SETL NEWSLETTER NO. 209

AUTOMATIC DATA STRUCTURE CHOICE.

E.SCHONBERG
MARCH 17,1978.

THIS NEWLETTER OUTLINES THE DESIGN OF THE AUTOMATIC DATA-STRUCTURING MODULE CURRENTLY INCORPORATED IN THE OPTIMIZER OF THE SETL COMPILER. ALTHOUGH SEVERAL ASPECTS OF THIS DESIGN HAVE BEEN CONSIDERABLY REVISED AND MODIFIED, IT IS PRESENTED IN UNRETOUCHED FORM , FIRST TO SERVE AS A SHORT INTRODUCTION TO THE USE OF BASES FOR AUTOMATIC DATA-STRUCTURING, AND SECOND, AS A BACKGROUND FOR M.SHARIR↑S RECENT NEWSLETTER ON THE SUBJECT (SETLNL.203 WHICH DETAILS SEVERAL MAJOR IMPROVEMENTS TO THE SCHEME PRESENTED HERE.

THE LAST PHASE OF THE SETL OPTIMIZER SELECTS APPROPRIATE REPRESENTATIONS FOR THE SETS AND MAPS APPEARING IN A USER PROGRAM. THESE REPRESENTATIONS ARE DEFINED IN TERMS OF BASE SETS. WE RECALL HERE BRIEFLY THE MEANING AND SIGNIFICANCE OF BASE SETS, AND THE RUN-TIME ADVANTAGES THAT FOLLOW FROM THEIR EXISTENCE.

A BASE SET, OR BASE FOR SHORT, IS AN AUXILIARY OBJECT, IN TERMS OF WHICH PROGRAM VARIABLES CAN BE DESCRIBED.
THREE BASIC FORMS ARE USED FOR THESE DESCRIPTIONS;
A) IF B IS A BASE SET, THEN AN OBJECT X MAY BE SAID TO BE AN ELEMENT OF THE BASE, I.E. TO HAVE THE MODE (↛B) ;
B) A SET S MAY BE DESCRIBED AS A SUBSET OF THE BASE, I.E. AS HAVING THE MODE ≤↛B≥ .
C) A MAP F MAY BE DESCRIBED AS HAVING THE BASE B FOR ITS DOMAIN, I.E. HAS HAVING THE MODE : MAP(↛B) *. (HERE AND IN WHAT FOLLOWS THE ASTERISK IS USED TO REPRESENT THE GENERAL MODE) .
IN EACH CASE, X, S AND F ARE PROGRAM VARIABLES APPEARING IN OPERATIONS WHOSE EXECUTION WILL BENEFIT FROM THE EXISTENCE OF THE BASE B. B ITSELF IS NOT A PROGRAM VARIABLE : IT DOES NOT APPEAR IN ANY EXECUTABLE INSTRUCTION OF THE PROGRAM. ITS VALUE IS DETERMINED AT ALL TIMES BY THE VALUES OF THE OBJECTS WHICH ARE BASED ON IT. FOR EXAMPLE, WHENEVER THE VARIABLE X RECEIVES A VALUE WHICH CANNOT BE ASCERTAINED TO BE AN ELEMENT OF B, THEN THIS VALUE IS INSERTED INTO THE REPRESENTATION OF B, THEREBY MAINTAINING THE VALIDITY OF THE BASING RELATION X : ↛B ;

BASES ARE USEFUL FOR SEVERAL REASONS :

A) FIRST AND FOREMOST, THEY PROVIDE ACCESSING MECHANISMS INTO BASED OBJECTS, WHICH ARE FASTER AND SIMPLER THAN THE STANDARD HASH-TABLE SEARCHES WHICH APPLY TO UNBASED SETS AND MAPS. THE RESULTING GAIN IS TWOFOLD : EXECUTION OF MANY SETL PRIMITIVE OPERATIONS IS FASTER, AND CODE FOR THEM IS SUFFICIENTLY SIMPLE TO BE EMITTED AS ON-LINE MACHINE CODE.

B) THE REPRESENTATIONS OF BASED OBJECTS ARE MORE COMPACT THAN THOSE OF UNBASED OBJECTS.

C) ITEM B) HOWEVER, IS ONLY TRUE IF BASED OBJECTS (SETS AND MAPS) ARE DENSELY DEFINED OVER THE CORRESPONDING BASES, I.E. IF SUBSETS OF THE BASE, OR DOMAINS OF DEFINITION OF BASED MAPS, HAVE A CARDINALITY COMPARABLE TO THAT OF THE BASE ITSELF. OTHERWISE, STORAGE ALLOCATED TO THE BASE MAY BE WASTED TO SOME DEGREE AND ITERATION OVER SPARSE BASED OBJECTS MAY BE ACTUALLY SLOWER THAN ITERATION OVER UNBASED REPRESENTATIONS OF THE SAME OBJECTS. TO BYPASS THE COUNTERPRODUCTIVE EFFECTS OF BASING SPARSE OBJECTS SPECIAL -SPARSE- REPRESENTATIONS CAN BE SPECIFIED FOR PROGRAM VARIABLES. THESE REPRESENTATION ALLOW THE EXECUTION OF SETL PRIMITIVES WITH AN EFFICIENCY WHICH IS INTERMEDIATE BETWEEN THAT ACHIEVABLE WITH A BASED REPRESENTATION, AND A COMPLETELY UNBASED ONE; STORAGE UTILIZATION IS COMPARABLE TO THAT OF THE CORRESPONDING UNBASED REPRESENTATION .

THE USEFULNESS OF BASES ARISES FROM THE EXISTENCE OF -BASE POINTERS- ; AN OBJECT WHICH IS AN ELEMENT OF A BASE IS IN FACT REPRESENTED BY A POINTER INTO A BLOCK IN THE BASE, WHICH HOLDS INFORMATION RELATING THAT OBJECT TO COMPOSITES (SETS AND MAPS) WHICH HAVE THE SAME BASE. (SEE NL.180 ON THE RUN-TIME DATA STRUCTURES OF SETL) . OPERATIONS ON BASED OBJECTS MAKE USE OF THESE BASE POINTERS ,INSTEAD OF RECREATING THEM BY A HASHING OPERATION AS IS THE CASE WHENEVER AN UNBASED OBJECT IS ACCESSED . FOR EXAMPLE, THE TEST (X IN S) CAN BE PERFORMED EASILY IF X AND S HAVE A COMMON BASE, AND X HOLDS THEREFORE A POINTER INTO THAT BASE. OTHERWISE, THE HASH-SEARCH NECESSARY TO LOCATE X IN THE HASH-TABLE REPRESENTING (IN UNBASED FORM) THE SET S, WILL CREATE SUCH A POINTER, BUT WILL NOT PRESERVE IT BEYOND EXECUTION OF THE INSTRUCTION WHICH CREATED IT. THE PROCESS OF FINDING USEFUL BASES AMOUNTS THEREFORE TO MINIMIZING THE NO. OF BASE POINTERS GENERATED DURING PROGRAM EXECUTION .

IT IS IMPORTANT TO NOTICE IN THIS CONNECTION, THAT A BASE IS USEFUL ONLY IF AT LEAST TWO COMPOSITE OBJECTS ARE BASED ON IT, BECAUSE THEN THE BASE POINTERS HELD BY ONE CAN BE USED TO ACCESS THE OTHER. IF A BASE IS SIMPLY THE DOMAIN OF A MAP (AND NOTHING ELSE) THEN NOTHING IS GAINED BY ITS EXISTENCE , BECAUSE THERE IS NO WAY TO GENERATE ELEMENTS OF THAT DOMAIN WITHOUT RECALCULATING THE CORRESPONDING BASE POINTER. THE SAME IS TRUE IF THE ONLY OBJECTS SUPPORTED BY A BASE ARE A SET AND ITS ELEMENTS. IN THIS CASE THE BASE IS IDENTICAL WITH THE BASED SET .IT IS USEFUL TO SAY THEN THAT THE SET IS ITS OWN BASE. WE CAN EXTEND THIS REMARK BY SAYING THAT IN A PROGRAM WITHOUT BASES, EACH SET AND EACH MAP IN THE PROGRAM HAS ITS OWN BASE. CHOOSING USEFUL BASES, (MANUALLY BY MEANS OF DECLARATIONS, OR AUTOMATICALLY), AMOUNTS TO MERGING THESE INDIVIDUAL BASES INTO SHARED ONES, WHICH SUPPORT BASED REPRESENTATIONS FOR SEVERAL COMPOSITE OBJECTS AT A TIME. CONCEIVABLY, THE PROCESS MAY LEAD TO A SINGLE -UNIVERSAL- BASE, IN TERMS OF WHICH ALL PROGRAM OBJECTS ARE DEFINED. THIS HOWEVER MAY RESULT IN WASTEFUL STORAGE ALLOTION (SEE REMARK C) ABOVE) . ONE OF THE CRITERIA IN CHOOSING BASES MUST THEREFORE BE TO OBTAIN WHENEVER POSSIBLE DENSE DEFI-

NITIONS FOR BASED OBJECTS. THIS MEANS THAT THE PROCESS OF BASE MERGING MUST BE A CAUTIOUS ONE, UNDERESTIMATING RATHER THAN OVER-ESTIMATING THE AMOUNT OF OVERLAP BETWEEN OBJECTS.

THIS REMARK IS OF PARTICULAR IMPORTANCE WHEN CHOOSING BASES FOR THE FORMAL PARAMETERS OF A PROCEDURE. IT IS REASONABLE TO POSTULATE A SIMILAR BASED STRUCTURE FOR EACH ACTUAL PARAMETER, BUT IT IS SAFEST TO ASSUME A DIFFERENT BASE (OR SET OF BASES) FOR EACH POINT OF CALL (SUBJECT TO POSSIBLE MERGING IF EXAMINATION OF OTHER PARTS OF THE PROGRAM SUGGESTS IT). THIS REMARK WILL BE EXPANDED IN WHAT FOLLOWS.

THE PROCESS OF AUTOMATIC DATA-STRUCTURING CAN NOW BE SKETCHED. INPUT TO IT IS THE MAP -TYPES- CREATED BY THE TYPEFINDER, WHICH MAPS VARIABLE OCCURRENCES INTO THE TYPES WHICH THEY ACQUIRE AT RUN-TIME. DATA-STRUCTURING IS ONLY POSSIBLE IF TYPE-DETERMINATION IS COMPLETE AND UNAMBIGUOUS. IF TYPEFINDING IS UNABLE TO DISTINGUISH BETWEEN SETS AND MAPS, OR BETWEEN SETS AND TUPLES, THEN AUTOMATIC DATA-STRUCTURING IS NOT FEASIBLE.

THE COMPLETE PROCESS CONSISTS OF 7 DISTINCT PHASES :

PHASE I INTRODUCES A BASE FOR EACH SET AND MAP IN THE PROGRAM. ON THE ASSUMPTION THAT IN MOST PROGRAMS ALL OCCURRENCES OF A GIVEN VARIABLE HAVE THE SAME MODE, THIS INITIAL BASING IS CHOSEN FOR VARIABLES INSTEAD OF OCCURRENCES . MORE PRECISELY, ALL OCCURRENCES OF A GIVEN VARIABLE WHICH ARE CHAINED BY -FFROM- RECEIVE THE SAME BASE.
THE OUPUT OF THIS PHASE IS THE FOLLOWING :

IA) SB : A SET OF BASES. THIS SET WILL BE DECREMENTED BY SUCCESSIVE PHASES OF THE PROCESS, BUT NO NEW BASES WILL BE GENERATED AFTER THIS PASS.

IB) MODE : A MAP FROM VARIABLE OCCURRENCES TO MODE DESCRIPTORS. THESE MODE DESCRIPTORS HAVE THE SAME FORMAT AS TYPE DESCRIPTORS, BUT INCLUDE THE TYPE -ELEMENT OF BASE- AND THE CORRESPONDING BASE NAME. THIS MAP IS MODIFIED AND REFINED BY SUCCESSIVE PHASES

PHASE II PROPAGATES THE INITIAL BASING CHOICES BY EXAMINING EXTRACTION OPERATIONS, AND MARKING THOSE OVARS WHICH ARE KNOWN ELEMENTS OF BASES, BY VIRTUE OF HAVING BEEN RETRIEVED FROM BASED SETS, MAPS OR TUPLES .
OUTPUT OF THIS PHASE IS THE SET -BASE¬ELMTS- WHOSE MEMBERS ARE PAIRS [OVAR, BASE]. THIS SET IS FURTHER UPDATED IN PHASE V AFTER :LOCATE- INSTRUCTIONS HAVE BEEN GENERATED.

PHASE III MERGES THE BASES CREATED IN THE PRECEDING PHASE, BY DECLARING AS EQUIVALENT THE BASES OF IVARIABLES OF THE SAME INSTRUCTION. IN ADDITION, THIS PHASE GENERATES THE BASE

INSERTION OPERATIONS WHICH ARE CONSISTENT WITH THE POSTULATED BASING RELATIONS OF COMPOSITE OBJECTS IN THE PROGRAM. THESE INSERTION OPERATIONS ARE GENERATED BY EXAMINING ALL OPERATIONS OF INCORPORATION (-WITH- , MAP STORAGE, ETC) AND BY DECLARING THE INCORPORATED ITEM AS BEING AN ELEMENT OF THE CORRESPONDING BASE. THIS MEMBERSHIP DECLARATION IS THEN ENFORCED BY EMITTING BASE INSERTION OPERATIONS FOR ALL VARIABLE OCCURRENCES WHOSE VALUE MIGHT BECOME THE VALUE OF THE INCORPORATED OBJECT.
THE OUPUT OF THIS PHASE IS TWOFOLD :

IIIA) RSB : AN EQUIVALENCE RELATION OVER SB. EACH EQUIVALENCE CLASS CORRESPONDS TO A SINGLE ACTUAL BASE.

IIIB) SLI : A SET OF -LOCATE- INSTRUCTIONS WHICH MUST BE INSERTED IN THE CODE TO ENFORCE THE CHOSEN BASED PREPRESENTATIONS. THE ELEMENTS OF SLI ARE PAIRS : [OI, B] OF VARIABLE OCCURRENCES AND THE BASES INTO WHICH THEY MUST BE INSERTED .

PHASE IV MERGES THE BASES ACCORDING TO THE EQUIVALENCE RELATION RSB, AND EXAMINES EACH RESULTING BASE TO SEE WHETHER IT ACTUALLY SUPPORTS MORE THAN ONE COMPOSITE BASED OBJECT. IF NOT , THEN THE BASE IS USELESS, AND THE CORRESPONDING -LOCATE- INSTRUCTIONS IN -SLI- CAN BE DELETED.

PHASE V OPTIMIZES THE PLACEMENT OF THE -LOCATE- INSTRUCTIONS GENERATED IN PREVIOUS STEPS. THIS AMOUNTS TO PERFORMING A TYPE OF FORWARD CODE MOTION ON THESE INSTRUCTIONS. THE NEED FOR SUCH CODE MOTION IS CLEAR FROM THE FOLLOWING FRAGMENT :

```
(∨X ∈ S) Y = Y + 1 ; END ∨; ..... Z = F(Y) ;
```

THE APPEARANCE OF Y IN A MAP RETRIEVAL OPERATION SUGGESTS THAT IT SHOULD BE AN ELEMENT OF THE (BASE) DOMAIN OF DEFINITION OF F. A -LOCATE- INSTRUCTION PLACED AT THE POINT OF CREATION OF Y WILL HOWEVER GENERATE A NUMBER OF USELESS BASE POINTERS BECAUSE ALL OF THEM (CORRESPONDING TO SUCCESSIVE VALUES OF Y IN THE LOOP) EXCEPT THE LAST ONE ARE DEAD (I.E. REDEFINED BEFORE THEY ARE USED AS BASE POINTERS) . PHASE IV MOVES LOCATE INSTRUCTIONS OUT OF SUCH LOOPS, AND PLACES THEM ON ENTRY TO THE INTERVALS WHERE THEY ARE ACTUALLY NEEDED.

PHASE VI COMPLETES THE DESCRIPTION OF THE MODES OF EACH VARIABLE OCCURRENCE, AND WHENEVER POSSIBLE FOLDS THESE MODES INTO A SINGLE MODE FOR EACH VARIABLE.

PHASE VII REFINES THE BASING CHOICES , BY SELECTING LOCAL, REMOTE OR SPARSE REPRESENTATIONS FOR BASED SETS AND MAPS .

THE TEXT THAT FOLLOWS IGNORES A NUMBER OF CODING DETAILS.THE COMMENTS WHICH ACCOMPANY THE CODE FRAGMENTS ARE IN ANY CASE LESS OBSOLETE THAT THE CODE ITSELF.

```
$ GLOBAL VARIABLES.
$ ------ ----------



VAR AMBIG ,                 $ AMBIG IS FLAG THAT INDICATES THAT
                            $ THE TYPEFINDER LEFT SOME TYPE AMBI-
                            $ GUITY, AND NO STRUCTURING IS
                            $ POSSIBLE.

    MODE ,                  $ MAP FROM OCCURRENCES TO THEIR MODES.

    SB ,                    $ SET OF BASES.

    CL¬MODE ,               $ MAP FROM INITIAL MODES TO OCCUR-
                            $ RENCES. EACH SET OF OCCURRENCES IN
                            $ CL¬MODE(MODE) RECEIVES THE SAME
                            $ FINAL MODE.

    REALB ,                   $ MAP ON SB : FROM EACH BASE TO ITS
                              $ EQUIVALENCE CLASS.


    IS¬FORMAL ,               $ MAP ON SB. INDICATES THAT A BASE
                              $  SUPPORTS THE FORMAL PARAMS OF A
                              $ PROCEDURE.

    CAN¬DROP ,                $ MAP ON SB. INDICATES THAT BASE
                              $ SUPPORTS ONLY 1 OBJECT.

    SLI ,                     $ SET OF -LOCATE- INSTRUCTIONS.

    RSB ,                     $ EQUIVALENCE RELATION OVER SB.

    BASE¬ELMTS,               $ SET OF OCCURRRENCES WHICH ARE KNOWN
                              $ TO BE ELEMENTS OF BASES.


    SBASEDON ;                $ MAP FROM BASES TO SETS AND MAPS
                              $ WHICH ARE BASED ON THEM.
```

```
$ THE FOLLOWING MACROS MANIPULATE MODE DESCRIPTORS. MODE DES-
$ CRIPTORS HAVE THE SAME FORMAT AS TYPE DESCRIPTORS, BUT IT IS
$ CONVENIENT TO DEFINE MACROS THAT CAN BE APPLIED DIRECTLY TO
$ VARIABLE OCCURENCES.

MACRO MGTYP(VAR) ; GROSSTYP(MODE(VAR)) ENDM ;

MACRO MLTYP(VAR) ; LENTYP(MODE(VAR))  ENDM ;

MACRO ELMBASE(SET) ; BASENAM(COMTYP(MODE(SET))) ENDM ;

MACRO DOMBASE(MAP) ; BASENAM(DOMTYP(MODE(MAP))) ENDM ;

MACRO RANBASE(MAP) ; BASENAM(RANTYP(MODE(MAP))) ENDM ;


MACRO COMBASE(TUP, N) ; BASENAM(CTYPN(MODE(TUP), N)) ENDM ;



PROCEDURE AUTO¬DATA ;

AMBIG := FALSE ;

MODE := SB := RSB := SLI := SBASEDON := BASE¬ELMTS := NL ;

GENBASE() ;
GENBELMTS() ;
GENLOCS() ;
MERGEBASE();
MOVELOCS() ;
UPDMODES() ;
REFINE() ;

END PROCEDURE AUTO¬DATA ;



PROCEDURE GENBASE ;

$ THIS PROCEDURE GENERATES A BASE FOR EACH INCARNATION OF A SET
$ OR MAP. AN INCARNATION AS USED HERE MEANS A SET OF OCCURRRENCES
$ OF A GIVEN VARIABLE, WHICH ARE LINKED BY THE CHAINING FUNCTION
$ -FFROM-, AND CAN THEREFORE BE EXPECTED TO HAVE THE SAME BASING.

$ THE FREQUENTLY USED INSTRUCTION :

$         S := S WITH X ;

$ MUST BE SPECIAL-CASED, BECAUSE THE TWO OCCURRENCES OF S ARE
$ NOT LINKED BY -FFROM- AS CURRENTLY DEFINED. NEVERTHELESS
```

```
$ IT IS DESIRABLE TO GIVE BOTH THE SAME BASING. THIS
$ IS DONE IN THE CODE BELOW.

$ IN THE SAME FASHION, GLOBAL OPERATIONS SUCH AS SET UNION
$ AND INTERSECTION SUGGEST THE SAME BASING FOR THE OVARIABLE
$ AS FOR ITS IVARIABLES. FOR SUCH INSTRUCTIONS WE ALSO PRO-
$ PAGATE FROM IVARS TO OVARS.

$ TO FACILITATE THE ADJUSTMENT OF MODES DURING PHASE V, IT IS
$ CONVENIENT TO ASSUME THAT TUPLES ARE ALSO BASED, I.E. THAT
$ THEIR COMPONENTS ARE ALSO ELEMENTS OF SOME BASE SET. THE TUPLE
$ ITSELF NEED NOT BE MARKED AS BEING A BASED OBJECT. INTRODUCTION
$ OF SUCH BASES IS HARMLESS, BECAUSE IF NO COMPOSITE OBJECTS
$ END UP BEING BASED ON THEM, THEY WILL BE DROPPED DURING THE BASE
$ MERGING PHASE.


(∨B↛BLOCKS, FORALLCODE(B,I))

    OV := GET¬OVAR(I) ;
    IF MODE(OV)/=OM THEN     $ VARIABLE ALREADY EXAMINED.
        CONTINUE ;
    END IF;

    NEWM := GENBMODE(OV) ;
    MODE(OV) := NEWM ;
$ IF THE VARIABLE IS OF PRIMITIVE TYPE, OR IF TYPE-FINDING
$ YIELDED INCOMPLETE INFORMATION, THERE IS NO MODE INFORMATION
$ TO PROPAGATE.
$ IF OV IS A -READ- VARIABLE, THEN ITS TYPE IS T¬G, BUT WE
$ WILL WANT TO GENERATE AN APPROPRIATE BASED MODE (IF OV IS
$ IN FACT STRUCTURED) FOR THE IVARIABLES CHAINED TO IT. THIS
$ IS DONE IN THE LOOP THAT FOLLOWS.
    IF PRIMITIVE(GROSSTYP(NEWM)) OR OPCODE(I) /= Q1¬READ
        THEN CONTINUE ;;

    $ THE MODE DESCRIPTOR FOR OV HAS NOW BEEN BUILT. PROPAGATE
    $ IT TO LINKED OCCURRENCES OF THE SAME VARIABLE.

    MODE(OV) := NEWM ;
    WORK := FFROM≤OV≥ ;

    (WHILE WORK/=NL)
        [P, OI] FROM WORK;

        IF MODE(OI)/=OM THEN CONTINUE ; END IF ;

    IF NEWM /= T¬G THEN      $ GENBMODE YIELDED A BASED MODE
        MODE(OI) := NEWM ;; $ PROPAGATE IT .
    ELSE                     $ OVARIABLE WAS READ.TRY TO FIND MODE
        MODE(OI) := GENBMODE(OI) ;$ FOR IVARIABLE,
    END IF ;
```

```
        IF MODE(OI) /= T¬G THEN $ CONTINUE PROPAGATING.
                WORK + ≤ [NNP, OI] : [NP, OI]↱FFROM≤OI≥ ↑

                (NNP:=P .CC NP) /= ERROR¬PATH ≥ ;;
            $ NOW TRY TO PROPAGATE FROM IVAR TO OVAR, FOR OPCODES
            $ -WITH- AND -LESS-.
            IF IS¬IVAR(OI) AND (( OI¬OP(OI) ↱ ≤Q1¬WITH, Q1¬LESS,
            Q1¬LESSF ≥ AND ARGNO(OI)=2)

            $ OR FOR UNION, INTERSECTION AND THE LIKE.
            OR OP¬OI(OI) ↱ ≤Q1¬ADD, Q1¬SUB, Q1¬MULT, Q1¬MOD≥)
                THEN
                WORK WITH [P, OI¬OVAR(OI)]  ;
            END IF ;

        END WHILE ;
    END ˇB ;

    $ NOW CREATE THE MAP CL¬MODE. THIS MAP IS USED DURING THE LAST
    $ PHASE OF DATA-STRUCTURING, WHEN SELECTING LOCAL VS. REMOTE
    $ ATTRIBUTES FOR BASED REPRESENTATIONS. ALL OCCURRENCES WHICH
    $ APPEAR IN THE SAME CLASS IN CL¬MODE RECEIVE THE SAME ATTRIBUTE.

    CL¬MODE := INVERSE MODE ;

    END PROCEDURE GENBASE ;




    FUNCTION GENBMODE(V) ;

    $ THIS FUNCTION GENERATES A BASED MODE DESCRIPTOR FOR VARIABLE V,
    $ USING AVAILABLE TYPE INFORMATION ON V. IF THE TYPE INFORMATION
    $ IS INCOMPLETE, THE FUNCTION RETURNS T¬G.

    T := TYPES≤V≥ ;
    IF ≡≤GROSSTYPE(TP) : [-, TP] ↱ T≥ > 1  THEN
        AMBIG := TRUE ; RETURN T¬G ; END IF ;

    V¬TYPE := (ARB T) (2) ;
    NEWM := [GROSSTYP(V¬TYPE)] ; $ TEMPLATE FOR MODE DESCRIPTOR.
    CASE GROSSTYP(V¬TYPE) OF
        (TSET): COMTYP(NEWM) := [TELMT, OM, BASE:=NEWAT] ;
            SBASEDON≤BASE≥ WITH OI¬NAME(V) ;
            SB WITH BASE ;

        (TMAP): COMTYP(NEWM) := [TKNT, []] ;
            DOMTYP(NEWM) := [TELMT, OM, BASE1:=NEWAT] ;
            RANTYP(NEWM) := [TELMT, OM, BASE2:=NEWAT] ;
            SBASEDON(BASE1≥ WITH OI¬NAME(V) ;
            SB + ≤BASE1,BASE2≥ ;
```

```
        (TUNT) : COMTYP(NEWM) := [TELMT, OM, BASE:=NEWAT] ;
            SB WITH BASE ;

        (TKNT) : COMTYP(NEWM) := [] ;
        (∨I := 1,..≡COMTYP(V¬TYPE));
            CTYPN(NEWM,I) := [TELMT, OM, BASE:=NEWAT] ;
            SB WITH BASE ;
        END ∨;

        ELSE                        $ MUST BE PRIMITIVE TYPE.
            NEWM := V¬TYPE;
    END CASE ;
    RETURN NEWM ;
    END FUNCTION GENBMODE ;



    PROCEDURE GENBELMTS ;

    $ AT THIS POINT, ALL STRUCTURED OBJECTS HAVE BASES ASSIGNED
    $ TO THEM. WE NOW EXAMINE EXTRACTION  OPERATIONS, AND MARK
    $ THOSE OVARIABLES THAT ARE KNOWN TO BE BASE ELEMENTS. THIS IS
    $ DONE BY INSERTING THE PAIR [OVAR, BASE] IN THE SET BASE¬ELMTS.
    $ THIS INFORMATION IS USED WHEN GENERATING BASE INSERTION OPS
    $ IN THE NEXT PHASE (SEE INSERTLOCS). IT ALSO OBVIATES THE
    $ NEED FOR -CRTHIS- WHEN ENFORCING BASING CHOICES :
    $ BFROM INFORMATION IS SUFFICIENT. (SEE GENLOCS AND RELATED
    $ ROUTINES) .

    (∨B ∈ BLOCKS, FORALLCODE(B,I))

      . . . . .
    $ NOW PROPAGATE KNOWN BASE MEMBERSHIP WHICH IS TRANSMITTED
    $ BY SIMPLE ASSIGNMENTS.

    WORK := BASE¬ELMTS ;
    (WHILE WORK /= NIL)
        [V, B] FROM WORK;
        (∨[-,IV]∈FFROM≤V≥↑OI¬OP(IV) = Q1¬ASN)
            WORK  WITH [OI¬OVAR(IV), B];
            BASE¬ELMTS WITH [OI¬OVAR(IV), B] ;
        END ∨;
    END WHILE ;
    END PROCEDURE GENBELMTS ;


    PROCEDURE GENLOCS ;

    $ THIS PROCEDURE ENFORCES THE BASINGS OF COMPOSITE OBJECTS, BY
    $ GENERATING -BASE INSERTION-  ( -LOCATE-) INSTRUCTIONS FOR ALL
    $ VARIABLE OCCURRENCES WHICH MIGHT BE INCORPORATED INTO A
    $ COMPOSITE OBJECT. FOR EXAMPLE, THE INSTRUCTION :
```

```
$       S1 := S WITH X;

$ LEADS TO THE BASING RELATION :

$       X : ↛B ;

$ WHERE B IS THE BASE PREVIOUSLY ASSIGNED TO THE VARIABLE OCCUR-
$ RENCE OF S. THIS BASING RELATION FOR X IS ENFORCED BY EMITTING
$ -LOCATE- INSTRUCTIONS FOR ALL THE OCCURRENCES IN BFROM≤X≥, I.E.
$ FOR ALL OCCURRENCES WHOSE VALUES  BECOME THE VALUE OF X IN
$ THE COURSE OF PROGRAM EXECUTION. THESE -LOCATE- INSTRUCTIONS
$ ARE NOT INSERTED INTO THE CODE, BUT KEPT IN THE SET -LSI- , FOR
$ THE FOLLOWING REASONS :

$ A) THE BASES BEING USED AT THIS STAGE ARE NOT THE ACTUAL BASES
$ WHICH WILL APPEAR AT RUN-TIME. ACTUAL BASES WILL EMERGE SUBSE-
$ QUENTLY AS EQUIVALENCE CLASSES OF THE BASE NAMES INTRODUCED SO
$ FAR .

$ B) SOME BASES MAY EVENTUALLY PROVE USELESS, BECAUSE THEY SUPPORT
$ ONLY ONE COMPOSITE OBJECT, IN WHICH CASE ALL -LOCATE- INSTRUC-
$ TIONS INTO THEM MUST BE DROPPED.


$ IN THE PROCESS OF ENFORCING BASING RELATIONS, EQUIVALENCE RELA-
$ TIONS EMERGE AMONG BASES. IF AN OCCURRENCE OCRX, WHICH IS AN
$ ELEMENT OF  BFROM≤X≥, HAS ALREADY RECEIVED A LOCATE INSTRUCTION
$ INTO SOME BASE BX, THEN INSTEAD OF GENERATING A NEW LOCATE
$ INSTRUCTION INTO B, WE EQUIVALENCE B AND BX. THIS IS DONE BY
$ SIMPLY ADDING THE PAIR [B, BX]  TO THE RELATION RSB; IT
$ INDICATES THAT B AND BX ARE TO BE CONSIDERED AS TWO NAMES OF THE
$ ACTUAL BASE AB, (WHICH WILL EMERGE AS THE REPRESENTATIVE OF
$ THE EQUIVALENCE CLASS TO WHICH B AND BX BELONG) .

$ CERTAIN INSTRUCTIONS INDICATE SIMILAR EQUIVALENCING OF BASES
$ WITHOUT GENERATING LOCATE INSTRUCTIONS : SET UNION, INTERSECTION
$ AND THE LIKE, INDICATE THAT THEIR ARGUMENTS MUST HAVE THE SAME
$ BASE. THE CURRENT PROCEDURE ALSO USES SUCH INSTRUCTIONS TO BUILD
$ THE RELATION RSB.


$ IN ORDER TO SIMPLIFY THE MODE ADJUSTMENT PHASE, THE ARBITRARY
$ BASINGS CHOSEN FOR TUPLE COMPONENTS AND FOR THE RANGE OF MAPS
$ IN THE PRECEDING PHASE, ARE PROPAGATED DURING THE PRESENT
$ PHASE, IN EXACTLY THE SAME WAYS AS SET ELEMENTS. OPERATIONS OF
$ INCORPORATION, I.E. TUPLE ASSIGNMENTS, ARE TREATED AS MAP STORES
$ AND THE SAME BASE EQUIVALENCING PROCEDURE IS USED IN ALL CASES.
```

```
(vB↠BLOCKS, FORALLCODE(B,I))

    [OV, IV1, IV2] = ARGS(I) ;$ UNPACK INSTRUCTION.

    CASE OPCODE(I) OF

        $ FIRST CONSIDER OPERATIONS WHICH ONLY EQUIVALENCE BASES.

     (Q1¬ADD, Q1¬SUB, Q1¬MULT, Q1¬MOD, Q1¬EQ, Q1¬NE, Q1¬INC) :

        IF MGTYP(IV1) NOTIN ≤TSET, TTUP, TMAP≥ THEN
            CONTINUE v ;     $ MUST BE PRIMITIVE TYPE.
        ELSE
            MERGEOBJ(IV1, IV2) ;
        END IF ;

        $ SET INSERTION AND MEMBERSHIP OPS :
        $ EQUIVALENCE BASES FOR COMPOSITE OBJECT, AND GENERATE
        $ LOCATE INSTRUCTION  FOR SECOND ARGUMENT.

        (Q1¬WITH, Q1¬LESS, Q1¬IN, Q1¬NOTIN) :

        IF MGTYP(IV1)=TSET OR MGTYP(IV1)=TUNT THEN
            PROPELMT(IV1, IV2) ;
        ELSE
        $ THE MAP CASE :  M WITH [X,Y]   IS TREATED BY EQUIVA-
        $ LENCING THE ASSIGNED BASES OF X , Y WITH THOSE OF M

        PROPELMAP(IV1, IV2) ;

        END IF ;

    $ MAP RETRIEVAL OPERATIONS.
    (Q1¬OF, Q1¬OFA, Q1¬OFB) :
        IF MGTYP(IV1)=TMAP THEN
            PROPOFMAP(IV1, IV2) ;
        END IF ;

    $ SINGLE-VALUED STORAGE OPS FOR MAPS AND TUPLES.
    (Q1¬SOF, Q1¬LESSF) :
        IF MGTYP(IV1)=TMAP THEN
            PROPSOFMAP(OV, IV1, IV2) ;
        ELSE
            PROPSOFTUP(OV, IV1, IV2) ;
        END IF ;

    $ MULTIVALUED STORAGE : F≤X≥ = S ; TREATS THE RIGHT-HAND SIDE
    $ DIFFERENTLY, AND REQUIRES A SEPARATE PROCEDURE.

    (Q1¬SOFA) : PROPSOFAMAP(OV, IV1, IV2) ;

    ELSE                        $ OTHER OPCODES ARE NOT EXAMINED.
        CONTINUE v ;
    END CASE ;
```

```
    END ∨B;
    END PROCEDURE GENLOCS ;


    PROCEDURE MERGEOBJ(V1, V2) ;

    $ THIS PROCEDURE EQUIVALENCES THE BASES OF COMPOSITE OBJECTS WHICH
    $ ARE IVARIABLES OF THE SAME INSTRUCTION.

    BF1 := BFROM≤V1≥ ;
    BF2 := BFROM≤V2≥ ;

    IF IS¬GLOB(OI¬NAME(V1)) OR IS¬GLOB(OI¬NAME(V2)) THEN
        $ MERGE THE BASES OF ALL ITEMS IN BFROM≤V1≥ AND BFROM≤V2≥ .

        MERGE((BF1+BF2)WITH V2, V1) ;
    ELSE
    . . .
    END IF IS¬GLOB;
    END PROCEDURE MERGEOBJ ;


    $ THE REMAINING SUBSIDIARY PROCEDURES  TO GENLOCS
    $ ARE VERY SIMILAR IN STRUCTURE, AND WILL NOT BE DETAILED HERE.


    PROCEDURE MERGE(VARSET, OBJ) ;

    $ THIS PROCEDURE EQUIVALENCES THE BASES OF ALL OBJECTS IN VARSET
    $ TO THE CORRESPONDING BASES OF OBJ. IF OBJ IS A SET OR H-TUPLE, A
    $ SINGLE BASE FROM EACH OBJECT IS INVOLVED. IF IT IS A MAP, THEN
    $ DOMAIN AND RANGE BASES ARE EQUIVALENCED SEPARATELY. FOR KNOWN
    $ LENGTH MIXED TUPLES, ONE BASE PER COMPONENT IS INVOLVED.

    IF MGTYP(OBJ) ∉ ≤TSET, TUNT≥ THEN
        (∨OI ∉ VARSET) EQUIV(ELMBASE(OI), ELMBASE(OBJ)) ; END ∨;
    ELSEIF MGTYP(OBJ)=TMAP THEN
        (∨OI ∉ VARSET)
            EQUIV(DOMBASE(OI), DOMBASE(OBJ)) ;
            EQUIV(RANBASE(OI), RANBASE(OBJ)) ;
         END ∨ ;

    ELSEIF MGTYP(OBJ) = TKNT THEN
        (∨OI ∉ VARSET, IX := 1..,MLTYP(OBJ))
            EQUIV(COMBASE(OI, IX), COMBASE(OBJ, IX)) ;
        END ∨ ;
    END IF ;
    END PROCEDURE MERGE ;
```

```
PROCEDURE EQUIV(B1, B2) ;

IF B1 /= OM AND B2 /= OM THEN
    RSB WITH [B1, B2] ;
END IF ;

END PROCEDURE  EQUIV ;



PROCEDURE INSERTLOCS(VARSET, BASE) ;


$ THIS PROCEDURE GENERATES -LOCATE- INSTRUCTIONS, INTO -BASE-,
$ FOR EACH VARIABLE OCCURRENCE IN VARSET. IF AN ELEMENT OF
$ VARSET HAS ALREADY RECEIVED A -LOCATE- INSTRUCTION, OR IF
$ IT IS KNOWN TO BE AN ELEMENT OF A BASE (BASE¬ELMTS(V)
$ NOT NIL) BY VIRTUE OF PREVIOUS EXTRACTION, THEN ITS OLD BASE
$ AND THE NEW BASE ARE EQUIVALENCED.
$ THE LOCATE INSTRCUCTIONS ARE INSERTED INTO THE GLOBAL MAP -LSI-.

(ᵛOI ↱ VARSET)
    IF (B1:= SLI(OI) /= NIL) OR
        (B1:= BASE¬ELMTS(OI) /= NIL) THEN
        EQUIV(BASE, B1);
    ELSE
        SLI(OI) := BASE ;
    END IF ;
END ᵛ ;
END PROCEDURE INSERTLOCS ;



PROCEDURE MERGEBASES ;

$ AT THIS POINT, THE SET OF BASES SB, AND THE EQUIVALENCE MAP
$ RSB HAVE BEEN BUILT. WE NOW COMPLETE THE DESCRIPTION OF THE
$ ACTUAL BASES WHICH WILL APPEAR IN THE FINAL MODE DESCRIPTORS.
$ THIS IS DONE IN THREE STEPS :

$ A) THE EQUIVALENCE CLASSES IN SB ARE BUILT OUT OF RSB. ANY
$ MEMBER OF EACH EQUIVALENCE CLASS IS CHOSEN AS A REPRESENTATIVE
$ OF ITS CLASS, AND A MAP -REALB- IS DEFINED FROM EACH BASE TO
$ THE REPRESENTATIVE OF ITS CLASS.
```

```
$ THE MAPS -SBASEDON- ( FROM BASES TO BASED OBJECTS) AND SLI (FROM
$ VARIABLE OCCURRENCES TO THE BASES INTO WHICH THEY ARE INSERTED)
$ ARE NOW FOLDED SO THAT ONLY REAL BASES APPEAR IN THEM.

$ B) THE MODES OF BASES ARE DETERMINED. THIS IS DONE BY EXAMINING
$ THE MODE OF ALL OBJECTS WHICH ARE INSERTED IN EACH BASE BY MEANS
$ OF -LOCATE- INSTRUCTIONS, AND PERFORMING THE DISJUNCTION OF ALL
$ THEIR MODES. THIS PROCESS WILL ALSO UNCOVER RELATIONS AMONG
$ BASES, FOR EXAMPLE THAT B1 HAS ELEMENTS OF MODE [↗B2, ↗B2 ] .

$ THIS PROCESS WILL UNCOVER FURTHER EQUIVALENCES BETWEEN BASES.
$ AS A RESULT, THE STEPS A) AND B) MUST BE ITERATED AS LONG AS
$ THE SET OF EQUIVALENCES RSB IS NOT EMPTY.

$ C) USELESS BASES, WHICH ONLY SUPPORT ONE COMPOSITE OBJECT AND
$ ITS ELEMENTS, ARE FLAGGED AS SUCH.


$ THE BASE MODES DETERMINED IN THIS FASHION WILL BE USED WHEN
$ COMPLETING THE MODE DESCRIPTION OF VARIABLES. WHENEVER THE MODE
$ DESCRIPTOR  (↗B) APPEARS, IF B IS A USELESS BASE, WE REPLACE
$ THIS DESCRIPTOR BY THE MODE DESCRIPTOR FOR THE  BASE ELEMENT OF
$ B. IT IS FOR THIS PURPOSE THAT TUPLES AND MAP RANGES WERE GIVEN
$ FICTICIOUS BASES : THEIR MODES CAN NOW BE USED TO DETERMINE
$ FULLY THE MODE OF UNBASED OBJECTS, WITHOUT HAVING TO PERFORM A
$ COMPLETE TYPEFINDING ANALYSIS ANEW.


(WHILE RSB/=NL)
    EQUIBASES() ;
    MODEBASES() ;
END WHILE ;

DROPBASES() ;



END PROCEDURE MERGEBASES ;



PROCEDURE EQUIBASES ;

$ THIS PROCEDURE BUILDS THE EQUIVALENCE CLASSES OF BASES OUT OF
$ THE EQUIVALENCE RELATION RSB. THE ALGORITHM USES THE STANDARD
$ SPANNING TREE TECHNIQUE DESCRIBED IN KNUTH, ALGORITHM 2.3.3E.
$ EACH CONNECTED TREE CORRESPONDS TO ONE EQUIVALENCE CLASS. WE
$ TAKE THE BASE AT THE ROOT OF THE TREE TO BE THE REPRESENTATIVE
$ BASE FOR THE CLASS, AND WE CREATE A MAP -REALB- FROM THE ORI-
$ GINAL BASES TO THEIR REPRESENTATIVES. THIS MAP IS THEN USED
$ TO UPDATE -SLI- (LOCATE INSTRUCTIONS) AND -SBASEDON- (FROM
```

```
$ BASES TO BASED OBJECTS) SO THAT ONLY THE REAL BASES APPEAR.
$ THE ALGORITHM USES THE AUXILIARY MAP -PARENT- WHICH POINTS
$ UP IN THE SPANNING TREE BEING BUILT, AND THE FUNCTION -ROOT-
$ WHICH FOLLOWS THE LINKS IN PARENT TO THE TOP OF THE TREE.

$ THE SET OF BASES -SB- IS REDUCED EVERY TIME THIS PROCEDURE
$ IS INVOKED. THE MAP -REALB- IS HOWEVER DEFINED OVER ALL BASES
$ WHICH HAVE BEEN INTRODUCED IN THE PROGRAM. ITS RANGE IS, AT EACH
$ STEP IN THE PROCESS, SOME SUBSET OF SB, WHICH AFTER CLEANUP
$ BECOMES THE FULL CONTENTS OF THE NEW SB.
. . .
$ THE SPANNING FOREST IS NOW BUILT. THE ROOT OF EACH TREE
$ IS CHOSEN AS THE REAL BASE CORRESPONDING TO EACH EQUIVALENCE
$ CLASS. WE MAP EACH ORIGINAL BASE INTO ITS REAL ONE.

(∨B ∈ SB) REALB(B) := ROOT(PARENT, B) ; END ∨;

$ UPDATE REALB FOR BASES WHICH HAVE BEEN EQUIVALENCED IN PRE-
$ VIOUS PASSES, AND DO NOT APPEAR ANY MORE IN SB.

(∨B ∈ DOMAIN REALB ↑ B NOTIN SB) REALB(B) := REALB(REALB(B)) ;

$ UPDATE THE -SBASEDON- MAP, BY TAKING THE UNION OVER ALL MEMBERS
$ OF EACH EQUIVALENCE CLASS.
(∨B ∈ SB) SBASEDON≤REALB(B)≥ + SBASEDON≤B≥) ; END ∨;

$ UPDATE THE SET OF BASE INSERTIONS, TO MENTION ONLY REAL BASES.

SLI := ≤ [OI, REALB(B) ] : [OI, B] ∈ SLI≥ ;

$ DO THE SAME FOR  BASE¬ELMTS.

BASE¬ELMTS := ≤ [OI, REALB(B)] : [OI, B] ∈ BASE¬ELMTS≥ ;

$ DISCARD THE EQUIVALENCE RELATIONS JUST PROCESSED, AND KEEP
$ ONLY REAL BASES IN SB.

RSB := NL ;
SB := RANGE REALB ;
END PROCEDURE EQUIBASES ;



FUNCTION ROOT(MAP ,NODE) ;

$ FOLLOW THE -NODE TO PARENT- MAP TO THE ROOT OF THE TREE.

(WHILE MAP(NODE)/=NODE) NODE := MAP(NODE) ; END WHILE;
RETURN NODE;
END FUNCTION ROOT ;
```

```
PROCEDURE MODEBASES ;

$ THIS PROCEDURE EVALUATES THE MODE OF THE ELEMENTS OF REAL
$ BASES, BY TAKING THE DISJUNCTION (IN THE SENSE OF THE TYPE
$ CALCULUS) OF THE OBJECTS LOCATED IN THEM. THE INVERSE OF -SLI-
$ MAPS EACH BASE INTO THE INSERTED OBJECTS, AND IS THE DRIVING
$ STRUCTURE OF THIS PROCEDURE.

$ THE DISJUNCTION OF TWO TYPES MAY LEAD TO FURTHER EQUIVALENCE
$ RELATIONS AMONG BASES, REQUIRING ANOTHER MERGING PASS.

$ IT IS CONVENIENT TO DISREGARD THE SET NATURE OF BASES, AND
$ DEFINE THEIR MODE AS BEING THE MODE OF THEIR ELEMENTS. THIS
$ SIMPLIFIES THE CODE BELOW.

$ FOR BASES WHICH BELONG TO THE EQUIVALENCE CLASS OF A GIVEN
$ REAL BASE, THE MODE IS OF COURSE THAT OF THE REAL BASE. THE
$ THE MAP -REALB- MAPS EACH BASE INTO ITS REAL REPRESENTATIVE.


SLI¬INV := INVERSE SLI ;

(∨LOCSET := SLI¬INV≤BASE≥)
    MODE(BASE) := .MODEDIS  : MODE[LOCSET]  ;
    END ∨ LOCSET ;

$ NOW PROCESS THE OLDER BASES, WHICH DO NOT FIGURE IN SB.

(∨B ∈ DOMAIN REALB ↑ B NOTIN SB)
    MODE(B) = MODE(REALB(B)) ;
END ∨B;


END PROCEDURE MODEBASES ;



FUNCTION M1 .MODEDIS M2 ;

$ THIS PROCEDURE EVALUATES THE DISJUNCTION OF TWO MODE DESCRIPTORS
$ THIS PROCEDURE  DIFFERS FROM THE ONE OF THE SAME NAME IN THE
$ TYPEFINDER, IN THAT IT HANDLES -ELEMENT OF BASE- DESCRIPTORS.

$ AT THIS POINT, SUCH MODES APPEAR ONLY AS EMBEDDED IN OTHER DES-
$ CRIPTORS SUCH AS ≤∈B≥, [∈B], AND MAP(∈B1)∈B2 . THE VARIABLES
$ BEING INSERTED INTO BASES HAVE NOT YET RECEIVED THE MODE : ∈B .
$ THE RULES FOR DISJUNCTION OF SUCH MODES ARE AS FOLLOWS :

$ A) THE DISJUNCTION OF DIFFERENT GROSSTYPES YIELDS -GENERAL-.

$ B) THE DISJUNCTION OF TWO SETS YIELDS THE SET WHOSE ELEMENTS ARE
$ THE DISJUNCTION OF THE RESPECTIVE ELEMENT DESCRIPTORS OF EACH.
```

```
$ C) THE DISJUNCTION OF TWO -ELEMENT OF BASE- MODES YIELDS AN ELE-
$ MENT MODE, AND HAS THE SIDE-EFFECT OF EQUIVALENCING THE TWO
$ BASES.

$ THE REMAINING RULES FOR TUPLES AND MAPS CAN BE GLEANED FROM
$ THE CODE THAT FOLLOWS.


IF GROSSTYP(M1) /= GROSSTYP(M2) THEN RETURN TGEN ;

ELSEIF ISPRIM(GROSSTYP(M1)) THEN RETURN M1 ;

ELSEIF GROSSTYP(M1)=TELMT THEN
    IF REALB(B1:=BASENAM(N1M1))/=REALB(B1:=BASENAM(M2)) THEN
        EQUIV(B1, B2) ;
    END IF ;
    RETURN M1 ;

ELSE                            $ COMPOSITE MODES.
    DISM := [GROSSTYP(M1)] ; $ TEMPLATE FOR NEW DESCRIPTOR.

    CASE GROSSTYP(M1) OF

    (TSET, TUNT) : COMPTYP(DISM) := COMTYP(M1) .MODEDIS
            COMPTYP(M2) ;


    (TMAP) : COMTYP(DISM) := [TKNT, []] ;

        DOMTYP(DISM) := DOMTYP(M1) .MODEDIS DOMTYP(M2) ;
        RANTYP(DISM) := RANTYP(M1) .MODEDIS RANTYP(M2) ;

    (TKNT) : COMTPTYP(DISM) := [TMTUP, []] ;
        (vIX := 1...LENTYP(M1))
            CTYPN(DISM, IX) := CTYPN(M1, IX) .MODEDIS
                                    CTYPN(M2, IX) ;
        END vIX;
    END CASE;

    RETURN DISM ;

END IF ;

END FUNCTION M1 .MODEDIS M2 ;


PROCEDURE DROPBASES ;

$ ANY BASE WHICH SUPPORT A SINGLE COMPOSITE OBJECT CAN IN PRIN-
$ CIPLE BE DROPPED, BECAUSE THERE IS NO USEFUL WAY TO ACCESS ITS
$ BASE POINTERS. THIS PROCEDURE FLAGS SUCH BASES, AND DELETES
$ INSERTION OPERATIONS WHICH REFER TO THEM.
```

```
. . .
END PROCEDURE DROPBASES ;



FUNCTION SUBSTMD(OBJ) ;

$ THIS PROCEDURE UPDATES MODE DESCRIPTORS, BY REPLACING REFERENCES
$ TO DROPPED BASES BY THE MODE DESCRIPTORS FOR THE CORRESPONDING
$ BASE ELEMENT. IT IS A STANDARD RECURSIVE SUBSTITUTION PROCEDURE.

$ THIS PROCEDURE ALSO REPLACES BASE NAMES BY THE NAMES OF THEIR
$ REPRESENTATIVES, SO THAT THE UPDATED MODE CONTAINS ONLY REFE-
$ RENCES TO REAL BASES .


$ OBJ CAN BE AN OCCURRENCE OR A BASE.

M := MODE(OBJ) ;


. . . .

END FUNCTION SUBSTMD ;



PROCEDURE COMPDMODES ;


$ THIS PROCEDURE BUILDS THE MODE OF DROPPED BASES. THE PURPOSE OF
$ INTRODUCING DUMMY BASES FOR TUPLES AND FOR THE RANGE OF MAPS HAS
$ BEEN TO USE THESE BASES AS MARKERS FOR POSSIBLY COMPLEX STRUC-
$ TURES WHICH MAY THEMSELVES BE BASED. THESE MARKERS ARE REPLACED
$ BY THE CORRESPONDING STRUCTURES BY MEANS OF PROCEDURE -SUBSTMD-
$ AS DESCRIBED ABOVE. AN IMPORTANT AND FREQUENT CASE WHERE THIS
$ MECHANISM IS USEFUL IS THAT OF MULTIVARIATE MAPS.THE OPERATION :

$       F(X, Y) := Z;

$ EXPANDS INTO UNIVARIATE RETRIEVALS AND STORAGES,

$       T := F≤X≥ ;
$       T(Y) := Z ;
$       F≤X≥ := T ;

$ WHICH GENERATE THE BASINGS :

$       F : MAP≤↱B1≥↱B2
$       T : MAP(↱B3)↱B4
```

```
$ AND ALSO PRODUCES LOCATE INSTRUCTIONS :

$       B2 WITH T ;
$       B3 WITH Y ;
$       B4 WITH Z ;

$ IF F IS ONLY USED AS A BIVARIATE MAP, THEN B2 WILL EVENTUALLY
$ BE RECOGNIZED AS DROPPABLE , AND AFTER DETERMINING THAT THE MODE
$ OF B2 IS : MAP(↛B3)↛B4 , THE FINAL MODE FOR F WILL BE :

$       F : MAP≤↛B1≥MAP(↛B3)↛B4 ;

$ WHICH IS THE DESIRED DESCRIPTOR.

$ THE PROCEDURE BELOW ITERATES OVER THE MODES OF DROPPABLE
$ BASES, UNTIL THEY CONTAIN ONLY REFERENCES TO PRIMITIVE TYPES
$ AND TO BASES WHICH CAN-T BE DROPPED.

WORK := ≤ B ↛ SB ↑ CAN¬DROP(B) ≥ ;

(WHILE WORK /= NL)
    B := ARB WORK ;
    MODE(B) := SUBSTMD(MODE(B)) ;

    IF (∨BA ↛ BASEA(MODE(B)) ↑ BA NOTIN WORK ) THEN
        WORK LESS B ;
    END IF ;
END WHILE ;

RETURN;

END PROCEDURE COMPDMODES ;

PROCEDURE MOVELOCS ;

$ THIS PROCEDURE MOVES -LOCATE- INSTRUCTIONS OUT OF LOOPS WHEN-
$ EVER THE BASE POINTER WHICH IT GENERATES IS NOT ACTUALLY USED
$ WITHIN THE LOOP.THE FOLLOWING  CASE IS TYPICAL : A VARIABLE X IS
$ KNOWN TO BE (↛B) ; BFROM≤X≥ INCLUDES THE FOLLOWING OCCURRENCE
$ OF X :

        (∨I := 1..100) X := X + Y ; END ∨;

$ THE PROCEDURE -GENLOCS- WILL HAVE INSERTED A LOCATE INSTRUCTION
$ WITHIN THE LOOP, FOR THE OVARIABLE X THEREIN. THIS IS CLEARLY
$ INCORRECT, BECAUSE EACH SUCH VALUE OF X, (EXCEPT THE LAST ONE )
$ IS NOT USED AS A BASE ELEMENT : ITS BASE POINTER IS DEAD. THE
$ PROPER PLACE FOR THE LOCATE INSTRUCTION IS OF COURSE ON EXIT
$ FROM THE LOOP. THE PROCEDURE BELOW SYSTEMATIZES THIS PROCESS.
$ A- LOCATE- INSTRUCTION CAN BE MOVED OUT OF AN INTERVAL IF NO
$ USE IS MADE OF THE BASE POINTER WHICH IT GENERATES, WITHIN ITS
```

```
    $ OWN INTERVAL. THIS CAN BE ASCERTAINED BY FOLLOWING -BFROM-
    $ OF THE (PREMATURELY) LOCATED VARIABLE. IF WE REACH AN OPERATION
    $ WHICH USES THE BASE POINTER WITHIN THE INTERVAL THEN THE -LOCATE
    $ CANNOT BE MOVED. IF THE USE APPEARS IN SOME SUCCESSOR INTERVAL,
    $ THEN IT MUST BE ADVANTAGEOUS TO MOVE THE -LOCATE- OPERATION TO
    $ THE HEAD OF THAT INTERVAL . IN SCANNING -BFROM- WE BUILD A MAP
    $ -MOVETO- FROM VARIABLE OCCURRENCES TO TARGET INTERVALS. THIS MAP
    $ AND THE ENTRIES IN -SLI- FOR WHICH -MOVETO- IS EMPTY , ARE THEN
    $  USED TO PERFORM THE ACTUAL CODE INSERTIONS.

    $ THE APPROACH JUST SKETCHED IS CLUMSY AND CAN PROBABLY BE
    $ REPLACED ALTOGETHER BY USING THE NAME-SPLITTING TECHNIQUE
    $ DESCRIBED IN NL.***. WE SHALL THEREFORE OMIT ALL SPECIFICS
    $ OF THE CODE FOR THIS PHASE.



    END PROCEDURE MOVELOCS ;




    PROCEDURE UPDMODES ;

    $ THIS CONSTITUTES THE FIFTH PHASE OF THE DATA-STRUCTURING PROCESS
    $ THE FINAL MODES OF REAL BASES HAVE BEEN DETERMINED. WE ADJUST
    $ ACCORDINGLY THE MODES OF OCCURRENCES. THIS IS DONE IN THREE
    $ STEPS :

    $ FOR COMPOSITE OBJECTS, FOR WHICH BASED MODES HAVE BEEN GENERATED
    $ AT THE BEGINNING, WE ADJUST THESE MODES USING THE PROCEDURE
    $ SUBSTMD, TO REPLACE REFERENCES TO DROPPED BASES BY THE MODE OF
    $ THEIR ELEMENTS.

    $ FOR ATOMIC VARIABLES, THE ONLY POSSIBLE MODIFICATION TO THEIR
    $ PREVIOUS TYPE IS THAT THEY MAY HAVE BECOME ELEMENTS OF A
    $ BASE. THIS CAN ONLY HAPPEN IF THEY RECEIVE THEIR VALUE BY AS-
    $ SIGNMENT OR BY EXTRACTION . FOR OVARS, THE INFORMATION IS
    $ ALREADY AVAILABLE IN BASE¬ELMTS, AND CAN BE USED TO SET THE
    $ CORRECT FINAL MODE .
    $ FOR IVARS, WE MUST EXAMINE BFROM TO DETERMINE WHETHER AN
    $ -ELEMENT OF BASE- MODE REACHES THEM.

    UPDMCOMPS() ;
    UPDMOVARS() ;
    UPDMIVARS() ;

    RETURN;

    END PROCEDURE UPDMODES ;
```

```
PROCEDURE REFINE ;

$ THIS PROCEDURE APPLIES THE HEURISTICS DESCRIBED IN THE C.A.C.M
$ PAPER, TO CHOOSE BETWEEN LOCAL, REMOTE AND SPARSE REPRESEN-
$ TATIONS FOR BASED OBJECTS. AT THIS POINT, THESE HEURISTICS
$ AMOUNT TO THE FOLLOWING :

$ A) A BASED OBJECT SHOULD BE SPARSE IF IT IS TO BE ITERATED
$ UPON, UNLESS WE CAN SHOW THAT THE OBJECT IS ACTUALLY IDENTICAL
$ WITH ITS BASE.THE ROUTINE -ID¬TO¬BASE- SPOTS SUCH IDENTITIES
$ IN SOME IMPORTANT CASES.

$ B) IF NO ITERATION IS PERFORMED ON AN OBJECT, BUT IT IS
$ SUBJECT TO ALGEBRAIC OPERATIONS (UNION, INTERSECTION, ETC)
$ OR IS PASSED AS A PARAMETER, ASSIGNED AND USED DESTRUCTIVELY,
$ OR INSERTED INTO A LARGER OBJECT, THEN IT SHOULD BE REMOTE.

$ C) IF ONLY DIFFERENTIAL UPDATING OPERATIONS ARE APPLIED
$ TO AN OBJECT, AND IT IS NEVER TRANSMITTED TO ANOTHER BY
$ ASSIGNMENT, INSERTION OR CALL, THEN IT CAN HAVE A LOCAL
$ REPRESENTATION.
$ THE MAP CL¬MODE IS THE STRUCTURE THAT DRIVES THIS PROCEDURE. THE
$ OCCURRENCES IN CL¬MODE(MODE) ARE EXAMINED IN TERMS OF THE OPERA-
$ TIONS WHICH APPLY TO THEM, AND A SINGLE CHOICE (LOCAL, REMOTE,
$ SPARSE) IS MADE FOR ALL OF THEM, THUS PRECLUDING ANY MODE
$ CONVERSION, (WHOSE EXPENSE WE DO NOT KNOW YET HOW TO EVALUATE).


CONST SPARSE, REMOTE, LOCAL ; END ;

(vCLASS := CL¬MODE(MODE))


. . . .
END PROCEDURE REFINE ;
```