File created: 16-May-90 21:37:57 **{DSK}<usr>local>lde>lispcore>sources>SEDIT-ATOMIC.;3**

changes to: (IL:VARS IL:SEDIT-ATOMICCOMS)

previous date: 30-Mar-90 01:10:14 {DSK}<usr>local>lde>lispcore>sources>SEDIT-ATOMIC.;2

Read Table: XCL

Package: SEDIT

Format: XCCS


```
(IL:RPAQQ IL:SEDIT-ATOMICCOMS
          ((IL:PROP IL:FILETYPE IL:SEDIT-ATOMIC)
           (IL:PROP IL:MAKEFILE-ENVIRONMENT IL:SEDIT-ATOMIC)
           (IL:LOCALVARS . T)
           (IL:DECLARE\: IL:DONTCOPY IL:DOEVAL@COMPILE (IL:FILES IL:SEDIT-DECLS))
           (IL:FNS ASSIGN-FORMAT-LITATOM ATOM-POINT-TYPE BACKSPACE-GAP BACKSPACE-LITATOM BACKSPACE-UNKNOWN
                  CLOSE-NODE-LITATOM COMPUTE-POINT-POSITION-LITATOM COMPUTE-SELECTION-POSITION-LITATOM CONS-ATOM
                  COPY-SELECTION-LITATOM COPY-STRUCTURE-STRING DELETE-LITATOM DETRANSLATE-CHARS
                  GET-BUTTON-STRING GROW-SELECTION-LITATOM HASFAT INITIALIZE-ATOMIC INSERT-LITATOM INSERT-STRING
                  OPEN-LITATOM PARSE--BROKEN-ATOM PARSE--LITATOM PARSE--STRING RELEASE-OPEN-STRING REPLACE-CHARS
                  REPLACE-STRING SCAN-STRING SELECT-SEGMENT-LITATOM SET-POINT-LITATOM SET-POINT-STRING
                  SET-SELECTION-LITATOM SET-SELECTION-STRING SPLIT-LITATOM STRINGIFY-ATOM TRANSLATE-CHARS
                  UNDO-ATOM-CHANGE)))

(IL:PUTPROPS IL:SEDIT-ATOMIC IL:FILETYPE :COMPILE-FILE)

(IL:PUTPROPS IL:SEDIT-ATOMIC IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE (DEFPACKAGE IL:SEDIT
                                                                                  (:USE IL:LISP IL:XCL))))

(IL:DECLARE\: IL:DOEVAL@COMPILE IL:DONTCOPY

(IL:LOCALVARS . T)
)

(IL:DECLARE\: IL:DONTCOPY IL:DOEVAL@COMPILE

(IL:FILESLOAD IL:SEDIT-DECLS)
)

(IL:DEFINEQ
```

(ASSIGN-FORMAT-LITATOM

```
  (IL:LAMBDA (NODE CONTEXT FORMAT)                                    ; Edited 19-Nov-87 16:43 by DCB
```

;;; atoms have no children, so the format will not propagate further. normal nodes can get relinearized if their placement changes, even if their format ;;; type did not change. however, the presentation of an atom changes only when its format type changes (from KeyWord to NIL or back). thus to avoid ;;; unnecessary relinearization, the parse method for atoms builds a prelinearized node, and here we fiddle the prelinearized node to account for the ;;; change in font between KeyWord atoms and normal ones.

;;; note that when building the SEdit tree from scratch, the format will necessarily have changed, so this routine will get called whether the prelinearized ;;; node needs patching (changed to keyword) or not.

```
    (LET* ((ATOM (IL:|fetch| STRUCTURE IL:|of| NODE))
           (BROKEN? (IL:|type?| BROKEN-ATOM ATOM))
           (ENVIRONMENT (IL:\\DTEST (IL:|fetch| ENVIRONMENT IL:|of| CONTEXT)
                                    'EDIT-ENV))
           (FONT (COND
                     (BROKEN? (IL:SETQ ATOM (IL:FETCH ATOM-CHARS IL:OF ATOM))
                              (IL:FETCH BROKEN-ATOM-FONT IL:OF ENVIRONMENT))
                     ((EQ FORMAT :KEYWORD)
                      (IL:|ffetch| KEYWORD-FONT IL:|of| ENVIRONMENT))
                     (T (IL:|ffetch| DEFAULT-FONT IL:|of| ENVIRONMENT))))
           (STRING-ITEM (CAR (IL:|ffetch| LINEAR-FORM IL:|of| NODE)))
           WIDTH)                                                     ; read table specific
          (WHEN (AND (NOT (AND (IL:|ffetch| OPEN-NODE-CHANGED? IL:|of| CONTEXT)
                               (EQ (IL:FFETCH OPEN-NODE IL:OF CONTEXT)
                                   NODE)))
                     (OR (IL:FFETCH CHANGED? IL:OF NODE)
                         (AND (NOT BROKEN?)
                              (IL:STRINGP (IL:|ffetch| STRING IL:|of| STRING-ITEM)))
                         (IL:NEQ FONT (IL:|fetch| FONT IL:|of| STRING-ITEM))))
              ;; this stuff gets run only if the prelinearized node is wrong. (see comment above.)
              (IL:SETQ WIDTH (STRINGWIDTH ATOM FONT (NOT BROKEN?)))
              (IL:|freplace| STRING IL:|of| STRING-ITEM IL:|with| ATOM)
              (IL:|freplace| FONT IL:|of| STRING-ITEM IL:|with| FONT)
              (IL:|freplace| PRIN-2? IL:|of| STRING-ITEM IL:|with| (NOT BROKEN?))
              (IL:|freplace| WIDTH IL:|of| STRING-ITEM IL:|with| WIDTH)
              (IL:|freplace| INLINE-WIDTH IL:|of| NODE IL:|with| WIDTH)
              (IL:|freplace| PREFERRED-WIDTH IL:|of| NODE IL:|with| WIDTH)
              (IL:|freplace| ACTUAL-WIDTH IL:|of| NODE IL:|with| WIDTH)
              (IL:|freplace| ACTUAL-LLENGTH IL:|of| NODE IL:|with| WIDTH)
              (WHEN (EQ (IL:|ffetch| OPEN-NODE IL:|of| CONTEXT)
                        NODE)
```

```
                    (IL:|freplace| OPEN-NODE IL:|of| CONTEXT IL:|with| NIL))))))
```

(ATOM-POINT-TYPE

```
(IL:LAMBDA (STR INDEX)                                          ; Edited  7-Jul-87 08:26 by DCB
```

;;; used to pass read.table, but now under profile just use *READTABLE* directly.

```
    (IL:FOR C IL:INSTRING STR IL:AS I IL:TO INDEX IL:BIND (RESULT IL:_ 'ATOM)
                                                          (ESC-CHAR IL:_ (ESCAPE-CHAR))
                                                          (MULT-ESC-CHAR IL:_ (IL:FETCH (READTABLEP
                                                                                          IL:MULTESCAPECHAR)
                                                                                IL:OF *READTABLE*))
                                                          ESCAPED
       IL:DO (COND
                ((ESCAPED (IL:SETQ ESCAPED NIL))
                ((EQ C ESC-CHAR)
                 (IL:SETQ ESCAPED T))
                ((EQ C MULT-ESC-CHAR)
                 (IL:SETQ RESULT (IF (EQ RESULT 'ATOM)
                                     'ESC-ATOM
                                     'ATOM))))
       IL:FINALLY (RETURN RESULT))))
```

(BACKSPACE-GAP

```
(IL:LAMBDA (NODE CONTEXT INDEX)                                 ; Edited  7-Jul-87 08:26 by DCB
```

;;; handle the case of backspacing onto a gap. should pending delete select it.

```
    (COND
       (INDEX (IL:SHOULDNT "point can't be in a gap"))
       (T (SET-SELECTION-ME (IL:FETCH SELECTION IL:OF CONTEXT)
                 CONTEXT NODE)
          (PENDING-DELETE (IL:FETCH CARET-POINT IL:OF CONTEXT)
                 (IL:FETCH SELECTION IL:OF CONTEXT))))))
```

(BACKSPACE-LITATOM

```
(IL:LAMBDA (NODE CONTEXT INDEX STRING)                          ; Edited 24-Nov-87 08:14 by DCB
```

;; the BackSpace method for litatoms and strings

```
(COND
   ((NULL INDEX)
    ;; backspacing from the right boundary puts the caret immediately after the last character
    (IL:SET-POINT-LITATOM (IL:FETCH CARET-POINT IL:OF CONTEXT)
          CONTEXT NODE NIL T)
    (SET-SELECTION-NOWHERE (IL:FETCH SELECTION IL:OF CONTEXT)))
   ((EQ INDEX 0)
    (COND
       ((EQ 0 (IL:NCHARS STRING))
        ;; backspacing from the front of an empty string deletes it
        (DELETE-NODES (IL:FETCH SUPER-NODE IL:OF NODE)
               CONTEXT NODE NIL (IL:FETCH CARET-POINT IL:OF CONTEXT)))
       (T ;; might be at the front of a quote to degrade
          (LET* ((SUPER-NODE (IL:FETCH SUPER-NODE IL:OF NODE))
                 (SUPER-TYPE (IL:FETCH NODE-TYPE IL:OF SUPER-NODE)))
                (WHEN (EQ SUPER-TYPE TYPE-QUOTE)
                    ;; used to call backspace-quote directly here, now indirect through type
                    (FUNCALL (IL:FETCH BACK-SPACE IL:OF SUPER-TYPE)
                           SUPER-NODE CONTEXT 0))))))
   (T ;; otherwise, delete the character to the left of the caret
      (LET ((START INDEX)
            END)
           (WHEN (IL:NEQ (IL:FETCH NODE-TYPE IL:OF NODE)
                         TYPE-STRING)                           ; read table specific
               (IL:FOR I IL:FROM (IL:SUB1 INDEX) IL:TO 1 IL:BY -1 IL:BIND (ESC IL:_ (ESCAPE-CHAR))
                  IL:WHILE (EQ (IL:NTHCHARCODE STRING I)
                               ESC)
                  IL:FINALLY (WHEN (EVENP (IL:IDIFFERENCE I INDEX))
                                   (IL:SETQ END START)
                                   (IL:SETQ START (IL:SUB1 START)))))
               (IL:DELETE-LITATOM NODE CONTEXT START END (IL:FETCH CARET-POINT IL:OF CONTEXT)
                      STRING)
               (WHEN (NOT (DEAD-NODE? NODE))
                   ;; if the atom's still there, cancel the selection.  otherwise don't worry about it, the delete method might have set it
                   (SET-SELECTION-NOWHERE (IL:FETCH SELECTION IL:OF CONTEXT))))))))
```

(BACKSPACE-UNKNOWN

```
(IL:LAMBDA (NODE CONTEXT INDEX)                                 ; Edited  7-Jul-87 08:26 by DCB
```

```
    (COND
       (INDEX (IL:SHOULDNT "point shouldn't be in an unknown node"))
       (T ;; jump caret to before the unknown node
          (SET-POINT (IL:FETCH CARET-POINT IL:OF CONTEXT)
                 CONTEXT NODE NIL NIL NIL 'STRUCTURE)
          (SET-SELECTION-NOWHERE (IL:FETCH SELECTION IL:OF CONTEXT))))))
```

(CLOSE-NODE-LITATOM

```
  (IL:LAMBDA (CONTEXT NODE)                                   ; Edited 19-Nov-87 16:10 by DCB
    (COND
       ((AND (EQ NODE (IL:|fetch| ATOM-STARTED IL:|of| CONTEXT))
             (EQ (IL:|fetch| UNDO-LIST IL:|of| CONTEXT)
                 (IL:|fetch| ATOM-STARTED-UNDO-POINTER IL:|of| CONTEXT))
             (NULL (IL:|fetch| UNDO-UNDO-LIST IL:|of| CONTEXT)))      ; don't record this as a separate undo event
        (IL:|replace| ATOM-STARTED IL:|of| CONTEXT IL:|with| NIL)
        (IL:|replace| ATOM-STARTED-UNDO-POINTER IL:|of| CONTEXT IL:|with| NIL))
       (T (UNDO-BY UNDO-ATOM-CHANGE NODE (IL:|fetch| STRUCTURE IL:|of| NODE))))
    (LET* ((STRING-ITEM (CAR (IL:|fetch| LINEAR-FORM IL:|of| NODE)))
           (OLD-STRING (IL:|fetch| STRING IL:|of| STRING-ITEM))
           (NEW-STRING (IL:|replace| STRUCTURE IL:|of| NODE IL:|with| (CONS-ATOM OLD-STRING
                                                                                 (IL:NEQ (IL:|fetch| NODE-TYPE IL:|of| NODE)
                                                                                         TYPE-STRING)))))
          (IL:|replace| STRING IL:|of| STRING-ITEM IL:|with| (COND
                                                              ((IL:STRINGP NEW-STRING)
                                                               NEW-STRING)
                                                              (T ;; this is a litatom, so we have to make sure the string item has a copy
                                                                 (IL:SETQ NEW-STRING (IL:CONCAT OLD-STRING)))))
          (RELEASE-OPEN-STRING OLD-STRING NEW-STRING (IL:|fetch| CARET-POINT IL:|of| CONTEXT)
                 (IL:|fetch| SELECTION IL:|of| CONTEXT))
          (SUBNODE-CHANGED NODE CONTEXT)
          (COND
             ((EQ (IL:|fetch| NODE-TYPE IL:|of| NODE)
                  TYPE-STRING)
              (NOTE-CHANGE-IN-SIMPLE NODE CONTEXT)
              (IL:REPLACE OPEN-NODE IL:OF CONTEXT IL:WITH NIL))
             (T (NOTE-CHANGE NODE CONTEXT))))))
```

(COMPUTE-POINT-POSITION-LITATOM

```
  (IL:LAMBDA (POINT CONTEXT)                                  ; Edited  7-Jul-87 08:26 by DCB
    ;; implements the ComputePointPosition method for a litatom or string.  string.offset does all the work
    (LET ((NODE (IL:FETCH POINT-NODE IL:OF POINT)))
         ;; read table specific.  used to pass read.table as 5th arg to string.offset.  now just flag prin2? for it and it will do the right thing using the
         ;; current readtable.
         (STRING-OFFSET (IL:FETCH POINT-STRING IL:OF POINT)
                NIL
                (IL:FETCH POINT-INDEX IL:OF POINT)
                (IL:FETCH FONT IL:OF (CAR (IL:FETCH LINEAR-FORM IL:OF NODE)))
                (EQ (IL:FETCH NODE-TYPE IL:OF NODE)
                    TYPE-STRING)
                POINT
                (IL:FETCH START-X IL:OF NODE))
         (IL:REPLACE POINT-LINE IL:OF POINT IL:WITH (IL:FETCH FIRST-LINE IL:OF NODE)))))
```

(COMPUTE-SELECTION-POSITION-LITATOM

```
  (IL:LAMBDA (SELECTION CONTEXT)                              ; Edited  7-Jul-87 08:26 by DCB
    ;; implements the ComputeSelectionPosition method for a litatom or string.  string.offset does all the work
    (LET* ((NODE (IL:FETCH SELECT-NODE IL:OF SELECTION))
           (STRING-ITEM (CAR (IL:FETCH LINEAR-FORM IL:OF NODE))))  ; read table specific
          (STRING-OFFSET (IL:FETCH SELECT-STRING IL:OF SELECTION)
                 (IL:FETCH SELECT-START IL:OF SELECTION)
                 (OR (IL:FETCH SELECT-END IL:OF SELECTION)
                     (IL:FETCH SELECT-START IL:OF SELECTION))
                 (IL:FETCH FONT IL:OF STRING-ITEM)
                 (EQ (IL:FETCH NODE-TYPE IL:OF NODE)
                     TYPE-STRING)
                 SELECTION
                 (IL:FETCH START-X IL:OF NODE))
          (IL:REPLACE SELECT-START-LINE IL:OF SELECTION IL:WITH (IL:FETCH FIRST-LINE IL:OF NODE))
          (IL:REPLACE SELECT-END-LINE IL:OF SELECTION IL:WITH (IL:FETCH SELECT-START-LINE IL:OF SELECTION)))))
```

(CONS-ATOM

```
  (IL:LAMBDA (CHARS ATOM?)                                    ; Edited 19-Nov-87 14:37 by DCB
```

;;; read table specific. used to pass in read.table, but now must run under sedit profile using *readtable*

```
    (IF ATOM?
        (LET (RESULT)
             (IF (IL:SETQ RESULT (IL:NLSETQ (IL:READ (IL:OPENSTRINGSTREAM CHARS 'IL:INPUT))))
```

```
                (CAR RESULT)
                (IL:CREATE BROKEN-ATOM
                       ATOM-CHARS IL:_ (IL:CONCAT CHARS))))
        (COND
           ((NULL CHARS)
            (IL:ALLOCSTRING 0))
           ((HASFAT CHARS)
            (IL:CONCAT CHARS))
           (T (IL:\\SMASHSTRING (IL:ALLOCSTRING (IL:NCHARS CHARS))
                     0 CHARS))))))
```

(COPY-SELECTION-LITATOM

```
  (IL:LAMBDA (SELECTION CONTEXT DESTINATION POINT DELETE?)       ; Edited 7-Jul-87 08:31 by DCB
```

;;; implements the CopySelection method for a litatom or string. Assume under sedit profile..

```
    (LET* ((NODE (IL:FETCH SELECT-NODE IL:OF SELECTION))
           (CHARS (IL:FETCH STRUCTURE IL:OF NODE))
           (START (IL:FETCH SELECT-START IL:OF SELECTION))
           (END (IL:FETCH SELECT-END IL:OF SELECTION))
           (STRING (IL:FETCH SELECT-STRING IL:OF SELECTION))
           (TYPE (IL:FETCH SELECT-TYPE IL:OF SELECTION))
           NOT-ALL-SELECTED)
          (WHEN (IL:TYPE? BROKEN-ATOM CHARS)
              (IL:SETQ CHARS (IL:FETCH ATOM-CHARS IL:OF CHARS)))
          (WHEN (EQ TYPE 'STRUCTURE)
              (IL:SETQ STRING (GET-BUTTON-STRING NODE CONTEXT))
              (IL:SETQ TYPE (IF (EQ (IL:FETCH NODE-TYPE IL:OF NODE)
                                    TYPE-STRING)
                                'STRING
                                'ATOM)))
          (WHEN (AND START (OR (IL:NEQ (OR END (IL:SETQ END START))
                                       (IL:NCHARS STRING))
                               (IL:NEQ START 1)))
              ;; some subset of the atom/string has been selected
              (IL:SETQ NOT-ALL-SELECTED T))
          (COND
             ((NULL DESTINATION)
              ;; it's going to a foreign sink;  bksysbuf it
              (IL:BKSYSBUF (IF NOT-ALL-SELECTED
                               (DETRANSLATE-CHARS (IL:SUBSTRING STRING START END)
                                      TYPE)
                               CHARS)
                     (IF (EQ TYPE 'STRING)
                         (NULL START)
                         (NOT NOT-ALL-SELECTED)))
              (WHEN DELETE? (DELETE-NODES NODE CONTEXT START END NIL STRING)))
             ((AND (EQ TYPE 'STRING)
                   (NULL START))
              ;; strings insert as whole structures
              (COPY-SELECTION-DEFAULT SELECTION CONTEXT DESTINATION POINT DELETE?))
             ((EQ (IL:FETCH POINT-TYPE IL:OF POINT)
                  'STRUCTURE)
              ;; make the selected characters into a new atom or string
              (WITH-PROFILE (IL:FETCH PROFILE IL:OF DESTINATION)
                     (COND
                        ((AND DELETE? (IL:NEQ (IL:FETCH NODE-TYPE IL:OF NODE)
                                              TYPE-STRING)
                              (NOT NOT-ALL-SELECTED)
                              (DELETE-NODES NODE CONTEXT))
                         ;; if we're moving the whole atom, we can just reuse the node
                         ;; assume under sedit profile for this call to stringwidth
                         (ADJUST-WIDTH NODE NIL (STRINGWIDTH CHARS (IL:FETCH FONT
                                                                    IL:OF (CAR (IL:FETCH LINEAR-FORM
                                                                                IL:OF NODE)))
                                                                T)))
                        (T (WHEN (OR (EQ (IL:FETCH NODE-TYPE IL:OF NODE)
                                         TYPE-STRING)
                                     NOT-ALL-SELECTED
                                     (IL:TYPE? BROKEN-ATOM (IL:FETCH STRUCTURE IL:OF NODE)))
                                 (IL:SETQ CHARS (CONS-ATOM (IF START
                                                               (IL:SUBSTRING STRING START END)
                                                               STRING)
                                                           T))
                                 (WHEN (AND DELETE? NOT-ALL-SELECTED)
                                       (DELETE-NODES NODE CONTEXT START END NIL STRING)))
                           ;; again here need sedit profile to create.simple.node.
                           (IL:SETQ NODE (CREATE-SIMPLE-NODE CHARS (IL:FETCH ENVIRONMENT IL:OF DESTINATION)
                                                             TYPE-LITATOM
                                                             (IF (IL:TYPE? BROKEN-ATOM CHARS)
```

```
                                                    (IL:FETCH ATOM-CHARS IL:OF CHARS)
                                                   CHARS)
                                                (NOT (IL:TYPE? BROKEN-ATOM CHARS))
                                                (IF (IL:TYPE? BROKEN-ATOM CHARS)
                                                    (IL:FETCH BROKEN-ATOM-FONT IL:OF (IL:FETCH ENVIRONMENT
                                                                                       IL:OF DESTINATION))
                                                    (IL:FETCH DEFAULT-FONT IL:OF (IL:FETCH ENVIRONMENT
                                                                                   IL:OF DESTINATION)))))))
                    (INSERT POINT DESTINATION (LIST NODE))
                    (WHEN START (SET-POINT-LITATOM POINT DESTINATION NODE NIL T))))
            (T ;; we're adding characters to an existing string or atom
               (WITH-PROFILE (IL:FETCH PROFILE IL:OF DESTINATION)
                      (LET ((NEW-CHARS (IL:CONCAT (IF NOT-ALL-SELECTED
                                                      (IL:SUBSTRING STRING START END)
                                                      STRING))))
                           (WHEN DELETE? (DELETE-NODES NODE CONTEXT START END NIL STRING))
                           (INSERT POINT DESTINATION (DETRANSLATE-CHARS NEW-CHARS TYPE)))))))))
```

(COPY-STRUCTURE-STRING

```
(IL:LAMBDA (NODE CONTEXT)                                   ; Edited  7-Jul-87 08:31 by DCB
```

;;; the CopyStructure method for strings and litatoms. the Structure and LinearForm fields have already been filled in with the values from the node ;;; we're a copy of, but we want to copy these structures in case we decide to smash them later. Assume under sedit profile.

```
    (LET* ((STRUCTURE (IL:FETCH STRUCTURE IL:OF NODE))
           (FONT (IL:FETCH FONT IL:OF (CAR (IL:FETCH LINEAR-FORM IL:OF NODE))))
           (PRIN-2? (NOT (IL:TYPE? BROKEN-ATOM STRUCTURE))))
          (IL:REPLACE STRUCTURE IL:OF NODE IL:WITH (COND
                                                       ((IL:STRINGP STRUCTURE)
                                                        (IL:SETQ STRUCTURE (IL:CONCAT STRUCTURE)))
                                                       ((IL:TYPE? BROKEN-ATOM STRUCTURE)
                                                        (IL:CREATE BROKEN-ATOM
                                                               ATOM-CHARS IL:_ (IL:SETQ STRUCTURE
                                                                                    (IL:CONCAT (IL:FETCH ATOM-CHARS
                                                                                                 IL:OF STRUCTURE)))))
                                                       (T STRUCTURE)))
          (RPLACA (IL:FETCH LINEAR-FORM IL:OF NODE)
                  (IL:CREATE STRING-ITEM
                         STRING IL:_ STRUCTURE
                         FONT IL:_ FONT
                         PRIN-2? IL:_ PRIN-2?))
          ;; assume running under sedit profile for this call to stringwidth          ; read table specific
          (ADJUST-WIDTH NODE NIL (STRINGWIDTH STRUCTURE FONT PRIN-2?)))))
```

(DELETE-LITATOM

```
(IL:LAMBDA (NODE CONTEXT START END SET-POINT? STRING)          ; Edited 23-Nov-87 19:10 by DCB
```

;; the Delete method for strings and litatoms

```
    (COND
       ((AND (IL:NEQ (IL:FETCH NODE-TYPE IL:OF NODE)
                     TYPE-STRING)
             (EQ START 1)
             (EQ (OR END START)
                 (IL:NCHARS STRING)))
        ;; deleting all the characters in an atom deletes it
        (DELETE-NODES (IL:FETCH SUPER-NODE IL:OF NODE)
              CONTEXT NODE NIL SET-POINT?))
       (T (REPLACE-STRING NODE CONTEXT START (OR END START)
                 "" SET-POINT? STRING (AND (IL:NEQ (IL:FETCH NODE-TYPE IL:OF NODE)
                                                   TYPE-STRING)
                                           (ATOM-POINT-TYPE STRING START)))
          T))))
```

(DETRANSLATE-CHARS

```
(IL:LAMBDA (STR TYPE)                                       ; Edited 16-Jul-87 15:42 by DCB
```

;;; read table specific. used to take read.table, now just uses *READTABLE* for profiles.

```
    (WHEN (IL:NEQ TYPE 'STRING)
        (IL:SETQ STR (IL:COPYALL STR))
        (IL:|for| C IL:|instring| STR IL:|bind| ESCAPED? (LENGTH IL:_ 0)
                                        (ESC IL:_ (ESCAPE-CHAR))
                                        (MULTI-ESC IL:_ (IL:|fetch| (READTABLEP IL:MULTESCAPECHAR) IL:|of|
                                                                                                   *READTABLE*
                                                                                                   ))
                                        (UPCASE? IL:_ (IL:|fetch| (READTABLEP IL:CASEINSENSITIVE) IL:|of|
                                                                                                   *READTABLE*
                                                                                                   ))
          IL:|first| (IL:SETQ TYPE (AND UPCASE? (EQ TYPE 'ATOM)))
          IL:|do| (COND
                     (ESCAPED? NIL)
                     ((EQ C MULTI-ESC)
```

```
                    (IL:SETQ TYPE (AND UPCASE? (NOT TYPE)))
                    (IL:SETQ C NIL))
                   ((EQ C ESC)
                    (IL:SETQ ESCAPED? T)
                    (IL:SETQ C NIL))
                   ((AND TYPE (IL:IGEQ C (IL:CHARCODE \a))
                         (IL:ILEQ C (IL:CHARCODE \z)))
                    (IL:SETQ C (IL:IPLUS C (IL:CONSTANT (IL:IDIFFERENCE (IL:CHARCODE A)
                                                                        (IL:CHARCODE \a)))))))
               (WHEN C
                   (IL:RPLCHARCODE STR (IL:SETQ LENGTH (IL:ADD1 LENGTH))
                     C))
         IL:|finally| (IL:|replace| (IL:STRINGP IL:LENGTH) IL:|of| STR IL:|with| LENGTH)))
    STR))
```

(GET-BUTTON-STRING

```
  (IL:LAMBDA (NODE CONTEXT)                                          ; Edited  7-Jul-87 08:31 by DCB
```

;;; assume this is running under sedit profile.

```
    (COND
       ((EQ BUTTON-STRING-NODE NODE)
        BUTTON-STRING)
       (T (IL:SETQ BUTTON-STRING-NODE NODE)
          (IL:SETQ BUTTON-STRING (CAR (IL:FETCH LINEAR-FORM IL:OF NODE)))
                                                                     ; read table specific
          (IL:SETQ BUTTON-STRING (IF (AND (IL:NEQ (IL:FETCH NODE-TYPE IL:OF NODE)
                                                  TYPE-STRING)
                                          (IL:FETCH PRIN-2? IL:OF BUTTON-STRING))
                                     (PRIN1-TO-STRING (IL:FETCH STRING IL:OF BUTTON-STRING))
                                     (PRINC-TO-STRING (IL:FETCH STRING IL:OF BUTTON-STRING)))))))
```

(GROW-SELECTION-LITATOM

```
  (IL:LAMBDA (SELECTION CONTEXT NODE)                                ; Edited  7-Jul-87 08:31 by DCB
```

;;; the GrowSelection method for litatoms and strings. if the whole node is already selected, select the super; otherwise select the whole node

```
    (IF (NULL (IL:FETCH SELECT-START IL:OF SELECTION))
        (GROW-SELECTION-DEFAULT SELECTION CONTEXT NODE)
        (SET-SELECTION-ME SELECTION CONTEXT NODE))))
```

(HASFAT

```
  (IL:LAMBDA (STR)                                                   ; Edited 19-Nov-87 14:35 by DCB
    (IL:FOR C IL:INSTRING STR IL:THEREIS (IL:IGREATERP C IL:\\MAXTHINCHAR))))
```

(INITIALIZE-ATOMIC

```
  (IL:LAMBDA NIL                                                     ; Edited  7-Jul-87 08:31 by DCB
    (IL:SETQ TYPES (LIST* (IL:SETQ TYPE-LITATOM (IL:CREATE EDIT-NODE-TYPE
                                                           NAME IL:_ 'LITATOM
                                                           ASSIGN-FORMAT IL:_ 'ASSIGN-FORMAT-LITATOM
                                                           COMPUTE-FORMAT-VALUES IL:_ 'IL:NILL
                                                           LINEARIZE IL:_ NIL
                                                           SUB-NODE-CHANGED IL:_ 'IL:SHOULDNT
                                                           COMPUTE-POINT-POSITION IL:_
                                                           'COMPUTE-POINT-POSITION-LITATOM
                                                           COMPUTE-SELECTION-POSITION IL:_
                                                           'COMPUTE-SELECTION-POSITION-LITATOM
                                                           SET-POINT IL:_ 'SET-POINT-LITATOM
                                                           SET-SELECTION IL:_ 'SET-SELECTION-LITATOM
                                                           GROW-SELECTION IL:_ 'GROW-SELECTION-LITATOM
                                                           SELECT-SEGMENT IL:_ 'SELECT-SEGMENT-LITATOM
                                                           INSERT IL:_ 'INSERT-LITATOM
                                                           DELETE IL:_ 'DELETE-LITATOM
                                                           COPY-STRUCTURE IL:_ 'COPY-STRUCTURE-STRING
                                                           COPY-SELECTION IL:_ 'COPY-SELECTION-LITATOM
                                                           STRINGIFY IL:_ 'STRINGIFY-ATOM
                                                           BACK-SPACE IL:_ 'BACKSPACE-LITATOM
                                                           CLOSE-NODE IL:_ 'CLOSE-NODE-LITATOM))
                          (IL:SETQ TYPE-STRING (IL:CREATE EDIT-NODE-TYPE IL:USING TYPE-LITATOM NAME IL:_
                                                                                  'STRING ASSIGN-FORMAT IL:_
                                                                                  'IL:NILL SET-POINT IL:_
                                                                                  'SET-POINT-STRING SET-SELECTION
                                                                                  IL:_ 'SET-SELECTION-STRING INSERT
                                                                                  IL:_ 'INSERT-STRING))
                          TYPES))))
```

(INSERT-LITATOM

```
  (IL:LAMBDA (NODE CONTEXT WHERE CHAR POINT)                         ; Edited 17-Jul-87 09:47 by DCB
```

;;; the Insert method for litatoms

```
                                                                     ; read table specific
    (IL:INSERT-STRING NODE CONTEXT WHERE (AND CHAR (TRANSLATE-CHARS CHAR (IF (IL:TYPE? EDIT-SELECTION WHERE)
```

```
                                                                    (IL:FETCH SELECT-TYPE IL:OF WHERE)
                                                                    (IL:FETCH POINT-TYPE IL:OF WHERE))
                                                            (EQ *PRINT-CASE* :UPCASE)))
          POINT)))
```

(INSERT-STRING

```
  (IL:LAMBDA (NODE CONTEXT WHERE CHARS POINT)                    ; Edited 30-Nov-87 12:58 by DCB
```

;;; the Insert method for strings

```
    (LET (START END STRING TYPE)
         (COND
            ((IL:TYPE? EDIT-SELECTION WHERE)
             (IL:SETQ START (IL:FETCH SELECT-START IL:OF WHERE))
             (IL:SETQ END (OR (IL:FETCH SELECT-END IL:OF WHERE)
                              START))
             (IL:SETQ STRING (IL:FETCH SELECT-STRING IL:OF WHERE))
             (IL:SETQ TYPE (IL:FETCH SELECT-TYPE IL:OF WHERE)))
            (T (IL:SETQ END (IL:FETCH POINT-INDEX IL:OF WHERE))
               (IL:SETQ START (IL:ADD1 END))
               (IL:SETQ STRING (IL:FETCH POINT-STRING IL:OF WHERE))
               (IL:SETQ TYPE (IL:FETCH POINT-TYPE IL:OF WHERE))))
         ;; first replace any old chars with new chars
         (REPLACE-STRING NODE CONTEXT START END (OR CHARS "")
                POINT STRING TYPE)
         ;; now do any indicated split
         (UNLESS (OR CHARS (DEAD-NODE? NODE))
             (SPLIT-LITATOM NODE POINT CONTEXT START (1- START)
                    (IL:FETCH STRING IL:OF (CAR (IL:FETCH LINEAR-FORM IL:OF NODE)))))))
```

(OPEN-LITATOM

```
  (IL:LAMBDA (CONTEXT NODE STRING LENGTH)                        ; Edited  7-Jul-87 08:38 by DCB
    (WHEN (NULL LENGTH)
          (IL:SETQ LENGTH 0))
    (COND
       ((IL:NEQ (IL:FETCH OPEN-NODE IL:OF CONTEXT)
                NODE)
        (CLOSE-OPEN-NODE CONTEXT)
        (IL:REPLACE OPEN-NODE IL:OF CONTEXT IL:WITH NODE)
        (LET ((OPEN-STRING (IL:FETCH OPEN-NODE-INFO IL:OF CONTEXT))
              (STRING-LENGTH (IL:NCHARS STRING))
              (STRING-ITEM (CAR (IL:FETCH LINEAR-FORM IL:OF NODE)))
              SUB-STRING)
             (IL:REPLACE PRIN-2? IL:OF STRING-ITEM IL:WITH (EQ (IL:FETCH NODE-TYPE IL:OF NODE)
                                                               TYPE-STRING))
             (IL:REPLACE REAL-LENGTH IL:OF OPEN-STRING IL:WITH STRING-LENGTH)
             (IL:SETQ SUB-STRING (IL:FETCH SUBSTRING IL:OF OPEN-STRING))
             (WHEN (IL:ILESSP (IL:NCHARS (IL:FETCH BUFFER-STRING IL:OF OPEN-STRING))
                              (IL:SETQ LENGTH (IL:IPLUS STRING-LENGTH (IL:IMAX LENGTH 0))))
                 (IL:SUBSTRING (IL:REPLACE BUFFER-STRING IL:OF OPEN-STRING IL:WITH (IL:ALLOCSTRING LENGTH NIL NIL
                                                                                                   T))
                               1 1 SUB-STRING))
             (IL:RPLSTRING (IL:FETCH BUFFER-STRING IL:OF OPEN-STRING)
                    1 STRING)
             (IL:REPLACE (IL:STRINGP IL:LENGTH) IL:OF SUB-STRING IL:WITH STRING-LENGTH)
             (IL:REPLACE STRING IL:OF STRING-ITEM IL:WITH SUB-STRING)))
       (T (LET ((OPEN-STRING (IL:FETCH OPEN-NODE-INFO IL:OF CONTEXT)))
               (WHEN (IL:IGREATERP LENGTH 0)
                   (WHEN (IL:ILESSP (IL:NCHARS (IL:FETCH BUFFER-STRING IL:OF OPEN-STRING))
                                    (IL:SETQ LENGTH (IL:IPLUS (IL:FETCH REAL-LENGTH IL:OF OPEN-STRING)
                                                              LENGTH)))
                       (IL:SUBSTRING (IL:REPLACE BUFFER-STRING IL:OF OPEN-STRING
                                                 IL:WITH (IL:RPLSTRING (IL:ALLOCSTRING LENGTH NIL NIL T)
                                                                       1
                                                                       (IL:FETCH BUFFER-STRING IL:OF OPEN-STRING)))
                                     1 1 (IL:FETCH SUBSTRING IL:OF OPEN-STRING))))
               (IL:REPLACE STRING IL:OF (CAR (IL:FETCH LINEAR-FORM IL:OF NODE)) IL:WITH (IL:FETCH SUBSTRING
                                                                                                  IL:OF OPEN-STRING)))))))
```

(PARSE--BROKEN-ATOM

```
  (IL:LAMBDA (STRUCTURE CONTEXT MODE)                            ; Edited 17-Jul-87 09:04 by DCB
```

;;; parse a BrokenAtom structure (presumably left there by a previous editing session)

```
    (BUILD-PRELINEARIZED-NODE STRUCTURE CONTEXT TYPE-LITATOM (IL:FETCH ATOM-CHARS IL:OF STRUCTURE)
           NIL
           (IL:FETCH BROKEN-ATOM-FONT IL:OF (IL:FETCH ENVIRONMENT IL:OF CONTEXT))))
```

(PARSE--LITATOM

```
  (IL:LAMBDA (STRUCTURE CONTEXT)                                 ; Edited  7-Jul-87 08:38 by DCB
```

```
;;; parse a litatom (this actually includes numbers).

;;; this used to take a parse mode as an argument, and if the parse mode was BindingList, it would call parse..list.  parse..list now knows to make sure
;;; that its second child gets parsed as a list.

;;; when the atom turns to a keyword, its linearization will have to be twiddled.  see the comments in assign.format.litatom.

    (BUILD-PRELINEARIZED-NODE STRUCTURE CONTEXT TYPE-LITATOM STRUCTURE T (IL:FETCH DEFAULT-FONT
                                                                            IL:OF (IL:FETCH ENVIRONMENT
                                                                                    IL:OF CONTEXT)))))

(PARSE--STRING
  (IL:LAMBDA (STRUCTURE CONTEXT)                                    ; Edited  7-Jul-87 08:38 by DCB

;;; parse a string

    (BUILD-PRELINEARIZED-NODE STRUCTURE CONTEXT TYPE-STRING STRUCTURE T (IL:FETCH DEFAULT-FONT
                                                                          IL:OF (IL:FETCH ENVIRONMENT
                                                                                  IL:OF CONTEXT)))))

(RELEASE-OPEN-STRING
  (IL:LAMBDA (OLD-STRING NEW-STRING POINT SELECTION)                ; Edited  7-Jul-87 08:38 by DCB
    (WHEN (EQ (IL:FETCH POINT-STRING IL:OF POINT)
              OLD-STRING)
          (IL:REPLACE POINT-STRING IL:OF POINT IL:WITH NEW-STRING))
    (WHEN (EQ (IL:FETCH SELECT-STRING IL:OF SELECTION)
              OLD-STRING)
          (IL:REPLACE SELECT-STRING IL:OF SELECTION IL:WITH NEW-STRING))))

(REPLACE-CHARS
  (IL:LAMBDA (NODE CONTEXT START END CHARS POINT TYPE STRING-ITEM)
                                                                    ; Edited 28-Mar-90 19:14 by jds

;;; replace the substring of this (open) node bounded by start and end (inclusive) with the characters in chars.  set point after the inserted characters.
    ;; read table specific
    (IL:SETQ BUTTON-STRING-NODE (IL:SETQ BUTTON-STRING NIL))
    (LET* ((DELTA-LENGTH (IL:IDIFFERENCE (IL:NCHARS CHARS)
                                         (IL:ADD1 (IL:IDIFFERENCE END START))))
           (NEW-END (IL:IPLUS END DELTA-LENGTH))
           (PRIN-2? (AND (EQ TYPE 'STRING)
                         (IL:|fetch| PRIN-2? IL:|of| STRING-ITEM)))
           (MULTI-ESCAPE (AND (IL:NEQ TYPE 'STRING)
                              (IL:|fetch| (READTABLEP IL:MULTESCAPECHAR) IL:|of| *READTABLE*)))
           (ADD-MULTI-ESCAPE?)
           (COMPUTE-NEW-POINT-TYPE?)
           (OPEN-STRING (IL:|fetch| OPEN-NODE-INFO IL:|of| CONTEXT))
           (STRING (IL:|fetch| BUFFER-STRING IL:|of| OPEN-STRING))
           (LENGTH (IL:|fetch| REAL-LENGTH IL:|of| OPEN-STRING))
           (FONT (IL:|fetch| FONT IL:|of| STRING-ITEM))
           (DELTA-WIDTH (IL:IDIFFERENCE (STRINGWIDTH (IL:MKSTRING CHARS)
                                                     FONT PRIN-2?)
                                        (STRINGWIDTH (IF (IL:ILEQ START END)
                                                         (IL:SUBSTRING STRING START END)
                                                         "")
                                                     FONT PRIN-2?))))
      (WHEN MULTI-ESCAPE
          (IL:|bind| (ESCAPE IL:_ (ESCAPE-CHAR))
                C IL:|for| I IL:|from| START IL:|to| END IL:|do| (IL:SETQ C (IL:NTHCHARCODE STRING I))
                                                                 (COND
                                                                    ((EQ C ESCAPE)
                                                                     (IL:SETQ I (IL:ADD1 I)))
                                                                    ((EQ C MULTI-ESCAPE)
                                                                     (IL:SETQ ADD-MULTI-ESCAPE? (NOT
                                                                                                  ADD-MULTI-ESCAPE?
                                                                                                  )))))
          (WHEN ADD-MULTI-ESCAPE?
              (SETF COMPUTE-NEW-POINT-TYPE? T)
              (IL:SETQ ADD-MULTI-ESCAPE? (COND
                                           ((AND (IL:NEQ END LENGTH)
                                                 (EQ (IL:NTHCHARCODE STRING (IL:ADD1 END))
                                                     MULTI-ESCAPE))
                                            (IL:SETQ END (IL:ADD1 END))
                                            -1)
                                           (T 1)))
              (IL:SETQ DELTA-LENGTH (IL:IPLUS DELTA-LENGTH ADD-MULTI-ESCAPE?))
              (IL:SETQ DELTA-WIDTH (IL:IPLUS DELTA-WIDTH (IL:ITIMES ADD-MULTI-ESCAPE? (IL:CHARWIDTH
                                                                                        MULTI-ESCAPE
                                                                                        FONT))))))
      (IL:|replace| REAL-LENGTH IL:|of| OPEN-STRING IL:|with| (IL:IPLUS LENGTH DELTA-LENGTH))
      (COND
         ((AND (EQ 0 (IL:IPLUS LENGTH DELTA-LENGTH))
               (IL:NEQ (IL:FETCH NODE-TYPE IL:OF NODE)
                       TYPE-STRING))
```

```
            (CLOSE-OPEN-NODE CONTEXT)
            (DELETE-NODES (IL:FETCH SUPER-NODE IL:OF NODE)
                   CONTEXT NODE NIL POINT))
           (T (WHEN (IL:NEQ END LENGTH)
```

;; there are characters after the replacement, so shift them forward or backward as appropriate

```
                  (SHIFT-STRING STRING (IL:ADD1 END)
                         (IL:IPLUS END DELTA-LENGTH 1)
                         (IL:IDIFFERENCE LENGTH END)))
              (IL:RPLSTRING STRING START CHARS)
              (WHEN (EQ ADD-MULTI-ESCAPE? 1)
                  (IL:RPLCHARCODE STRING (IL:ADD1 NEW-END)
                         MULTI-ESCAPE))
              (IL:|replace| (IL:STRINGP IL:LENGTH) IL:|of| (IL:SETQ STRING (IL:|fetch| SUBSTRING IL:|of| OPEN-STRING))
                 IL:|with| (IL:IPLUS LENGTH DELTA-LENGTH))
              (ADJUST-WIDTH NODE CONTEXT (IL:IPLUS (IL:|fetch| INLINE-WIDTH IL:|of| NODE)
                                                   DELTA-WIDTH))
              (IL:|replace| OPEN-NODE-CHANGED? IL:|of| CONTEXT IL:|with| T)
              (WHEN POINT
                  (IL:|replace| POINT-NODE IL:|of| POINT IL:|with| NODE)
                  (IL:|replace| POINT-STRING IL:|of| POINT IL:|with| STRING)
                  (IL:|replace| POINT-INDEX IL:|of| POINT IL:|with| NEW-END)
                  (IL:|replace| POINT-TYPE IL:|of| POINT IL:|with| (IF COMPUTE-NEW-POINT-TYPE?
                                                                       (ATOM-POINT-TYPE STRING NEW-END)
                                                                       TYPE)))
              (LET ((CARET (IL:|fetch| CARET-POINT IL:|of| CONTEXT)))
                   (WHEN (AND (IL:NEQ CARET POINT)
                              (EQ (IL:FETCH POINT-NODE IL:OF CARET)
                                  NODE)
                              (IL:IGEQ (IL:FETCH POINT-INDEX IL:OF CARET)
                                       START))
```

;; if the caret was within or after replaced characters, it will need to be fixed up

```
                         (IL:REPLACE POINT-INDEX IL:OF CARET IL:WITH (IL:IPLUS DELTA-LENGTH
                                                                               (IL:IMAX (IL:FETCH POINT-INDEX
                                                                                                  IL:OF CARET)
                                                                                        END)))
                         (IL:REPLACE POINT-STRING IL:OF CARET IL:WITH STRING))))))))
```

(REPLACE-STRING

```
  (IL:LAMBDA (NODE CONTEXT START END CHARS POINT STRING TYPE)     ; Edited  7-Jul-87 08:39 by DCB
```

;;; replace the substring of this string node bounded by start and end (inclusive) with the characters in chars. set point after the inserted characters.

```
    (OPEN-LITATOM CONTEXT NODE STRING (IL:IDIFFERENCE (IL:NCHARS CHARS)
                                                      (IL:ADD1 (IL:IDIFFERENCE END START))))
    (REPLACE-CHARS NODE CONTEXT START END CHARS POINT (OR TYPE 'STRING)
           (CAR (IL:FETCH LINEAR-FORM IL:OF NODE)))))
```

(SCAN-STRING

```
  (IL:LAMBDA (POINT-OR-SEL NODE READ-TABLE STRING FONT OFFSET STRING?)
                                                                  ; Edited 17-Jul-87 09:07 by DCB
```

;;; given a string item and pixel offset from the start of the string, find the character pointed to. if string?, assume that string delim characters in the string
;;; are escaped and the string is preceded and followed by stringdelims

```
    (IL:BIND IN-MULTI-ESC C CWIDTH (INDEX IL:_ 0)
           (X IL:_ 0)
           (LENGTH IL:_ (IL:NCHARS STRING))
           (POINT? IL:_ (IL:TYPE? EDIT-POINT POINT-OR-SEL))
           (ESC-CHAR IL:_ (ESCAPE-CHAR READ-TABLE))
           (MULTI-ESC-CHAR IL:_ (IL:FETCH (READTABLEP IL:MULTESCAPECHAR) IL:OF (OR READ-TABLE *READTABLE*)))
       IL:FIRST (WHEN STRING?
                    (IL:SETQ CWIDTH (IL:CHARWIDTH (IL:CHARCODE \")
                                                  FONT))
                    (WHEN (AND (NOT POINT?)
                               (IL:ILEQ OFFSET CWIDTH))
                        (SET-SELECTION-ME POINT-OR-SEL NIL (IL:FETCH SELECT-NODE IL:OF POINT-OR-SEL))
                        (RETURN))
                    (IL:SETQ OFFSET (IL:IDIFFERENCE OFFSET CWIDTH))
                    (IL:SETQ X (IL:IPLUS X CWIDTH)))
       IL:WHILE (IL:ILEQ (IL:SETQ INDEX (IL:ADD1 INDEX))
                         LENGTH)
       IL:DO (IL:SETQ CWIDTH (IL:CHARWIDTH (IL:SETQ C (IL:NTHCHARCODE STRING INDEX))
                                           FONT))
             (IF STRING?
                 (COND
                    ((OR (EQ C ESC-CHAR)
                         (EQ C (IL:CHARCODE \")))
                     (IL:SETQ CWIDTH (IL:IPLUS CWIDTH (IL:CHARWIDTH ESC-CHAR FONT))))
                    ((IL:ILESSP C (IL:CHARCODE IL:SPACE))
                     (IL:SETQ CWIDTH (IL:IPLUS (IL:CHARWIDTH (IL:CHARCODE ^)
                                                             FONT)
                                               (IL:CHARWIDTH (IL:IPLUS C 64)
                                                             FONT)))))
```

```
                (WHEN (EQ C ESC-CHAR)
                    (IL:SETQ CWIDTH (IL:IPLUS CWIDTH (IL:CHARWIDTH (IL:NTHCHARCODE STRING (IL:ADD1 INDEX))
                                                                   FONT)))))
            (WHEN (IL:ILEQ OFFSET (IF POINT?
                                      (IL:HALF CWIDTH)
                                      CWIDTH))
                  (GO IL:$$OUT))
            (IL:SETQ OFFSET (IL:IDIFFERENCE OFFSET CWIDTH))
            (IL:SETQ X (IL:IPLUS X CWIDTH))
            (WHEN (NOT STRING?)
                (COND
                   ((EQ C ESC-CHAR)
                    (IL:SETQ INDEX (IL:ADD1 INDEX)))
                   ((EQ C MULTI-ESC-CHAR)
                    (IL:SETQ IN-MULTI-ESC (NOT IN-MULTI-ESC)))))
        IL:FINALLY (WHEN (AND STRING? (NOT POINT?)
                              (IL:IGREATERP INDEX LENGTH))
                       (SET-SELECTION-ME POINT-OR-SEL NIL (IL:FETCH SELECT-NODE IL:OF POINT-OR-SEL))
                       (RETURN))
            (IL:SETQ IN-MULTI-ESC (COND
                                      (STRING? 'STRING)
                                      (IN-MULTI-ESC 'ESC-ATOM)
                                      (T 'ATOM)))
            (COND
               (POINT? (IL:REPLACE POINT-INDEX IL:OF POINT-OR-SEL IL:WITH (IL:SUB1 INDEX))
                       (IL:REPLACE POINT-TYPE IL:OF POINT-OR-SEL IL:WITH IN-MULTI-ESC)
                       (IL:REPLACE POINT-LINE IL:OF POINT-OR-SEL IL:WITH (IL:FETCH FIRST-LINE IL:OF NODE))
                       (IL:REPLACE POINT-X IL:OF POINT-OR-SEL IL:WITH (IL:IPLUS (IL:FETCH START-X IL:OF NODE)
                                                                                X)))
               (T (WHEN (AND (NOT STRING?)
                             (IL:IGREATERP INDEX LENGTH))
                        (IL:SHOULDNT "select past end of atom"))
                  (COND
                     ((NOT (IL:FETCH SELECT-START IL:OF POINT-OR-SEL))
                      (IL:REPLACE SELECT-START IL:OF POINT-OR-SEL IL:WITH INDEX)
                      (IL:REPLACE SELECT-END IL:OF POINT-OR-SEL IL:WITH (AND (NOT STRING?)
                                                                             (EQ C ESC-CHAR)
                                                                             (IL:ADD1 INDEX)))
                      (IL:REPLACE SELECT-TYPE IL:OF POINT-OR-SEL IL:WITH IN-MULTI-ESC)
                      (IL:REPLACE SELECT-START-LINE IL:OF POINT-OR-SEL IL:WITH (IL:REPLACE SELECT-END-LINE
                                                                                  IL:OF POINT-OR-SEL
                                                                                  IL:WITH (IL:FETCH FIRST-LINE
                                                                                             IL:OF NODE)))
                      (IL:REPLACE SELECT-END-X IL:OF POINT-OR-SEL
                         IL:WITH (IL:IPLUS CWIDTH (IL:REPLACE SELECT-START-X IL:OF POINT-OR-SEL
                                                     IL:WITH (IL:IPLUS (IL:FETCH START-X IL:OF NODE)
                                                                       X)))))
                     (T                                              ; extending a point.or.sel
                        (WHEN (NOT (IL:FETCH SELECT-END IL:OF POINT-OR-SEL))
                            (IL:REPLACE SELECT-END IL:OF POINT-OR-SEL IL:WITH (IL:FETCH SELECT-START IL:OF
                                                                                                POINT-OR-SEL
                                                                                                    )))
                        (COND
                           ((IL:ILESSP INDEX (IL:FETCH SELECT-START IL:OF POINT-OR-SEL))
                                                                     ; extend the point.or.sel to the left
                            (IL:REPLACE SELECT-START IL:OF POINT-OR-SEL IL:WITH INDEX)
                            (IL:REPLACE SELECT-START-X IL:OF POINT-OR-SEL IL:WITH (IL:IPLUS (IL:FETCH START-X
                                                                                              IL:OF NODE)
                                                                                            X))
                            (IL:REPLACE SELECT-TYPE IL:OF POINT-OR-SEL IL:WITH IN-MULTI-ESC))
                           ((IL:IGREATERP (IF (AND (NOT STRING?)
                                                   (EQ C ESC-CHAR))
                                              (IL:SETQ INDEX (IL:ADD1 INDEX))
                                              INDEX)
                                          (IL:FETCH SELECT-END IL:OF POINT-OR-SEL))
                                                                     ; extend the point.or.sel to the right
                            (IL:REPLACE SELECT-END IL:OF POINT-OR-SEL IL:WITH INDEX)
                            (IL:REPLACE SELECT-END-X IL:OF POINT-OR-SEL IL:WITH (IL:IPLUS (IL:FETCH START-X
                                                                                            IL:OF NODE)
                                                                                          X CWIDTH)))))))))))
```

(SELECT-SEGMENT-LITATOM

```
(IL:LAMBDA (SELECTION CONTEXT NODE SUBNODE INDEX OFFSET ITEM)  ; Edited 24-Nov-87 09:53 by DCB
```

;;; the SelectSegment method for litatoms and strings. scan.string does most of the work

```
    ;; pass NIL as readtable
    (SCAN-STRING SELECTION NODE NIL (IL:FETCH SELECT-STRING IL:OF SELECTION)
           (IL:FETCH FONT IL:OF ITEM)
           OFFSET
           (EQ (IL:FETCH NODE-TYPE IL:OF NODE)
               TYPE-STRING))))
```

(SET-POINT-LITATOM

```
(IL:LAMBDA (POINT CONTEXT NODE INDEX OFFSET ITEM TYPE COMPUTE-LOCATION?)
```

; Edited 24-Nov-87 09:54 by DCB

;;; the SetPoint method for litatoms

```
(COND
   ((EQ TYPE 'STRUCTURE)
    ;; structure points will have to be handled by our super
    (PUNT-SET-POINT POINT CONTEXT NODE (IF INDEX
                                           (IL:IGEQ OFFSET (IL:HALF (IL:FETCH WIDTH IL:OF ITEM)))
                                           OFFSET)
          COMPUTE-LOCATION?))
   (T (IL:REPLACE POINT-NODE IL:OF POINT IL:WITH NODE)
      (IL:REPLACE POINT-STRING IL:OF POINT IL:WITH (GET-BUTTON-STRING NODE CONTEXT))
      (COND
         ((NOT INDEX)
          ;; placing the caret at the beginning or end of the atom
          (IL:REPLACE POINT-INDEX IL:OF POINT IL:WITH (IF OFFSET
                                                          (IL:NCHARS BUTTON-STRING)
                                                          0))
          (IL:REPLACE POINT-TYPE IL:OF POINT IL:WITH 'ATOM)
          (WHEN COMPUTE-LOCATION?
              (IL:REPLACE POINT-X IL:OF POINT IL:WITH (IL:IPLUS (IL:FETCH START-X IL:OF NODE)
                                                                (IF OFFSET
                                                                    (IL:FETCH INLINE-WIDTH IL:OF NODE)
                                                                    0)))
              (IL:REPLACE POINT-LINE IL:OF POINT IL:WITH (IL:FETCH FIRST-LINE IL:OF NODE))))
         (T ;; pass NIL as readtable
            (SCAN-STRING POINT NODE NIL BUTTON-STRING (IL:FETCH FONT IL:OF ITEM)
                  OFFSET)))))))
```

(SET-POINT-STRING

```
(IL:LAMBDA (POINT CONTEXT NODE INDEX OFFSET ITEM TYPE COMPUTE-LOCATION?)
```

; Edited 7-Jul-87 08:39 by DCB

;;; the SetPoint method for strings. the point must be *inside* the delimiting quotes

```
(COND
   ((EQ TYPE 'STRUCTURE)
    (PUNT-SET-POINT POINT CONTEXT NODE (IF INDEX
                                           (IL:IGEQ OFFSET (IL:HALF (IL:FETCH WIDTH IL:OF ITEM)))
                                           OFFSET)
          COMPUTE-LOCATION?))
   ((NOT INDEX)
    (PUNT-SET-POINT POINT CONTEXT NODE OFFSET COMPUTE-LOCATION?))
   (T (IL:REPLACE POINT-NODE IL:OF POINT IL:WITH NODE)
      (IL:REPLACE POINT-STRING IL:OF POINT IL:WITH (GET-BUTTON-STRING NODE CONTEXT))
      ;; pass NIL as readtable
      (SCAN-STRING POINT NODE NIL BUTTON-STRING (IL:FETCH FONT IL:OF ITEM)
            OFFSET T))))
```

(SET-SELECTION-LITATOM

```
(IL:LAMBDA (SELECTION CONTEXT NODE INDEX OFFSET ITEM TYPE)
```

; Edited 24-Nov-87 09:55 by DCB

;;; the SetSelection method for litatoms

```
(COND
   ((EQ TYPE 'STRUCTURE)
    ;; structure selections get it all
    (SET-SELECTION-ME SELECTION CONTEXT NODE))
   (T (IL:REPLACE SELECT-NODE IL:OF SELECTION IL:WITH NODE)
      (IL:REPLACE SELECT-STRING IL:OF SELECTION IL:WITH (GET-BUTTON-STRING NODE CONTEXT))
      (IL:REPLACE SELECT-START IL:OF SELECTION IL:WITH NIL)
      ;; pass NIL as readtable
      (SCAN-STRING SELECTION NODE NIL BUTTON-STRING (IL:FETCH FONT IL:OF ITEM)
            OFFSET))))
```

(SET-SELECTION-STRING

```
(IL:LAMBDA (SELECTION CONTEXT NODE INDEX OFFSET ITEM TYPE)
```

; Edited 7-Jul-87 08:39 by DCB

;;; the SetSelection method for strings

```
(COND
   ((EQ TYPE 'STRUCTURE)
    ;; structure selections or pointing at the delimiting quotes gets the whole string
    (SET-SELECTION-ME SELECTION CONTEXT NODE))
   (T (IL:REPLACE SELECT-NODE IL:OF SELECTION IL:WITH NODE)
      (IL:REPLACE SELECT-STRING IL:OF SELECTION IL:WITH (GET-BUTTON-STRING NODE CONTEXT))
```

```
          (IL:REPLACE SELECT-START IL:OF SELECTION IL:WITH NIL)
          ;; pass NIL as readtable
          (SCAN-STRING SELECTION NODE NIL BUTTON-STRING (IL:FETCH FONT IL:OF ITEM)
                 OFFSET T)))))
```

(SPLIT-LITATOM

```
  (IL:LAMBDA (NODE POINT CONTEXT START END STRING)                    ; Edited  7-Jul-87 08:39 by DCB
```

;;; the Split method for litatoms and strings

```
    (IL:SETQ BUTTON-STRING-NODE (IL:SETQ BUTTON-STRING NIL))
    (LET ((LENGTH (IL:NCHARS STRING))
          SUFFIX)
         (COND
            ((AND (IL:NEQ (IL:FETCH NODE-TYPE IL:OF NODE)
                          TYPE-STRING)
                  (EQ START 1)
                  (EQ END LENGTH))
             ;; deleting all the characters in an atom deletes it
             (CLOSE-OPEN-NODE CONTEXT)
             (DELETE-NODES (IL:FETCH SUPER-NODE IL:OF NODE)
                   CONTEXT NODE NIL POINT))
            (T (WHEN (NOT (AND (EQ START (IL:ADD1 END))
                               (OR (EQ START 1)
                                   (EQ END LENGTH))))
                   ;; something's got to be changed
                   (OPEN-LITATOM CONTEXT NODE STRING)
                   (LET ((OPEN-STRING (IL:FETCH OPEN-NODE-INFO IL:OF CONTEXT))
                         NEW-LENGTH)
                        (COND
                           ((EQ END LENGTH)
                            (IL:SETQ NEW-LENGTH (IL:SUB1 START)))
                           ((EQ START 1)
                            (IL:SETQ NEW-LENGTH (IL:IDIFFERENCE LENGTH END))
                            (SHIFT-STRING (IL:FETCH BUFFER-STRING IL:OF OPEN-STRING)
                                   (IL:ADD1 END)
                                   1 NEW-LENGTH))
                           (T (IL:SETQ NEW-LENGTH (IL:SUB1 START))
                              (IL:SETQ SUFFIX (IL:SUBSTRING (IL:FETCH BUFFER-STRING IL:OF OPEN-STRING)
                                                            (IL:ADD1 END)
                                                            LENGTH))))
                        (IL:REPLACE REAL-LENGTH IL:OF OPEN-STRING IL:WITH NEW-LENGTH)
                        (IL:REPLACE (IL:STRINGP IL:LENGTH) IL:OF (IL:FETCH SUBSTRING IL:OF OPEN-STRING)
                           IL:WITH NEW-LENGTH)
                        (IL:REPLACE OPEN-NODE-CHANGED? IL:OF CONTEXT IL:WITH T)))
               (WHEN SUFFIX (START-UNDO-BLOCK))
               (CLOSE-OPEN-NODE CONTEXT)
               (PUNT-SET-POINT POINT CONTEXT NODE (OR (IL:NEQ START 1)
                                                      (EQ END LENGTH))
                      NIL)
               (WHEN SUFFIX
                   (WHEN (IL:FETCH POINT-NODE IL:OF POINT)
                       (LET (STRING)
                            ;; use string to handle broken atoms: if the suffix is a broken atom, string will be the chars
                            (COND
                               ((IL:NEQ (IL:FETCH NODE-TYPE IL:OF NODE)
                                        TYPE-STRING)              ; read table specific
                                (IL:SETQ SUFFIX (CONS-ATOM SUFFIX T))
                                (WHEN (IL:TYPE? BROKEN-ATOM SUFFIX)
                                     (IL:SETQ STRING (IL:FETCH ATOM-CHARS IL:OF SUFFIX))))
                               (T (IL:SETQ SUFFIX (IL:CONCAT SUFFIX))))
                            (IL:SETQ SUFFIX (CREATE-SIMPLE-NODE SUFFIX (IL:FETCH ENVIRONMENT IL:OF CONTEXT)
                                                               (IL:FETCH NODE-TYPE IL:OF NODE)
                                                               (OR STRING SUFFIX)
                                                               (NULL STRING)
                                                               (IL:FETCH DEFAULT-FONT IL:OF (IL:FETCH ENVIRONMENT
                                                                                              IL:OF CONTEXT)))))
                       (INSERT POINT CONTEXT SUFFIX)
                       (PUNT-SET-POINT POINT CONTEXT SUFFIX NIL NIL))
               (END-UNDO-BLOCK))))))
```

(STRINGIFY-ATOM

```
  (IL:LAMBDA (NODE ENVIRONMENT)                                       ; Edited  7-Jul-87 08:39 by DCB
                                                                      ; read table specific
    (IL:MKSTRING (IL:FETCH STRUCTURE IL:OF NODE)
           T)))
```

(TRANSLATE-CHARS

```
  (IL:LAMBDA (CHARS POINT-TYPE UPCASE?)                               ; Edited 16-Jul-87 15:36 by DCB
```

;;; read table specific. used to take read.table, now just uses *READTABLE* for profiles.

```
    (WHEN (NOT (IL:FETCH (READTABLEP IL:CASEINSENSITIVE) IL:OF *READTABLE*))
          (IL:SETQ UPCASE? T))
    (IL:BIND (ESC IL:_ (ESCAPE-CHAR))
           (MULT-ESC IL:_ (IL:FETCH (READTABLEP IL:MULTESCAPECHAR) IL:OF *READTABLE*))
           (R IL:_ "") IL:FIRST (WHEN (EQ (IL:NCHARS CHARS)
                                          1)
                                      (IL:SETQ C (IL:CHCON1 CHARS))
                                      (RETURN (IF (AND (IL:NEQ C ESC)
                                                       (IL:NEQ C MULT-ESC)
                                                       (OR (EQ POINT-TYPE 'ESC-ATOM)
                                                           (NOT (ATOM-CHAR-ESCAPED C))))
                                                  (IF (OR UPCASE? (IL:ILESSP C (IL:CHARCODE A))
                                                          (IL:IGREATERP C (IL:CHARCODE Z))
                                                          (EQ POINT-TYPE 'ESC-ATOM))
                                                      CHARS
                                                      (IL:CHARACTER (IL:IPLUS C (IL:CONSTANT (IL:IDIFFERENCE
                                                                                              (IL:CHARCODE \a)
                                                                                              (IL:CHARCODE A))))))
                                                  (IL:CONCAT (IL:CHARACTER ESC)
                                                             CHARS))))
      IL:FOR C IL:INSTRING CHARS IL:DO (IL:SETQ R (IF (AND (IL:NEQ C ESC)
                                                           (IL:NEQ C MULT-ESC)
                                                           (OR (EQ POINT-TYPE 'ESC-ATOM)
                                                               (NOT (ATOM-CHAR-ESCAPED C))))
                                                      (IL:CONCAT R
                                                             (IL:CHARACTER
                                                              (IF (OR UPCASE? (IL:ILESSP C (IL:CHARCODE A))
                                                                      (IL:IGREATERP C (IL:CHARCODE Z))
                                                                      (EQ POINT-TYPE 'ESC-ATOM))
                                                                  C
                                                                  (IL:IPLUS C (IL:CONSTANT (IL:IDIFFERENCE
                                                                                            (IL:CHARCODE \a)
                                                                                            (IL:CHARCODE A))))))
                                                             )
                                                      (IL:CONCAT R (IL:CHARACTER ESC)
                                                             (IL:CHARACTER C))))
      IL:FINALLY (RETURN R))))
```

(**UNDO-ATOM-CHANGE**

```
  (IL:LAMBDA (CONTEXT NODE OLD-VALUE)                    ; Edited 7-Jul-87 08:40 by DCB
    (UNDO-BY UNDO-ATOM-CHANGE NODE (IL:FETCH STRUCTURE IL:OF NODE))
    (IL:REPLACE STRUCTURE IL:OF NODE IL:WITH OLD-VALUE)
    (SUBNODE-CHANGED NODE CONTEXT)
    (IF (EQ (IL:FETCH NODE-TYPE IL:OF NODE)
            TYPE-STRING)
        (NOTE-CHANGE-IN-SIMPLE NODE CONTEXT)
        (NOTE-CHANGE NODE CONTEXT))))

)

(IL:PUTPROPS IL:SEDIT-ATOMIC IL:COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1990))
```

FUNCTION INDEX

PROPERTY INDEX