SETL NEWSLETTER NO. 207

A SECOND SIMPLIFIED APPROACH TO AUTOMATIC DATA STRUCTURE CHOICE

MICHA SHARIR
8 MAR 1978

THIS NOTE FOLLOWS SETL NEWSLETTER NO. 203, WHERE A NEW APPROACH TO AUTOMATIC DATA-STRUCTURE SELECTION HAS BEEN DESCRIBED. THE APPROACH SUGGESTED THERE, ALTHOUGH IT HAS A RATHER SIMPLE STRUCTURE, SUFFERS FROM SEVERAL DEFFICIENCIES WHICH HAVE LED US TO LOOK FOR AN ALTERNATIVE APPROACH. THE NEW AND PROBABLY IMPROVED APPROACH, TO BE DESCRIBED IN THIS NEWSLETTER, IS CLOSER TO ED SCHONBERG≠S ALGORITHM (TO BE DESCRIBED IN A COMING NEWSLETTER) AND SEEMS TO BE FASTER THAN THE FIRST APPROACH. IN SPITE OF THE DIFFERENCES BETWEEN THESE TWO METHODS, THEY STILL HAVE A SIMILAR OVERALL LOGIC WHICH IS MUCH SIMPLER THAN THAT OF PREVIOUSLY SUGGESTED ALGORITHMS. AMONG THESE SIMPLIFICATIONS ARE: USING THE BFROM AND FFROM MAPS INSTEAD OF VALUE-FLOW MAPS AND DISPENSING ALTOGETHER WITH A PHASE WHICH INSERTS ≠LOCATE≠ INSTRUCTIONS INTO THE CODE.

WE SHALL FIRST DESCRIBE THE NEW AUTOMATIC DATA-STRUCTURE SELECTION ALGORITHM HEURISTICALLY:

(1) INITIALLY, ALL INSTRUCTIONS I IN THE CODE TO BE PROCESSED ARE ANALYZED SEPARATELY. IN THIS ANALYSIS WE PROCEED IN A MANNER DEPENDING ON THE OPCODE OF I AND THE TYPES OF ITS ARGUMENTS, AND GENERATE BASES B1, B2 ... BN, SUCH THAT SOME OR ALL OF THE OCCURENCES IN I CAN BE REPRED IN SUCH A WAY THAT EACH OF THE ABOVE BASES APPEAR IN AT LEAST ONE SUCH REPR, AND SUCH THAT IF THESE REPRS ARE USED THE EXECUTION TIME OF I WILL EITHER REMAIN SUBSTANTIALLY THE SAME, OR ELSE BECOME FASTER.

IN OTHER WORDS, A BASE BI IS GENERATED ONLY IF AT LEAST ONE OCCURENCE IN I ACCESSES ITS ELEMENTS, AND INTRODUCTION OF THIS BASE DOES NOT SLOW I DOWN. I COULD SLOW DOWN IF HASHING OF A VALUE INTO BI IS NEWLY REQUIRED (E.G. IF A NEW VALUE OF AN ELEMENT OF BI MAY HAVE BEEN CREATED IN EXECUTING I), OR IF BASE CONVERSIONS MAY BE NEWLY REQUIRED (E.G. IF DIFFERENT BASES ARE ASSIGNED TO THE ARGUMENTS IN A SET UNION INSTRUCTION).

A SECOND PROPERTY THAT WE REQUIRE THE GENERATED BASES TO POSSESS, IS THAT EVEN AFTER THE INTRODUCTION OF REPRED ARGUMENTS, THE INSTRUCTION I SHOULD BE EQUIVALENT TO I WITH ITS ARGUMENTS HAVING THEIR ORIGINAL FORMS. THUS, IN AN INSTRUCTION ≠C := A + B;≠, IF A IS A SET OF INTEGERS, B IS A SET OF CHARACTERS AND C IS (NECESSARILY) A SET OF GENERAL ELEMENTS, NO BASES ARE GENERATED, FOR IN ORDER FOR THAT INSTRUCTION NOT TO SLOW DOWN, ALL THREE ARGUMENTS MUST BE BASED ON THE SAME BASE, WHOSE ELEMENTS MUST THEREFORE BE OF GENERAL TYPE. THUS, A AND B BECOME SETS OF GENERAL ELEMENTS, CONFLICTING WITH THEIR PREVIOUS TYPES.

THIS RESTRICTION REFLECTS ONE OF THE UNDERLYING PRINCIPLES IN OUR APPROACH, NAMELY: THE TYPES OF VARIABLE OCCURENCES, AS PRODUCED BY THE TYPE FINDER, SHOULD NOT BE MODIFIED DURING AUTOMATIC DATA-STRUCTURE SELECTION. SUCH MODIFICATIONS ARE POSSIBLE IN TWO CASES: (I) DURING THE PRE-PASS, IF TYPES ARE CONVERTED INTO BASED REPRS WHICH ARE NOT EQUIVALENT TO THE ORIGINAL TYPES (AS IN THE ABOVE SET UNION EXAMPLE). (II) BY MERGING BASED REPRS OF TWO OCCURENCES HAVING DIFFERENT TYPES. SUCH A MERGE MAY CAUSE EQUIVALENCING OF TWO BASES B1, B2, WHOSE ELEMENT-MODES ARE NOT EQUAL, SO THAT THE NEW BASE WILL NOT BE REALLY EQUIVALENT NEITHER TO B1 NOR TO B2, AND CONSEQUENTLY THE TYPES OF REPRS BASED ON B1 OR B2 WILL HAVE CHANGED.

IN BOTH CASES, TYPES BECOME MORE GENERAL, AND NEVER MORE RESTRICTED. HENCE, THE NEW TYPES WILL STILL BE AN OVER-ESTIMATION TO THE ACTUAL TYPES, SO THAT THE CODE WILL STILL BE SAFE. HOWEVER, BECAUSE TYPES MAY BECOME LESS SPECIFIC, WE ARE APT TO GENERATE LESS EFFICIENT CODE, IN THE SENSE THAT SOME Q2 INSTRUCTIONS MAY BECOME MORE GENERAL (AND THEREFORE MORE TIME CONSUMING), AND EXTRA TYPE CHECKS AND CONVERSIONS MAY BE REQUIRED.

FOR THESE REASONS, WE PREFER TO MAKE SURE THAT CASES (I) AND (II) WILL NOT OCCUR IN OUR ALGORITHM, AND SO KEEP THE TYPE OF OCCURENCES UNCHANGED (SEE ALSO REMARK (1) AT THE END OF THE ALGORITHM).

NOT ALL GENERATED BASES ACTUALLY SPEED UP THE PROGRAM EXECUTION. THOSE THAT DO NOT ARE USELESS, AND, UNLESS WE CAN LATER MERGE THEM WITH MORE USEFUL BASES, WILL BE SUPPRESSED (SEE (4) BELOW). HOWEVER, THE REASON FOR PROVISIONALLY INTRODUCING THESE EXTRA BASES IS TO ELIMINATE THE NECESSITY TO PROPAGATE BASINGS ACROSS INSTRUCTIONS (SEE (3) BELOW).

GENERATED BASES WHOSE INTRODUCTION SPEED UP THE EXECUTION OF THE INSTRUCTION IN CONNECTION WITH WHICH THEY WERE INTRODUCED WILL BE CALLED EFFECTIVE BASES AND ALL OTHER GENERATED BASES WILL BE CALLED NEUTRAL BASES.

EXAMPLES:

(A) T := S WITH X;

IF T AND S ARE SETS WITH ELEMENTS OF THE SAME TYPE, WHICH IS ALSO EQUAL TO THE TYPE OF X, GENERATE ONE EFFECTIVE BASE B, REPR S AND T AS SET(↑B), AND X AS ↑B.

IF S AND T ARE HOMOGENEOUS TUPLES HAVING THE SAME ELEMENT-TYPE, WHICH IS ALSO EQUAL TO THE TYPE OF X, GENERATE ONE NEUTRAL BASE B, WITH REPRS ANALOGOUS TO THE ABOVE.

(B) Y := F(X);

IF F IS A MAP, WITH A DOMAIN TYPE EQUAL TO THE TYPE OF X, AND A RANGE TYPE EQUAL TO THE TYPE OF Y, GENERATE ONE EFFECTIVE BASE B1 AND ONE NEUTRAL BASE B2, AND PRODUCE THE FOLLOWING REPRS: F: MAP(↱B1) ↱B2; X: ↱B1; Y: ↱B2;

IF THE TYPE OF Y IS NOT EQUAL TO THE RANGE TYPE OF F, DO NOT GENERATE B2, AND IF THE TYPE OF X IS NOT EQUAL TO THE DOMAIN TYPE OF F, DO NOT GENERATE B1.

IF F IS A HOMOGENEOUS TUPLE, AND THE TYPE OF Y IS EQUAL TO THE COMPONENT TYPE OF F, GENERATE ONE NEUTRAL BASE B AND REPR F AS TUP(↱B) AND Y AS ↱B.

IF F IS A STRING OR OF AN AMBIGUOUS TYPE, NO BASES ARE TO BE GENERATED.

(C) Y := X;

UNLESS THE TYPES OF X AND Y ARE UNEQUAL, GENERATE ONE NEUTRAL BASE B, AND REPR X AND Y AS ↱B.

NOTE THAT MOST OF THE RESTRICTIONS IMPOSED IN THE ABOVE EXAMPLES ARE AUTOMATICALLY FULFILLED BY THE FINAL PHASE OF THE TYPE FINDER, WHICH ASSIGNS TO EACH OVARIABLE THE ≠FORWARD≠ TYPE OF ITS IVARIABLES. FOR EXAMPLE, IN (A) ABOVE, IF S IS A SET AND THE TYPE OF ITS ELEMENTS IS EQUAL TO THE TYPE OF X, THEN THE TYPE OF T WILL BE EQUAL TO THAT OF S; SIMILARLY, IN (C) ABOVE, THE TYPE OF Y WILL ALWAYS BE EQUAL TO THE TYPE OF X. HOWEVER, THESE RESTRICTIONS HAVE BEEN STATED ABOVE IN ORDER TO MAKE OUR ALGORITHM AS INDEPENDENT OF THE TYPE FINDER AS POSSIBLE.

(2) AFTER THIS INITIAL BASE GENERATION PHASE, VARIABLE OCCURENCES WILL BE BASED ON EFFECTIVE BASES AND/OR NEUTRAL BASES. OUR ALGORITHM NOW ASSUMES THAT EFFECTIVE BASES, AS WELL AS BASES THAT CAN BE MERGED WITH EFFECTIVE BASES, ARE ADVANTAGEOUS. BASE MERGING WILL BE PERFORMED BY PASSING BASING INFORMATION BETWEEN INSTRUCTIONS ACCORDING TO THE FOLLOWING HEURISTICS:

LET VO1, VO2 BE TWO OCCURENCES OF THE SAME VARIABLE WHICH ARE LINKED BY THE BFROM MAP, AND SUPPOSE THAT WE WANT TO MERGE THE BASE INFORMATION OF VO1 WITH THAT OF VO2. LET REPR1, REPR2 BE THE GENERATED REPRS OF VO1, VO2 RESPECTIVELY; IN ORDER TO MERGE THESE REPRS, VO1 AND VO2 MUST HAVE THE SAME TYPE. IF THIS IS THE CASE, THEN REPR1 AND REPR2 BOTH DESCRIBE OBJECTS HAVING THE SAME TYPE AND BY COMPARING THEIR STRUCTURES WE CAN EQUIVALENCE BASES OR FIND OTHER RELATIONS BETWEEN THEM. A MORE DETAILED DESCRIPTION IS GIVEN IN PHASE 2 OF OUR ALGORITHM BELOW.

(3) THE BASE GENERATION PRE-PASS DESCRIBED ABOVE ENABLES US TO AVOID PROPAGATION OF BASE INFORMATION BETWEEN ARGUMENTS OF THE SAME INSTRUCTION, A TASK WHICH WOULD CALL FOR SOME MESSY ROUTINES, RESEMBLING THE ≠FORWARD≠ AND ≠BACKWARD≠ ROUTINES OF THE TYPE FINDER, AND WOULD ALSO INCREASE THE TIME COMPLEXITY OF OUR ALGORITHM (CF. NL. 203).

BASE PROPAGATION ACROSS INSTRUCTIONS IS ALREADY PERFORMED IMPLICITLY WITHIN THE INITIAL BASE GENERATION PHASE. LATER BASE MERGING NEED BE PERFORMED ONLY ALONG BFROM LINKS. TO CONVINCE OURSELVES THAT THIS IS INDEED THE CASE, WE SHALL CONSIDER FIRST SEVERAL EXAMPLES:

EXAMPLE A.

```
(I1)  S WITH X;          $ S IS A SET
(I2)  V(I) := X;       $ V IS A TUPLE
(I3)  Y := V(J);
(I4)  Z := F(Y);         $ F IS A MAP
```

ASSUME THAT V IS A HOMOGENEOUS TUPLE. THEN THE PRE-PASS WILL COME UP WITH THE FOLLOWING BASINGS (WHERE ONLY B1, B4 ARE EFFECTIVE):

```
S1: SET(↱B1);  X1: ↱B1;
V2: TUP(↱B2);  X2: ↱B2;
Y3: ↱B3;  V3: TUP(↱B3);
Y4: ↱B4;  F4: MAP(↱B4) ↱B5;  Z4: ↱B5;
```

THEN, WHEN BASINGS ARE MERGED ALONG BFROM LINKS, B1 AND B2 WILL BE MERGED (USING THE X-LINK FROM I1 TO I2); B2 AND B3 WILL BE MERGED (VIA THE V-LINK); B3 AND B4 WILL BE MERGED (VIA THE Y LINK). IF WE DID NOT INTRODUCE NEUTRAL BASES FOR I2 AND I3, WE WOULD HAVE TO PROPAGATE BASINGS ACROSS I2 AND I3 IN ORDER TO DEDUCE THAT B1 AND B4 SHOULD BE MERGED.

NOTE ALSO THAT IN THE STEPS WE HAVE JUST DESCRIBED B5 HAS NOT BEEN MERGED WITH AN EFFECTIVE BASE. IF THIS CONDITION PERSISTED, B5 WOULD BE DROPPED DURING THE BASE ADJUSTMENT PHASE.

EXAMPLE B. (S, U AND T ARE ASSUMED TO BE SETS)

```
(I1)  S WITH X;
(I2)  U WITH S;
(I3)  T FROM U;
(I4)  Y FROM T;
```

HERE, AFTER THE PRE-PASS, WE WOULD HAVE:

```
S1: SET(↱B1);  X1: ↱B1;
U2: SET(↱B2);  S2: ↱B2;
T3: ↱B3;  U3: SET(↱B3);
Y4: ↱B4;  T4: SET(↱B4);
```

THEN, USING THE BFROM LINKS, WE WOULD FIRST MERGE THE TWO REPRS ↱B2 AND SET(↱B1) OF THE S OCCURENCES; THIS WILL GIVE US INFORMATION ABOUT THE ELEMENT-MODE OF B2, I.E. ↱B2 = SET(↱B1). THEN WE EQUIVALENCE B2 AND B3 VIA THE U-LINK, AND FINALLY, USING THE T-LINK, WE DEDUCE THAT ↱B3 = SET(↱B4). THIS EXAMPLE SHOWS THAT REPR MERGING HAS TO BE DONE IN SOME RECURSIVE OR TRANSITIVE MANNER, FROM WHICH WE CAN DEDUCE THAT B1 AND B4 OUGHT TO BE EQUIVALENCED ONCE B2 AND B3 ARE EQUIVALENCED, SINCE THE MERGING OF B2 AND B3 CALLS FOR THE MERGING OF THE REPRS SET(↱B1) AND SET(↱B4). THIS WILL BE TAKEN CARE OF DURING THE BASE ADJUSTMENT PHASE OF OUR ALGORITHM.

(4) AFTER MERGING, (EQUIVALENCE CLASSES OF) BASES SHOULD BE SUPPRESSED, IF ALL THE EFFECTIVE BASES IN SUCH A CLASS SUPPORT OCCURENCES OF ONLY ONE COMPOSITE OBJECT (THIS REMARK APPLIES ALSO TO THE CASE IN WHICH THE CLASS CONTAINS NO EFFECTIVE BASES).

(5) A VERY DELICATE ISSUE ARISING IN PREVIOUS DATA-STRUCTURE CHOICE ALGORITHMS WAS THE INSERTION OF ≠LOCATE≠ OPERATIONS INTO THE CODE BEING PROCESSED. THESE OPERATIONS COMPUTE BASE POINTERS FOR ELEMENTS OF A BASE, INSERTING THEM INTO THE BASE IF NECESSARY. THIS PROBLEM IS STILL DELICATE, BUT WE HAVE SHIFTED IT TO THE NAME-SPLITING PHASE OF THE OPTIMIZER (TO BE DESCRIBED IN A COMING NEWSLETTER), WHERE IT IS TREATED AS A SPECIAL CASE OF A GENERAL CONVERSION INSERTION ALGORITHM. WE CAN THEREFORE IGNORE THIS PROBLEM COMPLETELY IN THE PRESENT ALGORITHM, SIMPLIFYING THE ALGORITHM CONSIDERABLY.

(6) A FINAL PHASE OF REPRESENTATION REFINEMENT CHOOSES REMOTE, LOCAL OR SPARSE REPRESENTATIONS FOR BASED OBJECTS. THIS PHASE IS NOT YET FULLY ELABORATED, AND AT THIS MOMENT WE DO NOT SUGGEST ANY NEW IDEAS, BUT CONTINUE TO USE COARSE HEURISTICS PREVIOUSLY SUGGESTED, TO DETERMINE THE DETAILED REPRESENTATION BASED OBJECTS.

THE ABOVE REMARKS SUGGEST A RATHER SIMPLE AUTOMATIC DATA-STRUCTURE SELECTION ALGORITHM. A SKETCH OF SUCH AN ALGORITHM IS GIVEN BELOW.

THE INPUT TO THIS ALGORITHM CONSISTS OF THE DATA FLOW MAPS BFROM AND FFROM, AND THE TYPE MAP ≠TYP≠, WHICH GIVES THE COMPUTED TYPE OF EACH VARIABLE OCCURENCE.

THE OUTPUT OF THE ALGORITHM IS ANOTHER MAP ON OCCURENCES, CALLED ≠OI¬REPR≠, MAPPING EACH OCCURENCE TO A SUGGESTED REPR. THE SYMBOL TABLE IS ALSO UPDATED BY ADDING NEW BASE DEFINITIONS, BUT THE ACTUAL FORM OF REPRED VARIABLES IS NOT MODIFIED TILL THE NAME-SPLITTING PHASE.

1. INITIALIZATION

```
$ UNTIL THE BASE ADJUSTMENT PHASE, REPRS ARE REPRESENTED IN
$ A PROVISIONAL FASHION, AND CARRY A FLAG WHICH INDICATES WHETHER
$ THEY ARE BASED OR NOT. THIS IS DONE IN ORDER TO BYPASS
$ MANIPULATION OF UNBASED REPRS. THUS REPRS ARE REPRESENTED
$ AS PAIRS [REPR¬PART, IS¬BASED], AND ACCESSED BY THE
$ FOLLOWING MACROS:

   MACRO REPR¬PART(REPR);   REPR(1)  ENDM;
   MACRO IS¬BASED(REPR);   REPR(2) = BASED  ENDM;
   MACRO IS¬UNBASED(REPR);   REPR(2) = UNBASED  ENDM;
   MACRO GROSSTYP(REPR);   REPR(1)  ENDM;
   MACRO COMPTYP(REPR);   REPR(2)  ENDM;
   MACRO CTYPN(REPR, N);   COMPTYP(REPR)(N)   ENDM;
   MACRO DOMTYP(REPR);   CTYPN(COMPTYP(REPR), 1)   ENDM;
   MACRO RANGETYP(REPR);   CTYPN(COMPTYP(REPR), 2)   ENDM;
   MACRO BASE¬OF(REPR);   REPR(2)   ENDM;

   CONST BASED; UNBASED; END CONST;  $  AUXILIARY REPR MNEMONICS
$ NOTE THAT A REAL REPR IS REPRESENTED IN MUCH THE SAME WAY AS A
$ TYPE, BUT MAY INVOLVE THE ADDITIONAL ≠GROSS TYPE≠ ELEMENT-OF-A-BASE
$ (DENOTED BY TELMT) WHOSE COMPONENT TYPE IS THE BASE NAME,
$ WHICH CAN BE RETRIEVED BY THE ≠BASE¬OF≠ MACRO GIVEN ABOVE
$ (SEE THE TYPE FINDER FOR MORE INFORMATION).

   AUX¬REPR := ≤ [VO, [TYPE, UNBASED]] : [VO, TYPE] ↛ TYP≥;
   BASES := NL;   $  SET OF ALL BASES
   ELMT¬MODE := NL;   $  A MAP FROM BASES TO THEIR ELEMENT-MODE.
   IS¬EFFECTIVE := NL; $ A MAP ON BASES INDICATING HOW EFFECTIVE
$ A BASE IS. IT CAN HAVE THREE KINDS OF VALUES:
$ (I) FALSE, IF THE BASE IS NEUTRAL.
$ (II) A VARIABLE NAME V, IF THE BASE IS EFFECTIVE, BUT THE ONLY
$ COMPOSITE OBJECT (SET OR MAP) IT SUPPORTS IS V.
$ (III) TRUE, IF AT LEAST TWO COMPOSITE OBJECTS ARE EFFECTIVELY
$ SUPPORTED BY THAT BASE.

   (∨ I ↛ CODE¬INSTS)
      GENBASES(I);
$ FOR EACH INSTRUCTION I GENERATE BASES AS IN (1) ABOVE.
$ THIS IS DONE USING A CASE STATEMENT ON THE OPCODE OF I
$ AND ON THE TYPE OF ITS ARGUMENTS. FOR EACH GENERATED
$ BASE, COMPUTE THE FORM OF ITS ELEMENTS FROM THE TYPES
$ OF THE ARGUMENTS OF I. MODIFY THE AUX¬REPR MAP OF THE BASED
$ ARGUMENTS OF I TO SHOW THE APPROPRIATE BASED REPRESENTATIONS.
$ CLASSIFY EACH GENERATED BASE AS EFFECTIVE OR NEUTRAL.
   END ∨ I;

   BASEDOCCS := ≤ VO : VO IS A BASED OCCURENCE ≥;
```

```
    PROC GENBASES(I);

$ THIS ROUTINE ANALYZES AN INSTRUCTION I, GENERATES INITIAL
$ BASES FOR I AND SETS UP APPROPRIATE BASINGS FOR THE OCCURENCES
$ IN I. THIS IS DONE USING A CASE STATEMENT ON THE OPCODE OF I
$ AND THE TYPES OF ITS ARGUMENTS. WE WILL GIVE BELOW ONLY
$ A FEW TYPICAL CASES

    OCCS := TUPLE OF OCCURENCES IN I;
    [VO1, VO2, VO3] := OCCS;  $ GET FIRST FEW ARGUMENTS

    CASE OPCODE(I) OF

    (Q1¬WITH, Q1¬LESS):

        CASE GROSSTYP(TYP(VO1)) OF

        (TSET):

$ GENERATE AN EFFECTIVE BASE IF THE ELEMENT-TYPE OF BOTH THE
$ COMPOSITE OCCURENCES IN I ARE EQUAL TO THE TYPE OF THE THIRD
$ OCCURENCE.
            IF TYP1 := TYP(VO1) = TYP(VO2) AND
                    COMPTYP(TYP1) = TYP(VO3) THEN
                B := .NEWAT;
                BASES WITH B;
                ELMT¬MODE(B) := [TYP(VO3), UNBASED];
                IS¬EFFECTIVE(B) :=       $ B IS EFFECTIVE
                    IF OI¬NAME(VO1) = OI¬NAME(VO2)
                    THEN OI¬NAME(VO1) ELSE TRUE END;
                ELMTB := [TELMT, B];  $ ELEMENT-OF-B REPR
                AUX¬REPR(VO1) := OI¬REPR(VO2) :=
                        [MAKEREPR(TSET, ELMTB), BASED];
                AUX¬REPR(VO3) := [ELMTB, BASED];
            END IF;

        (UNT) :     $ HOMOGENEOUS TUPLE

$ THE TREATMENT IS COMPLETELY ANALOGOUS TO THE SET CASE, EXCEPT
$ A NEUTRAL BASE RATHER THAN AN EFFECTIVE BASE IS GENERATED.
            IF TYP1 := TYP(VO1) = TYP(VO2) AND
                    COMPTYP(TYP1) = TYP(VO3) THEN
                B := .NEWAT;
                BASES WITH B;
                ELMT¬MODE(B) := [TYP(VO3), UNBASED];
                IS¬EFFECTIVE(B) = FALSE;  $ B IS NEUTRAL
                ELMTB := [TELMT, B];  $ ELEMENT-OF-B REPR
                AUX¬REPR(VO1) := OI¬REPR(VO2) :=
                        [MAKEREPR(UNT, ELMTB), BASED];
                AUX¬REPR(VO3) := [ELMTB, BASED];
            END IF;
```

```
        (KNT):    $ A MIXED, KNOWN LENGTH TUPLE

$ IN THIS CASE WE GENERATE A NEUTRAL BASE FOR EACH COMPONENT
$ IN THE LARGER TUPLE, PROVIDED ELEMENT TYPES ARE THE SAME
$ COMPONENTWISE.

$ ALTHOUGH THIS CASE IS VERY ATYPICAL FOR THE Q1¬WITH OPCODE,
$ IT DEMONSTRATES HOW MIXED TUPLES SHOULD BE PROCESSED IN
$ THIS PRE-PASS.

$ GET THE INDEX OF THE BIGGER TUPLE
            JBIG := IF OPCODE(I) = Q1¬WITH THEN 1 ELSE 2 END;
            COMPS := COMPTYP(TYP(OCCS(JBIG)));
            IF COMPS(↓COMPS) = TYP(V03) THEN
                (∨ K := 1 ... ↓COMPS)
                    B := .NEWAT;
                    BASES WITH B;
                    ELMT¬MODE(B) := [COMPS(K), UNBASED];
                    IS¬EFFECTIVE(B) := FALSE;  $ B IS NEUTRAL
                    ELMTB := [TELMT, B];  $ ELEMENT-OF-B REPR
                    COMPS(K) := ELMTB;
                END ∨;

                AUX¬REPR(V03) := [COMPS(↓COMPS), BASED];
                AUX¬REPR(OCCS(JBIG)) := [MAKEREPR(KNT, COMPS), BASED];
                AUX¬REPR(OCCS(3-JBIG)) :=
                        [MAKEREPR(KNT, COMPS(1:↓COMPS-1)), BASED];
            END IF;

$ WE HAVE NOT INCLUDED THE CASE OF TMAP, SINCE IT IS NOT
$ CLEAR AS YET HOW TO HANDLE THE ≠WITH≠ OPERATION ON MAPS
$ IN THE TYPE FINDER (CONCEIVABLY, THE OVARIABLE NEED NOT
$ BE A MAP UNLESS WE CAN ASSERT THAT THE IVARIABLE IS A PAIR,
$ BOTH OF WHICH COMPONENTS ARE DEFINED). THIS IS ONE OF THE
$ MAIN PROBLEMS CONCERNING MAPS THAT WE HAVE TO RESOLVE,
$ AND WE USE THIS OPPORTUNITY TO DRAW ATTENTION TO IT.

        END CASE;

    . . . . .

    END CASE;

    RETURN;
    END PROC GENBASES;
```

2. BASE MERGING AND ADJUSTMENT

```
$ IN THIS PHASE WE MERGE BASED REPRS OF ANY TWO OCCURENCES
$ LINKED BY A BFROM LINK. THIS MERGE CAN BE DESCRIBED
$ APPROXIMATELY AS FOLLOWS:

$ LET V0 AND V01 BE TWO OCCURENCES WHICH HAVE THE SAME
$ TYPE AND WHICH ARE LINKED BY BFROM.

$ IT FOLLOWS FROM THE MECHANISM OF THE BASE GENERATION PRE-PASS
$ THAT ANY REPR GENERATED AT THIS PHASE CAN ONLY HAVE ONE OF
$ THE FOLLOWING THREE FORMS:

$ (I) IT IS UNBASED.

$ (II) IT HAS THE FORM ↗B.

$ (III) IT HAS THE FORM COMPOSITE(↗B) (FOR SETS AND
$ HOMOGENEOUS TUPLES) OR THE FORM COMPOSITE(↗B1, ↗B2 ... ↗BN)
$ (FOR MIXED TUPLES AND MAPS).

$ THESE INITIAL REPRS WILL NOT BE MODIFIED DURING THIS
$ PHASE. CONSEQUENTLY, THE FOLLOWING CASES MAY ARISE WHEN
$ MERGING THE REPRS OF V0 AND V01:

$ A) IF ONE OF THESE REPRS IS UNBASED, DO NOTHING.

$ B) IF BOTH REPRS ARE OF THE SECOND CATEGORY, THE MERGE
$ SIMPLY EQUIVALENCES THE CORRESPONDING BASES. THIS
$ EQUIVALENCING ACTION CALLS FOR THE MERGING OF THE
$ ELEMENT-MODES OF THE BASES, SO THAT REPR MERGING IS
$ INDEED A RECURSIVE (OR TRANSITIVE) PROCESS. FOR MORE DETAILS,
$ SEE BELOW.

$ C) IF BOTH REPRS ARE OF THE THIRD CATEGORY, THEN THEIR
$ STRUCTURES MUST BE IDENTICAL, WITH POSSIBLY DIFFERENT BASES.
$ IN THIS CASE THE MERGE EQUIVALENCES ALL PAIRS OF
$ CORRESPONDING BASES IN THESE REPRS, AS IS DONE IN B).

$ D) IF ONE REPR IS ↗B, AND THE OTHER IS OF THE THIRD
$ CATEGORY, THEN IF ELMT¬MODE(B) IS UNBASED WE REPLACE IT
$ BY THE SECOND REPR, AND IF IT IS BASED, WE MERGE IT WITH
$ THE SECOND REPR (THIS IS ANOTHER SOURCE OF RECURSION, OR
$ OF TRANSITIVE CLOSURE IN THE MERGING PROCESS).

$ WE SUGGEST THE FOLLOWING IMPLEMENTATION:
$ REPRESENT THE SET ≠BASES≠ OF ALL GENERATED BASES AS A
$ FOREST WHERE EACH EQUIVALENCE CLASS OF BASES IS A TREE WHOSE
$ ROOT IS THE REPRESENTING BASE FOR THAT CLASS. WE CAN THEN USE
$ AN ULTRA-EFFICIENT COMPRESSED BALANCED TREE REPRESENTATION
```

```
$ TO MANIPULATE THIS FOREST. FOR THIS PURPOSE LET ≠REPBASE≠
$ DENOTE THE FATHER MAPPING IN THIS FOREST, AND LET
$ REPBASE .LIM B DENOTE THE ROOT MAPPING (SEE NL. 204).
$ IN ADDITION TO ≠REPBASE≠ WE ALSO MAINTAIN AN AUXILIARY MAP ≠NBASES≠
$ GIVING THE NUMBER OF DESCENDANTS OF EACH ROOT IN THE FOREST,
$ AND THE MAP ≠ELMT¬MODE≠, WHICH HAS TO BE KEPT ONLY FOR THE
$ ROOTS OF THE FOREST. SINCE ELMT¬MODE(B) HAS TO BE KEPT IFF
$ REPBASE(B) IS UNDEFINED, THESE TWO MAPS CAN BE MERGED INTO
$ ONE, TO OBTAIN A COMPACT DATA-STRUCTURE. HOWEVER, FOR CLARITY,
$ WE SHALL DISTINGUISH BETWEEN THESE MAPS IN WHAT FOLLOWS.

$ FOR THE SUPPRESSION OF USELESS BASES WE ALSO NEED MAINTAIN
$ THE ≠IS¬EFFECTIVE≠ MAP AT THE ROOTS OF THE FOREST, COMBINING
$ ITS VALUES AT THE ROOTS OF TWO TREES WHICH ARE TO BE MERGED
$ IN ORDER TO DETERMINE THE EFFECTIVITY OF THE MERGED TREE.
$ THIS MAP CAN ONLY HAVE THE THREE KINDS OF VALUES OUTLINED
$ AT THE PREVIOUS PHASE, AND ACCORDING TO OUR HEURISTIC
$ PRINCIPLE (4), ALL CLASSES FOR WHICH IS¬EFFECTIVE OF THEIR
$ ROOT IS NOT TRUE WILL BE SUPPRESSED IN THE NEXT PHASE
$ OF OUR ALGORITHM. SEE THE CODE BELOW FOR DETAILS.

$ INITIALLY,

    REPBASE := NL;
    NBASES := ≤ [B, 1] : B ↛ BASES≥

$ ESSENTIALLY, THERE ARE TWO OPERATIONS TO BE PERFORMED ON THIS
$ FOREST:

$ (A) COMPUTATION OF THE ROOT OF A GIVEN BASE. THIS IS THE .LIM
$ OPERATOR, AND IS COMPUTED AS EXPLAINED IN NL. 204, EXCEPT THAT
$ HERE THE ≠VIRTUAL FOREST≠ MENTIONED IN NL. 204 CAN BE IDENTICAL
$ WITH THE ACTUAL ONE, SO THAT PATH COMPRESSION CAN BE APPLIED
$ DIRECTLY TO ≠REPBASE≠.

$ (B) BASE EQUIVALENCING. THIS IS ACCOMPLISHED BY A BALANCED
$ LINKING OF TWO TREES INTO ONE, BUT WITH AN ADDITIONAL
$ OPERATION WHICH MERGES THE ELMT¬MODE ENTRIES OF THE ROOTS
$ OF THE LINKED TREES. WHILE DOING SO, WE MAY HAVE TO UPDATE
$ THE ELMT¬MODE ENTRY OF THE NEW ROOT. THIS IS REQUIRED WHEN
$ ELMT¬MODE(ROOT OF LARGER TREE) IS UNBASED, AND
$ ELMT¬MODE(ROOT OF SMALLER TREE) IS BASED. IN THIS CASE WE
$ REPLACE THE UNBASED MODE BY THE BASED ONE, SO THAT FURTHER
$ MERGINGS WITH THIS EQUIVALENCE CLASS OF BASES COULD INDUCE
$ MORE BASE EQUIVALENCES. THE CODE FOR BASE EQUIVALENCING IS
$ AS FOLLOWS:


    PROC EQUIBASE(B1, B2);
$ MOST OF THE CODE GIVEN BELOW IS THE STANDARD BALANCED LINKING
$ OF TWO TREES. WE GIVE IT TO SHOW THE ADDITIONAL MANIPULATION
$ OF ≠ELMT¬MODE≠ AND ≠IS¬EFFECTIVE≠.
```

```
$ WORKPILE IS GLOBAL

    RB1 := REPBASE .LIM B1;
    RB2 := REPBASE .LIM B2;

    IF RB1 /= RB2 THEN
        EFF1 := IS¬EFFECTIVE(RB1);
        EFF2 := IS¬EFFECTIVE(RB2);
$ COMPUTE THE EFFECTIVITY OF THE NEW CLASS
        EFF := CASE EFF1 OF
                (FALSE): EFF2
                (TRUE): TRUE
                ELSE  $ EFF1 IS NOW A VARIABLE NAME
                    CASE EFF2 OF
                    (FALSE): EFF1
                    (TRUE): TRUE
                    ELSE  $ BOTH ARE VARIABLE NAMES
                        IF EFF1 = EFF2 THEN EFF1 ELSE TRUE END
                    END
                END;
        IF NBASES(RB1) > NBASES(RB2) THEN  $ PERFORM ≠BALANCING≠
            REPBASE(RB2) := RB1;
            NBASES(RB1) := NBASES(RB1) + NBASES(RB2);
            IS¬EFFECTIVE(RB1) := EFF;
            IF IS¬BASED(ELMT¬MODE(RB2)) THEN
                IF IS¬UNBASED(ELMT¬MODE(RB1)) THEN
                    ELMT¬MODE(RB1) := ELMT¬MODE(RB2);
                ELSE
                    WORKPILE WITH [ELMT¬MODE(RB1), ELMT¬MODE(RB2)];
                END IF;
            END IF;
        ELSE
              . . .
$ A SYMMETRIC CODE, INTERCHANGING RB1 AND RB2.

        END IF;
    END IF;

    END PROC EQUIBASE;


$ HERE IS THE CODE FOR PHASE 2:


    (∨ VO ∈ BASEDOCCS, VO1 ∈ BFROM≤VO≥ ↑ VO1 IN BASEDOCCS AND
                                TYP(VO) = TYP(VO1))
$ LET US EMPHASIZE AGAIN THAT OUR ALGORITHM INSISTS ON MERGING
$ BASINGS ONLY BETWEEN OCCURENCES OF THE SAME TYPE. THIS IS A
$ RESTRICTION WHICH SIMPLIFIES THE LOGIC OF THE ALGORITHM,
$ AVOIDING SEVERAL ISSUES THAT OTHERWISE WILL ARISE. SEE ALSO
$ REMARK (1) AT THE END OF THE ALGORITHM.
```

```
        WORKPILE := ≤ [AUX¬REPR(V0), AUX¬REPR(V01)]≥;

        (WHILE WORKPILE /= NL)
            [AUX¬REPR1, AUX¬REPR2] FROM WORKPILE;

            IF IS¬UNBASED(AUX¬REPR1) OR IS¬UNBASED(AUX¬REPR2) THEN
                CONTINUE WHILE;
            END IF;
$ NOW BOTH REPRS ARE BASED, SO THAT THEIR CONJUNCTION MIGHT
$ YIELD ADDITIONAL MERGING ACTIONS.
            REPR1 := REPR¬PART(AUX¬REPR1);
            REPR2 := REPR¬PART(AUX¬REPR2);
            GRSTYP1 := GROSSTYP(REPR1);
            GRSTYP2 := GROSSTYP(REPR2);

            IF GRSTYP1 = GRSTYP2 THEN

                IF GRSTYP1 = TELMT THEN
$ BOTH REPRS ARE ELEMENT-OF-BASE. EQUIVALENCE THEIR BASES
$ WHICH ARE THE COMPONENT-TYPE OF THESE REPRS. THIS
$ EQUIVALENCING MAY TRIGGER THE MERGING OF THE ELEMENT-MODES
$ OF THESE BASES.
                    EQUIBASE(BASE¬OF(REPR1), BASE¬OF(REPR2))];
                ELSEIF GRSTYP1 = TMAP THEN
$ IF A MAP IS REPRED AS A BASED MAP, ITS DOMAIN TYPE AND RANGE
$ TYPE ARE BOTH BASE-ELEMENT TYPES, SO THAT WE HAVE TO
$ EQUIVALENCE THESE BASES.
                    EQUIBASE(BASE¬OF(DOMTYP(REPR1)),
                             BASE¬OF(DOMTYP(REPR2)));
                    EQUIBASE(BASE¬OF(RANGETYP(REPR1)),
                             BASE¬OF(RANGETYP(REPR2)));
                ELSEIF GRSTYP1 = KNT THEN
$ A SIMILAR PROPERTY IS POSSESSED BY MIXED TUPLES.
                    (∨ I := 1 ... ↓COMPTYP(REPR1))
                        EQUIBASE(BASE¬OF(CTYPN(REPR1, I)),
                                 BASE¬OF(CTYPN(REPR2, I)));
                    END ∨;
                ELSE
$ FOR SETS AND HOMOGENEOUS TUPLES, ONLY ONE BASE EQUIVALENCING
$ NEED BE PERFORMED.
                    EQUIBASE(BASE¬OF(COMPTYP(REPR1)),
                             BASE¬OF(COMPTYP(REPR2)));
                END IF;

            ELSEIF GRSTYP1 = TELMT THEN
                RB1 := BASE¬OF(REPR1);
                IF IS¬UNBASED(ELMT¬MODE(RB1)) THEN
                    ELMT¬MODE(RB1) := REPR2;
                ELSE
                    WORKPILE WITH [ELMT¬MODE(RB1), AUX¬REPR2];
                END IF;
```

```
            ELSE  $ NOW GRSTYP2 MUST BE TELMT
                RB2 := BASE¬OF(REPR2);
                IF IS¬UNBASED(ELMT¬MODE(RB2)) THEN
                    ELMT¬MODE(RB2) := REPR1;
                ELSE
                    WORKPILE WITH [ELMT¬MODE(RB2), AUX¬REPR1];
                END IF;
            END IF;

        END WHILE;
    END ∨;
```

3. BASE AND REPR ADJUSTMENT

THIS PHASE IS A ≠CLEAN-UP≠ PHASE WHICH SUPPRESS USELESS BASES, COMPUTES THE OI¬REPR MAP FOR ALL OCCURENCES, AND ENTERS THE SURVIVING BASES INTO THE SYMBOL TABLE. IT THUS CONSISTS OF THE FOLLOWING THREE SUBPHASES:

(A) WE FIRST SUPPRESS (EQUIVALENCE CLASSES OF) BASES THAT HAVE NOT TURNED OUT TO BE USEFUL, ACCORDING TO THE HEURISTIC PRINCIPLE (4) ABOVE. EACH DROPPABLE EQUIVALENCE CLASS IS FLAGGED AS SUCH, AND ANY REPR CONTAINING A BASE B1 IN SUCH A CLASS SHOULD BE MODIFIED SO THAT EACH ↱B1 APPEARANCE IN IT IS REPLACED BY ELMT¬MODE(REPBASE .LIM B1). THE OUTPUT OF THIS PHASE IS A SET ≠DROPPABLES≠ CONTAINING ALL REPRESENTING DROPPABLE BASES.

```
    REPBASES := BASES - DOM REPBASE;  $ SET OF ALL ROOT BASES
    DROPPABLES := ≤ RB ↱ REPBASES ↑ IS¬EFFECTIVE(RB) /= TRUE≥;
```

(B) NEXT, WE ITERATE OVER ALL OCCURENCES, COMPUTING THE OI¬REPR MAP. FOR EACH OCCURENCE VO, OI¬REPR(VO) IS A REAL REPR WHICH IS OBTAINED FROM AUX¬REPR(VO) BY REPLACING BASES BY THEIR REPRESENTATIVES, OR DROPPING THEM AS DESCRIBED IN (A) ABOVE. THE CODE IS:

```
    SEENDROPS := NL;
$ A GLOBAL SET OF ALL DROPPABLE BASES B FOR WHICH THE REAL
$ ELMT¬MODE(B) HAS ALREADY BEEN COMPUTED.
    (∨ VO ↱ BASEDOCCS)
        REPR := REPR¬PART(AUX¬REPR(VO));
        OI¬REPR(VO) := REAL¬REPR(REPR);
    END ∨;
```

```
    PROC REAL¬REPR(REPR);
$ AT THIS POINT, REPR IS BASED

    GRS := GROSSTYP(REPR);
    CASE GRS OF

    (TELMT):
        RB := REPBASE .LIM BASE¬OF(REPR);
        IF RB IN DROPPABLES THEN
            IF RB IN SEENDROPS THEN
                RETURN ELMT¬MODE(RB);
            ELSE
                SEENDROPS WITH RB;
                REPRX := ELMT¬MODE(RB);
                IF IS¬UNBASED(REPRX) THEN
                    RETURN ELMT¬MODE(RB) := REPR¬PART(REPRX);
                ELSE
                    RETURN ELMT¬MODE(RB) :=
                        REAL¬REPR(REPR¬PART(REPRX));
                END IF;
            END IF;
        ELSE
            RETURN [TELMT, RB];
        END IF;

    (KNT):
        RETURN MAKEREPR(KNT,
            [REAL¬REPR(COMPTYP(REPR)(I)) :
                I := 1 ... ↓COMPTYP(REPR)]);

    (TMAP):
        RETURN MAKEREPR(TMAP,
            MAKEREPR(KNT, [REAL¬REPR(DOMTYP(REPR)),
                           REAL¬REPR(RANGETYP(REPR))]));

    ELSE  $ SAME TREATMENT FOR SETS AND HOMOGENEOUS TUPLES
        RETURN MAKEREPR(GRS, REAL¬REPR(COMPTYP(REPR)));
    END IF;

    END PROC REAL¬REPR;
```

(C) THE THIRD STEP, ENTERING THE SURVIVING REPRESENTATIVE BASES (I.E. ALL ELEMENTS OF THE SET REPBASES - DROPPABLES) INTO THE SYMBOL TABLE, IS SIMPLE AND TECHNICAL AND WE OMIT ITS DESCRIPTION HERE.

REMARKS:
---

(1) THE TRANSITIVE CLOSURE OR RECURSION IN THE MERGING PROCEDURE IS ALWAYS FROM ≠MORE COMPOSITE≠ BASES TO ≠MORE PRIMITIVE≠ ONES. I.E. EQUIVALENCING TWO BASES WHOSE ELEMENT-MODES ARE COMPOSITE CAN CAUSE BASES APPEARING IN THESE MODES TO BE EQUIVALENCED TOO, AS IN EXAMPLE B ABOVE. HOWEVER, EQUIVALENCING IS NOT INDUCED IN THE OPPOSITE WAY. FOR EXAMPLE:

EXAMPLE C.

```
S WITH X;  $ S: SET(↱B1);  X: ↱B1;
U WITH S;  $ U: SET(↱B2);  S: ↱B2;

. . . .

T WITH X;  $ T: SET(↱B3);  X: ↱B3;
V WITH T;  $ V: SET(↱B4);  T: ↱B4;
```

IN THIS EXAMPLE, B1 AND B3 ARE EQUIVALENCED IN VIEW OF THE X-LINK, BUT B2 AND B4 ARE NOT MERGED. THIS APPROACH IS PROBABLY DESIRABLE, SINCE SUCH A MERGING WILL NOT IMPROVE THE EXECUTION OF THE ABOVE CODE FRAGMENT, BUT MAY MAKE U AND V SPARSE OVER THE MERGED BASE (OF COURSE, FURTHER INFORMATION MAY MAKE US MERGE B2 AND B4, E.G. AN INSTRUCTION SUCH AS ≠IF S IN V THEN ...≠).

(2) AS FOR ANY RECURSIVE OR TRANSITIVE-CLOSURE MECHANISM, WE MUST GUARANTEE CONVERGENCE OF THE MERGING PROCESS. SINCE THE NUMBER OF GENERATED BASES IS FINITE, DIVERGENCE CAN OCCUR ONLY IF THERE EXIST CYCLIC DEPENDENCIES BETWEEN BASES, THE SIMPLEST OF WHICH COULD BE: B1: BASE(SET(↱B1)). IF SUCH A CONFIGURATION OCCURS AND B1 IS EQUIVALENCED WITH B2: BASE(SET(↱B2)), THEN THE MERGING PROCESS MIGHT REPEAT EQUIVALENCING OPERATIONS INVOLVING B1 AND B2 INFINITELY MANY TIMES. ALSO, DURING THE BASE-DROPPING PHASE, IF B1 IS DROPPABLE THEN WE MIGHT ATTEMPT TO REPLACE EACH ↱B1 APPEARANCE IN A REPR BY SET(↱B1), WHICH OBVIOUSLY LEADS TO DIVERGENCE.

WE CLAIM, HOWEVER, THAT SUCH SITUATIONS WILL NEVER OCCUR, PROVIDED THAT THE TYPE FINDER FUNCTIONS PROPERLY. INDEED, A CYCLIC DEPENDENCY CAN BE DERIVED ONLY BY BASE MERGING ALONG A CYCLIC EXECUTION PATH, AND ONLY IF THERE IS A CYCLIC TYPE DEPENDENCY ALONG THIS PATH, AS IN THE LOOP (↻) X WITH X; END; BUT IN THIS SITUATION THE TYPE FINDER WILL PRODUCE DIFFERENT TYPES FOR THE OVARIABLE AND THE IVARIABLE OF THE STATEMENT IN THE LOOP, NAMELY - SET(GENERAL) AND GENERAL RESPECTIVELY (RECALL THAT OVARIABLES ARE ASSIGNED THE FORWARD TYPE OF THEIR IVARIABLES IN THE FINAL PHASE OF THE TYPE FINDER). HENCE, NO BASE MERGING WILL TAKE PLACE ALONG THIS LOOP.

IT MAY ALSO BE NOTED THAT IF THE BASE GENERATED FOR THIS STATEMENT IS NOT DROPPED (CALL IT B1), THEN THE NAME-SPLITTING PHASE WILL TRANSFORM THE ABOVE LOOP INTO

(v) XB := XA; XA WITH XB; END;

WHERE WE HAVE B1: BASE(GENERAL); XB: ↱B1; XA: SET(↱B1); AND THE ASSIGNMENT XB := XA; IS A ≠LOCATE≠ OF THE VALUE OF XA IN B1.

IT SHOULD ALSO BE NOTED THAT IN ORDER TO ENSURE SUCH A PROPER OPERATION OF THE TYPE FINDER, ITS ABOVE-MENTIONED FINAL PHASE SHOULD COMPUTE THE OVARIABLES≠ TYPES WITH NO NESTING-LEVEL LIMIT, SO AS TO BREAK ANY POSSIBLE TYPE DEPENDENCY.

FOR EXAMPLE, IF WE PROCESS THE LOOP (v) X := ≤X≥; END; BOTH X OCCURENCES MAY GET THE TYPE SET(SET(... SET(GEN)...)) WITH A MAXIMAL NESSTING LEVEL, UNLESS WE INCREASE, IN THE FINAL PHASE, THE NESTING LEVEL OF THE OVARIABLE BY 1.

(3) RETURN FOR THE MOMENT TO EXAMPLE B ABOVE. IN IT THERE OCCUR TWO LINKED OCCURENCES OF S, ONE OF WHICH IS REPRED SET(↱B1) AND THE OTHER ↱B2. AT A FIRST GLANCE IT SEEMS THAT WE OUGHT TO PRODUCE A COMMON REPR FOR THESE OCCURENCES, BUT A BETTER CHOICE WOULD HAVE BEEN TO LEAVE THESE REPRS AS THEY ARE. THEN, AFTER BASE MERGING AND NAME-SPLITTING, THE CODE WILL BE TRANSFORMED INTO:

```
      SA WITH X;        $ SA: SET(↱B1); X: ↱B1;
(A1)  SB := SA;
      U WITH SB;        $ U: SET(↱B2); SB: ↱B2;
      TB FROM U;        $ U: SET(↱B2); TB: ↱B2;
(A2)  TA := TB;
      Y FROM TA;        $ TA: SET(↱B1); Y: ↱B1;
```

WHERE A1 IS A BASE ≠LOCATE≠ OF THE VALUE OF SA IN B2 AND A2 IS ESSENTIALLY DEREFERENCING THE VALUE OF TB FROM A POINTER TO AN ELEMENT OF B2 TO A POINTER TO THE SET VALUE OF TA (NO TYPE CHECKING IS NECESSARY).

THIS CODE HANDLING REFLECTS ONCE MORE THE BASIC PHILOSOPHY OF THE FINAL PHASES OF THE OPTIMIZER, NAMELY: REPRS AND TYPES SHOULD BE ASSIGNED TO OCCURENCES IN SUCH A WAY THAT EACH INSTRUCTION WILL BE EXECUTED IN THE MOST EFFICIENT MANNER, AND ANY TYPE OR REPR CHECKS AND CONVERSIONS SHOULD BE MOVED AND INSERTED INTO THE CODE IN SOME OPTIMAL PLACE PRECEEDING THAT INSTRUCTION.

4. BASING REFINEMENT

THIS PHASE CAN ALSO BE TAKEN FROM SCHONBERG≠S ALGORITHM. AS NOTED BEFORE, WE HAVE NOT YET ATTEMPTED TO IMPROVE THIS PHASE.

FINAL REMARKS:

(1) OUR ALGORITHM MERGES REPRS ONLY IF THEY HAVE THE SAME TYPE, AND COSEQUENTLY EQUIVALENCES BASES ONLY IF THEY HAVE THE SAME ELEMENT-TYPE. FOR EXAMPLE, ≠SET OF INTEGERS≠ AND ≠SET OF GENERALS≠ ARE CONSIDERED AS DISTINCT TYPES. HENCE, IF THERE IS A LINK BETWEEN TWO OCCURENCES HAVING SUCH TYPES, THEIR BASES WILL NOT BE MERGED, AND EVENTUALLY WE SHALL HAVE TO CONVERT FROM ONE BASE TO THE OTHER. IT IS NOT CLEAR WHETHER THIS APPROACH IS TO BE PREFERRED, AND THERE MAY BE A POINT IN MERGING BASES OF THIS KIND, EVEN THOUGH THIS MAY LEAD TO CREATION OF ADDITIONAL TYPE CHECKS AND CONVERSIONS WHICH WOULD NOT HAVE BEEN OTHERWISE NEEDED. AT ANY RATE, OUR APPROACH IS THE SIMPLEST OF ALL SUCH ALTERNATIVES, AND SHOULD BE QUITE ACCEPTABLE IN MOST CASES.

(2) USER-SUPPLIED BASINGS APPEAR ALREADY IN THE TYP MAP, AND SO ARE PART OF THE INPUT TO THE ALGORITHM. HOWEVER, THEY RAISE SEVERAL PROBLEMS. FOR EXAMPLE, IT IS NOT CLEAR WHETHER WE WANT TO MERGE TWO USER-SUPPLIED BASES, OR ALWAYS KEEP THEM DISTINCT. AN ARGUMENT FOR NOT MERGING THEM IS THAT BY DOING SO WE MAY CAUSE SOME BASED OBJECTS TO BECOME SPARSE OVER THE MERGED BASE, WHICH MAY WELL HAVE BEEN THE REASON WHY THE USER SUPPLIED TWO DISTINCT BASES INSTEAD OF ONE.

NOTE THAT THIS APPROACH CALLS FOR MAINTAINING YET ANOTHER MAP AT THE ROOTS OF OUR FOREST, INDICATING WHICH USER-SUPPLIED BASE, IF ANY, IS A MEMBER OF THE CORRESPONDING CLASS. IN THIS WAY, WE CAN AVOID LINKING TWO CLASSES TOGETHER IF THEY CONTAIN DIFFERENT USER-SUPPLIED BASES. THIS ALSO CALLS FOR SOME MODIFICATIONS OF THE BASE GENERATION PRE-PASS. HOWEVER, FOR THE SAKE OF CLARITY, WE LEAVE OUT THE DETAILS OF SUCH MODIFICATIONS.

(3) THE RUNNING TIME OF OUR NEW ALGORITHM SHOULD BE SHORTER THAN THAT OF THE ALGORITHM DESCRIBED IN NL. 203. IN THE NEW ALGORITHM, BASE PROPAGATION IS ACCOMPLISHED BY A SINGLE PASS THROUGH THE BFROM LINKS BETWEEN BASED OCCURENCES, WITH VERY EFFICIENT PROCESSING OF EACH SUCH LINK. THE ONLY TIME CONSUMING PART OF OUR ALGORITHM IS THE MANIPULATION OF COMPLETELY USELESS DROPPABLE BASES AND CORRESPONDING BASED OCCURENCES. IT IS NOT CLEAR HOW TO ESTIMATE THIS ADDITIONAL TIME USAGE, WHICH DEPENDS HEAVILY ON THE NATURE OF THE PROGRAM BEING ANALYZED. IN THE EXAMPLE SHOWN BELOW, MOST OF THE GENERATED BASES EITHER ARE MERGED WITH

EFFECTIVE BASES, OR ARE NEEDED TO EQUIVALENCE OTHER BASES. IN ANOTHER PROGRAM, MORE ORIENTED TOWARD CREATION OF NEW PRIMITIVE VALUES, THE NUMBER OF USELESS BASES MIGHT INCREASE SIGNIFICANTLY.

SINCE MANY MORE BASES ARE GENERATED, THE PRESENT ALGORITHM CONSUMES MORE SPACE THAN THE PREVIOUS ONE. THIS SPACE USAGE CAN BE REDUCED BY PAINSTAKINGLY CAREFUL PROGRAMMING (E.G. BY FOLDING THE PRE-PASS INTO THE BASE MERGING PHASE IN AN APPROPRIATE MANNER), BUT THEN THE ALGORITHM WOULD LOSE ITS CLARITY AND WOULD TEND TO RESEMBLE THE PREVIOUS ALGORITHM.

NOTE, HOWEVER, THAT THE SPACE COMPLEXITY OF THE OLDER DATA-STRUCTURE CHOICE ALGORITHMS (OF SCHONBERG, SCHWARTZ AND LIU) WHICH USE VALUE-FLOW, IS AT LEAST THE CARDINALITY OF ALL THE VALUE-FLOW MAPS, IN COMPARISON WITH WHICH THE SPACE COMPLEXITY OF OUR ALGORITHM IS RATHER MODEST.

EXAMPLE:

CONSIDER THE FOLLOWING TOPOLOGICAL SORT PROGRAM:

```
    PROGRAM SAMPLE;
$ TOPOLOGICAL SORT OF A GIVEN GRAPH, ASSUMING THERE ARE NO CYCLES.
 1  NODES := NL;
 2  CESOR := NL;

 3  (DOING READ A,B; WHILE A /= OM)
 4      NODES WITH A;
 5      NODES WITH B;
 6      CESOR WITH [A, B];
    END;

 7  PRINT TOPSORT(NODES, CESOR);
    STOP;
    END PROGRAM SAMPLE;


 8  PROC TOPSORT(NODES, CESOR);

 9  NUMPREV := ≤ [N, 0] : N ∈ NODES≥;
10  (∨ [N, M] ∈ CESOR)
11      NUMPREV(M) + 1;
    END ∨;

12  NOPREV := ≤ N ∈ NODES ↑ NUMPREV(N) = 0≥;
13  SORTED := NULT;
```

```
14    (WHILE NOPREV /= NL)
15        N FROM NOPREV;
16        SORTED WITH N;
17        (∨ M ↱ CESOR≤N≥)
18            NUMPREV(M) - 1;
19            IF NUMPREV(M) = 0 THEN
20                NOPREV WITH M;
              END IF;
          END ∨;
      END WHILE;

21    RETURN SORTED;
      END PROC TOPSORT;
```

AFTER THE BASE GENERATION PHASE, WE OBTAIN THE FOLLOWING INITIAL BASINGS:

```
 1  NODES: SET(↱B1);
 2  CESOR: SET(↱B2);
 3  A,B: UNBASED;
 4  NODES: SET(↱B4);  A: ↱B4;
 5  NODES: SET(↱B5);  B: ↱B5;
 6  CESOR: SET([↱B61, ↱B62]);  A: ↱B61;  B: ↱B62;
    (THIS IS A FOLDING OF THE REPRS ACTUALLY PRODUCED. THIS
    INSTRUCTION EXPANDS IN THE Q1 CODE INTO  (6A) T := [A, B];
    (6B) CESOR WITH T; FROM WHICH THE FOLLOWING REPRS WILL BE
    PRODUCED: (6A) A: ↱B61; B: ↱B62; T: [↱B61, ↱B62];
    (6B) CESOR: SET(↱B6); T: ↱B6; USING THE T LINK BETWEEN THESE
    TWO INSTRUCTIONS, WE OBTAIN THE REPRS GIVEN ABOVE.)
 7  (IMPLIED ARGOUT):  OVARIABLE, IVARIABLE (RETURN-VALUE): ↱B7;
 8  (IMPLIED ARGINS): NODES(FORMAL), NODES(ACTUAL): ↱B81;
                      CESOR(FORMAL), CESOR(ACTUAL): ↱B82;
 9  NUMPREV: SET([↱B91, ↱B92]);  N: ↱B91;  NODES: SET(↱B91);
10  N: ↱B101; M: ↱B102;  CESOR: SET([↱B101, ↱B102]);
    (A SIMILAR REMARK TO THE ONE MADE AT LINE 6 ABOVE APPLIES HERE,
    WITH AN ADDITIONAL GENERATED BASE B10).
11  THIS INSTRUCTION IS EXPANDED AS FOLLOWS:
    T := NUMPREV(M); $ NUMPREV: MAP(↱B111) ↱B112; M: ↱B111; T: ↱B112;
    T := T + 1;  $ BOTH T ARE UNBASED
    NUMPREV(M) := T; $ NUMPREV: MAP(↱B113) ↱B114; M: ↱B113; T: ↱B114;
12  NODES, NOPREV: SET(↱B121); NUMPREV: MAP(↱B121) ↱B122; N: ↱B121;
13  SORTED: TUP(↱B13);  (I.E. A HOMOGENEOUS TUPLE)
14  NOPREV: SET(↱B14);
15  NOPREV: SET(↱B15);  N: ↱B15;
16  SORTED: TUP(↱B16);  N: ↱B16;
17  CESOR: MAP(↱B171) ↱B172;  M: ↱B172;  N: ↱B171;
18  EXPANDS AS LINE 11, BUT ESSENTIALLY WE HAVE:
    NUMPREV: MAP(↱B181) ↱B182;  M: ↱B181;
19  NUMPREV: MAP(↱B191) ↱B192;  M: ↱B191;
20  NOPREV: SET(↱B20);  M: ↱B20;
21  RETURN VALUE, SORTED: ↱B21;
```

NOW, IF MERGING IS PERFORMED ROUGHLY IN THE ORDER OF THE CODE, THE FOLLOWING MERGING ACTIONS WILL TAKE PLACE:

B1, B4, B5, B61, B62 ARE EQUIVALENCED, IN VIRTUE OF VARIOUS LINKS BETWEEN INSTRUCTIONS 1,4,5,6. LET B1 BE THE BASE REPRESENTING THIS CLASS.

ELMT¬MODE(B2) := [↗B1, ↗B11]; IN VIEW OF THE CESOR2-CESOR6 LINK.

IN VIEW OF THE LINK OF ≠NODES≠ TO LINE 8, WE PUT ELMT¬MODE(B81) := SET(↗B1) (ACTUALLY, THERE ARE TWO LINKS TO NODES(ACTUAL) AT LINE 8, ONE FROM NODES1 AND ONE FROM NODES5. HENCE, WE FIRST PUT ELMT¬MODE(B81) := SET(↗B1) AND THEN MERGE IT WITH SET(↗B5), THIS REDISCOVERING AN EQUIVALENCE OF B1 AND B5.

SIMILARLY, USING THE CESOR LINKS TO LINE 8, WE PUT ELMT¬MODE(B82) := SET(↗B2).

USING THE NODES8-NODES9 LINK, WE MERGE ELMT¬MODE(B81) WITH SET(↗B91) AND THUS EQUIVALENCE B91 WITH B1.

USING THE CESOR8-CESOR10 LINK, WE MERGE ELMT¬MODE(↗B82) WITH SET([↗B101, ↗B102]) AND EQUIVALENCE B101, B102 WITH B1 (NOTE THAT THIS EQUIVALENCE IS DERIVED RECURSIVELY: THE INITIAL MERGING OF SET(↗B10) WITH ↗B82 LEADS TO MERGING SET(↗B10) WITH SET(↗B2), WHICH LEADS TO EQUIVALENCING B101 AND B102 WITH B1).

USING THE M10-M11 LINK AND THE NUMPREV9-NUMPREV11 LINK, WE EQUIVALENCE B111, B113 WITH B1 AND B112, B114 WITH B92.

SIMILARLY, USING THE NODES AND NUMPREV LINKS TO LINE 12, WE EQUIVALENCE B121 WITH B1 AND B122 WITH B92.

FOLLOWING FURTHER LINKS, WE EQUIVALENCE B13, B14, B15, B16, B171, B172, B181, B191, B20 WITH B1, AND B182, B192 WITH B92.

USING THE LINKS OF ≠SORTED≠ TO LINE 21 AND THE IMPLICIT RETURN-VALUE LINK TO LINE 7, WE EQUIVALENCE B7 WITH B21, AND PUT ELMT¬MODE(B7) := TUP(↗B1).

THUS, AT THE END OF THE BASE MERGING PHASE, WE WILL BE LEFT WITH SIX EQUIVALENCE CLASSES, HAVING B1, B2, B81, B82, B92 AND B7 AS THEIR REPRESENTATIVES, WITH THE FOLLOWING ELEMENT-MODES:

```
ELMT¬MODE(B1) = GENERAL;
ELMT¬MODE(B2) = [↗B1, ↗B11];
ELMT¬MODE(B81) = SET(↗B1);
ELMT¬MODE(B82) = SET(↗B2);
ELMT¬MODE(B92) = INTEGER;
ELMT¬MODE(B7) = TUP(↗B1);
```

ALL BASES EXCEPT B1 ARE THEN SUPPRESSED, BECAUSE NONE OF THE BASES IN THEIR EQUIVALENCE CLASSES ARE EFFECTIVE. B92 IS THE ONLY COMPLETELY USELESS BASE; THE OTHER FOUR DROPPABLE BASES SERVED AS ≠TRANSMITTERS≠ OF BASING INFORMATION, AND PHASE 3 WILL REPLACE ALL REFERENCES TO THESE BASES BY APPROPRIATE SUBSTITUTIONS OF THEIR ELEMENT-MODES, ALL OF WHICH ARE BASED (NOTE THAT TWO SUCH SUBSTITUTIONS ARE REQUIRED WHEN REPLACING ↱B82 BY AN APPROPRIATE ELEMENT-MODE; THE FIRST ONE YIELDS SET(↱B2), BUT SINCE B2 IS ALSO DROPPABLE, WE NEED A SECOND SUBSTITUTION, YIELDING SET([↱B1, ↱B1])).

IT SHOULD BE OBSERVED THAT NO USER SUPPLIED REPR DECLARATIONS WERE USED TO DERIVE ALL THIS INFORMATION. THE BEST THAT SUCH A DECLARATION COULD DO IS TO CHANGE THE FORM OF B1 FROM BASE OF GENERALS (WHICH IS THE FORM OUR ALGORITHM WOULD USE) TO BASE OF ATOMS, SAY, BUT THIS CHANGE WILL HAVE LITTLE OR NO EFFECT ON EXECUTION EFFICIENCY.