

File created: 7-Nov-2022 09:54:34 {DSK}<home>larry>ilisp>medley>sources>CMLUNDO.;2

changes to: (IL:FUNCTIONS UNDOABLY)

previous date: 18-Oct-2022 16:24:32 {DSK}<home>larry>ilisp>medley>sources>CMLUNDO.;1

Read Table: XCL

Package: XEROX-COMMON-LISP

Format: XCCS

; Copyright (c) 1986-1988, 1990, 2022 by Venue & Xerox Corporation.

```
(IL:RPAQQ IL:CMLUNDOCOMS
  ((IL:VARIABLES *IN-DEFINER*)
   (IL:FUNCTIONS NOHOOK UNDOABLY UNDOABLY-FMAKUNBOUND UNDOABLY-MAKUNBOUND UNDOABLY-SETF UNDOHOOK
    UNDOABLY-PSETF UNDOABLY-POP UNDOABLY-PUSH UNDOABLY-PUSHNEW UNDOABLY-REMF UNDOABLY-ROTATEF
    UNDOABLY-SHIFTF DEFINE-UNDOABLE-MODIFY-MACRO UNDOABLY-DECF UNDOABLY-INCF UNDOABLY-PROCLAIM)
   (IL:FUNCTIONS MAKE-UNDOABLE STOP-UNDOABLY)
   (IL:FUNCTIONS UNDOABLY-SETF-SYMBOL-FUNCTION UNDOABLY-SETF-MACRO-FUNCTION)
   (IL:DECLARE\ : IL:DONTEVAL@LOAD IL:DONTEVAL@COMPILE IL:DOCOPY (IL:P (IL:MOVD
    UNDOABLY-SETF-SYMBOL-FUNCTION
    ,
    IL:UNDOABLY-SETF-SYMBOL-FUNCTION
    )
   (IL:MOVD
    ,
    UNDOABLY-SETF-MACRO-FUNCTION
    ,
    UNDOABLY-SETF-MACRO-FUNCTION
    )))
  (IL:ADDVARS (IL:LISPFNS (PROCLAIM . UNDOABLY-PROCLAIM)
    (POP . UNDOABLY-POP)
    (PSETF . UNDOABLY-PSETF)
    (PUSH . UNDOABLY-PUSH)
    (PUSHNEW . UNDOABLY-PUSHNEW)
    (REMF . UNDOABLY-REMF)
    (ROTATEF . UNDOABLY-ROTATEF)
    (SHIFTF . UNDOABLY-SHIFTF)
    (DECF . UNDOABLY-DECF)
    (INCF . UNDOABLY-INCF)
    (SET . UNDOABLY-SET-SYMBOL)
    (MAKUNBOUND . UNDOABLY-MAKUNBOUND)
    (FMAKUNBOUND . UNDOABLY-FMAKUNBOUND)))
  (IL:FUNCTIONS GET-UNDOABLE-SETF-METHOD UNDOABLY-SET-SYMBOL)
  (IL:FNS UNDOABLY-SETQ)
  (IL:SPECIAL-FORMS UNDOABLY UNDOABLY-SETQ)
  (IL:DECLARE\ : IL:DONTEVAL@LOAD IL:DONTEVAL@COMPILE IL:DOCOPY (IL:P (IL:MOVD 'UNDOABLY-SET-SYMBOL
    'IL:UNDOABLY-SET-SYMBOL)))
  (IL:PROP (IL:FILETYPE IL:MAKEFILE-ENVIRONMENT)
    IL:CMLUNDO)
  (IL:PROP :UNDOABLE-SETF-INVERSE SYMBOL-FUNCTION MACRO-FUNCTION)
  (IL:DECLARE\ : IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILETVARS (IL:ADDVARS (IL:NLAMA
    UNDOABLY-SETQ
    )
   (IL:NLAML)
   (IL:LAMA))))))
```

```
(DEFVAR *IN-DEFINER* NIL)
```

```
(DEFUN NOHOOK (FN ARGS &OPTIONAL ENV &AUX (*EVALHOOK* NIL))
  (APPLY FN ARGS))
```

```
(DEFMACRO UNDOABLY (&REST FORMS &ENVIRONMENT ENV) ; Edited 7-Nov-2022 09:52 by lmm
```

```
(WALK-FORM
  (IL:MKPROGN FORMS)
  :ENVIRONMENT ENV :WALK-FUNCTION
  #' (LAMBDA
    (X CONTEXT)
    (COND
      ((NOT (CONSP X))
       X)
      ((NOT (SYMBOLP (CAR X)))
       X)
      (T
       (CASE (CAR X)
         ((SETQ IL:SETQ SETF)
          (VALUES
            (IL:MKPROGN
             (WITH-COLLECTION
```

```

(DO ((TAIL (CDR X)
         (CDDR TAIL)))
     (NULL TAIL))
(COLLECT (IF (SYMBOLP (CAR TAIL))
             (IF (VARIABLE-LEXICAL-P (CAR TAIL))
                 `,(CAR X)
                 , (CAR TAIL)
                 ,(WALK-FORM-INTERNAL (CADR TAIL)))
             (PROGN (COND
                    ((NOT (OR (VARIABLE-SPECIAL-P (CAR TAIL))
                              (BOUNDP (CAR TAIL))))
                     ;; should possibly spelling correct?
                     (WHEN NIL
                      ;; this warning just seems useless; it doesn't proclaim anything or mark it as
                      ;; changed in FILEPKG or ...
                      (WARN "Variable ~S proclaimed SPECIAL UNDOABLY.. SETQ"
                           (CAR TAIL))))
                    ` (UNDOABLY-SET-SYMBOL ' , (CAR TAIL)
      , (WALK-FORM-INTERNAL (CADR TAIL))))))
(MULTIPLE-VALUE-BIND (FORMALS ACTUALS NEW-VALUE SETTER GETTER)
  (GET-UNDOABLE-SETF-METHOD (CAR TAIL))
  ` (,'LET* (,@ (MAPCAR #' (LAMBDA (X Y)
                        (LIST X (WALK-FORM-INTERNAL Y)))
                    FORMALS ACTUALS)
    (, (WALK-FORM-INTERNAL (CAR NEW-VALUE))
      , (CADR TAIL)))
    , SETTER))))))
T))
(STOP-UNDOABLY (VALUES (IL:MKPROGN (CDR X))
                       T))
(T (LET ((UNDONAME (CDR (ASSOC (CAR X)
                              IL:LISPFNS :TEST #'EQ))))
  (IF UNDONAME
      (CONS UNDONAME (CDR X))
      (IF (AND (OR (GET (CAR X)
                       ' :DEFINER-FOR)
                  (GET (CAR X)
                       ' IL:DEFINER-FOR))
              (NOT *IN-DEFINER*))
          (LET ((*IN-DEFINER* T))
              (VALUES (WALK-FORM-INTERNAL (MACROEXPAND-1 X))
                      T))
          X)))))))))

```

```

(DEFUN UNDOABLY-FMAKUNBOUND (SYMBOL)
  (IL:/PUTD SYMBOL NIL)
  (IL:/REMPROP SYMBOL 'IL:MACRO-FN)
  (IL:/REMPROP SYMBOL 'IL:SPECIAL-FORM)
  (IL:/REMPROP SYMBOL 'IL:CODE)
  (IL:/REMPROP SYMBOL 'IL:EXPR)
  SYMBOL)

```

```

(DEFUN UNDOABLY-MAKUNBOUND (SYMBOL)
  ;; Make a symbol unbound.
  (IL:SAVESET SYMBOL 'IL:NOBIND)
  (IL:/PUTHASH SYMBOL NIL IL:COMPVARMACROHASH)
  (IL:/REMPROP SYMBOL 'IL:GLOBALLY-SPECIAL)
  (IL:/REMPROP SYMBOL 'IL:GLOBALVAR)
  SYMBOL)

```

```

; unbound symbols are set to IL:NOBIND
; remove any constant entry
; left by PROCLAIM special
;

```

```

(DEFMACRO UNDOABLY-SETF (PLACE NEW-VALUE &ENVIRONMENT ENV)
  "UNDOable version of SETF"

```

```

;; note that this is a "one-shot", in that (UNDOABLY (SETF (CDR (RPLACA X Y)) Z) will make the RPLACA undoable, but (UNDOABLY-SETF (CDR
;; (RPLACA X Y)) Z) will not

```

```

(COND
  ((SYMBOLP PLACE)
   ;; assumes variable is not lexical !
   ` (UNDOABLY-SET-SYMBOL ' ,PLACE ,NEW-VALUE))
  (T (MULTIPLE-VALUE-BIND (DUMMIES VALS NEWVAL SETTER GETTER)
    (GET-UNDOABLE-SETF-METHOD PLACE ENV)
    ` (,'LET* (,@ (MAPCAR #'LIST DUMMIES VALS)
                (, (CAR NEWVAL)
                  ,NEW-VALUE))
      , SETTER))))))

```

```

(DEFUN UNDOHOOK (FORM ENV &AUX (*APPLYHOOK* NIL))
  (IF (ATOM FORM)
      (EVAL FORM ENV)

```

```

; Edited 14-Oct-2022 13:54 by Imm

```

```
(CASE (CAR FORM)
  ((SETQ IL:SETQ SETF)
    (DO ((TAIL (CDR FORM))
        VALUE)
      ((NULL TAIL)
        VALUE)
      (SETQ VALUE
        (IF (SYMBOLP (CAR TAIL))
          (UNDOABLY-SET-SYMBOL (POP TAIL)
            (UNDOHOOK (POP TAIL)
              ENV)
            ENV)
          (EVAL ;; real cop-out , just to EVAL of making it undoable
            (MULTIPLE-VALUE-BIND (FORMALS VALS NEW-VALUE SETTER GETTER)
              (GET-UNDOABLE-SETF-METHOD (POP TAIL)
                ENV)
              `(', 'LET* (,@(MAPCAR #'(LAMBDA (X Y)
                (LIST X (LIST 'UNDOABLY Y)))
                (LIST X (LIST 'UNDOABLY Y)))
                FORMALS VALS)
                (, (CAR NEW-VALUE)
                  (UNDOABLY , (POP TAIL))))
                , SETTER)))
            ENV))))))
  (STOP-UNDOABLY
    ;; special signal to not undo
    (IL:\EVAL-PROGN (CDR FORM)
      ENV)
    (T (LET ((UNDONAME (CDR (ASSOC (CAR FORM)
      IL:LISPPXFN :TEST #'EQ))))
      (IF UNDONAME
        (EVALHOOK (CONS UNDONAME (CDR FORM))
          'UNDOHOOK
          'NOHOOK ENV)
        (EVALHOOK FORM 'UNDOHOOK 'NOHOOK ENV))))))
```

```
(DEFMACRO UNDOABLY-PSETF (&REST ARGS &ENVIRONMENT ENV)
```

;; parallel version of UNDOABLY-SETF - simple minded version

```
(COND
  ((NULL ARGS)
    NIL)
  (T `(PROG1 NIL
    (UNDOABLY-SETF , (POP ARGS)
      (PROG1 , (POP ARGS)
        (UNDOABLY-PSETF ,@ARGS))))))
```

```
(DEFMACRO UNDOABLY-POP (PLACE &ENVIRONMENT ENV)
```

```
(IF (SYMBOLP PLACE)
  `(PROG1 (CAR ,PLACE)
    (UNDOABLY-SETQ ,PLACE (CDR ,PLACE)))
  (MULTIPLE-VALUE-BIND (DUMMIES VALS NEWVAL SETTER GETTER)
    (GET-UNDOABLE-SETF-METHOD PLACE ENV)
    `(', 'LET* (,@(MAPCAR #'LIST DUMMIES VALS)
      (LIST (CAR NEWVAL)
        GETTER))
      (PROG1 (CAR , (CAR NEWVAL))
        (SETQ , (CAR NEWVAL)
          (CDR , (CAR NEWVAL)))
        , SETTER))))
```

```
(DEFMACRO UNDOABLY-PUSH (OBJ PLACE &ENVIRONMENT ENV)
```

;; Takes an object and a location holding a list. Conses the object onto PLACE returning then modified list.

```
(IF (SYMBOLP PLACE)
  `(UNDOABLY-SETQ ,PLACE (CONS ,OBJ ,PLACE))
  (MULTIPLE-VALUE-BIND (DUMMIES VALS NEWVAL SETTER GETTER)
    (GET-UNDOABLE-SETF-METHOD PLACE ENV)
    `(', 'LET* (,@(MAPCAR #'LIST DUMMIES VALS)
      (CAR NEWVAL)
      (CONS ,OBJ ,GETTER)))
    , SETTER))))
```

```
(DEFMACRO UNDOABLY-PUSHNEW (OBJ PLACE &REST KEYS &ENVIRONMENT ENV)
```

```
(IF (SYMBOLP PLACE)
  `(UNDOABLY-SETQ ,PLACE (ADJOIN ,OBJ ,PLACE ,@KEYS))
  (MULTIPLE-VALUE-BIND (DUMMIES VALS NEWVAL SETTER GETTER)
    (GET-UNDOABLE-SETF-METHOD PLACE ENV)
    `(', 'LET* (,@(MAPCAR #'LIST DUMMIES VALS)
      (CAR NEWVAL)
      (ADJOIN ,OBJ ,GETTER ,@KEYS))
      , SETTER))))
```

```
(DEFMACRO UNDOABLY-REMF (PLACE INDICATOR &ENVIRONMENT ENV)
(MULTIPLE-VALUE-BIND (DUMMIES VALS NEWVAL SETTER GETTER)
  (GET-UNDOABLE-SETF-METHOD PLACE ENV)
  (LET ((IND-TEMP (GENSYM))
        (LOCAL1 (GENSYM))
        (LOCAL2 (GENSYM)))
    `('LET* (,@(MAPCAR #'LIST DUMMIES VALS)
            (,(CAR NEWVAL)
              ,GETTER)
            ,(IND-TEMP ,INDICATOR))
      (DO ((,LOCAL1 ,(CAR NEWVAL)
            (CDDR ,LOCAL1))
          (,LOCAL2 NIL ,LOCAL1))
        ((ATOM ,LOCAL1)
         NIL)
        (COND
         ((ATOM (CDR ,LOCAL1))
          (ERROR "Odd-length property list in REMF."))
         ((EQ (CAR ,LOCAL1)
              ,IND-TEMP)
          (COND
           (,LOCAL2 (IL:/RPLACD (CDR ,LOCAL2)
                                (CDDR ,LOCAL1))
                    (RETURN T))
           (T (SETQ ,(CAR NEWVAL)
                   (CDDR ,(CAR NEWVAL)))
              ,SETTER
              (RETURN T))))))))))
```

```
(DEFMACRO UNDOABLY-ROTATEF (&REST ARGS &ENVIRONMENT ENV)
(COND
  ((NULL ARGS)
   NIL)
  ((NULL (CDR ARGS))
   ` (PROGN ,(CAR ARGS)
            NIL))
  (T (DO ((A ARGS (CDR A))
          (LET-LIST NIL)
          (SETF-LIST NIL)
          (NEXT-VAR NIL)
          (FIX-ME NIL))
        (ATOM A)
        (RPLACA FIX-ME NEXT-VAR)
        `('LET* ,(REVERSE LET-LIST)
              ,@(REVERSE SETF-LIST)
              NIL))
      (MULTIPLE-VALUE-BIND (DUMMIES VALS NEWVAL SETTER GETTER)
        (GET-UNDOABLE-SETF-METHOD (CAR A)
                                     ENV)
        (DO ((D DUMMIES (CDR D))
            (V VALS (CDR V))
            (NULL D))
          (PUSH (LIST (CAR D)
                    (CAR V))
                LET-LIST))
          (PUSH (LIST NEXT-VAR GETTER)
                LET-LIST)
          (UNLESS FIX-ME
            (SETQ FIX-ME (CAR LET-LIST)))
          (PUSH SETTER SETF-LIST)
          (SETQ NEXT-VAR (CAR NEWVAL))))))
```

```
(DEFMACRO UNDOABLY-SHIFTF (&REST ARGS &ENVIRONMENT ENV)
(COND
  ((OR (NULL ARGS)
        (NULL (CDR ARGS)))
   (ERROR "SHIFTF needs at least two arguments"))
  (T (DO* ((A ARGS (CDR A))
          (LET-LIST NIL)
          (SETF-LIST NIL)
          (RESULT (GENSYM))
          (NEXT-VAR RESULT))
        (ATOM (CDR A))
        (PUSH (LIST NEXT-VAR (CAR A))
              LET-LIST)
        `('LET* ,(REVERSE LET-LIST)
              ,@(REVERSE SETF-LIST)
              ,RESULT))
      (MULTIPLE-VALUE-BIND (DUMMIES VALS NEWVAL SETTER GETTER)
        (GET-UNDOABLE-SETF-METHOD (CAR A)
                                     ENV)
        (DO ((D DUMMIES (CDR D))
            (V VALS (CDR V))
```

```

  ((NULL D))
  (PUSH (LIST (CAR D)
              (CAR V))
        LET-LIST))
  (PUSH (LIST NEXT-VAR GETTER)
        LET-LIST)
  (PUSH SETTER SETF-LIST)
  (SETQ NEXT-VAR (CAR NEWVAL))))))

```

```

(DEFDEFINER DEFINE-UNDOABLE-MODIFY-MACRO IL:FUNCTIONS (NAME LAMBDA-LIST FUNCTION &OPTIONAL DOC-STRING)
  (LET ((OTHER-ARGS NIL)
        (REST-ARG NIL))
    (DO ((LL LAMBDA-LIST (CDR LL))
         (ARG NIL)
         ((NULL LL))
         (SETQ ARG (CAR LL))
         (COND
          ((EQ ARG '&OPTIONAL))
          ((EQ ARG '&REST)
           (SETQ REST-ARG (CADR LL))
           (RETURN NIL))
          ((SYMBOLP ARG)
           (PUSH ARG OTHER-ARGS))
          (T (PUSH (CAR ARG)
                  OTHER-ARGS))))
        (SETQ OTHER-ARGS (REVERSE OTHER-ARGS))
      `(DEFMACRO ,NAME (SI::%$MODIFY-MACRO-FORM ,@LAMBDA-LIST &ENVIRONMENT SI::%$MODIFY-MACRO-ENVIRONMENT)
        ,DOC-STRING (MULTIPLE-VALUE-BIND (DUMMIES VALS NEWVAL SETTER GETTER)
          (GET-UNDOABLE-SETF-METHOD SI::%$MODIFY-MACRO-FORM SI::%$MODIFY-MACRO-ENVIRONMENT)
          )
          (MULTIPLE-VALUE-BIND (DUMMIES VALS NEWVALS SETTER GETTER)
            (GET-SETF-METHOD SI::%$MODIFY-MACRO-FORM SI::%$MODIFY-MACRO-ENVIRONMENT)
            `(,'LET* (,@(MAPCAR #'LIST DUMMIES VALS)
                       (,(CAR NEWVALS)
                        ,(IF REST-ARG
                            `(LIST* ',FUNCTION GETTER ,@OTHER-ARGS ,REST-ARG)
                            `(LIST ',FUNCTION GETTER ,@OTHER-ARGS))))
              ,SETTER))))))

```

```

(DEFINE-UNDOABLE-MODIFY-MACRO UNDOABLY-DECF (&OPTIONAL (DELTA 1))
  -)

```

```

(DEFINE-UNDOABLE-MODIFY-MACRO UNDOABLY-INCF (&OPTIONAL (DELTA 1))
  +)

```

```

(DEFUN UNDOABLY-PROCLAIM (PROCLAMATION)
  ;; Undoable version of PROCLAIM.
  (WHEN (CONSP PROCLAMATION)
    (CASE (CAR PROCLAMATION)
      (SPECIAL (DOLIST (X (CDR PROCLAMATION))
                       (UNDOABLY (SETF (IL:VARIABLE-GLOBALLY-SPECIAL-P X)
                                         T)
                                     (SETF (IL:VARIABLE-GLOBAL-P X)
                                           NIL)
                                     (SETF (CONSTANTP X)
                                           NIL))))))
      (GLOBAL (DOLIST (X (CDR PROCLAMATION))
                      (UNDOABLY (SETF (IL:VARIABLE-GLOBAL-P X)
                                         T)
                                     (SETF (IL:VARIABLE-GLOBALLY-SPECIAL-P X)
                                           NIL)
                                     (SETF (CONSTANTP X)
                                           NIL))))))
      (SI::CONSTANT (DOLIST (X (CDR PROCLAMATION))
                            (UNDOABLY (SETF (CONSTANTP X)
                                              T)
                                          (SETF (IL:VARIABLE-GLOBAL-P X)
                                                NIL)
                                          (SETF (IL:VARIABLE-GLOBALLY-SPECIAL-P X)
                                                NIL))))))
      (DECLARATION (DOLIST (X (CDR PROCLAMATION))
                            (UNDOABLY (SETF (DECL-SPECIFIER-P X)
                                              T))))))
      (NOTINLINE (DOLIST (X (CDR PROCLAMATION))
                          (UNDOABLY (SETF (GLOBALLY-NOTINLINE-P X)
                                              T))))))
      (INLINE (DOLIST (X (CDR PROCLAMATION))
                      (UNDOABLY (SETF (GLOBALLY-NOTINLINE-P X)
                                         NIL))))))

```

```

(DEFUN MAKE-UNDOABLE (FORM &OPTIONAL ENV)
  (LIST 'UNDOABLY FORM))

```

```

(DEFMACRO STOP-UNDOABLY (&REST FORMS)
  ;; evaluate forms -- inside UNDOABLY, stops transformation
  (IL:MKPROGN FORMS))

(DEFUN UNDOABLY-SETF-SYMBOL-FUNCTION (SYMBOL DEFINITION)
  ;; NOTE: If you change this version, be sure to change the not-undoable version on LLSYMBOL!
  ;; undoable inverse of SYMBOL-FUNCTION
  (IL:VIRGINFN SYMBOL T)
  (COND
    ((CONSP DEFINITION)
     ;; Either it's a LAMBDA form or one of the special lists put together by SYMBOL-FUNCTION for macros and special forms.
     (CASE (CAR DEFINITION)
       (:MACRO (UNDOABLY-SETF (MACRO-FUNCTION SYMBOL)
                               (CDR DEFINITION)))
       (:SPECIAL-FORM (UNDOABLY-SETF (GET SYMBOL 'IL:SPECIAL-FORM)
                                       (CDR DEFINITION)))
       (T (IL:/PUTD SYMBOL DEFINITION T))))
     ;; If it's (SETF (SYMBOL-FUNCTION 'FOO) 'BAR) then we give FOO the same definition as BAR. This isn't quite like Lucid and Symbolics, but
     ;; it will do for now.
     ((AND (SYMBOLP DEFINITION)
            (NOT (NULL DEFINITION)))
      (IL:/PUTD SYMBOL (IL:GETD DEFINITION)
                    T))
     ;; It's probably a compiled-code object or an interpreted closure. In any case, go ahead and put it in there; if it's illegal, we'll find out when we try
     ;; to apply it.
     (T (IL:/PUTD SYMBOL DEFINITION T)))
  ;; (SETF (SYMBOL-FUNCTION ...) ...) is supposed to remove macro definitions. We only remove the ones that could come from DEFMACRO.
  (UNLESS (OR (NULL DEFINITION)
              (AND (CONSP DEFINITION)
                   (EQ (CAR DEFINITION)
                       :MACRO)))
    (IL:/REMPROP SYMBOL 'IL:MACRO-FN)
    DEFINITION))

(DEFUN UNDOABLY-SETF-MACRO-FUNCTION (X BODY)
  ;; undoable setf of macro-function
  ;; NOTE: If you change this, be sure to change the not-undoable version on CMLMACROS!
  (PROG1 (UNDOABLY-SETF (GET X 'IL:MACRO-FN)
                        BODY)
    (AND (IL:GETD X)
         (CASE (IL:ARGTYPE X)
           ((1 3)
            )
           (OTHERWISE (IL:/PUTD X NIL))))))
  ; Leave Interlisp nlambda definition alone

(IL:DECLARE\ : IL:DONTEVAL@LOAD IL:DONTEVAL@COMPILE IL:DOCOPY
(IL:MOVD 'UNDOABLY-SETF-SYMBOL-FUNCTION 'IL:UNDOABLY-SETF-SYMBOL-FUNCTION)
(IL:MOVD 'UNDOABLY-SETF-MACRO-FUNCTION 'UNDOABLY-SETF-MACRO-FUNCTION)
)

(IL:ADDTOVAR IL:LISPFNS (PROCLAIM . UNDOABLY-PROCLAIM)
              (POP . UNDOABLY-POP)
              (PSETF . UNDOABLY-PSETF)
              (PUSH . UNDOABLY-PUSH)
              (PUSHNEW . UNDOABLY-PUSHNEW)
              (REMF . UNDOABLY-REMF)
              (ROTATEF . UNDOABLY-ROTATEF)
              (SHIFTF . UNDOABLY-SHIFTF)
              (DECF . UNDOABLY-DECF)
              (INCF . UNDOABLY-INCF)
              (SET . UNDOABLY-SET-SYMBOL)
              (MAKUNBOUND . UNDOABLY-MAKUNBOUND)
              (FMAKUNBOUND . UNDOABLY-FMAKUNBOUND))

(DEFUN GET-UNDOABLE-SETF-METHOD (FORM &OPTIONAL ENVIRONMENT &AUX TEMP)
  (COND
    ((SYMBOLP FORM)
     (VALUES NIL NIL (LIST (SETQ TEMP (GENSYM))
                          `(IL:UNDOABLY-SET-SYMBOL ',FORM ,TEMP)
                          FORM))
     (NOT (CONSP FORM))
     (CL::SETF-ERROR FORM))
  )

```

```

((SETQ TEMP (IL:LOCAL-MACRO-FUNCTION (CAR FORM)
ENVIRONMENT))
;; always expand local macros
(GET-UNDOABLE-SETF-METHOD (FUNCALL TEMP FORM ENVIRONMENT)
ENVIRONMENT))
((SETQ TEMP (GET (CAR FORM)
':UNDOABLE-SETF-INVERSE))
;; found a special undoable property -- use it
(CL::GET-SIMPLE-SETF-METHOD FORM TEMP))
(T (BLOCK DONE
(MULTIPLE-VALUE-BIND (DUMMIES VALS NEWVAL SETTER GETTER)
(COND
((SETQ TEMP (OR (GET (CAR FORM)
':SETF-INVERSE)
(GET (CAR FORM)
':IL:SETF-INVERSE)
(GET (CAR FORM)
':IL:SETFN)))
(CL::GET-SIMPLE-SETF-METHOD FORM TEMP))
((SETQ TEMP (GET (CAR FORM)
':SHARED-SETF-INVERSE))
(CL::GET-SHARED-SETF-METHOD FORM TEMP))
((SETQ TEMP (OR (GET (CAR FORM)
':SETF-METHOD-EXPANDER)
(GET (CAR FORM)
':IL:SETF-METHOD-EXPANDER))))
(FUNCALL TEMP FORM ENVIRONMENT))
(T (MULTIPLE-VALUE-BIND (MAC MORE)
(MACROEXPAND-1 FORM ENVIRONMENT)
(IF (AND MORE (NOT (EQ MAC FORM)))
(RETURN-FROM DONE (GET-UNDOABLE-SETF-METHOD MAC ENVIRONMENT))
(ERROR "~S is not a known location specifier for SETF." (CAR FORM))))))
;; this is lexically correct, but doesn't work in bytecompiler, interlisp
;; (cl:values dummies vals newval '(cl:labels ((undostore (,@newval) (undosave (list #'undostore .getter) .setter)) (undostore
;; ,@newval)) getter)
;; so, instead we do the following, which binds the dummies too so that there are no free references. LABELS is used because the
;; thing saved on the undo list is also saved when the UNDO is undefined.
;;
(VALUEs DUMMIES VALS NEWVAL `(IL:COMMON-LISP (LABELS ((UNDOSTORE (,@DUMMIES ,@NEWVAL)
(IL:UNDOSAVE (LIST #'UNDOSTORE
,@DUMMIES
,GETTER))
,SETTER))
(UNDOSTORE ,@DUMMIES ,@NEWVAL)))
GETTER))))))

```

```

(DEFUN UNDOABLY-SET-SYMBOL (SYMBOL VALUE &OPTIONAL ENVIRONMENT)
(BLOCK UNDOABLY-SET-SYMBOL
(WHEN ENVIRONMENT

```

;; This function only saves undo info when there is no lexical binding for the variable.

```

(SETQ ENVIRONMENT (IL:ENVIRONMENT-VARS ENVIRONMENT))
(LOOP (IF (NULL ENVIRONMENT)
(RETURN NIL))
(IF (EQ SYMBOL (CAR ENVIRONMENT))
;; found a binding for this symbol
(PROGN (IF (EQ (CAR (SETQ ENVIRONMENT (CDR ENVIRONMENT)))
IL:*SPECIAL-BINDING-MARK*)
;; it is a special binding, or a mark that we are using the special value
(RETURN NIL) ; return from WHILE
)
(RPLACA ENVIRONMENT VALUE)
;; smash new value in
(RETURN-FROM UNDOABLY-SET-SYMBOL VALUE))
(SETQ ENVIRONMENT (CDDR ENVIRONMENT))))))

```

;; no environment, or not found.

```

(LET ((VP (IL:\\STKSCAN SYMBOL)))
(COND
((EQ (IL:\\HILOC VP)
IL:\\STACKHI)
(IL:\\PUTBASEPTR VP 0 VALUE))
(T (WHEN (CONSTANTP SYMBOL)
(UNLESS (EQL VALUE (IL:GETTOPVAL SYMBOL))
(CERROR "Go ahead and set it" "Attempt to set constant ~S to ~S" SYMBOL VALUE)))
(LET ((OLDVAL (IL:\\GETBASEPTR VP 0))
TEM)
(UNLESS (OR (NULL IL:LISPHIST)

```

```

(AND (SETQ TEM (SOME #'(LAMBDA (X)
                        (AND (CONSP X)
                             (EQ (CAR X)
                                  'IL:/SETTOPVAL)
                             (EQ (CADR X)
                                  SYMBOL)))
                        (IL:LISTGET1 IL:LISPHIST 'IL:SIDE)))
      (NOT (TAILP TEM (IL:LISTP IL:UNDOSIDE))))
      ;; special optimization from Interlisp: don't save more than one assignment of the same variable in the same event(!)
      (IL:UNDOSAVE (LIST 'IL:/SETTOPVAL SYMBOL OLDVAL)))
      (IL:\\RPLPTR VP 0 VALUE))))))

```

(IL:DEFINEQ

(UNDOABLY-SETQ

(IL:NLAMBDA VARVALUE

; Edited 8-Oct-87 18:54 by job
; Interlisp version

(UNDOABLY-SET-SYMBOL (CAR VARVALUE)
 (IL:\\EVPROG1 (CDR VARVALUE))))

)

(DEFINE-SPECIAL-FORM **UNDOABLY** (&REST FORMS &ENVIRONMENT ENV)

(LOOP (IF (NULL (CDR FORMS))
 (RETURN (UNDOHOOK (CAR FORMS)
 ENV))
 (UNDOHOOK (POP FORMS)
 ENV))))

(DEFINE-SPECIAL-FORM **UNDOABLY-SETQ** (&REST TAIL &ENVIRONMENT ENV)

(LET (VALUE)
 (LOOP (IF (NULL TAIL)
 (RETURN NIL)
 (SETQ VALUE (UNDOABLY-SET-SYMBOL (POP TAIL)
 (EVAL (POP TAIL)
 ENV)
 ENV))))
 VALUE))

(IL:DECLARE\ : IL:DONTEVAL@LOAD IL:DONTEVAL@COMPILE IL:DOCOPY

(IL:MOVD 'UNDOABLY-SET-SYMBOL 'IL:UNDOABLY-SET-SYMBOL)
)

(IL:PUTPROPS **IL:CMLUNDO IL:FILETYPE** :COMPILE-FILE)

(IL:PUTPROPS **IL:CMLUNDO IL:MAKEFILE-ENVIRONMENT** (:READTABLE "XCL" :PACKAGE "XCL"))

(IL:PUTPROPS **SYMBOL-FUNCTION :UNDOABLE-SETF-INVERSE** UNDOABLY-SETF-SYMBOL-FUNCTION)

(IL:PUTPROPS **MACRO-FUNCTION :UNDOABLE-SETF-INVERSE** UNDOABLY-SETF-MACRO-FUNCTION)

(IL:DECLARE\ : IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVERS

(IL:ADDTOVAR **IL:NLAMA** UNDOABLY-SETQ)

(IL:ADDTOVAR **IL:NLAML**)

(IL:ADDTOVAR **IL:LAMA**)

)

(IL:PUTPROPS **IL:CMLUNDO IL:COPYRIGHT** ("Venue & Xerox Corporation" 1986 1987 1988 1990 2022))

FUNCTION INDEX

GET-UNDOABLE-SETF-METHOD	6	UNDOABLY-INCF	5	UNDOABLY-SETF-SYMBOL-FUNCTION	6
MAKE-UNDOABLE	5	UNDOABLY-MAKUNBOUND	2	UNDOABLY-SETQ	8
NOHOOK	1	UNDOABLY-PROCLAIM	5	UNDOHOOK	2
UNDOABLY-DECF	5	UNDOABLY-SET-SYMBOL	7		
UNDOABLY-FMAKUNBOUND	2	UNDOABLY-SETF-MACRO-FUNCTION	6		

MACRO INDEX

STOP-UNDOABLY	6	UNDOABLY-POP	3	UNDOABLY-PUSH	3	UNDOABLY-REMF	4	UNDOABLY-SETF	2
UNDOABLY	1	UNDOABLY-PSETF	3	UNDOABLY-PUSHNEW ..	3	UNDOABLY-ROTATEF ..	4	UNDOABLY-SHIFTF ..	4

PROPERTY INDEX

IL:CMLUNDO	8	MACRO-FUNCTION	8	SYMBOL-FUNCTION	8
------------------	---	----------------------	---	-----------------------	---

SPECIAL-FORM INDEX

UNDOABLY	8	UNDOABLY-SETQ	8
----------------	---	---------------------	---

VARIABLE INDEX

IN-DEFINER	1	IL:LISPFNS	6
--------------------	---	------------------	---

DEFINER INDEX

DEFINE-UNDOABLE-MODIFY-MACRO	5
------------------------------------	---
