<AFFIRM>LIMBO..10 17-Jun-81 17:40:44

```
(FILECREATED " 1-Apr-81 10:43:19" <AFFIRM>LIMBO..10 43908

     changes to:  LIMBOFNS

     previous date: "31-Mar-81 18:56:41" <AFFIRM>LIMBO..9)


(PRETTYCOMPRINT LIMBOCOMS)

(RPAQQ LIMBOCOMS [(FNS * LIMBOFNS)
                  (PROP UCASE * LOWERCASE)
                  (MACROS * LIMBOMACROS)
                  (DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
                            (ADDVARS (NLAMA)
                                     (NLAML LocalDec)
                                     (LAMA])

(RPAQQ LIMBOFNS (ASKUSERLC ASKUSERLC? AffirmRATOMS AnyErrorCases AskRaise CONFIRMED Ceval CevalHelper
                           Checks Commatize DisjunctedChainingsAndNarrowings Error GET\WRITE\DATE
                           GRIPE\XIVUS Implementation Induct IsLemma LetError ListRules LocalDec
                           OnStanfordDisplay Programs ReadRest Recheck Recheck1 RemoveErrorCases
                           RepInv RestoreRules SOMEGensym SemanticRules SimpAnd SimpImp SimpNot Split
                           Subgoal Translate1 TryAllChainingsAndNarrowings TryToProve TypeOf
                           UpdateTimer UseEqs annotate apply assume clear copy declareFun define dtvs
                           equus fix implementation induct infix nextis normalize print1 put quit
                           readNodeExpression recheck reclaim rule saveproof schema show subgoal
                           supersearch try turn type types undo vcs))
(DEFINEQ

                                                                                              1

(ASKUSERLC
  [LAMBDA (LCWAIT LCDEFAULT LCMESS LCKEYLST LCTYPEAHEAD LCLISPXPRNTFLG LCOPTIONSLST LCFILE)
                                                                    (* Edited by R.Bates on 7-NOV-77:
                                                                    from version 50)

          (* * This function is used to get lowercase recognized by ASKUSER  Used when system flag USESLOWERCASE and the
          KEYLST is a list of things like functions. VC names. etc )


    (RESETFORM (AskRaise NIL)
               (ASKUSER LCWAIT LCDEFAULT LCMESS LCKEYLST LCTYPEAHEAD LCLISPXPRNTFLG LCOPTIONSLST
                        LCFILE])

                                                                                              2

(ASKUSERLC?
  [LAMBDA (LCWAIT LCDEFAULT LCMESS LCKEYLST LCTYPEAHEAD LCLISPXPRNTFLG LCOPTIONSLST LCFILE)
                                                                    (* Edited by R.Bates on 7-NOV-77:
                                                                    from version 54)

          (* * This function is used to get lowercase recognized by ASKUSER if USESLOWERCASE is true.)


    (RESETFORM (AskRaise ~USESLOWERCASE)
               (ASKUSER LCWAIT LCDEFAULT LCMESS LCKEYLST LCTYPEAHEAD LCLISPXPRNTFLG LCOPTIONSLST
                        LCFILE])

                                                                                              3

(AffirmRATOMS
  [LAMBDA (stop file readTable)                                     (* D.Thompson " 5-Nov-79 22:50")
    (InterLispRATOMS stop file readTable])

                                                                                              4

(AnyErrorCases
  [LAMBDA (x)                                                       (* Edited by R.Bates on 24-JAN-78:
                                                                    from version 43)

    (if (NLISTP x)
        then NIL
      elseif (RenameAtoms x:Operator)=ErrorOp
        then T
      else (for y in x:Arguments thereis (AnyErrorCases y])

                                                                                              5
```

```
(AskRaise
  [LAMBDA (FLG)
    (RAISE FLG ASKUSERTTBL])
```

6

```
(CONFIRMED
  [LAMBDA (MESSAGE DEFAULT)
    (PROG (RESPONSE)
          (RESPONSE←(ASKUSER TIME\DEFAULT DEFAULT MESSAGE '((Y "es" RETURN T)
                                (N "o" RETURN NIL))
                             TYPE\AHEAD LISPX\SAVE))
          (TERPRI)
          (RETURN RESPONSE])
```

7

```
(Ceval
  [LAMBDA (x)
    (CevalHelper x])
```

8

```
(CevalHelper
  [LAMBDA (x)                                          (* D Thompson "27-Jul-79 15:38")
    (if LessOutputDesired
      else (PrettyPrint x)
           (printout NIL T "Normalization:"))
    x←(EVAL x)
    x←(TreatFreeAsGiven x)
    (/SET 'AfterNorm x)
    (/SET 'PrettyNorm NIL)
    (if AfterNorm:Operator=QOP and AfterNorm:expr=TRUE
        then (/SET 'AfterNorm TRUE))
    (PrettyPrint AfterNorm NIL T)                      (* T->saves PrettyNorm)
    (if AfterNorm~=TRUE and (AnyInnerIfs AfterNorm:expr NIL)
        then (if AutoCases
                 then (Heading "case analysis:")
                      (cases NIL NIL T)
               else (printout NIL T "(case analysis can be performed)")))
    AfterNorm])
```

9

```
(Checks
  [LAMBDA (ex checkword)                               (* Edited by R.Bates on 24-JAN-78.
                                                          from version 43)
    (if (NLISTP ex) or ex:Operator='Quote
        then TRUE
      else (PROG (p name)
                 (name←(PACK <checkword ex:Operator>))
                 (p←(if (FGETD name)
                        then <name ! ex:Arguments>
                      else TRUE))
                 (for arg in (REVERSE ex:Arguments) unless (NLISTP arg) or ex:Operator='Quote
                    do p←(SimpAnd (Checks arg checkword)
                                  p))
                 (RETURN p])
```

10

```
(Commatize
  [LAMBDA (list)                                       (* D.Thompson "19-Sep-80 16:50")
    (for a on (MKLIST list) join (if a::1
                                     then <a:1 Comma>
                                   else <a:1>])
```

11

```
(DisjunctedChainingsAndNarrowings
  [LAMBDA (x level substitutions)                      (* edited: " 6-Apr-79 09:14")
    (PROG (clist)
          (clist←(Chainings x))
          (if clist
```

```
            then (PRINTLINES T (LENGTH clist)
                               "chaining(s)" T)
                 (for c in clist as chainingNo from 1
                    do (PRINTLINES T "at level" level T)
                       (AfterNorm←(EVAL <IFOP x TRUE (SUBLIS c x)
                                        >))
                       (PrettyNorm←NIL)
                       (if AfterNorm=TRUE
                           then (PRINTLINES T "Proved by chaining and narrowing" T
                                            "using the substitution "
                                            T)
                                (PROG (subs n)
                                      (subs←(PutForm (for c in <c ! substitutions> join c)))
                                      (PrettyPrint subs)
                                      (n←(AddPredicate subs 'put))
                                      (AssumptionsUsed← < !! AssumptionsUsed n>))
                         elseif ~(EQUAL AfterNorm x)
                           then (DisjunctedChainingsAndNarrowings AfterNorm level+1
                                                                  <c ! substitutions>))
                       repeatuntil AfterNorm=TRUE))
          (RETFROM 'supersearch])
```

12

```
(Error
  [LAMBDA (s i)
    <'Error s i>])
```

13

```
(GET\WRITE\DATE
  [LAMBDA (FILE)

          (* * Given a File GET\WRITE\DATE return the date the file was written as a number. To convert to a string call IDATE
          To convert a string which is a date call IDATE. The number is such that if date1 is before date2
          (both strings) then IDATE (date1) < IDATE (date2) )


     (PROG (TEMP JFN)
           (if (INFILEP FILE)
               then JFN←(GTJFN FILE NIL NIL 8590721024))
           (if JFN=NIL
               then (PRIN1 FILE)
                    (PRIN1 " not found")
                    (RETURN 0))
           (TEMP←(JSYS 51 JFN 262156 1 1))
           (RLJFN JFN)
           (RETURN TEMP])
```

14

```
(GRIPE\XIVUS
  [LAMBDA (GRIPE\SUBJECT)                                  (* Edited by R.Bates on 27-JAN-76.
                                                           from version 64)

     (PROG (outfile)
           (if (NCHARS GRIPE\SUBJECT) gt 39
               then (PRIN1 "Subject is too long")
             else outfile←(OUTFILEP (PACKFILENAME 'NAME 'XIVUS 'EXTENSION 'TEMP))
                  (if outfile=NIL
                      then (PRIN1 "Can not write XIVUS.TEMP")
                           (TERPRI)
                           (RETURN))
                  (OUTPUT (OUTFILE outfile))
                  (PRIN1 GRIPE\SUBJECT outfile)
                  (CLOSEF outfile)
                  (KFORK (SUBSYS (PACKFILENAME 'DIRECTORY VERIFICATIONDIRECTORY 'NAME
                                               "INTERLISP↑V\GRIPES↑V\XIVUS")))
                  (RETURN (DELFILE outfile])
```

15

```
(Implementation
  [LAMBDA (Type ImpName)                                   (* D.Thompson "19-Aug-80 11:11")
    (PROG (TypeName ImpType)
          (TypeName←(MakeExtension (Name Type)
                                   ImpName))
          (ImpType←(if (NLISTP Type)
```

```
                then TypeName
              else <TypeName ! Type:Arguments>))
        (DeclareType ImpType)
        (PUTPROP ImpName 'Type Type)
        (IMPLIST← <ImpName>)
        (for f in (FUNCTION Type) do (MakeFunction <(MakeExtension f ImpName) !(ARGLIST f)
                                                   >))
        (Edit ImpType])
```

16

```
(Induct
  [LAMBDA (variable predicate)                          (* D.Musser "18-OCT-78 10:49")
    (PROG (schema)
          (if ~(ASSOC variable predicate:given)
              then (PRINTLINES T "*** illegal induction variable" (Shorten variable)
                               T)
                   (ERROR!))
          (schema←(GETPROP (TypeOf variable)
                           'InductionSchema))
          (/SET 'AssumptionsUsed <(AddPredicate schema 'induction) ! AssumptionsUsed>)
          (CompileRuleIntoLisp <EQOP <'Prop\Induction variable>
                                     (create Qexpression
                                             given ←(REMOVE <variable> predicate:given)
                                             free ←(<variable>) using predicate)
                                     >
                               'rule)
          (RETURN schema])
```

17

```
(IsLemma
  [LAMBDA (p)
    p:Operator:lemma])
```

18

```
(LetError
  [LAMBDA (alist subspair)
    (PRINTLINES T "*** " (INFIX\PRINT3 subspair:1)
                "does not appear in the find list." T
                "  therefore substitution for it is not permitted"
                T)
    (ERROR!])
```

19

```
(ListRules
  [LAMBDA (Type Kinds)                                  (* D.Thompson "13-Jun-79 12:01")
    (PROG (ifuns funs)
          (ifuns←(Functions Type T))
          (funs←(Functions Type))
          (for k in Kinds bind rules do (if k='decl
                                            then (if rules←Type:LocalDeclarations
                                                     then (Heading "decls")
                                                          (PrintDecls rules))
                                                 else rules←(for f in (if (MEMB k '(interface macro))
                                                                          then ifuns
                                                                        else funs)
                                                                    join (Rules f <k>))
                                                 (if k='interface
                                                     then rules←(for r in rules
                                                                   collect <'ExpressionWithType
                                                                            (REMOVE NIL r:LHS)
                                                                            r:RHS:Type>))
                                                 (if rules
                                                     then (Heading (if (EQLENGTH rules 1)
                                                                       then k
                                                                     else (CONCAT k "s")))
                                                          (for r in rules
                                                             do (printout NIL .)
                                                                (INFIX\PRINT3 (REMOVE NIL r:LHS))
                                                                (if k MEMB '(interface representation)
                                                                    then (PRIN1 ": ")
                                                                  else (if (ILEQ ((LINELENGTH)
                                                                                  -(POSITION))*100/(
                                                                                   LINELENGTH)
```

```
                                                    RuleLHSPercentage)
                                                then (printout NIL 8 "=" .)
                                               else (printout NIL . "=" .)))
                                        (INFIX\PRINT3 r:RHS)
                                        (printout NIL .TAB0 0])
```

20

```
(LocalDec
  [NLAMBDA (ivars% vars% type% )
    (for ivar% in ivars% as var% in vars% do (SET ivar% <'ExpressionWithType var% type% >)
                                             (SET var% var% ))
    (MAPRINT (for var% in vars% collect (RenameAtoms var% ))
             T NIL ': ',)
    (INFIX\PRINT3 type% )
    (TERPRI])
```

21

```
(OnStanfordDisplay
  [LAMBDA NIL                                    (* Selects the stanford characters)
    CHARSET←'STANFORDCHARSET])
```

22

```
(Programs
  [LAMBDA (Type)
    (for f in (Functions Type) join (Rules f <'program >])
```

23

```
(ReadRest
  [LAMBDA (file stop)                            (* D.Lusser "25-Apr-79 14.38")
    (PROG (rest)
          (rest←(RATOMS stop file PASCAL\READ\TABLE))
          (LISPXHIST:1← <command ! rest ': >)
          (RETURN rest])
```

24

```
(Recheck
  [LAMBDA (rootnumb)                             (* edited: " 6-SEP-78 17.14")
    (PROG (visited)                              (* run through the proof of this theorem
                                                    We subgoal (if not proven) so user can take up where
                                                    left off.)
          (if AfterNorm~=TRUE
              then (Subgoal))
          (Recheck1 rootnumb])
```

25

```
(Recheck1
  [LAMBDA (tno)                                  (* edited: " 6-Apr-79 09:11")
    (PROG ((subproofs (SubProofs (ASSOC tno TheoremList:ProofList)))
           didsubgoals)
          (* recursively reenact the proof of this thm and all descendants. we use recheckLemma? to determine whether we
          should do the proofs of propositions used.)

          (visited← <tno ! visited>)
          (retrycaller 'try tno)
          [for sub in subproofs bind n event pred quit eachtime (n←sub:1)
                                                                (event←sub:2)
                                                                (pred←(GetPredicate n TheoremList))
             repeatuntil quit do (SELECTQ event
                                          ((let put)
                                            (retrycaller event pred))
                                          (invoke (retrycaller event pred:Arguments)
                                                                (* definitions (--))
                                                  )
                                          (schema (retrycaller 'employ pred)

          (* employ induction: hop directly to the generated cases (not via intervening subgoal). as would user
          (1 thm 2 schema (3 subgoal (4 case) (5 case))))
```

```
                                        quit←didsubgoals←T
                                        (for s in (SubProofs (LAST subproofs):1)
                                           do (Recheck1 s:1)))
                                    (complete (retrycaller 'complete (FetchLog 'complete)))
                                    (PROG ((type (sub:3)))
                                          (SELECTQ type
                                                  (proposition (retrycaller 'use pred))
                                                  ((subgoal subgoaling)
                                                    (PROG ((l (FetchLog 'SUB)))
                                                          (quit←T)
                                                     (* rest of subproofs came from event)
                                                          (retrycaller l:1 l:2)
                                                     (* split subgoal or suppose)
                                                     ))
                                                  (PROGN (PRINTLINES T "unable to retry the "
                                                                     event "command")
                                                         (ERROR!]
        (if ~didsubgoals
            then (for s in subproofs when s:3 MEMB '(proposition subgoal subgoaling)
                                      and s:1 ~MEMB visited
                                      and (if s:3='proposition
                                             then (recheckLemma? s:1 tno)
                                           else T)
                 do (Recheck1 s:1)                    (* we may have split or whatever.
                                                      Recursively prove subgoals. lemmas)])
```

26

```
(RemoveErrorCases
  [LAMBDA (x)                                        (* Edited by R.Bates on 24-JAN-78;
                                                     from version 43)

    (if (NLISTP x)
        then x
      elseif x:Operator=IFOP
        then [PROG (y z)
                   (y←(RemoveErrorCases x:ThenPart))
                   (z←(RemoveErrorCases x:ElsePart))
                   (RETURN (if y=x:ThenPart and z=x:ElsePart
                               then x
                            else (IfThenElse x:Test y z]
      elseif (AnyErrorCases x)
        then TRUE
      else x])
```

27

```
(RepInv
  [LAMBDA (ex type repOp)
    (PROG (a r)
          (a←(AtomsOfType ex type))
          (a←(INTERSECTION a a))
          (r←TRUE)
          (for b in a do r←(SimpAnd <repOp b> r))
          (RETURN r])
```

28

```
(RestoreRules
  [LAMBDA NIL                                        (* edited: " 6-Apr-79 10:30")
    (PROG (added deleted)
          (added←AddedRules)
          (deleted←DeletedRules)
          (for r in deleted unless (MEMBER r added) do (CompileRuleIntoLisp r 'lemma))
          (for r in added unless (MEMBER r deleted) do (RemoveRule r])
```

29

```
(SOMEGensym
  [LAMBDA (a b)                                      (* Edited by R.Bates on 8-FEB-78:
                                                     from version 37)

    (PROG (w atom)
          (atom←(if (LISTP a)
                    then a:1
                 else a))
          [w←(if (SASSOC b atom:NewSymbolMapping)::1
```

```
            else (PROG (name)
                       (name←(for n from (LENGTH atom:NewSymbolMapping)
                                 bind (base ←(RenameAtoms atom))
                                      newvar
                                 do newvar←(PACK <base n>) repeatuntil (EVALV newvar)='NOBIND
                                 finally (RETURN newvar)))
                       (atom:NewSymbolMapping← <!! atom:NewSymbolMapping <b ! name>>)
                       (RETURN name]
      (SET w w)
      [SET (PACK <w '\Interface >)
           (ExpressionWithType w (fetch Type of (Translate (RenameAtoms atom]
      (RETURN (SUBST w a b])
```

30

```
(SemanticRules
  [LAMBDA (type)
    (for f in (Functions type) join (Rules f T])
```

31

```
(SimpAnd
  [LAMBDA (x y)
    (if x=TRUE
        then y
      elseif y=TRUE
        then x
      else <ANDOP x y>])
```

32

```
(SimpImp
  [LAMBDA (h c)                                    (* Edited by R.Bates on 24-JAN-78
                                                   from version 43)

    (if (NLISTP c)
        then (if c=TRUE
                 then TRUE
               elseif c=FALSE
                 then (SimpNot h)
               else <IMPOP h c>)
      elseif h=TRUE
        then c
      elseif c:1=IMPOP
        then <IMPOP (SimpAnd h c:2)
                    c:3>
      else <IMPOP h c>])
```

33

```
(SimpNot
  [LAMBDA (x)
    (if x=TRUE
        then FALSE
      elseif x=FALSE
        then TRUE
      elseif x:Operator=NEOP
        then <EQOP ! x:Arguments>
      elseif x:Operator=EQOP
        then <NEOP ! x:Arguments>
      else <NOTOP x>])
```

34

```
(Split
  [LAMBDA (goals)                                  (* R.Erickson "12-Apr-79 16:36")
                                                   (* accepts a list of fragments, which are subgoals
                                                   if they are not quantified. uses AfterNorm)

    (Subgoal)
    (PROG (new)
          [new←(for s in goals collect (AddPredicate (if (type? Qexpression s) or s=TRUE
                                                         then s
                                                       else (create Qexpression
                                                                    expr ← s using AfterNorm))
                                                     '(new proposition]
          (if new::1=NIL and new:1=CurrentTheoremNumber
            else (if (UpdateProof CurrentTheoremNumber new TheoremList:ProofList)='proved
```

```
                then (UpdateProofList TheoremList:ProofList <CurrentTheoremNumber>)
              else (RETURN new:1])
```

35

```
(Subgoal
  [LAMBDA NIL                                                    (* D.Musser "28-DEC-78 22:32")
    (PROG (n)
          (n←(AddPredicate AfterNorm '(new subgoaling)))
          (if n~=CurrentTheoremNumber
              then (/SET 'AssumptionsUsed <n ! AssumptionsUsed>)
                   (UpdateProof CurrentTheoremNumber (REVERSE AssumptionsUsed)
                                TheoremList:ProofList)
                   (/SET 'CurrentTheorem AfterNorm)
                   (/SET 'CurrentTheoremNumber n)
                   (/SET 'AssumptionsUsed NIL])
```

36

```
(Translate1
  [LAMBDA (atom)                                                 (* D.Thompson "12-Aug-80 08:54")
                                                                 (* faster version for LITATOMS)
    (EVAL (SUBLIS CurrentContext (MakeExtension atom 'Interface])
```

37

```
(TryAllChainingsAndNarrowings
  [LAMBDA (x path)                                               (* edited " 6-Apr-79 10:30")
    (PROG (clist)
          (clist←(Chainings x))
          (if clist
              then path← <(LENGTH clist) ! path>
                   (for c in clist as chainingNo from 1
                      do (AfterNorm←(EVAL (SUBLIS c x)))
                         (PrettyNorm←NIL)
                         (if AfterNorm=TRUE
                             then NoOfProofs←NoOfProofs+1
                                  SearchSpace← <<'proof ! path> ! SearchSpace>
                           elseif (MEMBER AfterNorm SearchList)
                             then Duplications←Duplications+1
                           else SearchList← <AfterNorm ! SearchList> (TryAllChainingsAndNarrowings
                                  AfterNorm path)))
            else SearchSpace← <path ! SearchSpace>])
```

38

```
(TryToProve
  [LAMBDA (p TL)                                                 (* D.Musser "18-OCT-78 11:52")
    (PROG (n proof)
          (n←(if (NUMBERP p)
                 then p
               else (SASSOC p TL:PredicateList)::1))
          (proof←(GetProof n TL))
          (for f in SkolemFunctions do (PUTD f NIL))
          (/SET 'SkolemFunctions NIL)
          (if p:Operator=QOP)
          (Ceval p)
          (CheckForSuccess])
```

39

```
(TypeOf
  [LAMBDA (x)                                                    (* R.Bates "14-Dec-79 14:43")
    (fetch Type of (CDR (FASSOC (ExtendName (Shorten x)
                                            'Interface)
                                (Extension x):LocalDeclarations])
```

40

```
(UpdateTimer
  [LAMBDA NIL                                                    (* D Thompson "17-Sep-79 12:34")
    (PROG (elapsedTime previousTimingMark)
          (previousTimingMark←CurrentTimingMark)
          (CurrentTimingMark←(CLOCK 2))
          (if Timer and (FIXP previousTimingMark)
```

```
        then (elapsedTime ← CurrentTimingMark -previousTimingMark)
             (if elapsedTime gt 0
                 then (printout NIL .TAB0 0 (QUOTIENT elapsedTime 1000.0)
                                . "seconds." T])
```

41

```
(UseEqs
  [LAMBDA (ex)                                    (* Unchecked is modified destructively. so must be
                                                     copied to save it)
    (RESETVARS ((RuleList RuleList)
                (Unchecked (COPY Unchecked)))
               (RETURN (if (ATOM ex)
                           then ex
                         elseif ex:Operator=IFOP
                           then [if (EvalRules ex:Test)=TRUE
                                    then (UseEqs ex:ThenPart)
                                  else (PROG ((oldRuleList RuleList)
                                              (oldUnchecked (COPY Unchecked)))
                                             (Completer
                                               <(if ex:Test:Operator=EQOP
                                                    then ex:Test
                                                  else <EQOP ex:Test TRUE>)
                                               >
                                               'assumption)
                                             (RETURN (if ContradictionFound
                                                         then RuleList←oldRuleList
                                                              (UseEqs ex:ElsePart)
                                                       else (PROG (thenpart elsepart)
                                                                  (thenpart←(UseEqs ex:ThenPart))
                                                                  (if thenpart=TRUE
                                                                      and ex:ElsePart=TRUE
                                                                    then (RETURN TRUE))
                                                                  (RuleList←oldRuleList)
                                                                  (Unchecked←oldUnchecked)
                                                                  (Completer <<EQOP ex:Test FALSE>>
                                                                             'assumption)
                                                                  (if ContradictionFound
                                                                      then (RETURN thenpart)
                                                                    else elsepart←(UseEqs ex:ElsePart)
                                                                         (RETURN
                                                                           <IFOP ex:Test thenpart
                                                                             elsepart>]
                                  else ex←(UseEqs <IFOP ex TRUE FALSE>)
                                       (if ex:ThenPart=TRUE and ex:ElsePart=FALSE
                                           then ex:Test
                                         else ex])
```

42

```
(annotate
  [LAMBDA (ratoms)                                (* D.Thompson " 1-Nov-79 17:46")
    (PROG (node)
          (if ratoms:2=','
              then node←ratoms:1
                   ratoms←ratoms::2
            elseif (GLC ratoms:1)=','
              then                                (* user omitted space)
                   node←(UNPACK ratoms:1)
                   node::-1←NIL
                   node←(PACK node)
                   ratoms←ratoms::1
            elseif ~ratoms
              then (AffirmError "You need an annotation."))
          (Annotate ratoms node])
```

43

```
(apply
  [LAMBDA (file stop)
    (use file stop T])
```

44

```
(assume
  [LAMBDA (file stop)                             (* R.Erickson "22-Oct-79 16 15")
    (Assume (readNodeExpression file stop])
```

45

```
(clear
  [LAMBDA (file stop)                                    (* R.Erickson "15-Oct-79 16:02")
    (PROG (what)
          (LISPXHIST:1←command)
          (what←(RATOM file PASCAL\READ\TABLE))
          (SELECTQ what
                   (implist (/SET 'IMPLIST NIL))
                   (proof (ClearProof))
                   (AffirmError " clear what? (proof or implist are only choices currently)"))
          (if what~='';
              then (RATOMS stop file PASCAL\READ\TABLE])
```

46

```
(copy
  [LAMBDA (file stop)                                    (* D.Thompson " 5-Oct-79 11:09")
    (PROG (file)
          (file←(PACK (RATOMS stop file PASCAL\READ\TABLE)))
          (TypeFile file])
```

47

```
(declareFun
  [LAMBDA (File Stop)                                    (* R.Erickson "26-Sep-79 13:15")
    (PROG (x Type)
          (x←(parse 'functionDecl File Stop))
          (if x
              then x:expression#←(reduceExpression x:expression#)
                   Type←(CheckForType x:expression#)
                   (for y in x:expression do (y←(reduceExpression y))
                                             (DeclareFun y Type])
```

48

```
(define
  [LAMBDA (file stop)                                    (* R.Bates "18-Sep-79 12:43")
    (ruleSeqParse file stop 'Define])
```

49

```
(dtvs
  [LAMBDA (file)
    (DTVS file])
```

50

```
(equus
  [LAMBDA (file stop)
    (RATOMS stop file PASCAL\READ\TABLE)
    (RESETVARS ((UsingRuleList T)
                (RuleList RuleList)
                (Unchecked NIL)
                impform hypothesis conclusion)
               (impform←(RemoveIfs (Instance NIL AfterNorm)))
               (if impform:Operator=IMPOP
                   then AssumptionsUsed← <!! AssumptionsUsed (AddPredicate TRUE 'equus) >
                        (Completer (for hypothesis in (ListOfConjuncts impform:Arg1)
                                        collect (if hypothesis:Operator=EQOP
                                                    then hypothesis
                                                  elseif hypothesis:Operator=NOTOP
                                                    then <EQOP hypothesis:Arg1 FALSE>
                                                  else <EQOP hypothesis TRUE>)))
                        (if ContradictionFound
                            then (PRINTLINES T "Contradiction found in hypotheses" T)
                                 AfterNorm←TRUE
                                 PrettyNorm←NIL
                          else conclusion←TRUE
                               (for c in (REVERSE (ListOfConjuncts impform:Arg2))
                                  do conclusion←(SimpAnd (EvalRules c)
                                                         conclusion))
                               (if conclusion=TRUE
                                   then AfterNorm←TRUE
```

```
                                    PrettyNorm←NIL
                                    (PRINTLINES T TRUE T)
                               else hypothesis←RuleList:1
                                    (for h in RuleList::1
                                       do hypothesis←(SimpAnd (if h:RHS=TRUE
                                                                  then h:LHS
                                                                else h)
                                                              hypothesis))
                                    AfterNorm←(create Qexpression
                                                      expr ←(MakeImplies hypothesis conclusion)
                                                    using AfterNorm)
                                    PrettyNorm←NIL
                                    (PrettyPrint AfterNorm NIL T)))
                          (CheckForSuccess)
                     else (PRINTLINES T
                               "Current goal must be split into implications before using equalities"
                                     T])
```

51

```
(fix
  [LAMBDA (file stop)
    (redo file stop T])
```

52

```
(implementation
  [LAMBDA (File Stop)                                    (* Edited by R.Bates on 6-FEB-78:
                                                            from version 55)
    (PROG (x)
          (x←(parse 'expression File Stop))
          (if x
              then (if x:1='by
                       then (Implementation x:2 x:3)
                     else (Heading "**** Corrent form is implementation TYPE by IMPNAME"])
```

53

```
(induct
  [LAMBDA (file stop)                                    (* D.Musser "16-OCT-78 10:17")
    (PROG (atoms)
          (atoms←(RATOMS stop file PASCAL\READ\TABLE))
          (atoms←(REMOVE 'on (REMOVE 'ON atoms)))
          (if atoms::1
              then (PRINTLINES T "can only induct on one variable" T)
                   (ERROR!)
            else (/SET 'AfterNorm (Induct (ExtendName atoms:1 'Dtvs)
                                          AfterNorm))
                 (/SET 'PrettyNorm NIL)
                 (Ceval AfterNorm)
                 (CheckForSuccess])
```

54

```
(infix
  [LAMBDA (opList)                                       (* D.Thompson " 4-Oct-79 10:20")
    (Infix opList])
```

55

```
(nextis
  [LAMBDA (given)                                        (* edited: " 6-Apr-79 08:48")
    (PROG [from spl (opens '(new untried assumed]
          (from←(RootOf (if given
                            else CurrentTheoremNumber)))
          (spl←(ASSOC from TheoremList:ProofList))
          (if spl
              then [RETURN (if spl MEMB opens
                               then <from>
                             else (REMOVEDUPLICATES (for l in (SubProofs spl) join (leaves l opens]
            else (PRINTLINES T from "?" T])
```

56

```
(normalize
```

```
[LAMBDA (file stop)                                    (* R.Erickson "29-Sep-79 23:35")
  (ReadRest file stop)
  (Normalize T T])
```

57

(print1

```
[LAMBDA (type target option)                           (* D.Thompson " 3-Nov-79 14:22")

        (* * This routine is called by print. The real work is done here.)


   (SELECTQ type
            (implist (MAPRINT IMPLIST)
                     (TERPRI))
            (original (PrettyPrint (GetExpression CurrentNode:prop#)))
            (result (PrettyPrint CurrentPropn NIL T))
            (variables (PROG (temp)
                             (if CurrentPropn:1~=QOP
                                 then (RETURN))
                             (temp←(COPY CurrentPropn))
                             (temp:expr←'...)
                             (PrettyPrint temp)))
            [next (PROG (nx)
                        (nx←(Next))
                        (SELECTQ nx:1
                                 (STAY (printout NIL T "At the only leaf"))
                                 (LEAF (printout NIL T "Next leaf is:")
                                       (printPropn nx:2:1))
                                 (DOWN (AFFIRMMAPRINT "Next lemmas are:"
                                                      (for n in nx::1 collect n:-1)))
                                 (UP (if nx::2
                                         then (AFFIRMMAPRINT "Unproven ancestors are" nx::1)
                                       else (printout NIL T "Unproven ancestor is" . nx:2)))
                                 (NIL (printout NIL T "proof of this part done" T)
                                      (printHelper 'status '(unproven)))
                                 (SHOULDNT]
            [(proof both named)
              (for nodes on (printWhat? target T) do (SELECTQ type
                                                              (proof (PrintProof nodes:1 option))
                                                              (both (PrintBoth nodes:1 option))
                                                              (named (PrintProof nodes:1 'named))
                                                              (SHOULDNT))
                                                     (if nodes::1
                                                         then (printout NIL .SKIP 4]
            (prop (for it in (printWhat? target '*) do (printPropn it)))
            [status (for node in (printWhat? target 'theorems) bind thm
                       do (printout NIL .TAB0 0 (NodeId node)
                                    " is "
                                    (if thm←(NodeToThm node)
                                        then (PrettyStatus thm)
                                      else "not a theorem!"]
            [uses (printout NIL T "Theorem" .TAB0 25 "Uses:" .TAB0 45 "Used by:")
                  (for node in (printWhat? target 'theorems) bind thm
                     do (thm←(NodeToThm node))
                        (if thm
                            then                                   (* else ignore it)
                                 (printout NIL .TAB0 0 (TheoremId thm)
                                           "," . (PrettyStatus thm)
                                           .TAB0 25 ((for t in (Facts thm) collect (TheoremId t))
                                              or "-")
                                           .TAB0 45 ((for t in (UsedBy thm) collect
                                                                              (TheoremId t))
                                              or "-"]
            (names (printout NIL T)
                   (AFFIRMMAPRINT "known names are:" (Names)
                                  (printout NIL T)))
            (type target← (CheckForType target)
                  (printout NIL T (printName 'type 'SystemPrintedCommand)
                            .
                            (printName target 'TypeName)
                            ";" T)
                  (listTypeParts target option)
                  (printout NIL T (printName 'end 'SystemPrintedCommand)
                            . "{" (printName target 'TypeName)
                            "} ;" T))
            [assumption                                            (* list all assumptions and where used)
                        (PROG (pairs)
                              (pairs←(for t in Theorems when t:status='assumed
                                        collect <(TheoremId t)
```

```
                                        (UsedBy t)
                                        >))
                        (SORT pairs T)                  (* by ALPHORDER (CAR))
                        (printout NIL ('assumptions:))
                        (for pair in pairs do (printout NIL T T pair:1 .TAB0 10)
                                                    (* name)
                                              (if pair:2
                                                  then (printout NIL "used by" .)
                                                       (MAPRINT (for t in pair:2
                                                                   collect (TheoremId t))
                                                                NIL NIL NIL ", ")
                                                else (printout NIL "unused"))
                                              (printout NIL .TAB0 10)
                                              (PrettyPrint (NodeToExpr pair:1)
                                                           T]

              (SHOULDNT])
```

58

```
(put
  [LAMBDA (file stop retrycom)                    (* Edited by Erickson on 16-AUG-78:
                                                  from version 4)
    (let file stop T retrycom])
```

59

```
(quit
  [LAMBDA NIL                                     (* D Thompson " 1-Nov-79 16:41")

          (* * This routine performs the LISP LOGOUT command -
          An AFFIRM command function.)


    (printout NIL .TAB0 0 "Type CONTINUE to return to AFFIRM.")
    (LOGOUT];
```

60

```
(readNodeExpression
  [LAMBDA (file stop)                             (* D Thompson " 5-Nov-79 14:03")

          (* * Used to input a theorem, - = the current one. Does Pexec interfacing, and allows the user to give a name.
          Returns the node


    (PROG (input name node)
          (input←(parse 'expressionSeq file stop):expression)
          (if input::1
              then name←input:1
                   (pop input))
          (node←(ExprToNode (PexecLists input:1 T)))
          (if name
              then (SetName name node))
          (RETURN node])
```

61

```
(recheck
  [LAMBDA (file stop)                             (* Edited by Erickson on 16-AUG-78:
                                                  no file)
    (PROG (n)
          (n←(parse 'expression file stop))
          (if n
              then (if n←(pexec n)
                       then (if (NUMBERP n)
                                then (CheckForPropositionStatus n)
                              else n←(SASSOC n TheoremList:PredicateList)::1
                                   (if n=NIL
                                       then (PRINTLINES T "theorem not found")
                                            (ERROR!)))
                     (Recheck n])
```

62

```
(reclaim
  [LAMBDA (file stop)                             (* R.Erickson "25-May-79 15:57")
```

```
(* reclaims space from unwanted theorems (short of clear proof.) GetPredicate (and maybe others) use NTH to access
PredicateList. so they assume it has no holes  We do put holes in ProofList. however.)


   (PROG (roots bad kept discard au)
         (discard←(RATOMS stop file PASCAL\READ\TABLE))
         (if ~discard
             then (PRINTLINES T "throw nothing out??")
           else (if bad←(for k in discard unless (NUMBERP k) and (GetPredicate k TheoremList)
                                                 and (ASSOC k TheoremList:ProofList)
                           collect k)
                    then (printout NIL T "not existing propns:" , bad)
                         discard←(LDIFFERENCE discard bad))
                roots←(Roots)
                (if bad←(LDIFFERENCE discard roots)
                    then (printout NIL T "not roots:" , bad)
                         discard←(LDIFFERENCE discard bad))
                (if ~discard
                    then (RETURN))
                kept←(LDIFFERENCE roots discard)
                kept←(REMOVEDUPLICATES < !! kept !(for k in kept join (SubProofNumbers k))
                                       >)
                kept←(SORT kept 'IGREATERP)                    (* this makes the order of propns in TL right)
                discard←(for i to TheoremList:NumberPreds unless (MEMB i kept)
                           when (ASSOC i TheoremList:ProofList) collect i)
                (PRINTLINES T "The following will be discarded:" T)
                au←(ASKUSER NIL NIL (LIST discard "
is this acceptable"))
                (if au='Y
                    then (for p in TheoremList:PredicateList do (if ~(MEMB p::1 kept)
                                                                    then p:1←'youshouldntseethis
                                                          (* smash the proposition)
                                                                         ))
                         TheoremList:ProofList←(for k in kept collect (ASSOC k TheoremList:ProofList)
                                                                      )    (* only proof of kept propositions)
                         (if ~(MEMB CurrentTheoremNumber kept)
                             then CurrentTheoremNumber←kept:1
                                  (if kept:1
                                      then AfterNorm←(GetPredicate kept:1 TheoremList)
                                           PrettyNorm←NIL])
```

63

```
(rule
  [LAMBDA (File Stop Kind)                                    (* R.Bates "16-Sep-79 12:47")
    (ruleSeqParse File Stop 'Rule Kind])
```

64

```
(saveproof
  [LAMBDA (file stop)                                         (* R.Erickson "22-Mar-79 16:36")
    (PROG (f)
          [f←(U-CASE (PACK (RATOMS stop file PASCAL\READ\TABLE]
          (/SET (PACK* f 'COMS)
                <<'P <'CheckLoad '(QUOTE PROOF)
                     (KWOTE <AFFIRMVERSION ! AFFIRMSYSOUTFILE>)
                     (KWOTE <PROOFVARS>)
                     >> <'COMS
                   '* 'SYSTEMSTATECOMS >>)
          (Heading (MAKEFILE f])
```

65

```
(schema
  [LAMBDA (file stop)                                         (* R.Bates "18-Sep-79 12:40")
    (ruleSeqParse file stop 'Schema])
```

66

```
(show
  [LAMBDA (file stop)                                         (* edited: " 6-Apr-79 09:10")
    (PROG (kind skind names relation ex callers)
          (kind←(RATOM file PASCAL\READ\TABLE))
          (skind←(Singularize (L-CASE kind)))
          (if ~(skind='all or skind MEMB ListingKinds)
              then (PRINTLINES T kind "?" T)
```

```
            (RATOMS stop file PASCAL\READ\TABLE)
            (ERROR!))
      (ex←(parse 'expression file stop))
      (if ex
          then ex←(pexec ex)
               (if ex
                   then names←(Operators ex)
                        names←(INTERSECTION names names)
                 else (ERROR!))
        else (ERROR!))
      (relation←(PARSERELATION '(CALLS OR USES FREELY)))
      (callers←(GETRELATION names:1 relation T))
      (for name in names::1 repeatwhile callers do callers←(INTERSECTION callers
                                                                  (GETRELATION name
                                                                               relation T)))
      (if callers
          then (for caller in callers do (for rule in (Rules caller
                                                              (if skind='all
                                                                  then ListingKinds
                                                                else skind))
                                                       when (EQUAL (INTERSECTION names (FLATTEN rule))
                                                                   names)
                                                       do (PrettyPrint rule)))
        else (PRINTLINES T "None" T])
```

67

(subgoal

```
  [LAMBDA (file stop retrycom)                          (* Edited by Erickson on 21-AUG-78:
                                                        from version 15)

    (if retrycom
      else (RATOMS stop file PASCAL\READ\TABLE))
    (Log 'SUB '(subgoal dummy))
    (Subgoal])
```

68

(supersearch

```
  [LAMBDA (file stop)                                   (* R.Erickson "15-Jun-79 18:43")
    (RESETVARS ((AfterNorm AfterNorm))
               (RATOMS stop file PASCAL\READ\TABLE)
               (DisjunctedChainingsAndNarrowings (Instance? NIL AfterNorm)
                                                 0)
               (CheckForSuccess])
```

69

(try

```
  [LAMBDA (file stop)                                   (* R.Erickson "21-Oct-79 10:52")

        (* if the target is above us in the chain, rise to it. Else if a child of any theorems above us, pop to the common
        parent, and shift to target. Otherwise, descend to target. In other words, pop whenever possible.
        Command correction to arc has been removed)


    (Try (readNodeExpression file stop])
```

70

(turn

```
  [LAMBDA (file stop)                                   (* D.Thompson "17-Sep-79 14:58")
    (RATOMS stop file PASCAL\READ\TABLE)
    (printout T .TAB0 0 "Please use the 'profile' command instead." T])
```

71

(type

```
  [LAMBDA (file stop)                                   (* D.Thompson "5-Oct-79 11:39")
    (PROG (x)
          (x←(parse 'expression file stop))
          (if x
              then (DeclareType x)
                   (Edit x)
                   (RETURN x])
```

72

```
(types
  [LAMBDA (file stop)                                    (* R.Erickson "14-Sep-79 17:56")
    (ReadRest file stop)
    (MAPRINT KnownTypes NIL " { " " } " ", "])
```

73

```
(undo
  [LAMBDA (file stop)                                    (* R.Bates "20-OCT-78 14:50")
    (PROG (line)
          (SETBRK '(;)
                  1)
          (line←(RATOMS stop file NIL))
          (SETBRK '(;)
                  0)
          (LISPXHIST:1← <'UNDO ! line>)
          (UNDOLISPX line])
```

74

```
(vcs
  [LAMBDA (Imp)                                          (* Edited by R.Bates on 14-SEP-77.
                                                         from version 26)

    (PROG (Type)
          (Type←(AssociatedType Imp))
          (IMPLIST← <Imp>)
          (for ax in (Axioms Type) bind (names ←(NameSubList Imp))
             do TheoremList←(AddPredicate (APPLY 'Using
                                                 <(SimpImp (SimpAnd (RepInv ax Type
                                                                            (PACK
                                                                              <'Rep\ Type '\ Imp>))
                                                                    (Checks ax 'OkToCall))
                                                           ax)
                                                  names>)
                                                'untried TheoremList))
          (for p in (Programs (PACK <Type '\ Imp>)) bind th
             do (th←(SimpImp (Checks p:LHS 'OkToCall)
                             (LegalityChecks p:RHS)))
                (if th~=TRUE
                    then TheoremList←(AddPredicate th 'untried TheoremList])
)

(RPAQQ LOWERCASE (all alters and array assert asserting assume begin by case const difference div do
                      downto else end entry eq eqv exists exit expt false file first for forall
                      function ge go goto gt if imp imports inline label last le lt maintain mod ne
                      not of or otherwise packed plus post pre procedure program prove public record
                      repeat return returns set some then thus times to true type until var while
                      with xpublic))

(PUTPROPS all UCASE ALL)

(PUTPROPS alters UCASE ALTERS)

(PUTPROPS and UCASE AND)

(PUTPROPS array UCASE ARRAY)

(PUTPROPS assert UCASE ASSERT)

(PUTPROPS asserting UCASE ASSERTING)

(PUTPROPS assume UCASE ASSUME)

(PUTPROPS begin UCASE BEGIN)

(PUTPROPS by UCASE BY)

(PUTPROPS case UCASE CASE)

(PUTPROPS const UCASE CONST)

(PUTPROPS difference UCASE DIFFERENCE)

(PUTPROPS div UCASE DIV)

(PUTPROPS do UCASE DO)
```

```
(PUTPROPS downto UCASE DOWNTO)
(PUTPROPS else UCASE ELSE)
(PUTPROPS end UCASE END)
(PUTPROPS entry UCASE ENTRY)
(PUTPROPS eq UCASE EQ)
(PUTPROPS eqv UCASE EQV)
(PUTPROPS exists UCASE EXISTS)
(PUTPROPS exit UCASE EXIT)
(PUTPROPS expt UCASE EXPT)
(PUTPROPS false UCASE FALSE)
(PUTPROPS file UCASE FILE)
(PUTPROPS first UCASE FIRST)
(PUTPROPS for UCASE FOR)
(PUTPROPS forall UCASE FORALL)
(PUTPROPS function UCASE FUNCTION)
(PUTPROPS ge UCASE GE)
(PUTPROPS go UCASE GO)
(PUTPROPS goto UCASE GOTO)
(PUTPROPS gt UCASE GT)
(PUTPROPS if UCASE IF)
(PUTPROPS imp UCASE IMP)
(PUTPROPS imports UCASE IMPORTS)
(PUTPROPS inline UCASE INLINE)
(PUTPROPS label UCASE LABEL)
(PUTPROPS last UCASE LAST)
(PUTPROPS le UCASE LE)
(PUTPROPS lt UCASE LT)
(PUTPROPS maintain UCASE MAINTAIN)
(PUTPROPS mod UCASE MOD)
(PUTPROPS ne UCASE NE)
(PUTPROPS not UCASE NOT)
(PUTPROPS of UCASE OF)
(PUTPROPS or UCASE OR)
(PUTPROPS otherwise UCASE OTHERWISE)
(PUTPROPS packed UCASE PACKED)
(PUTPROPS plus UCASE PLUS)
(PUTPROPS post UCASE POST)
(PUTPROPS pre UCASE PRE)
(PUTPROPS procedure UCASE PROCEDURE)
(PUTPROPS program UCASE PROGRAM)
(PUTPROPS prove UCASE PROVE)
```

```
(PUTPROPS public UCASE PUBLIC)

(PUTPROPS record UCASE RECORD)

(PUTPROPS repeat UCASE REPEAT)

(PUTPROPS return UCASE RETURN)

(PUTPROPS returns UCASE RETURNS)

(PUTPROPS set UCASE SET)

(PUTPROPS some UCASE SOME)

(PUTPROPS then UCASE THEN)

(PUTPROPS thus UCASE THUS)

(PUTPROPS times UCASE TIMES)

(PUTPROPS to UCASE TO)

(PUTPROPS true UCASE TRUE)

(PUTPROPS type UCASE TYPE)

(PUTPROPS until UCASE UNTIL)

(PUTPROPS var UCASE VAR)

(PUTPROPS while UCASE WHILE)

(PUTPROPS with UCASE WITH)

(PUTPROPS xpublic UCASE XPUBLIC)

(RPAQQ LIMBOMACROS (ASKUSERLC? CONFIRMED))
(DECLARE: EVAL@COMPILE

(PUTPROPS ASKUSERLC? MACRO [LAMBDA (LCWAIT LCDEFAULT LCMESS LCKEYLST LCTYPEAHEAD LCLISPXPRNTFLG
                                    LCOPTIONSLST LCFILE)
                            (RESETFORM (AskRaise (NOT USESLOWERCASE))
                                    (ASKUSER LCWAIT LCDEFAULT LCMESS LCKEYLST LCTYPEAHEAD
                                            LCLISPXPRNTFLG LCOPTIONSLST LCFILE])

(PUTPROPS CONFIRMED MACRO [LAMBDA (MESSAGE DEFAULT)
                           (PROG (RESPONSE)
                                 (SETQ RESPONSE (ASKUSER TIME\DEFAULT DEFAULT MESSAGE
                                                         (QUOTE ((Y "es" RETURN T)
                                                                 (N "o" RETURN NIL)))
                                                         TYPE\AHEAD LISPX\SAVE))
                                 (TERPRI)
                                 (RETURN RESPONSE])
)
(DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML LocalDec)

(ADDTOVAR LAMA )
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (1217 40019 (ASKUSERLC 1229 . 1808) (ASKUSERLC? 1812 . 2317) (AffirmRATOMS 2321 . 2480)
(AnyErrorCases 2484 . 2819) (AskRaise 2823 . 2883) (CONFIRMED 2887 . 3144) (Ceval 3148 . 3199) (
CevalHelper 3203 . 3968) (Checks 3972 . 4552) (Commatize 4556 . 4783) (
DisjunctedChainingsAndNarrowings 4787 . 5872) (Error 5876 . 5923) (GET\WRITE\DATE 5927 . 6572) (
GRIPE\XIVUS 6576 . 7263) (Implementation 7267 . 7853) (Induct 7857 . 8580) (IsLemma 8584 . 8635) (
LetError 8639 . 8849) (ListRules 8853 . 10278) (LocalDec 10282 . 10614) (OnStanfordDisplay 10618 .
10773) (Programs 10777 . 10880) (ReadRest 10884 . 11139) (Recheck 11143 . 11557) (Recheck1 11561 .
13840) (RemoveErrorCases 13844 . 14397) (RepInv 14401 . 14637) (RestoreRules 14641 . 15036) (
SOMEGensym 15040 . 15893) (SemanticRules 15897 . 15999) (SimpAnd 16003 . 16133) (SimpImp 16137 . 16586
) (SimpNot 16590 . 16856) (Split 16860 . 17705) (Subgoal 17709 . 18201) (Translate1 18205 . 18482) (
TryAllChainingsAndNarrowings 18486 . 19246) (TryToProve 19250 . 19712) (TypeOf 19716 . 19948) (
UpdateTimer 19952 . 20469) (UseEqs 20473 . 22031) (annotate 22035 . 22581) (apply 22585 . 22642) (
assume 22646 . 22804) (clear 22808 . 23269) (copy 23273 . 23488) (declareFun 23492 . 23898) (define
23902 . 24047) (dtvs 24051 . 24096) (equus 24100 . 25730) (fix 25734 . 25790) (implementation 25794 .
26189) (induct 26193 . 26704) (infix 26708 . 26837) (nextis 26841 . 27373) (normalize 27377 . 27541) (
print1 27545 . 31336) (put 31340 . 31514) (quit 31518 . 31821) (readNodeExpression 31825 . 32420) (
recheck 32424 . 32948) (reclaim 32952 . 35104) (rule 35108 . 35254) (saveproof 35258 . 35687) (schema
```