File created: 23-Mar-87 13:37:16 **{DSK}<LISPFILES2>H.BETA>HSOURCE.;6**

previous date: 18-Mar-87 14:22:10 {DSK}<LISPFILES2>H.BETA>HSOURCE.;5

Read Table: OLD-INTERLISP-FILE

Package: INTERLISP

Format: XCCS

```
(RPAQQ HSOURCECOMS
       [(VARS *functions* *int-mode* *l-trace* *limit* *match-trace* *predicates* *primitiveSA* *shortform*
              *variables* H.FROM.WINDOW)
        (FNS BuildAssert FirstAntec H.? H.?! H.?v H.RmAxiom H.absent H.addaxiom H.all H.another? H.any H.attach
             H.axioms H.continue? H.del H.expand1 H.index H.isfunc? H.match H.prove H.show H.strip H.the IsCar?
             IsPrimitive? MacroAxiom MapNull MkVars NewAxiom NotCorrect PredicateOf RmAxiom SET.H.MODE Write
             fact? with?)
        (MACROS H.strip)
        (PROP axiom /)
        (PROP prim-funct assert delete set)
        (DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA Write H.show H.attach H.?)
                                                                              (NLAML)
                                                                              (LAMA])

(RPAQQ *functions* NIL)

(RPAQQ *int-mode* first)

(RPAQQ *l-trace* NIL)

(RPAQQ *limit* 200)

(RPAQQ *match-trace* NIL)

(RPAQQ *predicates* NIL)

(RPAQQ *primitiveSA* (set assert delete))

(RPAQQ *shortform* T)

(RPAQQ *variables* NIL)

(RPAQQ H.FROM.WINDOW NIL)

(DEFINEQ
```

(**BuildAssert**

```
  [LAMBDA (x)                                                    (* edited: " 2-Oct-86 12:08")
    (SETQ assert-numb (PLUS 1 assert-numb))
    (MacroAxiom (MkVars x])
```

(**FirstAntec**

```
  [LAMBDA (p)                                                    (* edited: " 2-Oct-86 13:09")
    (CADDR p])
```

(**H.?**

```
  [NLAMBDA conj.to.prove                                         (* edited: " 5-Feb-87 11:15")
    (LET ((*lisp-use* T)
          (lisp-channel NIL)
          (formulas conj.to.prove))
         (MkVars conj.to.prove)
         (H.?v conj.to.prove)
         lisp-channel])
```

(**H.?!**

```
  [LAMBDA (conj.to.prove)                                        (* edited: " 5-Feb-87 17:28")
    (MkVars conj.to.prove HWINDOW)                               (* Used by the H window)
    (LET ((lisp-channel NIL)
          (*lisp-use* NIL))
         (H.?v conj.to.prove])
```

(**H.?v**

```
  [LAMBDA (formulas)                                             (* edited: " 3-Feb-87 11:16")
    (CATCH (QUOTE first-level)
           (LET ((counter (PLUS 1 *limit*))
                 (globalslashflag NIL)
                 (IsCar NIL))
                (MkVars formulas)
                (H.prove (LIST (LIST formulas NIL))
                       1])
```

(**H.RmAxiom**

```
  [LAMBDA (ax)                                                   (* edited: "10-Feb-87 15:16")
```

```
    (SETQ *predicates* (DREMOVE ax *predicates*))
    (PUTPROP ax (QUOTE axiom)
           NIL])
```

(H.absent

```
  [LAMBDA (x xlst y ylst)                                   (* edited: " 4-Feb-87 14:20")
    (PROG NIL
      a   (COND
             [(ATOM y)
              (COND
                 ([OR (NUMBERP y)
                      (AND (MEMBER y (\SEE.AT *variables*))
                           (NOT (GETPROP y (QUOTE variable]
                  (RETURN T))
                 ((SETQ temp (ASSOC y (CAR ylst)))
                  (SETQ y (CADR temp))
                  (SETQ ylst (CDDR temp))
                  (GO a))
                 ((RETURN (NOT (AND (EQ x y)
                                    (EQ xlst ylst]
              ((H.absent x xlst (CAR y)
                         ylst)
               (SETQ y (CDR y))
               (GO a])
```

(H.addaxiom

```
  [LAMBDA (ax-list)                                         (* edited: " 7-Oct-86 09:48")
    (CATCH (QUOTE first-level)
           (LET ((assert-numb 0))
                (MapNull ax-list (QUOTE BuildAssert])
```

(H.all

```
  [LAMBDA (formulas cong)                                   (* edited: " 5-Feb-87 11:17")
    (MkVars cong)
    (LET ((globalslashflag NIL)
          (*int-mode* (QUOTE all))
          (*lisp-use* T)
          (counter (PLUS 1 *limit*))
          (lisp-channel NIL))
         (H.prove (BQUOTE (((\, cong)
                            NIL)))
                  1)
         lisp-channel])
```

(H.another?

```
  [LAMBDA NIL                                               (* edited: " 4-Feb-87 16:17")
    (COND
       ((EQ (\SEE.AT *int-mode*)
            (QUOTE all))
        T)
       ((OR (EQ (\SEE.AT *int-mode*)
                (QUOTE first))
            (ZEROP (\SEE.AT *int-mode*)))
        NIL)
       ((NUMBERP (\SEE.AT *int-mode*))
        (if H.FROM.WINDOW
            then (PUTWINDOWPROP HWINDOW (QUOTE *int-mode*)
                                (SUB1 (\SEE.AT *int-mode*)))
          else (SETQ *int-mode* (SUB1 *int-mode*)))
        T)
       (T (H.continue?])
```

(H.any

```
  [LAMBDA (quant formulas cong)                             (* edited: " 5-Feb-87 12:12")
    (MkVars cong)
    (CATCH (QUOTE first-level)
           (LET ((globalslashflag NIL)
                 (*int-mode* quant)
                 (*lisp-use* T)
                 (IsCar T)
                 (counter (PLUS 1 *limit*))
                 (lisp-channel NIL))
                (H.prove (BQUOTE (((\, cong)
                                   NIL)))
                         1)
                lisp-channel])
```

(H.attach

```
  [NLAMBDA flist                                            (* edited: " 3-Oct-86 11:56")
    [COND
       ((NOT (MEMBER (CAR flist)
```

```
                    *functions*))
        (SETQ *functions* (CONS (CAR flist)
                                *functions*]
    (PUTPROP (CAR flist)
           (QUOTE funct)
           (CADR flist])
```

(H.axioms

```
  [LAMBDA (ax-list)                                          (* edited: " 7-Oct-86 09:49")
    (CATCH (QUOTE first-level)
           (LET ((assert-numb 0)
                 (IsCar NIL))
                (MapNull ax-list (QUOTE RmAxiom))
                (MapNull ax-list (QUOTE BuildAssert])
```

(H.continue?

```
  [LAMBDA NIL                                                (* edited: " 7-Oct-86 14:11")
    (TERPRI)
    (MEMBER (CAR (TTYIN (QUOTE Another?)))
           (QUOTE (y yes ok Y YES S SI si s])
```

(H.del

```
  [LAMBDA (arg)                                              (* edited: " 5-Feb-87 13:29")
    (PUTPROP (CAR arg)
           (QUOTE axiom)
           (H.delete arg (GETPROP (CAR arg)
                                  (QUOTE axiom])
```

(H.expand1

```
  [LAMBDA (lst x)                                            (* edited: " 4-Feb-87 14:21")
    (COND
       ((STRINGP x)
        x)
       [(NOT (ATOM x))
        (CONS (H.expand1 lst (CAR x))
              (H.expand1 lst (CDR x]
       ((NUMBERP x)
        x)
       ((AND (MEMBER x (\SEE.AT *variables*))
             (GETPROP x (QUOTE variable)))
        (COND
           ((SETQ temp (ASSOC x (CAR lst)))
            (H.expand1 (CDDR temp)
                   (CADR temp)))
           (T x)))
       (T x])
```

(H.index

```
  [LAMBDA (z zlst)                                           (* edited: " 3-Oct-86 09:22")
    (PROG NIL
      a   (COND
             ((NOT (ATOM z))
              (SETQ z (CAR z))
              (GO a))
             ((SETQ temp (ASSOC z (CAR zlst)))
              (SETQ z (CADR temp))
              (SETQ zlst (CDDR temp))
              (GO a))
             (z (RETURN z])
```

(H.isfunc?

```
  [LAMBDA (arg)                                              (* edited: " 6-Feb-87 09:43")
    (COND
       ((ATOM arg)
        NIL)
       ((IsPrimitive? (CAR arg))
        (QUOTE primitive))
       (T (AND (MEMBER (CAR arg)
                       (\SEE.AT *functions*))
               (GETPROP (CAR arg)
                       (QUOTE funct])
```

(H.match

```
  [LAMBDA (x xlst y ylst)                                    (* edited: "17-Mar-87 14:10")
    (AND (\SEE.AT *match-trace*)
         (H.PrintInfo (QUOTE PM)
                (LIST (QUOTE X=)
                      x
                      (QUOTE Y=)
                      y)))
```

```
    (PROG (temp)
      a   [COND
             ((ATOM x)
              (COND
                 [(NOT (NUMBERP x))
                  (COND
                     [(AND (MEMBER x (\SEE.AT *variables*))
                           (GETPROP x (QUOTE variable)))
                      (COND
                         ((SETQ temp (ASSOC x (CAR xlst)))
                          (SETQ x (CADR temp))
                          (SETQ xlst (CDDR temp))
                          (GO a))
                         (T [PROG NIL
                              b   (COND
                                     ([AND (ATOM y)
                                           (NOT (NUMBERP y))
                                           (AND (MEMBER y (\SEE.AT *variables*))
                                                (GETPROP y (QUOTE variable)))
                                           (SETQ temp (ASSOC y (CAR ylst]
                                      (SETQ y (CADR temp))
                                      (SETQ ylst (CDDR temp))
                                      (GO b]
                                 (COND
                                    ((AND (EQ x y)
                                          (EQ ylst xlst))
                                     (RETURN T))
                                    ((OR (\SEE.AT *shortform*)
                                         (H.absent x xlst y ylst))
                                     (SETQ save (CONS xlst save))
                                     (RPLACA xlst (CONS (CONS x (CONS y ylst))
                                                        (CAR xlst)))
                                     (RETURN T))
                                    (T (RETURN NIL]
                     (T (GO c]
                (T (GO c]
      b   [COND
             ((ATOM y)
              (COND
                 [(NOT (NUMBERP y))
                  (COND
                     [(AND (MEMBER y (\SEE.AT *variables*))
                           (GETPROP y (QUOTE variable)))
                      (COND
                         ((SETQ temp (ASSOC y (CAR ylst)))
                          (SETQ y (CADR temp))
                          (SETQ ylst (CDDR temp))
                          (GO b))
                         (T (COND
                               ((OR (\SEE.AT *shortform*)
                                    (H.absent y ylst x xlst))
                                (SETQ save (CONS ylst save))
                                (RPLACA ylst (CONS (CONS y (CONS x xlst))
                                                   (CAR ylst)))
                                (RETURN T))
                               (T (RETURN NIL]
                     (T (RETURN (EQ x y]
                (T (RETURN (EQ x y]
          (COND
             ((H.match (CAR x)
                       xlst
                       (CAR y)
                       ylst)
              (SETQ x (CDR x))
              (SETQ y (CDR y))
              (GO a))
             (T (RETURN NIL)))
      c   (COND
             ((ATOM y)
              (COND
                 [(NOT (NUMBERP y))
                  (COND
                     [(AND (MEMBER y (\SEE.AT *variables*))
                           (GETPROP y (QUOTE variable)))
                      (COND
                         ((SETQ temp (ASSOC y (CAR ylst)))
                          (SETQ y (CADR temp))
                          (SETQ ylst (CDDR temp))
                          (GO c))
                         (T (COND
                               ((OR (\SEE.AT *shortform*)
                                    (H.absent y ylst x xlst))
                                (SETQ save (CONS ylst save))
                                (RPLACA ylst (CONS (CONS y (CONS x xlst))
                                                   (CAR ylst)))
                                (RETURN T))
                               (T (RETURN NIL]
```

```
                    (T (RETURN (EQ x y]
                (T (RETURN (EQ x y])
```

(H.prove

```
 [LAMBDA (stack level)                                          (* edited: "17-Mar-87 14:11")
   (PROG ((save (LIST))
          val ASS w z f)
         (SETQ counter (SUB1 counter))
     c   [COND
            ((ZEROP counter)
             (LET [(answer (CAR (TTYIN (QUOTE continue?]
                  (COND
                     [(MEMBER answer (QUOTE (y yes ok s si)))
                      (SETQ counter (PLUS 1 (\SEE.AT *limit*]
                     ((MEMBER answer (QUOTE (n no stop)))
                      (THROW (QUOTE first-level)
                             NIL))
                     (T (SHOWPRINT (EVAL answer)
                               (if LISPXUSERFN
                                   then HWINDOW
                                 else T))
                        (GO c]
     b   [COND
            [(NULL (CAAR stack))
             (SETQ level (SUB1 level))
             (COND
                [(NULL (CDR stack))
                 (SETQ val (H.expand1 (CDAR stack)
                                      formulas))
                 [COND
                    (*lisp-use* (SETQ lisp-channel (CONS (H.expand1 (CDAR stack)
                                                                    formulas)
                                                         lisp-channel)))
                    (T (SHOWPRINT val (if LISPXUSERFN
                                          then HWINDOW
                                        else T]
                 (COND
                    ((H.another?)
                     (RETURN NIL))
                    (T (THROW (QUOTE first-level)
                              val]
                ((SETQ stack (CDR stack))
                 (OR [AND (\SEE.AT *l-trace*)
                          (H.PrintInfo (QUOTE TR)
                                 (LIST level (QUOTE <--)
                                       (H.expand1 (CDAR stack)
                                              (CAAAR stack]
                     T)
                 (SETQ stack (CONS (CONS (CDAAR stack)
                                         (CDAR stack))
                                   (CDR stack)))
                 (GO b]
             ([SETQ f (H.isfunc? (H.expand1 (CDAR stack)
                                            (CAAAR stack]
              [COND
                 ((\SEE.AT *l-trace*)
                  (H.PrintInfo (QUOTE TR)
                         (LIST level (QUOTE fun--->)
                               (H.expand1 (CDAR stack)
                                      (CAAAR stack]
              (COND
                 ((APPLY (GETPROP (H.expand1 (CDAR stack)
                                             (CAAAAR stack))
                                  (if (EQ f (QUOTE primitive))
                                      then (QUOTE prim-funct)
                                    else (QUOTE funct)))
                         (H.expand1 (CDAR stack)
                                (CDAAAR stack)))
                  (SETQ stack (CONS (CONS (CDAAR stack)
                                          (CDAR stack))
                                    (CDR stack)))
                  (GO b]
         (SETQ ASS (GETPROP (H.index (CAAAR stack)
                                     (CDAR stack))
                            (QUOTE axiom)))
     a   (COND
            ((ZEROP counter)
             (RETURN))
            ((NULL ASS)
             [COND
                ((EQ (CAAAR stack)
                     (QUOTE /))
                 (OR globalslashflag (SETQ globalslashflag (CDAR stack]
              (RETURN))
            ([AND (H.match (CAAR ASS)
                           (SETQ z (LIST NIL))
```

```
                        (CAAAR stack)
                        (CDAR stack))
                   (OR [AND (\SEE.AT *l-trace*)
                            (H.PrintInfo (QUOTE TR)
                                   (LIST level (QUOTE -->)
                                         (CAAAR stack)
                                         (H.expand1 z (CAR ASS]
                       T)
                   (SETQ w (H.prove (CONS (CONS (H.strip (CDAR ASS))
                                                z)
                                          stack)
                                    (PLUS 1 level]
              (RETURN w))
             (T [MapNull save (FUNCTION (LAMBDA (p)
                                        (RPLACA p (CDAR p]
                (SETQ save NIL)
                [SETQ ASS (COND
                             ((NULL globalslashflag)
                              (CDR ASS))
                             ((EQ globalslashflag z)
                              (SETQ globalslashflag NIL]
                (GO a])
```

(H.show

```
[NLAMBDA args                                                    (* edited: " 4-Feb-87 14:48")
  (for I in args do (PRINTOUT T (AND (MEMBER I (\SEE.AT *predicates*))
                                     (GETPROP I (QUOTE axiom)))
                           T])
```

(H.strip

```
[LAMBDA (x)                                                      (* edited: " 2-Oct-86 12:03")
  (COND
     ((NULL x)
      NIL)
     (T (CDR x])
```

(H.the

```
[LAMBDA (formulas cong)                                          (* edited: " 5-Feb-87 12:13")
  (MkVars cong)
  (CATCH (QUOTE first-level)
         (LET ((globalslashflag NIL)
               (*int-mode* (QUOTE first))
               (*lisp-use* T)
               (IsCar T)
               (counter (PLUS 1 *limit*))
               (lisp-channel NIL))
              (H.prove (BQUOTE (((\, cong)
                                 NIL)))
                     1)
              lisp-channel])
```

(IsCar?

```
[LAMBDA (arg)                                                    (* edited: " 5-Feb-87 11:24")
  (FOO)
  (if IsCar
      then (CAR arg)
    else arg])
```

(IsPrimitive?

```
[LAMBDA (pred)                                                   (* edited: " 6-Feb-87 09:44")
  (MEMBER pred *primitiveSA*])
```

(MacroAxiom

```
[LAMBDA (assert)                                                 (* edited: " 5-Feb-87 11:08")
  (COND
     ((fact? assert)
      (NewAxiom assert))
     ((with? assert)
      (BuildWith assert))
     ((NotCorrect assert)
      (PRINTOUT T " Incorrect definition in " assert-numb)
      (THROW (QUOTE first-level)
             NIL))
     ((EQ (FirstAntec assert)
          (QUOTE if))
      (BuildIf assert))
     ((EQ (FirstAntec assert)
          (QUOTE or))
      (BuildOr assert))
     (T (NewAxiom assert])
```

(MapNull

```
[LAMBDA (lst foo)                                          (* edited: " 2-Oct-86 11:47")
  (COND
     ((NULL lst)
      NIL)
     (T (APPLY foo (LIST (CAR lst)))
        (MapNull (CDR lst)
               foo])
```

(MkVars

```
[LAMBDA (assert win)                                       (* edited: " 6-Feb-87 11:16")
  (COND
     ((NULL assert)
      NIL)
     ((STRINGP assert)
      assert)
     ((NOT (ATOM assert))
      (CONS (MkVars (CAR assert)
                    win)
            (MkVars (CDR assert)
                    win)))
     ((EQ (CAR (UNPACK assert))
          (QUOTE :))
      [COND
         ((NOT (MEMBER assert (if win
                                  then (GETWINDOWPROP win (QUOTE *variables*))
                                else *variables*)))
          (PUTPROP assert (QUOTE variable)
                 T)
          (if win
              then [PUTWINDOWPROP win (QUOTE *variables*)
                                  (CONS assert (GETWINDOWPROP win (QUOTE *variables*]
            else (SETQ *variables* (CONS assert *variables*]
      assert)
     (T assert])
```

(NewAxiom

```
[LAMBDA (assert)                                           (* edited: " 3-Feb-87 18:29")
  (COND
     ((OR (ATOM assert)
          (ATOM (CAR assert)))
      (PRINTOUT T "Error in atomic formula " assert-numb T)
      (THROW (QUOTE first-level)
             NIL))
     (T (LET ((predicate-name (PredicateOf assert)))
             [COND
                ((NOT (MEMBER predicate-name (\SEE.AT *predicates*)))
                 (if H.FROM.WINDOW
                     then (PUTWINDOWPROP HWINDOW (QUOTE *predicates*)
                                         (CONS predicate-name (\SEE.AT *predicates*)))
                   else (SETQ *predicates* (CONS predicate-name *predicates*]
             (PUTPROP predicate-name (QUOTE axiom)
                    (APPEND (GETPROP predicate-name (QUOTE axiom))
                            (LIST assert])
```

(NotCorrect

```
[LAMBDA (as)                                               (* edited: " 2-Oct-86 13:10")
  (NOT (EQ (CADR as)
           (QUOTE <])
```

(PredicateOf

```
[LAMBDA (assert)                                           (* edited: " 5-Feb-87 10:55")
  (COND
     ((ATOM (CAR assert))
      (CAR assert))
     (T (CAAR assert])
```

(RmAxiom

```
[LAMBDA (assert)                                           (* edited: " 5-Feb-87 11:00")
  (AND (LISTP assert)
       (OR (SETQ *predicates* (DREMOVE (PredicateOf assert)
                                       *predicates*))
           T)
       (PUTPROP (PredicateOf assert)
              (QUOTE axiom)
              NIL])
```

(SET.H.MODE

```
[LAMBDA (MODE NUM)                                         (* edited: "10-Feb-87 15:11")
  (COND
```

```
       ((NULL MODE)

          (* Set the mode of demonstration for H read-prove-print cicle : MODE may be "all" , "interactive" or "first")

         (LIST (IF (EQ MODE interactive)
                   THEN (QUOTE interactive)
                 ELSE *int-mode*)
               *l-trace* *limit*))
       ((NUMBERP NUM)
        (SETQ *limit* NUM))
       ((MEMBER MODE (QUOTE (all interactive first)))
        (IF (EQ MODE interactive)
            THEN (SETQ *int-mode* T)
          ELSE (SETQ *int-mode* MODE)))
       (T (ERROR "Unknown mode in SET.H.MODE"])
```

(Write

```
  [NLAMBDA ARGS                                                    (* edited: " 3-Sep-86 11:42")
    (for i in ARGS do (PRINTOUT T (EVAL i)
                              T])
```

(fact?

```
  [LAMBDA (fact)                                                   (* edited: " 2-Oct-86 13:06")
    (NULL (CDR fact])
```

(with?

```
  [LAMBDA (assert)                                                 (* edited: " 2-Oct-86 13:42")
    (EQ (CAR assert)
        (QUOTE with])

)

(DECLARE: EVAL@COMPILE

(PUTPROPS H.strip MACRO NIL)
)

(PUTPROPS / axiom ((/)))

(PUTPROPS assert prim-funct [LAMBDA (ARGS)
                               (if H.FROM.WINDOW then [if [NOT (MEMBER (CAR ARGS)
                                                                       (GETWINDOWPROP HWINDOW (QUOTE
                                                                                                *predicates*
                                                                                                ]
                                                                 then
                                                                 (PUTWINDOWPROP HWINDOW (QUOTE *predicates*)
                                                                        (CONS (CAR ARGS)
                                                                              (GETWINDOWPROP HWINDOW (QUOTE
                                                                                                *predicates*
                                                                                                 ]
                                                      else
                                                      (SETQ *predicates* (CONS (CAAR ARGS)
                                                                               *predicates*)))
                                                [CAR (PUTPROP (CAR ARGS)
                                                              (QUOTE axiom)
                                                              (CONS (LIST ARGS)
                                                                    (GETPROP (CAR ARGS)
                                                                             (QUOTE axiom]
                                                T])

(PUTPROPS delete prim-funct [LAMBDA (axiom)
                               (PUTPROP (CAR axiom)
                                      (QUOTE axiom)
                                      (H.delete axiom (GETPROP (CAR axiom)
                                                               (QUOTE axiom])

(PUTPROPS set prim-funct [LAMBDA (x1 y1)
                           (LET ((lispvalue (EVAL y1)))
                                (COND ((OR (NOT (ATOM x1))
                                           (NUMBERP x1))
                                       (Write "ERROR IN SET")
                                       NIL)
                                      ((GETPROP x1 (QUOTE variable))
                                       (SETQ save (CONS (CDAR stack)
                                                        save))
                                       (RPLACA (CDAR stack)
                                              (CONS [CONS x1 (CONS lispvalue (QUOTE (NIL]
                                                    (CADAR stack)))
                                       T)
                                      (T NIL])

(DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA Write H.show H.attach H.?)
```

```
(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX

MACRO INDEX