COMMENT'THIS FILE GENERALIZES THE TERM ABOUT TO BE PROVED BY INDUCTION.
WE GENERALIZE ON THE COMMON SUBTERMS ON EITHER SIDE
OF "EQUAL", "IMPLIES" AND "OR" STMTS.';

COMMENT'FIND ALL COMMON NON-ATOMIC NON-PRIMITIVE SUBTERMS OF TWO TERMS.';

VARS T2 GENRLTLIST ATOMLIST;

FUNCTION COMSUBT1 T1;
VARS X;
IF ATOM(T1)
   THEN
      OCCUR(T1,T2);
   ELSE
      TL(T1)->X;
      IF (1; LOOPIF X/=NIL THEN LOGAND(COMSUBT1(HD(X));TL(X)->X;CLOSE;) 
          THEN
              IF NOT(LISPPRIM(T1)) AND OCCUR(T1,T2) 
                  THEN
                      IF NOT(MEMBERQ(T1,GENRLTLIST)) 
                          THEN T1:=GENRLTLIST->GENRLTLIST;CLOSE; 
                          1;EXIT;
                      CLOSE;
              0; 
          CLOSE;
   END;

FUNCTION COMSUBTERMS T1 T2;
IF CONSCNT(T1)>CONSCNT(T2) THEN T1;T2->T1->T2;CLOSE;
ERASE(COMSUBT1(T1));
END;

COMMENT'FIND ALL COMMON SUBTERMS OCCURRING ACROSS EQS AND
IMPLIES AND ORS.';

FUNCTION GENRLT1 TERM;
IF ATOM(TERM) THEN EXIT;
IF HD(TERM)="EQUAL"
   THEN
       COMSUBTERMS(HD(TL(TERM)),HD(TL(TL(TERM))));
ELSEIF HD(TERM)="IF"
   THEN
       IF ATOM(HD(TL(TERM))) THEN
       ELSEIF HD(TL(TL(TERM)))=T
           THEN
               APPLIST(TL(HD(TL(TERM))),
                       LAMBDA TERM1;
COMSUBTERMS(TERM1,HD(TL(TL(TL(TL(TERM))))));
ELSEIF HD(TL(TL(TL(TL(TERM)))))=T THEN
APPLIST(TL(HD(TL(TL(TERM))))),
LAMBDA TERM1;
COMSUBTERMS(TERM1,HD(TL(TL(TL(TL(TERM))))));
CLOSE;
CLOSE;
APPLIST(TL(TERM),GENRLT1); END;

FUNCTION GENRLTERMS;
VARS GENRLTL1ST;
NIL->GENRLTL1ST;
GENRLT1();
GENRLTL1ST;
END;

COMMENT'THIS FUNCTION MAKES A VERBOSE REPORT ON THE PROGRESS OF GENERALIZATION.';

FUNCTION GENREPORT;
IF VERBOSE THEN
POPPTON();
NL(2);
PRSEQAND(4,'GENERALIZE COMMON SUBTERMS BY REPLACING ', SUBSTLIST,LAMBDA P;PR(FRONT(P));PRSTRING(' BY ');PR(BACK(P));END);
NL(2);
PRSTRING('THE GENERALIZED TERM IS:');
NL(2);
PPR(TERM);
NL(2);
CLOSE;
END;

COMMENT'THIS IS THE TOP-LEVEL FUNCTION. IT GENERALIZES ITS ARGUMENT AS DESCRIBED, AND THEN PRINTS A VERBOSE COMMENT IF NEEDED. NOTE THAT IF THE TERM GENERALIZED IS NUMERIC, A NUMERIC SKOLEM CONSTANT IS GENERATED FOR IT.';

FUNCTION GENERALIZE TERM;
VARS X SUBSTLIST;
GENRLTERMS(TERM)->X;
IF X=NIL THEN TERM;EXIT;
MAPLIST(X, LAMBDA TERM;
IF NUMERIC(TERM) THEN "INTGR"; ELSE "XLIST";CLOSE;
GENSKO()=>(X;
CONSPAIR(TERM,X);
END)->SUBSTLIST;
IF SUBSTLIST=NIL THEN TERM;EXIT;
APP SUBST(SUBSTLIST,TERM) =TERM;
(REPORT(["G",APPLIST(SUBSTLIST,BACK)],GENREPORT,"GENERALIZE"));
APPLIST(SUBSTLIST,
    LAMBDA X;
    FRONT(X);BACK(X)->FRONT(X)->BACK(X);
    END);
SUBSTLIST<>GENRLALIST->GENRLALIST;
TERM;
END;
FUNCTION FERTREPORT;
  IF VERHOSE THEN POPSTOP();
  NL(4);PRSTRING('FERTILIZE WITH ');
  PRIND(TERM1,15,1);
  PRSTRING( '.' );NL(2);
  PRIND(9,7);
  CLOSE;
END;

FUNCTION FERTILIZE TERM;
  VARS TERM1 TERM2 TERM3 LHS1 RHS1;
  IF SHD(TERM)/="IF" THEN 0;EXIT;
  HD(TL(TERM))->TERM1;
  HD(TL(TL(TERM)))->TERM2;
  HD(TL(TL(TL(TERM))))->TERM3;
  COMMENT 'LOOK FOR TERMS OF THE FORM (IF (EQUAL LHS RHS) BOOL1 BOOL2)
  WHERE BOOL2 IS NOT NIL. IF FOUND, FERTILIZE LHS=RHS INTO BOOL1 AND HIDE IT.';

  IF SHD(TERM1)="EQUAL" AND TERM3/=NIL AND TERM3 AND BOOLEAN(TERM2) AND BOOLEAN(TERM3)
  THEN HD(TL(TERM1))->LHS1;
  HD(TL(TL(TERM1)))->RHS1;
  IF ISSPEC(LHS1) OR ISSPEC(RHS1)
  THEN GOTO NOFERT;CLOSE;
  IF FERTILIZE(TERM2) THEN TERM2;
  IF FERTILIZE(TERM2) THEN TERM2;CLOSE;
  ["IF",TERM2,T,GENSTAR(["IF",TERM1,NIL,TERM3])]
  ->TERM2;
  REPORT(["F",STARCOUNT%],FERTREPORT,"FERTILIZE");
  IF FERTILIZE(TERM3) THEN TERM3;CLOSE;
  IF TERM3=T
  THEN TERM2;
  ELSE
    ["IF",TERM2,
    ["IF",TERM3,
    T,
    TERM1%],
    NIL%];
  CLOSE;
  1;
  EXIT;
  CLOSE;
NOFERT;
  0->TERM1;
  ["IF",APPLIST(TL(TERM),
  LAMBDA TERM;
IF FERTILIZE(TERM) THEN 1->TERM1; ELSE TERM; CLOSE;
END),
( IF TERM1 THEN ELSE
   ERASE(ERASE(),ERASE(),ERASE(),ERASE());0;EXIT)]%;
1;
END;

FUNCTION FERTIL1 TERM;
VARS LHS2 RHS2;
IF ATOM(TERM) THEN 0;EXIT;
IF HD(TERM)="EQUAL" THEN
   HD(TL(TERM))->LHS2;
   HD(TL(TL(TERM)))->RHS2;
   COMMENT 'NOW LOOK FOR (THE BEST) CROSS FERTILIZATION';
   IF OCCUR(RHS1,RHS2) THEN
      IF OCCUR(LHS1,LHS2) THEN
         IF CONSCNT(RHS1)<CONSCNT(LHS1) THEN
            SUBST(RHS1,LHS1,LHS2)->LHS2;
            ELSE SUBST(LHS1,RHS1,RHS2)->RHS2;CLOSE;
            ELSE SUBST(LHS1,RHS1,RHS2)->RHS2;CLOSE;
            END;
         ELSE
            IF OCCUR(LHS1,LHS2) THEN
               SUBST(RHS1,LHS1,LHS2)->LHS2;
               ELSE GOTO MASSSUBST;EXIT;
               END;
         END;
      ELSE L="EQUAL",LHS2,RHS2%;
      1;
      EXIT;
   END;
   COMMENT 'IF TERM IS AN IF, LOOK FOR ITS "CORE" AND FERTILIZE IT'
   IF HD(TERM)="IF" THEN
      COMMENT '(IF X CORE T) => (IF X FERT(CORE) T), PROVIDED
X DOES NOT CONTAIN LHSl OR RHSl.';
      IF HD(TL(TL(TL(TERM))))=T THEN
         IF OCCUR(LHS1,HD(TL(TERM))) THEN GOTO CHKFRHSOCC;
         ELSEIF OCCUR(RHS1,HD(TL(TERM))) THEN GOTO SUBSTLR;
         ELSEIF FERTIL1(HD(TL(TL(TL(TERM)))))) THEN
            ->FO01;
            [%"IF",HD(TL(TERM)),FO01,T%];
            1;
            EXIT;
         END;
      COMMENT'(IF CORE T (*N)) => (IF FERT(CORE) T (*N));
      ELSEIF HD(TL(TL(TERM)))=T AND SHD(HD(TL(TL(TL(TERM))))))="*
      THEN
         IF FERTIL1(HD(TL(TERM))) THEN
            ->FO01;
            [%"IF",FO01,T,HD(TL(TL(TERM))))%];
            1;
            EXIT;
         END;
      END;
   END;
   COMMENT 'IF NOT OF EITHER OF THE ABOVE FORMS, FALL THROUGH
TO MASSIVE SUBSTITUTION.';
   CLOSE;
COMMENT 'IF CROSS FERTILIZATION NOT POSSIBLE, TRY MASSIVE SUBSTITUTION';

MASSSUBST:
  IF OCCUR(LHS1,TERM)
  THEN
    CHKRHSOCC:
      IF OCCUR(RHS1,TERM)
      THEN
        IF CONSCNT(RHS1)<CONSCNT(LHS1)
        THEN
          SUBST(RHS1,LHS1,TERM);
        ELSE
          SUBST(LHS1,RHS1,TERM);CLOSE;
        ELSEIF OCCUR(RHS1,TERM)
        THEN
          SUBST(LHS1, RHS1, TERM);
        ELSE
          0; EXIT;
      END;
  END;

1; EXIT;

END;
COMMENT'THIS IS THE NORMALIZE FUNCTION. IN-LINE COMMENTS EXPLAIN
THE REWRITE RULES APPLIED;'

VARS REWRITEFN;
IDENTFN->REWRITEFN;

FUNCTION REWRITE TERM;
VARS TERM1 TERM2 TERM3;
COMMENT'IF TERM IS AN EQUALITY';

IF HD(TERM)="EQUAL" THEN
    HD(TL(TERM))->TERM1;
    HD(TL(TL(TERM)))->TERM2;

COMMENT'(EQUAL KNOWN1 KNOWN2) => T OR NIL';
    IF TERM1=TERM2 THEN T;EXIT;
    IF NOTIDENT THEN NIL;EXIT;

COMMENT'(EQUAL BOOL T) => BOOL';
    IF TERM1=T AND BOOLEAN(TERM2) THEN TERM2 EXIT;
    IF TERM2=T AND BOOLEAN(TERM1) THEN TERM1 EXIT;

COMMENT'(EQUAL (EQUAL A B) C) =>
    (IF (EQUAL A B) (EQUAL C T) (IF C NIL T))';
    IF SHD(TERM1)="EQUAL" OR SHD(TERM2)="EQUAL" AND (SWAP;1) THEN
        [%"IF",TERM1,
        REWRITE([%"EQUAL",TERM2,T%]),
        REWRITE([%"IF",TERM2,NIL,T%])]->TERM;
        GOTO CONDL;
    CLOSE;

COMMENT'(EQUAL X NIL) => (IF X NIL T)';
    IF TERM1=NIL OR TERM2=NIL AND (SWAP;1) THEN
        [%"IF",TERM2,NIL,T%]->TERM;
        GOTO CONDL;
    CLOSE;

COMMENT'GO SEE IF ONE ARG IS A IF';
    GOTO CONDARG;

COMMENT'TERM IS A IF';

ELSEIF HD(TERM)="IF" THEN
    CONDL:
    TL(TERM)->TERM3;
HD(TERM3)->TERM1;
TL(TERM3)->TERM3;
HD(TERM3)->TERM2;
HD(TL(TERM3))->TERM3;

COMMENT'(IF KNOWN X Y) => X OR Y;
  IF TERM1=Nil THEN TERM3;EXIT;
  IF NOTIDENT THEN TERM2;EXIT;

COMMENT'(IF X Y Y) => Y;
  IF TERM2=TERM3 THEN TERM2;EXIT;

COMMENT'(IF X X NIL) => X;
  IF TERM1=TERM2 AND TERM3=NIL THEN TERM1;EXIT;

COMMENT'(IF BOOL T NIL) => BOOL;
  IF BOOLEAN(TERM1) AND TERM2=T AND TERM3=NIL
  THEN TERM1;EXIT;

COMMENT'(IF X T (IF Y NIL T)) => (IF Y (IF X T NIL) T);
  IF TERM2=T AND SHD(TERM3)="IF" AND
  HD(TL(TL(TERM3)))=NIT AND HD(TL(TL(TL(TERM3))))=T
  THEN
  IF BOOLEAN(TERM1)
    THEN TERM1;
    ELSE [%"IF",TERM1,T,NIL%]CLOSE;
    ->TERM2;
    HD(TL(TERM3))->TERM1;
    T->TERM3;
  [%"IF",TERM1,TERM2,TERM3%]->TERM;
  CLOSE;

COMMENT'IF TERM1 IS AN IF, DECIDE IF IT SHOULD BE DISTRIBUTED.';
  IF SHD(TERM1)="IF" THEN

COMMENT'(IF (IF A T2 T3) B C) => (IF A (IF T2 B C)
  (IF T3 B C)) WHERE T2 OR T3 ISNIL';

IF HD(TL(TL(TERM1)))=NIT OR HD(TL(TL(TL(TERM1))))=NIT
  THEN
  GOTO CONDCOND;
  CLOSE;

COMMENT'(IF (IF A T (* N)) T (* M)) => (IF A T (* N M))';
  IF TERM2=T AND SHD(TERM3)="*" AND HD(TL(TL(TERM1)))=T
  AND SHD(HD(TL(TL(TL(TERM1)))))="*
  THEN
  [%"IF",HD(TL(TERM1)),T,"*":(TL(HD(TL(TL(TL(TERM1))))))
  <>TL(TERM3)%];
  EXIT;

COMMENT'(IF (IF A B C) D E)=>(IF A (IF B C E) (IF C D E))
  WHERE D AND E ARE NOT NIL OR D AND E ARE T AND NIL';

IF TERM2=NIT AND TERM3=T THEN GOTO SKIP;
ELSEIF TERM3==NIL AND TERM2/=T THEN GOTO SKIP;CLOSE;
CONDCOND:
IF SHD(TERM2)="*" OR SHD(TERM3)="*" THEN GOTO SKIP;CLOSE;
REWRITE(["IF",HD(TL(TL(TERM1)))),TERM2,TERM3]);
REWRITE(["IF",HD(TL(TL(TERM1)))),TERM2,TERM3]);
->TERM3->TERM2;
["IF",HD(TL(TERM1))),TERM2,TERM3]->TERM;
GOTO CONDL;
SKIP:
CLOSE;
COMMENT 'TERM IS A NON-IF, NON-EQ FUNCTION CALL';
ELSE
COMMENT ' (FOO X (IF A B C) Y) => (IF A (FOO X B Y) (FOO X C Y))';

CONDARG:
TL(TERM)->TERM1;
LOOP IF TERM1/=NIL AND SHD(HD(TERM1))/="IF"
THEN
TL(TERM1)->TERM1;
CLOSE;
IF TERM1/=NIL
THEN
HD(TERM1)->TERM1;
["IF",HD(TL(TERM1)),REWRITE(SUBST(HD(TL(TL(TERM1)))),TERM1,TERM)),REWRITE(SUBST(HD(TL(TL(TL(TERM1))))),TERM1,TERM))]->TERM;
GOTO CONDL;
CLOSE;
REWRITEFN();
TERM
END

FUNCTION NORMALIZE TERM;
IF ATOM(TERM) THEN TERM EXIT;
REWRITE(HD(TERM)::MAPLIST(TL(TERM),NORMALIZE));
END
VARS SLASH9 SLASH22 SLASH36;

DTRACK(9);
APPLIST(SLASH9,DCOMP);
DTRACK(22);
APPLIST(SLASH22,DCOMP);

DTRACK(36);
APPLIST(SLASH36,DCOMP);
APPFILE([/DEFS],DEFINE);
COMMENT 'THIS IS THE REDUCE FUNCTION. IN-LINE COMMENTS EXPLAIN THE REWRITE RULES APPLIED.';

VARS REDUCE;

FUNCTION REDUCE1 TERM CONSULIST;
VARS TERM1 TERM2 TERM3;
RECURSE:
COMMENT 'IF TERM IS ATOM OR NON-IF, QUIT';
IF ATOM(TERM) OR HD(TERM) /= "IF"
THEN
TERM;
EXIT;

COMMENT 'GET COMPONENTS OF THE IF';
HD(TL(TERM)) -> TERM1;
HD(TL(TL(TERM))) -> TERM2;
HD(TL(TL(TL(TERM)))) -> TERM3;

COMMENT 'IF TERM1 IS NIL OR CONS, EVAL IT';
IF TERM1 == NIL
THEN
TERM3 -> TERM;
GOTO RECURSE;
ELSEIF EXPLCONS(TERM1) OR MEMBERID(TERM1, CONSULIST)
THEN
TERM2 -> TERM;
GOTO RECURSE;
CLOSE;

COMMENT '(IF ATOM A B) => (IF ATOM R(A(ATOM/CONS)) R(B(ATOM/NIL)))';
IF ATOM(TERM1)
THEN
GOTO SUBSTCONS;
CLOSE;

COMMENT '(IF (EQUAL A SPECLIST) B C) => (IF (EQUAL A SPECLIST)
R(B(A/SPECLIST))
R(C((EQUAL A SPECLIST) / NIL)))';
IF HD(TERM1) == "EQUAL"
THEN
IF ISSPEC(HD(TL(TERM1)))
THEN SUBST(HD(TL(TERM1)), HD(TL(TL(TERM1))), TERM2) -> TERM2;
ELSEIF ISSPEC(HD(TL(TL(TERM1))))
THEN SUBST(HD(TL(TL(TERM1))), HD(TL(TL(TERM1))), TERM2) -> TERM2;
ELSE GOTO SUBSTTRUE; CLOSE;
GOTO ASSEMBOOL;
CLOSE;

COMMENT '(IF (IF ...) A B) => (IF (IF) R(A R(B)))';
IF HD(TERM1) == "IF"
THEN
REDUCE1(TERM1, CONSLIST) -> TERM1;
REDUCE1(TERM2, CONSLIST) -> TERM2;
REDUCE1(TERM3, CONSLIST) -> TERM3;
IF TERM3 = = NIL THEN GOTO CONTINUE; CLOSE;
["IF", TERM1, TERM2, TERM3%];
EXIT;

CONTINUE:

COMMENT' (IF BOOL A B) => (IF BOOL R(A(BOOL/T)) R(B(BOOL/NIL)))';
IF BOOLEAN(TERM1)
THEN
SUBST TRUE:
SUBST(T, TERM1, TERM2) -> TERM2;
ASSEMBLE:
["IF", TERM1,
 REDUCE1(TERM2, CONSLIST),
 REDUCE1(SUBST(NIL, TERM1, TERM3), CONSLIST)%];
EXIT;

COMMENT' (IF RANDOM A B) => (IF RANDOM R(A(RANDOM/CONS))
 R(B(RANDOM/NIL))');
SUBST CONS:
["IF", TERM1, REDUCE1(TERM2, TERM1::CONSLIST),
 REDUCE1(SUBST(NIL, TERM1, TERM3), CONSLIST)%];
END;

REDUCE1(%NIL%) -> REDUCE;
FUNCTION INDUCTABLE TERM;
VARS X;
LOOP IF ISLINK(TERM) THEN HD(TERM) -> X;
IF X != "CDR" AND X != "CAR" THEN 0; EXIT;
HD(TL(TERM)) -> TERM;
CLOSE;
TERM; 1;
END;

FUNCTION GETARG TERM;
LOOP IF ISLINK(TERM) THEN HD(TL(TERM)) -> TERM; CLOSE;
TERM; END;

FUNCTION INDUCTSweep;
[%APPLIST(TOPLEXP, LAMBDA X;
VARS BOMBLIST OTHERFAILS HDTERM;
NIL->BOMBLIST;
NIL->OTHERFAILS;
HD(HD(X)) -> HDTERM;
INDSW1(HD(TL(X)));
IF BOMBLIST /= NIL THEN [%BOMBLIST, OTHERFAILS%]; CLOSE;
END)%];

FUNCTION INDSW1 TERM1;
IF ATOM(TERM1) THEN EXIT;
IF ISINTER(TERM1) THEN TERM1::OTHERFAILS -> OTHERFAILS;
ELSE IF HD(TERM1) = HDTERM THEN [%APPLIST(TL(TERM1), LAMBDA TERM2; IF ISINTER(TERM2) THEN TERM2; CLOSE; END)%]; -> F001;
IF F001 /= NIL THEN F001::BOMBLIST -> BOMBLIST; CLOSE EXIT;
FUNCTION ISINTER TERM;
IF ATOM(TERM) THEN 0;EXIT;
HD(TERM)->TERM;
IF TERM="CDR"
   THEN 1; ELSE TERM="CAR";CLOSE;
END;

COMMENT'THE FOLLOWING FUNCTION TRANSFORMS FAULT DESCRIPTIONS INTO FOUR TUPLES TO MAKE IT EASIER TO SORT THROUGH THEM TO FIND WHAT TO INDUCT UPON. IT THROWS OUT ANY REQUIRING INDUCTION ON NON SKOLEM CONSTANTS.';

FUNCTION TRANSFAULT FAULTDESC;
VARS ARGLIST X;
NIL->ARGLIST;
XAPPLIST(HD(FAULTDESC),
   LAMBDA POCKET;
   XAPPLIST(POCKET,
      LAMBDA TERM;
      IF INDUCTABLE(TERM) THEN
         ->X;
      IF MEMBER(X,ARGLIST) THEN ELSE X::ARGLIST->ARGLIST;CLOSE;
      ELSE 1->XAPPFLAG;CLOSE;
   END);
END);
IF XAPPFLAG THEN EXIT;
[%1,ARGLIST,HD(FAULTDESC),
   [%APPLIST(HD(TL(FAULTDESC))),LAMBDA TERM;
      IF INDUCTABLE(TERM) AND MEMBER((),ARGLIST) THEN TERM;CLOSE;
      END]%]%];
END;

COMMENT'THE FIRST COMPONENT ABOVE WILL BE USED TO SCORE THE CANDIDATES';

FUNCTION GETCANDS FAULTLIST;
[%APPLIST(FAULTLIST,TRANSFAULT)%];
END;

FUNCTION MERGECANDS CANDLIST;
VARS CAND1;
CANDLIST;
LOOP IF TL(CANDLIST)/=NIL
THEN
   HD(CANDLIST)->CAND1;
   TL(CANDLIST)->CANDLIST;
   XAPPLIST(CANDLIST,
      LAMBDA CAND2;
      IF INTERSECTP(HD(TL(CAND1)),HD(TL(CAND2)),NONOP=)
      THEN
         1->XAPPFLAG;
         UNION(HD(TL(CAND1)),HD(TL(CAND2)),NONOP=)->HD(TL(CAND2));
         UNION(HD(TL(TL(CAND1))),HD(TL(TL(CAND2))),EQUAL)->HD(TL(TL(CAND2)));
      UNION(HD(TL(TL(TL(CAND1)))),HD(TL(TL(TL(CAND2)))),EQUAL)->
         HD(TL(TL(TL(CAND2))));
         HD(CAND2)+HD(CAND1)->HD(CAND2);
FUNCTION CHOOSEHIGH CANDLIST;
VARS HIGH ANS;
-10000->HIGH;
IF TL(CANDLIST)=NIL THEN CANDLIST;EXIT;
LOOP IF CANDLIST/=NIL THEN
IF HD(HD(CANDLIST))>HIGH AND HD(HD(CANDLIST))
THEN
HD(HD(CANDLIST))->HIGH;
HD(CANDLIST):=NIL->ANS;
ELSEIF HD(HD(CANDLIST))=HIGH THEN
HD(CANDLIST):=ANS->ANS;
CLOSE;
TL(CANDLIST)->CANDLIST;
CLOSE;
ANS;
END;

FUNCTION CHOOSENEW CANDLIST;
APPLIST(CANDLIST,
LAMBDA CAND;
1->HD(CAND);
APPLIST(HD(TL(CAND)),
LAMBDA TERM;
IF NOT(MEMBER(TERM,INDLIST))
THEN 1+HD(CAND)->HD(CAND);
CLOSE;
END);
CHOOSEHIGH(CANDLIST);
END;

COMMENT 'THE FUNCTION BELOW MERGES ALL RECURSIVE POCKETS WHICH
HAVE NON-NIL INTERSECTIONS.';

FUNCTION MERGEPOCKETS POCKETLIST;
IF POCKETLIST=NIL THEN NIL;
ELSE ADDPOCKET(HD(POCKETLIST),MERGEPOCKETS(TL(POCKETLIST)));CLOSE;
END;

FUNCTION ADDPOCKET POC POCLIST;
IF POCLIST=NIL THEN [%POC%];
ELSEIF INTSECTP(POC,HD(POCLIST),EQUAL)
THEN UNION(POC,HD(POCLIST),EQUAL):=TL(POCLIST);
ELSE HD(POCLIST):=ADDPOCKET(POC,TL(POCLIST));CLOSE;
END;

COMMENT 'A POCKET IS SUBSUMED BY ANOTHER IF ALL OF ITS
TERMS OCCUR AS SUBTERMS IN ANY TERM IN THE OTHER.

FUNCTION SUBSUMED POCKET1 POCKET2;
VARS TERM1;
LOOP IF POCKET1/=NIL THEN
   HD(POCKET1)->TERM1;
   IF (XAPPLIST(POCKET2,
     LAMBDA TERM2;OCCUR(TERM1,TERM2)->XAPPFLAG;END);XAPPFLAG)
     THEN ELSE 0;EXIT;
   TL(POCKET1)->POCKET1;
   CLOSE;
1;
END;

COMMENT'THIS FUNCTION TRANSFORMS A LIST OF POCKETS INTO
A LIST OF POCKETS THAT IS SUBSUMPTION FREE.

FUNCTION SUBSUME POCLIST;
[XAPPLIST(POCLIST,
  LAMBDA POCKET1;
  XAPPLIST(POCLIST,
    LAMBDA POCKET2;
    IF POCKET1=POCKET2 THEN EXIT;
    SUBSUMED(POCKET1,POCKET2)->XAPPFLAG;END);
    IF XAPPFLAG THEN ELSE POCKET1;CLOSE;
    END)];
END:

COMMENT'THE FOLLOWING SUBSTITUTION IS USED TO REPLACE
CAR, AND CDRS OCCURING EXPLICITLY IN THE
THEOREM BY DUMMY SYMBOLS TO AVOID CONFUSING THEM WITH
RECURSIVE ONES IN THE EXPANDED FN DEFS.

[CONSPAIR("CDR","DUMMYCDR"),
  CONSPAIR("CAR","DUMMYCAR")%] ->DUMMYSUBST;

FUNCTION PICKINDCONST S INTERM;
VARS CANLIST;
1->ININDUCT;
ERASE(SYMEVAL(APPSUBST(DUMMYSUBST,INDTERM)));
0->ININDUCT;
GETCANDS(INDUCTSweep())->CANDLIST;
IF CANDLIST=NIL THEN 0;EXIT;
MERGECANDS(CANDLIST)->CANDLIST;
CHOOSEHIGH(CANDLIST)->CANDLIST;
IF TL(CANDLIST)/=NIL THEN
   THEN
   CHOOSENEW(CANDLIST)->CANDLIST;
   CLOSE;
   HD(CANDLIST)->CANDLIST;
   HD(TL(TL(CANDLIST))));
SUBSUME(MERGEPOCKETS(HD(TL(TL(CANDLIST)))));
HD(TL(CANDLIST));
1;
END;
COMMENT'THIS IS THE FILE WHICH CONSTRUCTS THE INDUCTION FORMULA AND LINKS THE INDUCTION PKG WITH THE REST OF THE THEOREM PROVER.';

COMMENT'THE FOLLOWING FUNCTION IS USED TO PROCESS THE POCKET LIST AND FAILURES LIST RETURNED BY PICKINDCONST, BEFORE THE INDUCTION FORMULA IS ACTUALLY CONSTRUCTED. THIS FUNCTION CREATES ALIISTS OF THE FORM:

(INDUCTION CONSTANT . LIST OF DESTRUCTORS APPLIED).

IT RETURNS TWO SUCH ALIISTS, ONE CORRESPONDING TO JUST THE RECURSIVE DESTRUCTIONS, AND THE OTHER TO BOTH RECURSIVE AND NON-RECURSIVE ONES.';

FUNCTION GENDRALISTS RECPOCKETS FAILURES;
VARS F X Y DESTALIST L1 L2;
NIL->DESTALIST;
LAMBDA L1;
APPLIST(L1,
  LAMBDA TERM;
  GETARG(TERM)->X;
  IF ASSOC(X,DESTALIST)
    THEN
      ->Y;
      IF MEMBEREQUAL(TERM,BACK(Y))
        THEN ELSE TERM::BACK(Y)->BACK(Y);CLOSE;
      ELSE CONSPAIR(X,[%TERM%])::DESTALIST->DESTALIST;CLOSE;
    END);
END->F;
APPLIST(RECPOCKETS,F);
COPYLIST(DESTALIST);
APPLIST(FAILURES::NIL,F);
DESTALIST;
END;
COMMENT'(USE OF MEMBEREQUAL RATHER THAN MEMBERID IS OK HERE SINCE TERM IS JUST A COLLECTION OF CARS AND CDRS APPLIED TO A SKOLEM CONSTANTS, AND HENCE IS IDENT IFF EQUAL.)';

COMMENT'IMPORTANT NOTE: THIS INDUCTION ROUTINE KNOWS ABOUT NUMBERS. THE FOLLOWING IS ASSUMED: THE EVAL ROUTINE KNOWS THAT THE CAR OF A NUMERIC SKOLEM CONSTANT IS NIL. THIS GUARANTEES THAT NO "CAR" TERMS WILL OCCUR IN THE LIST OF DESTRUCTORS OF A NUMERIC SKOLEM CONSTANT TO BE INDUCTED UPON.';

COMMENT'THE FOLLOWING ROUTINES KNOW ABOUT NUMBERS:
STEPFOR (GENERATES A NEST OF ADD1S AS DEEP AS THE DEEPEST CDR NEST AROUND A NUMERIC SKOLEM CONST),
BASESFOR (GENERATES A LIST OF THE NUMBERS BETWEEN 0 AND N INSTEAD OF THE CORRESPONDING CONSES, AND GETCOMP (WHICH IS USED RATHER THAN EVAL BECAUSE OF THE PRESENCE OF "ADD1"S IN THE CONCLUSION). AS OF THIS WRITING, NO OTHER ROUTINES ARE AFFECTED (INFECTED).'
COMMENT \[NOW ON TO INDUCTION. THE FIRST SET OF FUNCTIONS IS CONCERNED WITH GROWING THE LEAST STRUCTURED TERM ALLOWING EACH Destructor COMBINATION TO FULLY OPERATE ON IT. FOLLOWING THIS IS A SET OF FUNCTIONS WHICH CONSTRUCT ALL OF THE BASES THAT MUST BE ALLOWED, GIVEN THE TERM GROWN ABOVE.\]

FUNCTION GROW TERM;
VARS Y;
IF ATOM(TERM) THEN MUNG;EXIT;
GROW(HD(TL(TERM)))->Y;
IF HD(TERM)="CAR"
THEN
IF HD(Y)="CONS"
THEN IF ATOM(HD(TL(Y))) THEN TL(Y); ELSE HD(TL(Y));CLOSE;
ELSE [%"CONS",GENSKO(CONST),HD(Y)%]->HD(Y);
   TL(HD(Y));
   CLOSE;
ELSE
IF HD(Y)="CONS"
THEN IF ATOM(HD(TL(TL(Y)))) THEN TL(TL(Y)); ELSE HD(TL(TL(Y)));CLOSE;
ELSE [%"CONS",GENSKO(CONST),HD(Y)%]->HD(Y);
   TL(TL(HD(Y)));
   CLOSE;
END;

COMMENT \[THIS FUNCTION TAKES A SKOLEM CONSTANT AND A LIST OF DESTRUCTORS APPLIED TO IT, AND CONSTRUCTS THE LEAST STRUCTURED TERM ALLOWING EACH DESTRUCTOR TO OPERATE.\]

[%CONSPAIR("CDR","ADD1")%]->CDRTOADD1;

FUNCTION STEPFOR CONST TERMLIST;
VARS TERM X;
IF NUMSKO(CONST)
THEN
   HD(TERMLIST)->TERM;
   CONSCNT(TERM)->X;
   LOOPIF (TL(TERMLIST)->TERMLIST;TERMLIST/=NIL)
      THEN
         IF CONSCNT(HD(TERMLIST))>X
            THEN HD(TERMLIST)->TERM;CONSCNT(TERM)->X;CLOSE;
            CLOSE;
         APPSUBST(CDRTOADD1,TERM);
      EXIT;
   [%"CONS",GENSKO(CONST),CONST%]->SEED;
   LOOPIF TERMLIST/=NIL
      THEN
         SEED->MUNG;
         ERASE(GROW(HD(TERMLIST)));
         TL(TERMLIST)->TERMLIST;
         CLOSE;
      SEED;
   END;

END;
FUNCTION SMALLER TERM:
IF ATOM(TERM) THEN NIL; EXIT;
NIL::TL([%APPLIST(HD(TL(TERM))::SMALLER(HD(TL(TERM)))]),
    LAMBDA ARG1;
  APPLIST(HD(TL(TL(TERM)))::SMALLER(HD(TL(TL(TERM)))]),
    LAMBDA ARG2;
  [%"CONS", ARG1, ARG2%];
END);
END;

FUNCTION BASESFOR CONST TERM;
IF NUMSKO(CONST)
    THEN 0->F001;
    [% LOOPIF ISLINK(TERM)
        THEN F001; F001+1->F001; HD(TL(TERM))->TERM; CLOSE%];
    EXIT;
SMALLER(TERM);
END;

FUNCTION CONJOIN L;
IF TL(L)=NIL THEN HD(L);
    ELSE [%"AND", HD(L), CONJOIN(TL(L))%]; CLOSE;
END;

FUNCTION GETCOMP TERM;
IF ATOM(TERM)
    THEN TERM; BACK(ERASE(ASSOC(TERM, STEPALIST))); EXIT;
GETCOMP(HD(TL(TERM)))->F001;
IF HD(TERM)="CAR" OR HD(F001)="ADD1"
    THEN HD(TL(F001));
    ELSE HD(TL(TL(F001)))); CLOSE;
END;

COMMENT'THIS IS THE FUNCTION WHICH CONSTRUCTS THE INDUCTION
FORMULA. FIRST IT SETS UP THE STEPALIST, A LIST OF THE
THINGS INDUCTED UPON, PAIRED WITH THE TERM TO REPLACE THEM
IN THE CONCLUSION. THIS TERM IS THE LEAST STRUCTURED TERM
WHICH ALLOWS ALL THE DESTRUCTORS TO FULLY OPERATE ON IT.
THEN IT SETS UP THE HYPALISTLIST, WHICH IS A LIST OF ALISTS;
EACH ALIST PAIRS A CONST TO BE INDUCTED UPON WITH WHAT IT
IS TO BE REPLACED BY IN THE HYPOTHESIS. THIS IS GENERATED
BY APPLYING THE RECURSIVE DESTRUCTORS TO THE LEAST
STRUCTURED TERM DESCRIBED ABOVE. THERE IS SUCH AN ALIST FOR
EACH RECURSIVE POCKET.

COMMENT NEXT, IT SETS UP THE BASES LIST,
WHICH IS THE LIST OF ALL THE BASES TO BE ESTABLISHED. THESE
ARE JUST THE THEOREM INSTANTIATED TO ALL THE TERMS
SMALLER THAN THE ONE IN THE CONCLUSION, FOR EACH INDUCTION CONST.
FINALLY, IT SETS UP THE HYPALIST, WHICH IS A LIST OF
ALL THE HYPOTHESES, ONE FOR EACH ALIST ON THE HYPALISTLIST.

COMMENT ONCE ALL THIS IS DONE, IT CONSTRUCTS THE FORMULA IN
THE-obvious WAY.

FUNCTION INDFORMULA RECP0CKETS DESTALIST INDTERM;
VARS ALIST;
[APPLIST(DESTALIST,
  LAMBDA X;
  CONSPAIR(FRONT(X),STEPFOR(FRONT(X),BACK(X)));END)];
  STEPALIST;

[APPLIST(RECP0CKETS,
  LAMBDA POCKET;
  APPLIST(POCKET,
    LAMBDA TERM;
    CONSPAIR(GETCOMP(TERM));
  END)];
  HYPALISTLIST;

[APPLIST(STEPALIST,
  LAMBDA X;
  FRONT(X)->CONST;
  APPLIST(BASESFOR(CONST,BACK(X)),
    LAMBDA TERM;SUBST(TERM,CONST,INDTERM);END)];
  BASES;

[APPLIST(HYPALISTLIST,
  LAMBDA ALIST;
  APPSUBST(ALIST,INDTERM);
  END)];
  HYPLIST;

["AND",CONJOIN(BASES),
 ["IMPLIES",CONJOIN(HYPLIST),
  APPSUBST(STEPALIST,INDTERM)]%]
END;

FUNCTION INDREPORT;
IF VERBOSE
THEN
POPTTON();
PRSEQAND(4,'INDUCT ON \',INDCONSTS,PR);
FUNCTION INDUCT INTERM;
IF NOT(PICKINDCONSTS(INTERM)) THEN 0;EXIT;
->INDCONSTS;
->RECPOCKETS;
->OTHERFAILS;
GENDRALS(RECPOCKETS,OTHERFAILS)->DESTALIST->RECALIST;
INDCONSTS<->INDLIST->INDLIST;
INDFORMULA(RECPOCKETS,DESTALIST,INDTERM);
REPORT("I":INDCONSTS,INDREPORT,"INDUCT");
1;
END;