File created: 28-Mar-89 11:18:05 **{NB:PARC:XEROX}<NOTECARDS>1.3M>LIBRARY>NCPATHPARSE.;1**

previous date: 5-Nov-86 16:38:38 {QV}<NOTECARDS>1.3L>LIBRARY>NCPATH>NCPATHPARSE.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ NCPATHPARSECOMS
       [(* * This file is intended to be the genesis/testbed for ideas about parsing the user language for a
           path-specification faclility into an NCPathFSM for the functions of NCPath to use. The first list of
           functions are the current ones in use. Right now, the system only parses linear specifications.)
        (FNS NCPathParse NCPathParse.Path NCPathParse.PathStep NCPathParse.PathStepDescriptor
             NCPathParse.LiteralPathDescription NCPathParse.FunctionalPathDescription NCPathParse.CreateFSMNode
             NCPathParse.CreatePredicateForm NCPathParse.RepeaterExpression NCPathParse.RepeaterToken
             NCPathParse.LimitedRepeaterToken NCPathParse.LoopDecision NCPathParse.CreateLoop
             NCPathParse.PathStepOperation)
        (FNS NCPathParse.FunctionP NCPathParse.CheckAndComputeFlag NCPathParse.CombinePredicates)
        (FNS NCPathParse.CombinationExpressions)
        (FNS NCPathParse.CoalesceStates NCPathParse.CoalesceRepeaterStates)
        (* * The following functions are utilities from other sources)
        (FNS NAND LOGICAL.EQUAL)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA NAND)
                                                                                (NLAML)
                                                                                (LAMA LOGICAL.EQUAL])
```

(* * This file is intended to be the genesis/testbed for ideas about parsing the user language for a path-specification faclility into an NCPathFSM for the functions of NCPath to use. The first list of functions are the current ones in use. Right now, the system only parses linear specifications.)

```
(DEFINEQ
```

(NCPathParse

```
  [LAMBDA (Expression NoteFile DepthLimit)                    (* Newman " 4-Nov-86 09:49")
```

(* * This function is intended to be the top-level function that parses an expression into an FSM that NCPath.FSM.PathCollect can deal with.)

```
    (if (NCP.OpenNoteFileP NoteFile)
        then [LET [(FirstState (NCPathParse.CoalesceStates (NCPathParse.Path Expression NoteFile]
                  (if FirstState
                      then (create NCPathFSM
                                   InitialState _ FirstState
                                   CurrentState _ FirstState
                                   AbsoluteDepthLimit _ (OR DepthLimit 0]
      else (NCP.ReportError NoteFile " is not an open notefile."])
```

(NCPathParse.Path

```
  [LAMBDA (Expression NoteFile)                               (* Newman " 5-Nov-86 16:21")
```

(* * This function parses an individual path as a list of steps or repeater expressions.)

```
    (if (NULL Expression)
        then NIL
      else (OR (NCPathParse.PathStep Expression NoteFile)
               [if (LISTP Expression)
                   then (for Item in Expression collect (OR (NCPathParse.Path Item NoteFile)
                                                            (RETURN NIL]
               (if (EQUAL 1 (LENGTH Expression))
                   then (NCPathParse.Path (CAR Expression)
                                          NoteFile))
               (if (EQUAL 2 (LENGTH Expression))
                   then (NCPathParse.RepeaterExpression (CADR Expression)
                                                        (MKLIST (NCPathParse.Path (CAR Expression)
                                                                                  NoteFile])
```

(NCPathParse.PathStep

```
  [LAMBDA (Expression NoteFile)                               (* Newman " 4-Nov-86 10:19")
```

(* * This function parses an individual step of a path. A Step is either a path step descriptor or some combination of descriptors.)

```
    (if (NULL Expression)
        then NIL
      else (OR (NCPathParse.PathStepDescriptor Expression NoteFile)
               (if (MEMBER (CAR Expression)
                           '(OR AND NOT))
                   then (NCPathParse.PathStepOperation Expression NoteFile])
```

(NCPathParse.PathStepDescriptor

```
[LAMBDA (Expression NoteFile)                                   (* Newman " 5-Nov-86 16:21")
```

(* * Parses a path step descriptor. A descriptor is a literal descriptor, a functional descriptor, a "don't care" expression, or a variable pointing to a descriptor of one of the other types. The variable is checked for immediate circularity, but not for indirect circularity; either of these conditions could cause an infinite loop.)

```
    (if (NULL Expression)
        then NIL
      else (OR (NCPathParse.LiteralPathDescription Expression NoteFile)
               (NCPathParse.FunctionalPathDescription Expression)
               (if (EQUAL Expression 'ANY)
                   then (NCPathParse.CreateFSMNode (FUNCTION TRUE)
                                T T T))
               (if [AND (BOUNDP Expression)
                        (NOT (EQUAL Expression (EVAL Expression]
                   then (NCPathParse.PathStepDescriptor (EVAL Expression)
                                NoteFile])
```

(NCPathParse.LiteralPathDescription

```
[LAMBDA (Expression NoteFile)                                   (* Newman " 4-Nov-86 13:11")
```

(* * This function parses literal path descriptions. These are path descriptions that are valid link labels or valid card types with appropriate prefixes as described in the grammar.)

```
    (OR (if (NCP.ValidLinkLabel Expression NoteFile)
            then (NCPathParse.CreateFSMNode Expression T T))
        (if (AND (EQUAL (SUBATOM Expression 1 1)
                        '@)
                 (NCP.ValidCardType (SUBATOM Expression 2 -1)))
            then (NCPathParse.CreateFSMNode (SUBATOM Expression 2 -1)
                         NIL T))
        (if (AND (EQUAL (SUBATOM Expression 1 1)
                        '_)
                 (NCP.ValidLinkLabel (SUBATOM Expression 2 -1)
                        NoteFile))
            then (NCPathParse.CreateFSMNode (SUBATOM Expression 2 -1)
                         T NIL))
        (if (AND (MEMBER (SUBATOM Expression 1 2)
                         '(_@
                           @_))
                 (NCP.ValidCardType (SUBATOM Expression 3 -1)))
            then (NCPathParse.CreateFSMNode (SUBATOM Expression 3 -1)
                         T NIL])
```

(NCPathParse.FunctionalPathDescription

```
[LAMBDA (Expression)                                            (* Newman " 5-Nov-86 16:21")
```

(* * This function parses functional path descriptions. These are path descriptions which are the names of Lisp predicates with the appropriate prefixes.)

```
    (OR (if (AND (EQUAL (SUBATOM Expression 1 1)
                        '%#)
                 (NCPathParse.FunctionP (SUBATOM Expression 2 -1)))
            then (NCPathParse.CreateFSMNode (SUBATOM Expression 2 -1)
                         T T T))
        (if (AND (MEMBER (SUBATOM Expression 1 2)
                         '(@# %#@))
                 (NCPathParse.FunctionP (SUBATOM Expression 3 -1)))
            then (NCPathParse.CreateFSMNode (SUBATOM Expression 3 -1)
                         NIL T T))
        (if (AND (MEMBER (SUBATOM Expression 1 2)
                         '(_#
                           %#_))
                 (NCPathParse.FunctionP (SUBATOM Expression 3 -1)))
            then (NCPathParse.CreateFSMNode (SUBATOM Expression 3 -1)
                         NIL T T))
        (if (AND (MEMBER (SUBATOM Expression 1 3)
                         '(_#@
                           _@#
                           @_#
                           @#_ %#@_ %#_@))
                 (NCPathParse.FunctionP (SUBATOM Expression 4 -1)))
            then (NCPathParse.CreateFSMNode (SUBATOM Expression 4 -1)
                         NIL T T])
```

(NCPathParse.CreateFSMNode

```
[LAMBDA (Item Link/CardFlag DirectionFlag FunctionFlag)         (* Newman " 4-Nov-86 11:37")
```

(* * This function creates an instance of the NCPathFSMNode data type according to the arguments passed in.
Its chief purpose is to call NCPathParse.CreatePredicateForm)

```
    (create NCPathFSMNode
           Predicate _ (if FunctionFlag
                           then Item
                         else (NCPathParse.CreatePredicateForm Item Link/CardFlag))
           Card/Link _ Link/CardFlag
           Direction _ DirectionFlag])
```

(NCPathParse.CreatePredicateForm

```
  [LAMBDA (Type Link/CardFLAG)                                    (* Newman " 4-Nov-86 08:52")

          (* * This function creates a LAMBDA expression that will serve as a predicate.
          See PARSE.COMBINE.PREDICATES.)

    `(LAMBDA (Item)
       (EQUAL (%, (COND
                      (Link/CardFLAG (FUNCTION NCP.LinkType))
                      (T 'NCP.CardType))
                  Item)
              (QUOTE %, Type])
```

(NCPathParse.RepeaterExpression

```
  [LAMBDA (RepeaterExpression LoopSteps)                          (* Newman "18-Mar-86 09:21")

          (* * This function parses repeater expressions. Unfortunately, I have not yet determined how to deal with limited repeater
          expressions.)

    (OR (NCPathParse.RepeaterToken RepeaterExpression LoopSteps)
        (NCPathParse.LimitedRepeaterToken RepeaterExpression LoopSteps)
        (NCP.ReportError " Repeater Expression mucked up " RepeaterExpression)
        (BREAK1 NIL T])
```

(NCPathParse.RepeaterToken

```
  [LAMBDA (RepeaterExpression LoopSteps)                          (* Newman "19-Mar-86 15:41")

          (* * creates a repeater loop with no limit, or with infinite limit)

    (if (EQUAL RepeaterExpression '*)
        then (NCPathParse.CreateLoop LoopSteps 0 0)
      elseif (EQUAL RepeaterExpression '+)
        then (NCPathParse.CreateLoop LoopSteps 1 0])
```

(NCPathParse.LimitedRepeaterToken

```
  [LAMBDA (RepeaterExpression LoopSteps)                          (* Newman "18-Mar-86 09:20")

          (* * This function parses repeater tokens that have an integral limit.)

    (OR (NCPathParse.LoopDecision RepeaterExpression LoopSteps 1 '+)
        (NCPathParse.LoopDecision RepeaterExpression LoopSteps 0 '*)
        (NCP.ReportError " Repeater Expression mucked up " RepeaterExpression)
        (BREAK1 NIL T])
```

(NCPathParse.LoopDecision

```
  [LAMBDA (Expression LoopSteps MinimumRepeat Symbol)             (* Newman "18-Mar-86 09:18")

          (* * This function decides what kind of loop is to be created and then creates it.)

    (if (EQUAL Symbol (SUBATOM Expression 1 1))
        then (NCPathParse.CreateLoop LoopSteps (OR MinimumRepeat 0)
                    (OR (NUMBERP (SUBATOM Expression 2 -1))
                        (NCP.ReportError " Repeater Expression mucked up " Expression)))
      elseif (NUMBERP Expression)
        then (NCPathParse.CreateLoop LoopSteps 1 Expression])
```

(NCPathParse.CreateLoop

```
  [LAMBDA (LoopSteps MinTimes MaxTimes)                           (* Newman "15-Mar-86 12:17")

          (* * Here we create a loop expression that PARSE.CoalesceStates understands.)

    (LIST 'PARSE.DO.REPEAT MinTimes MaxTimes LoopSteps])
```

(NCPathParse.PathStepOperation

```
  [LAMBDA (Expression NoteFile)                                   (* Newman " 4-Nov-86 14:42")

          (* * Here we parse combinations of pathsteps.)

    (NCPathParse.CombinationExpressions (for Step in (CDR Expression) collect (OR (NCPathParse.PathStep Step
                                                                                          NoteFile)
                                                                                  (RETURN NIL)))
           (SELECTQ (CAR Expression)
                (AND 'AND)
```

```
                (OR 'OR)
                (NOT 'NAND)
                (NCP.ReportError " Operator not AND, OR, or NOT in NCPathParse.PathStepOperation. "])

)

(DEFINEQ
```

(NCPathParse.FunctionP

```
  [LAMBDA (Function)                                                    (* Newman " 4-Nov-86 11:39")
```

(* * This function determines whether or not a function passed in is an appropriate function for use by NCPath functions. The criterial is rather limited at the moment, including only that the function must be defined, must accept only one argument, and must not be an NLAMBDA function.)

```
    (AND (GETD Function)
         (EQUAL 1 (NARGS Function))
         (NOT (NLAMBDAFNP Function])
```

(NCPathParse.CheckAndComputeFlag

```
  [LAMBDA (StepList FlagName)                                           (* Newman " 5-Nov-86 10:22")
```

(* * This function computes the Direction and Card/Link flags when parsing a combination FSMNode.)

```
    (if [APPLY 'LOGICAL.EQUAL (for Step in StepList collect (EVAL `(fetch (NCPathFSMNode %, FlagName) of Step]
        then [EVAL `(fetch (NCPathFSMNode %, FlagName) of (CAR StepList]
      else (NCP.ReportError " Flags don't all match " StepList)
           'ERROR])
```

(NCPathParse.CombinePredicates

```
  [LAMBDA (StepList Operation)                                          (* Newman "14-Mar-86 16:07")
```

(* * This function builds the LAMBDA expression that will be the predicate in a combination FSMNode. I wish we could compile the LAMBDA expression for speed. Perhaps we could use a GENSYM, and compile the function that way?)

```
    `(LAMBDA (Item)
       ,(CONS Operation (for I in StepList collect (LIST (fetch (NCPathFSMNode Predicate) of I)
                                                         'Item])

)

(DEFINEQ
```

(NCPathParse.CombinationExpressions

```
  [LAMBDA (StepList Operation)                                          (* Newman "24-Mar-86 15:19")
```

(* * This function combines pathstep specifications using AND, OR, or NOT.)

```
    (if (NULL StepList)
        then NIL
      else (LET [(Card/LinkFlag (NCPathParse.CheckAndComputeFlag StepList 'Card/Link))
                 (DirectionFlag (NCPathParse.CheckAndComputeFlag StepList 'Direction]
                (if (OR (EQUAL Card/LinkFlag 'ERROR)
                        (EQUAL DirectionFlag 'ERROR))
                    then NIL
                  else (create NCPathFSMNode
                               Predicate _ (NCPathParse.CombinePredicates StepList Operation)
                               Card/Link _ Card/LinkFlag
                               Direction _ DirectionFlag])

)

(DEFINEQ
```

(NCPathParse.CoalesceStates

```
  [LAMBDA (NodeList)                                                    (* Newman "18-Mar-86 09:28")
```

(* * This function takes a lisp-style list of NCPathFSMNodes and turns them into a linked list. The linked list will include circularities where repeater expressions exist.)

```
    (if (NULL NodeList)
        then NIL
      elseif (EQUAL (TYPENAME NodeList)
                    'NCPathFSMNode)
        then NodeList
      elseif (EQUAL (TYPENAME (CAR NodeList))
                    'NCPathFSMNode)
        then (replace (NCPathFSMNode NextNodes) of (CAR NodeList) with (NCPathParse.CoalesceStates (CDR NodeList)))
             (CAR NodeList)
      elseif (AND (LISTP NodeList)
                  (EQUAL (CAR NodeList)
                         'PARSE.DO.REPEAT))
        then (NCPathParse.CoalesceRepeaterStates (MKLIST (CADDDR NodeList))
                     NIL
```

```
                    (CADDR NodeList))
      elseif (AND (LISTP (CAR NodeList))
                  (EQUAL (CAAR NodeList)
                         'PARSE.DO.REPEAT))
        then (LET ((Rest (NCPathParse.CoalesceStates (CDR NodeList)))
                   (Expression (CAR NodeList)))
                  (if (AND (ZEROP (CADR Expression))
                           Rest)
                      then (LIST Rest (NCPathParse.CoalesceRepeaterStates (MKLIST (CADDDR Expression))
                                                                          Rest
                                                                          (CADDR Expression)))
                    else (NCPathParse.CoalesceRepeaterStates (MKLIST (CADDDR Expression))
                                                             Rest
                                                             (CADDR Expression])
```

(NCPathParse.CoalesceRepeaterStates

```
  [LAMBDA (LoopList Next Limit)                                    (* Newman "19-Mar-86 15:39")

          (* * This function creates the circular linked list structure that repeater expressions need.)

    (LET [(FirstNode (CAR LoopList))
          (LastNode (CAR (LAST LoopList]
         (NCPathParse.CoalesceStates LoopList)
         (replace (NCPathFSMNode LoopLimit) of FirstNode with (OR Limit 0))
         [replace (NCPathFSMNode NextNodes) of LastNode with (CONS Next (CONS FirstNode (fetch (NCPathFSMNode
                                                                                                  NextNodes)
                                                                                           of LastNode]
         FirstNode])

)


          (* * The following functions are utilities from other sources)

(DEFINEQ
```

(NAND

```
  [NLAMBDA Args                                                    (* Newman " 4-Nov-86 11:40")

          (* * This function is the logical function NOT AND.)

    (NOT (APPLY 'AND Args])
```

(LOGICAL.EQUAL

```
  [LAMBDA ARGS                                                     (* Newman " 8-Nov-84 09:42")

          (* * This function is a logical operator. It determines if the arbitrary number of arguments are logically equal or not.
          --DVN)

    (OR (for I from 1 to ARGS always (ARG ARGS I))
        (for I from 1 to ARGS never (ARG ARGS I])

)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA NAND)

(ADDTOVAR NLAML )

(ADDTOVAR LAMA LOGICAL.EQUAL)
)

(PUTPROPS NCPATHPARSE COPYRIGHT ("Xerox Corporation" 1986 1989))
```

FUNCTION INDEX

VARIABLE INDEX