

File created: 19-Dec-88 15:40:33 {DSK}<LISPPFILES>LOGIC>MEDLEY>LOGIC-DEVEL.;7

changes to: (IL:VARS IL:LOGIC-DEVELCOMS)  
(IL:FUNCTIONS EDIT-AXIOM EDIT-SA PROMPTREAD)

previous date: 19-Dec-88 14:47:38 {DSK}<LISPPFILES>LOGIC>MEDLEY>LOGIC-DEVEL.;6

Read Table: INTERLISP

Package: USER

Format: XCCS

::  
:: Copyright (c) 1987, 1988 by ROBERTO GHISLANZONI. All rights reserved.

(IL:RPAQQ **IL:LOGIC-DEVELCOMS**

```
((IL:* IL:NOW IL:THESE IL:ARE IL:MACROS)
(IL:FUNCTIONS DRIBBLEP SEE-ENV-P SET-MENU TRACINGP)
(IL:* IL:THESE IL:ARE IL:FUNCTIONS)
(IL:FUNCTIONS CREATE-DEVEL-THEORY CREATE-THEORY-MENU EDIT-AXIOM EDIT-SA GET-LIST-PROP GET-TB-PROPERTY
GET-THEORY-INTERNAL LIST-ALL-THEORIES-INTERNAL LOAD-DEVEL-THEORY LOGIC-BUTTONFN
LOGIC-DEVELOPER LOGIC-MENU-FUNCTION MERGE-THEORIES-DEVEL MY-CREATE-TBRECORD PRINT-TB-ITEMS
PROMPTREAD SAVE-DEVEL-THEORY SHOW-PROFILE SOLVE-DEBUGGER START-PROVING UNIFY-DEBUGGER)
(IL:ADDVARS (IL:BackgroundMenuCommands ("Logic" '(IL:ADD.PROCESS '(LOGIC-DEVELOPER))
"Open a window on logic programming environment")))
(IL:VARS *LOGIC-MENU-ITEMS* *LOGIC-RELEASE-NUMBER* *LOGIC-CLOSE-ON-COMPLETION-FLG* (IL:BackgroundMenu
NIL)
(IL:LogicMiddleMenu NIL)
IL:LogicMiddleMenuCommands)
(IL:P (IL:FILESLOAD IL:TABLEBROWSER))
(IL:RECORDS IL:TABLEBROWSER IL:TABLEITEM)
(IL:CONSTANTS IL:TB.LEFT.MARGIN))
```

(IL:\* IL:\* IL:NOW IL:THESE IL:ARE IL:MACROS)

(DEFMACRO **DRIBBLEP** (WINDOW TYPE)

```
`[COND
((EQ ,TYPE 'SOLVE)
(IL:GETWINDOWPROP (IL:GETWINDOWPROP ,WINDOW 'IL:SOLVE-WINDOW)
'IL:TYPESCRIPTSTREAM))
((EQ ,TYPE 'UNIFY)
(IL:GETWINDOWPROP (IL:GETWINDOWPROP ,WINDOW 'IL:UNIFY-WINDOW)
'IL:TYPESCRIPTSTREAM])
```

(DEFMACRO **SEE-ENV-P** (WINDOW)

```
`(IL:GETWINDOWPROP WINDOW 'IL:SEE))
```

(DEFMACRO **SET-MENU** (MENU FIELD VALUE)

```
`(SETF (IL:FETCH ,FIELD IL:OF ,MENU)
,VALUE))
```

(DEFMACRO **TRACINGP** (WINDOW TYPE)

```
`[COND
((EQ ,TYPE 'SOLVE)
(EQ (IL:GETWINDOWPROP ,WINDOW 'IL:SOLVE)
'TRACE))
((EQ ,TYPE 'UNIFY)
(EQ (IL:GETWINDOWPROP ,WINDOW 'IL:UNIFY)
'TRACE))
```

(IL:\* IL:\* IL:THESE IL:ARE IL:FUNCTIONS)

(DEFUN **CREATE-DEVEL-THEORY** (MAIN-WINDOW)

```
[PROG* [(PW (IL:GETPROMPTWINDOW MAIN-WINDOW))
(THEORY-NAME (PROGN (IL:CLEARW PW)
(PROMPTREAD "Theory name" PW T])
```

:: Every theory is stored in a tablebrowser as a tableitem

```
(AND THEORY-NAME (LET* [(ACTUAL-THEORY (MAKE-HASH-TABLE))
(TB-ITEM (MY-CREATE-TBRECORD (ACONS 'THEORY ACTUAL-THEORY
(ACONS 'THEORY-NAME THEORY-NAME NIL)
(IL:CLEARW PW)
(IL:TB.INSERT.ITEM (IL:GETWINDOWPROP MAIN-WINDOW 'IL:TABLEBROWSER)
TB-ITEM)
(IL:CLEARW PW)
(FORMAT PW "~%%Theory created"))])
```

(DEFUN **CREATE-THEORY-MENU** (MAINW)

```
;; For speed improving, the list of all theories are kept in a menu; this menu is updated every time the user makes a change
(IL:PUTWINDOWPROP MAINW 'IL:THEORIES-MENU (PROG ((MENU (IL:CREATE IL:MENU)))
(SET-MENU MENU IL:TITLE "Which theory?")
(SET-MENU MENU IL:ITEMS (IL:SORT (LIST-ALL-THEORIES MAINW)
#'IL:ALPHORDER))
(RETURN MENU))))
```

(DEFUN **EDIT-AXIOM** (WINDOW)

```
[LET [(CHOSEN-THEORY-NAME (IL:MENU (IL:GETWINDOWPROP WINDOW 'IL:THEORIES-MENU)
(AND CHOSEN-THEORY-NAME (LET* ((THEORY (GET-THEORY CHOSEN-THEORY-NAME WINDOW))
[CHOSEN-AXIOM (PROG ((MENU (IL:CREATE IL:MENU)))
(SET-MENU MENU IL:TITLE "Which axiom?")
[SET-MENU MENU IL:ITEMS (APPEND
(LIST '--NEW--)
(IL:SORT (ALL-PREDS
THEORY)
(RETURN (IL:MENU MENU]
*AXIOM-TEMPLATE*)
(AND CHOSEN-AXIOM
(COND
[ (EQ CHOSEN-AXIOM '--NEW--)
(LET* ((PW (IL:GETPROMPTWINDOW WINDOW))
(NEWNAME (PROGN (IL:CLEARW PW)
(PROMPTREAD "Axiom name" PW T)))
PROC-NAME)
(PROG NIL
[SETF *AXIOM-TEMPLATE*
(LIST (LIST (LIST 'PREDICATE]
LP (IL:SPAWN.MOUSE)
[SETF *AXIOM-TEMPLATE*
(IL:EDITE (IL:COPYALL *AXIOM-TEMPLATE*)
NIL
(FORMAT NIL "New Predicate: ~A in ~A
theory " NEWNAME
CHOSEN-THEORY-NAME)
NIL NIL (AND
*LOGIC-CLOSE-ON-COMPLETION-FLG*
':CLOSE-ON-COMPLETION]
[PROG ((CLAUSE-NUMBER 1)
(AXS *AXIOM-TEMPLATE*))
LP1 (COND
((NULL AXS)
(RETURN))
((NOT (EQ NEWNAME (CAAR AXS)))
(IL:CLEARW PW)
(FORMAT PW "Clause number ~D: incorrect
predicate name: ~A" CLAUSE-NUMBER
(CAAR AXS))
(GO LP))
(T (SETF AXS (CDR AXS))
(INCF CLAUSE-NUMBER)
(GO LP1]
(SETF (GETHASH NEWNAME THEORY)
*AXIOM-TEMPLATE*])
(T (PROG NIL
(IL:TTYDISPLAYSTREAM (IL:GETPROMPTWINDOW WINDOW))
(SETF *AXIOM-TEMPLATE* (GETHASH CHOSEN-AXIOM THEORY))
LP (IL:SPAWN.MOUSE)
[SETF *AXIOM-TEMPLATE* (IL:EDITE (IL:COPYALL
*AXIOM-TEMPLATE*
)
NIL
(FORMAT NIL "Predicate: ~A
in ~A theory "
CHOSEN-AXIOM
CHOSEN-THEORY-NAME
)
NIL NIL
(AND
*LOGIC-CLOSE-ON-COMPLETION-FLG*
':CLOSE-ON-COMPLETION
]
[PROG ((CLAUSE-NUMBER 1)
(AXS *AXIOM-TEMPLATE*))
LP1 (COND
((NULL AXS)
(RETURN))
((NOT (EQ CHOSEN-AXIOM (CAAR AXS)))
(IL:CLEARW (IL:GETPROMPTWINDOW WINDOW))
(FORMAT (IL:GETPROMPTWINDOW WINDOW)
"Clause number ~D: incorrect predicate
name: ~A" CLAUSE-NUMBER (CAAR AXS))
(GO LP))
```

```
(T (SETF AXS (CDR AXS))
   (INCF CLAUSE-NUMBER)
   (GO LP1)
 (SETF (GETHASH CHOSEN-AXIOM THEORY)
 *AXIOM-TEMPLATE*])
```

```
(DEFUN EDIT-SA (WINDOW)
  [LET* [(CHOSEN-THEORY-NAME (IL:MENU (IL:GETWINDOWPROP WINDOW 'IL:THEORIES-MENU)
    (AND CHOSEN-THEORY-NAME (LET* ((THEORY (GET-THEORY CHOSEN-THEORY-NAME WINDOW))
      [CHOSEN-SA (PROG ((MENU (IL:CREATE IL:MENU)))
        (SET-MENU MENU IL:TITLE "Which sa?")
        [SET-MENU MENU IL:ITEMS (APPEND (LIST '--NEW--)
          (IL:SORT (ALL-SAS THEORY)
            (RETURN (IL:MENU MENU)
              *SA-TEMPLATE*)
            (AND CHOSEN-SA
              (COND
                [(EQ CHOSEN-SA '--NEW--)
                  (LET* [(PW (IL:GETPROMPTWINDOW WINDOW))
                    (NEWNAME (PROGN (IL:CLEARW PW)
                      (PROMPTREAD "SA name" PW T]
                    (PROGN (SETF *SA-TEMPLATE* (LIST 'LAMBDA
                      (LIST 'ARGS)
                        '<BODY>))
                    (IL:SPAWN.MOUSE)
                    (SETF *SA-TEMPLATE*
                      (IL:EDITE (IL:COPYALL *SA-TEMPLATE*)
                        NIL
                          (FORMAT NIL "New SA: ~A in ~A theory
                            (SETF (GETHASH NEWNAME THEORY)
                              (CONS 'SA *SA-TEMPLATE*])
                            (T (PROGN (IL:TTYDISPLAYSTREAM (IL:GETPROMPTWINDOW WINDOW))
                              (SETF *SA-TEMPLATE* (CDR (GETHASH CHOSEN-SA THEORY)))
                              (IL:SPAWN.MOUSE)
                              (SETF *SA-TEMPLATE* (IL:EDITE (IL:COPYALL
                                *SA-TEMPLATE*)
                                  NIL
                                    (FORMAT NIL "SA: ~A in ~A
                                      theory " CHOSEN-SA
                                        CHOSEN-THEORY-NAME)
                                      ))
                                (SETF (GETHASH CHOSEN-SA THEORY)
                                  (CONS 'SA *SA-TEMPLATE*])
                                  (DEFUN GET-LIST-PROP (TI-LIST PROPERTY)
                [PROG (RES)
                  LABEL
                    (COND
                      ((NULL TI-LIST)
                        (RETURN RES))
                      (T [SETF RES (APPEND RES (LIST (GET-TB-PROPERTY (CAR TI-LIST)
                        PROPERTY]
                          (SETF TI-LIST (CDR TI-LIST))
                          (GO LABEL])
                    (DEFUN GET-TB-PROPERTY (ITEM PROP)
                (IL:LISTGET (IL:FETCH IL:TIDATA IL:OF ITEM)
                  PROP))
                (DEFUN GET-THEORY-INTERNAL (THEORY-NAME &OPTIONAL WINDOW)
                [LET* ((TB (IL:GETWINDOWPROP WINDOW 'IL:TABLEBROWSER))
                  (ITEMS (IL:FETCH IL:TBITEMS IL:OF TB)))
                  (PROG NIL
                    LABEL
                      (COND
                        ((NULL ITEMS))
                        [(STRING-EQUAL (SYMBOL-NAME (GET-TB-PROPERTY (CAR ITEMS)
                          'THEORY-NAME))
                          (SYMBOL-NAME THEORY-NAME))
                          (RETURN (GET-TB-PROPERTY (CAR ITEMS)
                            'THEORY])
                          (T (SETF ITEMS (CDR ITEMS))
                            (GO LABEL])
                    (DEFUN LIST-ALL-THEORIES-INTERNAL (WINDOW)
                (GET-LIST-PROP (IL:TB.COLLECT.ITEMS (IL:GETWINDOWPROP WINDOW 'IL:TABLEBROWSER))
                  'THEORY-NAME))
                (DEFUN LOAD-DEVEL-THEORY (MAINW &OPTIONAL NAME-OF-THEORY)
```

```
(LET* [(PW (IL:GETPROMPTWINDOW MAINW))
      [THEORY-NAME (OR NAME-OF-THEORY (PROGN (IL:CLEARW PW)
                                             (PROMPTREAD "Theory name" PW T)
                                             [THEORY-FILE-NAME (MERGE-PATHNAMES (MAKE-PATHNAME :NAME THEORY-NAME :TYPE 'LGC)
                                             (ACTUAL-THEORY (MAKE-HASH-TABLE))
                                             (TB-ITEM (MY-CREATE-TBRECORD (ACONS 'THEORY ACTUAL-THEORY (ACONS 'THEORY-NAME THEORY-NAME NIL)
                                             (IL:CLEARW PW)
                                             (OR (AND (PROBE-FILE THEORY-FILE-NAME)
                                                    (PROGN [WITH-OPEN-FILE (FILE THEORY-FILE-NAME :DIRECTION :INPUT)
                                                            (FORMAT PW "Loading theory ")
                                                            (PROG (THEORY-NAME PRED-NUMBER SAS-NUMBER)
                                                                (READ FILE)

                                                                ;; skip on the theory name
                                                                (SETF SAS-NUMBER (READ FILE))
                                                                (DO ((SAS SAS-NUMBER (DECF SAS)))
                                                                    ((EQ SAS 0)
                                                                     NIL)
                                                                    (PROGN (FORMAT PW ".")
                                                                        (SETF (GETHASH (READ FILE)
                                                                    ACTUAL-THEORY)
                                                                    (READ FILE))))
                                                                (SETF PRED-NUMBER (READ FILE))
                                                                (DO ((PREDS PRED-NUMBER (DECF PREDS)))
                                                                    ((= PREDS 0)
                                                                     NIL)
                                                                    (PROGN (FORMAT PW ".")
                                                                        (SETF (GETHASH (READ FILE)
                                                                    ACTUAL-THEORY)
                                                                    (READ FILE)))))]
                                                    (IL:TB.INSERT.ITEM (IL:GETWINDOWPROP MAINW 'IL:TABLEBROWSER)
                                                                    TB-ITEM)
                                                    (IL:CLEARW PW)
                                                    (FORMAT PW "~%%Theory loaded")
                                                    T))
                                             (FORMAT PW "~%%Theory not found")))]
```

```
(DEFUN LOGIC-BUTTONFN (WINDOW)
  [COND
   ((IL:LASTMOUSESTATE IL:LEFT)
    T)
   ((IL:LASTMOUSESTATE IL:MIDDLE)
    (CASE (IL:MENU (OR IL:LogicMiddleMenu (PROGN (SETF IL:LogicMiddleMenu (IL:CREATE IL:MENU)
                                                    (SET-MENU IL:LogicMiddleMenu IL:ITEMS
                                                                IL:LogicMiddleMenuCommands)
                                                    (SET-MENU IL:LogicMiddleMenu IL:ITEMWIDTH 60)
                                                    (SET-MENU IL:LogicMiddleMenu IL:ITEMHEIGHT 14)
                                                                IL:LogicMiddleMenu))))
          (Dribble [LET ((PW (IL:GETPROMPTWINDOW WINDOW))
                       (STREAM NIL)
                       (FILENAME NIL))
                    (IL:CLEARW PW)
                    (SETF FILENAME (IL:PROMPTFORWARD "Typescript to file: " NIL NIL PW))
                    (IL:CLEARW PW)
                    (COND
                     ((NULL FILENAME)
                      (CLOSE (IL:GETWINDOWPROP WINDOW 'IL:TYPESCRIPTSTREAM))
                      (IL:CLEARW PW)
                      (IL:PUTWINDOWPROP WINDOW 'IL:TYPESCRIPTSTREAM NIL)
                      (FORMAT PW "File closed"))
                     (T (SETF STREAM (OPEN (MERGE-PATHNAMES (MAKE-PATHNAME :NAME FILENAME :TYPE
                                                                    'TRC))
                                                                    :DIRECTION :OUTPUT :IF-EXISTS :NEW-VERSION :IF-DOES-NOT-EXIST
                                                                    :CREATE))
                          (IL:CLEARW PW)
                          (FORMAT PW "~S opened" (NAMESTRING STREAM))
                          (IL:PUTWINDOWPROP WINDOW 'IL:TYPESCRIPTSTREAM STREAM)))))]
```

```
(DEFUN LOGIC-DEVELOPER ()
  (IN-PACKAGE 'USER)
  (LET* ((LOGIC-WINDOW (IL:CREATEW NIL (FORMAT NIL "Logic ~D -- Horn clauses programming environment"
                                             *LOGIC-RELEASE-NUMBER*)
                                             6 T))
         ; the main window
         (UNIFY-WINDOW (IL:CREATEW '(10 10 400 400)
                                   "Logic unifier Trace Window" 4 T))
         (SOLVE-WINDOW (IL:CREATEW '(410 10 400 400)
                                   "Logic solver Trace Window" 4 T))
         (REGION (IL:GETWINDOWPROP LOGIC-WINDOW 'IL:REGION))
         (THEORIES-WINDOW (IL:CREATEW (IL:CREATEREGION (- (FIRST REGION)
                                                         120)
                                                         (SECOND REGION)
                                                         120)
                                     (FOURTH REGION))
                       "Theories window" 4 T))
         [TABLE-BROWSER (IL:TB.MAKE.BROWSER NIL THEORIES-WINDOW ' (IL:FONT (HELVETICA 12 BRR))
```

IL:PRINTFN PRINT-TB-ITEMS]

```

(LOGIC-CONTROL-MENU (IL:CREATE IL:MENU))
(PROC (IL:THIS.PROCESS))
LOGIC-CONTROL-MENU-WINDOW)
(DECLARE (SPECIAL LOGIC-WINDOW))
(IL:CLEARW THEORIES-WINDOW)
(IL:DSPSCROLL 'IL:ON UNIFY-WINDOW)
(IL:DSPSCROLL 'IL:ON SOLVE-WINDOW)
(IL:DSPSCROLL 'IL:ON THEORIES-WINDOW)
(IL:PUTWINDOWPROP UNIFY-WINDOW 'IL:BUTTONEVENTFN 'LOGIC-BUTTONFN)
(IL:PUTWINDOWPROP SOLVE-WINDOW 'IL:BUTTONEVENTFN 'LOGIC-BUTTONFN)
(IL:PUTWINDOWPROP LOGIC-WINDOW 'IL:MODE 'IL:FIRST)
(IL:PUTWINDOWPROP LOGIC-WINDOW 'IL:TRUTH-VALUE-ONLY T)
(IL:PUTWINDOWPROP LOGIC-WINDOW 'IL:WINDOWENTRYFN 'IL:GIVE.TTY.PROCESS)
(IL:PUTWINDOWPROP LOGIC-WINDOW 'IL:UNIFY-WINDOW UNIFY-WINDOW)
(IL:PUTWINDOWPROP LOGIC-WINDOW 'IL:SOLVE-WINDOW SOLVE-WINDOW)
(IL:PUTWINDOWPROP LOGIC-WINDOW 'IL:TABLEBROWSER TABLE-BROWSER)
(IL:PUTWINDOWPROP LOGIC-WINDOW 'IL:SOLVE 'NOTRACE)
(IL:PUTWINDOWPROP LOGIC-WINDOW 'IL:UNIFY 'NOTRACE)
(SET-MENU LOGIC-CONTROL-MENU IL:TITLE "Control menu")
(SET-MENU LOGIC-CONTROL-MENU IL:MENUCOLUMNS 1)
(SET-MENU LOGIC-CONTROL-MENU IL:ITEMS *LOGIC-MENU-ITEMS*)
(SET-MENU LOGIC-CONTROL-MENU IL:WHENSELECTEDFN #'LOGIC-MENU-FUNCTION)
(SET-MENU LOGIC-CONTROL-MENU IL:CENTERFLG T)
(SET-MENU LOGIC-CONTROL-MENU IL:ITEMWIDTH 105)
(SET-MENU LOGIC-CONTROL-MENU IL:ITEMHEIGHT 14)
(IL:ATTACHWINDOW (IL:MENUWINDOW LOGIC-CONTROL-MENU)
  LOGIC-WINDOW
  'IL:RIGHT
  'IL:TOP)
(IL:ATTACHWINDOW THEORIES-WINDOW LOGIC-WINDOW 'IL:LEFT 'IL:TOP)
(IL:OPENW LOGIC-WINDOW)
(IL:TTYDISPLAYSTREAM LOGIC-WINDOW)
(IL:USEREXEC " Logic>" NIL #'START-PROVING)
(IL:DEL.PROCESS PROC)
(IL:CLOSEW LOGIC-WINDOW)
T))

```

(DEFUN LOGIC-MENU-FUNCTION (ITEM MENU BUTTON)

```

[LET [(ACTION (SECOND ITEM))
      (MAINW (IL:MAINWINDOW (IL:WFROMMENU MENU))
      (CASE ACTION
        (EXIT
          (MAPCAR #'IL:CLOSEW (IL:ATTACHEDWINDOWS MAINW))
          (IL:DEL.PROCESS (IL:GETWINDOWPROP MAINW 'IL:PROCESS))
          [AND (STREAMP (IL:GETWINDOWPROP (IL:GETWINDOWPROP MAINW 'IL:SOLVE-WINDOW)
            'IL:TYPESCRIPTSTREAM))
              (CLOSE (IL:GETWINDOWPROP (IL:GETWINDOWPROP MAINW 'IL:SOLVE-WINDOW)
                'IL:TYPESCRIPTSTREAM)
              [AND (STREAMP (IL:GETWINDOWPROP (IL:GETWINDOWPROP MAINW 'IL:UNIFY-WINDOW)
                'IL:TYPESCRIPTSTREAM))
              (CLOSE (IL:GETWINDOWPROP (IL:GETWINDOWPROP MAINW 'IL:UNIFY-WINDOW)
                'IL:TYPESCRIPTSTREAM)
          (IL:CLOSEW MAINW))
          (TRUTH-VALUE (IL:PUTWINDOWPROP MAINW 'IL:TRUTH-VALUE-ONLY T))
          (ALL-VALUES (IL:PUTWINDOWPROP MAINW 'IL:TRUTH-VALUE-ONLY NIL))
          (LOAD-THEORY
            (LOAD-DEVEL-THEORY MAINW)
            (CREATE-THEORY-MENU MAINW))
          (SAVE-THEORY (SAVE-DEVEL-THEORY MAINW))
          (CREATE-THEORY
            (CREATE-DEVEL-THEORY MAINW)
            (CREATE-THEORY-MENU MAINW))
          (MERGE-THEORIES (MERGE-THEORIES-DEVEL MAINW))
          (EDIT-SA [IL:ADD.PROCESS `(EDIT-SA ,MAINW)])
          (EDIT-AXIOM [IL:ADD.PROCESS `(EDIT-AXIOM ,MAINW)])
          (NO-SHOW-ENV (IL:PUTWINDOWPROP MAINW 'IL:SEE NIL))
          (DELETE-AXIOM [LET [(CHOSEN-THEORY-NAME (IL:MENU (IL:GETWINDOWPROP MAINW 'IL:THEORIES-MENU)
            (AND CHOSEN-THEORY-NAME
              (LET [(CHOSEN-AXIOM (PROG ((MENU (IL:CREATE IL:MENU)))
                (SET-MENU MENU IL:TITLE "Which axiom?")
                (SET-MENU MENU IL:ITEMS
                  (IL:SORT (ALL-PREDS (GET-THEORY
                    CHOSEN-THEORY-NAME
                    MAINW))
                    #'IL:ALPHORDER))
                (RETURN (IL:MENU MENU)
                  (AND CHOSEN-AXIOM (LOGIC-DELETE CHOSEN-AXIOM CHOSEN-THEORY-NAME
                    MAINW))
                (DELETE-SA [LET [(CHOSEN-THEORY-NAME (IL:MENU (IL:GETWINDOWPROP MAINW 'IL:THEORIES-MENU)
                  (AND CHOSEN-THEORY-NAME
                    (LET [(CHOSEN-SA (PROG ((MENU (IL:CREATE IL:MENU)))
                      (SET-MENU MENU IL:TITLE "Which SA?")
                      (SET-MENU MENU IL:ITEMS (IL:SORT (ALL-SAS (GET-THEORY
                        CHOSEN-THEORY-NAME

```

```

(MAINW))
      #' IL:ALPHORDER))
      (RETURN (IL:MENU MENU)
              (AND CHOOSEN-SA (LOGIC-DELETE CHOOSEN-SA CHOOSEN-THEORY-NAME MAINW))
              (SHOW-AXIOM [LET [(CHOOSEN-THEORY-NAME (IL:MENU (IL:GETWINDOWPROP MAINW 'IL:THEORIES-MENU)
              (AND CHOOSEN-THEORY-NAME (PROG [(MENU (IL:CREATE IL:MENU))
              CHOOSEN-AXIOM
              (ALL-ITEMS (IL:SORT (ALL-PREDS (GET-THEORY
              CHOOSEN-THEORY-NAME
              MAINW]
              (SET-MENU MENU IL:TITLE "Which axiom?")
              (SET-MENU MENU IL:ITEMS ALL-ITEMS)
              JUMP
              (AND (NULL ALL-ITEMS)
              (RETURN))
              (SETF CHOOSEN-AXIOM (IL:MENU MENU))
              (AND CHOOSEN-AXIOM (PROGN (SHOW-DEFINITION
              CHOOSEN-AXIOM
              CHOOSEN-THEORY-NAME
              MAINW)
              (GO JUMP]))
              (SHOW-SA [LET [(CHOOSEN-THEORY-NAME (IL:MENU (IL:GETWINDOWPROP MAINW 'IL:THEORIES-MENU)
              (AND CHOOSEN-THEORY-NAME (PROG [(MENU (IL:CREATE IL:MENU))
              CHOOSEN-SA
              (ALL-ITEMS (IL:SORT (ALL-SAS (GET-THEORY
              CHOOSEN-THEORY-NAME
              MAINW))
              #' IL:ALPHORDER]
              (SET-MENU MENU IL:TITLE "Which SA?")
              (SET-MENU MENU IL:ITEMS ALL-ITEMS)
              JUMP
              (AND (NULL ALL-ITEMS)
              (RETURN))
              (SETF CHOOSEN-SA (IL:MENU MENU))
              (AND CHOOSEN-SA (PROGN (SHOW-DEFINITION CHOOSEN-SA
              CHOOSEN-THEORY-NAME MAINW
              )
              (GO JUMP]))
              (FIRST (IL:PUTWINDOWPROP MAINW 'IL:MODE 'IL:FIRST))
              (SET-MODE (IL:PUTWINDOWPROP MAINW 'IL:MODE 'IL:FIRST))
              (ALL (IL:PUTWINDOWPROP MAINW 'IL:MODE 'IL:ALL))
              (INTERACTIVE (IL:PUTWINDOWPROP MAINW 'IL:MODE 'IL:INTERACTIVE))
              (TRACE-UNIFIER (IL:PUTWINDOWPROP MAINW 'IL:UNIFY 'TRACE))
              (TRACE-SOLVER (IL:PUTWINDOWPROP MAINW 'IL:SOLVE 'TRACE))
              (NOTRACE-SOLVER [PROGN (IL:PUTWINDOWPROP MAINW 'IL:SOLVE 'NOTRACE)
              (AND (STREAMP (IL:GETWINDOWPROP (IL:GETWINDOWPROP MAINW 'IL:SOLVE-WINDOW)
              'IL:TYPESCRIPTSTREAM))
              (CLOSE (IL:GETWINDOWPROP (IL:GETWINDOWPROP MAINW 'IL:SOLVE-WINDOW)
              'IL:TYPESCRIPTSTREAM))
              (NOTRACE-UNIFIER [PROGN (IL:PUTWINDOWPROP MAINW 'IL:UNIFY 'NOTRACE)
              (AND (STREAMP (IL:GETWINDOWPROP (IL:GETWINDOWPROP MAINW 'IL:UNIFY-WINDOW)
              'IL:TYPESCRIPTSTREAM))
              (CLOSE (IL:GETWINDOWPROP (IL:GETWINDOWPROP MAINW 'IL:UNIFY-WINDOW)
              'IL:TYPESCRIPTSTREAM))
              (DELETE-THEORY [LET [(TB (IL:GETWINDOWPROP MAINW 'IL:TABLEBROWSER)
              (DO ((ITEMS (IL:TB.COLLECT.ITEMS TB 'IL:SELECTED)
              (CDR ITEMS)))
              ((NULL ITEMS))
              ((IL:TB.DELETE.ITEM TB (CAR ITEMS))))]
              (UNDELETE [LET [(TB (IL:GETWINDOWPROP MAINW 'IL:TABLEBROWSER)
              (DO ((ITEMS (IL:TB.COLLECT.ITEMS TB 'IL:SELECTED)
              (CDR ITEMS)))
              ((NULL ITEMS))
              ((IL:TB.UNDELETE.ITEM TB (CAR ITEMS))))]
              (EXPUNGE [LET [(TB (IL:GETWINDOWPROP MAINW 'IL:TABLEBROWSER)
              (DO ((ITEMS (IL:TB.COLLECT.ITEMS TB 'IL:DELETED)
              (CDR ITEMS)))
              ((NULL ITEMS))
              ((IL:TB.REMOVE.ITEM TB (CAR ITEMS)))
              (CREATE-THEORY-MENU MAINW))]
              (ERASE [LET [(TB (IL:GETWINDOWPROP MAINW 'IL:TABLEBROWSER)
              (DO ((ITEMS (IL:TB.COLLECT.ITEMS TB)
              (CDR ITEMS)))
              ((NULL ITEMS))
              ((IL:TB.REMOVE.ITEM TB (CAR ITEMS)))
              (CREATE-THEORY-MENU MAINW))]
              (SHOW-PROFILE (SHOW-PROFILE MAINW))]]]

```

```

(DEFUN MERGE-THEORIES-DEVEL (MAINW &OPTIONAL NEW-THEORY LIST-OF-THEORIES)
  [LET* [(PW (IL:GETPROMPTWINDOW MAINW)
          (THEORY-NAME (OR NEW-THEORY (PROGN (IL:CLEARW PW)
          (PROMPTREAD "New theory name" PW T)
          (AND THEORY-NAME (LET* [(ACTUAL-THEORY (MAKE-HASH-TABLE)
          [TB-ITEM (MY-CREATE-TBRECORD (ACONS 'THEORY ACTUAL-THEORY (ACONS
          'THEORY-NAME
          THEORY-NAME

```



```

(FORMAT FILE "~D~%" (LENGTH SAS))
(DO ((SA-NAME SAS (CDR SA-NAME)))
  ((NULL SA-NAME)
   NIL)
 (PROGN (FORMAT PW ".")
        (FORMAT FILE "~S ~S ~%" (CAR SA-NAME)
                          (GETHASH (CAR SA-NAME)
                                   THEORY))))
(FORMAT FILE "~D~%" (LENGTH PRED))
(DO ((PRED-NAME PRED (CDR PRED-NAME)))
  ((NULL PRED-NAME)
   NIL)
 (PROGN (FORMAT PW ".")
        (FORMAT FILE "~S ~S ~%" (CAR PRED-NAME)
                          (GETHASH (CAR PRED-NAME)
                                   THEORY))))))

```

```

(DEFUN SHOW-PROFILE (WINDOW)
  [LET ((PW (IL:GETPROMPTWINDOW WINDOW))
        (IL:CLEARW PW)
        (FORMAT PW "~%Mode: ~A /Unifier: ~A /Solver: ~A /Values: ~A" (IL:GETWINDOWPROP WINDOW 'IL:MODE)
                (IL:GETWINDOWPROP WINDOW 'IL:UNIFY)
                (IL:GETWINDOWPROP WINDOW 'IL:SOLVE)
                (NOT (IL:GETWINDOWPROP WINDOW 'IL:TRUTH-VALUE-ONLY)))

```

```

(DEFUN SOLVE-DEBUGGER (TREE FORMULA CLAUSES WINDOW)
  [COND
   ((TRACINGP WINDOW 'SOLVE)
    ;; This is the part for debugging: the main features of the language are shown to the user in specified windows
    (FORMAT (IL:GETWINDOWPROP WINDOW 'IL:SOLVE-WINDOW)
            "Formula is ~A,~%%clauses are ~A,~%%conjs are ~A~%%~%" FORMULA CLAUSES (CONJ (AND-LEVEL TREE)))
    (AND (DRIBBLEP WINDOW 'SOLVE)
         (FORMAT (IL:GETWINDOWPROP (IL:GETWINDOWPROP WINDOW 'IL:SOLVE-WINDOW)
                                   'IL:TYPESCRIPTSTREAM)
                 "Formula is ~A,~%%clauses are ~A,~%%conjs are ~A~%%~%" FORMULA CLAUSES (CONJ (AND-LEVEL TREE))

```

```

(DEFUN START-PROVING (CONJS LINE)
  (IN-PACKAGE 'USER)
  [LET* ((*VARIABLES-COUNTER* 0)
         (SELECTED-TIS (IL:TB.COLLECT.ITEMS (IL:GETWINDOWPROP LOGIC-WINDOW 'IL:TABLEBROWSER)
                                             'IL:SELECTED))
         [THEORIES (APPEND (LIST '*BACKGROUND-THEORY*)
                           (GET-LIST-PROP SELECTED-TIS 'THEORY-NAME)]
         (TREE (MAKE-TREE (MAKE-AND-NODE CONJS NIL THEORIES)
                          NIL))
         (TRUTH-VALUE-ONLY (IL:GETWINDOWPROP LOGIC-WINDOW 'IL:TRUTH-VALUE-ONLY))
         RESULT NEXT-OR)
    (DECLARE (SPECIAL *VARIABLES-COUNTER*))
    (PROG NIL
     JUMP
     (SETF RESULT (LOGIC-PROVE TREE LOGIC-WINDOW))
     (COND
      ((NULL RESULT)
       ; The proof is failed
      )
     (T (CASE (IL:GETWINDOWPROP LOGIC-WINDOW 'IL:MODE)
              ((IL:FIRST [OR (AND TRUTH-VALUE-ONLY (PROGN (FORMAT T "~A~%" T)
                                                            T))
                           (FORMAT T "~S~%" (LOOKUP CONJS (UNIFICATION-ENV (AND-LEVEL RESULT))
                                                            T))
              ((IL:ALL
                (OR (AND TRUTH-VALUE-ONLY (PROGN (FORMAT T "~A~%" T)
                                                            T))
                    (PROGN [FORMAT T "~S~%" (LOOKUP CONJS (UNIFICATION-ENV (AND-LEVEL RESULT))
                                                            T))
                    (SETF NEXT-OR (FIRST (OR-LEVELS RESULT)))
                    (SETF TREE (SOLVE (NEW-TREE RESULT NEXT-OR)
                                      (FORMULA-OR NEXT-OR)
                                      (CLAUSES-OR NEXT-OR)))
                    (GO JUMP)))
              (IL:INTERACTIVE
                (OR (AND TRUTH-VALUE-ONLY (PROGN (FORMAT T "~A~%" T)
                                                            T))
                    (PROGN [FORMAT T "~S~%" (LOOKUP CONJS (UNIFICATION-ENV (AND-LEVEL RESULT))
                                                            T))
                    (FORMAT T "More? ")
                    (AND (Y-OR-N-P)
                        (PROGN (SETF NEXT-OR (FIRST (OR-LEVELS RESULT)))
                              (SETF TREE (SOLVE (NEW-TREE RESULT NEXT-OR)
                                                (FORMULA-OR NEXT-OR)
                                                (CLAUSES-OR NEXT-OR)))
                    (GO JUMP))))))

```



(DEFUN **UNIFY-DEBUGGER** (PATT DAT ENV WINDOW)

;; This part is devoted to debugging, on the window and on the file

(LET\* [(TRACE-WINDOW (IL:GETWINDOWPROP WINDOW 'IL:UNIFY-WINDOW))
(DRIBBLE? (DRIBBLEP WINDOW 'UNIFY))
(STREAM (AND DRIBBLE? (IL:GETWINDOWPROP TRACE-WINDOW 'IL:TYPESCRIPTSTREAM)
(FORMAT TRACE-WINDOW "~%%Unifying ~A ~%%with ~A~%%in ~A~%%" PATT DAT ENV)
(AND DRIBBLE? (FORMAT STREAM "~%%Unifying ~A ~%%with ~A~%%in ~A~%% " PATT DAT ENV)))]

(IL:ADDTOVAR **IL:BackgroundMenuCommands** ("Logic" '(IL:ADD.PROCESS '(LOGIC-DEVELOPER)
"Open a window on logic programming environment"))

(IL:RPAQQ **\*LOGIC-MENU-ITEMS\***

(("Show profile" SHOW-PROFILE "Show the profile on env")
("Truth value only" TRUTH-VALUE "The proof returns only T or NIL" (IL:SUBITEMS ("All values "
ALL-VALUES
>Returns the
goal with all
the variables")
))
("Show(Axiom)" SHOW-AXIOM "Shows definition of an axiom" (IL:SUBITEMS ("Show SA" SHOW-SA "Shows
definition of a semantic
attachment")))
("Edit(Axiom)" EDIT-AXIOM "Edits the specified axiom" (IL:SUBITEMS ("Edit SA" EDIT-SA "Edits the
specified SA")))
("Delete(Axiom)" DELETE-AXIOM "Deletes the specified axiom" (IL:SUBITEMS ("Delete SA" DELETE-SA
"Deletes the
specified semantic
attachment")))
("Set Mode(First)" SET-MODE "Set mode of demonstration" (IL:SUBITEMS ("First" FIRST "Stops at first
solution reached")
("All" ALL "Finds out all solutions")
("Interactive" INTERACTIVE "Ask user
to continue")))
("Trace unifier" TRACE-UNIFIER "Trace the unifier" (IL:SUBITEMS ("No trace" NOTRACE-UNIFIER "Do not
trace unifier")))
("Trace solver" TRACE-SOLVER "Trace the solver" (IL:SUBITEMS ("No trace" NOTRACE-SOLVER "Do not trace
solver")))
("Create theory" CREATE-THEORY "Creates new theory")
("Delete theory" DELETE-THEORY "Deletes the labelled theories" (IL:SUBITEMS ("Expunge deleted
theories" EXPUNGE
"Expunged deleted
theories")
("Undelete theories" UNDELETE
"Undelete theories")))
("Merge theories" MERGE-THEORIES "Merges the selected theories")
("Load theory" LOAD-THEORY "Prompts user for theory to load")
("Save theory" SAVE-THEORY "Saves selected theories")
("Erase env" ERASE "Erases all the environment")
("Exit" EXIT "Closes development window")))

(IL:RPAQQ **\*LOGIC-RELEASE-NUMBER\*** "1.3")

(IL:RPAQQ **\*LOGIC-CLOSE-ON-COMPLETION-FLG\*** T)

(IL:RPAQQ **IL:BackgroundMenu** NIL)

(IL:RPAQQ **IL:LogicMiddleMenu** NIL)

(IL:RPAQQ **IL:LogicMiddleMenuCommands** ((DRIBBLE 'Dribble "Dribbles on file")))

(IL:FILESLOAD IL:TABLEBROWSER)

(IL:DECLARE%: IL:EVAL@COMPILE

(IL:DATATYPE IL:TABLEBROWSER ((IL:TBREADY IL:FLAG)
(NIL 7 IL:FLAG)
(IL:TBITEMS IL:POINTER)
(IL:TB#ITEMS IL:WORD)
(IL:TB#DELETED IL:WORD)
(IL:TB#LINESPERITEM IL:WORD)
(IL:TBFIRSTSELECTEDITEM IL:WORD)
(IL:TB#LASTSELECTEDITEM IL:WORD)
(NIL IL:WORD)
(IL:TBMXXPOS IL:WORD)
(IL:TBFONTHEIGHT IL:WORD)
(IL:TBFONTASCENT IL:WORD)
(IL:TBFONTDESCENT IL:WORD)
(IL:TBWINDOW IL:POINTER)
(IL:TBLOCK IL:POINTER)
(IL:TBUSERDATA IL:POINTER)
(IL:TBFONT IL:POINTER)
(IL:TBEXTENT IL:POINTER)
(IL:TBUPDATEFROMHERE IL:POINTER)
(IL:TBCOLUMNS IL:POINTER)

```

(IL:TBPRINTFN IL:POINTER)
(IL:TBCOPYFN IL:POINTER)
(IL:TBFONTCHANGEFN IL:POINTER)
(IL:TBCLOSEFN IL:POINTER)
(IL:TBAFTERCLOSEFN IL:POINTER)
(IL:TBTITLEEVENTFN IL:POINTER)
(IL:TBAFTEREXPUNGEFN IL:POINTER)
(IL:TBORIGIN IL:POINTER)
(NIL IL:POINTER)
(NIL IL:POINTER)
(NIL IL:POINTER))

```

```

(IL:DATATYPE IL:TABLEITEM ((IL:TISELECTED IL:FLAG)
(IL:TIDELETED IL:FLAG)
(IL:TIUNDELETABLE IL:FLAG)
(IL:TIUNSELECTABLE IL:FLAG)
(IL:TIUNCOPYSELECTABLE IL:FLAG)
(NIL 3 IL:FLAG)
(IL:TIDATA IL:POINTER)
(IL:TI# IL:WORD)
(NIL IL:WORD)))
)

```

```

(IL:/DECLAREDATATYPE 'IL:TABLEBROWSER
' (IL:FLAG IL:FLAG IL:FLAG IL:FLAG IL:FLAG IL:FLAG IL:FLAG IL:FLAG IL:POINTER IL:WORD IL:WORD IL:WORD
IL:WORD IL:WORD IL:WORD IL:WORD IL:WORD IL:WORD IL:WORD IL:POINTER IL:POINTER IL:POINTER
IL:POINTER IL:POINTER IL:POINTER IL:POINTER IL:POINTER IL:POINTER IL:POINTER IL:POINTER
IL:POINTER IL:POINTER IL:POINTER IL:POINTER IL:POINTER IL:POINTER IL:POINTER)
;; ---field descriptor list elided by lister---
' 48)

```

```

(IL:/DECLAREDATATYPE 'IL:TABLEITEM '(IL:FLAG IL:FLAG IL:FLAG IL:FLAG IL:FLAG IL:FLAG IL:FLAG IL:FLAG IL:POINTER
IL:WORD IL:WORD)
;; ---field descriptor list elided by lister---
' 4)

```

(IL:DECLARE%: IL:EVAL@COMPILE

(IL:RPAQQ IL:TB.LEFT.MARGIN 8)

(IL:CONSTANTS IL:TB.LEFT.MARGIN)

)
(IL:PUTPROPS IL:LOGIC-DEVEL IL:COPYRIGHT ("ROBERTO GHISLANZONI" 1987 1988))

---

**FUNCTION INDEX**

CREATE-DEVEL-THEORY .....	1	LIST-ALL-THEORIES-INTERNAL .....	3	PRINT-TB-ITEMS .....	7
CREATE-THEORY-MENU .....	2	LOAD-DEVEL-THEORY .....	3	PROMPTREAD .....	7
EDIT-AXIOM .....	2	LOGIC-BUTTONFN .....	4	SAVE-DEVEL-THEORY .....	7
EDIT-SA .....	3	LOGIC-DEVELOPER .....	4	SHOW-PROFILE .....	8
GET-LIST-PROP .....	3	LOGIC-MENU-FUNCTION .....	5	SOLVE-DEBUGGER .....	8
GET-TB-PROPERTY .....	3	MERGE-THEORIES-DEVEL .....	6	START-PROVING .....	8
GET-THEORY-INTERNAL .....	3	MY-CREATE-TBRECORD .....	7	UNIFY-DEBUGGER .....	9

---

**VARIABLE INDEX**

*LOGIC-CLOSE-ON-COMPLETION-FLG* ..	9	IL:BackgroundMenu .....	9	IL:LogicMiddleMenuCommands .....	9
*LOGIC-MENU-ITEMS* .....	9	IL:BackgroundMenuCommands .....	9		
*LOGIC-RELEASE-NUMBER* .....	9	IL:LogicMiddleMenu .....	9		

---

**MACRO INDEX**

DRIBBLEP .....	1	SEE-ENV-P .....	1	SET-MENU .....	1	TRACINGP .....	1
----------------	---	-----------------	---	----------------	---	----------------	---

---

**RECORD INDEX**

IL:TABLEBROWSER .....	9	IL:TABLEITEM .....	10
-----------------------	---	--------------------	----

---

**CONSTANT INDEX**

IL:TB.LEFT.MARGIN .....	10
-------------------------	----

---