT NEXT!EVENT PROCESS!DISPLAY!EVENT PROCESS!INTERNAL!EVENT PROCESS!MESSAGE RANDOMCOMPARE STORE!OBJECT

Called by: CNET

Freevars: NET aacounter acccounter bdcstcounter bidcounter dacounter display!banners!flag display!events!flag display!node!flag display!parameter!flag display!statistics!flag display!time!flag eventcounter eventlist final!function frcounter initial!function ircounter messagecounter nacounter netsize nodelist recounter rtime tacounter tecounter time utilization

Explanation: Performs the main contract net simulation. Initializes the net and calls the initial user function. Sets up contracts as indicated by that function. Then processes events from the event list until no more events remain to be processed. Then displays statistics if required.
Explanation: Stores an object of type "xobject" in the knowledge base of node "xpnode". "xinstance" is the object. A knowledge base is a record with slots that correspond to the objects recognized by all nodes. Such objects are listed in each slot. A knowledge base also has an 'other' slot used to hold a list of lists of dynamically defined objects. Each such list has a header that corresponds to the type of object.
(STORE!TASK!OBJECT
  (LAMBDA (xpnode xtype xspecification)
    (PROG (xnode xtask xtemplate xtask)
      (SETQ xnode (ELT NET xpnode))
      (SETQ xtask (create TASK NAME ← (fetch (PNODE TASKCOUNTER) of xpnode))
        TYPE ← xtype ANNOUNCEMENT!PROCEDURE ← (fetch (TASK!TEMPLATE ANNOUNCEMENT!PROCEDURE)
          of xtask!template)
        ANNOUNCEMENT!RANKING!PROCEDURE ← (fetch (TASK!TEMPLATE ANNOUNCEMENT!RANKING!PROCEDURE)
          of xtask!template)
        BID!CONSTRUCTION!PROCEDURE ← (fetch (TASK!TEMPLATE BID!CONSTRUCTION!PROCEDURE)
          of xtask!template)
        BID!RANKING!PROCEDURE ← (fetch (TASK!TEMPLATE BID!RANKING!PROCEDURE)
          of xtask!template)
        AWARD!PROCEDURE ← (fetch (TASK!TEMPLATE AWARD!PROCEDURE)
          of xtask!template)
        REFUSAL!PROCEDURE ← (fetch (TASK!TEMPLATE REFUSAL!PROCEDURE)
          of xtask!template)
        REFUSAL!PROCESSING!PROCEDURE ← (fetch (TASK!TEMPLATE REFUSAL!PROCESSING!PROCEDURE)
          of xtask!template)
        REPORT!ACCEPTANCE!PROCEDURE ← (fetch (TASK!TEMPLATE REPORT!ACCEPTANCE!PROCEDURE)
          of xtask!template)
        TERMINATION!PROCEDURE ← (fetch (TASK!TEMPLATE TERMINATION!PROCEDURE)
          of xtask!template)
        INFORMATION!ACCEPTANCE!PROCEDURE ← (fetch (TASK!TEMPLATE INFORMATION!ACCEPTANCE!PROCEDURE)
          of xtask!template)
        EXECUTION!PROCEDURE ← (fetch (TASK!TEMPLATE EXECUTION!PROCEDURE)
          of xtask!template)
        SPECIFICATION ← xspecification))
      (STORE!OBJECT xpnode 'TASK xtask)
      (replace (TASK!TEMPLATE TASKS) of xtask!template with (CONS (fetch (TASK NAME) of xtask)
        (fetch (TASK!TEMPLATE TASKS) of xtask!template)))
      (RETURN (fetch (TASK NAME) of xtask))
    )
  )

Calls: RETRIEVE!OBJECT STORE!OBJECT
Called by: GENERATE!SUBTASK PROCESS!ANNOUNCED!AWARD PROCESS!DIRECTED!AWARD SIMULATE
Freevars: NET

Explanation: Stores a task object of type "xtype" in the knowledge base of node "xpnode". "xspecification" is the 
'task!specification'. The 'task!template' for the task type is used to copy pointers to the required 
procedures for the task. The name of the task is simply the number of tasks that have been generated by 
"xpnode". Returns the task name.

(SUSPEND
  (LAMBDA NIL
    (AU-REVOIR 'SUSPEND))

Called by: EXTEND!BOARD QRECEIVE

Explanation: Does an AU-REVOIR and returns the keyword 'SUSPEND as the possibilities list value. Called directly by 
user task execution procedures to suspend processing of tasks.
**TERMINATE**

(call (lambda (NIL) (adieu 'terminaTe)))

called by: $TEST EXTEND!BOARD QRECEIVE

Explanation: Does an ADIEU and returns the keyword 'TERMINATE as the possibilities list value. Called directly by user task execution procedures to terminate processing of tasks.

---

**TERMINATE|SUBCONTRACTS**

(call (lambda (xpnode xcontract xtime) (prog (tmpsc) (cond ((null xtime) (setq xtime 0)) (for x in (cdr (fetch (contract subcontracts) of xcontract)) do (cond ((not (equal 0 (fetch (subcontract contractor) of x))) (sendmessage xpnode (plus xtime time tt) (fetch (subcontract contractor) of x)) (create termination name ←(fetch (subcontract name) of x) 'announced t) (install!display!event time xpnode 'simulation (append 'terminated contract) (fetch (contract name) of x) (for y in (fetch (contract successors) of x) do (setq tmpsc (find!subcontract xpnode y)) (replace (contract successors) of tmpsc with (remove (fetch (subcontract name) of x) (fetch (contract successors) of tmpsc) of tmpsc) (for y in (fetch (contract predecessors) of x) do (setq tmpsc (find!subcontract xpnode y)) (replace (contract predecessors) of tmpsc with (remove (fetch (subcontract name) of x) (fetch (contract predecessors) of tmpsc) of tmpsc) (replace (contract subcontracts) of xcontract with nil)

(called by: process!termination update!node)

Freevars: time tt

Explanation: Terminates all outstanding subcontracts of the contract with name "xname" (held by node "xpnode") at time "xtime". Updates predecessors and successor (some question on this for future).
(UPDATE!ACTIVE!TASK!ANNOUNCEMENTS
 (LAMBDA (xpnode)
   (PROG (xpnode! active)
     (SETQ xpnode! (ELT NET xpnode))
     (SETQ active (fetch (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode!))
     (COND
      (active (replace (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode!)
               with (for x in active collect x when (IGREATERTP (fetch (ACTIVE!TASK!ANNOUNCEMENT EXPIRATION!TIME) of x)
                         time))
     ))
   ))
---------------------

Called by: NEXT!CONTRACT!PROCESS!TASK!ANNOUNCEMENT

Freevars: NET time

Explanation: Deletes all active task announcements in node "xpnode" whose expiration time has passed.
Freevars: NET ntermcs task!time time

Explanation: Continues processing of contract named "xname" in node "xpnode". "xdata" is the pointer to the task execution procedure (as a generator).

If "xdata" is 'TERMINATE, then the contract is terminated (this occurs when the task execution procedure calls the function TERMINATE).

All outstanding subcontracts are terminated.

If the terminated state contains more than 'ntermcs' contracts then the oldest contract is discarded, after presenting it to the 'termination!procedure' for its task (if such a procedure exists).

Processing starts on the next contract in the 'ready' state.

If "xdata" is 'SUSPEND, then the contract is suspended (this occurs when the task execution procedure calls the function SUSPEND).

Processing starts on the next contract in the 'ready' state.

Otherwise, TRYNEXT is executed and another 'node!update' event is placed on the event list.

Freevars: NET

Explanation: Replaces the value of the "xslot" slot of the object of type "xobject" in node "xpnode". "xkey" is the key that is matched to find the object, and "xvalue" is the new value for the named slot. "xkey" must be the value of the first slot for the object.
(UPDATE!NODE (UPDATE!NODE [LAMBDA (xpnode xname xdata) (PROG (xnode! xcontract temp temp1 xtn xta xtt xtermproc) (SETQ temp (CAR xdata)) (SETQ xnode! (ELT NET xpnode)) (COND (SAME!STATUS!CHECK xpnode xname) (SETQ xcontract (CAR (fetch (PNODE EXECUTING) of xpnode!)))) (if the value returned on the possibilities list is SUSPEND or TERMINATE, then take the associated action. Otherwise reschedule the contract through TRYNEXT and place a new node!update event on the event list) (SELECTQ (COR xdata) (TERMINATE (replace (CONTRACT STATE) of xcontract with 'TERMINATED)) (replace (PNODE TERMINATED) of xpnode! with (CONS xcontract (fetch (PNODE TERMINATED) of xpnode!)))) (COND ((ILESSP ntermcs (LENGTH (fetch (PNODE TERMINATED) of xpnode!)) (REVERSE (fetch (PNODE TERMINATED) of xpnode!)) (SETQ xtn (fetch (CONTRACT TASK) of (CAR (fetch (PNODE TERMINATED) of xpnode!)))) (SETQ xta (RETRIEVE!OBJECT xpnode 'TASK xtn)) (SETQ xtermproc (fetch (TASK TERMINATION!PROCEDURE) of xta)) (COND (xtermproc (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE xtermproc)) (LIST xpnode (CAR (fetch (PNODE TERMINATED) of xpnode!)))) (T (SETQ xtn (RETRIEVE!OBJECT xpnode 'TASK!TEMPLATE (fetch (TASK TYPE) of xta))) (SELECTQ (COR xdata) (TERMINATE (replace (CONTRACT STATE) of xcontract with 'TERMINATED)) (replace (PNODE TERMINATED) of xpnode! with (CONS xcontract (fetch (PNODE TERMINATED) of xpnode!)) (INSTALL!DISPLAY!EVENT (IPLUS time task!time) xpnode 'SIMULATION (APPEND 'Terminated Contract xname)) (TERMINATE!SUBCONTRACTS xpnode xcontract) (NEXT!CONTRACT xpnode 'TERMINATION)))) (SUSPEND (replace (CONTRACT STATE) of xcontract with 'SUSPENDED) (if the value returned on the possibilities list must retain the pointer to the possibilities list) (replace (PNODE SUSPENDED) of xpnode! with (CONS xcontract (fetch (PNODE SUSPENDED) of xpnode!))) (INSTALL!DISPLAY!EVENT (IPLUS time task!time) xpnode 'SIMULATION (APPEND 'Suspended Contract xname)) (TERMINATE!SUBCONTRACTS xpnode xcontract) (NEXT!CONTRACT xpnode 'TERMINATION))) (PROCN (SETQ templ (RESUME!TASK temp1)) (INSTALL!INTERNAL!EVENT (IPLUS time task!time) xpnode 'NODE!UPDATE (CONS temp templ)))) (T (RELEASE!TASK temp1)))) Calls: DELETE!OBJECT INSTALL!DISPLAY!EVENT INSTALL!INTERNAL!EVENT NEXT!CONTRACT RELEASE!TASK RESUME!TASK RETRIEVE!OBJECT SAME!STATUS!CHECK TERMINATE!SUBCONTRACTS Called by: PROCESS!INTERNAL!EVENT

-62-
UPDATE!TASK!TIME

(UPDATE!TASK!TIME
 (SETQ task!time (1PLUS task!time (ITIMES gain (COND
 ((IGREATERP t -1)
 t)
 (T 3))))

----------

Called by: $TEST EXTEND!BOARD

Freevars: gain task!time

Explanation: Updates the 'task!time' for the calling task by 'gain' times "t" units.

---

VALUEP

(VALUEP
 (LAMBDA (xvalue)
 (COND
 ((EQ (CAR xvalue) 'QUOTE)
 (CADR xvalue))
 (T NIL)))

----------
1. December 9, 1978 5:56PM in <VANMELLE>WW.SAV;8112
2. INTERPRET
3. BINDSYMBOLO
4. INTERPP
5. INTERPQ
6. INTERPT
7. INTERPV
8. QUERY
Fns on INTERPRET:

INTERPRETBLOCK        INTERPP        INTERPU
INTERPRET              INTERPQ        INTERPV
BINDSYMBOL             INTERPT        QUERY

Block INTERPRETBLOCK
Entries:  INTERPRET
Internal:  BINDSYMBOL, INTERPP, INTERPQ, INTERPT, INTERPU, INTERPV, QUERY
Specvars:  PHRASE, CLASSES, OR, ELLIPSISFLG, BINDINGS, FUNCTIONS, POSSIBLEGRAMMARS, ORIGINALPHRASE, TOPFLG, TEMPLATE, RETEMPLATE
(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET

(INTERPRET
optional sequences cannot be matched, but the remainder of the TEMPLATE can be matched, then the parse will still succeed. If they can be matched, then the matching is done in the usual way. (This is also the way in which recursive subgrammars are handled. For example: noun-phrase ← adjective (noun-phrase)

(CLASS-1 CLASS-2 (CLASS-3) ANCHOR!WORD-1 ... CLASS-N)

Each CLASS has one of the following properties:

PRONOUNS: If a CLASS has the PRONOUN property then, when it is matched (see POSSIBLEVALUES below), the previous "bindings" of the CLASS are returned; else the TEMPLATE match fails. If the value of PRONOUNS is atomic, then it is treated as a function, and evaluated in the context of the current bindings to get the list of possible values.

POSSIBLEVALUES: a list of values that a member of the CLASS can assume (e.g., this could be part of a dictionary). The value of POSSIBLEVALUES has the following form: (PV-1 PV-2 ... PV-N). If the value of POSSIBLEVALUES is atomic, then it is treated as a function, and evaluated in the context of the current bindings to get a list of this form. Each of the PV-i is used as a possible value for the CLASS. Each PV-i can be one of the following:

1) ATOM. If the ATOM is matched then it is returned as the binding for the match.
2) (ATOM-1 ATOM-2 ... ). If the ATOM list is matched then it is returned as the binding for the match.
3) ((ATOM-1 ATOM-2 ...) (ATOM-3 ATOM-4 ... ) (... ) VALUE). If any of the ATOM lists is matched then VALUE is returned as the binding for the match. VALUE is not evaluated.

PREDICATE: a predicate to be applied to test for class membership. If the predicate is atomic, then it gets one word to test, and is expected to return T or NIL. If it is enclosed in parentheses, then it gets the rest of the phrase, and is expected to return a PREDICATE record. Such a record has the following fields:

REST: the portion of the phrase that remains to be matched after the predicate has been applied.
PVALUE: the result of applying the predicate—either T or NIL.
KLEENE: the CLASS will match anything up to the next TEMPLATE element.
GRAMMARS: the class can itself be a subgrammar in the same format as the top-level grammar. The KLEENE property cannot be used in a subgrammar—stack overflow is the result.

---

(BINDSYMBOL
  (LAMBDA (CLASS VALUE)
    (PROG ((ENTRY (FASSOC CLASS BINDINGS)))
      (COND
        (ENTRY (SETQ BINDINGS (for BINDING in BINDINGS unless (EQ BINDING ENTRY) collect BINDING)
          (SEQ BINDINGS (CONS (CONS CLASS VALUE)
            BINDINGS))
          (RETURN BINDINGS)))
    )
  )
)

---

Called by: INTERPP INTERPT

Freevars: BINDINGS

Explanation: Actually binds a symbol (a word or sequence of words or... -- see INTERPRET) to a CLASS. "CLASS" is the class. "VALUE" is the symbol to be bound to the CLASS. BINDINGS is altered by this function.
(INTERPP
   (LAMBDA NIL
    (for old REMTEMPLATE on TEMPLATE as GSYMBOL is (CAR REMTEMPLATE) bind QR
     do (COND
         (LITERAL GSYMBOL)
         (COND
          [(MEMBER GSYMBOL CLASSES)
           (COND
            [(SETQ QR (GETP GSYMBOL 'KLEENE))
             (COND
              [(EVAL QR)
               [COND
                [(NOT (SETQ REMTEMPLATE (CDR REMTEMPLATE)))
                 [BINDSYMBOL GSYMBOL PHRASE]
                 [SETQ PHRASE NIL]]
                [(for XPHRASE on PHRASE when (fetch (QRESULTS QMATCH) of (SETQ QR (INTERPT XPHRASE REMTEMPLATE '('(T NIL) BINDINGS))))
                 do (SETQ BINDINGS (fetch (QRESULTS BINDINGS) of QR))
                 [BINDSYMBOL GSYMBOL (LDIFF PHRASE XPHRASE)]
                 [RETURN T])
                 [SETQ REMTEMPLATE NIL]
                 [SETQ PHRASE (fetch (QRESULTS RP) of QR)]
                 [RETURN]]
             [RETURN])
            [(T (BINDSYMGBOL GSYMGBOL)
             [(INTERPV (GETP GSYMGBOL 'PRONOUNS))
             [COND
             [(NOT (FASSOC GSYMGBOL BINDINGS))
             [RETURN]]
             [(INTERPV (GETP GSYMGBOL 'POSSIBLEVALUES))
             [BINDSYMGBOL GSYMGBOL QR]]
             [(INTERPV (GETP GSYMGBOL 'PREDICATE))
             [BINDSYMGBOL GSYMGBOL QR]]
             [(fetch (QRESULTS QMATCH) of (SETQ QR (QUERY PHRASE (GETP GSYMGBOL 'GRAMMARS) NIL ELLIPSISFLG)))
             [(* a sub-grammar matched)]]
             [(SETQ PHRASE (fetch (QRESULTS RP) of QR))
             [BINDSYMGBOL GSYMGBOL (fetch (QRESULTS VALUE) of QR)]
             [(NOT (AND ELLIPSISFLG (FASSOC GSYMGBOL BINDINGS)))]
             [(RETURN)]
             [(NEQ GSYMGBOL (CAR PHRASE))
             [RETURN]]
             [(T (SETQ PHRASE (CDR PHRASE)
             [(fetch (QRESULTS QMATCH) of (SETQ QR (INTERPT PHRASE GSYMGBOL '('(T NIL) BINDINGS))))
             [SETQ PHRASE (fetch (QRESULTS RP) of QR)]
             [SETQ BINDINGS (fetch (QRESULTS BINDINGS) of QR)]
             [(AND (LITERAL GSYMGBOL)
             [MEMBER GSYMGBOL CLASSES])
             [COND
             [(NOT (AND ELLIPSISFLG (FASSOC GSYMGBOL BINDINGS)))]
             [RETURN)]
             [RETURN])]})
    [RETURN])]
   [RETURN]))

Calls: BINDSYMGBOL INTERPV INTERPT INTERPV QUERY
Called by: INTERPT
Freevars: BINDINGS CLASSES ELLIPSISFLG PHRASE REMTEMPLATE TEMPLATE

Explanation: Tries to match a template against a phrase. "PHRASE" is the phrase to be matched. "TEMPLATE" is the template to be used. "BINDINGS" is the list of current bindings. Returns a QRESULTS record with the following field settings:
(INTERPQ
  ILAMBDA (PREDICATE)
  (COND
    (PREDICATE
      (COND
        ((LITATOM PREDICATE)
          (COND
            ((SETQ QR (APPLY* PREDICATE (CAR PHRASE)))
              (SETQ PHRASE (CDR PHRASE)) QR))
            ((SETQ QR (APPLY* (CAR PREDICATE) PHRASE))
              (SETQ PHRASE (fetch (PREDICATE REST) of QR))
              (SETQ QR (fetch (PREDICATE PVALUE) of QR)))
      )
    )
  )
)

Called by: INTERPP

Freevars: PHRASE QR

Explanation: Applies a predicate to the remaining phrase (see INTERPRET for more detail). "PREDICATE" is the predicate to be applied.

(INTERPT
  ILAMBDA (PHRASE TEMPLATE FUNCTIONS BINDINGS TOPFLG)
  (PROG (REMTEMPLATE)
    (for ENTRY in TEMPLATE when (NOT (LITATOM ENTRY)) do (for GSYMBOL in ENTRY when (AND (FMEMB GSYMBOL CLASSES)
      (NOT (FASSOC GSYMBOL BINDINGS)))
      do (BINDSYMBOL GSYMBOL)))
    (INTERPP)
    (COND
      (AND (NULL REMTEMPLATE)
        (OR (NOT TOPFLG)
          (NEQ ORIGINALPHRASE PHRASE))
        (EVALA (fetch (FUNCTIONS SEMANTIC!PREDICATE) of FUNCTIONS) BINDINGS)) (* it worked!)
      (RETURN (create QRESULTS QMATCH ← T RP ← PHRASE BINDINGS ← BINDINGS VALUE ← (EVALA (fetch (FUNCTIONS ACTION!FUNCTION) of FUNCTIONS) BINDINGS)
        (AND TOPFLG (NOT NOFAILRECORDS) (NOT ELLIPSISFLG) (NEQ REMTEMPLATE TEMPLATE))
        (SETQ POSSIBLEGRAMMARS (CONS (create FAILURE TEMPLATE ← TEMPLATE FUNCTIONS ← FUNCTIONS REMPHRASE ← PHRASE REMTEMPLATE ← REMTEMPLATE FBINDINGS ← BINDINGS) POSSIBLEGRAMMARS))
      )
    )
  )
)

Calls: BINDSYMBOL INTERPP

Called by: INTERPP QUERY

Freevars: CLASSES ELLIPSISFLG NOFAILRECORDS ORIGINALPHRASE POSSIBLEGRAMMARS

Explanation: Tries to match an element of a grammar against a phrase. If the match is successful, then the associated SEMANTIC!PREDICATE is evaluated. If that is successful, then the associated ACTION!FUNCTION is evaluated. "PHRASE" is the phrase to be matched. "TEMPLATE" is the element of the grammar (see INTERPRET for more detail). "BINDINGS" is the current list of bindings.
(INTERPU)
  (LAMBDA (PATTERN)
    (PROG [(REMPHRASE (FNTH PHRASE (FLENGTH PATTERN)
        (COND
          ((AND REMPHRASE (for PWORD in PATTERN as WORD in PHRASE always (EQ PWORD WORD)))
            (SETQ PHRASE (CDR REMPHRASE))
            (RETURN T))

    Called by: INTERPV
    Freevars: PHRASE
    Explanation: Returns T if the N words in "PATTERN" match the first N words in "PHRASE". Also removes the words from "PHRASE" when a match is found.

(bvm: 19-Feb-78 16:39)

(INTERPV)
  (LAMBDA (VALUES)
    (COND
      ((AND VALUES (ATOM VALUES))
        (SETQ VALUES (EVALA (LIST VALUES) BINDINGS))
        (for ENTRY in VALUES do (SETQ QR ENTRY)
          (COND
            ((ATOM ENTRY)
              (COND
                ((EQ ENTRY (CAR PHRASE))
                  (SETQ PHRASE (CDR PHRASE))
                  (RETURN T))
            ((LISTP (CAR ENTRY))
              (SETQ QR (CAR (FLAST ENTRY)))
              (COND
                ((for PATTERN in ENTRY unless (EQ QR PATTERN) thereis (INTERPU PATTERN))
                  (RETURN T))
                ((INTERPU ENTRY)
                  (RETURN T))

    Calls: INTERPU
    Called by: INTERPP
    Freevars: BINDINGS PHRASE QR
    Explanation: Tries to match a pronoun or possible value against the phrase. "VALUES" is the pronoun or possible value to be matched (see INTERPRET for more detail).
QUERY

(LAMBDA (PHRASE GRAMMAR BINDINGS ELLIPSISFLG TOPFLG) (* rgs: "25-Oct-78 01:11")

(PROG (QR)
  (COND
    (PHRASE (FOR ENTRY IN GRAMMAR UNLESS (FETCH (GRESULTS QMATCH) OF QR) BIND FUNCTIONS
      (SETQ FUNCTIONS (CAR (FLAST ENTRY)))
      (FOR TEMPLATE IN ENTRY UNLESS (EQ FUNCTIONS TEMPLATE)
        THEREIS (SETQ QR (INTERPT PHRASE TEMPLATE FUNCTIONS BINDINGS TOPFLG))))

  (RETURN QR))

------
Calls: INTERPT
Called by: INTERPP INTERPRET

Explanation: Actually carries out a parse attempt for the phrase. "PHRASE" is the phrase to be parsed, "GRAMMAR" is the grammar to be used for the parse, "BINDINGS" is the list of current bindings, "ELLIPSISFLG" is T if elliptical reference is to be resolved, "TOPFLG" is T if QUERY is working on the top-level grammar. Returns T if a parse has been successfully completed; else returns a QRESULTS record with the RI field set to PHRASE.
December 9, 1978 5:58PM in <VANNELLE>WW.SAV;8112

1. ASKFORFILENAME
2. ASKFORFILENAME
3. ASKFORFUNCTIONNAME
4. ASKYESNO
5. CTRLQ.NLSETQ
6. DISPLAY
7. DISPLAYHELP
8. GETFILE
9. INFILEDIR
   OPENHASHFILEVARS
10. OPENMYCINHASHFILE
11. PRINTPROP&VAL
12. PRINTNUMBER
13. SPRINT
14. SPRINTI
15. SPRINTATOM
16. SPRINTCOUNT
17. SPRINTPUNC
18. SPRINTSEPR
19. SPRINTSTRING
20. TTYOUT
21. UGETHASHFILE
   WRITE
22. WRITE1
   WRITE2
   WRITEARG
<table>
<thead>
<tr>
<th>Function</th>
<th>Display</th>
<th>PrintPropVal</th>
<th>PrintRecord</th>
<th>PrintCount</th>
<th>PrintPunc</th>
<th>PrintSEPR</th>
<th>PrintSTRING</th>
<th>TTYOut</th>
<th>Write</th>
<th>Write1</th>
<th>Write3</th>
<th>WriteARG</th>
</tr>
</thead>
<tbody>
<tr>
<td>AskForFilename</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AskForFunctionName</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AskForNumber</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AskYesNo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CtrlO.NLsetq</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ASKFORFILENAME

(LAMBDA (xmode xdefault)
  (PROG (tempjfn tempfile)
    (COND
      (xdefault (SETQ xdefault (SELECTQ xmode
        (INPUT (INFILEP xdefault))
        (OUTPUT (OUTFILEP xdefault))
        NIL)
      LOOP (WRITE1 "File Name for " xmode " ")
        (COND
          (xdefault (WRITE1 "[" xdefault "] ** "))
          (T (WRITE1 "** ")))
      (SETQ tempjfn (RESETLST (RESETSAVE (INTERRUPTCHAR 4))
        (RESETSAVE (INTERRUPTCHAR 5))
        (JSYS 16 (SELECTQ xmode
          (INPUT 1583171968)
          -10736631808))
        16777281)))))
    (SETQ tempfile (JFNS tempjfn))
    (COND
      (tempfile (SETQ xdefault tempfile))
      (COND
        ((NOT (AND (OR tempfile (EQP tempjfn 196685))
          xdefault))
          (OR (ZEROP (POSITION))
          (WRITE))
          (WRITE1 (OR (ERSTR tempjfn)
            "bad response. try again.")
          (WRITE)
          (GO LOOP)))
        (CON(D)
          (NOT tempfile)
          (WRITE))
        (RETURN xdefault))
    ))

----------
Called by: SET!PARAMETERS

Explanation: Asks for a filename. xmode is the mode to be used (READ or WRITE). xdefault is the default, which is returned if <cr> is the response (in the case of WRITE mode a new version is created). Full TENEX recognition is in effect.

ASKFORFUNCTIONNAME

(LAMBDA (xprompt xdefault xhelp)
  (PROG (temp)
    (SETQ temp T)
    (do (SETQ temp (CAR (TTYIN (LIST xprompt " [" xdefault "] ** ")
      NIL xhelp)))
        until (OR (NULL temp)
          (GETQ temp)))
    (COND
      (temp (RETURN temp))
      (T (RETURN xdefault))))

----------
Called by: SET!PARAMETERS

Explanation: Asks for the name of a function, using TTYIN. xprompt is the prompt that is displayed. xdefault is the default, which is returned if <cr> is the response. xhelp is the key to a hashfile entry. The response is only accepted if it is the name of a function.
(ASKFORNUMBER
   (LAMBDA (xprompt xdefault xhelp xlb xub)
     (PROG (temp)
       (SETQ temp T)
       (do (SETQ temp (CAR (TTYIN (LIST xprompt " [" xdefault "] \* \* ") NIL xhelp)))
         until (OR (NULL temp)
                   (AND (NUMBERP temp)
                        (IGREATERP temp xlb)
                        (COND
                          (xub (ILESSP temp xub))
                          (T T))
                        (COND
                          (temp (RETURN temp))
                          (T (RETURN xdefault))))
   Called by: QSET!PARAMETERS SET!PARAMETERS

Explanation: Asks for a number, using TTYIN. xprompt is the prompt that is displayed. xdefault is the default, which is returned if <cr> is the response. xhelp is the key to a hashfile entry. The response must be greater than xlb and less than xub.

rgs: 13-Oct-78 01:06 [UTILITY]  ASKFORYESNO

(ASKFORYESNO
   (LAMBDA (xprompt xdefault xhelp)
     (PROG (temp)
       (SETQ temp (CAR (TTYIN (LIST xprompt " [" (COND
                                 (xdefault 'YES)
                                 (T 'NO))
                                 " ] \* \* ")
                                 (LIST '(Yes . Y)
                                         'No . N))
                                 xhelp
                                 (LIST 'FIX))
       (RETURN (COND
                 ((NULL temp)
                  xdefault)
                 ((OR (EQ temp 'Y)
                     (EQ temp 'YES))
                  T)
                 (T NIL))
   Called by: RESIMULATE SET!PARAMETERS

Explanation: Returns T for an affirmative response, using TTYIN. xprompt is the prompt that is displayed. xdefault is the default, which is returned if <cr> is the response. xhelp is the key to a hashfile entry. Essentially like ASKYESNO with a default.

rgs: 13-Oct-78 01:06 [UTILITY]
(ASKYESNO
 (LAMBDA (QUESTION PROMPTYPE DEFAULT HELP)
  (SELECTQ PROMPTYPE
   (NIL CONFIRM)
   (RESETFORM (SETTERM TABLE ASKUSER TABLE))
   (PROG (ANSWER BUFS (TYPEAHEAD (READP T)))
    (COND
     (QUESTION (COND
      ((LITATOM QUESTION)
       (PRIN1 QUESTION T)
       (SETQ QUESTION " ? ")
      ((EQ PROMPTYPE 'CONFIRM)
       (SETQ QUESTION " [confirm] ")
      )
     ))
    TOP
     (COND
      (LISTP QUESTION)
      (MAPRINT QUESTION T)
      (OR (EQ PROMPTYPE 'CONFIRM)
       (PRIN1 " ? " T))
      (QUESTION (PRINQ QUESTION T)
       (COND
        ((EQ NTHCHAR QUESTION -1)
         " %")
        (SPACES 1 T)
       )
     )
    READ:
     (SELECTQ (JSYS 59)
      ((BS 121)
       (PRINI "Yes" T)
       (SETQ ANSWER T)
       (GO DONE:))
      ((BS 110)
       (PRINI "No" T)
       (GO DONE:))
      ((31 44)
       (COND
        ((EQ PROMPTYPE 'CONFIRM)
         (SETQ ANSWER T)
         (GO DONE:))
        (63)
        (TERPRI T)
        (SETQ TYPEAHEAD)
        (COND
         (HELP (COND
          ((LITATOM HELP)
           (DISPLAYHELP HELP))
          (T (SPRINTT HELP)))
          (GO TOP))
         (T (PRINQ (COND
          ((EQ PROMPTYPE 'CONFIRM)
           ([type carriage return to confirm] "
           (T "Type Yes or No: ")
           T)
          )
         ))
         (GO READ:))
      )
      ((127 24)
       (COND
        ((EQ PROMPTYPE 'CONFIRM)
         (PRINI "xxx" T)
         (DISMISS 500)
         (CLEAR BUF T)
         (GO DONE:))
        (NIL)
        (JSYS 60 T)
        (COND
         (TYPEAHEAD)
         (JSYS 34 64)
        )
       )
     )))
    (e Do a PBIN to get next character)
    (e Y)
    (e N)
    (e crtl)
    (e ?)
    (e Ring terminal bell for inappropriate response)
    (e User may have typed ahead before QUESTION was printed. Save buffers and gotry again)
    (e BKJFN puts back character the PBIN)
(DOBE)
(DISSISS 1000)
(SETQ BUFS (CLBUIFS))
(SETQ TYPEAHEAD)))

(GO READ:)

DONE:
(COND
 (BUFS (BKBUFS BUFS)))
 (TERPRI T)
 (RETURN ANSWER)

(PROGN (COND
 (QUESTION (SPRINT QUESTION)))
 (OR (EQ PROMPTYPE 'NOTERPRI)
 (TERPRI)))

(COND
 (BATCHFLG (WRITE "#..." (COND
 (DEFAULT "yes")
 (T "no"))))

 DEFAULT)
 (T (do (SELECTQ CAR TTYIN (SELECTQ PROMPTYPE
 (T NOTERPRI)
 (NIL)
 PROMPTYPE)
 '(YES NO)
 HELP))
 (YES (RETURN T))
 (NO (RETURN))
 (WRITE "Yes or No, please."))

-------------

Calls: DISPLAYHELP

Called by: OPENMYCINHASHFILE

Globalvars: BATCHFLG

Freevars: ASKUSERTTBL

Explanation: Returns T if the response to QUESTION is affirmative. There are two basic modes: immediate (for hacking-type questions) and standard (using TTYIN); which one depends on the value of PROMPTYPE:
NIL -- (immediate) function behaves roughly like ASKUSER with a yes/no keylist and typeahead permitted (rings bell for incorrect response (not Y or N), clears and saves typeahead if typeahead looks bogus). If QUESTION is a literal atom, it is printed, followed by a "?", if a list, it is MAPRINTed.
CONFIRM -- like above, except <crlf> is accepted (even expected) as the affirmative response, and <del> or TX disconfirm. If QUESTION is NIL, supplies "[confirm]".
T -- (standard) SPRINTs QUESTION and then calls TTYIN for the standard #d prompt.
NOTERPRI -- like T, but does not print crlf before #d.
<other> -- any other prompt is passed to TTYIN.
Additionally: if HELP is specified, it is given if user types a "?" (same as TTYIN's HELP arg). If BATCHFLG is set (i.e. user input is not being taken), DEFAULT (T or NIL) is the response supplied for the non-immediate types.
CTRLO.NLSETQ

(DECLARE (LOCALVARS . T)
  (SPECVARS CTRLO!))
(RESETLST (PROG (MACROX CTRLO! CTRL0!))
  (COND
    ((NOT CTRLO!)
      (RESETSAVE (INTERRUPTCHAR 15 'CTRLO!) T))
    (SETQ CTRL0! T))
  LP (SETQ MACROX (ERRORSET NLSETX))
  (COND
    ((AND NLSETY (NOT MACROX))
      (GO LP)))
RETURN MACROX)

Called by: DISPLAYHELP

Globalvars: CTRLO!

Explanation: Evaluates NLSETX under errorset protection, like NLSETQ. In addition, the TO interrupt is armed inside here, so that the user may abort with it. If the second argument (NLSETY) is true and a TO happens during the evaluation of NLSETX, it is reevaluated, i.e. a TO causes the function to loop, and the CTRLO.NLSETQ will only exit without error. The variable CTRLO! is bound to T inside here as a cheap flag to indicate that TO is on.

DISPLAY

(LAMBDA N
  (COND
    (fileflag (for I from 1 to N do (WRITEARG (ARG N I))))
    (TERPRI))
  (COND
    (termflag (for I from 1 to N do (WRITEARG (ARG N I) T))
    (TERPRI T))
RETURN)

Called by: WRITEARG

Called by: DISPLAY!CONTRACT DISPLAY!EVENT DISPLAY!MESSAGE DISPLAY!NODE DISPLAY!PARAMETERS DISPLAY!STATISTICS PROCESS!DISPLAY!EVENT QFINALIZE SIMULATE

Freevars: fileflag termflag

Explanation: Like WRITE, but writes to the primary output file if fileflag is set, and writes to the terminal if termflag is set. If both are set, then writes to both places.
(DISPLAYHELP)

(LAMBDA (KEY QUIET)
  (PROG (RESULT)
    (RETURN (OR NOT (SETQ RESULT (CTRLO.NLSETQ (UGETHASHFILE "HELPFILE KEY NIL NIL
                      (OR QUIET "Helpfile unavailable.")
                      (CAR RESULT))

        ------------------
        Calls:         CTRL0.NLSETQ UGETHASHFILE

        Called by:     ASKYESNO

        Explanation:  Copies to primary output the help blurb indexed by KEY. If QUIET is set, will not complain if the
                      hashfile is unavailable (not found or won't open). Returns NIL if no entry found for KEY (and hence
                      nothing was printed); T if the entry was found, or user typed ^O.)
GETFILE

(GETFILE
    (LAMBDA (FILE ASK SHOW)
    (PROG (FOUND ENTRY)
        (COND
            ((SETQ FOUND (OR (AND (SETQ ENTRY (FASSOC FILE PREFERREDFILES))
                                  (INFILEP (CDR ENTRY)))
              (INFILEP FILE)))
            (RETURN FOUND))
            ((SETQ ENTRY (FASSOC FILE (OR (LISTP (EVALV 'GETFILELST))
                                           (SETATOMVAL 'GETFILELST))
              (COND
                    ((NULL (CDR ENTRY))
                     (RETURN))
                    ((SETQ FOUND (INFILEP (OR (CDR ENTRY)
                                              (CDR FRPLACD (CDR ENTRY)
                                              (MKATOM (SUBSTRING (CADR ENTRY)
                                              1
                                              (STRPOS ' ; (CADR ENTRY
                                              (RETURN FOUND))
                        (T (DREMOVE ENTRY GETFILELST)
                        (COND
                            ((SETQ FOUND (for DIR in OTHERDIRS any (INFILEDIR DIR FILE)))
                             (COND
                              (SHOW (TTYOUT "...from " FOUND))
                              (GO FOUND:))
                              (NOT ASK)
                              (RETURN))
                          )))
          (RETURN FOUND))
          (T DELETE ENTRY GETFILELST)
          (COND
            ((SETQ FOUND (or (for DIR in OTHERDIRS any (INFILEDIR DIR FILE)))
               (COND
                 (SHOW (TTYOUT "...from " FOUND))
                 (GO FOUND:))
                 (NOT ASK)
                 (RETURN))
               )
           )))
    ))
    )))

# Explanation:
Locates FILE, looking first on the connected directory, then on OTHERDIRS, then if ASK is set asks the
user for help. Returns the complete file name of the first file (if any) found which is INFILEP. If SHOW
is set, prints file found if other than obvious.

PREFERREDFILES is an association list of (file . filename) indicating an override of this default
scheme; GETFILE will first check the indicated filename before trying anywhere else.

To speed up repeated calls on the same file, GETFILE keeps track of the last several files it looked up;
It will check this list (GETFILELST) before blindly searching other directories.
INFILEDIR
(INFLD (DIR NAME EXT)
(PROG (JFN)
  (RETURN (COND
    ((SETQ JFN (LGTJFN DIR NAME EXT))
      (PROG1 [MKATOM (JFNS JFN NIL (CONSTANT (CONCAT (RLJFN JFN)))))
    (T (RETURN NIL))))
  (T (RETURN NIL))))

Called by: GETFILE
Explanation: Returns full name of file on directory DIR, where file is NAME.EXT (or NAME if EXT is nil, or NAME itself contains a "."). If DIR is NIL, connected directory is used. DIR may or may not begin with a "<".

OPENHASHFILEVARS
(INFLD (VARS WRITE? SAVE NOERROR)
  (PROG (X)
    (RETURN (COND
      ((SETQ FILE (GETATOMVAL VAR))
        (COND
          (WRITE? (COND
            ((LISTP FILE)
              (CAR FILE))
            (T FILE))
          (WRITE))
          (T (OR (ARRAYP FILE)
            (LISTP FILE)))
          (SETATOMVAL VAR (COND
            ((NLISTP (SETQ FILE (CDR (FASSOC VAR MYCINHASHFILES))))
              (OPENMYCINHASHFILE (OR FILE VAR) WRITE? SAVE NOERROR VAR))
            ((SETQ X (OPENMYCINHASHFILE (CAR FILE) WRITE? SAVE NOERROR VAR))
              (CONS X (APPEND (CDR FILE) (RETURN))))
    )))))

Called by: UGETHASHFILE
Globalvars: MYCINHASHFILES
Explanation: Opens the hashfiles indicated by VARS, a list of handles (or atom). Opens for write if WRITE? is set. If SAVE is set, adds entry to surrounding resetlist to restore the current state of the hashfiles (closed, open read...). Sets toplevel value of each of VARS to the corresponding hashfile datum, and returns a list of these data.

The hashfile names are found in the association list MYCINHASHFILES. Any var not found there is treated as a filename itself. The "name" may be a list of names, in which case the value of the hashfile variable is a list of hashfiles, the first of which is opened (use UGETHASHFILE for these multiple guys; the other files are opened only as needed).

NOERROR controls the situation when file can't be opened. If NIL, a TE is generated; if T, just quietly returns NIL; if a string, the string is printed before returning NIL.

-9-
OPENMYCINHASHFILE

(LAMBDA (FILE WRITE? SAVE NOERROR VAR)
 (PROG (HASHFILE HELPFLAG)
 (RETURN (COND
 ((AND (NEQ FILE T)
   (SETQ FILE (GETFILE FILE T)))
  (OR (SETQ HASHFILE (HASHFILEP FILE WRITE?))
   (AND (OR (EQ HASHCONFIRMFLG 'QUIET)
    (NOT WRITE?)
    (PROGN (TTYOUT1 "(Writing " FILE ")
    (COND
    (HASHCONFIRMFLG (TERPRI T)
    T)
    ((ASKYESNO NIL 'CONFIRM)
    (T (HELP))
    (NLSETQ (PROG ((BUSYCNT 0))
      RETRY
      (COND
      ((NLSETQ (COND
      ((AND WRITE? (SETQ HASHFILE (HASHFILEP FILE)))
       (e File open for READ now, so close it
        and reopen for write)
       (AND SAVE (RESETSAVE NIL (LIST 'CLOSEHASHFILE HASHFILE
        'READ)
       (CLOSEHASHFILE FILE 'WRITE))
       (T (SETQ HASHFILE (OPENHASHFILE FILE WRITE?))
       (AND SAVE (RESETSAVE NIL
       (LIST 'RESTOREHASHFILE VAR
       HASHFILE))
       (e Give RESTOREHASHFILE the MULVAR to
clear, since file won't be reopened)
       HASHFILE))
      (COND
      ((NOT (ZEROP BUSYCNT))
      (TTYOUT "free")
       (RETURN))
      (EQ BUSYCNT 10)
      (TTYOUT "timed out")
       (ERROR))
    (COND
    ((NEQ BUSYCNT 0)
    (PRIN1 '- T)
    (EQ (CAR (ERRORN))
    9)
    (TTYOUT1 "FILE " busy--")
    (T (ERROR))
    (ADD1VAR BUSYCNT)
    (DISMISS 1500)
    (GO RETRY))
    HASHFILE)
  (T (COND
  ((EQ FILE T)
  (SETQ FILE VAR))
  ((FMEMB (CAR (ERRORN))
  '(9 15 22 23))
  (TTYOUT (ERSTR)
  (COND
  ((NULL NOERROR) (* no provision for error, so report
   condition and abort))
  (ERROR "Can't open file" FILE T))
  (NEQ NOERROR T)
  (TTYOUT NOERROR))

----------
Calls: ASKYESNO GETFILE
Called by: OPENHASHFILEVARS UGETHASHFILE
Globalvars: HASHCONFIRMFLG

Explanation: Opens a single hashfile named FILE (searches for it with GETFILE), where WRITE?, SAVE, and NOERROR are as in OPENHASHFILEVARS. VAR is an atom, the hashfile variable which will be set to the hashfile datum (or data) and which here is used only when SAVE is set, to construct an appropriate reset expression (the var is reset to NIL when the hashfile is closed).

When opening for write, a warning will be printed, unless HASHCONFIRMFLG = QUIET. If HASHCONFIRMFLG is NIL, confirmation will be required. HASHCONFIRMFLG is initially T.

If the file is busy, will wait a while before giving up. A ^E typed during this wait will abort it, resulting in the usual "file won't open" error condition.

----------

PRINTPROP&VAL

(PRINTPROP&VAL
 (LAMBDA (PROP VALUE PROSEFLG)
 (PROG (**COMMENT**FLG TB)
 (SETQ TB (IPLUS (NCHARS PROP) 5))
 (TAB 2)
 (WRITE1 PROP ':)
 (COND
 (PROSEFLG (SPRINT VALUE 2 TB (IPLUS TB 3)))
 ((NILISTP VALUE)
 (SPRINT VALUE 2 TB NIL NIL NIL T))
 (COND
 (AND (CDR VALUE)
 (FMEMB PROP RULEPTRS))
 (SETQ VALUE (CONS 'Rules (RULENUMBERS VALUE)
 (T (AND (IGREATERP (NCHARS (CAR VALUE))
 (PRINTDEF VALUE TB)))
 (TERPRI))

Called by: PRINTRECORD
Globalvars: RULEPTRS

Explanation: Prints property PROP and its VALUE in a nice property-style format. PROSEFLG is set if VALUE is the output of PROSE (as in rule translation).
(PRINTRECORD
  (LAMBDA (INSTANCE RECORDNAME)
    (PROG ((DEC (OR (RECLOOK RECORDNAME)
                   (ERROR RECORDNAME "not a record" T)))
           VALUE)
      (let (FIELD)
        (for FIELD in (DREVERSE (for F in (RECORDFIELDNAMES RECORDNAME) collect F when (ANYMEMB F (CADDR DEC)
                                      when (SETQ VALUE (RECORDACCESS FIELD INSTANCE DEC)) do (PRINTPROP&VAL FIELD VALUE))
        Calls: PRINTPROP&VAL

Explanation: Prints the fields of a record. INSTANCE is the pointer to an instance of a record of type RECORDNAME.
(SPRINT)

(LAMBDA (LST INDENT PMAR LMAR LEVEL SEPR INDICATE)

(ENDWITH LSTWORD N PAREN SEPRFLG SEPRLEN)

(IOR (ARRAYP (GETATOMVAL 'SPRINTBITTABLE))

(SETQ SPRINTBITTABLE (MAKEBITTABLE '(32 45 31) (* (space, -, eol), for finding separator chars in strings)

(SETQ SEPRLEN (SELECTQ SEPR

(t 2)

(NIL 1)

(INCHARS SEPR))

(COND

((NOT INDENT)

(SETQ INDENT 0))

((ZEROP INDENT))

((EQ INDENT T)

(TERPRI) (* means start new line, at paragraph indentation)

(COND

(PMAR (TAB PMAR))

(T (SETQ PMAR 0)

((MINUSP INDENT)

(TAB (OR PMAR (SETQ PMAR 0)) 0))

((NOT (GREATERP (SETQ N (PLUS INDENT (POSITION)))

LEN))

(TAB N))

(T

(TAB (OR LMAR PMAR 0)

(OR PMAR (SETQ PMAR INDENT))

(OR LMAR (SETQ LMAR PMAR))

(OR LEVEL (SETQ LEVEL 100))))

(COND

((NLISTP LST)

(RETURN (SPRINT1 (FRPLACA (CONSTANT (CONS))

LST)

((EQ (CAR LST)

'*I)

(SETQ LST (CDR LST)

(COND

(INDICATE

(SETQ PAREN 'X)

(SETQ ENWITH 'X))

(INDICATE

(SETQ PAREN 'X)

(SETQ ENWITH 'X))

(SPRINT1 LST LEVEL)

(COND

(PAREN (PRIN1 PAREN))

(ENWITH (PRIN1 ENWITH))))

-------------

Calls: SPRINT1

Called by: ASKYESNO PRINTPROPVAL

Freevars: SPRINTBITTABLE

SPRINT

(e bvm: "16-NOV-77 08:28")

(e (space, -, eol), for finding separator chars in strings)

(e Means ",", "")

(e default of space)

(e means start new line, at paragraph indentation)

(e means begin on new line, unless already there)

(e Too far over to space any, so start new line at appropriate indentation)

(e treat non-list as one-element list)

(e ignore initial indents)

(e Show this is a list)
Explanation: Prints LST, initially spacing INDENT spaces and indenting by PMAR spaces. Linefeeds forced by line length use LMAR instead. LMAR defaults to PMAR defaults to INDENT defaults to zero. INDENT=T means start a new line at the paragraph indentation; a negative INDENT means start a new line if not there already. The special atom $L is used to represent carriage-return, linefeed. The EOL character (an atom) may also serve this function. $I causes linefeed plus indentation; $O (outdent) undoes a $I. If LST is not a list, it is treated as a one-element list. LST may contain strings, in which case they are broken at spaces as needed.

LEVEL is a printlevel parameter — lists at depth greater than LEVEL are printed as & (default is 100). SEPR is a string or atom to print between elements of LST. Default is blank. T means comma, which will not be printed after the words 'and' and 'or'.

INDICATE is used to make SPRINT look like PRINT: if LST is a list, outer parens will appear, and strings in LST will be enclosed in quotes.
(SPRINT1
  (LAMBDA (LST LEVEL)
    (PROG (WORD OPENQUOTE)
      TOP (SETQ WORD (CAR LST))
      (SETQ LST (CDR LST))
      SEL (COND
        ((STRINGP WORD) (* Print string, splitting as necessary)
          (SPRINTSTRING WORD INDICATE))
        ((LISTP WORD) (* Do lists recursively)
          (COND
            ((GREATERP LEVEL 1)
              (SPRINTPUNC '%' T)
              (SPRINT1 WORD (SUB1 LEVEL))
              (SPRINTPUNC '%) '(T)
              (SETQ SEPRFLG T))
            (T (SELECTQ WORD
              ((SL %)
               (COND
                 ((NEQ (CAR LST) '$O)  (* End of line, possibly bare EOL)
                  (SPRINTSEPR PMAR)
                  (SETQ SEPRFLG NIL))
               ($)
                (SPRINTSEPR (SETQ PMAR (IPLUS PMAR 3))
                  T)
                (SETQ LMAR (IPLUS LMAR 3))
                (SETQ SEPRFLG NIL))
               ($0)
                (COND
                  (LST (SPRINTSEPR (SETQ PMAR (IDIFFERENCE PMAR 3)))
                    (SETQ LMAR (IDIFFERENCE LMAR 3))
                    (SETQ SEPRFLG NIL)
                  ((% , : ; X %) %
                    (SPRINTPUNC WORD))
                  (%)%
                    (SPRINTPUNC WORD T))
                (SPRINTPUNC WORD (SPRINTSEPR NOT OPENQUOTE)
                    (SPRINTATOM WORD)
                    (COND
                      (NOT LST)
                      (RETURN))
                      (NLSTP LST)
                      (SPRINTATOM '%'
                      (SETQ WORD LST)
                      (SPRINTATOM LST NIL)
                      (GO SEL))
                      (T (GO TOP))))
        (T (SETQ LSTWORD WORD)
          (COND
            ((NOT LST)
              (RETURN))
            (NLSTP LST)
            (SPRINTATOM '%'
              (SETQ WORD LST)
              (SPRINTATOM LST NIL)
              (GO SEL))
            (T (GO TOP))))
    )
  )
)

Calls: SPRINT1 SPRINTATOM SPRINTPUNC SPRINTSEPR SPRINTSTRING
Called by: SPRINT SPRINT1
Freevars: INDICATE LMAR LSTWORD PMAR SEPRFLG
Explanation: Recursive subfn of SPRINT which prints LST (recurring for any elements which are themselves lists and not in excess of LEVEL arg). Dispatches to other subfn according to each element of LST.
--- [UTILITY] ---

**SPRINTATOM**

```lisp
(LAMBDA (ATM)
  (PROG (POS (LIMIT (SPRINTCOUNT)))
    (COND
     ((IGREATERP (NCHARS ATM) LIMIT)
      (COND
       ((AND (SETQ POS (STRPOSL SPRINTBITTABLE ATM))
          (NOT (IGREATERP POS LIMIT)))
        (RETURN (SPRINTSTRING ATM)
          (SPRINTSEPR LMAR))
        (T (SPRINTSEPR)))
      (PRIN1 ATM)
      (SETQ SEPRFLG T))
    )
  )
)
```

**Calls:** SPRINTCOUNT SPRINTSEPR SPRINTSTRING

**Called by:** SPRINT1

**Freevars:** LMAR SEPRFLG SPRINTBITTABLE

**Explanation:** Subfn of SPRINT to print, with appropriate separation and checks for fit, the single atom ATM.

---

**SPRINTCOUNT**

```lisp
(LAMBDA NIL
  (IDIFFERENCE LEN (IPLUS (POSITION)
                       (SELECTQ SEPRFLG
                                (NIL 0)
                                (%) 2)
                       SEPRLEN))
  (COND
   (PAREN (T 0)
           (2))
   )
)
```

**Called by:** SPRINTATOM SPRINTSTRING

**Freevars:** LEN PAREN SEPRFLG SEPRLEN

**Explanation:** Returns number of useable character positions on line, taking into account any saved chars/sepators that we are already committed to printing.
(SPRINTPUNC
  (LAMBDA (CHAR OPEN?)
    (COND
      (OPEN?
        (COND
          (PAREN
            (SPRINTSEPR))
          (SETQ PAREN CHAR))
        (T
          (COND
            ((IGREATERP (IPLUS (POSITION)
                (SELECTQ CHAR
                  ('S 's)
                )
              )
              (COND
                (PAREN 1)
                (T 0))
            (LEN)
              (TAB LMAR))))
          (COND
            (PAREN
              (PRIN1 PAREN)
              (SETQ PAREN NIL))
            (PRIN1 CHAR)
            (SEPRFLG (COND
              ((EQ CHAR '%.)
                (T T))
            )))
    )
))

SPRINTPUNC
----------
(* bvm: "16-NOV-77 08:26")
(* Save open paren for printing right before next word; we can check then if it will fit)
(* Old paren to clean up first)
(* No spacing before these)
(* do NCHARS in line)
(* No separator printed, but we'd better clean up parens)

Calls: SPRINTSEPR
Called by: SPRINT1
Freevars: LEN LMAR PAREN SEPRFLG

Explanation: Handles punctuation for SPRINT. If OPEN? is true, treats CHAR as "opening" punctuation (spaces before, but not after, e.g. open paren); otherwise as "closing" (spaces after, not before). OPEN? is currently variable only for the character "."
bvm: 19-NOV-77 15:08 [UTILITY]

(SPRINTSEPR
  (LAMBDA (NEWLINE DONTFORCE)
    (SELECTQ SEPRFLG
      (%
        (COND
          ((NOT NEWLINE)
            (SPACES 2)
            (NIL)
            (SELECTQ SEPR
              [NIL
                [OR NEWLINE (PRIN1 ' \n)]]
              [T
                (COND
                  ((FMEMB LSTWORD '(and or))
                    (OR NEWLINE (PRIN1 ' \n) )
                    (NEWLINE (PRIN1 "", )))
                  (T (PRIN1 ", "))
                (PRINT SEPR))))
            )))
          )
        (DONTFORCE)
        (TAB NEWLINE))
        (T (TERPRI)
          (TAB NEWLINE 0)))))
    (COND
      (PAREN
        (PRIN1 PAREN)
        (SETQ PAREN NIL)))
    (SETQ SEPRFLG NIL))

********

Called by: SPRINT1 SPRINTATOM SPRINTPUNC SPRINTSTRING

Freevars: LSTWORD PAREN SEPR SEPRFLG

Explanation: Subfn of SPRINT to print any separator chars needed (when SEPRFLG is set). Also includes any backed up PAREN. If NEWLINE is set, the separation is between lines, and NEWLINE is the tab stop for the new line.
(SPRINTSTRING
 (LAMBDA (STRING SHOWQUOTE)
   (bind [#SPACES + (IPLUS (SPRINTCOUNT))]
     (COND
       (SHOWQUOTE -2)
       (T 0)
       (#CHARS -(NCHARS STRING))
       QFLG-SHOWQUOTE
       BRKPOS CH comment
     while (COND
       (ILESSP #CHARS #SPACES)
       (STRPOS EOL STRING))
     do (bind N-1 while (AND (SETQ N (STRPOS SPRINTBITTABLE STRING (ADD1 N)))
       (NOT (IGREATERP N #SPACES))
       (COND
         ((EQ (SETQ CH (NTHCHAR STRING N))
           '-)
          (ILESSP (ADD1 N)
            #CHARS)
         do

       (* Set BRKPOS to be the last space before linelength runs out, or where EOL appears.
       #CHARS check assures that we don't break a short hyphenated atom over a line, e.g. CULTURE-1)

       (SETQ BRKPOS N)
       repeatwhile (NEQ CH EOL))
     (COND
       (BRKPOS (SPRINTSEPR)
       (COND
         (QFLG (PRIN1 '%")
           (SETQ QFLG NIL)))
         (PRIN1 (SUBSTRING STRING 1 (COND
           ((EQ CH '-)
            BRKPOS)
           (T (SUB1 BRKPOS)))
           (CONSTANT (CONCAT) (* Scratch string, so we don't eat up too many string pointers)
           T (SETQ BRKPOS 0)))
         repeatuntil (NEQ (SETQ CH (NTHCHAR STRING (ADD1VAR BRKPOS)))
           'Z'))
       (COND
         ((EQ CH EOL)
           (ADD1VAR BRKPOS))
         (EQ BRKPOS 1))
       (NOT (IGREATERP (SETQ #CHARS (ADD1 (IDIFFERENCE #CHARS BRKPOS)))
         0))
       (RETURN))
     T (SETQ STRING (SUBSTRING STRING BRKPOS NIL (CONSTANT (CONCAT)
       (SPRINTSEPR LMR)
       (SETQ #SPACES (IDIFFERENCE (IDIFFERENCE LEN LMR) 2))
       (COND
         (SHOWQUOTE (SUBIVAR #SPACES))
         (SETQ BRKPOS #SPACES)
       finally (COND
         (OR QFLG (NOT (ZEROQ #CHARS)))
         (SPRINTSEPR)
       (COND
         (QFLG (PRIN1 'Z")
         (PRIN1 STRING)
       (e bvm: "9-JAN-78 22:58")
       (e Note that #SPACES and BRKPOS are always shorter than linelength, hence small integers. Thus the ADD1VARs work)

       (e Set BRKPOS to this in case there isn't a place to break)

       (e This is new value of SPRINTCOUNT)

       (e Scratch string, so we don't eat up too many string pointers)

       (e Must indicate quotes)
COND
(SHOWQUOTE (PRIN1 'Z')
(SETQ SEPRFLG (COND
((EQ (NTHCHAR STRING -1) '%) ')
'(T T))
(\T T)))

Calls: SPRINTCOUNT SPRINTSEPR
Called by: SPRINT1 SPRINTATOM
Globalvars: EOL
Freevars: LEN LMAR SEPRFLG SPRINTBITTABLE

Explanation: Subfn of SPRINT to print a string, splitting at spaces, hyphens and carriage returns as needed. SHOWQUOTE is true if the enclosing quotes are to be printed as well.

TTYOUT

TTYOUT
(UTILITY)

TTYOUT
ILAMBDA N
(for I from 1 to N do (WRITEARG (ARG N I)
T))

Calls: WRITEARG
Called by: CNET GETFILE OPENMYCINHASHFILE QSET!PARAMETERS RESIMULATE SET!PARAMETERS

Explanation: WRITE to terminal.
UGETHASHFILE

(UGETHASHFILE
 (LAMBDA (HASHFILES KEY1 KEY2 ACCESS NOERROR)
   (COND
    (OR (NOT (LITATOM HASHFILES))
      (SETQ HASHFILES (OR (ARRAYP (GETATOMVAL HASHFILES))
        (LISTP (GETATOMVAL HASHFILES))
        (CAR (OPENHASHFILEVARS HASHFILES NIL NIL NOERROR)
        (* HASHFILES can name a hashfile
           variable)
        (e normal get)
        (e See if there, but don't retrieve)
      (SELECTQ ACCESS
       (NIL (* normal get)
       (SETQQ ACCESS RETRIEVE))
       (LOOK (* See if there, but don't retrieve)
       (SETQ ACCESS NIL))
      NIL)
    (COND
     ((NLISTP HASHFILES)
      (LOOKUPHASHFILE KEY1 NIL HASHFILES ACCESS KEY2))
     (T any (LOOKUPHASHFILE KEY1 NIL OR (ARRAYP (CAR HASHFILES))
      (CAR (FRPLACA HASHFILES (OR (OPENMYCINHASHFILE (CAR HASHFILES) NIL NIL NOERROR)
      (RETURN)
      (* Returns if any value found)
      (RETURN)
      (* Returns T if any value found)
      (RETURN)
      repeatwhile (SETQ HASHFILES (CDR HASHFILES)
    Calls: OPENHASHFILEVARS OPENMYCINHASHFILE
    Called by: DISPLAYHELP

Explanation: Universal GETHASHFILE. Looks up in HASHFILES the entry indexed by KEY1 (and KEY2). HASHFILES may be
an open hashfile, list of hashfiles, or an atomic hashfile variable (i.e. anything that OPENHASHFILEVARS
will accept). ACCESS is NIL for a normal GETHASHFILE; ACCESS=LOOK means lookup but don’t return the value
(just return T if ANY value found); other values of ACCESS are passed directly to LOOKUPHASHFILE. If
HASHFILES is a hashfile variable (litatom), it will be opened. If HASHFILES is (or becomes thereby) a
list, UGETHASHFILE looks up in each hashfile, returning the first non-NIL value found; if an element of
this list is a filename instead of an open hashfile, it opens it and smashes the hashfile into the list.
If HASHFILES is a non-list, behaves like a single hashfile lookup. NOERROR is passed to OPENHASHFILEVARS.
KNOWNFILE that the function is on (used for advice in the editor). If FN is a list, the above is done for
each function in the list, and the result is the union of all the files. INTERNAL=T means KNOWNFILE is the
full filename (otherwise needs to obtain the full name for comparison’s sake). KNOWNFILE=T means simply
print out ALL files containing FN(s).

WRITE

(WRITE
 (LAMBDA N
   (for I from 1 to N do (WRITEARG (ARG N I)))
   (TERPRI)
   (-2 1 -)
   Calls: WRITEARG
   Called by: ASKFORFILENAME ASKYESNO ATTRIBUTEP CILPARSE INTERPRET OBJECTP

Explanation: Takes arbitrary number of arguments, each of which is PRIN1ed to the primary output file, followed by
crlf. If an argument is a list, it is MAPRINTed, i.e. the outer “parentheses” will not appear.
WRITE1

(\(\lambda\) N
  (for I from 1 to N do (WRITEARG (ARG N I)))
)

Calls: WRITEARG

Called by: ASKFORFILENAME PRINTPROP&VAL

Explanation: WRITE without the final crlf.

WRITE3

(\(\lambda\) N
  (for I from 1 to N do (PRIN3 (ARG N I)))
)

Explanation: A WRITE1 that ignores line length, i.e. does PRIN3's.

WRITEARG

(\(\lambda\) X FILE)

(COND
  ((NLISTP X)
   (PRIN1 X FILE))
  (T (MAPRINT X FILE)))

Called by: DISPLAY TTYOUT WRITE WRITE1

Explanation: If X is not a list, PRIN1's it to FILE, otherwise MAPRINT's it, so that the outer parens will not appear.
CONTRACT NET Simulation Parameters

Normal Mode
Number of Processor Nodes in Net: 5
Applications time unit expansion: 100
Contracts held in terminated state: 10

CONTRACT NET Delay Parameters:
- Time to make a task announcement: 1
- Time before a task is reannounced: 1000
- Time to process a task announcement: 1
- Time to make a node availability announcement: 1
- Time to process a node availability announcement: 1
- Time to make a bid: 1
- Time to process a bid: 1
- Time to make a standard award: 1
- Time to process a standard award: 1
- Time to make a directed award: 1
- Time to process a directed award: 1
- Time to acknowledge a directed award: 1
- Time to process an acknowledgement: 1
- Time to make a report to another node: 1
- Time to process a report: 1
- Time to generate a termination: 1
- Time to process a termination: 1
- Time to generate a request: 1
- Time to process a request: 1
- Time to generate an information message: 1
- Time to process an information message: 1

":::::::::::::::: Start of Simulation'::::::::::::::::"

Number of Queens [5] ** 4
Number of solutions [1] ** 2
Search Strategy [0] ** 2
Report Strategy [0] ** 1

: Time: 0

-- Node Status --
To: *
From: 1
Type: task announcement
Contract: 1 1

: Time: 202

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: (1 1)
Suspended: NIL
Terminated: NIL

To: 1
From: 2
Type: bid
Contract: 1 1

To: 1
From: 3
Type: bid
Contract: 1 1

To: 1
From: 4
Type: bid
Contract: 1 1

To: 1
From: 5
Type: bid
Contract: 1 1

: Time: 204

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

To: 1
From: 1
Type: standard award
Contract: 1 1

node: 2
contract: 1 1
internal event: contract processing

From: 2

: Time: 400

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

node: 1
contract: 1
internal event: node update

From: 1

Generated Board--> Queen-rows: 2

node: 1
contract: 1
internal event: node update

: Time: 401

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: (2 1)
Suspended: NIL
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL
To: 2
From: 1
Type: task announcement
Contract: 2 1

: Time: 402

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: (2 1)
Suspended: NIL
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

: Time: 404

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

To: 3
From: 1
Type: standard award
Contract: 2 1

node: 3
calendar: 2 1
internal event: contract processing

From: 3

Started Processing Contract 2 1

: Time: 600

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (2 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

node: 1
calendar: 1
internal event: node update

From: 1

Generated Board--> Queen-rows: 3

node: 1
calendar: 1
internal event: node update

: Time: 691

-- Node Status --
Node 1
Executing: (1)
Ready: NIL
Announced: (3 1)
Suspended: NIL
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (2 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

To: *
From: 1
Type: task announcement
Contract: 3 1

Time: 602

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: (3 1)
Suspended: NIL
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (2 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

To: 4
From: 1
Type: bid
Contract: 3 1

Time: 605

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (2 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

To: 4
From: 1
Type: standard award
Contract: 3 1

From: 2
Generated Board--> Queen-rows: 1 3

node: 2
contract: 1 1
internal event: node update

node: 2
contract: 1 1
internal event: node update

node: 4
contract: 3 1
internal event: contract processing

From: 4
Started Processing Contract 3 1

Time: 605
Node 1  
Executing: (1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

Node 2  
Executing: (1 1)  
Ready: NIL  
Announced: (1 1 1)  
Suspended: NIL  
Terminated: NIL  

Node 3  
Executing: (2 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

Node 4  
Executing: (3 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

To: 2  
From: 5  
Type: bid  
Contract: 1 1 1  
Time: 608  

-- Node Status --  

Node 1  
Executing: (1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

Node 2  
Executing: (1 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

Node 3  
Executing: (2 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

Node 4  
Executing: (3 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

To: 5  
From: 2  
Type: standard award  
Contract: 1 1 1  
Time: 606  

-- Node Status --  

Node 1  
Executing: (1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

Node 2  
Executing: (1 1)  
Ready: NIL  
Announced: (1 1 1)  
Suspended: NIL  
Terminated: NIL  

Node 3  
Executing: (2 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

Node 4  
Executing: (3 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

To: *  
From: 2  
Type: task announcement  
Contract: 1 1 1  
Time: 606
node: 5
contract: 1 1 1
Internal event: contract processing

From: 5

Started Processing Contract 1 1 1

: Time: 800

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (2 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 4
Executing: (3 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 5
Executing: (1 1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

node: 1
contract: 1
internal event: node update

From: 1

Generated Board--> Queen-rows: 4

node: 1
contract: 1
internal event: node update

From: 1

Suspended Contract 1

: Time: 801

-- Node Status --

Node 1
Executing: NIL
Ready: NIL
Announced: (4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (2 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 4
Executing: (3 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 5
Executing: (1 1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Tol: 4
From: 1
Type: task announcement
Contract: 4 1
To: 2
From: 1
Type: bid
Contract: 1 1

: Time: 802

-- Node Status --

Node 1
Executing: NIL
Ready: NIL
Announced: (4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (2 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 4
Executing: (3 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 5
Executing: (1 1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

node: 2
contract: 1 1
internal event: node update

node: 4
contract: 3 1
internal event: node update

To: 1
From: 1
Type: standard award
Contract: 4 1

From: 2
Generated Board--> Queen-rows: 1 4

node: 2
contract: 1 1
internal event: node update

From: 4
Generated Board--> Queen-rows: 3 1

node: 4
contract: 3 1
internal event: node update
node: 1  
contract: 4 1  
internal event: contract processing  

node: 2  
contract: 1 1  
internal event: node update  
From: 1  
Started Processing Contract 4 1  

; Time: 805  

-- Node Status --  

Node 1  
Executing: (4 1)  
Ready: NIL  
Announced: NIL  
Suspended: (1)  
Terminated: NIL  

Node 2  
Executing: NIL  
Ready: NIL  
Announced: (2 1 1)  
Suspended: (1 1)  
Terminated: NIL  

Node 3  
Executing: (2 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

Node 4  
Executing: (3 1)  
Ready: NIL  
Announced: (1 3 1)  
Suspended: NIL  
Terminated: NIL  

Node 5  
Executing: (1 1 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

To: *  
From: 4  
Type: task announcement  
Contract: 1 3 1  

To: 2  
From: 2  
Type: bid  
Contract: 1 1 1  

To: 2  
From: 2  
Type: task announcement  
Contract: 2 1 1  

; Time: 806  

-- Node Status --  

Node 1  
Executing: (4 1)  
Ready: NIL  
Announced: NIL  
Suspended: (1)  
Terminated: NIL  

Node 2  
Executing: NIL  
Ready: NIL  
Announced: (2 1 1)  
Suspended: (1 1)  
Terminated: NIL  

Node 3  
Executing: (2 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

Node 4  
Executing: (3 1)  
Ready: NIL  
Announced: (1 3 1)  
Suspended: NIL  
Terminated: NIL  

Node 5  
Executing: (1 1 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

To: 2  
From: 2  
Type: bid  
Contract: 2 1 1
To: 4
From: 2
Type: bid
Contract: 1 3 1

: Time: 888

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: NIL
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 3 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL

Node 3
Executing: (2 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 4
Executing: (3 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 5
Executing: (1 1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

To: 2
From: 2
Type: standard award
Contract: 2 1 1

node: 3
contract: 2 1
internal event: node update
From: 3
Generated Board--> Queen-rows: 2 4

node: 3
contract: 2 1
internal event: node update
From: 3
Suspected Contract 2 1

node: 3
contract: 2 1
internal event: node update
From: 3
Suspected Contract 2 1

: Time: 904

-- Node Status --
Node 5
Executing: (1 1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

To: 3
From: 3
Type: standard award
Contract: 1 2 1

node: 3
contract: 1 2 1
internal event: contract processing

From: 3
Started Processing Contract 1 2 1

Time: 1004

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: NIL
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 3 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (3 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 5
Executing: (1 1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL
	node: 1
contract: 4 1
internal event: node update

From: 2
Suspended Contract 1 1

From: 1
Generated Board--> Queen-rows: 4 1

node: 1
contract: 4 1
internal event: node update

Time: 1005

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 3 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (3 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 5
Executing: (1 1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL
To:  
From: 1  
Type: task announcement  
Contract: 1 4 1  

: Time: 1184  

-- Node Status --  

Node 1  
Executing: (4 1)  
Ready: NIL  
Announced: (1 4 1)  
Suspended: (1)  
Terminated: NIL  

Node 2  
Executing: (2 1 1)  
Ready: (1 3 1)  
Announced: NIL  
Suspended: (1 1)  
Terminated: NIL  

Node 3  
Executing: (1 2 1)  
Ready: NIL  
Announced: NIL  
Suspended: (2 1)  
Terminated: NIL  

Node 4  
Executing: (3 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

Node 5  
Executing: (1 1 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

node: 4  
contract: 3 1  
internal event: node update  

: Time: 1188  

-- Node Status --  

Node 1  
Executing: (4 1)  
Ready: NIL  
Announced: (1 4 1)  
Suspended: (1)  
Terminated: NIL  

Node 2  
Executing: (2 1 1)  
Ready: (1 3 1)  
Announced: NIL  
Suspended: (1 1)  
Terminated: NIL  

Node 3  
Executing: (1 2 1)  
Ready: NIL  
Announced: NIL  
Suspended: (2 1)  
Terminated: NIL  

Node 4  
Executing: (3 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: NIL  

node: 2  
contract: 2 1 1  
internal event: node update  

node: 3  
contract: 1 2 1  
internal event: node update  

node: 5  
contract: 1 1 1  
internal event: node update  

node: 5  
internal event: node update  

node: 5  
contract: 1 1 1  
internal event: node update  

From: 2  
Generated Board--> Queen-rows: 1 4 2  
node: 2  
contract: 2 1 1  
internal event: node update  

From: 3  
Generated Board--> Queen-rows: 2 4 1  

-12-
node 3
contract: 1 2 1
internal event: node update
From: 5

Terminated Contract 1 1 1

; Time: 1110

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 3 1)
Announced: (1 2 1 1)
Suspended: (1 1)
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: (1 1 2 1)
Suspended: (1)
Terminated: NIL

To: 2
From: 5
Type: bid
Contract: 1 2 1 1

To: 3
From: 5
Type: bid
Contract: 1 1 2 1

; Time: 1112

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: (1 2 1 1)
Suspended: NIL
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: (1 1 2 1)
Suspended: (2)
Terminated: NIL

To: 2
From: 4
Type: bid
Contract: 1 2 1 1

To: 3
From: 5
Type: bid
Contract: 1 1 2 1

To: 3
From: 5
Type: bid
Contract: 1 1 2 1

; Time: 1110

-- Node Status --
Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

To: 4
From: 2
Type: standard award
Contract: 1 2 1

node: 3
contract: 1 1
internal event: pseudo contract
node: 3
contract: 1 1
internal event: pseudo contract
node: 4
contract: 1 1
internal event: pseudo contract

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

node: 1
contract: 1 1
internal event: bid check

: Time: 1204

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

node: 2
contract: 1 1
internal event: pseudo contract
node: 3
contract: 1 1
internal event: pseudo contract
node: 4
contract: 1 1
internal event: pseudo contract

To: 4
From: 2
Type: standard award
Contract: 1 2 1 1

: Time: 1200

Started Processing Contract 1 2 1 1

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

node: 2
contract: 1 1
internal event: pseudo contract
node: 3
contract: 1 1
internal event: pseudo contract
node: 4
contract: 1 1
internal event: pseudo contract

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

node: 2
contract: 1 1
internal event: pseudo contract
node: 3
contract: 1 1
internal event: pseudo contract
node: 4
contract: 1 1
internal event: pseudo contract

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

node: 2
contract: 1 1
internal event: pseudo contract
node: 3
contract: 1 1
internal event: pseudo contract
node: 4
contract: 1 1
internal event: pseudo contract

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

node: 2
contract: 1 1
internal event: pseudo contract
node: 3
contract: 1 1
internal event: pseudo contract
node: 4
contract: 1 1
internal event: pseudo contract
node: 5
contract: 1 1
internal event: pseudo contract

node: 1
contract: 4 1
internal event: node update
From: 1
Generated Board--> Queen-rows: 4 2

node: 1
contract: 4 1
internal event: node update

Time: 1205

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (2 4 1) (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

To: 1
From: 5
Type: bid
Contract: 2 4 1

Time: 1208

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (2 4 1) (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

To: 1
From: 5
Type: task announcement
Contract: 2 4 1

Time: 1206

-- Node Status --
Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

To: 5
From: 1
Type: standard award
Contract: 2 4 1

node: 5
contract: 2 4 1
internal event: contract processing

From: 5
Started Processing Contract 2 4 1

: Time: 1304

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 1)

From: 4
Suspended Contract 3 1

: Time: 1308

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 1) (1 3 1)
Announced: NIL
Suspended: NIL
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 1)

node: 2
contract: 2 1 1
internal event: node update

: Time: 1309

-- Node Status --
Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (1 3 1)
Ready: (1 3 1)
Announced: NIL
Suspended: (1)
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: (1 2 1)
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 1)

node: 2
contract: 1 1
internal event: contract processing

node: 2
contract: 1 1
internal event: internal event: contract processing

node: 2
contract: 1 1
internal event: node update
From: 2
Suspension Contract 1 1

Time: 1310

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (1 3 1)
Ready: NIL
Announced: NIL
Suspended: (1)
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 1)

node: 2
contract: 1 3 1
internal event: contract processing

Time: 1400

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (1 3 1)
Ready: NIL
Announced: NIL
Suspended: (1)
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 1)
Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 1)

node: 1
contract: 2 1
internal event: bid check

node: 3
contract: 2 1
internal event: pseudo contract

node: 4
contract: 2 1
internal event: pseudo contract

node: 5
contract: 2 1
internal event: pseudo contract

node: 1
contract: 4 1
internal event: node update

Time: 1404

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (1 3 1)
Ready: NIL
Announced: NIL
Suspended: (1 1) (2 1 1)
Terminated: NIL

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 1)
Node 5  
Executing: (2 4 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: (1 1 1)

node: 3  
contract: 1 2 1  
internal event: node update

: Time: 1600

-- Node Status --

Node 1  
Executing: NIL  
Ready: NIL  
Announced: (1 4 1)  
Suspended: (4 1) (1)  
Terminated: NIL

Node 2  
Executing: (1 3 1)  
Ready: NIL  
Announced: NIL  
Suspended: (1 1) (2 1 1)  
Terminated: NIL

Node 4  
Executing: (1 2 1 1)  
Ready: (1 1 2 1)  
Announced: NIL  
Suspended: (3 1)  
Terminated: NIL

Node 5  
Executing: (2 4 1)  
Ready: NIL  
Announced: NIL  
Suspended: NIL  
Terminated: (1 1 1)

node: 1  
contract: 3 1  
internal event: bid check

: Time: 1604

-- Node Status --

Node 1  
Executing: NIL  
Ready: NIL  
Announced: (1 4 1)  
Suspended: (4 1) (1)  
Terminated: NIL

Node 2  
Executing: (1 3 1)  
Ready: NIL  
Announced: NIL  
Suspended: (1 1) (2 1 1)  
Terminated: NIL

Node 4  
Executing: (1 2 1 1)  
Ready: (1 1 2 1)  
Announced: NIL  
Suspended: (3 1)  
Terminated: NIL

node: 5  
contract: 3 1  
internal event: pseudo contract

node: 2  
contract: 1 1 1  
internal event: bid check

: Time: 1608

-- Node Status --

Node 1  
Executing: NIL  
Ready: NIL  
Announced: (1 4 1)  
Suspended: (4 1) (1)  
Terminated: NIL

Node 2  
Executing: (1 3 1)  
Ready: NIL  
Announced: NIL  
Suspended: (1 1) (2 1 1)  
Terminated: NIL
Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 1)

node: 5
contract: 1 1 1
internal event: pseudo contract

: Time: 1612

-- Node Status --

Node 1
Executing: NIL
Ready: NIL
Announced: (1 4 1)
Suspended: (4 1) (1)
Terminated: NIL

Node 2
Executing: (1 3 1)
Ready: NIL
Announced: NIL
Suspended: (1 1) (2 1 1)
Terminated: NIL

Node 4
Executing: (1 2 1 1)
Ready: (1 1 2 1)
Announced: NIL
Suspended: (3 1)
Terminated: NIL

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 1)

node: 4
contract: 1 2 1 1
internal event: node update

From: 4
Terminated Contract 1 2 1 1

: Time: 1613

-- Node Status --

Node 1
Executing: NIL
Ready: NIL
Announced: (1 4 1)
Suspended: (4 1) (1)
Terminated: NIL

Node 2
Executing: (1 3 1)
Ready: NIL
Announced: NIL
Suspended: (1 1) (2 1 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 1)

To: 2
From: 4
Type: final report
Contract: 1 2 1 1

node: 4
contract: 1 1 2 1
internal event: contract processing

From: 4
Started Processing Contract 1 1 2 1

: Time: 1708

-- Node Status --

-20-
Node 1
Executing: NIL
Ready: NIL
Announced: (1 4 1)
Suspended: (4 1) (1)
Terminated: NIL

Node 2
Executing: (1 3 1)
Ready: (2 1 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

Node 5
Executing: (2 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 2 1 1)

node: 5
contract: 2 4 1
internal event: node update

node: 5
contract: 2 4 1
internal event: node update

From: 5
Terminated Contract 2 4 1

: Time: 1709

-- Node Status --

Node 1
Executing: NIL
Ready: NIL
Announced: (1 4 1)
Suspended: (4 1) (1)
Terminated: NIL

Node 2
Executing: (1 3 1)
Ready: (2 1 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

To: 1
From: 5
Type: final report
Contract: 2 4 1

: Time: 1710

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: (1 4 1)
Suspended: (1)
Terminated: NIL

Node 2
Executing: (1 3 1)
Ready: (2 1 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

node: 1
contract: 4 1
internal event: node update

node: 1
contract: 4 1
internal event: node update

From: 1
Suspended Contract 4 1

: Time: 1800

-- Node Status --
<table>
<thead>
<tr>
<th>Node 1</th>
<th>Executing: NIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready: NIL</td>
<td></td>
</tr>
<tr>
<td>Announced: (1 4 1)</td>
<td></td>
</tr>
<tr>
<td>Suspended: (6 1) (1)</td>
<td></td>
</tr>
<tr>
<td>Terminated: NIL</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node 2</th>
<th>Executing: (1 3 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready: (2 1 1)</td>
<td></td>
</tr>
<tr>
<td>Announced: NIL</td>
<td></td>
</tr>
<tr>
<td>Suspended: (1 1)</td>
<td></td>
</tr>
<tr>
<td>Terminated: NIL</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node 4</th>
<th>Executing: (1 1 2 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready: NIL</td>
<td></td>
</tr>
<tr>
<td>Announced: NIL</td>
<td></td>
</tr>
<tr>
<td>Suspended: (3 1)</td>
<td></td>
</tr>
<tr>
<td>Terminated: (1 2 1 1)</td>
<td></td>
</tr>
</tbody>
</table>

---

Node 1
contract: 4 1
internal event: bid check

Time: 1803

---

--- Node Status ---

Node 1
executing: NIL
ready: NIL
announced: (1 4 1)
suspended: (6 1) (1)
terminated: NIL

node: 2
contract: 2 1 1
internal event: bid check

node: 4
contract: 1 3 1
internal event: bid check

Time: 1807

--- Node Status ---

Node 1
executing: NIL
ready: NIL
announced: (1 4 1)
suspended: (6 1) (1)
terminated: NIL

node: 2
contract: 2 1 1
internal event: pseudo contract

node: 4
contract: 1 3 1
internal event: bid check
Node 2
Executing: (1 3 1)
Ready: (2 1 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

node: 2
correct: 1 1 1
internal event: pseudo contract

: Time: 1888

-- Node Status --

Node 1
Executing: NIL
Ready: NIL
Announced: (1 4 1)
Suspended: (4 1) (1)
Terminated: NIL

Node 2
Executing: (1 3 1)
Ready: (2 1 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

node: 2
correct: 1 3 1
internal event: pseudo contract

From: 2
Generated Board--> Queen-rows: 3 1 4

node: 2
correct: 1 3 1
internal event: pseudo contract

node: 2
correct: 1 3 1
internal event: pseudo contract

From: 2
Suspended Contract 1 3 1

: Time: 1811

-- Node Status --

Node 1
Executing: NIL
Ready: NIL
Announced: (1 4 1)
Suspended: (4 1) (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: NIL
Announced: (1 3 1)
Suspended: (1 1) (1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)
Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

To: 4
From: 2
Type: task announcement
Contract: 1 1 3 1

node: 2
contract: 2 1 1
internal event: node update

node: 2
contract: 2 1 1
internal event: node update

node: 2
contract: 2 1 1
internal event: node update

node: 2
contract: 2 1 1
internal event: node update

To: 2
From: 3
Type: bid
Contract: 1 1 3 1

To: 2
From: 5
Type: bid
Contract: 1 1 3 1

To: 2
From: 2
Type: final report
Contract: 2 1 1

1 Time: 1813

-- Node Status --

Node 1
Executing: NIL
Ready: NIL
Announced: (1 4 1)
Suspended: (4 1) (1)
Terminated: NIL

Node 2
Executing: (1 1)
Ready: NIL
Announced: (1 1 3 1)
Suspended: (1 3 1) (1 1)
Terminated: (2 1 1)

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

To: 2
From: 1
Type: bid
Contract: 1 1 3 1

node: 2
contract: 1 1
internal event: node update

node: 2
contract: 1 1
internal event: node update

node: 2
contract: 1 1
internal event: node update

Time: 1812
node: 2
contract: 1 1
internal event: node update
From: 2

Terminated Contract 1 1

: Time: 1814

-- Node Status --

Node 1
Executing: NIL
Ready: NIL
Announced: (1 4 1)
Suspended: (4 1) (1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

To: 1
From: 2
Type: standard award
Contract: 1 1 3 1

To: 1
From: 2
Type: final report
Contract: 1 1

node: 1
contract: 1 1 3 1
internal event: contract processing
From: 1

Started Processing Contract 1 1 3 1

: Time: 1907

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: (1 4 1)
Suspended: (4 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

node: 3
contract: 1 2 1
internal event: bid check
From: 1

Suspended Contract 4 1

: Time: 1908

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: (1 4 1)
Suspended: (4 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

node: 3
contract: 1 1 1
internal event: pseudo contract

: Time: 1908

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: (1 4 1)
Suspended: (4 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

-25-
node: 3
contact: 1 2 1
internal event: pseudo contract

From: 3
Suspended Contract 1 2 1

To: *
From: 1
Type: task announcement
Contract: 1 4 1

: Time: 2004

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: (1 4 1)
Suspended: (4 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

node: 1
contract: 1 4 1
internal event: bid check

: Time: 2005

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: (1 4 1)
Suspended: (4 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

To: 1
From: 2
Type: bid
Contract: 1 4 1

To: 1
From: 3
Type: bid
Contract: 1 4 1

To: 1
From: 5
Type: bid
Contract: 1 4 1

: Time: 2008

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: NIL

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)
To: 2
From: 1
Type: standard award
Contract: 1 4 1

node: 2
contract: 1 4 1
internal event: contract processing

From: 2
Started Processing Contract 1 4 1

: Time: 2014

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: NIL

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)

Node 4
Executing: (1 1 2 1)
Ready: NIL
Announced: NIL
Suspended: (3 1)
Terminated: (1 2 1 1)

node: 4
contract: 1 1 2 1
internal event: node update

From: 4
Terminated Contract 1 1 2 1

: Time: 2013

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: NIL

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL

node: 4
contract: 1 1 2 1
internal event: node update

Generated Board--> Queen-rows: 2 4 1 3

node: 4
contract: 1 1 2 1
internal event: node update

From: 4
Type: final report
Contract: 1 1 2 1

: Time: 2015

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: NIL

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)

Node 3
Executing: (1 2 1)
Ready: NIL
Announced: NIL
Suspended: (2 1)
Terminated: NIL
node: 3
custom: 1 2 1
internal event: node update

node: 3
custom: 1 2 1
internal event: node update

node: 3
custom: 1 2 1
internal event: node update

From: 3

Terminated Contract 1 2 1

Time: 2016

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: NIL

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)

To: 3
From: 3
Type: final report
Contract: 1 2 1

Time: 2017

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: NIL

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)

From: 3

Terminated Contract 2 1

Time: 2018

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: NLT

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)

To: 1
From: 3
Type: final report
Contract: 2 1

Time: 2108

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: NLT

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)
Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)

node: 2
contract: 1 2 1 1
internal event: bid check

node: 3
contract: 1 1 2 1
internal event: bid check

: Time: 2112

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: NIL

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)

node: 4
contract: 1 2 1 1
internal event: pseudo contract

node: 5
contract: 1 2 1 1
internal event: pseudo contract

node: 4
contract: 1 1 2 1
internal event: pseudo contract

node: 5
contract: 1 1 2 1
internal event: pseudo contract

: Time: 2114

-- Node Status --

Node 1
Executing: (1 1 3 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: NIL

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)

node: 1
contract: 1 1 3 1
internal event: node update
From: 1

Generated Board--> Queen-rows: 3 1 4 2

node: 1
contract: 1 1 3 1
internal event: node update

node: 1
contract: 1 1 3 1
internal event: node update
From: 1

Terminated Contract 1 1 3 1

: Time: 2115

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: NIL
Suspended: (4 1)
Terminated: (1 1 3 1)

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1 3 1)
Terminated: (1 1) (2 1 1)

To: 2
From: 1
Type: final report
Contract: 1 1 3 1
node: 1
contract: 1
internal event: node update

node: 1
contract: 1
internal event: node update

From: 1
Suspended Contract 1

: Time: 2204

-- Node Status --

Node 2
Executing: (1 4 1)
Ready: (1 3 1)
Announced: NIL
Suspended: NIL
Terminated: (1 1) (2 1 1)

node: 1
contract: 2 4 1
internal event: bid check

: Time: 2208

-- Node Status --

Node 2
Executing: (1 4 1)
Ready: (1 3 1)
Announced: NIL
Suspended: NIL
Terminated: (1 1) (2 1 1)

node: 5
contract: 2 4 1
internal event: pseudo contract

: Time: 2408

-- Node Status --

Node 2
Executing: (1 4 1)
Ready: (1 3 1)
Announced: (1 1 4 1)
Suspended: NIL
Terminated: (1 1) (2 1 1)

node: 2
contract: 1 4 1
internal event: node update

From: 2
Generated Board--> Queen-rows: 4 1 3

node: 2
contract: 1 4 1
internal event: node update

: Time: 2409

-- Node Status --

Node 2
Executing: (1 4 1)
Ready: (1 3 1)
Announced: (1 1 4 1)
Suspended: NIL
Terminated: (1 1) (2 1 1)

To: *
From: 2
Type: task announcement
Contract: 1 1 4 1

: Time: 2410

-- Node Status --

Node 2
Executing: (1 4 1)
Ready: (1 3 1)
Announced: (1 1 4 1)
Suspended: NIL
Terminated: (1 1) (2 1 1)

To: 2
From: 1
Type: bid
Contract: 1 1 4 1

To: 2
From: 3
Type: bid
Contract: 1 1 4 1

To: 2
From: 4
Type: bid
Contract: 1 1 4 1

-38-
To: 2  
From: 5  
Type: bid  
Contract: 1 1 4 1

: Time: 2412

-- Node Status --

Node 2  
Executing: (1 4 1)  
Ready: (1 3 1)  
Announced: NIL  
Suspended: NIL  
Terminated: (1) (2 1 1)

To: 1  
From: 2  
Type: standard award  
Contract: 1 1 4 1

node: 1  
contract: 1 1 4 1  
internal event: contract processing

From: 1  
Started Processing Contract 1 1 4 1

: Time: 2508

-- Node Status --

Node 1  
Executing: (1 1 4 1)  
Ready: NIL  
Announced: NIL  
Suspended: (1) (4 1)  
Terminated: (1 1 3 1)

Node 2  
Executing: (1 4 1)  
Ready: (1 3 1)  
Announced: NIL  
Suspended: (1 4 1)  
Terminated: (1) (2 1 1)

node: 2  
contract: 1 3 1  
internal event: node update

node: 2  
contract: 1 3 1  
internal event: node update

From: 2  
Terminated Contract 1 3 1

: Time: 2510

-- Node Status --

Node 2  
Executing: (1 4 1)  
Ready: NIL  
Announced: NIL  
Suspended: (1 4 1)  
Terminated: (1 1 3 1)

To: 4  
From: 2  
Type: final report  
Contract: 1 3 1

: Time: 2511

-- Node Status --

Node 1  
Executing: (1 1 4 1)  
Ready: NIL  
Announced: NIL  
Suspended: (1) (4 1)  
Terminated: (1 1 3 1)

Node 1  
Executing: (1 1 4 1)  
Ready: NIL  
Announced: NIL  
Suspended: (1) (4 1)  
Terminated: (1 1 3 1)

node: 2  
contract: 1 4 1  
internal event: node update

: Time: 2509

-- Node Status --
Node 4
Executing: (3 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 1 2 1) (1 2 1 1)

node: 4
contract: 3 1
internal event: node update

node: 4
contract: 3 1
internal event: node update

node: 4
contract: 3 1
internal event: node update

From: 4
Terminated Contract 3 1

: Time: 2512

-- Node Status --

Node 1
Executing: (1 1 4 1)
Ready: (1)
Announced: NIL
Suspended: (1) (4 1)
Terminated: (1 1 3 1)

To: 1
From: 4
Type: final report
Contract: 3 1

: Time: 2810

-- Node Status --

node: 2
contract: 1 1 3 1
internal event: bid check

: Time: 2814

-- Node Status --

Node 1
Executing: (1 1 4 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: (1 1 3 1)

node: 1
contract: 1 1 3 1
internal event: pseudo contract

node: 3
contract: 1 1 3 1
internal event: pseudo contract

node: 5
contract: 1 1 3 1
internal event: pseudo contract

: Time: 2012

-- Node Status --

Node 1
Executing: (1 1 4 1)
Ready: (1)
Announced: NIL
Suspended: (4 1)
Terminated: (1 1 3 1)

node: 1
contract: 1 1 4 1
internal event: node update

node: 1
contract: 1 1 4 1
internal event: node update

From: 1
Terminated Contract 1 1 4 1

: Time: 2913

-- Node Status --
Node 1
Executing: (1)
Ready: NIL
Announced: NIL
Suspended: (4 1)
Terminated: (1 1 4 1) (1 1 3 1)

To: 2
From: 1
Type: final report
Contract: 1 1 4 1

node: 1
contract: 1
internal event: node update

node: 1
contract: 1
internal event: node update

From: 1
Suspended Contract 1

Time: 2914

-- Node Status --

Node 2
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (1 3 1) (1 1) (2 1 1)

node: 2
contract: 1 4 1
internal event: node update

node: 2
contract: 1 4 1
internal event: node update

node: 2
contract: 1 4 1
internal event: node update

From: 2
Terminated Contract 1 4 1

Time: 2915

-- Node Status --

Node 1
Executing: (1 4 1)
Ready: NIL
Announced: NIL
Suspended: (1)
Terminated: (1 1 4 1) (1 1 3 1)

node: 1
contract: 4 1
internal event: node update

node: 1
contract: 4 1
internal event: node update

node: 1
contract: 4 1
internal event: node update

From: 1
Terminated Contract 4 1

Time: 2917

-- Node Status --

To: 1
From: 2
Type: final report
Contract: 4 1

Time: 2918

-- Node Status --

Node 1
Executing: (1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: (4 1) (1 1 4 1) (1 1 3 1)
node: 1
contract: 1
internal event: node update

node: 1
contract: 1
internal event: node update

node: 1
contract: 1
internal event: node update
From: 1
Terminated Contract 1

: Time: 2919

-- Node Status --

To: 0
From: 1
Type: final report
Contract: 1
Solutions Found:
Queen-rows: 3 1 4 2
Queen-rows: 2 4 1 3

: Time: 3004

-- Node Status --

node: 1
contract: 1
internal event: bid check

: Time: 3008

-- Node Status --

node: 2
contract: 1
internal event: pseudo contract

node: 3
contract: 1
internal event: pseudo contract

node: 4
contract: 1
internal event: pseudo contract

node: 5
contract: 1
internal event: pseudo contract

From: 2
Suspended Contract 1

: Time: 3488

-- Node Status --

node: 2
contract: 1
internal event: bid check

: Time: 3412

-- Node Status --

node: 1
contract: 1
internal event: pseudo contract

node: 3
contract: 1
internal event: pseudo contract

node: 4
contract: 1
internal event: pseudo contract

node: 5
contract: 1
internal event: pseudo contract

........................ End of Simulation ........................

Time Units to Completion: 2919

Processor Node Utilization Statistics

<table>
<thead>
<tr>
<th>Node</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.7553957</td>
</tr>
<tr>
<td>2</td>
<td>.7211374</td>
</tr>
<tr>
<td>3</td>
<td>.3429257</td>
</tr>
<tr>
<td>4</td>
<td>.4795589</td>
</tr>
<tr>
<td>5</td>
<td>.3422405</td>
</tr>
</tbody>
</table>

Mean Processor Node Utilization: .5283316
Standard Deviation: .2000483

Another task [ YES ] No
NIL