

File created: 1-Aug-88 11:37:03 {DSK}<LISPPFILES>LOGIC>MEDLEY>UNIFIER.;1

previous date: 13-Jul-88 15:26:58 {DSK}<LISPPFILES>LOGIC>UNIFIER.;1

Read Table: INTERLISP

Package: USER

Format: XCCS

::
;; Copyright (c) 1987, 1988 by Roberto Ghislanzoni. All rights reserved.

```
(IL:RPAQQ IL:UNIFIERCOMS ((IL:FUNCTIONS BINDING BUILD-NEW-ENV CREATE-NEW-VARIABLE CREATE-VARIABLES  
FIND-IF-MEMBER FIND-VALUES FIND-VARIABLE-VALUE LOOKUP NULLP RENAME RENAME-VARS  
UNIFY VARIABLEP)))
```

```
(DEFUN BINDING (PREDICATE THEORY-NAME &OPTIONAL WINDOW)  
[COND  
[(EQ THEORY-NAME '*BACKGROUND-THEORY*]  
(COND  
[(EQ (CHAR-CODE (CHAR (SYMBOL-NAME PREDICATE)  
0))  
33)  
;; CUT is handled in a very particular way!!  
(GETHASH '! (GET 'THEORY '*BACKGROUND-THEORY*]  
(T (GETHASH PREDICATE (GET 'THEORY '*BACKGROUND-THEORY*]  
(T (GETHASH PREDICATE (GET-THEORY THEORY-NAME WINDOW]))
```

```
(DEFUN BUILD-NEW-ENV (PAT DAT ENV)  
;; It is better to make a distinction between the null value of a variable and the variables unbound  
(COND  
( (NULL DAT)  
(ACONS PAT '*NULL* ENV))  
(T (ACONS PAT DAT ENV))))
```

```
(DEFUN CREATE-NEW-VARIABLE ()  
[PROGN (SETF *VARIABLES-COUNTER* (+ 1 *VARIABLES-COUNTER*))  
(OR (GETHASH *VARIABLES-COUNTER* *VARIABLES-TABLE*)  
(SETF (GETHASH *VARIABLES-COUNTER* *VARIABLES-TABLE*)  
(MAKE-SYMBOL (FORMAT NIL "?~A" *VARIABLES-COUNTER*]))
```

```
(DEFUN CREATE-VARIABLES ()  
(DEFVAR *VARIABLES-TABLE* (MAKE-HASH-TABLE))  
;; all the variables used are cached in a hash-table: this is also for not generating a lot of symbols that will fill up the symbol table of the system  
;; This function must be called before starting to work with Logic  
(DO ((X 0 (+ X 1))  
( (= X 4095)  
T)  
(SETF (GETHASH X *VARIABLES-TABLE*)  
(MAKE-SYMBOL (FORMAT NIL "?~A" X))))
```

```
(DEFUN FIND-IF-MEMBER (ELT LST)  
(COND  
( (NULL LST)  
NIL)  
[ (LISTP LST)  
(OR (FIND-IF-MEMBER ELT (CAR LST))  
(FIND-IF-MEMBER ELT (CDR LST))  
( (ATOM LST)  
(EQ LST ELT))  
(T (MEMBER ELT LST)))]
```

```
(DEFUN FIND-VALUES (ELT ENV)  
(COND  
( (NULL ELT)  
NIL)  
( (LISTP ELT)  
(CONS (FIND-VALUES (CAR ELT)  
ENV)  
(FIND-VALUES (CDR ELT)  
ENV)))  
( (VARIABLEP ELT)  
(FIND-VARIABLE-VALUE ELT ENV))  
(T ELT)))
```

```

(DEFUN FIND-VARIABLE-VALUE (VAR ENV)
  [LET [(VAL (CDR (ASSOC VAR ENV)]
        (COND
          ((VARIABLEP VAL)
            (FIND-VARIABLE-VALUE VAL ENV))
          ((NULL VAL)
            ;; The variable is unbound, so the variable itself is returned
            VAR)
          ((NULLP VAL)
            ;; NULLP checks is the value is *NULL*
            NIL)
          (T ;; This is the statement for a partial occur check
            (OR (AND (NOT (FIND-IF-MEMBER VAR VAL))
                    (FIND-VALUES VAL ENV))
                VAL])
        ]])

```

```

(DEFUN LOOKUP (EXPR ENV)
  [COND
    ((NUMBERP EXPR)
     EXPR)
    ((SYMBOLP EXPR)
     (FIND-VALUES EXPR ENV))
    (T (CONS (FIND-VALUES (CAR EXPR)
                          ENV)
              (FIND-VALUES (CDR EXPR)
                          ENV)))

```

```

(DEFMACRO NULLP (ATOM)
  `(EQ ,ATOM '*NULL*))

```

```

(DEFUN RENAME (EXPR)
  (LET ((VARSTABLE (MAKE-HASH-TABLE)))
    (DECLARE (SPECIAL VARSTABLE))
    (RENAME-VARS EXPR)))

```

```

(DEFUN RENAME-VARS (EXPR)
  (COND
    ((NULL EXPR)
     NIL)
    [(LISTP EXPR)
     (CONS (RENAME-VARS (CAR EXPR))
           (RENAME-VARS (CDR EXPR))
           [(VARIABLEP EXPR)
            (LET ((ALREADY-RENAMED (GETHASH EXPR VARSTABLE)))
              (COND
                ((ALREADY-RENAMED ALREADY-RENAMED)
                 (T (LET ((CREATE-NEW-VARIABLE))
                     (SETF (GETHASH EXPR VARSTABLE)
                           NEW)
                     NEW)
                (T EXPR))
            ]))

```

```

(DEFUN UNIFY (PATT DAT ENV &OPTIONAL WINDOW)
  ;; This is a very fast implementation of unifier: no stack frames are generated. The tecnique used here is that of save-rest argument: the unifier is not
  ;; a true-recursive procedure, in the sense that it does not require a full stack for its implementation: in fact, when failure occurs, the value FAILED
  ;; must be immediately returned
  [PROG ([DEBUGFLG (AND WINDOW (TRACINGP WINDOW 'UNIFY))
        (REST-PAT)
        (REST-DAT)
        TEMP)
        HERE
        (AND DEBUGFLG (UNIFY-DEBUGGER PATT DAT ENV WINDOW)) ; debugging stuff
        [COND
          [(AND (NULL PATT)
                (NULL DAT))
            (COND
              ((AND (NULL REST-DAT)
                    REST-PAT)
               (RETURN 'FAILED))
              ((AND (NULL REST-PAT)
                    REST-DAT)
               (RETURN 'FAILED))
              ((AND (NULL REST-PAT)
                    (NULL REST-DAT))
               (RETURN ENV))
              (T (SETF PATT (CAR REST-PAT))
                  (SETF DAT (CAR REST-DAT))
                  (SETF REST-PAT (CDR REST-PAT))

```

```

      (SETF REST-DAT (CDR REST-DAT))
      (GO HERE]
    ((EQ ENV 'FAILED)
     (RETURN 'FAILED))
    (EQ PATT DAT)
    (GO OUT))
  [(VARIABLEP DAT)
   (SETF TEMP (CDR (ASSOC DAT ENV)))
   (COND
    ((NULL TEMP)
     (SETF ENV (BUILD-NEW-ENV DAT PATT ENV))
     (GO OUT))
    (T (SETF DAT TEMP)
        (GO HERE]
  [(VARIABLEP PATT)
   (SETF TEMP (CDR (ASSOC PATT ENV)))
   (COND
    ((NULL TEMP)
     (SETF ENV (BUILD-NEW-ENV PATT DAT ENV))
     (GO OUT))
    (T (SETF PATT TEMP)
        (GO HERE]
  [(NULL PATT)
   (COND
    ((NULLP DAT)
     (GO OUT))
    (T (RETURN 'FAILED]
  [(NULL DAT)
   (COND
    ((NULLP PATT)
     (GO OUT))
    (T (RETURN 'FAILED]
  [(LISTP PATT)
   (COND
    ((LISTP DAT)
     (SETF REST-PAT (CONS (REST PATT)
                          REST-PAT))
     (SETF REST-DAT (CONS (REST DAT)
                          REST-DAT))
     (SETF PATT (CAR PATT))
     (SETF DAT (CAR DAT))
     (GO HERE))
    (T (RETURN 'FAILED]
  (T (RETURN 'FAILED]

```

OUT

;; a check is made for the end of the procedure

```

(COND
 ((AND (NULL REST-PAT)
        (NULL REST-DAT))
  (RETURN ENV))
 (T (SETF DAT NIL)
     (SETF PATT NIL)
     (GO HERE]))

```

```

(DEFMACRO VARIABLEP (ITEM)
  `(AND (SYMBOLP ,ITEM)
        (EQ (CHAR-CODE (CHAR (SYMBOL-NAME ,ITEM)
                              0))
            63)))

```

(IL:PUTPROPS IL:UNIFIER IL:COPYRIGHT ("Roberto Ghislanzoni" 1987 1988))

FUNCTION INDEX

BINDING	1	CREATE-VARIABLES	1	FIND-VARIABLE-VALUE	2	RENAME-VARS	2
BUILD-NEW-ENV	1	FIND-IF-MEMBER	1	LOOKUP	2	UNIFY	2
CREATE-NEW-VARIABLE	1	FIND-VALUES	1	RENAME	2		

MACRO INDEX

NULLP	2	VARIABLEP	3
-------------	---	-----------------	---
