```
(FILECREATED "26-Sep-81 16:54:35" <AFFIRM>PARSERHELPER..14 18997

     changes to:  MakeAccessExp reduceExpression

     previous date: "29-Jun-81 17:04:25" <AFFIRM>PARSERHELPER..13)


(PRETTYCOMPRINT PARSERHELPERCOMS)

(RPAQQ PARSERHELPERCOMS [(* Auxiliary functions, records and properties for PARSER)
        (FNS * PARSERHELPERFNS)
        (VARS bracketListOp InternalInfixOps IMPORTOP)
        (PROP priority PLUS AND DIFFERENCE DIV EQ EQV GE GT IMP LE LT MOD TIMES NE OR EXPT)
        (BLOCKS (PARSERHELPERBLOCK MakeAccessExp UsePriorities UsePrioritiesHelper
                                   assocAllProgramUnits definePriority labelAssertionPairs
                                   lowerPriority reduceExpression reduceInfixExp reduceParseTree
                                   reduceProgramUnit reduceStatement renameInfixOp renamePrefixOp
                                   varFormalParameters (ENTRIES reduceExpression reduceParseTree)
                                   (GLOBALVARS EXPTOP LabelAssertionPairs LastProgramUnits
                                               ProgramUnits bracketListOp)
                                   (NOLINKFNS . T])
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* Auxiliary functions, records and properties for PARSER)]

```
(RPAQQ PARSERHELPERFNS (MakeAccessExp UsePriorities UsePrioritiesHelper assocAllProgramUnits
                                      definePriority formalParameters labelAssertionPairs
                                      lowerPriority reduceDeclareopt reduceExpression
                                      reduceFormalParameterSection reduceInfixExp reduceParseTree
                                      reduceProgramUnit reduceSimpleType reduceStatement reduceType
                                      reduceUnpackedStructuredType renameInfixOp renamePrefixOp
                                      varFormalParameters))
(DEFINEQ

                                                                                                    1

(MakeAccessExp
  [LAMBDA (x)                                                    (* R.Erickson "18-Jun-81 19:22")
                                                                 (* took out the qualifier stuff
                                                                 (arrays, records, heapsortiles))

    x:identifier])

                                                                                                    2

(UsePriorities
  [LAMBDA (x)
    (if x:expression
        then x:expression←(UsePriorities x:expression)
             (UsePrioritiesHelper x)
      else x])

                                                                                                    3

(UsePrioritiesHelper
  [LAMBDA (x)
    (if (LISTP x:infixOp)
        then (x:infixOp←(renameInfixOp x:infixOp)))
    (if x:expression:infixOp=NIL
        then x
      else (if (LISTP x:expression:infixOp)
               then (x:expression:infixOp←(renameInfixOp x:expression:infixOp)))
           (if (lowerPriority x:infixOp x:expression:infixOp)
               then x
             else (create expression
                          primary ←(UsePrioritiesHelper (create expression
                                                                expression ← x:expression:primary
                                                                using x))
                   using x:expression])

                                                                                                    4

(assocAllProgramUnits
  [LAMBDA (Key)                                                  (* R.Erickson "25-Feb-81 16:59")
    (FASSOC Key LastProgramUnits])

                                                                                                    5

(definePriority
  [LAMBDA (op left right)
    (if (ATOM op)
        then (op:priority←(create priorityRecord
                                  left ← left
                                  right ←(if right=NIL
                                             then left
                                           else right)))
      else (for x in op collect (definePriority x left right])

                                                                                                    6

(formalParameters
  [LAMBDA (fpSection)
    (for s in fpSection join (COPY s:parameterGroup:identifier])

                                                                                                    7

(labelAssertionPairs
```

```
[LAMBDA (stmt)
  (SELECTQ stmt:SYNTACTICTYPE
           (caseStatement < !! (for s in stmt:caseElementList join (labelAssertionPairs s:statement)
                                                                    )
                          !
                          (labelAssertionPairs stmt:statement)
                          >)
           ((compoundStatement repeatStatement)
             (for s in stmt:statement join (labelAssertionPairs s)))
           ((forStatement whileStatement withStatement)
             (labelAssertionPairs stmt:statement))
           (ifStatement < !! (labelAssertionPairs stmt:statement)
                        !
                        (labelAssertionPairs stmt:statement#)
                        >)
           (labelStatement <<stmt:label ! (reduceExpression
                                                  stmt:simpleStatement:assertion:expression)
                           >>)
           NIL])
```

8

(lowerPriority

```
[LAMBDA (leftop rightop)

  leftop←leftop:priority:right
  rightop←rightop:priority:left
  (if leftop=NIL
      then NIL
    elseif rightop=NIL
      then T
    else leftop lt rightop])
```

(if no priority is defined for an operator it is considered to have highest possible priority)*

9

(reduceDeclareopt

```
[LAMBDA (xlst)

  (for (x temp) in xlst join (temp←x:declareType)
                             (if temp:constDefinition
                                 then (for y in temp:constDefinition
                                         do (y:expression←(reduceExpression y:expression)))
                                      temp:constDefinition
                               elseif temp:typeDefinition
                                 then (for y in temp:typeDefinition
                                         do (y:type←(reduceType y:type)))
                                      temp:typeDefinition
                               elseif temp:varDeclaration
                                 then [for y in temp:varDeclaration
                                         do (y:type←(reduceType y:type))
                                            (for z in y:varDeclarePart
                                               do (z:expression←(reduceExpression z:expression))
                                                  (z:expression#←(reduceExpression z:expression#]
                                      temp:varDeclaration
                               elseif temp:label
                               else <temp>])
```

(Edited by R.Bates on 14-APR-78; from version 6)*

10

(reduceExpression

```
[LAMBDA (x)
  (SELECTQ x:SYNTACTICTYPE
           (bracketExprList <bracketListOp ! (for y in x:expression collect
                                                        (reduceExpression y))
                            >)
           (coord (if x:number
                    elseif x:number#
                      then (-x:number#)
                    elseif x:all
                      then 'ALL
                    elseif x:firstOne
                      then 'FIRST
                    elseif x:lastOne
                      then 'LAST
                    else x))
           (denoteSpec x:denotePair← (for i in x:denotePair collect (reduceExpression i))
                       x)
           (denotePair x:expression← (reduceExpression x:expression)
```

(R.Erickson "23-Sep-81 17:31")*

```
            x)
(expression (if x:infixOp
                then (reduceInfixExp (UsePriorities x))
              else (reduceExpression x:primary:ALTERNATIVESUBNODE)))
(expressionSeq x:expression← (for y in x:expression collect (reduceExpression y))
               x)
(functionDecl x:expression← (for y in x:expression collect (reduceExpression y))
              x:expression#←
              (reduceExpression x:expression#)
              x)
(identifierSeq x:identifier← (for i in x:identifier collect (reduceExpression i))
               x)
(ifExpr <IFOP (reduceExpression x:expression)
        (reduceExpression x:expression#)
        (if x:expression##
            then (reduceExpression x:expression##)
          else TRUE)
        >)
(interfaceList x:op← (for y in x:op collect (reduceExpression y))
               x)
(nochangeSpec x:expression← (reduceExpression x:expression)
              x)
[(op opOrExpression)
  (PROG (y)
        (RETURN (SELECTQ (fetch SYNTACTICTYPE of y←x:ALTERNATIVESUBNODE)
                         (expression (reduceExpression y))
                         (infixOp (renameInfixOp y))
                         (if (MapToInternal y T)
                           else y]
(parenExpr (reduceExpression x:expression))
[prefixExpr (PROG (ex imports fundef)
                  (ex← <(renamePrefixOp x:prefixOp)
                       !(for y in x:expression collect (reduceExpression y))
                       >)
                  (if (NUMBERP ex:1)
                      then (if ex:2~=8
                               then (ERROR "Don't know how to reduce" ex))
                           (RETURN (PACK < !(UNPACK ex:1) ! '(Q)
                                         >)))
                  [imports←(if x:identifier
                               else fundef←(assocAllProgramUnits x:prefixOp)
                                    (if fundef
                                        then (formalParameters
                                                  fundef::1:formalParameterSection#]
                  (RETURN (if imports
                              then < !! ex ! imports>
                            else ex]
(primary (reduceExpression x:ALTERNATIVESUBNODE))
(quantifiedExpression (PROG (quan exp)
                            (quan←x:quantifier:LEXEME)
                            (exp←(reduceExpression x:expression))
                            (for i in (REVERSE x:identifier) do exp← <quan i exp>)
                            (RETURN exp)))
(rangedInterfaceList x:rangedOp← (for y in x:rangedOp
                                    collect <(y:op←(reduceExpression y:op))
                                             (reduceExpression y:rangeSpec)
                                             >)
                     x)
(rangeSpec (for y in x:range collect <(y:coord←(reduceExpression y:coord))
                                      (y:coord#←(reduceExpression y:coord#))
                                      >))
(ruleSeq x:rule← (for y in x:rule collect (y:expression←(reduceExpression y:expression))
                                          (y:expression#←(reduceExpression y:expression#)
                                                         )
                                          y)
         x)
(specialPrefixExpr < (renamePrefixOp x:specialPrefixOp)
                   (reduceExpression x:primary:ALTERNATIVESUBNODE)
                   >)
(variable x:identifier)
(variableDecl x:expression← (reduceExpression x:expression)
              x)
x])
```


(reduceFormalParameterSection

```
  [LAMBDA (xlst)

    (for x in xlst bind temp do (temp←x:parameterKind)
```

(Edited by R.Bates on 28-MAR-78; no file)*

```
                    (if temp
                        then (x:parameterKind←temp:LEXEME))
                    (x:parameterGroup:type←(reduceType x:parameterGroup:type])
```

12

```
(reduceInfixExp
  [LAMBDA (x)                                    (* x can be either a primary or an expression node)
    (if x:SYNTACTICTYPE='primary
        then (reduceExpression x:ALTERNATIVESUBNODE)
      elseif x:infixOp
        then <x:infixOp (reduceInfixExp x:primary)
             (reduceInfixExp x:expression) >
      else (reduceInfixExp x:primary])
```

13

```
(reduceParseTree
  [LAMBDA (tree)                                 (* Edited by R.Bates on 12-APR-78;
                                                 no file)
                                                 (* Global variables (used by reduceStatement):
                                                 LastProgramUnits, ExitAssertion)
    (reduceProgramUnit tree)
    LastProgramUnits←(GETDEFINITIONS tree)
    (for pu in LastProgramUnits bind blk do (blk←pu::1:block)
                                            (blk:assertion←(reduceExpression blk:assertion:expression)
                                              )
                                            (ExitAssertion←(blk:assertion#←(reduceExpression
                                                                    blk:assertion#:expression)))
                                            (blk:declareopt←(reduceDeclareopt blk:declareopt))
                                            (LabelAssertionPairs←(labelAssertionPairs
                                                                    blk:compoundStatement))
                                            (reduceStatement blk:compoundStatement))
    tree])
```

14

```
(reduceProgramUnit
  [LAMBDA (x)                                    (* Edited by R.Bates on 12-APR-78;
                                                 from version 4)
    (SELECTQ x:SYNTACTICTYPE
            (procedureOrFunctionDeclaration x:identifier##← (UNION
                                              < !!(varFormalParameters x:formalParameterSection)
                                                !(varFormalParameters x:formalParameterSection#) >
                                              x:identifier##)
                                                    (* Do not call reduceFormalParamterSection before
                                                    varFormalParameters.)
                                             (reduceFormalParameterSection x:formalParameterSection)
                                             (reduceFormalParameterSection x:formalParameterSection#)
                                             x:type←
                                             (reduceType x:type)
                                             (for y in x:block:declareopt
                                                when y:declareType:procedureOrFunctionDeclaration
                                                do (reduceProgramUnit
                                                     y:declareType:procedureOrFunctionDeclaration)))
            [program (reduceProgramUnit (if x:procedureOrFunctionDeclaration
                                         else (x:procedureOrFunctionDeclaration←(create
                                                 procedureOrFunctionDeclaration
                                                 unitKind ←(create unitKind
                                                                  LEXEME ←('PROGRAM))
                                                 identifier ←('MAIN)
                                                 block ← x:block]
            x)
    x])
```

15

```
(reduceSimpleType
  [LAMBDA (x)                                    (* Edited by R.Bates on 31-MAR-78;
                                                 from version 2)
    (SELECTQ x:SYNTACTICSUBTYPE
            (scalarType x)
            (typeIdentifier x:ALTERNATIVESUBNODE:identifier)
            (subrangeType x:ALTERNATIVESUBNODE:expression← (reduceExpression
                                                              x:ALTERNATIVESUBNODE:expression)
                          x:ALTERNATIVESUBNODE:expression#←
                          (reduceExpression x:ALTERNATIVESUBNODE:expression#)
```

```
                    x)
          x])
```

16

```
(reduceStatement
  [LAMBDA (x)                                          (* R Bates "14-Dec-79 14:43")
    (SELECTQ x:SYNTACTICTYPE
          ((assertStatement assumeStatement proveStatement)
            x:assertion←
            (reduceExpression x:assertion:expression))
          (assignmentStatement x:variable← (MakeAccessExp x:variable)
                               x:expression←
                               (reduceExpression x:expression))
          (caseStatement x:expression← (reduceExpression x:expression)
                         (for y in x:caseElementList
                            do (y:caseLabel←(for z in y:caseLabel collect z:constant:LEXEME))
                               (reduceStatement y:statement))
                         (reduceStatement x:statement))
          (compoundStatement (for y in x:statement do (reduceStatement y)))
          (concurrentAssignmentStatement x:variable← (for y in x:variable
                                                        collect (MakeAccessExp y))
                                         x:expression←
                                         (for y in x:expression collect (reduceExpression y)))
          (forStatement x:assertion← (reduceExpression x:assertion:expression)
                        x:expression←
                        (reduceExpression x:expression)
                        x:direction←x:direction:LEXEME x:expression#← (reduceExpression
                                                                          x:expression#)
                        (reduceStatement x:statement)
                        x:assertion#←
                        (reduceExpression x:assertion#:expression))
          (goToStatement x:assertion← (if x:assertion
                                          then (reduceExpression x:assertion:expression)
                                        else (FASSOC x:label labelAssertionPairs)::1))
          (ifStatement x:expression← (reduceExpression x:expression)
                       (reduceStatement x:statement)
                       (reduceStatement x:statement#))
          (labelStatement (reduceStatement x:simpleStatement))
          [procedureStatement (PROG (pu formals imports)
                                    (pu←(assocAllProgramUnits x:identifier)::1)
                                    (formals←(formalParameters pu:formalParameterSection))
                                    (imports←(formalParameters pu:formalParameterSection#))
                                    (x:expression←(for y in x:expression
                                                     collect (reduceExpression y)))
                                    (if x:identifier#=NIL
                                        then (x:identifier#←imports))
                                    (x:variable←(if x:variable=NIL
                                                    then (for a in < ! x:expression
                                                                   ! x:identifier#>
                                                            as f in < !! formals ! imports>
                                                            when (MEMBER f pu:identifier##)
                                                            collect a)
                                                  else (for y in x:variable
                                                          collect (MakeAccessExp y]
          (repeatStatement (for y in x:statement do (reduceStatement y))
                           x:expression←
                           (reduceExpression x:expression))
          (returnStatement x:expression← (reduceExpression x:expression)
                           x:assertion←
                           (if x:assertion
                               then (reduceExpression x:assertion:expression)
                             else ExitAssertion))
          (whileStatement x:assertion← (reduceExpression x:assertion:expression)
                          x:expression←
                          (reduceExpression x:expression)
                          (reduceStatement x:statement)
                          x:assertion#←
                          (reduceExpression x:assertion#:expression))
          x])
```

17

```
(reduceType
  [LAMBDA (x)                                          (* Edited by R.Bates on 31-MAR-78;
                                                          from version 2)
    (SELECTQ x:SYNTACTICSUBTYPE
          (simpleType x:ALTERNATIVESUBNODE← (reduceSimpleType x:ALTERNATIVESUBNODE)
                      x)
```

```
        (structuredType x:ALTERNATIVESUBNODE:unpackedStructuredType←
                        (reduceUnpackedStructuredType
                                              x:ALTERNATIVESUBNODE:unpackedStructuredType)
                        x)
        (pointerType x)
        x])
```

18

```
(reduceUnpackedStructuredType
  [LAMBDA (x)                                          (* Edited by R.Bates on 31-MAR-78;
                                                       from version 2)
    (SELECTQ x:SYNTACTICSUBTYPE
        (arrayType x:ALTERNATIVESUBNODE:type← (reduceType x:ALTERNATIVESUBNODE:type)
                   x:ALTERNATIVESUBNODE:simpleType←
                   (for y in x:ALTERNATIVESUBNODE:simpleType collect (reduceSimpleType y))
                   x:ALTERNATIVESUBNODE)
        x:ALTERNATIVESUBNODE])
```

19

```
(renameInfixOp
  [LAMBDA (x)                                          (* D.Thompson "26-Sep-79 15:20")
    (if (ATOM x:ALTERNATIVESUBNODE)
        then (if (MapToInternal x:ALTERNATIVESUBNODE)
                 else x:ALTERNATIVESUBNODE)
      elseif (MapToInternal x:SYNTACTICSUBTYPE)
      else x])
```

20

```
(renamePrefixOp
  [LAMBDA (x)                                          (* D.Thompson "26-Sep-79 15:17")
    (if (MapToInternal x T)
      else x])
```

21

```
(varFormalParameters
  [LAMBDA (fpSection)
    (for s in fpSection when s:parameterKind:LEXEME='VAR join (COPY s:parameterGroup:identifier])
)

(RPAQQ bracketListOp Bracket)

(RPAQQ InternalInfixOps (ADDOP ANDOP DIFFOP DIVOP EQOP EQVOP GEOP GTOP IMPOP LEOP LTOP MODOP MULTOP
                               NEOP OROP PWROP))

(RPAQQ IMPORTOP IMPORTS)

(PUTPROPS PLUS priority (25 . 25))

(PUTPROPS AND priority (11 . 10))

(PUTPROPS DIFFERENCE priority (25 . 25))

(PUTPROPS DIV priority (30 . 30))

(PUTPROPS EQ priority (20 . 20))

(PUTPROPS EQV priority (1 . 1))

(PUTPROPS GE priority (20 . 20))

(PUTPROPS GT priority (20 . 20))

(PUTPROPS IMP priority (3 . 3))

(PUTPROPS LE priority (20 . 20))

(PUTPROPS LT priority (20 . 20))

(PUTPROPS MOD priority (30 . 30))

(PUTPROPS TIMES priority (30 . 30))

(PUTPROPS NE priority (20 . 20))
```

```
(PUTPROPS ORpriority (6 . 5))

(PUTPROPS EXPTpriority (36 . 35))
[DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY
(BLOCK: PARSERHELPERBLOCK MakeAccessExp UsePriorities UsePrioritiesHelper assocAllProgramUnits
        definePriority labelAssertionPairs lowerPriority reduceExpression reduceInfixExp
        reduceParseTree reduceProgramUnit reduceStatement renameInfixOp renamePrefixOp
        varFormalParameters (ENTRIES reduceExpression reduceParseTree)
        (GLOBALVARS EXPTOP LabelAssertionPairs LastProgramUnits ProgramUnits bracketListOp)
        (NOLINKFNS . T))
]
(DECLARE: DONTCOPY
  (FILEMAP (NIL (1553 17628 (MakeAccessExp 1565 . 1848) (UsePriorities 1852 . 2023) (UsePrioritiesHelper
 2027 . 2625) (assocAllProgramUnits 2629 . 2788) (definePriority 2792 . 3099) (formalParameters 3103 .
 3225) (labelAssertionPairs 3229 . 3997) (lowerPriority 4001 . 4389) (reduceDeclareopt 4393 . 5365) (
reduceExpression 5369 . 9240) (reduceFormalParameterSection 9244 . 9609) (reduceInfixExp 9613 . 10016)
 (reduceParseTree 10020 . 10901) (reduceProgramUnit 10905 . 12210) (reduceSimpleType 12214 . 12724) (
reduceStatement 12728 . 15987) (reduceType 15991 . 16480) (reduceUnpackedStructuredType 16484 . 16955)
 (renameInfixOp 16959 . 17285) (renamePrefixOp 17289 . 17459) (varFormalParameters 17463 . 17625)))))
STOP
```