File created: 14-Sep-2022 10:25:44 **{DSK}<users>kaplan>local>medley3.5>working-medley>sources>DWIMIFY.;2**

changes to: (FNS CLISPFOR0)

previous date: 16-May-90 16:21:27 {DSK}<users>kaplan>local>medley3.5>working-medley>sources>DWIMIFY.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS


```
(RPAQQ DWIMIFYCOMS
       ((FNS DWIMIFYFNS DWIMIFY DWIMIFY0 DWIMIFY0? DWMFY0 DWIMIFY1 DWIMIFY1? DWMFY1 DWIMIFY1A DWIMIFY2 DWIMIFY2?
             DWMFY2 DWIMIFY2A CLISPANGLEBRACKETS SHRIEKER CLISPRESPELL EXPRCHECK)
        (FNS CLISPATOM0 CLISPATOM1 CLRPLNODE STOPSCAN? CLUNARYMINUS? CLBINARYMINUS? CLISPATOM1A CLISPATOM1B
             CLISPATOM2 CLISPNOEVAL CLISPLOOKUP CLISPATOM2A CLISPBROADSCOPE CLISPBROADSCOPE1 CLISPATOM2C
             CLISPATOM2D CLISPCAR/CDR CLISPCAR/CDR1 CLISPCAR/CDR2 CLISPATOMIS1 CLISPATOMARE1 CLISPATOMARE2
             CLISPATOMIS2)
        (FNS WTFIX WTFIX0 WTFIX1 RETDWIM DWIMERRORRETURN DWIMARKASCHANGED RETDWIM1 FIX89TYPEIN FIXLAMBDA
             FIXAPPLY FIXATOM FIXATOM1 FIXCONTINUE FIXCONTINUE1 CLISPATOM GETVARS GETVARS1 FIX89 FIXPRINTIN
             FIX89A CLISPFUNCTION? CLISPNOTVARP CLISP-SIMPLE-FUNCTION-P CLISPELL FINDFN DWIMUNSAVEDEF CHECKTRAN)
        (FNS CLISPIF CLISPIF0 CLISPIF1 CLISPIF2 CLISPIF3)
        (FNS CLISPFOR CLISPFOR0 CLISPFOR0A CLISPFOR1 CLISPRPLNODE CLISPFOR2 CLISPFOR3 CLISPFORVARS CLISPFORVARS1
             CLISPFOR4 CLISPFORF/L CLISPDSUBST GETDUMMYVAR CLISPFORINITVAR)
        (COMS (FNS \DURATIONTRAN \CLISPKEYWORDPROCESS))
        (DECLARE%: EVAL@COMPILE DONTCOPY (MACROS DWIMUNDOCATCH))
        (BLOCKS (FORBLOCK (ENTRIES CLISPFOR)
                          CLISPFORVARS CLISPFOR0 CLISPFOR2 CLISPFORINITVAR CLISPDSUBST \CLISPKEYWORDPROCESS
                          CLISPFORF/L CLISPFOR4 CLISPFORVARS1 CLISPFOR3 CLISPFOR1 CLISPFOR0A CLISPFOR \DURATIONTRAN
                          (SPECVARS UNDOSIDE LISPXHIST BODY I.S.TYPE1 I.S.TYPE TERMINATEFLG FIRSTI.V. I.V. PROGVARS
                                 MAKEPROGFLG IVINITFLG INITVARS UNDOLST DWIMIFYING VARS DWIMIFYCHANGE DUMMYVARS
                                 I.S.OPRSLST CLISPCONTEXT UNDOSIDE0 EXP))
                (DWIMIFYBLOCK CLBINARYMINUS? CLISPANGLEBRACKETS CLISPATOM CLISPATOM0 CLISPATOM1 CLISPATOM1A
                       CLISPATOM1B CLISPATOM2 CLISPATOM2A CLISPATOM2C CLISPATOM2D CLISPATOMARE1 CLISPATOMARE2
                       CLISPATOMIS1 CLISPATOMIS2 CLISPBROADSCOPE CLISPBROADSCOPE1 CLISPCAR/CDR CLISPCAR/CDR1
                       CLISPCAR/CDR2 CLISPIF CLISPIF0 CLISPIF1 CLISPIF2 CLISPIF3 CLISPLOOKUP CLISPRESPELL
                       CLRPLNODE CLUNARYMINUS? DWIMIFY DWIMIFY0 DWIMIFY0? DWIMIFY1 DWIMIFY1? DWIMIFY1A DWIMIFY2
                       DWIMIFY2? DWIMIFY2A DWIMIFYFNS DWMFY0 DWMFY1 DWMFY2 FIX89 FIX89A FIX89TYPEIN FIXAPPLY
                       FIXATOM FIXATOM1 FIXCONTINUE FIXCONTINUE1 FIXLAMBDA GETDUMMYVAR GETVARS GETVARS1 RETDWIM
                       RETDWIM1 SHRIEKER STOPSCAN? WTFIX WTFIX0 WTFIX1
                       (ENTRIES WTFIX WTFIX1 DWIMIFYFNS DWIMIFY DWIMIFY0 DWIMIFY0? DWIMIFY1A GETDUMMYVAR DWIMIFY2
                              DWIMIFY2? DWIMIFY1? DWIMIFY1 DWIMIFY2A CLISPLOOKUP)
                       (SPECVARS 89CHANGE 89FLG BRACKET BRACKETCNT ATTEMPTFLG BACKUPFLG BODY BREAKFLG BROADSCOPE
                              CLISPCHANGE CLISPCHANGES CLISPCONTEXT CLISPERTYPE CLTYP CURRTAIL DWIMIFYCHANGE
                              DWIMIFY0CHANGE DWIMIFYFLG DWIMIFYING ENDTAIL EXP EXPR FAULTAPPLYFLG FAULTARGS
                              FAULTFN FAULTPOS FAULTX FAULTXX FIRSTI.V. FIXCLK FORMSFLG I.S.TYPE I.S.TYPE1
                              HISTENTRY I.S. I.V. INITVARS IVINITFLG LISPFN CHARLST MAKEPROGFLG NCONC1LKUP
                              NCONCLKUP NEGFLG NEWTAIL NEXTAIL SUBPARENT NOFIX89 NOSAVEFLG ONEFLG ONLYSPELLFLG
                              PARENT SIDES TAIL TENTATIVE TERMINATEFLG TYP TYPE-IN? UNDOLST UNDOSIDE UNDOSIDE0
                              VAR1 VAR2 VARS WORKFLAG UNARYFLG DEST FOR I.S.OPRSLST PROGVARS)))
        (GLOBALVARS DWIMINMACROSFLG CHECKCARATOMFLG TREATASCLISPFLG CLISPHELPFLG CLISPIFTRANFLG CLISPRETRANFLG
               DWIMCHECKPROGLABELSFLG DWIMCHECK#ARGSFLG SHALLOWFLG PRETTYTRANFLG CLEARSTKLST LCASEFLG
               LAMBDASPLST DURATIONCLISPWORDS CLISPTRANFLG CLISPIFWORDSPLST LPARKEY DWIMUSERFORMS DWIMKEYLST
               SPELLINGS3 SPELLINGS1 CLISPARRAY CLISPFLG CLISPCHARS CLISPISNOISEWORDS CLISPLASTSUB
               CLISPISWORDSPLST CLISPCHARRAY CLISPINFIXSPLST OKREEVALST WTFIXCHCONLST1 WTFIXCHCONLST RPARKEY
               NOFIXFNSLST0 NOFIXVARSLST0 LISPXHISTORY DWIMEQUIVLST COMMENTFLG USERWORDS SPELLINGS2 FILELST
               CLISPFORWORDSPLST CLISPDUMMYFORVARS LASTWORD COMPILERMACROPROPS)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY (ADDVARS (NLAML BREAK1)))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA DWIMIFYFNS)
                                                                              (NLAML)
                                                                              (LAMA)))
        (INITVARS (DWIM.GIVE.UP.TIME)
               (DWIM.GIVE.UP.INTERVAL 2000))))

(DEFINEQ

(DWIMIFYFNS
  [NLAMBDA FNS                                          (* lmm "20-May-84 19:57")
    (PROG ((CLK (CLOCK 0))
           TEM)
          (SETQ NOFIXFNSLST0 NOFIXFNSLST)
          (SETQ NOFIXVARSLST0 NOFIXVARSLST)
          [SETQ TEM (MAPCAR [COND
                                ((CDR FNS)
                                 FNS)
                                ((LISTP (CAR FNS))
                                 (STKEVAL 'DWIMIFYFNS (CAR FNS)
```

```
                                    NIL
                                    'INTERNAL))
                              (T                                    ; If (CAR FNS) is name of a file, do dwimifyfns on its functions.
                                 (OR (LISTP (EVALV (CAR FNS)
                                                   'DWIMIFYFNS))
                                     (AND (GETPROP (OR (AND DWIMFLG (MISSPELLED? (CAR FNS)
                                                                                 70 FILELST NIL FNS))
                                                       (CAR FNS))
                                                   'FILE)
                                          (FILEFNSLST (CAR FNS)))
                                     (STKEVAL 'DWIMIFYFNS (CAR FNS)
                                              'INTERNAL]
                           (FUNCTION (LAMBDA (X)
                                     (DWIMIFY0 X]
        (RETURN TEM])
```

(DWIMIFY

```
[LAMBDA (X QUIETFLG L)                                  (* lmm "20-May-84 19:57")
  (PROG (VAL)
        (COND
           ((NULL DWIMFLG)
            (LISPXPRIN1 "DWIM is turned off!
                   " T)
            (RETURN NIL)))
```

;; If X is an atom and L is NIL, X is treated as the name of a function, and its entire definition is DWIMIFIED. Otherwise, X is a piece of a function, ;; and L the edit puh down list that leads to X (i.e. L is the push-dwown list after performing a !0) L is used to compute the bound variables, as well ;; as to determine whether X is an element or tail.

```
        (SETQ NOFIXFNSLST0 NOFIXFNSLST)
        (SETQ NOFIXVARSLST0 NOFIXVARSLST)
        (SETQ VAL (DWIMIFY0 X L))
        (COND
           ((AND (LISTP X)
                 (NULL L)
                 (NULL QUIETFLG))
            (RESETFORM (OUTPUT T)
                   (RESETVARS ((PRETTYTRANFLG T))
                              (PRINTDEF VAL NIL T)))
            (TERPRI T)))
        (RETURN VAL])
```

(DWIMIFY0

```
[LAMBDA (X Y VARS EXPR)                                 (* lmm "27-FEB-83 10:55")
```

;; Some general comments: --- DWIMIFYFLG is bound in DWIMIFY0, WTFIX, and WTFIX0. It is set to T whenever WTFIX is called and given ;; EXPR, TAIL, PARENT, etc. as arguments, i.e. from DWIMIFY1 or DWIMIFY2. Note that this may occur due to an explicit call to DWIMIFY0, or ;; due to evaluating certain CLISP expressions, e.g. IF statements, which call DWIMIFY1 or DWIMIFY2. These two cases are distinguished by the ;; value of DWIMIFYING. --- DWIMIFYING is bound in DWIMIFY0 (to T), and whenever DWIMIFY1 or DWIMIFY2 are called from contexts where ;; DWIMIFYING may not be bound, e.g. from CLISPIF. In these latter cases, DWIMIFYING is bound to (AND DWIMIFYFLG DWIMIFYING). Thus ;; DWIMIFYING is always bound when DWIMIFYFLG is bound, and is T when under a call to DWIMIFY0, otherwise NIL. Note that checking ;; DWIMIFYING without also checking DWIMIFYFLG may cause a U.B.A. DWIMIFYING error. Similary, other state variables that are bound in ;; DWIMIFY0 but not rebound by DWIMIFY1 or DWIMIFY2 such as CLISPCONTEXT, DWIMIFYCHANGE, etc., are assumed to be bound when ;; DWIMIFYFLG is T, so that any call to DWIMIFY1 or DWIMIFY2 must also guarantee that these variables are bound. If the caller is not sure, it ;; should use DWIMIFY1? and DWIMIFY2? since these do the appropriate checks. --- NOFIXFNSLST0 and NOFIXVARSLST0 are global ;; varaales. They are initializaed to NOFIXFNSLST and NOFIXVARLST by DWIMIFY and DWIMIFYFNS, as well as CLISPIF, CLISPFOR, etc. ;; when they enter the DWIMIFY functions, i.e. DWIMIFY1 and DWIMIFY2 for the first time. NOFIXFNSLST and NOFIXVARLST are the variable ;; that the user can add things to. --- VARS is bound in WTFIX and in DWIMIFY0. DWIMIFY1 and DWIMIFY2 supply VARS in their call to WTFIX. ;; Otherwise WTFIX comptes them. --- ATTEMPTFLG is bound in DWIMIFY1 and DWIMIFY2. It is used to inform DWIMIFY1 or DWIMIFY2, in the ;; event that WTFIX was unable to make a correction, NOT to add the atom to NOFIXLST. For example, this occurs when a correction was offered ;; to the user but rejected, e.g. U.D.F. T, and user declines the fix, T is not added to NOFIXLST.

```
  (PROG (FN FAULTFN DWIMIFY0CHANGE DWIMIFYCHANGE TEM CLISPCONTEXT ONEFLG (DWIMIFYING T)
            (DWIMIFYFLG T)
            [SIDES (CDR (LISTGET1 LISPXHIST 'SIDE]
            TYPE-IN?
            (FIXSPELLDEFAULT 'n))
        (RETURN (COND
                   [(LISTP Y)                                   ; from DW command
                    [COND
                       ([LISTP (SETQ FAULTFN (EVALV 'ATM]
                        (SETQ FAULTFN (CAR FAULTFN]             ; ATM is bound in EDITE.
                    (SETQ VARS (VARSBOUNDINEDITCHAIN Y))
                    (SETQ EXPR (OR (CAR (LAST Y))
                                   X))
                    (LISPXPUT 'RESPELLS NIL NIL LISPXHIST)      ; Essentially, a new call to DW is treated as a new event.
                    (COND
                       ((TAILP X (CAR Y))
                        (DWIMIFY2 X (CAR Y)))
                       ((AND (EQ (OR (CDR (FASSOC (SETQ TEM (CAAR Y))
                                                  DWIMEQUIVLST))
                                     TEM)
                                 'COND)
                             (NEQ (OR (CDR (FASSOC (SETQ TEM (CAADR Y))
                                                   DWIMEQUIVLST))
                                      TEM)
                                  'SELECTQ))
```

```
                        (DWIMIFY2 (CDR X)
                                  X)
                        X)
                       ([AND (EQ (OR (CDR (FASSOC (SETQ TEM (CAAR Y))
                                                  DWIMEQUIVLST))
                                     TEM)
                                 'SELECTQ)
                             (NEQ X (CADAR Y))
                             (CDR (FMEMB X (CAR Y]
                        (DWIMIFY2 (CDR X)
                                  X)
                        X)
                        (T (DWIMIFY1 X]
                  (Y                                          ; called from compileuserfn or compile1a.  X is the expression to
                                                              ; be dwimified.
                     (SETQ FAULTFN Y)
                     (AND (NULL EXPR)
                          (SETQ EXPR X))
                     ;; EXPR is supplied on calls from compileuserfn.  it is the top level def.  on calls from compile1a, x and expr are the
                     ;; same
                     (SETQ TEM (DWIMIFY1 X))
                     (AND DWIMIFY0CHANGE (DWIMARKASCHANGED FAULTFN SIDES))
                     TEM)
                  ((LISTP X)                                  ; e.g. user types in a direct call to dwimify an xpression
                   (SETQQ FAULTFN TYPE-IN)
                   (SETQ EXPR X)
                   (DWIMIFY1 X))
                  (T                                          ; DWIMIFY (functon-name)
                     (SETQ TEM (EXPRCHECK X))                 ; If EXPRCHECK performs spelling correction, it will rset FN.
                     (SETQ FAULTFN (SETQ FN (CAR TEM)))
                     (DWIMIFY1 (SETQ EXPR (CDR TEM)))
                     [COND
                        (DWIMIFY0CHANGE
                               ;; DWIMIFY0CHANGE is only bound in DWIMIFY0.  it is only reset (in RETDWIM) when DWIMIFYFLG
                               ;; and DWIMIFYING are both T.  It is true if there was ANY change in the entire expression.
                               ;; DWIMIFYCHANGE on the other hand is bound wheever DWIMIFYFLG is T, and it is true if there was
                               ;; any change in the prticular level expression being worked on.
                               (DWIMARKASCHANGED FN SIDES)
                               (COND
                                  ([OR (NOT (FGETD FN))
                                       (AND (NEQ DFNFLG 'PROP)
                                            (NOT (EXPRP FN]
                                   (DWIMUNSAVEDEF FN T]
                     FN])
```

(DWIMIFY0?

```
[LAMBDA (TAIL PARENT SUBPARENT FORMSFLG ONEFLG FAULTFN CLISPCONTEXT)
                                                              (* lmm "27-MAY-82 09:54")
```

;; DWIMIFY0? is an external entry to DWIMIFYBLOCK It is used to dwimify an expression where the contxt may or may not be under aother call to ;; dwimify. it is used by RECORD, MATCH etc. as well s by CLISP4 in CLISPIFY.

;; The value of DWIMIFY0? is NOT the expression (dwiified) but T or NIL depending on whether or not there was any change, i.e. the value of ;; dwiifychange.

```
(PROG NIL
      (SELECTQ DWIMIFYFLG
           (NIL ;; Under a call to WTFIX, but not under a call to DWIMIFY, e.g. from evaluating a CREATE expression in a user program.
                (SETQ NOFIXFNSLST0 NOFIXFNSLST)
                (SETQ NOFIXVARSLST0 NOFIXVARSLST))
           ((CLISPIFY VARSBOUND) ;; e.g. call from clispify or record package.  WAnt it to look like we are inside of a call to dwimify.  calling
                                 ;; function has already set up VARS and EXPR.
               (SETQ NOFIXFNSLST0 NOFIXFNSLST)
               (SETQ NOFIXVARSLST0 NOFIXVARSLST)
               [RETURN (PROG ((DWIMIFY0CHANGE T)
                              (DWIMIFYING T))                 ; This is going to be treated as though were a caal to dwimify.
                             (RETURN (DWMFY0]
           (EVAL                                              ; random call to dwimify0?  EVAL IS THE TOP LEVEL VALUE
                                                              ; OF DWIMIFYFLG
                 (SETQ NOFIXFNSLST0 NOFIXFNSLST)
                 (SETQ NOFIXVARSLST0 NOFIXVARSLST)
                 [RETURN (PROG (DWIMIFYFLG FAULTPOS EXPR VARS)
                               (RETURN (DWMFY0]
           NIL)
      (RETURN (DWMFY0])
```

(DWMFY0

```
[LAMBDA NIL
  (PROG ((DWIMIFYING (AND DWIMIFYFLG DWIMIFYING))
         (DWIMIFYFLG T)
         DWIMIFYCHANGE)
        (COND
           ((AND (NULL FORMSFLG)
```

```
                (EQ TAIL PARENT))
            (DWIMIFY1 TAIL CLISPCONTEXT))
           (T (DWIMIFY2 TAIL PARENT SUBPARENT FORMSFLG ONEFLG)))
      (RETURN DWIMIFYCHANGE])
```

(DWIMIFY1

```
[LAMBDA (FORM CLISPCONTEXT FORMSFLG)
  (DWMFY1 FORM])
```

(DWIMIFY1?

```
[LAMBDA (FORM CLISPCONTEXT FORMSFLG)
  (COND
     (DWIMIFYFLG (DWMFY1 FORM))
     (T ;; See comment in dwimify0.  DWIMIFY1?  is used where caller is not sure whether state variables have been set up.
        (PROG ((DWIMIFYING (AND DWIMIFYFLG DWIMIFYING))
               (DWIMIFYFLG T)
               DWIMIFYCHANGE)
              (SETQ NOFIXFNSLST0 NOFIXFNSLST)
              (SETQ NOFIXVARSLST0 NOFIXVARSLST)
              (RETURN (DWMFY1 FORM])
```

(DWMFY1

```
[LAMBDA (FORM)                                                    (* lmm " 3-Jan-86 21:29")
  (PROG ((X FORM)
         CARFORM TEM CLISPCHANGE 89CHANGE ATTEMPTFLG CARISOKFLG)
        [COND
           ((NLISTP FORM)
            (SETQ TEM (LIST X))
            (DWIMIFY2 TEM T)
            (RETURN (COND
                       ((CDR TEM)
                        TEM)
                       (T (CAR TEM]
    TOP (SETQ CARFORM (OR (CDR (FASSOC (CAR X)
                                       DWIMEQUIVLST))
                          (CAR X)))
        [COND
           ([AND (NEQ CARFORM 'LAMBDA)
                 (NEQ CARFORM 'NLAMBDA)
                 (OR (NULL (CHECKTRAN X))
                     CLISPRETRANFLG
                     (RETURN X))
                 (NOT (COND
                         [(LISTP CARFORM)
                          ;; Checks whether CAR is a function object with a remote translation.  Also converts to hash array from CLISP  if
                          ;; hash array exists.  CARISOKFLG is set so dont have to recheck at LP1.
                          (OR (EQ (SETQ TEM (OR (CDR (FASSOC (CAAR X)
                                                             DWIMEQUIVLST))
                                                (CAAR X)))
                                  'LAMBDA)
                              (EQ TEM 'NLAMBDA)
                              (SETQ CARISOKFLG (AND (CHECKTRAN CARFORM)
                                                    (NULL CLISPRETRANFLG]
                         ((LITATOM CARFORM)
                          (CLISP-SIMPLE-FUNCTION-P CARFORM]      ; The AND is true if CAR of form is not recognized.
            (COND
               [(PROG (NEXTAIL)
                      (RETURN (WTFIX0 X X X X)))                ; Successful correction.
                (COND
                   ((CHECKTRAN X)
                    (RETURN X))
                   [CLISPCHANGE (COND
                                   ((NEQ CLISPCHANGE 'PARTIAL)
                                    ;; The tail must be DWIMIFIED if the transformation did not affect the entire form, e.g. (FOO<...> ...)
                                    (RETURN FORM))
                                   ((LISTP CARFORM)
                                    (GO DWIMIFYTAIL))
                                   (T (SETQ CLISPCHANGE NIL)
                                      (GO TOP)                  ; Recheck CAR of FORM, as it may still be misspelled.

                                      ]
                   (89CHANGE (SETQ 89CHANGE NIL)
                          (GO TOP)                              ; Recheck CAR of FORM, as it still may be misspelled, e.g.
                                                                ; (conss8car X)

                             ]
               ((AND CLISPCHANGE (NEQ CLISPCHANGE 'PARTIAL))
                ;; This means a CLISPCHANGE failed and not to bother with dwimifying rest of form, e.g. a bad IF or FOR statement.
                (RETURN FORM))
               ((AND (NULL ATTEMPTFLG)
```

```
                    (LITATOM CARFORM))
                ;; ATTEMPTFLG is used to distinguish between the case where DWIM does not recognize the problem at all, and that where it
                ;; did but was unable to make the correction, e.g. a malformed IF, or else the user vetoed the correction.
                (SETQ NOFIXFNSLST0 (CONS CARFORM NOFIXFNSLST0]
;; The call to WTFIX is made before specific checks on CAR of FORM, since CAR of the FORM may be misspelled.
      (COND
         ((LISTP CARFORM)                                          ; Skip selectq
          (GO DWIMIFYTAIL)))
      [SELECTQ CARFORM
           (* ; "NIL")
           (DECLARE [MAPC (CDR X)
                          (FUNCTION (LAMBDA (X)
                                    (SELECTQ (CAR X)
                                         ((USEDFREE GLOBALVARS)
                                                                   ; SPECVARS AND LOCALVARS WOULD PRESUMABLY BE
                                                                   ; BOUND SOMEWHERE SO NO NEED TO ADD THEM
                                             (SETQ NOFIXVARSLST0 (UNION (LISTP (CDR X))
                                                                        NOFIXVARSLST0)))
                                         NIL])
           ('GO
                (AND DWIMCHECK#ARGSFLG (CDDR X)
                     (DWIMIFY1A X 1)))
           (SELECTQ (DWIMIFY2 (CDR X)
                              FORM T NIL T)
                    (AND (NLISTP (CADDR X))
                         (DWIMIFY2 (CDDR X)
                                   FORM NIL NIL T))
                    (SETQ X (CDDR X))
                    (PROG NIL
                      LP  (COND
                             ((NULL (CDR X))
                              (DWIMIFY2 X FORM T)
                              (RETURN FORM)))
                          (DWIMIFY2 (CDAR X)
                                    (CDAR X)
                                    T T)
                          (SETQ X (CDR X))
                          (GO LP)))
           ((SETQ SETN RPAQ SETARG)
                (AND (NOT (FMEMB (CADR X)
                                 VARS))
                     (NOT (FMEMB (CADR X)
                                 NOFIXVARSLST0))
                     (SETQ NOFIXVARSLST0 (CONS (CADR X)
                                               NOFIXVARSLST0)))
                (DWIMIFY2 (CDDR X)
                          FORM T))
           (COND [MAPC (CDR X)
                       (FUNCTION (LAMBDA (X)
                                         (DWIMIFY2 X X NIL T])
           (FUNCTION [DWIMIFY1 (COND
                                  ((LISTP (CADR X)))
                                  ((NULL (CDDR X))           ; Doesnt DWIMIFY for (FUCNTION FOO (X Y)) i.e. FUNARY
                                                             ; with atomic argument.
                                   (CDR X])
           (RESETVAR (DWIMIFY2 (CDDR X)
                               FORM T))
           ([LAMBDA NLAMBDA]
                ([LAMBDA (VARS)
                   (DWIMIFY2 (CDDR X)
                             FORM T]
                 (APPEND [OR (LISTP (CADR X))
                             (AND (CADR X)
                                  (LIST (CADR X]
                         VARS)))
           (COND
              ((EQMEMB 'BINDS (GETPROP CARFORM 'INFO))       ; PROG EQUIVALENTS
               ([LAMBDA (VARS)
                  (DWIMIFY2 (CDDR X)
                            FORM T]
                (NCONC [MAPCAR (CADR X)
                               (FUNCTION (LAMBDA (X)
                                           (COND
                                              ((NLISTP X)
                                               X)
                                              (T (DWIMIFY2 (CDR X)
                                                           X T)
                                                 [COND
                                                    ((NOT (LITATOM (CAR X)))
                                                     (DWIMIFY1A (CADR FORM)
                                                                (FMEMB X (CADR FORM]
                                                 (CAR X]
                       VARS)))
              ((CLISPNOEVAL CARFORM)                         ; Don't DWIMIFY the tails of nlambdas.
```

```
                )
               (T (GO DWIMIFYTAIL]
         (RETURN FORM)
      DWIMIFYTAIL
         (DWIMIFY2 (CDR X)
               FORM)
         (SETQ CARFORM (OR (CDR (FASSOC (CAR X)
                                        DWIMEQUIVLST))
                           (CAR X)))                    ; CARFORM may have changed if DWIMIFY2 changed X
         (COND
            [(LISTP CARFORM)
             (AND (NULL CARISOKFLG)
                  (NULL CLISPCHANGE)
                  (DWIMIFY1 CARFORM))
             ;; Note that if CAR is a list, it itself has not yet been dwimified, e.g. may be a misspelled LAMBDA.  However If CLISPCHANGE is not
             ;; NIL, this expression was produced by the call to WTFIX and hence is already dwimified.
             (COND
                ((AND (NULL FORMSFLG)
                      (NEQ (SETQ TEM (OR (CDR (FASSOC (CAAR X)
                                                      DWIMEQUIVLST))
                                         (CAAR X)))
                           'LAMBDA)
                      (NEQ TEM 'NLAMBDA)
                      (NULL CARISOKFLG)
                      (NULL (CHECKTRAN CARFORM)))
                 (DWIMIFY1A X)
                 (RETURN X]
             ((AND DWIMCHECK#ARGSFLG (EQ (ARGTYPE CARFORM)
                                         0)
                   (SELECTQ (SETQ TEM (NARGS CARFORM))
                        (0 (CDR X))
                        (1 (CDDR X))
                        (2 (CDDDR X))
                        (3 (CDDDDR X))
                        NIL))
              (DWIMIFY1A X TEM)))
         (RETURN FORM])
```

(DWIMIFY1A

```
  [LAMBDA (PARENT TAIL FN)                                  (* wt%: "10-DEC-80 23:36")
    (COND
       ((AND (NULL DWIMESSGAG)
             (OR FN (AND DWIMIFYFLG DWIMIFYING))
             (NEQ CLISPCONTEXT 'IFWORD))                    ; clispif handles this itself.
        (AND (FIXPRINTIN (OR FN FAULTFN))
             (LISPXSPACES 1 T))
        (COND
           ((EQ CLISPCONTEXT 'IFWORD))
           (T (LISPXPRIN1 '"(possible) parentheses error in
                          " T)
              (LISPXPRINT (RETDWIM2 PARENT TAIL)
                     T T)))
        (COND
           ((NUMBERP TAIL)
            (LISPXPRIN1 "too many arguments (more than " T)
            (LISPXPRIN1 TAIL T)
            (LISPXPRIN1 ")
                    " T))
           (TAIL (LISPXPRIN1 '"at " T)
                 (LISPXPRINT (CONCAT '"... " (SUBSTRING (RETDWIM2 TAIL NIL 2)
                                                        2 -1))
                       T T])
```

(DWIMIFY2

```
  [LAMBDA (TAIL PARENT SUBPARENT FORMSFLG ONEFLG ONLYSPELLFLG)
    (DWMFY2])
```

(DWIMIFY2?

```
  [LAMBDA (TAIL PARENT SUBPARENT FORMSFLG ONEFLG ONLYSPELLFLG CLISPCONTEXT)
    (COND
       (DWIMIFYFLG (DWMFY2))
       (T ;; See comment in dwimify0.  DWIMIFY2?  is used where caller is not sure whether state variables have been set up.
          (PROG ((DWIMIFYING (AND DWIMIFYFLG DWIMIFYING))
                 (DWIMIFYFLG T)
                 DWIMIFYCHANGE)
                (SETQ NOFIXFNSLST0 NOFIXFNSLST)
                (SETQ NOFIXVARSLST0 NOFIXVARSLST)
                (RETURN (DWMFY2])
```

(DWMFY2

```
  [LAMBDA NIL                                               ; Edited  4-Dec-86 11:02 by jop:
```

```
                                                                    ; handles tails.
    (AND (LISTP TAIL)
         (PROG ((TAIL0 TAIL)
                X CARPARENT CLISPCHANGE 89CHANGE NEXTAIL ATTEMPTFLG TEM FNFLG NOTOKFLG)
               (AND (OR (EQ SUBPARENT T)
                        (EQ PARENT TAIL))
                    (SETQ SUBPARENT TAIL))
```

;; Means dont ever back up beyond this point, e.g. in prog variables, if you write (PROG ((X FOO Y LT 3) .. dont want LT to gobble the x.))

```
               (SETQ CARPARENT (OR (CDR (FASSOC (CAR PARENT)
                                                DWIMEQUIVLST))
                                   (CAR PARENT)))
          LP   (SETQ CLISPCHANGE NIL)
               (SETQ 89CHANGE NIL)
               (SETQ NEXTAIL NIL)
               (COND
                  ((NULL TAIL)
                   (GO OUT))
                  ((LISTP (SETQ X (CAR TAIL)))
                   (AND (NEQ CLISPCONTEXT 'LINEAR)
                        (DWIMIFY1 X))
                   [AND FORMSFLG (EQ TAIL PARENT)
                        (OR (EQ CARPARENT 'LAMBDA)
                            (EQ CARPARENT 'NLAMBDA))
                        (/RPLNODE TAIL (CONS (CAR TAIL)
                                             (CDR TAIL]
                   (GO DROPTHRU))
                  ((NOT (LITATOM X))                                ; e.g. number, string, etc.

                   )
                  [(EQMEMB 'LABELS (GETPROP CARPARENT 'INFO))       ; this is a prog label.  or resetvars label etc.
                   (COND
                      ((AND DWIMCHECKPROGLABELSFLG (NOT (FMEMB X NOFIXVARSLST0))
                            (STRPOSL CLISPCHARRAY X))
                       (AND (FIXPRINTIN FAULTFN)
                            (LISPXSPACES 1 T))
                       (LISPXPRIN1 "suspicious prog label: " T)
                       (LISPXPRINT X T]
                  (NOSPELLFLG                                       ; none of the following corrections wanted
                       )
                  ((CLISPNOTVARP X)
                   ;; (CAR TAIL) is not recognized as a variable.  Note that when DWIMIFYING, WTFIX will be called on a variable which is used
                   ;; freely, but does not have a top level binding, i.e. DWIMIFYING hile the variable is bound is not sufficient, because we do not
                   ;; do a STKSCAN for its value, as this would be expensive.  (STKSCAN is done when DWIMIFY2 is called out of an
                   ;; evaluation.)
                   (COND
                      [(AND FORMSFLG (EQ TAIL PARENT)
                            (DWIMIFY2A TAIL 'QUIET))
                       ;; DWIMIFY2A calls CLISPFUNCTION?  to see if (CAR TAIL) is the name of a function.  If FORMSFLG is true and (CAR
                       ;; TAIL) is name of function, then TAIL may be one form with parenteeses removed.
                       (COND
                          ((OR (NEQ X (CAR TAIL))
                               (NEQ FORMSFLG 'FORWORD))
                           ;; Either the user has approved the combined spelling correction and insertion of paentheses, or else we are not
                           ;; under an I>S> without an oerator.  (E.g. FOR X IN Y WHILE ATOM PRINT X, In this cae dont want to insert
                           ;; parentheses.) Note that if FOO is also the name of a variable as well as a function, no harm will be done in cases
                           ;; like IF A THEN FOO _ X.  Only possible problem is for case like IF A THEN FOO _ X Y, where FFO is both a
                           ;; functionand a variable.  In this case, parens would be inserted, and then an error generated.  HOwever, this is
                           ;; extremely unlikely, since in most cases it would be written as IF A THEN FOO_X Y (not to mention the added
                           ;; improbability of FOO being both the name of a function and a variable.)
                           (GO ASK))
                          (T ;; (CAR TAIL) is the name of a function, but user hasnt been consulted, and we are under a FOR with no
                             ;; operator, so wait.
                             (SETQ FNFLG T)                         ; Now drop through to next COND and call to WTFIX (because
                                                                    ; (CAR TAIL) may be a miispelled variable.)

                           ]
                      ((AND (EQ FORMSFLG 'FORWORD)
                            (EQ TAIL (CDR PARENT))
                            (OR (LISTP CARPARENT)
                                (NULL NOTOKFLG)
                                (NULL FNFLG))
                            (DWIMIFY2A TAIL 'QUIET)
                            (OR (NEQ X (CAR TAIL))
                                (LISTP CARPARENT)))
                       ;; Corresponds to the case where the user left a DO out of a for statement.  Already know that the first thing in TAIL is not
                       ;; the name of a function.  However, only take action if the usr approves combined correction, (or (CAR PARENT) is a
                       ;; list.) since it is still possible that X is the (misspelled) name of a variable.
                       (SETQQ FORMSFLG FOR1)
                       (GO INSERT))
                      ((AND [LISTP (SETQ TEM (GETPROP X 'CLISPWORD]
                            [NEQ (CAR TEM)
```

```
                              (CAR (GETPROP CARPARENT 'CLISPWORD]
                         (CDDR TAIL))
                    (AND (EQ TAIL PARENT)
                         (SETQ NOTOKFLG T))

                    ;; E.g. (LIST X FOR X IN A --) The CDDR check is because very seldom you have an iterative statement only two
                    ;; elements long, but lots of places where iterative words can appear in another context, e.g. OF, TO, etc. See comment
                    ;; below on NOTOKFLG.  Note that if FORMSFLG is true and (EQ TAIL PARENT), then CLISPFUNCTION?  (via
                    ;; DWIMIFY2A) above would have returned T.

                    (DWIMIFY1A PARENT TAIL)                    ; Stop dwimifying, strong evidence that expression is screwed
                                                               ; up.
                    (GO OUT)))
                 (COND
                    ((AND [NULL (AND ONLYSPELLFLG (OR (EQ NOSPELLFLG T)
                                                      (AND NOSPELLFLG (NULL TYPE-IN?]
                          (WTFIX0 X TAIL PARENT SUBPARENT ONLYSPELLFLG))

                    ;; If both ONLYSPELLFLG and NOSPELLFLG are true, no point in calling WTFIX.  ONLYSPELLFLG is true on calls fro
                    ;; CLISPATOM2A.

                    (COND
                       (89CHANGE (SETQ NOTOKFLG NIL)
                                 (SETQ FNFLG NIL)              ; If 89CHANGE, then want to look at (CAR TAIL) again, e.g. ...
                                                               ; (CONS (CAR XX9))
                                 (GO LP)))
                    (GO DROPTHRU)))                            ; At this point we know that (CAR TAIL) is not ok.
                 (AND (EQ TAIL TAIL0)
                      (SETQ NOTOKFLG T))                       ; NOTOKFLG=T means first expression in TAIL was not
                                                               ; recognized as a variable.
                 [COND
                    ((AND FORMSFLG (EQ TAIL PARENT))

                    ;; After DWIMIFYING the whole tail, if CAR is still an atom, we may want to insert parentheses, e.g. (FOO _ X Y) is ok,
                    ;; but (FOO X Y) may need to be converted to ((FOO X Y))

                    )
                    [(FGETD X)

                    ;; Don't add a function name to NOFIXVARSLST0 since this is tantamount to sanctiooning it as a variale.

                    (COND
                       ((AND (EQ FORMSFLG 'FORWORD)
                             (EQ TAIL (CDR PARENT))
                             (OR (LISTP CARPARENT)
                                 (NULL NOTOKFLG)
                                 (NULL FNFLG)))
                        (SETQQ FORMSFLG FOR1)
                        (GO INSERT))
                       ((AND (NEQ CLISPCONTEXT 'IFWORD)
                             (NLISTP CLISPCONTEXT)
                             (NEQ ONLYSPELLFLG 'NORUNONS)
                             (NOT (EXPRP X))
                             (NEQ X 'E)
                             (NEQ X COMMENTFLG))               ; Printx message but dwimify rest of tail --- might not be a
                                                               ; parentheses error.
                        (DWIMIFY1A PARENT TAIL]
                    ((NULL ATTEMPTFLG)
                     (COND
                        ([NOT (AND CLISPFLG (GETPROP X 'CLISPTYPE]
                         (SETQ NOFIXVARSLST0 (CONS X NOFIXVARSLST0]
              (GO DROPTHRU)))
DROPTHRU
    (COND
       (ONEFLG (GO OUT)))
    [SETQ TAIL (COND
                  ((NULL NEXTAIL)
                   (CDR TAIL))
                  ((EQ NEXTAIL T)
                   NIL)
                  (T (CDR NEXTAIL]
    (GO LP)
OUT (COND
       ([OR (NULL FORMSFLG)
            (NOT (LITATOM (CAR TAIL0]
        (GO OUT1))
       ([OR (EQ FORMSFLG 'FOR1)
            (AND (EQ FORMSFLG 'FORWORD)
                 (OR (NULL NOTOKFLG)
                     (NULL FNFLG))
                 (LISTP (CADR TAIL0]

         ;; Corresponds to the cse where the user left out a DO.  Want to check this before below as in this case dont want to stick in
         ;; paens around entire form.

        (GO OUT1))
       ((EQ FORMSFLG T)                                        ; (CAR TAIL0) is the name of a functionand NOT the name of a
                                                               ; variable.
        (AND NOTOKFLG FNFLG (GO ASK)))
       [(CDR TAIL0)                                            ; FORMSFLG is FOR or IF
        (COND
           ((OR NOTOKFLG (DWIMIFY2A TAIL0 'QUIET))
```

```
                    ;; (CAR TAIL) is not the name of a variable, or else IS the name of a function.  The reason for the call to
                    ;; CLISPFUNCTION?  (via DWIMIFY2A) instead of checking FNFLG is that in the case that (CAR TAIL) was the name of
                    ;; a variable as indicated by NOTOKFLG=NIL, CLISPFUNCTION?  would not have been called earlier.
                    (/RPLNODE TAIL0 (CONS (CAR TAIL0)
                                          (CDR TAIL0)))
                    (GO OUT1]
               ((AND NOTOKFLG FNFLG)
                ;; (CAR TAIL) is not the name of a variable and is the name of a function, but nothing follows it.  E.g. IF -- THEN RETURN
                ;; ELSE --
                (/RPLNODE TAIL0 (CONS (CAR TAIL0)
                                      (CDR TAIL0)))
                (GO OUT1)))
          OUT1
               (RETURN (COND
                           ((NULL ONEFLG)
                            TAIL0)
                           (NOTOKFLG
                                  ;; In this way, the function thatcaled DWIMIFY2 can find out whether or not the atom in question is OK.
                                  ;; NOte that if it appears on NOFIXLST, it is OK, i.e. havng been seen before, we treat it the same as a
                                  ;; variable or what not.
                                  NIL)
                           ((NULL NEXTAIL)
                            TAIL)
                           ((EQ NEXTAIL T)
                            PARENT)
                           (T NEXTAIL)))
          ASK (COND
                  ((NULL (FIXSPELL1 [COND
                                       (TYPE-IN? '"")
                                       (T (CONCAT '"... " (SUBSTRING (RETDWIM2 TAIL0)
                                                                     2 -1]
                                  (CONCAT '"... " (MKSTRING (RETDWIM2 TAIL0))
                                          '")")
                                  NIL T))
                   (GO OUT1)))
          INSERT
               (/RPLNODE TAIL (CONS (CAR TAIL)
                                    (CDR TAIL)))
               (DWIMIFY1 (CAR TAIL)
                      CLISPCONTEXT)
               (GO DROPTHRU])
```

(DWIMIFY2A

```
[LAMBDA ($TAIL $TYP)                                                    (* wt%: 25-FEB-76 1 54)
  (CLISPFUNCTION? $TAIL $TYP [FUNCTION (LAMBDA (X Y)
                                         (SUBSTRING (RETDWIM2 Y)
                                                    2 -1]
          [FUNCTION (LAMBDA (X Y)
                      (CONCAT [MKSTRING (RETDWIM2 (COND
                                                     [(LISTP X) ; Run-on.
                                                      (CONS (CAR X)
                                                            (CONS (CDR X)
                                                                  (CDR Y]
                                                     (T (CONS X (CDR Y]
                              '")"]
          $TAIL])
```

(CLISPANGLEBRACKETS

```
[LAMBDA (LST)                                                           (* wt%: "26-JUN-78 01:20")
  (PROG [WORKFLAG (NCONCLKUP (CLISPLOOKUP 'NCONC))
                 (NCONC1LKUP (CLISPLOOKUP 'NCONC1]
        (RETURN (SHRIEKER LST])
```

(SHRIEKER

```
[LAMBDA (LOOKAT)
```

;; Shrieker is designed to 'understand' expressions of the form (! A B !! C !! D E F), where A, B, C,... represent lists, ! indicates that the list following ;; it is to be (non-destructively) expanded (e.g. A's elements are to be brought to the top level of the list which contains A), and !! indicates that the ;; list following it is to be destructively expanded. Thus, if A= (H I J), B= (K L M), C= (N O P), the result of evaluating (! A !! B C) should be a list (H I ;; J K L M C). SHRIEKER does not actually evaluate the list given to it, but rather returns a form which will have the correct evaluation. Thus, if ;; SHRIEKER is given the (shriekified) list (! A !! B C), it will return the form (APPEND A (NCONC1 B C)). Should A,B,C have the values given ;; above, then evaluation of this form will leave A unchanged, but B will have been destructively altered, and will now evaluate to the list (K L M (N ;; O P)).

```
(PROG (CARTEST RESULTP)
      (COND
         ((OR (ATOM LOOKAT)
              (NLISTP LOOKAT))
          (SETQ WORKFLAG NIL)
          (RETURN LOOKAT)))
```

;; As is evident from a look at the code, SHRIEKER is a fairly straightforward recursive prog; analysis of the argument, LOOKAT, is doen in effect ;; from the tail of LOOKat to its head. I>e. given LOOKAT SHRIEKER separates it into two parts (roughly car and cdr), where one part

```
;; (CARTEST) is the first element of LOOKAT that is not ! or !! , and the other part is the tail of LOOKAT below CARTEST-- LOOKAT is reset to
;; evaluate to this tail and SHRIEKER is called recursively on the new LOOKAT, eventually returning a list structure, to which we setq RESULTP,
;; that is the LISP equivalent of LOOKAT (which, with its !'s and !!'s is an expression in CLISP).  The calling incarnation of SHRIEKER uses
;; RESULTP and its knowledge of the shriek-sysmbol (! or ! ! or !!) immediately before CARTEST, to determine how CARTEST and RESULTP
;; should be used to form the list structure that will be returned, possibly to higher level incarnations of SHRIEKER.  into then possibly incarnations
;; SHRIEKER.
     (SETQ CARTEST (CAR LOOKAT))
     (SETQ LOOKAT (CDR LOOKAT))
     [RETURN (COND
                ((EQ CARTEST '!!)
                 (GO A1))
                [(EQ CARTEST '!)
                 (COND
                    (LOOKAT (SETQ CARTEST (CAR LOOKAT))
                           (SETQ LOOKAT (CDR LOOKAT))
                           (COND
                              ((EQ CARTEST '!)
                               (GO A1)))
                           ;; This conditional insures that SHRIEKER will understnad that the sequence ! ! means the atom !!.  Control
                           ;; will be sent to the statement after A1, which will make sure that CARTEST is NCONCed onto RESULTP (if
                           ;; car of RESULTP is APPEND, CONS, NCONC1, or LIST) or will stuff CARTEST into second place in
                           ;; RESULTP, which is presumalby an NCONC expression-- all provided that WORKFLAG is NIL...
                           (SETQ RESULTP (SHRIEKER LOOKAT))
                                                          ; Here's our recursive call to SHRIEKER..
                           (COND
                              ((NULL RESULTP)
                               ;; WORKFLAG is a flag that is passed between incarnations of SHRIEKER and is the means by which
                               ;; SHRIEKER is able to distinguish between user-created code and SHRIEKER-created code.  If
                               ;; WORKFLAG eq's T then SHRIEKER knows that what has been returned as RESULTP is
                               ;; user-created code and should not be altered.
                               (SETQQ WORKFLAG !IT)
                               (LIST 'APPEND CARTEST))
                              ((ATOM RESULTP)
                               (SETQQ WORKFLAG APPENDING)
                               (LIST 'APPEND CARTEST RESULTP))
                              ((NULL WORKFLAG)
                               (SETQQ WORKFLAG APPENDING)
                               (LIST 'APPEND CARTEST RESULTP))
                              (T
                                 ;; If the COND falls througn to this point then we may assume that RESULTP is SHRIEKER-created
                                 ;; and do a SELECTQ on car of RESULTP (which should be either APPEND, NCONC, NCONC1,
                                 ;; CONS, or LIST) to determine whether we should stuff CARTEST into RESULTP or not.
                                 (SELECTQ WORKFLAG
                                      (APPENDING (ATTACH CARTEST (CDR RESULTP))
                                                 RESULTP)
                                      ((NCONCING CONSING LISTING NCONC1ING)
                                           (SETQQ WORKFLAG APPENDING)
                                           (LIST 'APPEND CARTEST RESULTP))
                                      (!IT (SETQQ WORKFLAG APPENDING)
                                           (ATTACH CARTEST (CDR RESULTP))
                                           RESULTP)
                                      (!!IT (SETQQ WORKFLAG APPENDING)
                                            (LIST 'APPEND CARTEST (CADR RESULTP)))
                                      (LIST 'APPEND CARTEST RESULTP]
                [LOOKAT
                       ;; If we arrive here then we know that SHRIEKER's arguemnt-- hte intial value of LOOKAT--is a list, the first
                       ;; element of which is not ! or !!.  Accordingly, we attempt to CONS or LIST together CARTEST and RESULTP,
                       ;; depending on the nature of RESULTP and the value of WORKFLAG left by the recursive call to SHRIEKER in
                       ;; the statement below.
                       (SETQ RESULTP (SHRIEKER LOOKAT))
                       (COND
                          ((NULL WORKFLAG)
                           (SETQQ WORKFLAG LISTING)
                           (LIST 'LIST CARTEST))
                          (T (SELECTQ WORKFLAG
                                  ((CONSING APPENDING NCONCING NCONC1ING)
                                       (SETQQ WORKFLAG CONSING)
                                       (LIST 'CONS CARTEST RESULTP))
                                  (LISTING (ATTACH CARTEST (CDR RESULTP))
                                           RESULTP)
                                  ((!!IT !IT)
                                       (SETQQ WORKFLAG CONSING)
                                       (LIST 'CONS CARTEST (CADR RESULTP)))
                                  (LIST 'CONS CARTEST RESULTP]
                (T ;; If we reach this point then we know that SHRIEKER was called on a singleton, i.e. the intial vlaue of LOOKAT was a
                   ;; list of one element, so we create the appropriate list structure around that element and setq WORKFLAG to NIL,
                   ;; enabling a possible parent SHRIEKER to modify our code.
                   (SETQQ WORKFLAG LISTING)
                   (LIST 'LIST CARTEST]
A1   (RETURN (COND
                (LOOKAT (SETQ CARTEST (CAR LOOKAT))
                       (SETQ LOOKAT (CDR LOOKAT))
```

```
                              (SETQ RESULTP (SHRIEKER LOOKAT))
                              (COND
                                 ((NULL RESULTP)
                                  (SETQQ WORKFLAG !!IT)
                                  (LIST NCONCLKUP CARTEST))
                                 ((ATOM RESULTP)
                                  (SETQQ WORKFLAG NCONCING)
                                  (LIST NCONCLKUP CARTEST RESULTP))
                                 ((NULL WORKFLAG)
                                  (SETQQ WORKFLAG NCONCING)
                                  (LIST NCONCLKUP CARTEST RESULTP))
                                 (T (SELECTQ WORKFLAG
                                         (NCONCING (ATTACH CARTEST (CDR RESULTP))
                                                   RESULTP)
                                         ((APPENDING CONSING)
                                              (SETQQ WORKFLAG NCONCING)
                                              (LIST NCONCLKUP CARTEST RESULTP))
                                         (NCONC1ING (SETQQ WORKFLAG NCONCING)
                                                    (LIST NCONCLKUP CARTEST (CADR RESULTP)
                                                          (CONS 'LIST (CDDR RESULTP))))
                                         (!IT (SETQQ WORKFLAG NCONCING)
                                              (LIST NCONCLKUP CARTEST (CADR RESULTP)))
                                         (!!IT (SETQQ WORKFLAG NCONCING)
                                               (ATTACH CARTEST (CDR RESULTP))
                                               RESULTP)
                                         (LISTING (COND
                                                     ((NULL (CDDR RESULTP))
                                                      (SETQQ WORKFLAG NCONC1ING)
                                                      (LIST NCONC1LKUP CARTEST (CADR RESULTP)))
                                                     (T (SETQQ WORKFLAG NCONCING)
                                                        (LIST NCONCLKUP CARTEST RESULTP))))
                                         (LIST NCONCLKUP CARTEST RESULTP])
```

(CLISPRESPELL

```
  [LAMBDA (TL WORDS FLG)                                    (* lmm " 4-SEP-83 23:31")
```

;; CLISPRESPELL essentially asks is it possible to inerpret (CAR TAIL) as one of WORDS. It first checks to make sure (CAR TAIL) isnt already
;; something else-- e.g. a function, variable, member of NOFIXFNSLST (which is the same as being a function) etc.

```
    (AND (NEQ NOSPELLFLG T)
         (OR (NOT NOSPELLFLG)
             TYPE-IN?)
         (LISTP TL)
         (LITATOM (CAR TL))
         (NOT (CLISP-SIMPLE-FUNCTION-P (CAR TL)))
         (CLISPNOTVARP (CAR TL))
         (MISSPELLED? (CAR TL)
                NIL WORDS FLG])
```

(EXPRCHECK

```
  [LAMBDA (X)                                               (* wt%: "14-FEB-78 00:06")
    (PROG (D)
          (COND
             ((NOT (LITATOM X))
              (ERROR X '"not a function."))
             [(EXPRP (SETQ D (VIRGINFN X T]
             ((GETD X)
              (GO NOEXPR))
             ([NULL (AND DWIMFLG (SETQ D (MISSPELLED? X 70 USERWORDS NIL NIL (FUNCTION GETD]
              (ERROR X '"not defined."))
             ([NOT (EXPRP (SETQ D (VIRGINFN (SETQ X D]
              (GO NOEXPR)))
          (AND ADDSPELLFLG (ADDSPELL X 0))
          (RETURN (CONS X D))
      NOEXPR
          (ERROR X '"not an expr."])

)

(DEFINEQ
```

(CLISPATOM0

```
  [LAMBDA (CHARLST TAIL PARENT)                             (* bvm%: "21-Nov-86 18:05")
    (AND (NULL SUBPARENT)
         (SETQ SUBPARENT PARENT))
    (PROG ((CURRTAIL TAIL)
           (NOFIXVARSLST1 NOFIXVARSLST0)
           89FLG TEM)
      TOP (SELECTQ (DWIMUNDOCATCH 'CLISPATOM1 (SETQ TEM (CLISPATOM1 TAIL)))
                (:RESPELL                                   ; A misspelling was detected.  Need to fix it now.
                          (SETQ NOFIXVARSLST0 NOFIXVARSLST1)
                          (AND DWIMIFYFLG (SETQ CLISPCHANGE NIL))
                          (COND
                             ((PROG1 (CLISPELL TAIL)
                                  (SETQ CHARLST (DUNPACK (CAR TAIL)
                                                         WTFIXCHCONLST)))
```

```
                        ;; MIsspelling found.  Note that even if the word wasnt found, LST is reset since some tentative changes were
                        ;; tried, it was probably clobbered.
                        (SETQ CURRTAIL TAIL)
                        (GO TOP))))
            (NIL                                                    ; error
                 (SETQ NOFIXVARSLST0 NOFIXVARSLST1))
            (RETURN TEM))
         (RETURN (COND
                    (89FLG                                          ; E.G. N*8FOO -- fix the 8-9 error first.
                           [PROG ((FAULTX (CAR CURRTAIL)))
                                 (SETQ TEM (FIX89 FAULTX (CAR 89FLG)
                                                  (LENGTH 89FLG]
                           (COND
                              ((AND TEM (LITATOM (CAR TAIL)))
                               (CLISPATOM0 (DUNPACK (CAR TAIL)
                                                    WTFIXCHCONLST1)
                                           TAIL PARENT])
```

(CLISPATOM1

```
  [LAMBDA (TAIL)                                                    (* lmm "29-Jul-86 00:25")
```

;;; This function and its subfunctions handle infix operators. LST is an exploded list of characters for CAR of TAIL, which is a tail of PARENT. If LST ;;; contains an CLISP operator, or CAR of TAIL is one, CLISPATOM1 scans the rest of tail until it reaches the end of this cluster. For example, if TAIL is ;;; (... A* B + C D+E ...), the scan will stop after C. The scan separates out the operators from the operands. Note that because any operand can be a ;;; list, and hence separated from its operator, an operator can occur interior to an atom, as in A*B, at the end of an atom, as in (A* (--)), at the front of an ;;; atom, as in ((--) *A), or by itself, as in ((--) * (--)). Therefore, we permit the same options when the operand is a atomic, i.e. the user can type A*B, A* ;;; B, A *B, or A * B. Note that in the latter two cases, the first argument to the operator is not contained in TAIL, and it is necessary for CLISPATOM1 to ;;; back tail up one element using PARENT.

;; After the scan has been completed, the form for the first operator is assembled. Since operators are always processed left to right, the first ;; operand to this operator is always the single element preceding it (unless it is a unary operator). The right boundary, and hence the second ;; operand, is determined by the operator, e.g. * is tighter than +, which is tighter than LS, etc. Thus ... A*B+C ... becomes ... (ITIMES A B) + C ... ;; while ... A+B*C ... becomes ... (IPLUS A B * C) In either case, the rest of this cluster is processed from within this call to CLISPATOM1, thereby ;; taking advantage of the fact that we know that the atoms do not contain operators, and therefore don't have to be unpacked and examined ;; character by character.

```
(PROG ((L CHARLST)
       (LST0 CHARLST)
       CURRTAIL-1 CLTYP CLTYP1 ENDTAIL BROADSCOPE BACKUPFLG OPRFLAG NOTFLG TYP ATMS NOSAVEFLG TENTATIVE TEM
       BRACKET (BRACKETCNT 0)
       ISFLG)
      (COND
         ((SETQ CLTYP1 (GETPROP (CAR CURRTAIL)
                                'CLISPTYPE))
          (GO NEXT2)))
  TOP (SETQ ATMS NIL)
  LP  (COND
         ((NULL L)                                                  ; End of an atom.
          (COND
             ((NULL TYP)                                            ; If we have gone through the first atom without finding an CLISP
                                                                    ; operator, we are done.
              (COND
                 ((NULL 89FLG)                                      ; The case where there was an 8 or 9 and an operator has been
                                                                    ; handled in CL89CHECK.

                  )
                 (CURRTAIL
                        ;; 8 and 9 errors are handled here instead of back in CLISPATOM where there is similar code, because there may
                        ;; be more than one 8 or 9 in the expression, and the first one may be ok, e.g. 8*X*8ADD1 Y
                        (AND [FIX89A (CAR CURRTAIL)
                                     (CAR (LISTP 89FLG))
                                     (IMINUS (SETQ TEM (LENGTH 89FLG]
                             (FIX89 FAULTX (CAR (LISTP 89FLG))
                                    TEM)
                             (GO OUT3)))
                 ((AND TYPE-IN? (NEQ NOSPELLFLG T)
                       (EQ (CAR (LISTP 89FLG))
                           LPARKEY)
                       (EQ (CAR (SETQ TEM (FLAST CHARLST)))
                           RPARKEY))
                  ;; This corresponds to the case where an atom was typed in containing both an 8 and a 9, e.g. FILES?89 or 8EDITF9.
                  ;; Note that if the atom were part of a larger expression, either CAR of form, or appearing in a tail, (as indicated by
                  ;; CURRTAIL being non-NIL), the fix is performed by FIX89, and involves editing the expression.  In the case covered
                  ;; here, the fix requires changing the EVAL to an apppropriate APPLY.
                  (FIX89TYPEIN (FMEMB LPARKEY (SETQ TEM (LDIFF CHARLST TEM)))
                               TEM T)))
              (RETURN NIL))
             (LST0 (SETQ OPRFLAG T)                                 ; OPRFLAG is T means the element just processed did NOT end
                                                                    ; in an operator, e.g. A+B, or just A.
                   (SETQ TEM (PACK LST0))
                   ;; Collects characters to the right of the last operator in the atom, or all the characters in the atom, if it contained no
                   ;; operator.
                   (SETQ ATMS (NCONC1 ATMS TEM))
```

```
                    (SETQ NOTFLG NIL)))
             (SETQ 89FLG NIL)
             (GO NEXT)))
          [COND
             ((FMEMB (CAR L)
                     CLISPCHARS)
              (COND
                 ((SETQ CLTYP1 (GETPROP (CAR L)
                                        'CLISPTYPE))
                  [SELECTQ (CAR L)
                       (- [COND
                             ((NULL (AND (EQ L LST0)
                                         (CLUNARYMINUS? OPRFLAG)))
                              ;; Says minus is binary.  See comments i CLUNARYMINUS?.  By replacing binary minus with +- in
                              ;; CLISPATOM1, all the rest of the CLISP function can treat minus as unary.
                              (FRPLACA L '+-)
                              (SETQ CLTYP1 (GETPROP '+- 'CLISPTYPE])
                       (%' (AND (NEQ L LST0)
                                (GO LP1))
                           ;; ' is ignored interior to atoms, e.g. USER can have a function named ATOM' or a variable named A' which is not
                           ;; necessarily defined or bound at time of DWIMIFYing.

                           )
                       (COND
                          [BRACKET (COND
                                      ((EQ (CAR L)
                                           (CAR BRACKET))
                                       (SETQ BRACKETCNT (ADD1 BRACKETCNT)))
                                      ((EQ (CAR L)
                                           (CADR BRACKET))
                                       (SETQ BRACKETCNT (SUB1 BRACKETCNT)))
                                      (T (GO OPR]
                          [(EQ CLTYP1 'BRACKET)
                           [SETQ BRACKET (LISTP (GETPROP (CAR L)
                                                         'CLISPBRACKET]
                           (COND
                              ((EQ (CAR L)
                                   (CAR BRACKET))
                               (SETQ BRACKETCNT (ADD1 BRACKETCNT)))
                              (T (DWIMERRORRETURN (LIST (LIST 'BRACKET (CAR BRACKET))
                                                        PARENT]
                          (89FLG)
                          ((AND (NEQ NOSPELLFLG T)
                                (OR (NULL NOSPELLFLG)
                                    TYPE-IN?)
                                (OR (EQ (CAR L)
                                        LPARKEY)
                                    (EQ (CAR L)
                                        RPARKEY)))
                           (SETQ 89FLG L]
                       (GO OPR))
                      ([AND BRACKET (CAR L)
                            (EQ (CAR L)
                                (LISTGET1 BRACKET 'SEPARATOR]
                       (GO OPR]
LP1   (COND
          ((AND OPRFLAG (NULL BROADSCOPE)
                (ZEROP BRACKETCNT)
                (EQ L CHARLST))
           ;; If OPRFLAG is T and the first character in LST is not an operator, no need to scan further, e.g. A*B C unless we are processing a
           ;; broad scope operator, e.g. (A EQ FOO B) or unless ANGCNT is not 0, i.e. we are inside of an <> pair.
           (OR ENDTAIL (SETQ ENDTAIL CURRTAIL))
           ;; If ENDTAIL has not been set yet, set it.  Note that ENDTAIL may already have been set, e.g. A*B+C D, in which case ENDTAIL
           ;; would correspnd to the position of the +.
           (GO OUT)
           ;; If this is the first character in an atom, then we cango to out, e.g. A+B C.  HOwever, this may be the first character following a >, as
           ;; in FOO_<A B>C, in which case we have to finish out the atom.

           ))
      (SETQ L (CDR L))                                  ; Peel off the current character and go on.
      (GO LP)
NEXT                                                    ; We have just exhausted the lit of characters for an atm.
      [COND
          ((NULL TAIL)                                  ; We were originally given just an atom, e.g. user types
                                                        ; FOO_FIE.
           (SETQ TAIL ATMS)
           (OR PARENT (SETQ PARENT TAIL)))
          ([AND TAIL (OR (CDR ATMS)
                         (NEQ (CAR (LISTP ATMS))
                              (CAR CURRTAIL]
           ;; Splice burst version of atom into CURRTAIL, and set CURRTAIL to point to the as yet unexamined part of it.  If the OR is not true,
           ;; CURRTAIL would not be changd so don't bother e.g. (LIST A + B * C)
```

```
            [/RPLNODE CURRTAIL (CAR (LISTP ATMS))
                   (NCONC (CDR ATMS)
                          (SETQ CURRTAIL (LISTP (CDR (SETQ CURRTAIL-1 CURRTAIL]
                                                                  ; CURRTAIL-1 is used for backing up, see below.

            )
            (T (SETQ CURRTAIL (LISTP (CDR (SETQ CURRTAIL-1 CURRTAIL]
         (COND
            ((NULL CURRTAIL)                                      ; We have reached the end of the faulty form.
             (GO OUT)))
      NEXT1
                                                                  ; Look at the next thing in CURRTAIL.
         (COND
            ([AND OPRFLAG DWIMIFYFLG ONEFLG (NULL BROADSCOPE)
                  (ZEROP BRACKETCNT)
                  (OR (NOT (LITATOM (CAR CURRTAIL)))
                      (AND (NOT (GETPROP (CAR CURRTAIL)
                                         'CLISPTYPE))
                           (NOT (GETPROP (NTHCHAR (CAR CURRTAIL)
                                                  1)
                                         'CLISPTYPE]
             (OR ENDTAIL (SETQ ENDTAIL CURRTAIL))
             (GO OUT))
            ((AND (OR OPRFLAG (NEQ (CAR CURRTAIL-1)
                                   '%'))
                  [OR (LITATOM (CAR CURRTAIL))
                      (AND (NUMBERP (CAR CURRTAIL))
                           (MINUSP (CAR CURRTAIL))
                           (AND (NULL BRACKET)
                                OPRFLAG
                                (CLBINARYMINUS? CURRTAIL-1 CURRTAIL]
                  (CLISPNOTVARP (CAR CURRTAIL)))
             ;; The OR check is to handle cases like (.. ' F/L) which I think means wquote the whole thing.  NOte that this comes up in expressions
             ;; like <A 'F/L> since when SHRIEKER calls DWIMIFY2, the ' and F/L have already been split apart.
                                                                  ; dont call clbinaryminus?  if last thing ended in an operator.  e.g.
                                                                  ; ((foo) + -2)
             (COND
                ([AND (SETQ CLTYP1 (GETPROP (CAR CURRTAIL)
                                            'CLISPTYPE))
                      (NOT (AND (EQ (CAR (LISTP CLTYP1))
                                    'BRACKET)
                                (NULL BRACKET]
                 (GO NEXT2)))
             [SETQ LST0 (SETQ L (SETQ CHARLST (UNPACK (CAR CURRTAIL]
             (COND
                ((AND BRACKET (SETQ TEM (FMEMB (CADR BRACKET)
                                               (CDDR L)))
                      (NOT (FMEMB (CAR BRACKET)
                                  L)))
                 ;; < and > are thought of as brackets, rather than operaaors.  Therefore this is necessary in order thatthings like <1 2 -1> work,
                 ;; i.e. --- not treated as binary in this case, also <a b 'c>, and finally if A*B is the name of a variable <X A*B> Note that this
                 ;; doesnt quite handle all cases: <A*B> where A*B is the name of a variable, will be broken apart, but then it isnt clear whats
                 ;; intended.
                 (CLRPLNODE CURRTAIL (PACK (LDIFF L TEM))
                        (CONS (PACK TEM)
                              (CDR CURRTAIL)))
                 (GO NEXT1)))
             (GO TOP))
            ((AND OPRFLAG (SETQ TEM (LISTP (NLEFT TAIL 2 CURRTAIL)))
                  (NEQ (CAR TEM)
                       '%')
                  (NEQ (CAR TEM)
                       '%:)
                  [OR (NULL (CAR CURRTAIL))
                      (AND (LISTP (CAR CURRTAIL))
                           (NOT (CLISPFUNCTION? (CAR CURRTAIL)
                                                'OKVAR]
                  (CLISPFUNCTION? (SETQ TEM (CDR TEM))
                         'NOTVAR
                         [FUNCTION (LAMBDA (X Y)
                                           (CONCAT X (COND
                                                       ((NULL Y)
                                                        '"()")
                                                       (T (RETDWIM2 Y]
                         [FUNCTION (LAMBDA (X Y)
                                           (MKSTRING (CONS X (RETDWIM2 Y]
                         (CAR CURRTAIL)))                         ; This clause checks for user typing in apply mode, e.g. X_CONS
                                                                  ; (A B)
             (SETQQ TENTATIVE CERTAINLY)                          ; Once you print a message, you dont want to go and try another
                                                                  ; interpretation.
             (/RPLNODE TEM (CONS (CAR TEM)
                                 (CAR CURRTAIL))
                    (CDR CURRTAIL))
             [SETQ CURRTAIL (LISTP (CDR (SETQ CURRTAIL-1 TEM]
             (SETQ OPRFLAG T)
```

```
            (SETQ NOTFLG NIL)
            (GO NEXT1)))
         (COND
            ((AND OPRFLAG (NULL BROADSCOPE)
                  (ZEROP BRACKETCNT))                          ; Finished.  E.g. A*B (--)
             (OR ENDTAIL (SETQ ENDTAIL CURRTAIL))
             (GO OUT))
            ([SETQ CURRTAIL (LISTP (CDR (SETQ CURRTAIL-1 CURRTAIL]
                                                               ; E.g. A* (--)
             (SETQ OPRFLAG T)
             (SETQ NOTFLG NIL)
             (GO NEXT1))
            (T (GO OUT)))
      NEXT2
                                                               ; (CAR CURRTAIL) is an operaaor.  CLTYP1 is its CLISPTYPe.
         [SELECTQ (CAR CURRTAIL)
             (- [COND
                   ((NULL (CLUNARYMINUS? OPRFLAG))             ; The minus is biary.  SEe comments at earlier call to
                                                               ; CLUNARYMINUS?  in CLSPATOM1.
                    (/RPLNODE CURRTAIL '+- (CDR CURRTAIL))
                    (SETQ CLTYP1 (GETPROP '+- 'CLISPTYPE])
             ((-> =>)
                  [COND
                     ((EQ TYP '%:)
                      (SETQ CLTYP CLTYP1)
                      (GO NEXT3))
                     (T (DWIMERRORRETURN (CAR CURRTAIL])
             (COND
                [BRACKET (COND
                             ((EQ (CAR BRACKET)
                                  (CAR CURRTAIL))
                              (SETQ BRACKETCNT (ADD1 BRACKETCNT)))
                             ((EQ (CADR BRACKET)
                                  (CAR CURRTAIL))
                              (SETQ BRACKETCNT (SUB1 BRACKETCNT]
                ([SETQ TEM (LISTP (GETP (CAR CURRTAIL)
                                        'CLISPBRACKET]
                 (COND
                    ((EQ (CAR CURRTAIL)
                         (CAR TEM))
                     (SETQ BRACKET TEM)
                     (SETQ BRACKETCNT (ADD1 BRACKETCNT)))
                    (T (DWIMERRORRETURN (LIST (LIST 'BRACKET (CAR TEM))
                                              PARENT]
         (COND
            (ENDTAIL)
            [(NULL TYP)                                        ; This is the first operator.
             (SETQ TYP (CAR CURRTAIL))
             (SETQ CLTYP CLTYP1)
             (SETQ BROADSCOPE (GETPROP TYP 'BROADSCOPE))
             (SETQ NOTFLG (EQ (SETQ TEM (GETPROP TYP 'LISPFN))
                              'NOT]
            (NOTFLG
                   ;; NOTFLG is true when we are processing a NOT opeator, and it immediately precedes the current operator.  In this case,
                   ;; the scope of the NOT is the scope of the next opeator, e.g. (X ~GR FOO Y)
                   (SETQ CLTYP CLTYP1)
                   (SETQ BROADSCOPE (GETPROP (CAR CURRTAIL)
                                             'BROADSCOPE))
                   (SETQ NOTFLG (EQ (GETPROP (CAR CURRTAIL)
                                             'LISPFN)
                                    'NOT))                     ; So that NOTFLG is not turned off when there are two ~'s in a
                                                               ; row, e.g. (X ~~GR FOO Y OR Z)

                   )
            ((STOPSCAN? CLTYP1 CLTYP (CAR CURRTAIL)
                    OPRFLAG)
             ;; This operator delimits the scope of the first operator found.  Set ENDTAIL to be the first thing not within the scope of the operator.
             ;; The AND is so that a unary operator will terminate the scope of a binary operator that has a right hand operand, e.g. X+Y -Z, X_Y
             ;; 'Z, etc.
             (SETQ ENDTAIL CURRTAIL)))
         (SETQ ISFLG (EQ [CAR (LISTP (GETPROP (CAR CURRTAIL)
                                              'CLISPCLASS]
                         'ISWORD))
      NEXT3
         [SETQ OPRFLAG (AND BRACKET (EQ (CAR CURRTAIL)
                                        (CADR BRACKET]         ; OPRFLAG is T aater > since no right hand operand is reuired.
         (COND
            ([SETQ CURRTAIL (LISTP (CDR (SETQ CURRTAIL-1 CURRTAIL]
             (GO NEXT1)))
      OUT                                                      ; We are finished scanning.  Now call CLISPATOM2 to assemble
                                                               ; the correct form.
         [COND
            ((NEQ (CAR (LISTP TAIL))
                  TYP)
             (GO OUT1))
```

```
            ((GETPROP TYP 'UNARYOP)
             (GO OUT1))
            ((OR (EQ PARENT TAIL)
                 (EQ SUBPARENT TAIL))                          ; E.g. (+ X) or (SETQ Y + X)
             (DWIMERRORRETURN (LIST 1 TAIL PARENT]
         (SETQ TAIL (NLEFT (OR SUBPARENT PARENT)
                           1 TAIL))

;; SUBPARENT can be used to mark that point in a list beyond which not to back up, e.g. (LAMBDA (X) FOO X LT Y)

         (SETQ BACKUPFLG T)
 OUT1
         (CLISPATOM2)
 OUT2
         [COND
            ([AND [OR [NUMBERP (SETQ CLTYP (GETPROP (SETQ TYP (CAR ENDTAIL))
                                                    'CLISPTYPE]
                      (NUMBERP (CAR (LISTP CLTYP]
                   (NULL (AND DWIMIFYFLG CLISPCONTEXT (EQ [CAR (LISTP (GETPROP (CAR ENDTAIL)
                                                                                'CLISPWORD]
                                                              'FORWORD)
                              (OR (EQ CLISPCONTEXT 'FORWORD)
                                  (EQ CLISPCONTEXT 'FOR/BIND]

             ;; i used to have just a (NULL (AND DWIMIFYFLG ONEFLG)) but this means tht if you have a predicate in an iterative statement, e.g.
             ;; when x=y+z that it doesnt dwimify completely.  the above clause handles it but i dont remember why i had the original one in there.
                                                               ; reason for the or check is so that DO doesnt get treated as an
                                                               ; IS word when coming from an i.s.
             (SETQ TEM (CLISPATOM1A TYP CLTYP TAIL))
             (COND
                ((OR DWIMIFYFLG (EQ TEM PARENT))
                 (SETQ TAIL TEM]
 OUT3
         (SETQ TEM (COND
                      ((AND (NULL FORMSFLG)
                            (EQ TAIL PARENT))
                       T)
                      (T TAIL)))
         (COND
            (DWIMIFYFLG (SETQ NEXTAIL TEM))
            (BACKUPFLG (SETQ NEWTAIL TEM)))
         [SETQ TEM (COND
                      ((AND (NULL FORMSFLG)
                            (OR (NULL PARENT)
                                (EQ TAIL PARENT)))
                       TAIL)
                      (T (CAR (LISTP TAIL]
         (COND
            ((AND TENTATIVE (NEQ TENTATIVE 'CERTAINLY))

             ;; Tentative is set to CERTAINLY when we are sure the correction will be CLISP, and to avoid somebody else setting to T .  IN this
             ;; casse there will be no message.  This occurs when a message has already been printed, e.g. in X*FOO Y , when user is asked
             ;; FOO Y -> (FOO Y), the approveal of the CLISP transformation is implicit.

             (SETQ CLISPCHANGES (LIST TEM (CLISPATOM1B)
                                      TAIL
                                      (CDR TAIL)
                                      TENTATIVE NOFIXVARSLST0))

             ;; note --- (CDR TAIL) used to be endtail in above expression, however, for situations where clispatom1a munches for a while, this
             ;; does not produce the right message, e.g. dwimifying ....  FOO:1='ZAP ...

             (AND DWIMIFYFLG (SETQ CLISPCHANGE NIL))
             (DWIMERRORRETURN)))
         (RETURN TEM)
 OPR                                                           ; We have hit an operator inside of an atom.
         (COND
            ((NEQ L LST0)
             (SETQ TEM (PACK (LDIFF LST0 L)))                  ; Collects characters to the right of the last operator in the atom.
             (COND
                ((AND (FLOATP TEM)
                      (OR (EQ (CAR L)
                              '+)
                          (EQ (CAR L)
                              '+-))
                      (EQ (CAR (NLEFT LST0 1 L))
                          'E))                                 ; E.G. X+1.0E-5*Y
                 (AND (EQ (CAR L)
                          '+-)
                      (FRPLACA L '-))
                 (GO LP1)))
             (SETQ ATMS (NCONC1 ATMS TEM))
             (SETQ NOTFLG NIL)))
         (SETQ ATMS (NCONC1 ATMS (CAR L)))
         [COND
            (ENDTAIL)
            [(NULL TYP)                                        ; First operator.
             (SETQ TYP (CAR L))
             (SETQ CLTYP CLTYP1)
             (SETQ BROADSCOPE (GETPROP TYP 'BROADSCOPE))
             (SETQ NOTFLG (EQ (GETPROP TYP 'LISPFN)
```

```
                          'NOT]
        [NOTFLG
               ;; It is not only necessary that we are processing a NOT, but that it immediately precede the current operator.
               (SETQ CLTYP CLTYP1)
               (SETQ BROADSCOPE (GETPROP (CAR L)
                                         'BROADSCOPE))
               (SETQ NOTFLG (EQ (GETPROP (CAR L)
                                         'LISPFN)
                                'NOT]
        ((STOPSCAN? CLTYP1 CLTYP (CAR L)
                (OR (NEQ L LST0)
                    OPRFLAG))
         ;; This operator delimits the scope of the first operator found.  Set ENDTAIL to be the first thing not within the scope of the operator.
         (SETQ ENDTAIL (COND
                          ((EQ L CHARLST)                    ; The scope delimiting operator was the first thing in an atom,
                                                             ; e.g. A*B +C or A*B + C.
                           CURRTAIL)
                          (T (FLAST ATMS]
      [SETQ OPRFLAG (AND BRACKET (EQ (CAR L)
                                     (CADR BRACKET]          ; OPRFLAG is T aater > since no right hand operand is reuired.
      (COND
         ([AND (CDR L)
               CURRTAIL
               (OR (AND BRACKET (EQ (CAR L)
                                    (CADR BRACKET)))
                   (EQ (CAR L)
                       '~]
          ;; So that the rest of the atom will be looked at as a unit before being unpacked, e.g. ~GR, want to look up GR.  Also want to look at
          ;; rest of atom as a unit following >, e.g. FOO_<A B>EQUAL C.  By starting over with a new atom, we also perform the OPRFLAG
          ;; terminating check, as in FOO_<A B>C.
          (/RPLNODE CURRTAIL (CAR CURRTAIL)
                 (CONS (PACK (CDR L))
                       (CDR CURRTAIL)))
          (SETQ L NIL)))
      (SETQ 89FLG NIL)
      (SETQ LST0 (CDR L))
      (SETQ L (AND (NEQ (CAR L)
                        '%')
                   (CDR L)))                                 ; Following a ' no operaars are recognized in the rest of the atm.
      (GO LP])
```

(CLRPLNODE

```
[LAMBDA (X A D)
   (PROG ((L (CDR UNDOSIDE)))
         (COND
            (NOSAVEFLG                                       ; X is not contained in original expression, so don't bother to
                                                             ; save
                       (GO OUT)))
     LP  (COND
            ((EQ L (CDR UNDOSIDE0))                          ; X has not previously been saved
             (/RPLNODE X A D)
             (RETURN X))
            ((NEQ X (CAAR L))
             ;; If X is EQ to CAR of one of the entries on UNDOOSIDE, then the contents of this node have already been saved, so it is ok to
             ;; smash it.
             (SETQ L (CDR L))
             (GO LP)))
     OUT (FRPLACA X A)
         (FRPLACD X D)
         (RETURN X])
```

(STOPSCAN?

```
[LAMBDA (CLTYP2 CLTYP1 OPR OPRFLAG)                          (* wt%: "16-AUG-78 21:47")
   ;; STOPSCAN?  is T if operator corresponding to CLTYPX would stop scan for operator corresponding to CLTYP, i.e. if former is of lower or same
   ;; precedence as latter.
   (AND CLTYP2 CLTYP1 (PROG NIL
                            (COND
                               [BROADSCOPE (COND
                                              ((OR (NOT (ZEROP BRACKETCNT))
                                                   (EQ CLTYP2 'BRACKET))
                                               (RETURN NIL]
                               [(EQ CLTYP2 'BRACKET)
                                (RETURN (COND
                                           [(EQ OPR (CAR BRACKET))
                                                                 ; a left bracket
                                      ;; e.g. for X+Y< -- stop scanning.  note that for binary brackets, it never stops as is consistent with
                                      ;; them being very tight operators, i.e. FOO_A{..} parses as FOO_ (A{..})
                                            (AND OPRFLAG (EQ BRACKETCNT 1)
                                                 (GETP OPR 'UNARYOP]
```

```
                                        ((EQ CLTYP1 'BRACKET)
                                         ;; i.e. if OPR is the right bracket for BRACKET, or if OPR is some other bracket inside of
                                         ;; scope of BRACKET.
                                    ;; if cltyp1 is ot a bracket, then bracket is not the operator, and should really treat the whole bracketed
                                    ;; expression as an operand and not stop the scan.
                                          (ZEROP BRACKETCNT]
                                   ((NOT (ZEROP BRACKETCNT))
                                    (RETURN NIL))
                                   ((GETPROP OPR 'UNARYOP)
                                    (RETURN OPRFLAG)
                                    ;; If OPRFLAG is NIL, we have just seen a unary operator with no operand, so under no circumstance stop
                                    ;; the scan.  E.g. X*-Y.  Note that this does NOT say do not consider next operand as possible operatr, so
                                    ;; that X*-+Y will generate an error, not try to multiply X by (minus +).  The case whee the unary operaar is '
                                    ;; is handled specially in CLISPATOM1 and CLISPATOM1A.

                                   ))
                                (RETURN (COND
                                           ([NOT (ILESSP (COND
                                                            ((ATOM CLTYP1)
                                                             CLTYP1)
                                                            (T (CDR CLTYP1)))
                                                         (COND
                                                            ((ATOM CLTYP2)
                                                             CLTYP2)
                                                            (T (CAR CLTYP2]
                                            T)
                                           ([AND (LISTP CLTYP2)
                                                 (ILESSP (CDR CLTYP2)
                                                         (COND
                                                            ((ATOM CLTYP1)
                                                             CLTYP1)
                                                            (T (CDR CLTYP1]
                                    ;; Not sure of this.  it is an attempt to handle the A*B_C+D case.  Here the initial cltyp is that of *, but
                                    ;; since the right precedence of _ is looser than that of *, means that it should be operative.
                                        (SETQ CLTYP CLTYP2)
                                        NIL])
```

(**CLUNARYMINUS?**

```
[LAMBDA (OPRFLAG)                                                    (* lmm "20-May-84 20:02")
```

;; True if minus is unary. This is the case when either (1) it immediately follows an operator (the (AND TYP (NULL OPRFLAG)) check) or (2) it is ;; the first thing in a list (the (EQ CURRTAIL SUBPARENT) check) or else, car of form is a function and not a variable, and --- negates its first ;; argument. The case where car of form is amisspeleed function is handled, because the tentatitve correction for binry minus will be tried, and ;; then when spelling correction on function name suceeds, this will be implemeneted. then there will be another call to clispatom when its ;; aagument is evaluated, and this time the functionis spelled right. Note that the cse where car of a form is a misspelled variable works also, even ;; when the variabl could be confusec for a function, since the correction on the variable is tried first.

```
(OR (AND TYP (NULL OPRFLAG))
    (EQ CURRTAIL SUBPARENT)
    (AND (EQ CURRTAIL (CDR SUBPARENT))
         (FNTYP (CAR SUBPARENT))
         (OR (LISTP (CAR SUBPARENT))
             (CLISPNOTVARP (CAR SUBPARENT)))
         (OR TYPE-IN? (AND CLISPHELPFLG (FIXSPELL1
                                         [CONS (CAR SUBPARENT)
                                               (CONS (CADR SUBPARENT)
                                                     (NCONC [AND (EQ (CADR SUBPARENT)
                                                                     '-)
                                                                 (LIST (RETDWIM2 (CADDR SUBPARENT]
                                                            (RETDWIM2 (CDDR (COND
                                                                               ((EQ (CADR SUBPARENT)
                                                                                    '-)
                                                                                (CDR SUBPARENT))
                                                                               (T SUBPARENT]
                                         '"the %"-%" is unary"
                                         '" " T])
```

(**CLBINARYMINUS?**

```
[LAMBDA ($TAIL MINUSTAIL)                                            (* wt%: "10-OCT-78 21:22")
```

;; used when a negative number follows a list. we dont know if a space was typed before the --- or not, so in situation ike ((list) -2) or (x* (list) -2) ;; we ask. warren ^Z

;; the EQ used to check tail against subparent. i changed it because on calls to dwimify0? from record, e.g. (ADD z:1 -1), was trying to treat -1 as ;; binary even though it shouldnt have.

```
(AND (EQ TAIL PARENT)
     [OR (LISTP (CAR TAIL))
         (NUMBERP (CAR TAIL))
         (AND (LITATOM (CAR TAIL))
              (NOT (CLISPNOTVARP (CAR TAIL)))
              (NOT (CLISPFUNCTION? TAIL]
     (OR TYPE-IN? (AND CLISPHELPFLG (FIXSPELL1 [CONS (CAR $TAIL)
                                                     (CONS (CADR $TAIL)
                                                           (NCONC [AND (EQ (CADR $TAIL)
```

```
                                                                    '-)
                                                                (LIST (RETDWIM2 (CADDR $TAIL]
                                                            (RETDWIM2 (CDDR (COND
                                                                              ((EQ (CADR $TAIL)
                                                                                   '-)
                                                                               (CDR $TAIL))
                                                                              (T $TAIL]
                                                '"the %"-%" is a clisp operator"
                                                " " T)))
        (OR (NULL MINUSTAIL)
            (/RPLNODE MINUSTAIL '- (CONS (MINUS (CAR MINUSTAIL))
                                         (CDR MINUSTAIL])
```

(**CLISPATOM1A**

```
  [LAMBDA (TYP CLTYP TAIL NOSAVEFLG)                                    (* lmm " 4-SEP-83 22:50")
```

;;; This function is similar to CLISPATOM1 except that elements of TAIL do not have to be unpacked. It is called from either CLISPATOM1 or
;;; CLISPATOM2 when more than one operator was encountered in a cluster. CADR of TAIL is TYP, the next operator to be processed, and CLTYP is
;;; its CLISPTYPE. CLISPATOM1A scans down TAIL looking for the right hand boundary of TYP, but does not unpack any atoms. It then calls
;;; CLISPATOM2 to assemble the form, and then if necessary repeats the process. For example, if the original cluster was A+B*C, the call to
;;; CLISPATOM2 from CLISPATOM1 would replace this with (IPLUS A B * C). CLISPATOM2 would then call CLISPATOM1A with TAIL= (B * C).
;;; Similary, if the original cluster were A*B+C, the call to CLISPATOM2 from CLISPATOM1 would replace this with (ITIMES A B) with + C having been
;;; spliced into the tail. CLISPATOM1 would then call CLISPATOM1A with TAIL= ((ITIMES A B) + C ...)

```
    (PROG (ENDTAIL OPRFLAG BROADSCOPE CLTYP0 BRACKETCNT BRACKET ISFLG)
      TOP (SETQ ISFLG (EQ (CAR (GETPROP TYP 'CLISPCLASS))
                          'ISWORD))
          (SETQ BRACKETCNT (COND
                             ((SETQ BRACKET (GETP TYP 'CLISPBRACKET))
                              1)
                             (T 0)))
          [SETQ ENDTAIL (COND
                          ((EQ TYP (CAR TAIL))                  ; TYP is car of TAIL for unary operatrs, CADR for binary.
                           TAIL)
                          (T (CDR TAIL]
          [COND
             ([AND (EQ TYP '~)
                   (SETQ CLTYP0 (GETPROP (CADR ENDTAIL)
                                         'CLISPTYPE]
              (SETQ CLTYP CLTYP0)
              (SETQ ENDTAIL (CDR ENDTAIL]
          (SETQ BROADSCOPE (GETPROP TYP 'BROADSCOPE))
          (SETQ OPRFLAG NIL)
      LP  [COND
             ((EQ (CAR ENDTAIL)
                  '%')
              (SETQ ENDTAIL (CDDR ENDTAIL))
              (SETQ OPRFLAG T))
             (T (SETQ ENDTAIL (CDR ENDTAIL]
          (COND
             ((NULL ENDTAIL)
              (GO OUT))
             ((SETQ CLTYP0 (GETPROP (CAR ENDTAIL)
                                    'CLISPTYPE))
              (SETQ ISFLG (EQ (CAR (GETPROP (CAR ENDTAIL)
                                            'CLISPCLASS))
                              'ISWORD))
              [COND
                 [BRACKET (COND
                            ((EQ (CAR ENDTAIL)
                                 (CAR BRACKET))
                             (SETQ BRACKETCNT (ADD1 BRACKETCNT)))
                            ((EQ (CAR ENDTAIL)
                                 (CADR BRACKET))
                             (SETQ BRACKETCNT (SUB1 BRACKETCNT]
                 ((EQ CLTYP0 'BRACKET)
                  (SETQ BRACKET (GETPROP (CAR ENDTAIL)
                                         'CLISPBRACKET))
                  (COND
                     ((EQ (CAR ENDTAIL)
                          (CAR BRACKET))
                      (SETQ BRACKETCNT (ADD1 BRACKETCNT)))
                     (T (DWIMERRORRETURN (LIST (LIST 'BRACKET (CAR BRACKET))
                                               PARENT]
              (AND (STOPSCAN? CLTYP0 CLTYP (CAR ENDTAIL)
                              OPRFLAG)
                   (GO OUT))
              [SETQ OPRFLAG (AND (EQ CLTYP0 'BRACKET)
                                 (EQ (CAR ENDTAIL)
                                     (CADR BRACKET]              ; E.g. X_<A B> see comment in CLISPATOM1

             )
             ((AND OPRFLAG (ZEROP BRACKETCNT)
                   (NULL BROADSCOPE))
              (GO OUT))
             (T (SETQ OPRFLAG T)))
```

```
          (GO LP)
      OUT (CLISPATOM2)
          (COND
             ([AND (SETQ CLTYP (GETPROP (SETQ TYP (CAR ENDTAIL))
                                        'CLISPTYPE))
                    (NULL (AND DWIMIFYFLG CLISPCONTEXT ONEFLG (EQ (CAR (GETPROP (CAR ENDTAIL)
                                                                                'CLISPWORD))
                                                                  'FORWORD)
                               (OR (EQ CLISPCONTEXT 'FORWORD)
                                   (EQ CLISPCONTEXT 'FOR/BIND]
              ;; E.g. A+B*C+D.  The first call to CLISPATOM1A is with TAIL (B * C + D).  The first call to CLISPATOM2 changes this to ((ITIMES B
              ;; C) + D), and then we loop back to the top of CLISPATOM1A.  The reason for the OR is so that do does not get treated as an IS
              ;; WORD when coming from an i.s.
              (GO TOP)))
          (AND TENTATIVE (SETQQ TENTATIVE PROBABLY))          ; Don't consider another interpretation if there are two or more
                                                              ; CLISP operators in this cluster.
          (RETURN TAIL])
```

(CLISPATOM1B

```
  [LAMBDA NIL                                                 (* wt%: 25-FEB-76 1 41)
```

;; Copies changes.

```
  (PROG ((L UNDOSIDE)
         (L1 (CDR UNDOSIDE0))
         LST)
    LP   [COND
            ((EQ (SETQ L (CDR L))
                 L1)
             (RETURN LST))
            ((LISTP (CAAR L))
             (SETQ LST (CONS (CONS (CAAR L)
                                   (CONS (CAAAR L)
                                         (CDAAR L)))
                             LST)))
            ((EQ (CAAR L)
                 '/PUTHASH)                                   ; Pattern match.
             (SETQ LST (CONS (LIST '/PUTHASH (CADAR L)
                                   (GETHASH (CADAR L)
                                            CLISPARRAY)
                                   CLISPARRAY)
                             LST]
          (GO LP])
```

(CLISPATOM2

```
  [LAMBDA NIL                                                 (* bvm%: "21-Nov-86 11:56")
```

;; Assembles LISP forms from the CLISP expressions

```
  (PROG ((PARENT PARENT)
         VAR1 VAR2 Z (UNARYFLG (GETPROP TYP 'UNARYOP))
         (LISPFN (GETPROP TYP 'LISPFN))
         TEM NEGFLG (CLISPCLASS (GETPROP TYP 'CLISPCLASS))
         ENDTAIL-1)
        (AND (NEQ TYP (CAR TAIL))
             UNARYFLG
             (SETQ TAIL (CDR TAIL)))
```

;; On calls from CLISPATOM1A, TYP is always CADR of TAIL. e.g. in X+Y 'Z, on the call to CLISPATOM2 to process ', TAIL would be (IPLUS X
;; Y) ' Z.

```
        [COND
           ((AND (SETQ TEM (GETP (CAR ENDTAIL)
                                 'CLISPBRACKET))
                 (EQ (CAR ENDTAIL)
                     (CADR TEM)))
            (SETQ ENDTAIL-1 ENDTAIL)
            (SETQ ENDTAIL (CDR ENDTAIL]
        [COND
           ((AND (NOT (EQ 0 BRACKETCNT))
                 (EQ CLTYP 'BRACKET))
            (DWIMERRORRETURN (LIST [LIST 'BRACKET (COND
                                                    ((MINUSP BRACKETCNT)
                                                     (CAR BRACKET))
                                                    (T (CADR BRACKET]
                                   PARENT)))
           ((NULL (CDR TAIL))
            (DWIMERRORRETURN 1))
           ((NULL ENDTAIL))
           [(AND (NULL FORMSFLG)
                 (GETPROP (CAR ENDTAIL)
                          'CLISPTYPE))
            (COND
               ((NEQ TAIL PARENT))
               ([OR (NULL (GETPROP (CAR ENDTAIL)
                                   'UNARYOP))
                    (AND (EQ (CAR ENDTAIL)
```

```
                        '~)
                     (GETPROP (CADR ENDTAIL)
                           'CLISPTYPE]                    ; X+Y~=Z is OK.

          )
         ((AND UNARYFLG (CLISPATOM2C TAIL))               ; E.G. (~FOO 'X Y) is OK.

          )
         (T                                               ; E.G. (X + Y ' Z)
            (DWIMERRORRETURN (LIST 2 ENDTAIL PARENT]
      [(AND (NULL FORMSFLG)
            (EQ PARENT TAIL))
       ;; An missing operand error is going to be generated if something isnt done in the next COND, e.g (X*Y Z)
       (COND
          ((AND ENDTAIL DWIMIFYFLG (EQ CLISPCONTEXT 'IFWORD)
                (CLISPRESPELL ENDTAIL CLISPIFWORDSPLST)) ; Found a correction; tell CLISPIF to try again.
           (CL:THROW 'CLISPIF0 :RESPELL))
          [(AND ENDTAIL (CLISPRESPELL ENDTAIL CLISPINFIXSPLST))
           ;; E.g. (X + Y LSS Z). Note that we do not try to correct spelling on infixes unless the form is otherwise going to cause an eror,
           ;; e.g. in (FOO X_Y ORR --), the ORR is not checked for here. Thus in the event that the next thing on ENDTAIL is a CLISP
           ;; transformation, e.g. (FOO X_Y Z_W), we do not have to do any extra work. This algorithm contains the implicit assumption
           ;; that all the operatrs on CLISPINFIXSPLST (i.e. the ones we correct for) will terminate the scope of all non-broadscope
           ;; operators. Otherwise, if FOO is a non-broadscope operator, and FIE would not terminate FOO, and FIE is on
           ;; CLISPINFIXSPLST, the form (LIST A FOO B FIEE C) would parse as (LIST (A FOO B) FIE C), which is wrong. In this case,
           ;; not only would we have to backup to CLISPATOM1 using RETEVAL as in CLIPATOMB, we would also have to check for
           ;; misspelled operaaors appearng in CAR of ENDTAIL even when an error would not otherwise be generated, e.g. in (LIST X_Y
           ;; Z_W) we would have to check the spelling of Z_W. Note that when the current operator is broadscope, we always perform
           ;; spelling correction (via the call to DWIIFY! in CLISPTOM2B) since once parentheses are inserted, we can't distinguish e.g. (X
           ;; AND Y ORR Z) from (X AND (Y ORR Z)).
           (COND
              (DWIMIFYFLG (CL:THROW (COND
                                       ((LISTP CLISPCONTEXT)
                                        ;; We want to go back to the clispatom1 above this call to wtfix, e.g. consider X AND Y_T ORR Z. In
                                        ;; this case, we are dwimifying (Y_T ORR Z) but we want to go back to higher level. Used to do this
                                        ;; via (RETDWIM0 'CLISPATOM1 (RETDWIM0 'WTFIX)), but now we just tell WTFIX to throw again.
                                        'WTFIX)
                                       (T 'CLISPATOM1))
                                    :RESPELL]
          ([CLISPATOM2C (COND
                           (UNARYFLG TAIL)
                           (T (CDR TAIL]                  ; E.G. FOO_GETP 'FIE 'EXPR

          )
          (T                                              ; E.g. (LIST * X Y)
             (DWIMERRORRETURN (LIST 2 ENDTAIL PARENT]
      ((CLISPATOM2C (COND
                       (UNARYFLG TAIL)
                       (T (CDR TAIL]
(COND
   ((EQ CLTYP 'BRACKET)
    ;; Note that as currently implemented, ENDTAIL can be NIL. i.e. there is no check for whether or not matching > where actually
    ;; found. This enables user to insert expressions like <<X <Y and terminate them with a ] instead of having to balance out the angle
    ;; brackets.
    [SETQ Z (COND
               (UNARYFLG
                     ;; IFCONTEXT is preserved because in the case of an unpaired <, i.e., if CAR of ENDTAIL was not >, the
                     ;; scope may include the entire IF statement, e.g. IF A THEN <B C ELSSE D If we generated an error when
                     ;; there was no matching <, then IFCONTEXT could be bound to NIL here. This is not done in order that
                     ;; angle brackets can be closed with a ')' or ']' , e.g. typing in << -- < -- ']'
                     (DWIMIFY2? (SETQ TEM (LDIFF (CDR TAIL)
                                                 ENDTAIL-1))
                           TEM NIL T NIL NIL (AND DWIMIFYFLG (EQ CLISPCONTEXT 'IFWORD)
                                                  CLISPCONTEXT)))
               (T (DWIMIFY2? (SETQ TEM (LDIFF (CDDR TAIL)
                                              ENDTAIL-1))
                        TEM NIL T NIL NIL (AND DWIMIFYFLG (EQ CLISPCONTEXT 'IFWORD)
                                               CLISPCONTEXT]
    [SETQ Z (COND
               [[NULL (SETQ TEM (LISTGET1 BRACKET 'DWIMIFY]
                (SETQ TEM (LISTGET1 BRACKET 'SEPARATOR))
                [SETQ Z (MAPCONC Z (FUNCTION (LAMBDA (X)
                                               (AND (OR (NULL TEM)
                                                        (NEQ X TEM))
                                                    (LIST X]
                (CONS (OR LISPFN (LISTGET1 BRACKET 'LISPFN))
                      (COND
                         (UNARYFLG Z)
                         (T (CONS (CAR TAIL)
                                  Z]
               [UNARYFLG (COND
                            ((EQ TEM 'CLISPANGLEBRACKETS)
                             (CLISPANGLEBRACKETS Z))
```

```
                                    (T (APPLY* TEM Z]
                           (T (APPLY* TEM (CAR TAIL)
                                  Z]
                  (COND
                     ((AND (NULL ENDTAIL)
                           (NULL FORMSFLG)
                           (OR (NULL PARENT)
                               (EQ PARENT TAIL)))
                      (CLRPLNODE TAIL (CAR Z)
                             (CDR Z))
                      (SETQ Z T))
                     (T (CLRPLNODE TAIL Z ENDTAIL)
                        (SETQQ Z PARTIAL)))
                  (AND DWIMIFYFLG (SETQ CLISPCHANGE Z))
                  (GO OUT)))
TOP [COND
       [(EQ TYP '~)
        (COND
           ((NEQ (CAR TAIL)
                 TYP)
```

;; In most cases, CAR TAIL will be TYP. However, can also case where user leaves out a space, e.g. (LIST~X Y) or (X~MEMB ;; Y) or (X~=Y). Performing this check simplifies the code considerably.

```
            (SETQ TAIL (CDR TAIL))
            (SETQ UNARYFLG T)))
        (COND
           ((EQ TAIL PARENT)                                  ; E.g. (~ FOO X) or (~ X) or (~FOO XXA) or (~X)

            )
           ([OR (EQ (SETQ TEM (CADR TAIL))
                    '~)
                (AND (GETPROP TEM 'CLISPTYPE)
                     (NOT (GETPROP TEM 'UNARYOP]           ; E.G. (X ~MEMB Y), or (X~MEMB Y), i.e. ~ is being used to
                                                           ; negate an operator.
            (SETQ BACKUPFLG T)
            (SETQ TAIL (NLEFT PARENT 1 TAIL))
            (CLRPLNODE TAIL (CAR TAIL)
                   (CDDR TAIL))                            ; Remove the NOT.
            (SETQ UNARYFLG NIL)
            (SETQ TYP (CADR TAIL))
            (SETQ LISPFN (GETPROP TYP 'LISPFN))
            (SETQ NEGFLG (NOT NEGFLG))
            (GO TOP]
       ((OR (EQ TYP 'NOR)
            (EQ TYP 'nor))
        (SETQQ TYP AND)
        (CLRPLNODE (CDDR TAIL)
               'NOT
               (CONS (CADDR TAIL)
                     (CDDDR TAIL]
    (COND
       ((EQ ENDTAIL (CDR TAIL))
```

;; Occurs when a unary operator is immediately followed by another operatr, e.g. X='+ or -*X. In former case, ' will be transformed ;; above in CLISPATOM. Note that ' MUST be a CLISP operator in order for things like FOO_' (--) to work.

```
        (DWIMERRORRETURN 2)))
A   (COND
       ((AND (NULL (SETQ VAR2 (LDIFF (COND
                                        (UNARYFLG (CDR TAIL))
                                        (T (CDDR TAIL)))
                                     ENDTAIL)))
             (NULL UNARYFLG))                              ; E.G. (LIST X*)
        (DWIMERRORRETURN 1)))
    [COND
       (BROADSCOPE (CLISPBROADSCOPE TYP VAR2 (COND
                                                ((EQ (CAR CLISPCLASS)
                                                     'ISWORD)
                                                 'IS)
                                                (T PARENT)))
```

;; Inserts parens in VAR2 e.g. converts (FOO X AND FIE Y) to (FOO X AND (FIE Y)) see comment in clispbroadscope

```
              (COND
                 ((NEQ TAIL (SETQ TEM (OR SUBPARENT PARENT)))
```

;; SUBPARENT can be used to mark that point in a list beyond which not to back up, e.g. (LAMBDA (X) FOO X LT Y)

```
                  (CLRPLNODE TEM (LDIFF TEM (CDR TAIL))
                         (CDR TAIL))                       ; inserts parens in VAR1, e.g. (FOO X AND Y) -> ((FOO X) AND
                                                           ; Y)
                  (SETQ BACKUPFLG T)
                  (SETQ TAIL TEM)))
              (COND
                 ((OR (NULL DWIMIFYFLG)
                      (NULL CLISPCHANGE))
                  (CLISPBROADSCOPE1 TAIL PARENT BACKUPFLG]
B   (SETQ VAR1 (CAR TAIL))
    (SELECTQ TYP
        (%: (AND LISPFN (GO C))                            ; means user has redefined : as a normal lisp operator
```

```
                (SETQ Z (CLISPCAR/CDR (SETQ TEM VAR2)))
                ;; the value returned by CLISPCAR/CDR indicates whether there was more than one operator involved, and is used to set
                ;; CLISPCHANGE below.
                (SETQ TEM (CLISPATOM2D NIL VAR1))                ; Inserts new expressioninto TAIL.
                (COND
                   (DWIMIFYFLG (AND CLISPCHANGE (GO OUT))
                            (SETQ CLISPCHANGE TEM))
                   ((NOT (ATOM (CADR VAR2)))
                    (GO OUT)))
                (CLISPATOM2A (CDR VAR2)
                      VAR2)
                (AND TENTATIVE Z (SETQQ TENTATIVE PROBABLY))     ; Means there was more than one : operator.
                (GO OUT))
             (_ [COND
                   ((NLISTP VAR1)
                    (SETQ TEM TYP))
                   (T                                            ; _ in connection with a : operator.
                      [SETQ TEM (SELECTQ (CAR VAR1)
                                     (CAR 'RPLACA)
                                     (CDR 'RPLACD)
                                     ((NCONC NCONC1)
                                         (CAR VAR1))
                                     ((replace REPLACE)          ; From record declaration assigmnent.
                                          (CLISPATOM2D NIL (CLISPRECORD VAR1 VAR2 T))
                                                                 ; Where the right hand operand to the _ will be DWIMIFIED, and
                                                                 ; TENTATIVE set, etc.
                                          (GO C1))
                                     (COND
                                        ([OR (SETQ TEM (GETPROP (CAR VAR1)
                                                                'SETFN))
                                             (PROGN (DWIMIFY1? VAR1)
                                                    (SETQ TEM (GETPROP (CAR VAR1)
                                                                       'SETFN]
                                         ;; E.G. User converts X \ FOO to (GETP X FOO), and puts PUT on SETFN of GETP, so that X \
                                         ;; FOO_T becomes (PUT X FOO T)
                                         (CLISPATOM2D NIL (CONS (CLISPLOOKUP TEM (CADR VAR1))
                                                                (APPEND (CDR VAR1)
                                                                        VAR2)))
                                     ;; SETFN.  Must be handled this way because VAR1 may correspond to more than one operand, e.g.
                                     ;; X \ FOO_T -> (ELT X FOO) _T and must go to (SETA X FOO T)
                                          (GO C1))
                                         (T (DWIMERRORRETURN '_]
                      (SETQ LISPFN (GETPROP TEM 'LISPFN))
                      (SETQ VAR1 (CADR VAR1]
                (SETQ LISPFN (CLISPLOOKUP TEM VAR1 NIL LISPFN))
                [COND
                   ((AND (EQ LISPFN 'SETQ)
                         (EQ (CAR VAR2)
                             '%')
                         (NULL (CDDR VAR2)))                     ; Last AND clause to detect FOO _ ' FIE : 2 type of operations.
                    (SETQQ LISPFN SETQQ)
                    (SETQ VAR2 (CDR VAR2]
                (COND
                   ((AND TYPE-IN? (EQ VAR1 'Ÿ))
                    (PRIN1 '= T)
                    (PRINT (SETQ VAR1 LASTWORD)
                           T T)))
                (GO INSERT))
             NIL)
      C   (SETQ LISPFN (CLISPLOOKUP TYP VAR1 (CAR VAR2)
                           LISPFN))
          (COND
             (UNARYFLG [SETQ VAR1 (COND
                                     ((CDR VAR2)                 ; E.G. NOT is a unary operator which may take more than one
                                                                 ; expression, e.g. NOT A = B
                                      VAR2)
                                     ((AND TYPE-IN? (EQ LISPFN 'QUOTE)
                                           (EQ (CAR VAR2)
                                               'Ÿ))
                                      (PRIN1 '= T)
                                      (PRINT LASTWORD T T))
                                     (T (CAR VAR2]
                       (SETQ VAR2 NIL)
                       (GO INSERT)))
          [SETQ TEM (COND
                       ((AND VAR2 (NULL (CDR VAR2)))
                        (CAR VAR2]                               ; TEM is the right-hand argument, if it is a single item.
          (COND
             ((SELECTQ LISPFN
                   (EQ (COND
                          ((AND VAR2 (NULL (CDR VAR2))
                                (NULL (CAR VAR2)))
                           (SETQQ LISPFN NULL))))
                   (IPLUS (COND
```

```
                          ((AND (LISTP VAR1)
                                (EQ (CAR VAR1)
                                    'IPLUS))                    ; Leave asis, so X+Y+1 goes to (IPLUS X Y 1) instead of (ADD1
                                                                ; (IPLUS X Y))
                           NIL)
                          ((EQ TEM 1)
                           (SETQQ LISPFN ADD1))
                          ((EQ TEM -1)
                           (SETQQ LISPFN SUB1))))
               (IDIFFERENCE (COND
                                ((AND (LISTP VAR1)
                                      (EQ (CAR VAR1)
                                          'IPLUS)
                                      (NULL (CDR VAR2)))
                                 [SETQ VAR2 (LIST (COND
                                                     ((NUMBERP (CAR VAR2))
                                                      (MINUS (CAR VAR2)))
                                                     (T (LIST 'IMINUS (CAR VAR2]
                                 (SETQQ LISPFN IPLUS)
                                 NIL)
                                ((EQ TEM 1)
                                 (SETQQ LISPFN SUB1))))
                    NIL)
            (SETQ VAR2 NIL)))
    INSERT
        (SETQ TEM (CLISPATOM2D LISPFN (CONS VAR1 VAR2)))
        (COND
           ((AND PARENT (ATOM PARENT))
            (CLISPATOM2A TAIL TAIL)
            (GO OUT)))
```

;; Corresponds to the case where the entire expression became an atom, e.g. X~=NIL gging to X, or --- 3 going to -3.0

```
        (SETQ Z (CDR PARENT))
```

;; Z is used to find the operands for DWIMIFYING. It is now set so that CAR of it coresponds VAR1 and CADR of it coresponds CAR of VAR2.

```
        (COND
           ((CLISPNOEVAL LISPFN)
            (AND DWIMIFYFLG (SETQ CLISPCHANGE TEM))
            (GO NEG))
           (DWIMIFYFLG (AND CLISPCHANGE (NULL UNARYFLG)
                            (GO C1))
```

;; If CLISPCHANGE is T and this is not a UNARY operation, the first operand has already been dwimified.

```
                  (SETQ CLISPCHANGE TEM))
           ((NOT (ATOM (CAR Z)))
            (GO C1)))
        (AND (NEQ LISPFN 'SETQ)
             (CLISPATOM2A Z PARENT))
```

;; Dwimifies VAR1, e.g. ((A+B) *C). If CLISPCHANGE is T, VAR1 has already been processed, e.g. A*B+C, becomes ((ITIMES A A) + C), and
;; the A and B have already been checked by the first call to CLISPATOM2. VAR1 is also dwimified when running provided it is atomic. so that if
;; it or VAR2 is unbound, an alternate correction will be tried, e.g. mistyping a variable named FOO-1 as FOOO-1.

```
    C1  [COND
           (UNARYFLG (GO C2))
           ((AND (LISTP VAR1)
                 (EQ LISPFN (CAR VAR1))
                 (FMEMB LISPFN '(AND OR IPLUS ITIMES FPLUS FTIMES PLUS TIMES))
                 (NEQ VAR1 (CAR CLISPLASTSUB)))
```

;; Handles nospreads, e.g. A+B+C becomes (IPLUS A B C) Note that where necessary, VAR1 has already been dwimified. The
;; CLISPLASTSUB check is to prevent parens from beig taken out when VAR1 is the result of an IS PHRASE since this is needed
;; later.

```
            (CLRPLNODE Z (CADR VAR1)
                   (APPEND (CDDR VAR1)
                           VAR2]
        (SETQ Z VAR2)
        (COND
           ((OR DWIMIFYFLG (LITATOM (CAR Z)))
            (CLISPATOM2A Z PARENT)))
    C2                                                          ; Z is now set so that it corresponds to the right hand argument
                                                                ; of the oprator.
        (COND
           ([AND Z (SETQ CLTYP (GETPROP (SETQ LISPFN (CAR Z))
                                        'CLISPTYPE]             ; The second operand is itself an operator, e.g. a+*b.
            (COND
               ([OR (NULL (CDR Z))
                    (NULL (GETPROP LISPFN 'UNARYOP]             ; The GETP check is because this is not an error if the operator
                                                                ; is unary.
                (DWIMERRORRETURN 2)))
            (CLISPATOM1A LISPFN CLTYP Z ENDTAIL)
```

;; If ENDTAIL is non-nil, the LDIFF copied this portion of TAIL, so it is not necessary to do any saving.

```
            )
           ((NULL (CDR Z)))
           ((SETQ CLTYP (GETPROP (SETQ LISPFN (CADR Z))
                                 'CLISPTYPE))
            (CLISPATOM1A LISPFN CLTYP Z ENDTAIL)))
```

```
      NEG [COND
             (NEGFLG                                          ; An operator was negated, e.g. X ~MEMB y
                    (CLRPLNODE PARENT 'NOT (LIST (CONS (CAR PARENT)
                                                       (CDR PARENT]
          [COND
             ([AND (EQ (CAR PARENT)
                       'NOT)
                    (LISTP (SETQ TEM (CADR PARENT)))
                    (NOT (EQUAL PARENT (SETQ TEM (NEGATE TEM]
```

;; Special stuff for negation. Done fter everything to take care of both X~=Y, and ~ (EQ X Y) in the same way.

```
              [COND
                 ((EQ PARENT (CAR TAIL))
                  (CLRPLNODE TAIL TEM (CDR TAIL)))
                 ((LISTP TEM)
                  (CLRPLNODE TAIL (CAR TEM)
                             (CDR TEM]
              (AND TENTATIVE (SETQQ TENTATIVE PROBABLY]
      OUT (RETURN TAIL])
```

(CLISPNOEVAL

```
  [LAMBDA (FN DEFAULT)                                        (* lmm "29-Jul-86 00:00")
```

;; returns true if FN doesn't evaluate its args. If not sure, return DEFAULT

```
    (PROG (TEM)
          [COND
             ((SETQ TEM (FASSOC FN DWIMEQUIVLST))
              (SETQ FN (CDR TEM]
          (RETURN (AND (SELECTQ (ARGTYPE FN)
                                ((1 3)                        ; NLAMBDA
                                    T)
                                (NIL                          ; udf -- see what else we know about it
                                     (OR (FMEMB FN NLAMA)
                                         (FMEMB FN NLAML)
                                         (COND
                                            ((NOT (OR (GETPROP FN 'MACRO-FN)
                                                      (GETLIS FN MACROPROPS)))
                                             DEFAULT)
                                            [DWIMINMACROSFLG  ; Macros are treated as LAMBDA forms unless INFO prop says
                                                              ; otherwise
                                                  (RETURN (EQMEMB 'NOEVAL (GETPROP FN 'INFO]
                                            (T T))))
                                (OR (FMEMB FN NLAMA)
                                    (FMEMB FN NLAML)))
                       (NOT (EQMEMB 'EVAL (GETPROP FN 'INFO])
```

(CLISPLOOKUP

```
  [LAMBDA (WORD $VAR1 $VAR2 $LISPFN)                          (* lmm "20-May-84 19:08")
```

;; In most cases, it is not necessary to do a full lookup. This is quick an dirty check inside of the block to avoid calling CLISPLOOKUP0 It will work ;; whenever there are no local declarations.

```
    (PROG (TEM CLASS CLASSDEF)
          (SETQ CLASS (GETPROP WORD 'CLISPCLASS))
          (SETQ CLASSDEF (GETPROP CLASS 'CLISPCLASSDEF))
```

;; used to be getprop word, but this meant GT worked differently than gt. also this new way is consistent with clispifylooup. shuld it bb (OR ;; (getprop word) (getprop class))?

```
          [SETQ TEM (COND
                       ((AND CLASSDEF (SETQ TEM (GETLOCALDEC EXPR FAULTFN)))
```

;; must do full lookup. Note that it is not necessary to do a call to CLISPLOOKUP0 if word has a CLASS, but no ;; CLASSDEF, e.g. FGTP, FMEMB, etc., since if these are ued as infix operators, they mean the corresponding functin ;; regardless of declaraton. I.e. The CLASSDEF property says that this is the name of an infix operator. The CLASS ;; property is used as a back pointer to the name of the operator/class of which this word is a member.

```
                        (CLISPLOOKUP0 WORD $VAR1 $VAR2 TEM $LISPFN CLASS CLASSDEF))
                       (T (SELECTQ CLASS
                               (VALUE (RETURN (GETATOMVAL WORD)))
                               ((RECORD RECORDFIELD)
                                    (RETURN NIL))
                               (OR $LISPFN (GETPROP WORD 'LISPFN)
                                   WORD]
          [COND
             ([AND (EQ (CAR CLASSDEF)
                       'ARITH)
                    (EQ TEM (CADR CLASSDEF))
                    (OR [COND
                           ((NLISTP $VAR1)
                            (FLOATP $VAR1))
                           (T (EQ (CAR $VAR1)
                                  (CADDR CLASSDEF]
                        (COND
                           ((NLISTP $VAR2)
                            (FLOATP $VAR2))
                           (T (EQ (CAR $VAR2)
                                  (CADDR CLASSDEF]
```

```
            (SETQ TEM (CADDR CLASSDEF]
          (RETURN TEM])
```

(CLISPATOM2A

```
(LAMBDA (TAIL PARENT)                                   (* lmm "21-Jun-85 16:49")
  (AND TAIL (NULL BROADSCOPE)
       (PROG ((DWIMIFYING (AND DWIMIFYFLG DWIMIFYING))
              (CLISPCONTEXT (AND DWIMIFYFLG CLISPCONTEXT))
              DWIMIFYCHANGE TEM)
```

;; If BROADSCOPE is T, everything has already been dwimified. See comments in clispatm2 and clispatom2b1

;; CLISPATOM2A sets up state variables itself rather than calling DWIMIFY1? or DWIMIFY2? because it wants to be able to add to ;; NOFIXVARSLST0.

```
             (COND
                ((NULL DWIMIFYFLG)
                 (SETQ NOFIXFNSLST0 NOFIXFNSLST)
                 (SETQ NOFIXVARSLST0 NOFIXVARSLST)))
             [SETQ TEM (COND
                          ((OR (AND (NEQ TYP '_)
                                    (NEQ TYP '←))
                               (LISTP VAR1))                ; VAR1 is a list when the _ is a record expression.
                           'DONTKNOW)
                          ((OR (FMEMB VAR1 VARS)
                               (FMEMB VAR1 NOFIXVARSLST0))
                           'PROBABLY)
                          ((OR (BOUNDP VAR1)
                               (AND (NULL DWIMIFYING)
                                    (STKSCAN VAR1 FAULTPOS))
                               (GETPROP VAR1 'GLOBALVAR)
                               (FMEMB VAR1 GLOBALVARS))     ; Added to NOFIXVARSLST0 so will be avilable for spelling
                                                            ; correction in the future.
                           (SETQ NOFIXVARSLST0 (CONS VAR1 NOFIXVARSLST0))
                           'PROBABLY)
                          ([AND (NEQ CLISPCONTEXT 'FOR/BIND)
                                (EQ VAR1 (CADR PARENT))
                                (NEQ NOSPELLFLG T)
                                (OR (NULL NOSPELLFLG)
                                    TYPE-IN?)
                                (OR [AND VARS (SETQ TEM (FIXSPELL VAR1 NIL VARS NIL NIL NIL NIL NIL T
                                                              'MUSTAPPROVE]
                                    (SETQ TEM (FIXSPELL VAR1 NIL SPELLINGS3 NIL NIL NIL NIL NIL T
                                                        'MUSTAPPROVE]
```

;; FIXSPELL is called instead of CLISPRESPELL because we dont want runon corrections, and also we have ;; performed msot of the checks of CLISPRESPELL.

```
                           (CLRPLNODE (CDR PARENT)
                                  TEM
                                  (CDDR PARENT))
                           'CERTAINLY)
                          (T (SETQ NOFIXVARSLST0 (CONS VAR1 NOFIXVARSLST0))
                                                            ; Added to NOFIXVARSLST0 so that it will be available for
                                                            ; spelling correction in the future.
                             'DONTKNOW]
             (RETURN (COND
                        [(LISTP (CAR TAIL))
                         (COND
                            ((NEQ CLISPCONTEXT 'LINEAR)
                             (DWIMIFY1 (CAR TAIL)))
                            (T (CAR TAIL]
                        ([AND TAIL (CAR TAIL)
                              (LITATOM (CAR TAIL))
                              (NOT (GETPROP (CAR TAIL)
                                            'CLISPTYPE]     ; We already know that the atom has no operators internal to it,
                                                            ; having scanned through it earlier.
                         (SETQ CLISPCONTEXT NIL)
                         (COND
                            ((AND (NULL (DWIMIFY2 TAIL PARENT T NIL T 'NORUNONS))
                                  (NULL TENTATIVE)
                                  (FMEMB TYP CLISPCHARS))
                             (SETQ TENTATIVE TEM])
```

(CLISPBROADSCOPE

```
(LAMBDA ($TYP L CONTEXT)                                (* lmm "29-Jul-86 00:26")
  (PROG ((BRACKETCNT 0)
         (L0 L))
    LP  [COND
           ((NULL L)
            (COND
               ((NULL (CDR L0))
                (CLISPBROADSCOPE1 L0 CONTEXT))
               (T (CLRPLNODE L0 (CONS (CAR L0)
                                      (CDR L0))
                             NIL)
                  (CLISPBROADSCOPE1 L0 CONTEXT T)))
            (RETURN))
```

```
            ((AND (EQ (CAR L)
                      '<)
                  (GETP '< 'CLISPTYPE))
             (SETQ BRACKETCNT (ADD1 BRACKETCNT)))
            ((AND (EQ (CAR L)
                      '>)
                  (GETP '> 'CLISPTYPE))
             (SETQ BRACKETCNT (SUB1 BRACKETCNT)))
            ((AND (EQ (CAR L)
                      $TYP)
                  (ZEROP BRACKETCNT))
             (COND
                ((EQ L (CDR L0))
                 (CLISPBROADSCOPE1 L0 CONTEXT)
                 (SETQ L0 (SETQ L (CDR L)))
                 (GO LP))
                (T (CLRPLNODE L0 (LDIFF L0 L)
                              L)
                   (CLISPBROADSCOPE1 L0 CONTEXT T)
                   (SETQ L0 (SETQ L (CDR L)))
                   (GO LP]
          (SETQ L (CDR L))
          (GO LP])
```

(CLISPBROADSCOPE1

```
[LAMBDA (X CONTEXT FLG)
  (PROG (TEM)
        (RETURN (COND
                   [(NLISTP (CAR X))
                    [SETQ TEM (DWIMIFY2? (SETQ TEM (LIST (CAR X)))
                                         TEM TEM NIL NIL NIL (COND
                                                                ((EQ CONTEXT 'IS)
                                                                 'IS)
                                                                (T
                                                                   ;; Reason for the OR is to handle things like X IS A NUMBER AND NOT LT Y.  In this case would be
                                                                   ;; dwimifying (NOT LT Y) but when go to dwimify (NOT) want CLISPATOMIS?  to be able to se the
                                                                   ;; higher context.
                                                                    (OR (AND DWIMIFYFLG (LISTP CLISPCONTEXT))
                                                                        CONTEXT]
                    (FRPLACA X (COND
                                  ((CDR TEM)
                                   TEM)
                                  (T (CAR TEM]
                   ((EQ CONTEXT 'IS)
                    (DWIMIFY2? (CAR X)
                               (CAR X)
                               (CAR X)
                               NIL NIL NIL CONTEXT))
                   (T ;; FLG says that the parens were inserted here, so that CONTEXT should be passed on to DWIMIFY1 in case there is a
                      ;; spelling error, e.g. (TAIL AND Y ORR Z) gets handled differently than (TAIL AND Y OR Z)
                      (DWIMIFY1? (CAR X)
                                 (AND FLG CONTEXT])
```

(CLISPATOM2C

```
[LAMBDA (TAIL0)                                                    (* lmm "20-May-84 19:55")
   ;; Checks for the case where user leaves out arentheses in front of functon name that follows an operator, e.g. (LIST X+ADD1 Y)
   (SETQ TAIL0 (CDR TAIL0))                                        ; TAIL0 is as of the right hand operand.
   (COND
      ([AND (NEQ TYP '%')
            (NEQ TYP '%:)
            (NEQ TYP '<)
            (AND (LITATOM (CAR TAIL0))
                 (NULL (GETPROP (CAR TAIL0)
                                'UNARYOP))
                 (CLISPFUNCTION? TAIL0 'NOTVAR [FUNCTION (LAMBDA (X Y)
                                                          (CONCAT [COND
                                                                     ((EQ (CDR Y)
                                                                          (CDAR Y))
                                                                      ; Unary operator
                                                                       (CAAR Y))
                                                                     (T (CONCAT (RETDWIM2 (CAAR Y))
                                                                                (COND
                                                                                   ((EQ (CADAR Y)
                                                                                        '+-)
                                                                                    '-)
                                                                                   (T (CADAR Y]
                                                                  (SUBSTRING (RETDWIM2 (CDR Y))
                                                                             2 -1]
                                                [FUNCTION (LAMBDA (X Y)
                                                            (CONCAT [COND
                                                                       ((EQ (CDR Y)
                                                                            (CDAR Y))   ; Unary operator
```

```
                                                    (CAAR Y))
                                                   (T (CONCAT (RETDWIM2 (CAAR Y))
                                                              (COND
                                                                 ((EQ (CADAR Y)
                                                                      '+-)
                                                                  '-)
                                                                 (T (CADAR Y]
                                              [RETDWIM2 (COND
                                                           [(LISTP X)
                                                            (CONS (CAR X)
                                                                  (CONS (CDR X)
                                                                        (CDDR Y]
                                                           (T (CONS X (CDDR Y]
                                              '")"]
                                 (CONS TAIL TAIL0]
```

;; The GETP check is for situations like (LIST X_'FOO Y) i.e. a unary operator could never take care of the rest of the list.

```
(/RPLNODE TAIL0 (CONS (CAR TAIL0)
                      (CDR TAIL0)))
(SETQ ENDTAIL NIL)                                  ; Once you print a message, you dont want to go and try another
                                                    ; interpretation.
(SETQQ TENTATIVE CERTAINLY])
```

(CLISPATOM2D

```
[LAMBDA (X Y)
```

;; Inserts new expression into TAIL. Value is T if expression was not parenthesized, PARTIAL if it was, i.e. if it corresponded to the new CAR of ;; TAIL. If X is NIL, Y is the whole expression.

```
(COND
   ((AND (NULL ENDTAIL)
         (NULL FORMSFLG)
         (OR (NULL PARENT)
             (EQ PARENT TAIL)))
```

;; This is the case in which we do not want to 'subordinate' the expression with an extra pair of parentheses. E.g. (LIST (A+B)). The ;; ENDTAIL check is necessary because if it is not NIL, there are more expressions following the first one, e.g. (LIST (A*B+C)) and we must ;; keep this expression separate, i.e. make (A*B+C) become ((ITIMES A A) + C)

```
    (COND
       ((NULL X)
```

;; Y is the entire expression to be inserted, but we can't use it because we have to 'take out' the parentheses.

```
        (CLRPLNODE TAIL (CAR Y)
                   (CDR Y))
        (AND (SETQ X (GETHASH Y CLISPARRAY))
             (CLISPTRAN TAIL X))
```

;; Must move translation to new expression. This only occurs if the expression is enclosed in prentheses, e.g. (X: (--))

```
        (AND (EQ Y (CAR CLISPLASTSUB))
             (FRPLACA CLISPLASTSUB TAIL))
```

;; Y is the expression returned by CLISPATOMIS but it is not going to apear in the new expression, so must change clisplastsub to ;; correspnd

```
        )
       (T (CLRPLNODE TAIL X Y)))
    (SETQ PARENT TAIL)
    T)
   (T                                               ; Here we must parenthesize the expression so as to subordinate
                                                    ; it.
      [SETQ Y (COND
                 ((NULL X)
                  Y)
                 ((AND (EQ TYP '-)
                       (NUMBERP (CAR Y)))
                  (MINUS (CAR Y)))
                (T (CONS X Y]
      (CLRPLNODE TAIL Y ENDTAIL)                    ; ENDTAIL being all the stuff not belonging to the CLISP
                                                    ; expression, i.e. beyond its scope.
      (SETQ PARENT (CAR TAIL))
      'PARTIAL])
```

(CLISPCAR/CDR

```
[LAMBDA (LST)                                       (* lmm "21-Jun-85 16:50")
```

;; Handles the : infix operatr.

```
(PROG ([SETQFLG (OR (EQ (CAR ENDTAIL)
                        '_)
                    (EQ (CAR ENDTAIL)
                        '←]
       TAILFLG N TEM VAL)
      (SETQ VAR2 NIL)
  LP  (SETQ TAILFLG NIL)
      [COND
         ((EQ (CAR LST)
              '%:)                                  ; Tail
          (SETQ TAILFLG T)
```

```
            (SETQ LST (CDR LST]
         (COND
            ((NULL LST)
             (SETQ VAR1 (LIST (COND
                                 ((NULL SETQFLG)
                                  (GO ERROR))
                                 (TAILFLG                          ; X::_
                                       'NCONC)
                                 (T 'NCONC1))
                               VAR1))
             (RETURN VAL)))
         (COND
            ((EQ (SETQ N (CAR LST))
                 '-)
             (COND
                ([NOT (NUMBERP (SETQ N (CADR LST]
                 (GO ERROR)))
             (SETQ N (MINUS N))
             (SETQ LST (CDR LST))
             (GO NEG))
            ((NOT (NUMBERP N))
             [COND
                (TAILFLG (GO ERROR))
                ((LISTP N)
                 (SETQ VAR1 (LIST 'match VAR1 'with N))
                 (COND
                    ((OR (EQ (CADR LST)
                             '->)
                         (EQ (CADR LST)
                             '=>))
                     (NCONC VAR1 (CDR LST))
                     (DWIMIFY2? (SETQ TEM (CDDR LST))
                           VAR1 TEM)
                     (SETQ LST NIL)))
                 (CLISPTRAN VAR1 (MAKEMATCH VAR1))
                 (AND (NULL VAR2)
                      (SETQ VAR2 VAR1)))
                [[SETQ TEM (CLISPRECORD VAR1 N (AND SETQFLG (NULL (CDR LST]
                 (SETQ VAR1 TEM)
                 (AND (NULL VAR2)
                      (SETQ VAR2 (NLEFT VAR1 2]
                ((SETQ TEM (GETPROP N 'ACCESSFN))
                 (SETQ VAR1 (LIST TEM VAR1))
                 (AND (NULL VAR2)
                      (SETQ VAR2 VAR1)))
                (T (DWIMERRORRETURN 'FIELDNAME]
             (GO LP2))
            ((ILESSP N 0)
             (GO NEG)))
      LP1 [COND
            ((AND (IGREATERP N 4)
                  (ILESSP N 9))                                    ; X:N for N greater than 8 goes to (NTH X N)
             (SETQ N (IPLUS N -4))
             (SETQ VAR1 (LIST 'CDDDDR VAR1))
             (AND (NULL VAR2)
                  (SETQ VAR2 VAR1))                                ; VAR2 marks the TAIL where the original operand appears, so
                                                                   ; thaadwimifying will continue from there.
             (GO LP1))
            ((AND SETQFLG (NULL (CDR LST)))
             (SETQ VAR1 (CLISPCAR/CDR1 1 (CLISPCAR/CDR1 (SUB1 N)
                                                VAR1 T)
                                   TAILFLG T)))
            (T (SETQ VAR1 (CLISPCAR/CDR1 N VAR1 TAILFLG]
      LP2 (COND
            ((NULL (SETQ LST (CDR LST)))
             (RETURN VAL))
            ((EQ (CAR LST)
                 '%:)
             (SETQ VAL T)
             (SETQ LST (CDR LST))
             (GO LP)))
      ERROR
          [DWIMERRORRETURN (COND
                             (TAILFLG '|::|)
                             (T '%:]
      NEG (COND
            ((AND SETQFLG (NULL (CDR LST))
                  TAILFLG)
             [SETQ VAR1 (LIST 'NLEFT VAR1 (ADD1 (IMINUS N]
             (AND (NULL VAR2)
                  (SETQ VAR2 VAR1))
             (SETQ VAR1 (LIST 'CDR VAR1))
             (RETURN VAL)))
          [SETQ VAR1 (COND
                       ((EQ N -1)
                        (LIST (CLISPLOOKUP 'LAST VAR1)
                              VAR1))
```

```
                    (T (LIST 'NLEFT VAR1 (IMINUS N]
          (AND (NULL VAR2)
               (SETQ VAR2 VAR1))
          [COND
             ((NULL TAILFLG)
              (SETQ VAR1 (LIST 'CAR VAR1]
          (GO LP2])
```

(CLISPCAR/CDR1

```
[LAMBDA (N X TAILFLG SETQFLG)                                   (* lmm "20-May-84 19:56")
```

;; All three level car and cdr operations go back to the corresponding function, i.e. CDAAR clispifies to X:1:1::1 and goes back to CDAAR.

```
(PROG (TEM)
      (COND
         ((ZEROP N)
          (RETURN X))
         ((AND (NULL DWIMIFYFLG)
               CHECKCARATOMFLG)                                 ; If CHECKCARATOMFLG is T, then checks to see if the car/cdr
                                                                ; chain goes through an atom (non-list)
          (CLISPCAR/CDR2 N X)))
      [SETQ TEM (COND
                  ([AND (NULL SETQFLG)
                        (LISTP X)
                        (SETQ TEM (COND
                                     ((EQ N 1)
                                      (SELECTQ (CAR X)
                                           (CAR
```

;; The apparent incompleteness of the SELECTQ is bcause CAR of CDR would appear in CLISS
;; as 2 and be handled directly, similarly for CDR of CDR.

```
                                                   (COND
                                                      (TAILFLG 'CDAR)
                                                      (T 'CAAR)))
                                           (CAAR
```

;; Similarly, CAR of CDAR would come in as CADR of CAR, CDR of CDAR as CDDR of
;; CAR, so checks for CDAR and CDDR are not necessary.

```
                                                  (COND
                                                     (TAILFLG 'CDAAR)
                                                     (T 'CAAAR)))
                                           (CADR (COND
                                                     (TAILFLG 'CDADR)
                                                     (T 'CAADR)))
                                          NIL))
                                     ((AND (EQ N 2)
                                           (EQ (CAR X)
                                               'CAR))
```

;; CADR of CDR would be written as X:3, similaly CAAR of CDR, CDAR of CDR, and CDDR
;; of CDR are all taken care of.

```
                                      (COND
                                         (TAILFLG 'CDDAR)
                                         (T 'CADAR]                 ; If SETQFLG is T, want to leave the outer CAR or CDR because
                                                                    ; gets replaced by rplaca/d later.
                    (FRPLACA X TEM))
                   [(IGREATERP N 4)
                    (SETQ TEM (CLISPLOOKUP 'NTH VAR1))
                    (COND
                       (TAILFLG (LIST TEM X (ADD1 N)))
                       (T (SETQ TEM (LIST TEM X N))
                          (AND (NULL VAR2)
                               (SETQ VAR2 TEM))
                          (LIST 'CAR TEM]
                   ([NULL (SETQ TEM (FASSOC N '((1 CAR . CDR)
                                                (2 CADR . CDDR)
                                                (3 CADDR . CDDDR)
                                                (4 CADDDR . CDDDDR]
                    (SHOULDNT 'CLISPCAR/CDR))
                   (TAILFLG (LIST (CDDR TEM)
                                  X))
                   (T (LIST (CADR TEM)
                            X]
          (AND (NULL VAR2)
               (SETQ VAR2 TEM))
          (RETURN TEM])
```

(CLISPCAR/CDR2

```
[LAMBDA (N X)                                                   (* lmm "20-May-84 19:56")
  (PROG ((NODE (STKEVAL FAULTPOS X)))
    LP  [COND
           ((ZEROP N)
            (RETURN))
           ((AND NODE (NLISTP NODE))
            (DWIMERRORRETURN 'CARATOM]
          (SETQ NODE (CDR NODE))
```

```
          (SETQ N (SUB1 N))
          (GO LP])
```

(CLISPATOMIS1

```
  [LAMBDA (SUBJ OBJ ALST EXP NEGATE)                          (* lmm "20-May-84 20:03")
```

;; ALST is cdr of the value returned by clispmatchup. CAR is split into the two arguments SUBJ and OBJ.

```
    (SELECTQ (CAR SUBJ)
         ((AND OR)
             [CONS (CAR SUBJ)
                   (MAPCAR (CDR SUBJ)
                           (FUNCTION (LAMBDA (X)
```

;; The AND is bcause it is ok for NEGFLG to be T instead of LISTONLY on recursive calls, because
;; (NOT (NULL X)) can go to X in this case since we have the tail to put it in.

```
                                     (CLISPATOMIS1 X OBJ ALST EXP (AND NEGATE T])
         (PROGN (SETQ EXP (SUBLIS (CONS (CONS OBJ SUBJ)
                                        ALST)
                                  EXP T))
                (COND
                   (NEGATE (NEGATE EXP))
                   (T EXP])
```

(CLISPATOMARE1

```
  [LAMBDA (X FLG)                                             (* lmm "29-Jul-86 00:27")
```

;; value is an edit pushdown list (of tails) leding to the place of the last is subject.

```
    (PROG (L TEM)
          (SETQ L (CDR X))
      LP  (COND
             ((EQ (CAR L)
                  (CAR CLISPLASTSUB))
              (RETURN (LIST L)))
             ((AND (LISTP (CAR L))
                   (SETQ TEM (SELECTQ (CAAR L)
                                      ((AND OR)
                                           (CLISPATOMARE1 (CAR L)
                                                          FLG))
                                      NIL)))
              (RETURN (NCONC1 TEM L)))
             ((SETQ L (CDR L))
              (GO LP)))
          (RETURN NIL])
```

(CLISPATOMARE2

```
  [LAMBDA (L Z)                                               (* lmm " 4-SEP-83 23:07")
    (PROG (X X1)
          [COND
             ((NULL (CDR L))
              (COND
                 (Z (RETURN (CAR Z)))
                 (T                                           ; E.g. X AND Y IS A NUMBER ARE ATOMS.
                    (DWIMERRORRETURN (LIST 'PHRASE (CDR TAIL)
                                           PARENT]
          (SETQ X (CAADR L))                                  ; the parent of (CAR L)
          (SETQ X1 (CDAR L))
          [COND
             ((AND DEST (EQ (CAR L)
                            (CDR X)))
```

;; move inner expression out. case 1: (A OR B ARE NUMBERS AND C OR D ARE LISTS) VAR1 is (OR (AND (OR (NUMBERP A)
;; (NUMBERP B)) C) D) but the AND is reaaly the top leveloperator. case 2: (A OR B IS A NUMBER AND C OR D ARE LISTS)
;; VAR1 is (OR A (AND (NUMBERP B) C) D) here the OR should be the top leveloperator. The difference is that

```
              (FRPLACA (CADR L)
                       (CADR X)))
             (T (FRPLACD (CAR L]
          [COND
             ((AND X1 (NULL DEST))
              (SETQ DEST (CAR L]
          (SETQ X1 (APPEND Z X1))
          (RETURN (CLISPATOMARE2 (CDR L)
                                 (COND
                                    ((CDR X1)
                                     (LIST (CONS (CAR X)
                                                 X1)))
                                    (T X1])
```

(CLISPATOMIS2

```
  [LAMBDA (X)                                                 ; wt: 25-FEB-76 1 51
```

;; Used by clispatomaRE and clispatomlis? to eliminate unnecessary nesting of ands and ors after finishing processing. (Too hard to do on the fly
;; as we built pushdown list of tails etc.) NOte that we cant remove parens from around clisplastsub since it might be needed later in parsing. Thus
;; X AND Y ARE NUMBERS AND GREATER THAN 3 must be left as (AND (NUMBERP X) (NUMBER Y) (AND (IGREATERP X 3) (IGREATERP Y
;; 3)))

```
    (PROG (($TYP (CAR X)))
      LP  [AND (LISTP (CAR X))
               (NEQ (CAR X)
                    (CAR CLISPLASTSUB))
               (COND
                  ((EQ (CAAR X)
                       $TYP)
                   (CLRPLNODE X (CADAR X)
                          (APPEND (CDDAR X)
                                  (CDR X)))
                   (GO LP))
                  ((OR (EQ (CAAR X)
                           'AND)
                       (EQ (CAAR X)
                           'OR))
                   (CLISPATOMIS2 (CAR X]
          (COND
             ((SETQ X (CDR X))
              (GO LP])

)

(DEFINEQ
```

(WTFIX

```
  [LAMBDA (FAULTX FAULTARGS FAULTAPPLYFLG)                    (* lmm "15-Apr-86 09:59")
    (PROG (FAULTPOS FAULTFN EXPR VARS TAIL PARENT SUBPARENT FORMSFLG ONLYSPELLFLG DWIMIFYFLG TEM)
          (RETURN (WTFIX1])
```

(WTFIX0

```
  [LAMBDA (FAULTX TAIL PARENT SUBPARENT ONLYSPELLFLG)
```

;; Internal entry from dwimify1 and dwimify2. EXPR, FAULTFN, VARS, TAIL, and FORMSFLG already correctly bound.

```
    (PROG (FAULTARGS FAULTAPPLYFLG (FAULTPOS (COND
                                             ((NULL (AND DWIMIFYFLG DWIMIFYING))
                                                                  ; Originally started out evaluting, so there is a higher faultpos.
                                              FAULTPOS)))
                (DWIMIFYFLG T))
          (RETURN (WTFIX1])
```

(WTFIX1

```
  [LAMBDA NIL                                                 (* bvm%: "21-Nov-86 18:37")
```

;; Replaces FAULT1 when DWIM is on. on u.b.a.'s FAULTX is the atom. On u.d.f.'s involving forms, FAULTX is the form. On u.d.f.'s from APPLY, ;; faultx is the name of the function, FAULTARGS the arguments, and FAULTAPPLYFLG is T. Also is called directly to process a form from ;; DWIMIFY. In this case, EXPR, VARS, ..., NOSPELLFLG0 are supplied, and FINDFN is not called.

```
    (AND
     DWIMFLG
     (LET
      [(RESULT
        (CL:CATCH 'WTFIX
            (XNLSETQ
             (PROG ((NOSPELLFLG0 NOSPELLFLG)
                    (CLISPERTYPE)
                    (DWIM.GIVE.UP.TIME (OR DWIM.GIVE.UP.TIME (SETUPTIMER DWIM.GIVE.UP.INTERVAL)))
                    TYPE-IN? BREAKFLG FAULTXX CHARLST FAULTEM1 NEWTAIL HISTENTRY FIXCLK CLISPCHANGES SIDES)
                                                                  ; LIST because this used to be a XNLSETQ. I think callers only
                                                                  ; want to know whether we returned something interesting, or
                                                                  ; somebody called (RETDWIM)
                   [COND
                      (DWIMIFYFLG
                              ;; Call from WTFIX0. Note that while this call from DWIMIFY1 or DWIMIFY2, the user may or may not have been
                              ;; DWIMIFYING, e.g. when IF's are encountered in evaluation, DWIMIFY1 and DWIMIFY2 are used. The
                              ;; variable DWIMIFYING is T if the call to DWIMIFY! or DWIMIFY2 is from an explicit call to DWIMIFY (or
                              ;; DWIMIFYFNS)
                              (SETQ TYPE-IN? (EQ FAULTFN 'TYPE-IN))
                              ;; DWIMIFY is called on typein for processing FOR's and IF's. In this case, want to treat user approval the same
                              ;; as for type-in.

                              )
                      (T (SETQ FIXCLK (CLOCK 2))
                         ;; If EXPR is given, i.e. if DWIMIFYFLG is gong to be T, the clkock is being measured at some higher caal to WTFIX or
                         ;; DWIMIIY.
                         [SETQ FAULTPOS (STKPOS (COND
                                                   (FAULTAPPLYFLG 'FAULTAPPLY)
                                                   (T 'FAULTEVAL]
                         (AND (NEQ CLEARSTKLST T)
                              (SETQ CLEARSTKLST (CONS FAULTPOS CLEARSTKLST)))
                                                                  ; In case user control-ds out of correction, this will relstk faultpos
                         (SETQ FAULTFN (FINDFN (FSTKNTH -1 FAULTPOS)
                                               T))
```

```
                    ;; The value of FINDFN is the name of the (interpreted) function in which the error occurred.  FINDFN also sets the
                    ;; free variable EXPR to the definition of that function.  If the error occurred under a call to EVAL, the value of FINDFN
                    ;; is EVAL, and EXPR is set to the expression being evaluated, i.e. the argument to EVAL.  If the error occurred under
                    ;; an APPLY, the value of FINDFN is the first argument to APPLY, and EXPR is set to the second argument to APPLY,
                    ;; i.e. the list of arguments.  In this case, FAULTX will usually be EQ to the value returned by FINDFN, and
                    ;; FAULTARGS EQ to EXPR.  However, WTFIX may also be called from FAULTAPPLY, and FINDFN not find an
                    ;; APPLY, as occurs on undefined functions called from compiled code.  For this reason, FIXAPPLY always uses
                    ;; FAULTX and FAULTARGS, not FAULTFN and EXPR.
                     (SETQ VARS (AND (SETQ FAULTEM1 (OR BREAKFLG (LISTP EXPR)))
                                     (GETVARS FAULTEM1]
                [AND (NULL TYPE-IN?)
                     (SETQ SIDES (CDR (LISTGET1 LISPXHIST 'SIDE]
                (AND TYPE-IN? (NULL DWIMIFYFLG)
                     [COND
                        (FAULTAPPLYFLG (EQ FAULTX (CAAAAR LISPXHISTORY)))
                        (T (OR (EQ FAULTX (CAAAAR LISPXHISTORY))
                               (EQUAL FAULTX (CAAAR LISPXHISTORY]
                     (SETQ HISTENTRY (CAAR LISPXHISTORY)))
                [COND
                   ([LITATOM (SETQ FAULTXX (COND
                                              (FAULTAPPLYFLG FAULTX)
                                              ((NLISTP FAULTX)
                                               FAULTX)
                                              (T (CAR FAULTX]
                    (SETQ CHARLST (DUNPACK FAULTXX WTFIXCHCONLST]
                (COND
                   ((AND (NULL FAULTAPPLYFLG)
                         (LITATOM FAULTX))
                    (FIXATOM)
                    (SHOULDNT))
                   (FAULTAPPLYFLG (FIXAPPLY)
                          (SHOULDNT))
                   ([AND TYPE-IN? (EQ FAULTXX (CAAR HISTENTRY))
                         (AND (NEQ NOSPELLFLG T)
                              (AND (SETQ FAULTEM1 (FMEMB LPARKEY CHARLST))
                                   (NULL (AND CLISPFLG (STRPOSL CLISPCHARRAY FAULTXX]
                    ;; LPARKEY is the lowercase version of left prentheses, normally 8, rparkey is normally 9, but user can reset them for
                    ;; different terminals.  The EQ distinguishes between (CONS8ADD1 3) which is handled by a call to FIX89 from
                    ;; CLISPATOM, and FOO8A B C ']' , which is handled by FIX89TYPEIN, since it requires changing an EVAL to an
                    ;; APPLY.
                    (FIX89TYPEIN FAULTEM1 CHARLST))
                   ((AND CLISPFLG CHARLST (LITATOM (SETQ FAULTEM1 (CADR FAULTX)))
                         (OR (GETPROP FAULTEM1 'CLISPTYPE)
                             (FMEMB (SETQ FAULTEM1 (NTHCHAR FAULTEM1 1))
                                    CLISPCHARS))
                         [OR (NOT (GETPROP FAULTEM1 'UNARYOP))
                             (AND (EQ FAULTEM1 '~)
                                  (GETPROP (PACK (CDR (DUNPACK (CADR FAULTX)
                                                               WTFIXCHCONLST1)))
                                           'CLISPTYPE]
                         (NOT (CLISPNOTVARP (CAR FAULTX)))
                         (CLISPNOTVARP (CADR FAULTX)))          ; So that things like (SUM + X) will work, i.e. not be interpreted
                                                                ; as iterative statement.
                    (GO NX0))
                   ((NULL CHARLST)
                    (GO NX2)))                                  ; Both FIXAPPLY and FIXATOM exit via RETDWIM so there is
                                                                ; no need for a return here in WTFIX.
           TOP [SELECTQ (CAR FAULTX)
                        (F/L [/RPLNODE FAULTX 'FUNCTION
                                       (LIST (CONS 'LAMBDA
                                                   (COND
                                                      ([AND (CDDR FAULTX)
                                                            [OR (NULL (CADR FAULTX))
                                                                (AND (LISTP (CADR FAULTX))
                                                                     (EVERY (CADR FAULTX)
                                                                            (FUNCTION (LAMBDA (X)
                                                                                              (AND X (NEQ X T)
                                                                                                   (LITATOM X]
                                                            (OR (MEMB (CAADR FAULTX)
                                                                      (FREEVARS (CDDR FAULTX)))
                                                                (NOT (CLISPFUNCTION? (CADR FAULTX)
                                                                                     'OKVAR]
                                                       (CDR FAULTX))
                                                      (T (CONS (LIST 'X)
                                                               (CDR FAULTX]
                             (GO OUT))
                        (CLISP%: (ERSETQ (CLISPDEC0 FAULTX FAULTFN))
                                 (SETQ FAULTX T))
                        (COND
                           [[CAR (LISTP (SETQ FAULTEM1 (GETPROP (CAR FAULTX)
                                                                'CLISPWORD]
                            (RESETVARS [(LCASEFLG (AND LCASEFLG (NULL TYPE-IN?]
                                       (SELECTQ (CAR FAULTEM1)
                                            (FORWORD (SETQ FAULTX (OR (CLISPFOR FAULTX)
                                                                      (RETDWIM))))
                                            (IFWORD (SETQ FAULTX (CLISPIF FAULTX))
```

```
                                        (SETQ HISTENTRY NIL))
                                  (MATCHWORD                    ; CAR of FAULTX either MATCH or match.
                                             (CLISPTRAN FAULTX (MAKEMATCH FAULTX)))
                                  (PREFIXFN (PROG ((EXPR FAULTX))
                                                  (SETQ FAULTEM1 (CDR FAULTX))
                                                  [COND
                                                     ((EQ (CAR (LISTP (CAR FAULTEM1)))
                                                          'CLISP%:)
                                                      (ERSETQ (CLISPDEC0 (CAR FAULTEM1)
                                                                         FAULTFN]
                                                  [COND
                                                     ((EQ (CAR (LISTP (CAR FAULTEM1)))
                                                          COMMENTFLG)
                                                      (SETQ FAULTEM1 (CDR FAULTEM1]
                                                  [SETQ FAULTEM1 (APPEND (COND
                                                                           [(AND (NULL (CDR FAULTEM1))
                                                                                 (LISTP (CAR FAULTEM1]
                                                                           (T FAULTEM1]
                                                  (RESETVARS ((CLISPFLG T))
                                                             (DWIMIFY1? FAULTEM1))
                                                  (CLISPELL FAULTX)
                                                  (CLISPTRAN FAULTX FAULTEM1)))
                                  (SETQ FAULTX (APPLY* (CAR FAULTEM1)
                                                       FAULTX]
            (T (GO NX0]
    (AND DWIMIFYFLG (SETQ CLISPCHANGE T))
    (GO OUT)
NX0 (COND
       [(GETD (CAR FAULTX))
        (COND
           ([NULL (PROG (TYPE-IN? (FAULTFN (CAR FAULTX)))
                        (RETURN (COND
                                   ((FIXLAMBDA (GETD (CAR FAULTX)))
                                                    ; This is the case where (FOO --) is being evaluated, and the
                                                    ; definition of FOO is bad.
                                    (AND FILEPKGFLG (LITATOM FAULTFN)
                                         (MARKASCHANGED FAULTFN 'FNS))
                                    T]
            (SETQ NOSPELLFLG0 T)
            (GO NX3)                                ; So DWIMUSERFN can be called.

            ]
       ((AND (OR (GETPROP (CAR FAULTX)
                          'EXPR)
                 (GETPROP (CAR FAULTX)
                          'CODE))
             (DWIMUNSAVEDEF (CAR FAULTX)))
        (SETQ FAULTFN NIL)                          ; So that RETDWIM won't do a MARKASCHANGED

        )
       ((SETQ FAULTEM1 (GETPROP (CAR FAULTX)
                                'FILEDEF))
        (COND
           ((WTFIXLOADEF FAULTEM1)
            (GO OUT)))
        (RETDWIM))
       (T (GO NX1)))
    (GO OUT)
NX1 (COND
       ((AND (CLISPNOTVARP (CAR FAULTX))
             (SETQ FAULTEM1 (CLISPATOM CHARLST FAULTX FAULTX)))
                                                    ; E.g. (FOO_ATOM) OR (FOO_ form)
        (SETQ FAULTX FAULTEM1)
        (GO OUT)))
NX2 (COND
       ([AND CLISPFLG (SETQ FAULTEM1 (CADR FAULTX))
             (OR (LITATOM FAULTEM1)
                 (AND (NUMBERP FAULTEM1)
                      (MINUSP FAULTEM1)
                      (CLBINARYMINUS? FAULTX)))
             (OR (GETPROP FAULTEM1 'CLISPTYPE)
                 (FMEMB (CAR (SETQ FAULTEM1 (DUNPACK FAULTEM1 WTFIXCHCONLST1)))
                        CLISPCHARS))
             (SETQ FAULTEM1 (CLISPATOM FAULTEM1 (CDR FAULTX)
                                       FAULTX T))
             (COND
                [(OR (NEQ FAULTXX (CAR FAULTX))
                     (AND CLISPARRAY (GETHASH FAULTX CLISPARRAY]
                (DWIMIFYFLG (SETQ CHARLST (DUNPACK FAULTXX WTFIXCHCONLST))
                                                    ; LST may have been clobbered
                            (SETQ CLISPCHANGE NIL]
```

;; E.g. (FOO _atom) or (FOO _ form). The NEQ check is necessary to handle situations like (FOOO N-1) where an
;; CLISP transformation is performed, but it does not correct CAR of the form. (In this case, we must continue to the
;; spelling correction part below, and set CLISPCHANGE to NIL so that DWIMIFY1 will not be confused.) Note that if FOO
;; also happens to be the name of a function, then WTFIX will not be called and the CLISP transformation not be
;; performed until the arguments of FOO are evaluated and cause a u.b.a. error. Then DWIM will have to back up as
;; described in FIXATOM and FIXATOM1.

```
                    (SETQ FAULTX FAULTEM1)
                    (GO OUT))
                   ((AND (NULL NOSPELLFLG0)
                         DWIMIFYFLG
                         (LISTP (CADR FAULTX))
                         (FIXLAMBDA FAULTX))
                    ;; The DWIMIFYFLG check is because in normal course of events, it never makes sense for LAMBDA to appear as CAR
                    ;; of a FORM.  However, DWIMIFY1 is called on open LAMBDA expressions.
                    (GO OUT))
                   ((AND (NULL NOSPELLFLG0)
                         (LISTP (CAR FAULTX))
                         (LISTP (CADAR FAULTX))
                         (FIXLAMBDA (CAR FAULTX)))
                    ;; This corresponds to the case where LAMBDA is misspelled in an open LAMBDA expression.  Note that an open
                    ;; lambda expression only makes sense when there is a non-atomic argument list, so dont both spelling correcting if this
                    ;; is notthe case.
                    (GO OUT)))
            NX3 (COND
                   [[SOME DWIMUSERFORMS (FUNCTION (LAMBDA (DWIMUSERFORM)
                                                    (SETQ FAULTEM1 (EVAL DWIMUSERFORM]
                    (COND
                       (FAULTAPPLYFLG (RETDWIM FAULTPOS FAULTEM1 T FAULTARGS))
                       (T (RETDWIM FAULTPOS FAULTEM1]
                   (NOSPELLFLG0 (GO FAIL))
                   [[AND CHARLST (SETQ FAULTXX (OR (FIXSPELL (CAR FAULTX)
                                                          NIL SPELLINGS2 NIL FAULTX NIL NIL NIL T)
                                                   (AND DWIMIFYFLG NOFIXFNSLST0
                                                        (FIXSPELL (CAR FAULTX)
                                                               NIL NOFIXFNSLST0 NIL FAULTX NIL NIL NIL T]
                                                               ; The extra argument to FIXSPELL indicates that SPLITS re
                                                               ; tolerated, e.g. (BREAKFOO)
                    (COND
                       ((EQ (CAAR HISTENTRY)
                            (CAR FAULTX))
                        (/RPLNODE HISTENTRY FAULTX (CDR HISTENTRY))
                        ;; Normally, RETDWIM patches the histroy entry to corresond to a list input, even if it was typed in as a line.  In the
                        ;; special case of a pselling correction, we leave the entry as a line.

                        ))
                    (SETQ HISTENTRY NIL)
                    (COND
                       ((NOT (FGETD FAULTXX))                    ; E.g. USER misspells FOR, IF, F/L etc. These are all contained
                                                                 ; on SPELLINGS2.
                        (GO TOP]
                   ((AND CLISPFLG DWIMIFYFLG (CDR FAULTX)
                         (LISTP CLISPCONTEXT)
                         (FIXSPELL (CAR FAULTX)
                                NIL CLISPISWORDSPLST NIL FAULTX NIL NIL NIL T)
                         (SETQ FAULTEM1 (CLISPATOM (DUNPACK (CAR FAULTX)
                                                            WTFIXCHCONLST)
                                                   TAIL PARENT)))
                    ;; E.g. X IS A NUMBER AND LESS THAN Y.  CLISPATOM will call CLISPATOMIS?  which will retfrom back past here or
                    ;; generate an error.  NOte that if (CAR FAUULTX) had been spelled correctly, thiswold have happened in first call to
                    ;; CLISPATOM at NX1 earlir.  However, we dont do the misspelled check until here because it is more likely user has
                    ;; misspelled the name of one of his functions.

                    )
                   ([AND CLISPFLG (NULL CLISPCHANGES)
                         (NULL CLISPERTYPE)
                         (SETQ FAULTEM1 (CADR FAULTX))
                         (LITATOM FAULTEM1)
                         (SETQ FAULTEM1 (FIXSPELL FAULTEM1 NIL CLISPINFIXSPLST NIL (OR (AND DWIMIFYFLG
                                                                                             (LISTP
                                                                                                  CLISPCONTEXT
                                                                                                   ))
                                                                                        (CDR FAULTX))
                                                                    NIL NIL NIL T))
                         (COND
                            ((AND DWIMIFYFLG (LISTP CLISPCONTEXT))
                             ;; Return from the corresponding DWIMUNDOCATCH with a value telling CLISPATOM to try again.
                             (CL:THROW 'CLISPATOM1 :RESPELL))
                            (T (LET (CLISPERTYPE)
                                    (SETQ FAULTEM1 (CLISPATOM FAULTEM1 (CDR FAULTX)
                                                              FAULTX T]
                    (SETQ FAULTX FAULTEM1))
                   (T (GO FAIL)))
            OUT (RETDWIM FAULTPOS FAULTX)
            FAIL
                (RETDWIM))))]
        (SELECTQ RESULT
            (:RESPELL                                            ; from CLISPATOM2 -- wants us to throw this message back to a
                                                                 ; higher CLISPATOM
```

```
                (CL:THROW 'CLISPATOM1 :RESPELL))
          (PROGN                                                    ; something interesting to return, or a value from RETDWIM
                 RESULT])
```

(**RETDWIM**

```
  [LAMBDA (POS X APPLYFLG ARGS)                                     (* bvm%: "21-Nov-86 18:02")
    (PROG NIL
          [AND FIXCLK HELPCLOCK (SETQ HELPCLOCK (IPLUS HELPCLOCK (IDIFFERENCE (CLOCK 2)
                                                                              FIXCLK]
                                                                    ; So time spent in DWIM will not count towards a break.
      TOP [COND
             [(OR POS X)                                            ; Successful correction.
              (AND (EQ (CAR SIDES)
                       'CLISP% )
                   [NCONC1 (CADR SIDES)
                           (CDR (LISTGET1 LISPXHIST 'SIDE]
                   (LISPXPUT '*LISPXPRINT* (LIST SIDES)
                             T LISPXHIST))
              ;; Some messages were printed, and the undo informaton marked.  This completes the process enabling user to undo just the effects
              ;; associated with the dwim change corresponding to the message printed between (CADR of this mark) and the place where the
              ;; mark appears.  The use of CLISP  makes the mark invisible to the editor, and also does not i nterefere with printing the event.
              [COND
                 ((AND DWIMIFYFLG DWIMIFYING)
                  (SETQ DWIMIFY0CHANGE T))
                 (T (AND (NULL TYPE-IN?)
                         (EXPRP FAULTFN)
                         (DWIMARKASCHANGED FAULTFN SIDES]
              (COND
                 (DWIMIFYFLG (SETQ DWIMIFYCHANGE T)
                        (CL:THROW 'WTFIX X)))
              (COND
                 ((NULL APPLYFLG)
                  [COND
                     (HISTENTRY (/RPLNODE HISTENTRY (LIST X)
                                          (CDR HISTENTRY))
                                (AND (ATOM X)
                                     (SETQ LASTWORD X]
                  (RETEVAL POS (FIXLISPX/ X)
                           T))
                 (T (AND HISTENTRY (/RPLNODE HISTENTRY (LIST X ARGS)
                                             (CDR HISTENTRY)))
                    (RETAPPLY POS (FIXLISPX/ X)
                              (FIXLISPX/ ARGS X)
                              T]
             ((AND CLISPFLG DWIMIFYFLG FORMSFLG (NEQ NOSPELLFLG T)
                   (OR (NULL NOSPELLFLG)
                       TYPE-IN?)
                   (EQ FAULTX (CAR TAIL))
                   (EQ TAIL PARENT)
                   (STRPOSL CLISPCHARRAY (CAR TAIL))
                   (DWIMIFY2A TAIL CHARLST))
              ;; In the event that a parenthesis was left out, and (CAR TAIL) is really the name of a function (or misspelled function), spelling
              ;; correction would nothave been attempted earlier in DWIMIFY2 until seeing if this was ok CLISP, so try it now.  E.g. (IF A THEN
              ;; FOOX-1), where FOO is name of a function, or (IF A THEN R/PLNODE X).  Note that CLISPCHANGES might be NIL in the case
              ;; that the clisp transformationdidn't go throuh, e.g. missing operand.
              (/RPLNODE TAIL (CONS (CAR TAIL)
                                   (CDR TAIL)))
              (SETQ X (DWIMIFY1? (CAR TAIL)))
              (SETQ POS FAULTPOS)
              (GO TOP))
             (CLISPCHANGES (COND
                              (DWIMIFYFLG (SETQ ATTEMPTFLG T)
                                          (SETQ CLISPCHANGE T)))
                           (COND
                              ((NULL (RETDWIM1 (CDDR CLISPCHANGES)))
                               (RELSTK FAULTPOS)
                               (ERROR!)))
                           (SETQ X (CAR CLISPCHANGES))
                           [MAPC (CADR CLISPCHANGES)
                                 (FUNCTION (LAMBDA (X)
                                             (COND
                                                ((LISTP (CAR X))
                                                 (/RPLNODE (CAR X)
                                                           (CADR X)
                                                           (CDDR X)))
                                                (T (APPLY (CAR X)
                                                          (CDR X]
                           (SETQ POS FAULTPOS)
                           (GO TOP))
             (CLISPERTYPE
                    ;; Error messages are postponed till this point because what looks like a bad clisp expression may be interpreted correctly in
                    ;; a different way --- e.g. _PENP will correct to openp.
                    (AND DWIMIFYFLG (SETQ ATTEMPTFLG T)
```

```
                        (SETQ CLISPCHANGE T))
                ;; ATTEMPTFLG to inform DWIMMFY not to add FAUTX to NOFIXLST.  CLISPCHANGE is to prevent analysing cdr of the
                ;; form in the case the error occurred in CAR of the form.
                (AND (OR DWIMIFYFLG (NULL TYPE-IN?))
                     (CLISPERROR CLISPERTYPE]
        (COND
           (DWIMIFYFLG                                          ; ERROR! instead of CL:THROW so that UNDONLSETQ
                                                                ; changes are undone
                   (ERROR!))
           (T (RELSTK FAULTPOS)
              [CL:THROW 'WTFIX (AND (NULL TYPE-IN?)
                                    (CONS FAULTFN (COND
                                                     ((ATOM FAULTX)
                                                      (RETDWIM2 PARENT TAIL))
                                                     (T (RETDWIM2 FAULTX NIL 2]
                                                                ; The vaue retunred by WTFIX is used on the call to
                                                                ; OLDFAULT1 for printing out a message.

              ])
```

(DWIMERRORRETURN

```
[LAMBDA (ARG)                                                   (* lmm " 5-SEP-83 23:51")
   (AND ARG (SETQ CLISPERTYPE ARG))
   (ERROR!])
```

(DWIMARKASCHANGED

```
[LAMBDA (FN $SIDES)                                             (* rmk%: "18-FEB-83 17:07")
```

;; Informs the file package that FN has been changed, giving CLISP as the reason if we detect (because no messages were printed) that the only
;; changes are because of valid clisp dwimifications. Otherwise, the reason is CHANGED

```
   (AND (LITATOM FN)
        (PROG [(L (CDR (LISTGET1 LISPXHIST 'SIDE]
          LP  (COND
                 ((OR (NULL L)
                      (EQ L $SIDES))
                  (RETURN)))
              [SELECTQ (CAAR L)
                   ((/PUTHASH CLISPRPLNODE *)                   ; For some reason (ask wt!), these aren't counted as real
                                                                ; changes
                        NIL)
                   (RETURN (MARKASCHANGED FN 'FNS (COND
                                                     ((FASSOC 'CLISP%  (LISTGET1 LISPXHIST '*LISPXPRINT*))
                                                      'CHANGED)
                                                     (T 'CLISP]
              (SETQ L (CDR L))
              (GO LP])
```

(RETDWIM1

```
[LAMBDA (L)                                                     (* lmm "20-May-84 19:58")
```

;; Called when about to make a CLISP transformation for which one of the atmic operands are not bound.

```
   (PROG (($TAIL (CAR L))
          ($CURRTAIL (CADR L))
          FLG TEM)                                              ; CLISPCHANGES rebound so that FIXSPELL1 will only ask for
                                                                ; approval if dwim mode indicates.
         [SETQ TEM (COND
                      ((EQ (CDR $TAIL)
                           $CURRTAIL)
                       (RETDWIM2 (CAR $TAIL)))
                      (T (APPLY 'CONCAT (MAPCON (COND
                                                   ((TAILP $CURRTAIL $TAIL)
                                                    (LDIFF $TAIL $CURRTAIL))
                                                   (T $TAIL))
                                                (FUNCTION (LAMBDA (X)
                                                            (COND
                                                               [(LISTP (CAR X))
                                                                (LIST (RETDWIM2 (CAR X]
                                                               ((OR (FMEMB (NTHCHAR (CAR X)
                                                                                    -1)
                                                                           CLISPCHARS)
                                                                    (FMEMB (CADR X)
                                                                           CLISPCHARS))
                                                                (LIST (CAR X)))
                                                               (T (LIST (CAR X)
                                                                        '" "]
         (COND
            ([OR TREATASCLISPFLG (AND (EQ (CADDR L)
                                          'PROBABLY)
                                      (OR (AND DWIMIFYFLG DWIMIFYING)
                                          (NULL TYPE-IN?]
```

;; The idea here is that it does not make sense to automatcaaly go ahead and perform a transformation to typein that is then going to
;; produce an error, e.g. user type FOO_FIE where FIE is unbound. Therefore we will always ask him for type-in? Note that he may
;; say YES even though it will produce an error, so that he can then say Ÿ 'Ÿ or -> something. --- In functons, if the operation involves

```
                ;; more than one CLISP operator (or an assignment where the variable is one of the bound varables.) we will just tell him.
                (SETQQ FLG NEEDNOTAPPROVE))
               (T (SETQQ FLG MUSTAPPROVE)))
            (COND
               ((COND
                   ((AND TREATASCLISPFLG (NULL CLISPHELPFLG))      ; dont print any message, but do treat it as clisp
                    T)
                   ((OR TREATASCLISPFLG CLISPHELPFLG)              ; interact (ask or inform) with user if either treatasclispflg is T, or
                                                                   ; clisphelpflg is T , or both.
                    (FIXSPELL1 TEM (COND
                                      (LCASEFLG '" as clisp")
                                      (T
                                        ;; The reason for the check is that the user may want to key on this message for an UNDO :
                                        ;; operation, and if he is on a 33 and it is printed as a lowercase string (even though he sees it in
                                        ;; uppercase) he wont be able to fnd it.
                                          '" AS CLISP"))
                                   (COND
                                      [(EQ FLG 'NEEDNOTAPPROVE)
                                       (COND
                                          (LCASEFLG '" treated")
                                          (T '" TREATED"]
                                      [(EQ FLG 'MUSTAPPROVE)
                                       (COND
                                          (LCASEFLG '" treat")
                                          (T '" TREAT"]
                                      (T (SHOULDNT)))
                                   T FLG))
                   ((EQ FLG 'NEEDNOTAPPROVE)                       ; dont interact, but treat it as clisp, e.g. when transformation is a
                                                                   ; PROBABLY and we are dwimifying.
                    T))
                (SETQ NOFIXVARSLST0 (CADDDR L))
                ;; Since user has approved CLISP, it is ok to set NOFIXVARSLST0 to include any variabes detected during analysis of CLISP
                ;; expression, e.g. if expression were A*B A and B can now be added NOFIXVARSLST0
                (RETURN T)))
            (RETURN (COND
                       (DWIMIFYFLG (SETQ NEXTAIL (NLEFT (CAR L)
                                                        1 $CURRTAIL))
                                                                   ; Tells DWIMIFY where to continue.
                                   (COND
                                      ((LISTP (CAR NEXTAIL))
                                       (SETQ NEXTAIL (NLEFT (CAR L)
                                                            2 $CURRTAIL))
                                       ;; E.G. In A* (FOO --), this will enable (FOO --) to be processed.  If the expression immediately before
                                       ;; CURRTAIL is an atom, we have no way of knowing if it contains a CLISP operator or not, e.g. is it A + B, or
                                       ;; A+B.  If we were to back up NEXTAIL so that DWIMIFYING continued as of this atom, it might cause a
                                       ;; loop.

                                       ))
                                   NIL]))
```

(FIX89TYPEIN

```
  [LAMBDA (X CLST APPLYFLG)
    (PROG (TEM)
          (PRIN1 '= T)
          (COND
             [(EQ X CLST)                                          ; THE 8 is the first character.
              (PRINT (SETQ TEM (PACK (CDR X)))
                     T T)
              (RETDWIM FAULTPOS (CONS TEM (COND
                                             ((NULL APPLYFLG)      ; E.g. 8FOO X Y
                                              (CDR FAULTX))
                                             (FAULTARGS            ; E.G. 8FOO (A B)
                                                    (LIST FAULTARGS]
             (T [SETQ FAULTARGS (COND
                                   ((AND APPLYFLG FAULTARGS)       ; E.g. 'FOO8)' or 'FOO8A)' or 'FOO8A B]'
                                    (LIST FAULTARGS))
                                   (T                              ; E.g. 'FOO8A B C]' (or 'FOO8 A B]')
                                      (CDR FAULTX]
                (RETDWIM FAULTPOS (PRINT (SETQ TEM (PACK (LDIFF CLST X)))
                                         T T)
                         T
                         (COND
                            ((NULL (CDR X))
                             FAULTARGS)
                            (T (CONS (PACK (CDR X))
                                     FAULTARGS])
```

(FIXLAMBDA

```
  [LAMBDA (DEF)                                                    (* lmm "20-May-84 19:57")
```

;; LAMBDASPLST is initialized to (LAMBDA NLAMBDA). HOwever users can add to it for 'function' handled by DWIMMUSERFN. QLISP uses this
;; feature.

```
    (AND (LITATOM (CAR DEF))
         (CDDR DEF)
         (NOT (FMEMB (CAR DEF)
                     LAMBDASPLST))
         (FIXSPELL (CAR DEF)
                NIL LAMBDASPLST NIL DEF NIL NIL NIL T])
```

(**FIXAPPLY**

```
  [LAMBDA NIL                                                    (* lmm "19-MAY-84 21:44")
    (PROG (X TEM)
          (COND
             ((NEQ FAULTFN FAULTX)
```

;; means the call came out of compiled code, e.g. user types in FOO which contains a call to a mispelled function.

```
              (SETQ TYPE-IN? NIL)))
          (COND
             ((AND (LITATOM FAULTX)
                   (SETQ X (FGETD FAULTX)))
              (COND
                 ([NULL (PROG (TYPE-IN?)
                              (RETURN (FIXLAMBDA X]
                  (GO NX)))
              (AND FILEPKGFLG (LITATOM FAULTX)
                   (MARKASCHANGED FAULTX 'FNS))
              (SETQ X FAULTX)
              (GO OUT))
             ((AND (OR (GETPROP FAULTX 'EXPR)
                       (GETPROP FAULTX 'CODE))
                   (DWIMUNSAVEDEF FAULTX))
              (SETQ X FAULTX)
              (SETQ FAULTFN NIL)                                 ; So that RETDWIM won't do a NEWFILE?
              (GO OUT))
             ((SETQ TEM (GETPROP FAULTX 'FILEDEF))
              (COND
                 ((WTFIXLOADEF TEM)
                  (SETQ X FAULTX)
                  (GO OUT1)))
              (RETDWIM))
             ((AND TYPE-IN? CLISPFLG (STRPOSL CLISPCHARRAY FAULTX)
                   (SETQ X (CLISPATOM CHARLST (SETQ TEM (LIST FAULTX FAULTARGS))
                                      TEM T)))                   ; E.g. FOO_ form.  FOO _form is caught by a special check in
                                                                 ; LISPX and treated as (FOO _form)
              (RETDWIM FAULTPOS X))
             ((AND TYPE-IN? (NEQ NOSPELLFLG T)
                   (EQ FAULTXX (CAAR HISTENTRY))
                   (SETQ TEM (FMEMB LPARKEY CHARLST)))
              (FIX89TYPEIN TEM CHARLST T))
             ((AND (LISTP FAULTX)
                   (FIXLAMBDA FAULTX))                           ; LAMBDA or NLAMBDA misspelled in LAMBDA expression
                                                                 ; being applied, e.g. a functional argument.
              (SETQ X FAULTX)
              (GO OUT)))
      NX  (COND
             [[AND DWIMUSERFORMS (SOME DWIMUSERFORMS (FUNCTION (LAMBDA (DWIMUSERFORM)
                                                                (SETQ TEM (EVAL DWIMUSERFORM]
              (COND
                 (FAULTAPPLYFLG (RETDWIM FAULTPOS TEM T FAULTARGS))
                 (T (RETDWIM FAULTPOS TEM]
             ([NULL (SETQ X (OR (FIXSPELL FAULTX NIL SPELLINGS1 NIL NIL NIL NIL NIL T)
                                (FIXSPELL FAULTX NIL SPELLINGS2 NIL NIL NIL NIL NIL T]
              (RETDWIM)))
      OUT (FIXAPPLY1 (FSTKNTH -1 FAULTPOS)
                 FAULTX X)
      OUT1
          (RETDWIM FAULTPOS X T FAULTARGS])
```

(**FIXATOM**

```
  [LAMBDA NIL                                                    (* bvm%: "21-Nov-86 16:38")
    (PROG (X Y TAIL0)
          (COND
             ((NULL TAIL)
              (SETQ TAIL (FINDATOM FAULTX (SETQ X (STKNTH -1 FAULTPOS))
                                   (BLIPVAL '*FORM* X)))
              (RELSTK X)))
          (SETQ TAIL0 (AND (NEQ ONLYSPELLFLG 'NORUNONS)
                           TAIL))
```

;; ONLYSPELLFLG is NORUNONS for calls from CLISPATOM2A, i.e. when DWIMIYING one of the operands to an infix operator. IN this case it ;; never makes sense to do a runon spelling correction, e.g. FOOX*A shouldnt correct to (ITIMES FOO X A), althouh it may correct to FOO X*A.

```
          (COND
             ((SETQ X (CLISPATOM CHARLST TAIL PARENT))
              (GO OUT))
             ([AND (CDR TAIL)
                   (LITATOM (SETQ Y (CADR TAIL)))
                   (FMEMB (CHCON1 Y)
```

```
                    (CHARCODE (_ ←)))
              (PROG (CLISPERTYPE)
                    (RETURN (SETQ X (CLISPATOM (UNPACK Y)
                                               (CDR TAIL)
                                               PARENT T]        ; E.G. (LIST FOO _ 3) where FOO is unbound at the time. See
                                                                ; comment in WTFIX.
               (GO OUT))
              ([AND DWIMUSERFORMS (SOME DWIMUSERFORMS (FUNCTION (LAMBDA (DWIMUSERFORM)
                                                                  (SETQ X (EVAL DWIMUSERFORM]
               (GO OUT))
              ((OR (EQ NOSPELLFLG T)
                   (AND NOSPELLFLG (NULL TYPE-IN?))
                   (GETPROP FAULTX 'GLOBALVAR)
                   (FMEMB FAULTX GLOBALVARS))
               ;; For efficiency, GLOBALVARS is a global variable itself for DWIMBLOCK.  Thus FIXATOM obtains the top level value, not the one
               ;; rebound by BCOMPL2.  However, in the case that there are block declarations aafecting globalvars, the variables would also have
               ;; been added to NOFIXVARSLST, so this is ok.
               (RETDWIM))
              ((AND VARS (SETQ X (FIXSPELL FAULTX NIL VARS NIL TAIL0 NIL NIL NIL T)))
               ;; Corrects spellings using LAMBDA and PROG variables of function in which error occurred, or function that is broken.

               )
              ((SETQ X (FIXSPELL FAULTX NIL SPELLINGS3 NIL TAIL0 NIL NIL NIL T)))
              ((AND DWIMIFYFLG (EQ CLISPCONTEXT 'IFWORD)
                    (FIXSPELL FAULTX NIL CLISPIFWORDSPLST NIL T NIL NIL NIL T))
               (CL:THROW 'CLISPIF0 :RESPELL))
              ((AND DWIMIFYFLG (EQ CLISPCONTEXT 'FORWORD)
                    (FIXSPELL FAULTX NIL CLISPFORWORDSPLST NIL T NIL NIL NIL T))
               (CL:THROW 'CLISPFOR0 :RESPELL))
              [(AND DWIMIFYFLG NOFIXVARSLST0 (SETQ X (FIXSPELL FAULTX NIL NOFIXVARSLST0 NIL TAIL0 NIL NIL NIL T]
              ((AND DWIMIFYFLG CLISPFLG (OR (EQ CLISPCONTEXT 'IS)
                                            (AND (LISTP CLISPCONTEXT)
                                                 TAIL
                                                 (EQ TAIL PARENT)))
                    (SETQ X (FIXSPELL FAULTX NIL CLISPISWORDSPLST NIL TAIL NIL NIL NIL T)))
               (COND
                  ((EQ CLISPCONTEXT 'IS)
                   ;; In this case, we are dwimifying the tail before processing it in clispatomis so is sufficient just to correct spelling and return.

                   )
                  ((SETQ X (CLISPATOM (DUNPACK X WTFIXCHCONLST)
                                      TAIL PARENT))             ; E.g. X IS A NUMBER OR STRNG, STRNG being misspelled.
                                                                ; Will call CLISPATOMIS?  which will retfrom.

                   ))
               (GO OUT))
              ([AND CLISPFLG (NULL CLISPCHANGES)
                    (NULL CLISPERTYPE)
                    (SETQ X (FIXSPELL FAULTX NIL CLISPINFIXSPLST NIL (COND
                                                                       (DWIMIFYFLG (LISTP CLISPCONTEXT))
                                                                       (T TAIL))
                                      NIL NIL NIL T))
                    (COND
                       ([AND DWIMIFYFLG (OR (LISTP CLISPCONTEXT)
                                            (EQ CLISPCONTEXT 'IS]
                        (CL:THROW 'CLISPATOM1 :RESPELL))
                       (T (SETQ X (CLISPATOM (SETQ CHARLST (DUNPACK (SETQ FAULTX X)
                                                                    WTFIXCHCONLST1))
                                             TAIL PARENT]
               (GO OUT))
              ((AND (EQ FAULTX (CAR TAIL))
                    (NUMBERP (CAR CHARLST))
                    [SETQ X (SOME CHARLST (FUNCTION (LAMBDA (X)
                                                      (NOT (NUMBERP X]
                    (FIXSPELL1 FAULTX (SETQ Y (CONS (PACK (LDIFF CHARLST X))
                                                    (PACK X)))
                          NIL CHARLST 'MUSTAPPROVE))
               (/RPLNODE TAIL (CAR Y)
                      (CONS (CDR Y)
                            (CDR TAIL)))
               (SETQ X (CAR Y))
               (GO OUT))
              (T (RETDWIM)))
        [COND
           ((AND (NULL TAIL0)
                 (EQ FAULTX (CAR TAIL)))                        ; If TAIL0 is not NIL, the RPLNODE has aleady been done.
            (/RPLNODE TAIL X (CDR TAIL]
   OUT [COND
           ((AND NEWTAIL (NULL DWIMIFYFLG))
            ;; The interpreter has already made up its mind about how to handle the first operand of the CLISP expression, e.g. it has already
            ;; been evaluated as an argument, or else is about to be called as a function.  Therefore continuing the computation requires some
            ;; fiddling around.
            (SETQ X (FIXATOM1]
        (RETDWIM FAULTPOS X])
```

(FIXATOM1

```
[LAMBDA NIL                                                    (* lmm "20-SEP-83 23:37")
```

;; Called when evaluation went too far before DWIM fixed an CLISP expression. See comment in FIXATOM

```
(PROG ((POS (STKNTH -1 FAULTPOS))
       X OLDTAIL OLDFN)
      (SETQ OLDTAIL (BLIPVAL '*TAIL* POS))
      (AND (LISTP NEWTAIL)
           (SELECTQ (CAR PARENT)
                ((AND OR PROG PROG2 PROG1 PROGN LAMBDA NLAMBDA)
                     (COND
                        ((NEQ TAIL OLDTAIL)
                         (GO ERROR)))
                     (SETBLIPVAL '*TAIL* POS NIL NEWTAIL)      ; Change the binding for the tai
                     (FIXCONTINUE (CADAR NEWTAIL))
                     (SETQ X (CAR NEWTAIL))
                     (GO OUT))
                NIL))
      (SETQ OLDFN (BLIPVAL '*FN* POS))
      [COND
         ([COND
             ((NEQ TAIL OLDTAIL)                               ; E.g. (COND (ZAP _ T 3)) where ZAP is A u.b.a.
              T)
             ((LISTP NEWTAIL)                                  ; E.G. (LIST FOO X + Y)
              (NEQ OLDFN (CAR PARENT)))
             [(ATOM (CADR PARENT))
```

;; e.g. (FOO AND T) where FOO is the name of a function as well as a variable. the check here used to be (NEQ OLDFN ;; (CADR PARENT)). however this fails for things like (FOO : FIE) which at this point would be (fetch FIE of FOO), i.e. cant ;; assume that car of form is now CADR

```
              (AND (NEQ OLDFN (CADR PARENT))
                   (NEQ OLDFN (CADDDR PARENT]
             (T
```

;; For infixes like EQ, AND, OR, the function that was about to be called may now be parenthesized, e.g. (FOO X EQ Y) ;; becomes (EQ (FOO X) Y) However, it is also possible that it was not a function at all, e.g. (FOO GT 4 AND FOO LT 6)

```
                (NOT (FMEMB OLDFN (CADR PARENT]
```

;; The procedure followed assumes that Y gives the binding for TAIL, and Z gives the binding for the name of the function that is ;; about to be called. This checks to make sure that this is in fact the cas

```
              (GO BAD))
             ((NLISTP NEWTAIL)
```

;; Occurs when CAR of an xpression in which a CLISP operator is used is the name of a function, e.g. (FOO + X), (FOO X AND FIE ;; Y). Note that at this point in the evaluton, the nterpreter is evaluating the 'arguments' for that function, and plans to call it when they ;; have all been evaluated

```
              NIL)
             ((OR (CDR NEWTAIL)
                  (ZEROP (LOGAND (ARGTYPE (CAR PARENT))
                                 2)))
```

;; Either there are more arguments following the CLISP expression, or, in the case of a spread, evaluate, it doesn't matter if an extra ;; NIL is passed. Therefore, proceed by smashing the last argument with the value of the CLISP expression, (CAR NEWTAIL), ;; change the binding for the tail to be (CDR NEWTAIL), and RETDWIM with the next expression on TAIL, (CADR NEWTAIL) e.g. ;; (LIST T 2 + 3 6)

```
              [SETBLIPVAL '*ARGVAL* POS NIL (STKEVAL POS (FIXLISPX/ (CAR NEWTAIL]
              (SETBLIPVAL '*TAIL* POS NIL (CDR NEWTAIL))
              (SETQ X (CADR NEWTAIL))
              (GO OUT))
             (T
```

;; The function to be called is a nospread function, e.g. LIST, and the CLISP expression was its last argument, e.g. (LIST X (--) *2) ;; Therefore can only continue by reevaluating the whole form

```
                (FIXCONTINUE (CADAR NEWTAIL)
                       (AND (NULL TYPE-IN?)
                            FAULTFN]
      (SETBLIPVAL '*TAIL* POS NIL NIL)                         ; Makes tail of the argument list be NIL
      (SETBLIPVAL '*FN* POS NIL 'FIXATOM2)                     ; A nospread, evaluate function whose value is the value of its
                                                               ; last argument
      (SETQ X PARENT)
      (GO OUT)
```

;; PARENT will be evaluated, and its value stored on the stack. Then since the tail of the argument list is now NIL, the interpreter figures that the ;; evaluation of arguments is finished, and calls the function. However since Z was changed, FIXATOM2 will be called instead, and it will return ;; as its value its last argument, which will be the value of PARENT. Voila

```
 BAD                                                           ; Stack not in normal state
      (SELECTQ (STKNAME (SELECTQ (SYSTEMTYPE)
                                 ((JERICHO D)                  ; Skip over internal frames
                                      (REALSTKNTH -1 POS T POS))
                                 POS))
          (COND (COND
                    ((EQ PARENT NEWTAIL)
```

;; The CLISP transformation changed the predicate of a COND clause, e.g. (COND (FOO _ form --) --) Since the ;; COND would ordinarily continue down that clause, it is necessary to continue by constructing an appropriate COND ;; expression, and returning its value as the value of the entire COND

```
                     [SETQ X (CONS 'COND (FMEMB PARENT (STKARG 1 POS]
```

```
                        (RELSTK FAULTPOS)
                        (SETQ FAULTPOS POS)
                        (GO OUT))
                       (T ;; The CLISP transformation did not affect the predicate of a COND clause, so can continue by just evaluating
                          ;; PARENT E.G. (COND (T FOO _ 2))
                          (SETQ X (CAR NEWTAIL))
                          (GO OUT))))
             ((PROGN PROG1)                                    ; Error in SELECTQ clause, e.g. (SELECTQ -- (-- A * B)) or error
                                                               ; in savesetq
                  (SETQ X (CONS (STKNAME POS)
                                NEWTAIL))
                  (RELSTK FAULTPOS)
                  (SETQ FAULTPOS POS)
                  (GO OUT))
             NIL)
      ERROR
          (ERROR '"DWIM is confused about the stack" '"" T)
      OUT (AND (NEQ POS FAULTPOS)
               (RELSTK POS))
          (RETURN X])
```

(**FIXCONTINUE**

```
[LAMBDA (X FN)
  (SETQ X (OR (AND CLISPARRAY (GETHASH X CLISPARRAY))
              X))
  (COND
     ((OR (NLISTP X)
          (FIXCONTINUE1 X))
      T)
     (T (FIXPRINTIN FN)
        (OR (EQ (ASKUSER (ITIMES DWIMWAIT 3)
                         'Y
                         (LIST '" ok to reevaluate " (RETDWIM2 X NIL 2))
                         DWIMKEYLST)
                'Y)
            (RETDWIM])
```

(**FIXCONTINUE1**

```
[LAMBDA (X)                                                    (* True if it is ok to reevaluate X.)
  (OR (EQ (CAR X)
          'QUOTE)
      (AND [OR (FMEMB (CAR X)
                      OKREEVALST)
               (GETPROP (CAR X)
                        'CROPS)
               (EQ (CAR (GETPROP (GETPROP (CAR X)
                                          'CLISPCLASS)
                                 'CLISPCLASSDEF))
                   'ARITH)
               (AND (EQ (CAR X)
                        'SETQ)
                    (NOT (EDITFINDP (CADDR X)
                                    (CADR X]
           (PROG NIL
             LP  (COND
                    ((NULL (SETQ X (CDR X)))
                     (RETURN T))
                    ([AND (LISTP (CAR X))
                          (NULL (FIXCONTINUE1 (CAR X]
                     (RETURN NIL)))
                (GO LP])
```

(**CLISPATOM**

```
[LAMBDA (CLST TAIL PARENT NOFIX89)                             (* lmm "20-May-84 19:46")
```

;; CLST is an exploded character list for CAR of TAIL, which is a tail of PARENT, although not necessarily a proper tail. ONLYSPELLFLG=T
;; indicates that the ONLY corrections to be attempted are spelling corrections. Occurs on calls from CLISPATOM2a.

```
  (AND (NULL ONLYSPELLFLG)
       (PROG (TEM)
             (COND
                [(AND (NULL CLISPCHANGES)
                      (OR (EQ CLISPFLG T)
                          (AND (EQ CLISPFLG 'TYPE-IN)
                               TYPE-IN?)))
```

;; If CLISPCHANGES is not NIL, a CLISP correction has already been found, so don't bother to find another, e.g. in (X+Y + Z),
;; if X and Y are not bound vriables, after ggetting (IPLUS X Y Z), this would be undone and saved, pending spelling correction
;; on X+Y. Therefore don't do the transformation that staats with +Z.

```
                 (RETURN (COND
                            ((SETQ TEM (CLISPATOM0 CLST TAIL PARENT))
                             TEM)
                            (CLISPCHANGES (SETQ CHARLST (DUNPACK FAULTXX WTFIXCHCONLST))
                                                               ; Since DWIMIFY2, and hence WTFIX, may have been called,
                                                               ; LST may have been clobbered.
```

```
                                NIL]
                ((AND (EQ (CAR CLST)
                          '%')
                      (GETPROP '%' 'CLISPTYPE))                ; So ' can be disabled when CLISP is turned off as well.
                 [COND
                    [(CDR CLST)
                     [SETQ TEM (LIST 'QUOTE (PACK (CDR CLST]
                     (COND
                        ((NULL TAIL))
                        ((NEQ TAIL PARENT)
                         (/RPLNODE TAIL TEM (CDR TAIL)))
                        (T (RETDWIM]
                    ((NULL (CDR TAIL))
                     (RETDWIM))
                    ((EQ TAIL PARENT)
                     (/RPLNODE TAIL 'QUOTE (CDR TAIL))
                     (SETQ TEM TAIL))
                    (T (/RPLNODE TAIL (SETQ TEM (LIST 'QUOTE (CADR TAIL)))
                               (CDDR TAIL]
                 (RETURN TEM)))
              (COND
                 ([OR NOFIX89 (EQ NOSPELLFLG T)
                      (AND NOSPELLFLG (NULL TYPE-IN?))
                      (NULL (OR (SETQ TEM (FMEMB LPARKEY CLST))
                                (SETQ TEM (FMEMB RPARKEY CLST]
                  NIL)
                 [(AND (OR (LISTP FAULTX)
                           TAIL)
                       (FIX89 FAULTX (CAR TEM)))
                  (RETDWIM FAULTPOS (COND
                                       ((ATOM FAULTX)
                                        (CAR TAIL))
                                       (T FAULTX]
                 ((AND TYPE-IN? (EQ (CAR TEM)
                                    LPARKEY)
                       (EQ (CAR (SETQ TEM (FLAST CLST)))
                           RPARKEY))
                  ;; This corresponds to the case where an atom was typed in containing both an 8 and a 9, e.g. FILES?89 or 8EDITF9.  Note
                  ;; that if the atom were part of a larger expression, either CAR of form, or appearing in a tail, (as indicated by TAIL being
                  ;; non-NIL), the fix is performed by FIX89, and involves editing the expression.  In the case covered here, the fix requires
                  ;; changing the EVAL to an apppropriate APPLY.  The case where the 8 or 9 error appears in an APPLY context, or line
                  ;; format, is taken care of in WTFIX.
                  (FIX89TYPEIN (FMEMB LPARKEY (SETQ TEM (LDIFF CLST TEM)))
                         TEM T])
```

(GETVARS

```
[LAMBDA (X)                                                    (* lmm "20-May-84 19:24")
  (PROG (L POS TEM)
        (COND
           ((EQ X T)                                           ; context is inside of a BREAK --- Gets variables of BRKFN.
            (SETQ POS (STKPOS 'BREAK1 -1 FAULTPOS))
            [COND
               ((AND [NOT (EQ 0 (STKNARGS (SETQ TEM (FSTKNTH -1 POS]
                     (LITATOM (STKARGNAME 1 TEM)))             ; If the first argument's name is #0 or #100, there are no genuine
                                                               ; variables.
                (SETQ L (VARIABLES TEM]
            (SETQ X (STKARG 1 POS))
            (RELSTK TEM)
            (RELSTK POS)                                       ; Sets X to BRKEXP the first argument to BREAK1.  Used for
                                                               ; getting PROG variables below.


            )
           [(EQ (CAR X)
                'LAMBDA)                                       ; Gets variables for expression X.
            (SETQ L (APPEND (CADR X]
           (T (RETURN NIL)))
        (RETURN (NCONC L (AND (LISTP X)
                              (MAPCAR (CADR (GETVARS1 X))
                                      (FUNCTION (LAMBDA (X)
                                                  (COND
                                                     ((NLISTP X)
                                                      X)
                                                     (T (CAR X])
```

(GETVARS1

```
[LAMBDA (X)                                                    (* DD%: " 2-Dec-81 16:49")
  ;; Looks for a PROG.
  (SELECTQ [CAR (SETQ X (CAR (LISTP (LAST (LISTP X]
       ((PROG RESETVARS)
            X)
       ((RESETLST
            RESETVAR
            RESETFORM)
```

```
            (GETVARS1 X))
        NIL])
```

(FIX89

```
[LAMBDA (FORM N POS)                                        (* bvm%: "21-Nov-86 18:47")
```

;; Handles corrections for 8 and 9 errors. N is either 8 or 9.0 POS is optional, and if given, it is the position of the 8 or 9 in the offending atom, and ;; also indicates that the user has already approved the correction.

```
(PROG [SPLIT89FLG (C (COND
                       ((EQ N LPARKEY)
                        'FIX8)
                       (T 'FIX9]
      (COND
         ([OR (AND (ATOM FAULTX)
                   (NULL TAIL))
              (AND (NULL POS)
                   (NULL (FIX89A FAULTX N]                  ; pointless to attempt an 8 or 9 correction if TAIL is NIL.
          (RETURN NIL)))                                    ; Gets user approval if necessary, i.e. if TYPE-IN? is NIL and
                                                            ; APPROVEFLG is T.
      (EDITE EXPR (LIST (LIST 'ORR (LIST (LIST (COND
                                                 ((ATOM FORM)
                                                  'F)
                                                 (T 'F=))
                                               FORM T)
                                         (LIST C NIL POS))
                              NIL)))                        ; Constructs command of form ((ORR ((F= FORM T) C) NIL)) C
                                                            ; is either FIX8 or FIX9 depending on call.
      (RETURN (COND
                 ((NULL SPLIT89FLG)                         ; Set in SPLIT89 if successful.
                  (EXEC-FORMAT "couldn't~%%")
                  NIL)
                 (T (AND DWIMIFYFLG (SETQ 89CHANGE T))
                    T])
```

(FIXPRINTIN

```
[LAMBDA (FN FLG)                                            (* wt%: 12-OCT-76 21 40)
```

;; If FLG is T, printing goes on history lst.

```
(AND FN (NEQ FN 'TYPE-IN)
     (PROG ((LISPXHIST (AND FLG LISPXHIST)))
           (AND (NEQ (POSITION T)
                     0)
                (LISPXSPACES 1 T))
           (LISPXPRIN1 '"{" T)
           (LISPXPRIN1 (COND
                          [(OR (AND DWIMIFYFLG DWIMIFYING)
                               (NULL FAULTAPPLYFLG))
                           (COND
                              (LCASEFLG
```

;; Done this way instead of just printing the lower case version because users may want to efer to the ;; message to undo a dwim correction, e.g. by typing UNDO : IN.

```
                                  '"in ")
                              (T '"IN "]
                           (LCASEFLG '"below ")
                           (T '"BELOW "))
                       T)
           (LISPXPRIN2 FN T T)
           (LISPXPRIN1 '"}" T)
           (RETURN FN])
```

(FIX89A

```
[LAMBDA (X N POS)                                           (* wt%: 25-FEB-76 1 40)
  [COND
     ((LISTP X)
      (SETQ X (CAR X]
  (OR POS (SETQ POS (STRPOS N X)))
  (COND
     ((FIXSPELL1 X (CONS [CONCAT (OR (SUBSTRING X 1 (SUB1 POS))
                                     '"")
                                 (COND
                                    ((EQ N LPARKEY)
                                     '" (")
                                    (T '" )"]
                         (OR (SUBSTRING X (ADD1 POS))
                             '""))
           NIL CHARLST (AND (NULL TYPE-IN?)
                            'MUSTAPPROVE))
      T)
     (DWIMIFYFLG (SETQ ATTEMPTFLG T)
           NIL])
```

(CLISPFUNCTION?

```
[LAMBDA (TL TYPE FN1 FN2 Y)                                    (* lmm "20-May-84 18:56")
```

;; returns TRUE if (CAR TAIL) corresponds to the name of a function (Possibly misspelled). If TYP=NOTVAR, checks first to make sure (CAR ;; TAIL) does not correspond to the name of a variable.

;; FN1 and FN2 are used to compute the arguments to FIXSPELL1. FN1 is given (CAR TAIL) and Y as its arguments, FN2 (CAR TAIL) or the ;; corrected spelling, and Y. If FN1 is supplied, FIXSPELL is called so as not to print any messages, and the interaction takes place under ;; CLISPUNCTION? control via a direct call to FIXSPELL1. In this case, if TYP=QUIET, no message is printed at all. --- If FN1 is not suppied, ;; FIXSPELL will take care of the interaction, if any, othrwisre there are no error messages.

```
(PROG (TEM CHRLST)
      (COND
         ((NULL (LITATOM (CAR TL)))
          (RETURN NIL))
         ((LISTP TYPE)
```

;; Means that we already know that (CAR TAIL) is not the name of a variable, and is also not the name of a function.

```
          (SETQ CHRLST TYPE)
          (GO SPELL))
         ([AND (EQ TYPE 'NOTVAR)
               (NULL (CLISPNOTVARP (CAR TL]
          (RETURN NIL))
         ([OR (CLISP-SIMPLE-FUNCTION-P (CAR TL))
              (FMEMB (CAR TL)
                     (COND
                        (DWIMIFYFLG NOFIXFNSLST0)
                        (T NOFIXFNSLST)))
              (LISTP (GETPROP (CAR TL)
                              'CLISPWORD]
          (GO OUT))
         ((OR (EQ NOSPELLFLG T)
              (AND NOSPELLFLG (NULL TYPE-IN?))
              (STRPOSL CLISPCHARRAY (CAR TL)))
          (RETURN NIL)))
      (SETQ CHRLST (UNPACK (CAR TL)))
  SPELL
      (COND
         ([NULL (SETQ TEM (CAR (MISSPELLED? (CAR TL)
                                            NIL SPELLINGS2 (AND FN1 'NO-MESSAGE)
                                            (COND
                                               ((NULL FN1)
                                                TL)
                                               (T T]
          (RETURN NIL)))
  OUT (RETURN (COND
                 ([OR (NULL FN1)
                      (AND (EQ TYPE 'QUIET)
                           (NULL TEM))
                      (AND CLISPHELPFLG (FIXSPELL1 [COND
                                                      (TYPE-IN? '"")
                                                      (T (CONCAT "in ... " (APPLY* FN1 (CAR TL)
                                                                                   Y]
                                                   (COND
                                                      (TYPE-IN? (APPLY* FN2 (OR TEM (CAR TL))
                                                                        Y))
                                                      (T (CONCAT "is '" (COND
                                                                           ((NULL TEM)
                                                                            (CAR TL))
                                                                           ((LISTP TEM)
                                                                            (CAR TEM))
                                                                           (T TEM))
                                                                 "' meant to be used as a function")))
                                                   NIL T (AND (OR FN1 (LISTP TEM))
                                                              'MUSTAPPROVE)
                                                   (AND (LISTP TEM)
                                                        'n]
```

;; If TYP=QUIET (from DWIMIFY2), the message is printed only on spelling correction. For other calls, e.g. TYP=OKVAR, ;; or TYP=NOTVAR, the message is printed even if no correction involved.

```
                      [AND TEM FN1 (COND
                                     ((LISTP TEM)              ; Run on correction.
                                      (/RPLNODE TL (CAR TEM)
                                             (CONS (CDR TEM)
                                                   (CDR TL)))
                                      (SETQ TEM (CAR TEM)))
                                     (T (/RPLNODE TL TEM (CDR TL]
```

;; If FN1 is NIL, TAIL would have been given to FIXSPPELL, and in this case the correction would already have been ;; stmashed into TAIL.

```
                      (CAR TL])
```

(CLISPNOTVARP

```
[LAMBDA (X)                                                    (* lmm "20-May-84 19:45")
  (AND (NOT (BOUNDP X))
       (NOT (FMEMB X VARS))
       [NOT (FMEMB X (COND
                        (DWIMIFYFLG NOFIXVARSLST0)
                        (T NOFIXVARSLST]
```

```
        [OR (AND DWIMIFYFLG DWIMIFYING)
            SHALLOWFLG
            (NULL (RELSTK (STKSCAN X FAULTPOS]
        (NOT (GETPROP X 'GLOBALVAR))
        (NOT (FMEMB X (LISTP GLOBALVARS)))
        (NOT (FMEMB X (LISTP LOCALVARS)))
        (NOT (FMEMB X (LISTP SPECVARS])
```

(CLISP-SIMPLE-FUNCTION-P

```
  [LAMBDA (CARFORM)                                          (* lmm "18-Jul-86 16:45")
    (AND (OR (FGETD CARFORM)
             (GET CARFORM 'EXPR)
             (AND (NOT (GET CARFORM 'CLISPWORD))
                  (CL:MACRO-FUNCTION CARFORM))
             (GETLIS CARFORM COMPILERMACROPROPS))
         T])
```

(CLISPELL

```
  [LAMBDA (FORM TYPE)                                        (* lmm "20-May-84 18:54")
    (PROG (VAL TEM RESPELLTAIL)
          [MAPC (LISTGET1 LISPXHIST 'RESPELLS)
                (FUNCTION (LAMBDA (X)
                            (COND
                               ((SETQ RESPELLTAIL (FMEMB (CAR X)
                                                         FORM))
                                (SETQ TEM (CDR X))
                                [COND
                                   [(LISTP TEM)
                                    (/RPLNODE RESPELLTAIL (CAR TEM)
                                              (CONS (CDR TEM)
                                                    (CDR RESPELLTAIL]
                                   (T (/RPLNODE RESPELLTAIL TEM (CDR RESPELLTAIL]
                                (AND (OR (NULL TYPE)
                                         (EQ (CAR (GETPROP (CAR RESPELLTAIL)
                                                           'CLISPWORD))
                                             TYPE))
                                     (SETQ VAL T]
          (RETURN VAL])
```

(FINDFN

```
  [LAMBDA (POS FLG)                                          (* lmm "21-May-84 00:40")
```

;; Used by HELPFIX and WTFIX. Locates highest interpreted form in the current chain of interpretation, sets free variable EXPR to this expression ;; and returns the NAME of the corresponding function, or 'BREAK-EXP', 'EVAL', or 'TYPE-IN' depending on context. also sets free variable ;; TYPE-IN? to T if the expression was typed in by the user.

;; When called from WTFIX, (FLG is T) and sets the variable BREAKFLG to T if the expression was typed into a BREAK, (In this case, DWIM uses ;; the lambda and/or prog variables for spelling corrections.)

```
    (PROG1 [PROG (NAME TOKEN TEM)
                 [COND
                    ((NULL POS)
                     (SETQ POS (STKNTH 1]
              LP  (COND
                     ((NULL POS)
                      (RETURN NIL)))
                  (SETQ NAME (STKNAME POS))
              LP1 (SELECTQ NAME
                       ((APPLY BLKAPPLY)
                            (SETQ TOKEN (STKARG 3 POS))
                            (GO APPLYTYPE))
                       (ENVAPPLY [SETQ TOKEN (COND
                                                ((OR (EQ (SETQ NAME (STKNTHNAME -1 POS))
                                                         'RETAPPLY)
                                                     (EQ NAME 'STKAPPLY))
                                                 (PROG1 (STKARG 5 (SETQ TEM (STKNTH -1 POS)))
                                                        (RELSTK TEM]
                                 (GO APPLYTYPE))
                       ((STKAPPLY RETAPPLY)
                            (SETQ TOKEN (STKARG 5 POS))
                            (GO APPLYTYPE))
                       (APPLY* [RETURN (COND
                                          (FLG (SETQ TEM (STKARGS POS))
                                               (SETQ EXPR (CDR TEM))
                                               (CAR TEM))
                                          (T (SETQ EXPR (STKARG 1 POS])
                       ((EVAL \SAFEEVAL)
                            (SETQ TOKEN (STKARG 2 POS))
                            (GO EVALTYPE))
                       (ENVEVAL [SETQ TOKEN (COND
                                                ((OR (EQ (SETQ NAME (STKNTHNAME -1 POS))
                                                         'RETEVAL)
                                                     (EQ NAME 'STKEVAL))
                                                 (PROG1 (STKARG 4 (SETQ TEM (STKNTH -1 POS)))
                                                        (RELSTK TEM]
                                 (GO EVALTYPE))
```

```
                    ((STKEVAL RETEVAL)
                         (SETQ TOKEN (STKARG 4 POS))
                         (GO EVALTYPE))
                    NIL)
               LP2 [COND
                      ((LITATOM NAME)
                       (COND
                          ([EXPRP (SETQ EXPR (GETD (COND
                                                   ((SETQ TEM (GETPROP NAME 'BROKEN))
                                                    (OR (CDR (GETPROP NAME 'ALIAS))
                                                        TEM))
                                                   (T NAME]
                           (RETURN NAME]
               LP3 (SETQ POS (REALSTKNTH -1 POS NIL POS))
                   (GO LP)
               EVALTYPE
                   (SETQ EXPR (STKARG 1 POS))
                   [RETURN (SELECTQ TOKEN
                                ((SKIP SELECTQ)
                                     (SETQ POS (STKNTH -2 POS POS))
                                     (GO LP))
                                (INTERNAL (GO LP3))
                                (NIL 'EVAL)
                                (%:                                  ; Call to EVAL comes from a BREAK (i.e. via a LISPX which was
                                                                     ; called from BREAK1.)
                                    (AND FLG (SETQ BREAKFLG T))
                                    (SETQ TYPE-IN? T)
                                    'TYPE-IN)
                                (BREAK                               ; Call to EVAL from evaluation of a breakcommand.
                                       (AND FLG (SETQ BREAKFLG T))
                                       'BREAKCOMS)
                                (BREAK-EXP                           ; Call to EVAL from EVAL, OK, or GO command.
                                           (COND
                                              ((NULL (EVALV 'BRKTYPE POS))
                                     ;; Since BRKTYPE is NIL, we are in a user BREAK.  Therefore, if broken function is an EXPR, want to
                                     ;; stop searching, otherwise continue (latter can only occur when FINDFN is called as result of EDIT
                                     ;; command since WTFIX will never be called out of compiled function.)
                                               (SETQ TEM (STKPOS 'BREAK1 -1 POS))
                                               (RELSTK POS)
                                               [SETQ NAME (STKNAME (SETQ POS (STKNTH -1 TEM TEM]
                                               (GO LP2))
                                              (T
                                     ;; EVAL, OK, or GO command to non-user BREAK expression, e.g. get a non-numeric arg
                                     ;; BREAK, fix the BRKEXP, do an EVAL, and get another error.
                                                 'BREAK-EXP)))
                                (COND
                                   ((LISTP TOKEN)
                                    (COND
                                       ((NLISTP EXPR)                ; permits caller to specify the tail
                                        (SETQ TAIL TOKEN)))
                                    'EVAL)
                                   (T (SETQ TYPE-IN? T)
                                      'TYPE-IN]
               APPLYTYPE
                   (SELECTQ TOKEN
                        ((SKIP SELECTQ)
                             (SETQ POS (STKNTH -2 POS POS))
                             (GO LP))
                        (INTERNAL (GO LP3))
                        NIL)
                   (SETQ TYPE-IN? TOKEN)
          ;; WTFIX would already know that this was an apply error because of FAULTAPPLYFLG.  However, FINDFN is called to find out
          ;; whether the expression was typed in or not.
                   (RETURN (COND
                              (FLG (SETQ EXPR (STKARG 2 POS))
                                   (STKARG 1 POS))
                              (T (SETQ EXPR (STKARG 1 POS]
        (RELSTK POS])
```

(**DWIMUNSAVEDEF**

```
 [LAMBDA (FN FLG)                                                    (* lmm "11-DEC-81 21:23")
   (LISPXPRIN2 FN T T)
   [AND (NULL FLG)
        (NULL TYPE-IN?)
        (NEQ (CAR SIDES)
             'CLISP% )
        (SETQ SIDES (LIST 'CLISP%  (LIST COMMENTFLG (FLAST (LISTGET1 LISPXHIST '*LISPXPRINT*))
                                         SIDES]                      ; FLG is TRUE on calls from CLISPIFY, in which case SIDES is
                                                                     ; not relevant (or even bound)
   (LISPXPRIN1 '" unsaved" T)
   (LISPXTERPRI T)
   (UNSAVEDEF FN])
```

```
(CHECKTRAN
  [LAMBDA (X)                                                    (* lmm "20-May-84 19:01")
    (DECLARE (GLOBALVARS %#CLISPARRAY CLISPARRAY CLISPTRANFLG))
    (OR (AND CLISPARRAY (GETHASH X CLISPARRAY))
        (AND CLISPTRANFLG (EQ (CAR X)
                              CLISPTRANFLG)
             (CADR X])

)

(DEFINEQ

(CLISPIF
  [LAMBDA (FORM)                                                 (* bvm%: "21-Nov-86 18:09")
    ;; Translates (IF -- THEN -- ELSEIF -- THEN -- ELSE --) to equivalent COND.
    (COND
       (DWIMIFYFLG (SETQ ATTEMPTFLG T)
              (SETQ CLISPCHANGE T)))
    (PROG ((CLISPCONTEXT 'IFWORD)
           (DWIMIFYING (AND DWIMIFYFLG DWIMIFYING))
           [VARS (OR VARS (AND (NULL DWIMIFYFLG)
                               (GETVARS (OR BREAKFLG (LISTP EXPR]
           (FNSLST0 NOFIXFNSLST0)
           (VARSLST0 NOFIXVARSLST0)
           TEM)
      LP   (COND
             ((NULL DWIMIFYFLG)
              (SETQ NOFIXFNSLST0 NOFIXFNSLST)
              (SETQ NOFIXVARSLST0 NOFIXVARSLST)))
           (SELECTQ (DWIMUNDOCATCH 'CLISPIF0 (SETQ TEM (CLISPIF0 FORM)))
                (:RESPELL ;; A misspelled IF word was detected.  We now go through respellings and make any corrections that occur in FORM.
                          ;; Note that more than one correction may have been involved, e.g. IF FOO XTHENN PRINT X.
                          (COND
                             ((CLISPELL FORM 'IFWORD)
                              (SETQ NOFIXFNSLST0 FNSLST0)
                              (SETQ NOFIXVARSLST0 VARSLST0)
                              ;; The additions made to these lists may be wrong as a result of the misspelling of the IF word, e.g. a variaae kay
                              ;; have appeared in a function slot.
                              (GO LP))))
                (NIL                                             ; error
                     )
                (RETURN TEM))
           (RETDWIM])

(CLISPIF0
  [LAMBDA (FORM)                                                 (* lmm " 4-SEP-83 22:54")
    (PROG (X Y PRED TEM L L0 L-1 CLAUSE DWIMIFYCHANGE $SIDES)
          (SETQ L FORM)
          [AND CLISPIFTRANFLG (SETQ Y (LIST (CONS (CAR L]
          (GO LP0)
      LP  (SELECTQ (CAR L)
                ((IF if)
                     (COND
                        [(EQ L (CDR L-1))
                         ;; No IF's should be seen after the initial one except when immediately following an ELSE.  In this case the two words are
                         ;; treated the same as ELSEIF.
                         (SETQ PRED NIL)
                         (COND
                            (CLISPIFTRANFLG (OR [EQ (CAR L-1)
                                                    (CAR (LISTP (CAR Y]
                                                (SHOULDNT 'ELSE))
                                    (RPLACA Y (SELECTQ (CAAR Y)
                                                       (ELSE (CONS 'ELSEIF))
                                                       (else (CONS 'elseif))
                                                       (SHOULDNT 'ELSE]
                        (T (GO ERROR))))
                ((ELSEIF elseif)
                     (SETQ X (NCONC1 X (CLISPIF1 PRED L0 L FORM)))
                     (SETQ PRED NIL)
                     (AND CLISPIFTRANFLG (SETQ Y (CONS (CONS (CAR L))
                                                       Y))))
                ((ELSE else)
                     (SETQ X (NCONC1 X (CLISPIF1 PRED L0 L FORM)))
                     (SETQ L-1 L)                                ; To enable ELSE IF as two words.
                     (SETQ PRED T)
                     (AND CLISPIFTRANFLG (SETQ Y (CONS (CONS (CAR L))
                                                       Y))))
                ((THEN then)
                     [SETQ PRED (COND
                                   ((EQ L0 L)
                                    (GO ERROR))
```

```
                                (T ;; The reason for doing the LDIFF even when L is (CDR L0) is that can't just set pred to CAR of L is
                                   ;; becuase then couldnt distinguish no predicate from IF NIL THEN -- (Actually encountered by one
                                   ;; user.)
                                   (LDIFF L0 L]
                            [COND
                               (CLISPIFTRANFLG (OR (LISTP (CAR Y))
                                                   (SHOULDNT 'THEN))
                                       (RPLACD (CAR Y)
                                               (CAR L])
                             (GO LP1))
                      LP0 (SETQ L0 (CDR L))
                      LP1 (COND
                             ((SETQ L (CDR L))
                              (GO LP)))
                          (SETQ X (NCONC1 X (CLISPIF1 PRED L0 L FORM)))
                          (AND CLISPIFTRANFLG (SETQ Y (DREVERSE Y)))
                          (/RPLNODE FORM 'COND X)
                          [SETQ $SIDES (CDR (LISTGET1 LISPXHIST 'SIDE]
                          (SETQ L (CDR FORM))
```

;; The COND must appear in the original definition before DWIMIFYing can be done, or else correction of 8 and 9 errors won't work. Some ;; unnecessary work may be done by virtue of DWIMIFYING the whole IF statement, even when it is being evaluated (as opposed to being ;; dwimified). however, in most cases, if the user employs IF, there will be other CLISP constructs in the predicates and consequents.

```
                      LP2 (SETQ CLAUSE (CAR L))
                          (COND
                             [(LISTP (CAR CLAUSE))
                              (DWIMIFY1 (CAR CLAUSE)
                                     'IFWORD)
                              (COND
                                 ([AND (LISTP (CAAR CLAUSE))
                                       (NOT (FNTYP (CAAR CLAUSE]
                                  (LISPXPRIN1 (COND
                                                 ((EQ (CAADAR CLAUSE)
                                                      COMMENTFLG)
                                                  '"misplaced comment
                                                    ")
                                                 (T '"parentheses error
                                                     ")))
                                              T)
                                  (SETQ L FORM)
                                  (GO ERROR]
                             (T (SETQ TEM (CDR CLAUSE))
                                (FRPLACD CLAUSE NIL)
                                (DWIMIFY2 CLAUSE CLAUSE NIL 'IFWORD)
                                (NCONC CLAUSE TEM)))
                          (DWIMIFY2 (SETQ TEM (CDR CLAUSE))
                                 TEM NIL 'IFWORD)
                          (COND
                             ((SETQ L (CDR L))
                              (GO LP2)))
                          (CLISPIF2 FORM)
                          (COND
                             (CLISPIFTRANFLG
                                    ;; Bletcherous PROG here because fool Interlisp-D compiler can't handle MAP2CAR right when inside a BLOCKS
                                    (PROG ((LF (CDR FORM))
                                           (LY Y)
                                           (FIRSTP T)
                                           L)
                                      LP   [COND
                                              ((OR (NLISTP LF)
                                                   (NLISTP LY))
                                               (RETURN (SETQ X (APPLY (FUNCTION NCONC)
                                                                      (DREVERSE L]
                                           (SETQ L (CONS (CLISPIF3 (CAR LF)
                                                                   (CAR LY)
                                                                   FIRSTP)
                                                         L))
                                           (SETQ LF (CDR LF))
                                           (SETQ LY (CDR LY))
                                           (SETQ FIRSTP)
                                           (GO LP))
                                    (SETQ TEM (CONS (CAR FORM)
                                                    (CDR FORM)))          ; the conditional expression, which is now in the function, and is
                                                                          ; going to be smashed
                                    (RPLNODE FORM (CAR X)
                                           (CDR X))                       ; puts the clisp back in /rplnode unnecessary since this was
                                                                          ; already saved above.
                                    [COND
                                       ((AND (EQ (CAAR $SIDES)
                                                 FORM)
                                             (EQUAL (CAAR $SIDES)
                                                    (CDAR $SIDES)))
```

;; so function wont be marked as changed reason for EQUAL check is if it was converted to lower case, than do want to ;; retain side informaton.

```
                        (FRPLACA (CAR $SIDES)
                                 '*]
                    (CLISPTRAN FORM TEM)))
          (RETURN FORM)
      ERROR
          (DWIMERRORRETURN (LIST 4 L FORM])
```

(CLISPIF1

```
  [LAMBDA (PRED L0 L FORM)                                    (* lmm "26-Jul-84 05:01")
    (COND
       (PRED (CONS (COND
                      ((OR (NLISTP PRED)
                           (CDR PRED))
                       PRED)
                      (T (CAR PRED)))
                   (LDIFF L0 L)))
       ((EQ L0 L)
        ;; Note that ELSE or ELSEIF can imediately follow a THEN by virtue of the PRED check in earlier clause.
        (DWIMERRORRETURN (LIST 4 L FORM)))
       ((EQ (CDR L0)
            L)
        (LIST (CAR L0)))
       (T (LIST (LDIFF L0 L])
```

(CLISPIF2

```
  [LAMBDA (X)                                                 (* lmm "16-Sep-85 18:15")
    (PROG (TEM1 TEM2 TEM3)
          (COND
             ((NEQ (CAR X)
                   'COND))
             ((AND (EQ [CADR (SETQ TEM1 (CAR (SETQ TEM2 (FLAST X]
                       X)
                   (EQ (CAR TEM1)
                       T)
                   (NULL (CDDR TEM1)))
              ;; Changes expression of X (COND -- (T (COND **))) to (COND -- **) useful for producing more aesthetic code when the 'DO' portion
              ;; of a 'FOR' statement is an 'IF' Converts
              (/RPLNODE TEM2 (CADR X)
                        (CDDR X)))
             ((AND (EQ (CAR TEM1)
                       T)
                   (EQ [CADR (LISTP (SETQ TEM3 (CAR (SETQ TEM2 (NLEFT X 2]
                       X)
                   (NULL (CDDR TEM2)))                        ; Converts expression of X (COND (& (COND --)) (T **)) to
                                                              ; (COND ((NEGATION &) **) --)
              (/RPLNODE TEM1 (CAR TEM3)
                        (CDR TEM1))
              (/RPLNODE TEM2 TEM1 (CDADR TEM3])
```

(CLISPIF3

```
  [LAMBDA (CLAUSE ORIGWORDPAIR FIRSTCLAUSEFLG)                (* JonL "22-APR-83 19:46")
    (PROG NIL
          (RETURN (CONS [COND
                           [FIRSTCLAUSEFLG (COND
                                              (LCASEFLG 'if)
                                              ((CAR ORIGWORDPAIR))
                                              (T 'IF]
                           [(EQ (CAR CLAUSE)
                                T)
                            (RETURN (CONS (COND
                                             (LCASEFLG 'else)
                                             ((CAR ORIGWORDPAIR))
                                             (T 'ELSE))
                                          (APPEND (CDR CLAUSE]
                           (T (COND
                                 (LCASEFLG 'elseif)
                                 ((CAR ORIGWORDPAIR))
                                 (T 'ELSEIF]
                        (CONS (CAR CLAUSE)
                              (COND
                                 ((CDR CLAUSE)
                                  (CONS (COND
                                           (LCASEFLG 'then)
                                           ((CDR ORIGWORDPAIR))
                                           (T 'THEN))
                                        (APPEND (CDR CLAUSE])

)

(DEFINEQ

(CLISPFOR
```

```
[LAMBDA (FORM)                                                  (* bvm%: "21-Nov-86 18:10")
  ;; Translates iterative statements, e.g., (for X in Y until --)
  (COND
     (DWIMIFYFLG (SETQ ATTEMPTFLG T)
           (SETQ CLISPCHANGE T)))
  (PROG ((CLISPCONTEXT 'FORWORD)
         (DWIMIFYING (AND DWIMIFYFLG DWIMIFYING))
         (VARS VARS)
         TEM)
    LP   (COND
            ((NULL DWIMIFYFLG)
             (SETQ NOFIXFNSLST0 NOFIXFNSLST)
             (SETQ NOFIXVARSLST0 NOFIXVARSLST)))
         (SELECTQ (DWIMUNDOCATCH 'CLISPFOR0 (SETQ TEM (CLISPFOR0 FORM)))
              (:RESPELL ;; A misspelled I.S. word was detected.  We now go through respellings and make any corrections that occur in FORM.
                        ;; Note that more than one correction may have been involved, e.g. FOR X IN YWHILLE Z FOO XTHENN PRINT X.
                        (COND
                           ((CLISPELL FORM 'FORWORD)
                            (GO LP))))
              (NIL                                              ; error
                   )
              (RETURN TEM))
         (RETURN])
```

(CLISPFOR0

```
[LAMBDA (EXP)                                                   ; Edited 14-Sep-2022 10:24 by rmk
                                                                (* rmk%: " 6-Oct-84 12:03")
  (DECLARE (SPECVARS EXP))
  (PROG (DWIMIFYCHANGE (I.S. EXP)
               I.S.TYPE LASTPTR I.S.PTRS I.S.BODY OPR TEM I.S.TYPE1 I.V. FIRSTI.V. IVINITFLG PROGVARS INITVARS
               MAKEPROGFLG TERMINATEFLG TERM ITER LSTVAR (LSTVARS '($$LST1 $$LST2 $$LST3 $$LST4 $$LST5 $$LST6)
                                                                 )
               (DUMMYVARS CLISPDUMMYFORVARS)
               EXCEPTPREDS RETPREDS AFTERPREDS RETEXP OUTEXP UNDOLST FOR BIND DECLARELST AS FROM TO IN ON BY
               FINALLY EACHTIME FIRST CLISPWORD (VARS (APPEND '(I.V. BODY $$VAL)
                                                              VARS))
               I.S.OPRSLST I.S.OPR)
        (DECLARE (SPECVARS LASTPTR I.S.PTRS))                   ; Used freely by I.S.OPRS in IDL -- Ron
        (COND
           ((NULL DWIMIFYFLG)
            (SETQ NOFIXFNSLST0 NOFIXFNSLST)
            (SETQ NOFIXVARSLST0 NOFIXVARSLST)))
    LP  (COND
           ([NOT (LITATOM (SETQ OPR (CAR I.S.]
            (GO LP2)))
    RECHECK
        (COND
           ([NULL (SETQ CLISPWORD (GETPROP OPR 'CLISPWORD]
            (GO LP2))
           ((OR (NLISTP CLISPWORD)
                (NEQ (CAR CLISPWORD)
                     'FORWORD))                                 ; E.g. OR, AND,
            (GO LP2)))
        [AND LCASEFLG (EQ OPR (CAR I.S.))
             ([LAMBDA (LC)
                 ;; Replaces the uppercase word with the lowercase.  the EQ check is so that synonyms are not replaced by their antecedents.
                 ;; the NEQ check so that the replacement is not done when it is already in lowercase.
                 (AND (NEQ LC (CAR I.S.))
                      (/RPLNODE I.S. LC (CDR I.S.]
              (COND
                 ((NLISTP (CDR CLISPWORD))
                  (CDR CLISPWORD))
                 (T (CADR CLISPWORD]
        (COND
           ((EQ (GETP (CDR CLISPWORD)
                      'I.S.OPR)
                'MODIFIER)                                      ; modifier
            (GO LP2)))
        (COND
           ((AND LASTPTR (NULL (CDDR LASTPTR)))
            ;; X identifies the end of the operand for the previous i.s.opr needs to be done before the caal to CLISPFOR0A because it might
            ;; return a new X with some OTHERS in front.  e.g. if the i.s.  is (FOR X IN Y SUM X + 1 WHILE Z), at the time the WHILE is
            ;; encountered, the range of the opeand for SUM is X + 1 after the call to CLISPISTYPE, x will be (FIRST (SETQ $$VAL 0) WHILE Z)
            (NCONC1 LASTPTR I.S.)))
        (COND
           (I.S.OPR (SETQ I.S. (CLISPFOR0A I.S.OPR I.S. LASTPTR))
                                                                ; see comment at end of selectq
                    (SETQ I.S.OPR NIL)
                    (GO LP)))
        (COND
           ((NLISTP (CDR CLISPWORD))
            ;; This converts everything to the lowercase version thereby simplifying the selectq.  (There is no information tored to enable getting
```

```
            ;; back to uppercase from lowercase (using properties) so that lowercase is the only available canonical representation.)
             (SETQ OPR (CDR CLISPWORD)))
            (T                                                    ; This implements synonyms, e.g. WHERE is the same as
                                                                  ; WHEN.
               (SETQ OPR (CADDR CLISPWORD))
               (GO RECHECK)))
         (COND
            ((EQ OPR 'original)
             (GO SELECT))
            [(EQ (CAR LASTPTR)
                 'ORIGINAL)
             (COND
                ((EQ (CADDR LASTPTR)
                     (CDADR LASTPTR))
                 (GO SELECT))
                (T (CLISPFORERR (CADR LASTPTR)
                          (CADDR LASTPTR)
                          'MISSING]
            ([LISTP (SETQ I.S.OPR (GETPROP OPR 'I.S.OPR]
             [COND
                [(NULL (CAR I.S.OPR))
                 ;; The i.s.type does not define the i.s.type for the i.s.  e.g. Larry's UPTO which is defined as (BIND $$MAX_BODY TO $$MAX)
                 (COND
                    ((NULL (CDR I.S.OPR))
                     ;; the i.s.opr terminates (terminted) the scope of the prvious i.s.opr, but is otherwise a nop, i.e. invisible.  this featre is used
                     ;; for i.s.oprs in which one does not want the argument dwimified, but wants a postprocessor to handle it, e.g. (for i from 1
                     ;; to 10 decl (x fixp) do (foo))
                     (SETQ LASTPTR (LIST NIL I.S.)))
                    (T (SETQ LASTPTR (LIST OPR I.S.]
                [(NULL I.S.TYPE)                                  ; e.g. COLLECT.  DO, JOIN, SUM ETC.
                 (SETQ I.S.TYPE1 OPR)
                 (SETQ LASTPTR (LIST 'I.S.TYPE (SETQ I.S.TYPE I.S.]
                ((AND (EQ I.S.TYPE1 'do)
                      (EQ I.S. (CDR I.S.TYPE)))                   ; E.g. DO COLLECT, DO JOIN.  Ignore the DO
                 (SETQ I.S.TYPE1 OPR)
                 (/RPLNODE I.S.TYPE (CAR I.S.)
                        (CDR I.S.))
                 (FRPLACD (CDR LASTPTR)))
                ((AND (EQ I.S.TYPE1 'thereis)
                      (OR (EQ (CAR I.S.)
                              'SUCHTHAT)
                          (EQ (CAR I.S.)
                              'suchthat))
                      (NULL FOR))                                 ; special glitch to allow ISTHERE -- SUCHTHAT --
                 (SETQ I.S.TYPE1 OPR)
                 (SETQ LASTPTR (LIST 'I.S.TYPE (SETQ I.S.TYPE I.S.)))
                 (SETQ OPR (FASSOC 'I.S.TYPE I.S.PTRS))
                 (FRPLACA OPR 'FOR)
                 (SETQ FOR (CADR OPR)))
                (T (CLISPFORERR I.S.TYPE I.S. 'BOTH]
             (GO LP0))
            ((GETP OPR '\DURATIONTRAN)                            ; Foo, punt out by calling \DURATIONTRAN since it's too
                                                                  ; complicated to express as an I.S.OPRS
             (SETQ I.S. (\DURATIONTRAN EXP))
             (GO OUT)))
      SELECT
          [SELECTQ OPR
               (original (SETQ LASTPTR (LIST 'ORIGINAL I.S.)))
               ((for from in on to by)
                    [SETQ LASTPTR (SELECTQ OPR
                                          (for (LIST 'FOR (SETQ FOR I.S.)))
                                          (from (AND (SETQ OPR (OR IN ON))
                                                     (CLISPFORERR OPR I.S. 'BOTH))
                                                (LIST 'FROM (SETQ FROM I.S.)))
                                          (in (AND (SETQ OPR (OR FROM TO ON))
                                                   (CLISPFORERR OPR I.S. 'BOTH))
                                              (LIST 'IN (SETQ IN I.S.)))
                                          (on (AND (SETQ OPR (OR FROM TO IN))
                                                   (CLISPFORERR OPR I.S. 'BOTH))
                                              (LIST 'ON (SETQ ON I.S.)))
                                          (to (AND (SETQ OPR (OR IN ON))
                                                   (CLISPFORERR OPR I.S. 'BOTH))
                                              (LIST 'TO (SETQ TO I.S.)))
                                          (by (LIST 'BY (SETQ BY I.S.)))
                                          (SHOULDNT 'CLISPFOR0]
                     (GO TWICECHECK))
               (as (OR TERMINATEFLG (SETQ TERMINATEFLG (OR IN ON RETPREDS TO)))
                   (COND
                      ((OR FOR AS I.V.))
                      ((OR FROM IN ON TO)                         ; E.g. IN X AS I FROM 1 TO 10 DO --.
                       (SETQ FIRSTI.V. (SETQ I.V. (GETDUMMYVAR T)))
                                                                  ; getdummyvar also adds to progvars and vars

                       ))
                   (SETQ IN NIL)
```

```
                (SETQ ON NIL)
                (SETQ FROM NIL)
                (SETQ TO NIL)
                (SETQ BY NIL)
                ;; Primarily for error detection, i.e. now can just check to see if say both IN/ON and FRM appear in one stretch.
                (SETQ LASTPTR (LIST 'AS (SETQ AS I.S.))))
            (bind (SETQ LASTPTR (LIST 'BIND (SETQ BIND I.S.))))
            (declare (SETQ DECLARELST (CONS (SETQ LASTPTR (LIST 'DECLARE I.S.))
                                            DECLARELST)))
            (while ;; WHILE, UNTIL, UNLESS< WHEN, Finally, FIRST, and EACHTIME can appear more than once.  the corresponding
                   ;; FORPTR'S are gathered on a list and processed by a call to either CLISPFOR2 for the first four, and CLISPFOR3 for
                   ;; latter three (which can have imlicit progns as well.)
                   (SETQ RETPREDS (CONS (SETQ LASTPTR (LIST 'WHILE I.S.))
                                        RETPREDS)))
            (until (SETQ RETPREDS (CONS (SETQ LASTPTR (LIST 'UNTIL I.S.))
                                        RETPREDS)))
            (repeatwhile                                          ; Like WHILE except test is mae after body of iterative statement.
                         (SETQ AFTERPREDS (CONS (SETQ LASTPTR (LIST 'REPEATWHILE I.S.))
                                                AFTERPREDS)))
            (repeatuntil (SETQ AFTERPREDS (CONS (SETQ LASTPTR (LIST 'REPEATUNTIL I.S.))
                                                AFTERPREDS)))
            (unless (SETQ EXCEPTPREDS (CONS (SETQ LASTPTR (LIST 'UNLESS I.S.))
                                            EXCEPTPREDS)))
            (when (SETQ EXCEPTPREDS (CONS (SETQ LASTPTR (LIST 'WHEN I.S.))
                                          EXCEPTPREDS)))
            (finally (SETQ FINALLY (CONS (SETQ LASTPTR (LIST 'FINALLY I.S.))
                                         FINALLY)))
            (eachtime (SETQ EACHTIME (CONS (SETQ LASTPTR (LIST 'EACHTIME I.S.))
                                           EACHTIME)))
            (first (SETQ FIRST (CONS (SETQ LASTPTR (LIST 'FIRST I.S.))
                                     FIRST)))
            (COND
               ((EQ I.S.OPR 'MODIFIER)                            ; e.g. OLD
                (FRPLACD (CDR LASTPTR))                           ; The OLD does not terminate the scope of the previous i.s.
                (GO LP2))
               (T (GO LP2]
        (GO LP0)
    TWICECHECK
        (AND (SETQ TEM (FASSOC (CAR LASTPTR)
                               I.S.PTRS))
             (NULL AS)
             (CLISPFORERR (CADR TEM)
                    (CADR LASTPTR)
                    'TWICE))
    LP0 (SETQ I.S.PTRS (NCONC1 I.S.PTRS LASTPTR))
    LP1 (COND
           ((AND (NULL (CDR I.S.))
                 (EQUAL EXP (CAR HISTENTRY)))
            (PRIN1 '|...| T)
            (PEEKC T)
            (NCONC EXP (READLINE T))
            (GO LP0)))
    LP2 (COND
           ((LISTP (SETQ I.S. (CDR I.S.)))
            (GO LP))
           (I.S.                                                  ; i.s. ends in a non-nil tail
                 (AND (NULL DWIMESSGAG)
                      (ERRORMESS (LIST 25 EXP)))
                 (ERROR!))
           (LASTPTR (NCONC1 LASTPTR NIL))
           ((NULL I.S.PTRS)                                       ; shouldnt happen
            (AND (NULL DWIMESSGAG)
                 (ERRORMESS1 "No operator in:" EXP))
            (ERROR!)))
        [COND
           (I.S.OPR (SETQ I.S. (CLISPFOR0A I.S.OPR NIL LASTPTR))
                    (SETQ I.S.OPR NIL)
                    (COND
                       (I.S. (GO LP]
        (SETQ TEM VARS)
        [MAP I.S.PTRS (FUNCTION (LAMBDA (PTRS)
                                 (SETQ PROGVARS (SELECTQ (CAAR PTRS)
                                                         ((BIND AS)
                                                              (APPEND PROGVARS (CLISPFORVARS PTRS)))
                                                         (FOR
                                                           ;; The reason for the reverse in order in the APPEND beloow is in caseBIND appears before FOR,
                                                           ;; and a PROG is not being made, must have FOR variables first.  NOte if a prog is being made, it
                                                           ;; doesnt matter.                ; The call to CLISPFORVARS will also set I.V. and FIRSTI.V.
                                                                (APPEND (CLISPFORVARS PTRS)
                                                                        PROGVARS))
                                                         ((IN ON)
                                                              (PROG [(VARS (COND
                                                                             (I.S.TYPE TEM)
                                                                             (T VARS]
                                                                    (CLISPFOR1 PTRS T))
```

```
                                        ;; IN/ON should be handled before adding VARS because that is when its operand is evaluqted.
                                        ;; (Except when there is no FOROPR, because we really might be DWIMIFYING what will be the
                                        ;; FOROPR.)

                                                                PROGVARS)
                                                            PROGVARS]

;; Need to do this before CLISPFOR1 to get all ofthe variables 'bound' i.e. added to rs, and to note the names of the i.v. (s)

      [COND
         ((AND (NULL I.V.)
                (OR FROM IN ON TO))
          ;; This can only occur if there is no FOR and no AS.  If thee is no FOR and an AS, the I.V. for the initial segment, if one is needed, is
          ;; set up in the SELECTQ at LP.
          (SETQ I.V. (GETDUMMYVAR T]
      (SETQ TEM I.S.PTRS)
  LP3 (COND
         ((SETQ TEM (CLISPFOR1 TEM))
          ;; maps down forpotrs applying clispfor1 to each one.  for most calls, clispfor1 returns CDR of TEM, but for things on i.s.typelst, it
          ;; jumps ahead and does the next few before finishing up this one so it can substitute.
          (GO LP3)))
      [SETQ I.S.BODY (AND I.S.TYPE (COND
                                      [(NLISTP (CAR I.S.TYPE))
                                       (LIST (COND
                                                ((AND (EQ [CAR (SETQ TEM (LISTP (GETPROP (CADR I.S.TYPE)
                                                                                         'CLISPWORD]
                                                                  'FORWORD)
                                                              (EQ (GETPROP (CDR TEM)
                                                                           'I.S.OPR)
                                                                  'MODIFIER))
                                                       (CADDR I.S.TYPE))
                                                      (T (CADR I.S.TYPE]
                                      (T
                                        ;; This occurs when the FOROPR specifies more than one operation, i.e. an implicit PROGN.  In this
                                        ;; case, FOROPR was reset to the body of the PROGN.
                                           (CAR I.S.TYPE]
      (COND
         ((OR RETPREDS AFTERPREDS)
          (GO MAKEPROG))
         ((NULL I.S.TYPE)
          (AND (NULL DWIMESSGAG)
               (ERRORMESS1 '"No DO, COLLECT, or JOIN in:" EXP))
          (ERROR!))
         (TO (GO MAKEPROG))
         ((AND (NULL IN)
               (NULL ON))
          (COND
             ([AND (NULL DWIMESSGAG)
                   (NULL TERMINATEFLG)
                   (NULL (CLISPFOR4 (GETPROP I.S.TYPE1 'I.S.OPR]
              ;; Before printing this message, check I>S>TYPE for possilb RETURN or GO, as with THEREIS, SUCHTHAT, etc.
              (PRIN1 '"Possible non-terminating iterative statement:
                      " T)
              (PRINT [MAPCAR EXP (FUNCTION (LAMBDA (I.S.)
                                             (RETDWIM2 I.S. NIL 1]
                     T T)))
          (GO MAKEPROG))
         ([OR FROM AS (CDR PROGVARS)
              INITVARS MAKEPROGFLG FINALLY FIRST EACHTIME [NOT (FMEMB I.S.TYPE1 '(collect join do]
              EXCEPTPREDS
              (AND ON (EDITFINDP I.S.BODY (LIST 'SETQ I.V. '&]
          ;; On TYPE-IN?  do not convert to MAPCONC, i.e. convert to a PROG, as otherwise the MAPCONC would be converted toa
          ;; /MAPCONC, which is unnecessary.
          (GO MAKEPROG)))
      [SETQ I.S. (CONS [COND
                          [IN (SELECTQ I.S.TYPE1
                                  (subset 'SUBSET)
                                  (collect 'MAPCAR)
                                  ((JOIN join)
                                       (CLISPLOOKUP 'MAPCONC (CADR IN)))
                                  ((DO do)
                                       'MAPC)
                                  (SHOULDNT 'CLISPFOR0]
                          [ON (SELECTQ I.S.TYPE1
                                   (collect 'MAPLIST)
                                   (join (CLISPLOOKUP 'MAPCON (CADR ON)))
                                   (do 'MAP)
                                   (SHOULDNT 'CLISPFOR0]
                          (T (SHOULDNT 'CLISPFOR0]
                       (CONS (CADR (OR IN ON))
                             (LIST (CLISPFORF/L I.S.BODY PROGVARS DECLARELST]
      (COND
         (BY (NCONC1 I.S. (CLISPFORF/L (LIST (SUBST I.V. (CADR (OR IN ON))
```

```
                                                    (CADR BY)))
                                    PROGVARS DECLARELST))
              ;; The reason for the subst is the manual says you can refer to the current tail in a BY by using either the I.V> or the operand to
              ;; IN/ON.  This normalizes it to I>V., which is always (CAR PROGVARS).  NOte similar operation in SUBPAIR about 3 pages
              ;; from here.

              ))
        (GO OUT)
    MAKEPROG
        [COND
           ([AND (EQ I.S.TYPE1 'collect)
                 (SETQ I.S. (GETPROP 'fcollect 'I.S.OPR]
            ;; This is the form for MAPCAR used by the compiler.  Its advantage is it doesnt call NCONC1 and results in no extra function calls.
            ;; User can disable this by removing the property of FCOLLECT.
            [SETQ PROGVARS (APPEND PROGVARS (SETQ TEM (LISTGET1 (CDR I.S.)
                                                                'BIND]
            (SETQ VARS (APPEND VARS TEM)))
           ((NULL I.S.TYPE1)
            (GO MP0))
           ([NULL (SETQ I.S. (GETPROP I.S.TYPE1 'I.S.OPR]
            (SHOULDNT 'CLISPFOR0]
        [COND
           [(EQ (CAAR I.S.)
                '=)
            (SETQ I.S. (EVAL (CDAR I.S.]
           (T (SETQ I.S. (CAR I.S.]
        [SETQ I.S.BODY (SUBPAIR '(BODY I.V.)
                                (LIST (COND
                                         ((CDR I.S.BODY)
                                          (CONS 'PROGN I.S.BODY))
                                         (T (CAR I.S.BODY)))
                                      (OR FIRSTI.V. I.V.))
                                (COND
                                   ((LISTP I.S.)
                                    (DWIMIFY1 (COPY I.S.)))
                                   (T                                ; For DO, its just BODY.
                                      I.S.]
        [SETQ I.S.BODY (COND
                           ((EQ (CAR I.S.BODY)
                                'PROGN)
                            (APPEND (CDR I.S.BODY)))
                           (T (LIST I.S.BODY]                        ; FORBODY is now a list of forms.
        (CLISPFOR4 I.S.BODY)                                         ; Checks for GO's so know where you need an $$OUT typeof
                                                                     ; structure.
    MP0 [COND
           ((NOT (FASSOC '$$VAL PROGVARS))
            (SETQ PROGVARS (CONS '$$VAL PROGVARS]
        [SETQ RETEXP (LIST (LIST 'RETURN '$$VAL]
        (COND
           ((NULL AS)
            (GO NX)))
        (SETQ I.V. FIRSTI.V.)
    MP1 (SETQ IN NIL)
        (SETQ ON NIL)
        (SETQ FROM NIL)
        (SETQ TO NIL)
        (SETQ BY NIL)
    MP2 (SELECTQ (CAAR I.S.PTRS)
             (FROM (SETQ FROM (CADAR I.S.PTRS)))
             (BY (SETQ BY (CADAR I.S.PTRS)))
             (IN (SETQ IN (CADAR I.S.PTRS)))
             (ON (SETQ ON (CADAR I.S.PTRS)))
             (TO (SETQ TO (CADAR I.S.PTRS)))
             (AS (GO NX))
             NIL)
        (COND
           ((SETQ I.S.PTRS (CDR I.S.PTRS))
            (GO MP2)))
    NX  (SETQ LSTVAR (CAR LSTVARS))
        (COND
           ((OR IN ON)
            (SETQ TEM (CADR (OR IN ON)))
            [COND
               [(AND [COND
                        [(OR (EQ TEM 'OLD)
                             (EQ TEM 'old))                          ; IN OLD --
                         (SETQ TEM (CADDR (OR IN ON]
                        ((OR (EQ (CAR TEM)
                                 'OLD)
                             (EQ (CAR TEM)
                                 'old))                              ; IN (OLD --)
                         (SETQ TEM (CADR TEM]
                     (COND
                        ((LITATOM TEM)                               ; IN OLD X or IN (OLD X)
                         (SETQ LSTVAR TEM))
                        ((OR (EQ (CAR TEM)
```

```
                              'SETQ)
                          (EQ (CAR TEM)
                              'SETQQ))                    ; IN OLD X _ ..  or IN (OLD X _ ..), or IN OLD (X _ ..) or IN (OLD
                                                          ; (X _ ..))
                      (CLISPFORINITVAR (SETQ LSTVAR (CADR TEM))
                              (CADDR TEM)))
                     (T (SHOULDNT 'CLISPFOR0]
            (ON                                           ; Normal case, no 'OLD'.  No need for dummy variable for ON.
                (SETQ LSTVAR I.V.)
                (CLISPFORINITVAR I.V. TEM))
            (T (SETQ PROGVARS (CONS (LIST LSTVAR TEM)
                                    PROGVARS]
        [COND
            ((EQ I.V. LSTVAR)
             (SETQ RETPREDS (NCONC1 RETPREDS (LIST 'NLISTP LSTVAR)))
                                                          ; put it on last so when it is revrsed by CLISPFOR2 will come out
                                                          ; first.

            )
            (T (SETQ TERM (NCONC1 TERM (LIST 'SETQ I.V.
                                             (COND
                                                [IN

                                    ;; reason for checking here rather in retpreds is to avoid user setting a pointer to garbage, e.g. (FOR
                                    ;; OLD X IN (QUOTE (19 . 20)) DO PRINT) would leave X set to (CAR 20) otherwise

                                                   (SETQ MAKEPROGFLG T)
                                                         ; to make sure that a $$OUT gets added
                                                   (SUBST LSTVAR 'VAR '(CAR (OR (LISTP VAR)
                                                                                (GO $$OUT]
                                                (T (SETQ RETPREDS (NCONC1 RETPREDS (LIST 'NLISTP LSTVAR)))
                                                   LSTVAR]
        [SETQ ITER (NCONC1 ITER (CONS 'SETQ (CONS LSTVAR
                                                  (COND
                                                     [BY (SUBPAIR (LIST I.V. (CADR (OR IN ON)))
                                                                  (LIST LSTVAR LSTVAR)
                                                                  (LIST (CADR BY]
                                                     (T (LIST (LIST 'CDR LSTVAR]
        (GO BUILDIT)))
    (COND
       (FROM [COND
                  [(SETQ TEM (FMEMB I.V. PROGVARS))
                   (RPLACA TEM (LIST I.V. (CADR FROM]
                  (T (CLISPFORINITVAR I.V. (CADR FROM]
                ;; the reason for IVINITFLG (instead of simply searching the PROGVAR lst) is that the iv may bbe an OLD variable and it
                ;; wont appear anywhere.  neverhtless need to know if it is being initialized, because in case of TO, it must be initialzed to 1 if
                ;; not.
                (SETQ IVINITFLG T)))
    [COND
       (TO [SETQ TEM (COND
                          [(NUMBERP (CADR BY))
                           (COND
                              ((MINUSP (CADR BY))
                               'LT)
                              (T 'GT]
                          [BY                             ; RMK 2022:  Removed call to WARNUSER
                              [SETQ BY (LIST 'BY (LIST 'SETQ (GETDUMMYVAR T)
                                                       (CADR BY]
                              (LIST 'AND (CAR PROGVARS)
                                    (LIST 'OR (LIST 'ZEROP (CAR PROGVARS))
                                          (LIST 'COND (LIST (LIST 'MINUSP (CAR PROGVARS))
                                                            (LIST (CLISPLOOKUP 'LT I.V.)
                                                                  I.V.
                                                                  (CADR TO)))
                                                      (LIST T (LIST (CLISPLOOKUP 'GT I.V.)
                                                                    I.V.
                                                                    (CADR TO]
                          ((AND (FIXP (CADR FROM))
                                (FIXP (CADR TO))
                                (ILESSP (CADR TO)
                                        (CADR FROM)))
                           (SETQQ BY (BY -1))
                           'LT)
                          (T 'GT]
            [COND
               ((NULL IVINITFLG)
                (SETQ INITVARS (NCONC1 INITVARS (LIST 'SETQ I.V. 1]
            (SETQ PROGVARS (CONS (LIST (GETDUMMYVAR)
                                       (CADR TO))
                                 PROGVARS))               ; RMK 2022:  Remove call to WARNUSER
            (SETQ RETPREDS (NCONC1 RETPREDS (COND
                                               ((NLISTP TEM)
                                                (LIST (CLISPLOOKUP TEM I.V.)
                                                      I.V.
                                                      (CAAR PROGVARS)))
                                               (T TEM]
    [COND
```

```
            ((OR BY FROM TO)
             (SETQ ITER (NCONC1 ITER (LIST 'SETQ I.V. (COND
                                                ((OR FROM TO (NUMBERP (CADR BY)))
                                                 (LIST (CLISPLOOKUP '+ I.V.)
                                                       I.V.
                                                       (OR (CADR BY)
                                                           1)))
                                                (T (CADR BY]
      BUILDIT
          (COND
             ((AND AS I.S.PTRS)
              (SETQ TEM (CDDDAR I.S.PTRS))
              (SETQ I.V. (CAR TEM))
              (SETQ IVINITFLG (CADR TEM))
              (SETQ I.S.PTRS (CDR I.S.PTRS))
              (AND (NULL (SETQ LSTVARS (CDR LSTVARS)))
                   (CLISPFORERR '"too many concurrent loops
                                "))
              (GO MP1)))
          [COND
             (FINALLY (SETQ TEM (CLISPFOR3 FINALLY))
                    (SETQ RETEXP (COND
                                    ((EQ (CAAR (FLAST TEM))
                                         'RETURN)
                                     TEM)
                                    (T (NCONC TEM RETEXP]
          [COND
             ((OR MAKEPROGFLG (AND RETPREDS AFTERPREDS))
              (SETQ OUTEXP (CONS '$$OUT RETEXP))
              (SETQ RETEXP (LIST (LIST 'GO '$$OUT]
          [COND
             ((SETQ AFTERPREDS (CLISPFOR2 AFTERPREDS))
              (SETQ AFTERPREDS (LIST (LIST 'COND (CONS (COND
                                                ((CDR AFTERPREDS)
                                                 (CONS 'OR AFTERPREDS))
                                                (T (CAR AFTERPREDS)))
                                               RETEXP]
          [COND
             ((SETQ RETPREDS (CLISPFOR2 RETPREDS))
              (SETQ RETPREDS (CONS (COND
                                      ((CDR RETPREDS)
                                       (CONS 'OR RETPREDS))
                                      (T (CAR RETPREDS)))
                                   RETEXP]
          [COND
             ((SETQ EXCEPTPREDS (CLISPFOR2 EXCEPTPREDS))
              [SETQ EXCEPTPREDS (LIST (COND
                                         ((CDR EXCEPTPREDS)
                                          (CONS 'OR EXCEPTPREDS))
                                         (T (CAR EXCEPTPREDS)))
                                      (LIST 'GO '$$ITERATE]
              (SETQ I.S.BODY (CONS (COND
                                      (RETPREDS (LIST 'COND RETPREDS EXCEPTPREDS))
                                      (T (LIST 'COND EXCEPTPREDS)))
                                   I.S.BODY)))
             (RETPREDS (SETQ I.S.BODY (CONS (LIST 'COND RETPREDS)
                                            I.S.BODY]
          [SETQ I.S. (CONS 'PROG (CONS PROGVARS (NCONC [AND DECLARELST
                                                       (LIST (CONS 'DECLARE (MAPCONC
                                                                             (DREVERSE DECLARELST)
                                                                             (FUNCTION (LAMBDA (X)
                                                                                       (LDIFF
                                                                                        (CDADR X)
                                                                                        (CADDR X]
                                                INITVARS
                                                (AND FIRST (CLISPFOR3 FIRST))
                                                (CONS '$$LP (NCONC TERM (AND EACHTIME (CLISPFOR3 EACHTIME
                                                                                                 ))
                                                                   I.S.BODY
                                                                   (LIST '$$ITERATE)
                                                                   AFTERPREDS ITER
                                                                   (LIST (LIST 'GO '$$LP))
                                                                   OUTEXP]
      OUT [SETQ TEM (CDR (LISTGET1 LISPXHIST 'SIDE]              ; TEM holds a list of side info

      ;; Restores those places where I.V.'s where stuck in, e.g. FOR X IN Y COLLECT FOO was temporarily converted to FOR X IN Y COLLECT (FOO
      ;; X), and IN Y COLLECT FOO would have been chaged to IN Y COLLECT (FOO $$TEM)

          [MAPC UNDOLST (FUNCTION (LAMBDA (X)
                                    (FRPLACA (CAR X)
                                             (CADR X))
                                    (FRPLACD (CAR X)
                                             (CDDR X))
                                    (COND
                                       ((SETQ X (FASSOC (CAR X)
                                                        TEM))     ; to tell dwimnewfile?  thatthis change was undone, so not to
                                                                  ; count the function as being changed
                                        (FRPLACA X '*]
```

```
          (CLISPTRAN EXP I.S.)
          (RETURN EXP])
```

(CLISPFOR0A

```
  [LAMBDA ($I.S.OPR I.S. LASTPTR)                                    (* rmk%: " 6-Oct-84 12:11")
```

;; Thisfunction is called when we hit the first i.s.opr following one defined via an istype property. The problems with such operaaors is that we ;; cannot dwiify their operands (or any operands in the i.s.) until we have scanned the entire i.s. and found aal the VARS. This requires that we ;; obtain the definitions of each i.s.opr from its property list, since there may be BIND's in the defiition. However, we cannot substiute in the ;; operands until after we dwimify the operands, since otherwise any errors corrected in the operands wont be seen in the original i.s. when the ;; user prints it after it is dwimified. Furthermore, if we substitute in before we dwimify, we cant distinguish the case where the usr writes a $$VAL, ;; thereby requiring a PROG in the translation, from that where a $$VAL is specified in the definition for the i.s.opr e.g. for COLLECT or JOIN, but ;; nevertheless it is ok to translate to a mapping function. Therefore we insert the definition and take note of thoe things requiring substiution later. ;; and furthermore leave in the original i.s.opr so its operand can also be dwimified.

```
   (DECLARE (SPECVARS LASTPTR))                                      ; Used freely by IS.OPRS in IDL -- Ron
   [COND
      ((CDR (LISTP $I.S.OPR))
```

;; OTHERS. Note that an i.s.opr defned by an i.s.opr property can specify an i.s.type, OTHERS, or both.

```
       (SETQ I.S.OPRSLST (CONS LASTPTR I.S.OPRSLST))
       (SETQ I.S. (NCONC [COPY (COND
                                  ((EQ (CADR $I.S.OPR)
                                       '=)
                                   (EVAL (CDDR $I.S.OPR)))
                                  (T (CDR $I.S.OPR]
                         I.S.]
   I.S.])
```

(CLISPFOR1

```
  [LAMBDA (PTRS FLG)                                                 (* wt%: "28-APR-80 16:11")
    (PROG ((OPRTAIL (CADAR PTRS))
           BODYTAIL
           (NXTOPRTAIL (CADDAR PTRS))
           Z TEM LSTFLG BODY)
```

;; X is the TAIL of the iterative statement beginning with the operator, Y the tail beginning with the next opeator.

```
          (SELECTQ (CAAR PTRS)
               ((FOR BIND DECLARE ORIGINAL NIL)
                    (GO OUT))
               ((IN ON)
                    (AND (NULL FLG)
                         (GO OUT))                                   ; Already done.

                    )
               (AS (SETQ I.V. (CADDDR (CAR PTRS)))
                   (GO OUT))
               NIL)
          [SETQ BODYTAIL (COND
                            ((OR (EQ (CADR OPRTAIL)
                                     'OLD)
                                 (EQ (CADR OPRTAIL)
                                     'old))
                             (OR MAKEPROGFLG (SETQ MAKEPROGFLG T))
                             (CDDR OPRTAIL))
                            ((AND (EQ [CAR (SETQ TEM (LISTP (GETPROP (CADR OPRTAIL)
                                                                     'CLISPWORD]
                                                     'FORWORD)
                                  (EQ (GETPROP (CDR TEM)
                                               'I.S.OPR)
                                      'MODIFIER))
                             (CDDR OPRTAIL))
                            ((CDR OPRTAIL))
                            (T
                               ;; special kluge to allow an i.s.opr to smash lastptr to indicate that this operator/operand is to be ignored, e.g.
                               ;; for handling (EVERY CHARACTER IN Z IS --)
                               (GO OUT]
          (COND
             ((EQ BODYTAIL NXTOPRTAIL)                               ; 2 FORWORDS in a row.
              (CLISPFORERR OPRTAIL NXTOPRTAIL 'MISSING))
             ((NEQ (CDR BODYTAIL)
                   NXTOPRTAIL)                                       ; More than one expression between two forwords.
              (GO BREAK)))
          [COND
             ((NLISTP (CAR BODYTAIL))
              (COND
                 ([AND (NEQ (CAAR PTRS)
                            'FROM)
                       (NEQ (CAAR PTRS)
                            'IN)
                       (NEQ (CAAR PTRS)
                            'ON)
                       (NEQ (CAAR PTRS)
                            'TO)
                       (SETQ Z (CLISPFUNCTION? BODYTAIL 'NOTVAR]
                                                                     ; E.G. DO PRINT, BY SUB1, etc.
```

```
                [COND
                   ((NULL (SETQ TEM (OR FIRSTI.V. I.V.)))
                    (CLISPFORERR OPRTAIL NIL 'WHAT))
                   ((EQ (COND
                           ((EQ OPRTAIL I.S.TYPE)
                            TEM)
                           (T (SETQ TEM I.V.)))
                        (CAR DUMMYVARS))                          ; In the case that an i.v. was supplied, make the change
                                                                  ; permanent. For $$TEM, undo it later.
                    (SETQ UNDOLST (CONS (CONS BODYTAIL (CONS (CAR BODYTAIL)
                                                             (CDR BODYTAIL)))
                                        UNDOLST]
                   (/RPLNODE BODYTAIL (LIST Z TEM)
                             (CDR BODYTAIL)))
                  (T (DWIMIFY2 BODYTAIL OPRTAIL BODYTAIL T T)))
            (SETQ Z (CAR BODYTAIL))
            (GO C))
           ((OR (EQ (CAAR BODYTAIL)
                    'OLD)
                (EQ (CAAR BODYTAIL)
                    'old))
            (OR MAKEPROGFLG (SETQ MAKEPROGFLG T))
            (DWIMIFY2 (CDAR BODYTAIL)
                   (CAR BODYTAIL)
                   T)
            (SETQ Z (CAR BODYTAIL))
            (GO C))
           (T (DWIMIFY1 (CAR BODYTAIL)
                     NIL T)
              (SETQ Z (CAR BODYTAIL))
              (COND
                 ([AND (LISTP (CAAR BODYTAIL))
                       (NOT (FNTYP (CAAR BODYTAIL]
                  (SETQ LSTFLG T)
                  (GO A))
                 (T (GO C]
    BREAK
        (COND
           (NXTOPRTAIL (CLISPRPLNODE (SETQ Z (NLEFT OPRTAIL 1 NXTOPRTAIL))
                                     (CAR Z)
                                     NIL)))                       ; Breaks the list justbefore the next operator.
        (CLISPRPLNODE BODYTAIL (SETQ Z (CONS (CAR BODYTAIL)
                                             (CDR BODYTAIL)))
              NXTOPRTAIL)
```

;; Puts parentheses in --- E.g. For X in FOO Y do -- becomes for X in (FOO Y) do necessary in order to call DWIMIFY. Maybe should give
;; DWIMIFY an rgument like stoptail?

;; Done this way instead of changing CDR X because CDR of first PTR is not EQ to the entry in the history list.

```
        [DWIMIFY2 Z Z T (COND
                           (I.S.TYPE 'IFWORD)
                           (T                                     ; so if it sees a function in a variable position, it will insert parens,
                                                                  ; e.g. FOR X IN Y PRINT Z
                              'FORWORD]
    A   (COND
           ((NULL (CDR Z))
```

;; Because DWIMIFY2 was called with FORMSFLG T, this came out as a list of forms, but there was only one form. E.g. X_ (FOO)
;; became ((SETQ X (FOO))).

```
            (/RPLNODE Z (CAAR Z)
                   (CDAR Z))
            (GO C)))
    B   [SELECTQ (CAAR PTRS)
             ((I.S.TYPE FIRST FINALLY EACHTIME)                   ; More than one form permitted in operator --- means implicit
                                                                  ; progn.
                 (SETQ UNDOLST (CONS (CONS OPRTAIL (CONS (CAR OPRTAIL)
                                                         (CDR OPRTAIL)))
                                     UNDOLST))
                 (SETQ BODY (CONS 'PROGN (APPEND Z)))             ; for possible use in substituting into an i.s.opr
                 (CLISPRPLNODE OPRTAIL (CDR BODY)
                        (CDR OPRTAIL))
```

;; Smashes the operatr itself with the body of i.s. so that when we get back to clispfor0, can distinguish the implicit progn case
;; from others. The setting of UNDOLST is to enable restoration.

```
                 [AND (NULL LSTFLG)
                      (CLISPRPLNODE BODYTAIL (CAR Z)
                             (NCONC (CDR Z)
                                    (CDR BODYTAIL]                ; Takes parentheses back out.
                 (GO C))
             (COND
                [(FMEMB (CAR PTRS)
                        I.S.OPRSLST)
```

;; ok for a user defined opeator to have several arguments. (maybe we should phase out the errors and insertion of automatic
;; DO??)

```
                 (SETQ BODY (CONS 'PROGN (APPEND Z)))
                 (SETQ UNDOLST (CONS (CONS OPRTAIL (CONS (CAR OPRTAIL)
                                                         (CDR OPRTAIL)))
```

```
                                    UNDOLST))
                (AND (NULL LSTFLG)
                     (CLISPRPLNODE (CDR OPRTAIL)
                            (CAR Z)
                            (NCONC (CDR Z)
                                   (CDDR OPRTAIL]
                (LSTFLG (CLISPFORERR OPRTAIL))
                (I.S.TYPE (CLISPFORERR I.S.TYPE BODYTAIL))
                ((EVERY (CDR Z)
                        (FUNCTION LISTP))                    ; E.g. For X in Y print Z --.
                                                             ; This really should be taken care of in DWIMIFY2 --- I.e. (Y
                                                             ; prinnt Z)
                 (/RPLNODE BODYTAIL (CAR Z)
                        (/NCONC (CDR Z)
                                NXTOPRTAIL))
                 (SETQQ I.S.TYPE1 do)
                 (SETQ I.S.TYPE (/ATTACH 'DO (CDR BODYTAIL)))
                 (RPLACD PTRS (CONS (LIST 'I.S.TYPE I.S.TYPE NXTOPRTAIL)
                                    (CDR PTRS)))
                 (SETQ Z (CAR Z]
    C   (AND (LISTP Z)
             (CLISPFOR4 Z))
        [COND
           ((FMEMB (CAR PTRS)
                   I.S.OPRSLST)                              ; I.S.OPRLST is the list of those entries on forptrs defined by an
                                                             ; I.S.OPR.
            (RETURN (PROG ((END (CADDAR PTRS))
                           LST)
                          [OR BODY (COND
                                      ((EQ (CAR (GETPROP (CADR (SETQ BODY (CADAR PTRS)))
                                                         'CLISPWORD))
                                           'FORWORD)         ; modifier
                                       (SETQ BODY (CADDR BODY)))
                                      (T (SETQ BODY (CADR BODY]
                    ;; BODY is the operand to the I.S.OPR operator.  END is the tail of the i.s.  beginning with the next operator following it.
                    ;; The in between operators are the result of the expansion, and need to be dwiified, i.e. processed by clispfor1, and then
                    ;; have i.v.  and body substituted into them.
                          (SETQ LST (CDR PTRS))
                     LP1  (COND
                             ((NEQ (CADAR LST)
                                   END)                      ; CADR of each entry on PTRS is the actual tail.
                              (SETQ LST (CLISPFOR1 LST))
                              (GO LP1)))
                          (SETQ LST (CDR PTRS))
                     LP2  (COND
                             ((NEQ (CADAR LST)
                                   END)
                              (PROG ((LST1 (CADAR LST))
                                     (END1 (CADDAR LST)))    ; The tail of the iterative statement begining with the opeator
                                ;; tail of iterative statement beginning with next operator the segment between tem and nxt corresponds to
                                ;; the body of this opeator
                                LP3 (COND
                                       ((EQ (SETQ LST1 (CDR LST1))
                                            END1)
                                        (RETURN)))
                                    [SELECTQ (CAR LST1)
                                         (BODY (FRPLACA LST1 BODY))
                                         (I.V. (FRPLACA LST1 I.V.))
                                         (AND (LISTP (CAR LST1))
                                              (CLISPDSUBST (CAR LST1]
                                    (GO LP3))
                              (SETQ LST (CDR LST))
                              (GO LP2)))
                          (RETURN LST]
    OUT (RETURN (CDR PTRS])
```

(CLISPRPLNODE

```
[LAMBDA (X A D)                                              (* wt%: 16-DEC-75 23 43)
```

;; like /rplnode, except that dwimnewfile? does not count it as a change to the function

```
(COND
   ((LISTP X)
    [AND LISPXHIST (UNDOSAVE (LIST 'CLISPRPLNODE X (CAR X)
                                   (CDR X]
    (FRPLACA X A)
    (FRPLACD X D))
   (T (ERRORX (LIST 4 X])
```

(CLISPFOR2

```
[LAMBDA (LST FLG)                                            (* lmm "13-Aug-84 16:42")
  [MAP (SETQ LST (DREVERSE LST))
       (FUNCTION (LAMBDA (X)
                 (SELECTQ (CAAR X)
```

```
                        (WHEN [RPLACA X (COND
                                           (FLG
                                                ;; When FLG is true, we are computing a condition forDOING it, and when FLG=NIL,
                                                ;; for not doing it, hence difference in sign.
                                                (CADADR (CAR X)))
                                          (T (NEGATE (CADADR (CAR X])
                        (UNLESS [RPLACA X (COND
                                             [FLG (NEGATE (CADADR (CAR X]
                                             (T (CADADR (CAR X])
                        ((WHILE REPEATWHILE)
                            [RPLACA X (NEGATE (CADADR (CAR X])
                        ((UNTIL REPEATUNTIL)
                            (RPLACA X (CADADR (CAR X))))
                        NIL]
    LST])
```

(CLISPFOR3

```
  [LAMBDA (LST)                                                    (* wt%: 25-FEB-76 1 59)
```

;; Used to process FINALLY, EACHTIME, and FIRST lists. LST is a list of form (FINALLY . tail)

```
    (PROG (TEM)
          (RETURN (MAPCONC (DREVERSE LST)
                           (FUNCTION (LAMBDA (X)
                                             (SETQ TEM (CADR X))
                                             (OR (LISTP (CAR TEM))
                                                 (LIST (CADR TEM])
```

(CLISPFORVARS

```
  [LAMBDA (PTRS)                                                   (* lmm "20-Jul-86 12:40")
```

;; Does for FOR and BIND what CLISPFOR1 does for the rest of the ptrs. LST is either a (FOR --) or (BIND --) entry from PTRS. CLISPFOR3
;; handles the following pathological cases. The variables may be spread out, or listed, they may involve assignments, either spread out or listed,
;; and they may be terminated by a form or function in the case that there is no FOROPR. E.g. FOR X Y Z (PRINT X), FOR (X Y Z) PRINT X, FOR
;; X Y _ T Z PRINTT X, FOR (X (Y_T) Z) (PRINT X) etc.

```
    (PROG (TEM OLDFLG LST LST0 L1 VARLST IV (CLISPCONTEXT 'FOR/BIND))
                                                                   ; clispcontext tells CLISPATOM2 not to try spelling correction on
                                                                   ; the variable name.
          (SETQ L1 (CADDR (CAR PTRS)))
          [SETQ LST0 (SETQ LST (CDR (CADAR PTRS]
      LP  (COND
             ((EQ LST0 L1)
              (GO NX)))
          (COND
             ((LITATOM (CAR LST0))
              (SELECTQ (CADR LST0)
                   ((_ ←)
                        (RPLACA LST0 (LIST 'SETQ (CAR LST0)
                                           (CADDR LST0)))
                        (RPLACD LST0 (CDDDR LST0))
                        (GO LP))
                   NIL)
              (AND CLISPFLG (STRPOSL LEFT.ARROWS.BITTABLE (CAR LST0))
                   (SETQ TEM (DWIMIFY2 LST0 LST NIL T T))
                   (SETQ LST0 TEM)))
             [(LISTP (CAR LST0))
              (SELECTQ (CAAR LST0)
                   ((SETQQ SAVESETQQ)                              ; SAVESETQ and SAVESETQQ can occur on typein if the user
                                                                   ; should happen to DW a portion of the l.s.

                        )
                   ((SETQ SAVESETQ)
                        (DWIMIFY2 (CDDAR LST0)
                                  (CAR LST0)
                                  T))
                   (COND
                      ((AND (OR (EQ (CADAR LST0)
                                    '_)
                                (EQ (CADAR LST0)
                                    '←))
                            (NULL (CDDDAR LST0)))
                       [FRPLACA LST0 (CONS 'SETQ (CONS (CAAR LST0)
                                                       (CDDAR LST0]
                       (GO LP))
                      [(AND CLISPFLG (PROG ((X (CAR LST0)))
                                           LX  (COND
                                                  ((NLISTP X)
                                                   (RETURN NIL))
                                                  ((AND (LITATOM (CAR X))
                                                        (STRPOSL LEFT.ARROWS.BITTABLE (CAR X)))
                                                   (RETURN T)))
                                               (SETQ X (CDR X))
                                               (GO LX)))
                       (CLISPFORVARS1 (CAR LST0)
                              (EQ L1 (CDR LST)))
```

;; The second argument to CLISPFORVARS1 corresonds to FORMSFLG in the call to DWIMIFY2, e.g. FOR X (Y_T) want

```
                    ;; FORMSFLG to be NIL.  but FOR (X_T Y) want it to be T.
                    (COND
                       ((AND (LISTP (CAAR LST0))
                             (NULL (CDAR LST0)))
                        ;; form was (A_form) and now is ((SETQ A form)) so remove extra parentheses inserted because formsflg was
                        ;; (incorrectly) T.  Note that when we called clispforvars1, we donot know whether (CAR LST0) is of the form (A_B
                        ;; C_D) or (A _ B), i.e. one or two assignments.
                        (FRPLACA LST0 (CAAR LST0]
                    ((AND (EQ LST0 LST)
                          (EQ L1 (CDR LST0)))                    ; Says this is the first argument.
                     (CLISPFORVARS1 (CAR LST0)
                            T))
                    (I.S.TYPE (CLISPFORERR LST0 I.S.TYPE))
                    (T ;; Necessary because LST0 may not really correspnd to ssructure in the original statement, because of ldiff.
                       (GO ADDDO]
              (T (CLISPFORERR LST0)))
        (SETQ LST0 (CDR LST0))
        (GO LP)
 NX
;; The area between LST and LST0 now corresponds to the (dwimified) variables.  They may appears as a segment or as a list.
        (SETQ LST0 (COND
                      ([AND (EQ LST0 (CDR LST))
                            (LISTP (CAR LST))
                            (NOT (FMEMB (CAAR LST)
                                        '(SETQ SETQQ OLD old SAVESETQ SAVESETQQ]
                       (SETQ L1 NIL)
                       (CAR LST))
                      (T LST)))
;; LST0 now corresponds to the beginning of the list of variables, L1 to its end.  VARLST will be used to assemble the vlue.
 LP1 [COND
        ((EQ LST0 L1)
         [COND
            ((AND IV (NEQ (CAAR PTRS)
                          'BIND)
                  (NULL I.V.))
             (SETQ FIRSTI.V. (SETQ I.V. IV]                      ; IV is the first variable encountered in the variable list (may be
                                                                 ; OLD vriable)
         (RETURN (DREVERSE VARLST)))
        ((FMEMB (CAR LST0)
                '(OLD old))
         (SETQ OLDFLG T)
         (SETQ MAKEPROGFLG T)
         (SETQ LST0 (CDR LST0))
         (SETQ TEM (CAR LST0)))
        ((FMEMB (CAR (LISTP (CAR LST0)))
                '(OLD old))
         (SETQ OLDFLG T)
         (SETQ MAKEPROGFLG T)
         (SETQ TEM (CADAR LST0)))
        (T (SETQ OLDFLG NIL)
           (SETQ TEM (CAR LST0]
     [COND
        [(AND TEM (LITATOM TEM))
         (SETQ VARS (CONS TEM VARS))
         (COND
            ((NULL IV)
             (SETQ IV TEM)
             [COND
                ((EQ (CAAR PTRS)
                     'AS)
                 (FRPLACD (CDDAR PTRS)
                          (LIST IV]
                ;; Marks the i.v. for this AS.  used by clispfor11 when you specify an operatand which is just a functon name.

             ))
         (COND
            ((NULL OLDFLG)
             (SETQ VARLST (CONS TEM VARLST]
        ((SELECTQ (CAR TEM)
             ((SETQ SAVESETQ)
              T)
             ((SETQQ SAVESETQQ)
                  (FRPLACA TEM 'SETQ)
                  (FRPLACA (CDDR TEM)
                           (LIST 'QUOTE (CADDR TEM)))
              T)
             NIL)
         (SETQ MAKEPROGFLG T)                                    ; Says the expression must translate into an open prog.
         (SETQ VARS (CONS (CADR TEM)
                          VARS))
         [COND
            ((NULL IV)
```

```
                (SETQ IV (CADR TEM))
                (SELECTQ (CAAR PTRS)
                     (BIND)
                     (FOR (SETQ IVINITFLG T))
                     (AS (FRPLACD (CDDAR PTRS)
                                  (LIST IV T)))
                     (SHOULDNT 'CLISPFORVARS]
             [COND
                (OLDFLG (SETQ INITVARS (CONS TEM INITVARS)))
                (T (SETQ VARLST (CONS (CDR TEM)
                                      VARLST]
             (SETQ UNDOLST (CONS (CONS LST0 (CONS (LIST (CADR TEM)
                                                        LEFT.ARROW
                                                        (CADDR TEM))
                                                  (CDR LST0)))
                                 UNDOLST)))
          (T (CLISPFORERR (LIST TEM]
        (SETQ LST0 (CDR LST0))
        (GO LP1)
     ADDDO
        (/RPLNODE LST0 'DO (CONS (CAR LST0)
                                 (CDR LST0)))
        (SETQ L1 LST0)
        (FRPLACD PTRS (CONS (LIST 'I.S.TYPE (SETQ I.S.TYPE LST0)
                                  (CADDR (CAR PTRS)))
                            (CDR PTRS)))
        (GO NX])
```

(CLISPFORVARS1

```
[LAMBDA (L FLG)                                          (* lmm "21-Jun-85 16:59")
  (PROG ($TAIL)
        (SETQ $TAIL L)
    LP  [COND
           ((NULL $TAIL)
            (RETURN))
           ((STRPOSL LEFT.ARROW.BITTABLE (CAR $TAIL))
            (COND
               ((LITATOM (CAR $TAIL))
                (SETQ $TAIL (TAILP (DWIMIFY2 $TAIL L NIL FLG T)
                                   L)))
               (T (CLISPFORVARS1 (CAR $TAIL]
        (SETQ $TAIL (CDR $TAIL))
        (GO LP])
```

(CLISPFOR4

```
[LAMBDA (X)                                              (* wt%: 17-DEC-76 19 8)
  (SELECTQ (CAR X)
       ((GO RETURN ERROR! RETFROM RETEVAL)
           (SETQ TERMINATEFLG T)
           (SETQ MAKEPROGFLG T))
       (PROG NIL)
       (SOME X (FUNCTION (LAMBDA (X)
                         (COND
                            ((EQ X '$$VAL)
                             (SETQ MAKEPROGFLG T)        ; keep on looking for RETURN or GO
                             NIL)
                            ((LISTP X)
                             (CLISPFOR4 X])
```

(CLISPFORF/L

```
[LAMBDA (EXP VAR DECLARELST)                             (* lmm "29-Jul-86 00:24")
```

;; Build the FUNCTIONal expression to be executed as the MAPFN for the FOR loop

```
  (LIST 'FUNCTION (COND
                    (NIL ;; This originally tried to elimate the dummy variable when the FOR was a unary function, but in this case, there
                         ;; was still a problem --- thus this is commented out
                         (CAAR EXP))
                    (T                                   ; Otherwise, build a LAMBDA expression that contains all the
                                                         ; expressions to be evaluated.
                       `(LAMBDA ,VAR
                          ,@[AND DECLARELST `((DECLARE ,@(MAPCONC (DREVERSE DECLARELST)
                                                                  (FUNCTION (LAMBDA (X)
                                                                      (LDIFF (CDADR X)
                                                                             (CADDR X]
                          ,@EXP])
```

(CLISPDSUBST

```
[LAMBDA (X)                                              (* wt%: "21-JAN-80 20:11")
  (PROG (TEM)
```

;; goes through X and does a dsubst of I.V. for (QUOTE I.V.) and BODY for (QUOTE BODY) in X AND all of the translations in the hasharray

```
        [MAP X (FUNCTION (LAMBDA (X)
```

```
                              (SELECTQ (CAR X)
                                   (BODY (FRPLACA X BODY))
                                   (I.V. (FRPLACA X I.V.))
                                   (AND (LISTP (CAR X))
                                        (CLISPDSUBST (CAR X]
          (COND
             ((SETQ TEM (GETHASH X CLISPARRAY))
              (COND
                 ((EQ (CAR (GETP (CAR X)
                                 'CLISPWORD))
                      'CHANGETRAN)
                  ;; these constructs have the property that translation differs depending on expression, e.g. while (fetch foo of x) is always the
                  ;; same regardless of what x is, (change x y) differs depending on what x is.
                  (PUTHASH X NIL CLISPARRAY)
                  (DWIMIFY1 X))
                 (T (CLISPDSUBST TEM])
```

(GETDUMMYVAR

```
  [LAMBDA (BINDITFLG)                                                    (* lmm "28-MAY-83 18:01")
    (PROG (VAR)
          [SETQ VAR (CAR (SETQ DUMMYVARS (OR (CDR DUMMYVARS)
                                             (CDR (RPLACD DUMMYVARS (LIST (GENSYM]
          [COND
             (BINDITFLG (SETQ VARS (CONS VAR VARS))
                        (SETQ PROGVARS (CONS VAR PROGVARS]
          (RETURN VAR])
```

(CLISPFORINITVAR

```
  [LAMBDA (VAR EXP)                                                      (* wt%: "21-JAN-80 20:44")
```

;; this function is called when is necessary to initialize a variable to an expression outside of tje scope of anyvariables bound by i.s., i.e. in the prog
;; binding. it generates a dummy variabe, binds it to exp, and then initializes var to that expresssin

```
    (SETQ PROGVARS (CONS (LIST (GETDUMMYVAR)
                               EXP)
                         PROGVARS))
    (SETQ INITVARS (NCONC1 INITVARS (LIST 'SETQ VAR (CAAR PROGVARS])

)

(DEFINEQ
```

(\DURATIONTRAN

```
  [LAMBDA (FORM)                                                         (* JonL "23-Jul-84 15:39")
    (PROG ((BODY FORM)
           (OLDTIMER)
           (EXPANSION)
           (SETUPFORM '(SETUPTIMER FORDURATION OLDTIMER . TIMERUNITSLST))
           (EXPIREDFORM '(TIMEREXPIRED? \DurationLimit . TIMERUNITSLST))
           USINGTIMER USINGBOX FORDURATION RESOURCENAME UNTILDATE TIMERUNITS TIMERUNITSLST TEMP)
          (DECLARE (SPECVARS TIMERUNITS USINGTIMER USINGBOX FORDURATION RESOURCENAME UNTILDATE)
                   (GLOBALVARS DURATIONCLISPWORDS LCASEFLG))
```

;; DURATIONCLISPWORDS is a list of lists, each one of which has the canonical word for some CLISPWORD as second element. First element
;; is the all-caps version, so that SPECVARS communication can take place.

```
          (PROG ((L DURATIONCLISPWORDS)
                 (Z BODY))
            LP   (AND (NLISTP L)
                      (RETURN (SETQ BODY Z)))
                 (SETQ Z (\CLISPKEYWORDPROCESS Z (CAR L)))
                 (SETQ L (CDR L))
                 (GO LP))
          [COND
             ((NOT (LITATOM RESOURCENAME))
              (SETERRORN 14 FORM)
              (ERRORX))
             ((EQ RESOURCENAME T)
              (SETQ RESOURCENAME '\ForDurationOfBox]
          (COND
             (USINGBOX (AND RESOURCENAME (ERROR "Both 'usingTimer' and 'resourceName' specified" FORM))
                       (SETQ USINGTIMER USINGBOX)))
          [COND
             ((NULL TIMERUNITS)                                          ; Standard case
              NIL)
             (UNTILDATE (ERROR "Can't specify timerUnits for 'untilDate'" FORM))
             [(SETQ TEMP (CONSTANTEXPRESSIONP TIMERUNITS))
              (COND
                 ((AND (SETQ TEMP (\CanonicalizeTimerUnits (CAR TEMP)))
                       (NEQ TEMP 'MILLISECONDS))
                  (SETQ TIMERUNITSLST (LIST (LIST 'QUOTE TEMP]
             (T (SETQ TIMERUNITSLST (LIST TIMERUNITS]
          (COND
             ((AND (NULL FORDURATION)
                   (NULL UNTILDATE))
              (ERROR "No duration interval" FORM))
```

```
            ((AND FORDURATION UNTILDATE)
             (ERROR "Both 'untilDate' and 'forDuration' specified" FORM)))
         [COND
            (UNTILDATE (SETQ FORDURATION UNTILDATE)            ; Make the 'interval' be the thing supplied for the 'date'
                   (SETQ SETUPFORM '(SETUPTIMER.DATE FORDURATION OLDTIMER))
                   (SETQ TIMERUNITSLST '('SECONDS]
         (COND
            ([AND (PROG1 RESOURCENAME                          ; Comment PPLossage
                         )
                  (NOT (\TIMER.TIMERP (EVAL (LISTGET (GETDEF RESOURCENAME 'RESOURCES NIL 'NOERROR)
                                                     'NEW]
             (ERROR RESOURCENAME "is not a timer RESOURCE")))
         (SETQ EXPANSION (LIST [LIST 'LAMBDA '(\DurationLimit)
                                     '(DECLARE (LOCALVARS \DurationLimit))
                                     (CONS 'until (CONS EXPIREDFORM 'BODY]
                               SETUPFORM))
         [AND (LISTP (CAR TIMERUNITSLST))
              (NEQ (CAAR TIMERUNITSLST)
                   'QUOTE)
              (SETQ EXPANSION (LIST (LIST 'LAMBDA '(\TimerUnit)
                                          '(DECLARE (LOCALVARS \TimerUnit))
                                          EXPANSION)
                                    (CAR TIMERUNITSLST)))
              (SETQ TIMERUNITSLST '(\TimerUnit]
         (SETQ OLDTIMER (OR RESOURCENAME USINGTIMER))
         (SETQ EXPANSION (SUBPAIR '(BODY FORDURATION OLDTIMER TIMERUNITSLST)
                                  (LIST BODY FORDURATION OLDTIMER TIMERUNITSLST)
                                  EXPANSION))
         [COND
            (RESOURCENAME (SETQ EXPANSION (LIST 'WITH-RESOURCES RESOURCENAME EXPANSION]
         [COND
            (LCASEFLG (MAP FORM (FUNCTION (LAMBDA (X)
                                          (PROG [(Y (GETPROP (CAR X)
                                                             'CLISPWORD]
                                                (COND
                                                   ((AND (LISTP Y)
                                                         (SETQ Y (CDR Y))
                                                         [LITATOM (COND
                                                                     ((NLISTP Y)
                                                                      Y)
                                                                     (T (SETQ Y (CAR Y]
                                                         (NEQ Y (CAR X)))
                                                    (/RPLACA X Y]
         (RETURN EXPANSION])
```

(**CLISPKEYWORDPROCESS**

```
  [LAMBDA (FORM WORDLST)                                   (* JonL "27-APR-83 04:39")
```

;; Looks for the first 'keyword' in the list FORM which is mentioned in the WORDLST -- and if one is found, the the first keyword in WORDLST is ;; presumed to be the name of a variable to be set to the keyword's value. Returns the original list with the keyword pair non-destructively spliced ;; out.

```
    (COND
       ((NULL FORM)
        NIL)
       ((FMEMB (CAR FORM)
               WORDLST)
        (SET (CAR WORDLST)
             (CADR FORM))
        (CDDR FORM))
       ((NLISTP FORM)
        FORM)
       (T (PROG ((X WORDLST)
                 TMP)
            LP   (COND
                    ([AND (LISTP X)
                          (NOT (SETQ TMP (FMEMB (CAR X)
                                                FORM]
                     (SETQ X (CDR X))
                     (GO LP)))
                 (RETURN (COND
                            (TMP (SET (CAR WORDLST)
                                      (CADR TMP))
                                 (NCONC (LDIFF FORM TMP)
                                        (CDDR TMP)))
                            (T FORM])
)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS DWIMUNDOCATCH MACRO ((TAG UNDOFORM)
                               ;; Hairy control structure used by DWIMIFY.  Effectively (CATCH TAG (UNDONLSETQ UNDOFORM)),
                               ;; except that it ensures that the undoing occurs not only when the UNDONLSETQ returns NIL (from
                               ;; ERROR!), but also when a non-list is thrown to TAG.  THROW is used in various places to tell the caller
```

```
                                        ;; to do something different (usually try again after a successful spelling correction).  The body of this
                                        ;; macro is a copy of the UNDONLSETQ macro appropriately modified.

                                        (PROG ((LISPXHIST LISPXHIST)
                                               UNDOSIDE0 UNDOSIDE UNDOTEM)
                                              (DECLARE (SPECVARS LISPXHIST))
                                              [COND
                                                 ([LISTP (SETQ UNDOSIDE (LISTGET1 LISPXHIST 'SIDE]
                                                  (SETQ UNDOSIDE0 (CDR UNDOSIDE)))
                                                 (T (SETQ UNDOSIDE0 UNDOSIDE)
                                                    (SETQ UNDOSIDE (LIST 0))
                                                    (COND
                                                       (LISPXHIST (LISTPUT1 LISPXHIST 'SIDE UNDOSIDE))
                                                       (T (SETQ LISPXHIST (LIST 'SIDE UNDOSIDE]
                                              [SETQ UNDOTEM (RESETVARS (%#UNDOSAVES)
                                                                       (RETURN (CL:CATCH TAG (XNLSETQ UNDOFORM]
                                              (COND
                                                 ((EQ UNDOSIDE0 'NOSAVE)
                                                  (LISTPUT1 LISPXHIST 'SIDE 'NOSAVE))
                                                 (T (UNDOSAVE)))
                                              [COND
                                                 ((NLISTP UNDOTEM)    ; undo side effects on "error" return
                                                  (UNDONLSETQ1 (CDR UNDOSIDE)
                                                        (LISTP UNDOSIDE0]
                                              (RETURN UNDOTEM))))
)
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK%: FORBLOCK (ENTRIES CLISPFOR)
       CLISPFORVARS CLISPFOR0 CLISPFOR2 CLISPFORINITVAR CLISPDSUBST \CLISPKEYWORDPROCESS CLISPFORF/L CLISPFOR4
       CLISPFORVARS1 CLISPFOR3 CLISPFOR1 CLISPFOR0A CLISPFOR \DURATIONTRAN
       (SPECVARS UNDOSIDE LISPXHIST BODY I.S.TYPE1 I.S.TYPE TERMINATEFLG FIRSTI.V. I.V. PROGVARS MAKEPROGFLG
              IVINITFLG INITVARS UNDOLST DWIMIFYING VARS DWIMIFYCHANGE DUMMYVARS I.S.OPRSLST CLISPCONTEXT
              UNDOSIDE0 EXP))

(BLOCK%: DWIMIFYBLOCK CLBINARYMINUS? CLISPANGLEBRACKETS CLISPATOM CLISPATOM0 CLISPATOM1 CLISPATOM1A CLISPATOM1B
       CLISPATOM2 CLISPATOM2A CLISPATOM2C CLISPATOM2D CLISPATOMARE1 CLISPATOMARE2 CLISPATOMIS1 CLISPATOMIS2
       CLISPBROADSCOPE CLISPBROADSCOPE1 CLISPCAR/CDR CLISPCAR/CDR1 CLISPCAR/CDR2 CLISPIF CLISPIF0 CLISPIF1
       CLISPIF2 CLISPIF3 CLISPLOOKUP CLISPRESPELL CLRPLNODE CLUNARYMINUS? DWIMIFY DWIMIFY0 DWIMIFY0? DWIMIFY1
       DWIMIFY1? DWIMIFY1A DWIMIFY2 DWIMIFY2? DWIMIFY2A DWIMIFYFNS DWMFY0 DWMFY1 DWMFY2 FIX89 FIX89A FIX89TYPEIN
       FIXAPPLY FIXATOM FIXATOM1 FIXCONTINUE FIXCONTINUE1 FIXLAMBDA GETDUMMYVAR GETVARS GETVARS1 RETDWIM
       RETDWIM1 SHRIEKER STOPSCAN? WTFIX WTFIX0 WTFIX1
       (ENTRIES WTFIX WTFIX1 DWIMIFYFNS DWIMIFY DWIMIFY0 DWIMIFY0? DWIMIFY1A GETDUMMYVAR DWIMIFY2 DWIMIFY2?
              DWIMIFY1? DWIMIFY1 DWIMIFY2A CLISPLOOKUP)
       (SPECVARS 89CHANGE 89FLG BRACKET BRACKETCNT ATTEMPTFLG BACKUPFLG BODY BREAKFLG BROADSCOPE CLISPCHANGE
              CLISPCHANGES CLISPCONTEXT CLISPERTYPE CLTYP CURRTAIL DWIMIFYCHANGE DWIMIFY0CHANGE DWIMIFYFLG
              DWIMIFYING ENDTAIL EXP EXPR FAULTAPPLYFLG FAULTARGS FAULTFN FAULTPOS FAULTX FAULTXX FIRSTI.V.
              FIXCLK FORMSFLG I.S.TYPE I.S.TYPE1 HISTENTRY I.S. I.V. INITVARS IVINITFLG LISPFN CHARLST
              MAKEPROGFLG NCONC1LKUP NCONCLKUP NEGFLG NEWTAIL NEXTAIL SUBPARENT NOFIX89 NOSAVEFLG ONEFLG
              ONLYSPELLFLG PARENT SIDES TAIL TENTATIVE TERMINATEFLG TYP TYPE-IN? UNDOLST UNDOSIDE UNDOSIDE0 VAR1
              VAR2 VARS WORKFLAG UNARYFLG DEST FOR I.S.OPRSLST PROGVARS))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS DWIMINMACROSFLG CHECKCARATOMFLG TREATASCLISPFLG CLISPHELPFLG CLISPIFTRANFLG CLISPRETRANFLG
       DWIMCHECKPROGLABELSFLG DWIMCHECK#ARGSFLG SHALLOWFLG PRETTYTRANFLG CLEARSTKLST LCASEFLG LAMBDASPLST
       DURATIONCLISPWORDS CLISPTRANFLG CLISPIFWORDSPLST LPARKEY DWIMUSERFORMS DWIMKEYLST SPELLINGS3 SPELLINGS1
       CLISPARRAY CLISPFLG CLISPCHARS CLISPISNOISEWORDS CLISPLASTSUB CLISPISWORDSPLST CLISPCHARRAY
       CLISPINFIXSPLST OKREEVALST WTFIXCHCONLST1 WTFIXCHCONLST RPARKEY NOFIXFNSLST0 NOFIXVARSLST0 LISPXHISTORY
       DWIMEQUIVLST COMMENTFLG USERWORDS SPELLINGS2 FILELST CLISPFORWORDSPLST CLISPDUMMYFORVARS LASTWORD
       COMPILERMACROPROPS)
)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(ADDTOVAR NLAML BREAK1)
)

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

(ADDTOVAR NLAMA DWIMIFYFNS)

(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)

(RPAQ? DWIM.GIVE.UP.TIME )

(RPAQ? DWIM.GIVE.UP.INTERVAL 2000)

(PUTPROPS DWIMIFY COPYRIGHT ("Venue & Xerox Corporation" T 1978 1984 1985 1986 1990))
```

FUNCTION INDEX

CHECKTRAN 48
CLBINARYMINUS? 18
CLISP-SIMPLE-FUNCTION-P 46
CLISPANGLEBRACKETS 9
CLISPATOM 42
CLISPATOM0 11
CLISPATOM1 12
CLISPATOM1A 19
CLISPATOM1B 20
CLISPATOM2 20
CLISPATOM2A 26
CLISPATOM2C 27
CLISPATOM2D 28
CLISPATOMARE1 31
CLISPATOMARE2 31
CLISPATOMIS1 31
CLISPATOMIS2 31
CLISPBROADSCOPE 26
CLISPBROADSCOPE1 27
CLISPCAR/CDR 28
CLISPCAR/CDR1 30
CLISPCAR/CDR2 30
CLISPDSUBST 63
CLISPELL 46
CLISPFOR 50
CLISPFOR0 51
CLISPFOR0A 58
CLISPFOR1 58
CLISPFOR2 60
CLISPFOR3 61
CLISPFOR4 63
CLISPFORF/L 63
CLISPFORINITVAR 64
CLISPFORVARS 61
CLISPFORVARS1 63
CLISPFUNCTION? 44
CLISPIF 48
CLISPIF0 48
CLISPIF1 50
CLISPIF2 50
CLISPIF3 50
CLISPLOOKUP 25
CLISPNOEVAL 25
CLISPNOTVARP 45
CLISPRESPELL 11
CLISPRPLNODE 60
CLRPLNODE 17
CLUNARYMINUS? 18
DWIMARKASCHANGED 37
DWIMERRORRETURN 37
DWIMIFY 2
DWIMIFY0 2
DWIMIFY0? 3
DWIMIFY1 4
DWIMIFY1? 4
DWIMIFY1A 6
DWIMIFY2 6
DWIMIFY2? 6
DWIMIFY2A 9
DWIMIFYFNS 1
DWIMUNSAVEDEF 47
DWMFY0 3
DWMFY1 4
DWMFY2 6
EXPRCHECK 11
FINDFN 46
FIX89 44
FIX89A 44
FIX89TYPEIN 38
FIXAPPLY 39
FIXATOM 39
FIXATOM1 41
FIXCONTINUE 42
FIXCONTINUE1 42
FIXLAMBDA 38
FIXPRINTIN 44
GETDUMMYVAR 64
GETVARS 43
GETVARS1 43
RETDWIM 36
RETDWIM1 37
SHRIEKER 9
STOPSCAN? 17
WTFIX 32
WTFIX0 32
WTFIX1 32
\CLISPKEYWORDPROCESS ...65
\DURATIONTRAN 64

VARIABLE INDEX

DWIM.GIVE.UP.INTERVAL ..66
DWIM.GIVE.UP.TIME 66

MACRO INDEX

DWIMUNDOCATCH 65