SETL NEWSLETTER NO. 203
A SIMPLIFIED APPROACH TO AUTOMATIC DATA STRUCTURE CHOICE

MICHA SHARIR
5 JAN 1978

IN THIS NOTE, WE WILL DESCRIBE A NEW APPROACH TO AUTOMATIC DATA-STRUCTURE SELECTION. THE BASIC IDEAS ARE TAKEN FROM PREVIOUSLY SUGGESTED ALGORITHMS BY SCHWARTZ, LIU AND SCHONBERG. HOWEVER, THERE ARE SEVERAL MAJOR CHANGES THAT MAKE THE NEW ALGORITHM MUCH SIMPLER. AMONG THESE ARE: USING BFROM AND FFROM MAPS INSTEAD OF VALUE-FLOW MAPS, AND DISPENSING ALTOGETHER WITH A PHASE WHICH INSERTS ≠LOCATE≠ INSTRUCTIONS INTO THE CODE.

LET US FIRST DESCRIBE OUR AUTOMATIC DATA-STRUCTURE SELECTION ALGORITHM HEURISTICALLY:

(1) VARIABLE OCCURENCES IN A CODE TO BE PROCESSED ARE DIVIDED INTO THREE CATEGORIES:

(A) PRO-BASING OCCURENCES: THESE ARE OCCURENCES IN INSTRUCTIONS THAT WOULD HAVE BEEN EXECUTED MUCH FASTER IF THESE OCCURENCES WERE PROPERLY BASED. WE REQUIRE THAT THE GROSS TYPE OF EACH SUCH OCCURENCE (AND SOMETIMES ALSO OF ITS SIBLING OCCURENCES IN THE SAME INSTRUCTION) BE UNAMBIGUOUS, AND THAT IT SUGGEST A BASING (WITH THE EXCEPTION OF PRO-BASING OCCURENCES THAT ARE TO BE REPRED AS ELEMENT-OF-BASE), FOR EXAMPLE, IN A Q1-ADD INSTRUCTION, EACH OCCURENCE IS PRO-BASING, IF ALL ARE SETS OR MAPS, BUT NOT IF THEY ARE TUPLES, INTEGERS, OR STRINGS. IN THE INSTRUCTION ≠T := S WITH X;≠ EACH OCCURENCE IS PRO-BASING IF S AND T ARE SETS, BUT NOT IF THEY ARE TUPLES. IN THE INSTRUCTION ≠F(X) := Y;≠ Y IS NOT PRO-BASING; X AND F ARE PRO-BASING IF F IS A MAP, BUT NOT IF F IS A TUPLE OR A STRING, AND CERTAINLY NOT IF F IS AMBIGUOUS.

(B) NEUTRAL OCCURENCES (USUALLY WITH RESPECT TO SOME GIVEN BASING): THESE ARE OCCURENCES, GENERALLY HAVING UNAMBIGUOUS TYPE, IN INSTRUCTIONS WHOSE EXECUTION SPEED IS VIRTUALLY UNAFFECTED IF THESE OCCURENCES HAVE THE GIVEN BASING, OR ARE UNBASED. THESE OCCURENCES NEITHER SUGGEST SUCH A BASING NOR FORBID IT, AND WHETHER THEY SHOULD BE BASED OR NOT DEPENDS ON THEIR LINKS TO PRO-BASING OCCURENCES. FOR EXAMPLE, IN ≠Y := F(X);≠ Y IS NEUTRAL WITH RESPECT TO ANY GIVEN BASING, EVEN IF IT HAS AN AMBIGUOUS TYPE, AS LONG AS F IS A MAP OR A TUPLE. IF F IS AMBIGUOUS, OR A STRING, THEN Y IS NOT NEUTRAL, BUT ANTI-BASING (SEE (C) BELOW). IN THE ASSIGNMENT ≠Y := X;≠ BOTH X AND Y ARE NEUTRAL FOR A REPR ELEMENT-OF-BASE, NO MATTER WHAT THEIR TYPE IS, BUT FOR OTHER REPRS, THEY ARE NEUTRAL ONLY IF THEIR TYPES ARE NOT AMBIGUOUS; IF THEY ARE, THEN THESE OCCURENCES ARE ANTI-BASING.

(C) ANTI-BASING OCCURENCES: THESE ARE OCCURENCES IN INSTRUCTIONS WHOSE EXECUTION WILL BE SLOWED DOWN CONSIDERABLY IF THESE OCCURENCES ARE BASED. IN THIS CATEGORY WE ALSO INCLUDE OCCURENCES HAVING AMBIGUOUS TYPE, WHICH BARS ANY MEANINGFUL BASING, FOR EXAMPLE, AN OCCURENCE OF ANY NEWLY CREATED PRIMITIVE VALUE IS ANTI-BASING, AS Y IN ≠Y := X + 1;≠. (IT IS NOT CLEAR WHETHER X SHOULD ALSO BE ANTI-BASING, OR MERELY PASSIVE.)

(2) AFTER THIS CATEGORISATION OF OCCURENCES, OUR AIM IS TO ENSURE THAT EACH PRO-BASING OCCURENCE SHOULD RECEIVE PROPER BASING. THIS IS ACCOMPLISHED BY THE INITIAL PHASE OF OUR ALGORITHM. EACH INSTRUCTION WITH PRO-BASING OCCURENCES WILL GENERATE ITS OWN BASE(S) AT THIS STAGE. LATER ON, THESE BASES WILL BE MERGED WITH OTHER BASES, AS BASING INFORMATION IS PROPAGATED BETWEEN INSTRUCTIONS.

(3) LET VO1, VO2 BE TWO OCCURENCES OF THE SAME VARIABLE. IF THEY ARE LINKED BY THE BFROM MAP, IF BOTH ARE OF THE SAME TYPE, AND IF ONE OF THEM HAS ALREADY RECEIVED A BASING AND THE OTHER IS PRO-BASING, OR NEUTRAL WITH RESPECT TO THIS BASING, THEN THE SECOND OCCURENCE SHOULD HAVE THE SAME BASING AS THE FIRST ONE, AND IF IT HAS ALREADY RECEIVED SOME OTHER BASING, THEN THESE BASINGS SHOULD BE MERGED.

(4) NO ANTI-BASING OCCURENCE SHOULD RECEIVE A BASING. THIS CONDITION WILL BE FULFILLED AUTOMATICALLY, IF BASINGS ARE ASSIGNED ONLY BY THE PRINCIPLES (2) AND (3) ABOVE.

(5) BASES SHOULD BE SUPPRESSED IF THEY SUPPORT ONLY ONE COMPOSITE OBJECT (SET OR MAP).

(6) 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 IT HAS NOW BEEN SHIFTED 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. THUS, WE CAN IGNORE THIS PROBLEM COMPLETELY IN THE PRESENT ALGORITHM, SIMPLIFYING IT CONSIDERABLY.

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

THESE HEURISTICS SUGGEST A RATHER SIMPLE ALGORITHM TO PERFORM AUTOMATIC DATA STRUCTURE SELECTION, SOMEWHAT RESEMBLING THE TYPE FINDER. 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 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

```
  OI¬REPR := TYP;

  FOR EACH INSTRUCTION CONTAINING PRO-BASING OCCURENCES,
  GENERATE A BASE (TEMPORARILY UNIQUE TO THE INSTRUCTION),
  COMPUTE ITS FORM FROM THE ≠TYP≠ OF THE PRO-BASING OCCURENCES,
  AND MODIFY THE OI¬REPR MAP OF THESE OCCURENCES TO THE
  APPROPRIATE BASED REPRESENTATIONS;

  WORKPILE := ≤ [VO, OI¬REPR(VO), UP¬DOWN]: VO IS PRO-BASING≥;
      (WHERE UP¬DOWN IS A PROPAGATION DIRECTIVE,
      INDICATING PROPAGATION THROUGH BOTH BFROM AND FFROM LINKS).
```

2. BASE PROPAGATION

```
  EQBASES := NL;  $ AN EQUIVALENCE RELATION ON BASES

  (WHILE WORKPILE /= NL)  [VO, PROP¬REPR, KEY] FROM WORKPILE;

      CASE KEY OF

      (UP¬DOWN):
          (∀ VO1 → BFROM≤VO≥ + FFROM≤VO≥ ∣ TYP(VO1) = TYP(VO)
$ LET US EMPHASIZE AGAIN THAT OUR ALGORITHM INSISTS ON PROPAGATING
$ BASINGS ALONG BFROM LINKS ONLY TO OCCURENCES WITH 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,
                      AND NOT IS¬ANTI¬BASING(VO1, PROP¬REPR))

              NEW¬REPR := MERGE(VO1, PROP¬REPR);
              IF NEW¬REPR /= NL THEN
                  WORKPILE + ≤ [VO1, NEW¬REPR, IN¬INST],
                               [VO1, NEW¬REPR, UP¬DOWN] ≥;
              END IF;
          END ∀;
```

```
        (IN¬INST):
            INST¬OCCS := TUPLE OF THE OCCURENCES IN THE
                         INSTRUCTION OF VO;

            NEW¬REPRS := PROP¬IN¬INST(INST¬OCCS, VO, PROP¬REPR);
$ PROP¬IN¬INST PROPAGATES THE BASING OF VO THROUGHOUT ITS
$ INSTRUCTION, RETURNING A TUPLE CONTAINING NEW REPRS OF EACH
$ OCCURENCE IN THE INSTRUCTION.

            (∀ VO1 := INST¬OCCS(K) | NEW¬REPR := NEW¬REPRS(K) /= NL)
                WORKPILE WITH [VO1, NEW¬REPR, UP¬DOWN];
            END ∀;

        END CASE;
    END WHILE;



    PROC MERGE(VO, REPR);
$ THIS ROUTINE MERGES THE CURRENT REPR OF VO WITH THE GIVEN REPR,
$ ALLOWING ALSO MULTIPLE BASINGS. THUS, THE OI¬REPR OF AN
$ OCCURENCE WILL BE A SET CONTAINING ONE OR SEVERAL REPRS.
$ MERGE(VO, PROP¬REPR) COMPARES EACH REPR IN THE PRESENT
$ OI¬REPR(VO) WITH EACH REPR IN PROP¬REPR, LOOKING FOR MATCHING
$ REPR PATTERNS. E.G. ∈B1 AND ∈B2 ARE CONSIDERED AS HAVING THE
$ SAME PATTERN, WHEREAS ∈B1 AND SET(∈B2) ARE NOT. EACH MATCHED
$ PAIR OF PATTERNS IMPLIES EQUIVALENCING THE CORRESPONDING
$ BASES, WITHOUT CAUSING ANY MODIFICATION OF OI¬REPR(VO), AND
$ IF THERE EXISTS A REPR IN PROP¬REPR WITH NO MATCHING REPR
$ IN OI¬REPR(VO), WE ADD THIS REPR TO OI¬REPR(VO), AND CONSIDER
$ OI¬REPR(VO) TO HAVE BEEN MODIFIED.

    OLD¬REPR := OI¬REPR(VO);

    NEW¬REPR := NL;
    (∀ REPR2 ∈ PROP¬REPR)
        IS¬NEW := TRUE;
        (∀ REPR1 ∈ OLD¬REPR)
            [IS¬SAME¬REPR, BASEA1, BASEA2] :=
                        EQUATE¬REPRS(REPR1,REPR2);
$ EQUATE¬REPRS COMPARES TWO REPRS OF THE SAME OCCURENCE.
$ WHILE DOING SO, IT BUILDS TWO BASE ARRAYS CORRESPONDING TO
$ THESE REPRS. THE BASE ARRAY OF A GIVEN REPR IS DEFINED AS
$ FOLLOWS: EACH MODE OF THE FORM ∈B IS CONSIDERED AS BEING
$ PRIMITIVE. EQUATE¬REPRS PARSES EACH GIVEN REPR TO OBTAIN
$ A PARSE TREE, WHOSE LEAVES ARE THE PRIMITIVE MODES IN THAT
$ REPR. THE LEAVES ARE THEN ARRANGED IN THEIR POSTORDER, AND
$ IF THE I-TH LEAF IS AN ELEMENT-OF-BASE MODE, THEN THE I-TH
$ COMPONENT OF THE BASE ARRAY IS THE CORRESPONDING BASE NAME;
$ OTHERWISE, THAT COMPONENT IS UNDEFINED. FOR EXAMPLE, THE BASE
$ ARRAY OF THE REPR  MAP(∈B1) MAP(INT) ∈B2  IS [B1, OM, B2].
$ IF THE SAME PARSE TREE (EXCEPT FOR THE LEAVES) IS OBTAINED FOR
$ BOTH REPRS, THEN THE BASES IN THEIR ARRAYS SHOULD BE MERGED
$ COMPONENTWISE. IF THE ARRAY OF THE SECOND REPR CONTAINS A
$ COMPONENT B, AND THE CORRESPONDING COMPONENT OF THE FIRST
$ ARRAY IS UNDEFINED, THEN NO MERGING IS NEEDED; REPR1 IS DELETED
$ FROM OI¬REPR(VO) AND IS REPLACED BY A REPR WITH THE SAME PARSE
$ TREE, BUT IN WHOSE BASE ARRAY UNDEFINED COMPONENTS ARE REPLACED
$ BY THE CORRESPONDING BASE NAMES IN THE BASE ARRAY OF REPR2.
$ OI¬REPR(VO) SHOULD BE REGARDED TO HAVE BEEN MODIFIED.

          IF IS¬SAME¬REPR THEN
              IS¬NEW := FALSE;
              (∀ I := 1 ..., ↓BASEA2)
                  IF (B1 := BASEA1(I)) = OM THEN
                      IS¬NEW := TRUE;
                      BASEA1(I) := BASEA2(I);
                  ELSE
                      EQBASES WITH [B1, BASEA2(I)];
                  END IF;
              END ∀;

              IF IS¬NEW THEN
$ A PRIMITIVE TYPE IN REPR1 HAS BEEN REPLACED BY AN ELEMENT OF
$ BASE MODE, CONSTRUCT THE CORRESPONDING NEW REPR, AND DELETE
$ REPR1 FROM OI¬REPR.
                  REPR2 := MODIFY¬REPR(REPR1, BASEA1);
                  OI¬REPR(VO) LESS REPR1;
              END IF;
              QUIT ∀ REPR1;
          END IF;

      END ∀ REPR1;

      IF IS¬NEW THEN
$ REPR2 HAS NO MATCHING REPR IN OI¬REPR. IT SHOULD BE ADDED TO IT
          NEW¬REPR WITH REPR2;
      END IF;
  END ∀ REPR2;

  OI¬REPR(VO) + NEW¬REPR;
  RETURN NEW¬REPR;
  END PROC MERGE;



  PROC PROP¬IN¬INST(OCCS, VO, PROP¬REPR);

$ THIS ROUTINE RESEMBLES THE ≠FORWARD≠ AND ≠BACKWARDS≠
$ ROUTINES IN THE TYPE FINDER. IT CONSISTS OF A HUGE CASE
$ STATEMENT ON THE OPCODE OF THE INSTRUCTION. FOR EACH
$ OPCODE, WE PROPAGATE THE REPR OF ONE OCCURENCE TO THE OTHER
$ OCCURENCES IN THE INSTRUCTION, MERGING REPRS AND LOOKING
$ FOR CHANGED REPRS, BY USING THE MERGE ROUTINE ABOVE.
$ ITS OUTPUT IS A TUPLE ≠NEW¬REPRS≠, CONTAINING THE OUTPUT
$ OF THE MERGE ROUTINE FOR EACH ARGUMENT OF THE INSTRUCTION.
$ WE SHALL GIVE BELOW THE TREATMENT FOR ≠Q1¬WITH≠:

     NVO := ARGNO(VO);
     CASE OPCODE(INSTNO(VO)) OF

     (Q1¬WITH, Q1¬LESS):
$ ≠T := S WITH X;≠

         IF NVO = 3 THEN      $ PROPAGATING FROM THE ≠X≠
             NEW¬REPRS :=
                 [MERGE(OCCS(I), MAKEREPR(GROSSTYP(TYP(OCCS(I))),
                                PROP¬REPR)) : I := 1 ... 2]
                 WITH NL;
$ THE THIRD COMPONENT OF NEW¬REPRS IS EMPTY, AS WE PROPAGATE
$ FROM THE THIRD ARGUMENT IN THIS CASE. THE FIRST TWO COMPONENTS
$ ARE OBTAINED BY MERGING THE REPRS OF THE FIRST TWO ARGUMENTS
$ WITH THE REPR ≠SET(PROP¬REPR)≠ OR ≠HOMOGENEOUS TUPLE(PROP¬REPR)≠
$ DEPENDING ON THE GROSS TYPE OF THE FIRST TWO OCCURENCES.
         ELSE
             NEW¬REPR := MERGE(OCCS(3), COMPTYP(PROP¬REPR));
$ MERGE THE REPR OF THE ≠X≠ OCCURENCE WITH THE ELEMENT REPR
$ OF PROP¬REPR, DO NOT MERGE IF PROP¬REPR IS NOT COMPOSITE.
$ THEN MERGE PROP¬REPR WITH THE REPR OF THE OTHER COMPOSITE
$ ARGUMENT, AND CONSTRUCT NEW¬REPRS ACCORDINGLY.
             IF NVO = 1 THEN
                 NEW¬REPRS :=
                     [NL, MERGE(OCCS(2), PROP¬REPR), NEW¬REPR];
             ELSE
                 NEW¬REPRS :=
                     [MERGE(OCCS(1), PROP¬REPR), NL, NEW¬REPR];
             END IF;
         END IF;

     . . . . .

     END CASE;

     RETURN NEW¬REPRS;
     END PROC PROP¬IN¬INST;




     PROC IS¬ANTI¬BASING(VO, PROP¬REPR);

$ THIS ROUTINE DETERMINES, FOR A GIVEN OCCURENCE AND A
$ GIVEN REPR, WHETHER THIS REPR CAN BE PROPAGATED TO THAT
$ OCCURENCE. IT ALSO CONSISTS OF A CASE BRANCHING ON THE
$ OPCODE OF THE OCCURENCE AND ITS TYPE.
$ AS BEFORE WE SHALL GIVE BELOW ONLY FEW TYPICAL CASES:

    NVO := ARGNO(VO);
    CASE OPCODE(INSTNO(VO)) OF

    (Q1¬WITH, Q1¬LESS):

$ FOR EACH ARGUMENT IN SUCH INSTRUCTIONS, ANY PROP¬REPR IS
$ PERMISSIBLE, AS LONG AS THE COMPOSITE ARGUMENTS HAVE EQUAL,
$ UNAMBIGUOUS TYPES.
        RETURN IF T := TYP(OI¬SIB(VO,1)) /= TYP(OI¬SIB(VO,2))
$ UNEQUAL TYPES OF THE FIRST TWO ARGUMENTS
            OR IS¬AMBIG(GROSSTYP(T)) THEN TRUE ELSE FALSE END;

    (Q1¬NELT):

        RETURN IF NVO = 1
            OR PRIMITIVE(PROP¬REPR)
$ AN ELEMENT OF BASE REPR (WHICH IS REGARDED AS A PRIMITIVE REPR)
$ FOR THE COMPOSITE ARGUMENT SLOWS DOWN THE COMPUTATION, SO THAT
$ VO IS ANTI-BASING FOR SUCH A REPR (RECALL THAT SUCH A REPR IS
$ CONSIDERED TO BE PRIMITIVE).
            OR IS¬AMBIG(GROSSTYP(TYP(VO)))
            THEN TRUE ELSE FALSE END;

    . . . .

    END CASE;

    END PROC IS¬ANTI¬BASING;
```

REMARK:

THE APPROACH SKETCHED ABOVE IS MORE CONCERNED TO MINIMIZE SPACE USAGE THAN TIME USAGE BY THE ALGORITHM.
AN ALTERNATIVE APPROACH MIGHT COMPUTE SEVERAL AUXILIARY MAPS IN A PRE-PASS THROUGH THE CODE, WHICH CONTAIN THE NECESSARY INFORMATION CONCERNING BASING PROPAGATION THROUGH AN INSTRUCTION, AND THE PRO- OR ANTI-BASING NATURE OF OCCURENCES, SO THAT THE CORRESPONDING ROUTINES CAN BECOME SHORTER AND FASTER. A STRONG ARGUMENT IN FAVOR OF THE SECOND APPROACH, IS THAT OUR ALGORITHM PROPAGATES INFORMATION ALONG PATHS WHICH HAVE ALREADY BEEN USED BY THE TYPE FINDER, SO THAT POSSIBLY WE CAN AVOID DUPLICATION OF THE CUMBERSOME TYPE FINDING ROUTINES IN OUR ALGORITHM (WHICH IS, UNFORTUNATELY, WHAT THE FIRST APPROACH DOES). AT PRESENT, THE EXACT NATURE OF THE INFORMATION TO BE COLLECTED IN SUCH A PRE-PASS IS NOT YET CLEAR TO US. THIS IS WHY WE HAVE DESCRIBED ABOVE THE OTHER APPRCACH. THIS IS A SUBJECT FOR FURTHER RESEARCH.

3. BASE ADJUSTMENTS

THIS PHASE IS VERY SIMILAR TO SEVERAL PARTS OF ED SCHONBERG≠S ALGORITHM, AS IMPLEMENTED IN THE CURRENT SETL OPTIMIZER. IT WORKS OUT THE FULL EQUIVALENCE RELATION BETWEEN BASES, MAPPING EACH EQUIVALENCE CLASS INTO A REPRESENTING BASE, AND UPDATES THE OI¬REPR MAP BY REPLACING BASES BY THEIR REPRESENTATIVES. BASES THAT STILL SUPPORT ONLY ONE COMPOSITE OBJECT ARE THEN DROPPED, AND, FOR EACH OCCURENCE VO, SUCH THAT OI¬REPR(VO) CONTAINS REPR(S) INVOLVING SUCH BASES, WE EITHER DELETE THESE REPRS FROM OI¬REPR(VO), OR ELSE, IF THIS IS THE ONLY REPR IN OI¬REPR(VO), REPLACE THE ELEMENT MODE OF SUCH A BASE BY THE TYPE OF THE ELEMENTS OF THAT BASE. THEN FOR EACH OCCURENCE THAT STILL HAS MORE THAN ONE REPR, WE KEEP ONLY THE SHORTEST SUCH REPR, AND USE THE OTHER REPRS TO MODIFY THE FORM OF THE CORRESPONDING BASES. FOR EXAMPLE, IF OI¬REPR(VO) = ≤ →B1, SET(→B2) ≥, THEN ITS FINAL OI¬REPR SHOULD BE →B1, AND THE FORM OF B1 SHOULD BE REPLACED BY SET(→B2). NOTE THAT AT THIS PHASE SUCH MODIFICATIONS WILL NOT CAUSE NEW BASE MERGINGS. THIS IS IMPORTANT IN IN CONNECTION WITH CODE SEQUENCES LIKE THE FOLLOWING: X WITH A; S WITH X; Y FROM S; B FROM Y; WHERE THE BASE THAT SUPPORTS S SHOULD HAVE THE MODE: SET OF ELEMENTS OF THE BASE THAT SUPPORTS X (AND Y), SO THAT NO CONVERSION IS NECESSARY FOR VALUES FLOWING FROM A TO B. FINALLY, ALL SURVIVING BASES ARE ENTERED INTO THE SYMBOL TABLE.

4. BASING REFINEMENT

THIS PHASE CAN ALSO BE TAKEN FROM SCHONBERG≠S ALGORITHM. AS NOTED BEFORE, WE DO NOT ATTEMPT TO IMPROVE THIS PHASE AT PRESENT.

REMARKS:

(1) OUR ALGORITHM DISTINGUISHES BETWEEN DISTINCT TYPES. 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 THESE TYPES UNDER CERTAIN RESTRICTIONS (FOR ONE, THE FLOW MUST ALWAYS BE FROM A MORE SPECIFIC TYPE TO A MORE GENERAL ONE); ANYWAY, 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, THEY RAISE, HOWEVER, 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, AND THIS WAS THE REASON WHY THE USER HAS SUPPLIED TWO DISTINCT BASES INSTEAD OF ONE.

(3) ANOTHER PROBLEM CONCERNS FORMAL BASES, WHICH ARE BASES SUPPORTING FORMAL PARAMETERS OF A PROCEDURE. ACCORDING TO SCHONBERG≠S PHILOSOPHY, THEY SHOULD NOT BE MERGED WITH THE BASES OF THE ACTUAL ARGUMENTS, AND THIS IS ACHIEVED SIMPLY BY MAKING OCCURENCES IN ARGIN INSTRUCTIONS ANTI-BASING.

EXAMPLE:

CONSIDER THE FOLLOWING TOPOLOGICAL SORT PROGRAM:

```
    PROGRAM TEST3;
$ 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 TEST3;


 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 FIRST TWO PHASES OF OUR ALGORITHM, WE OBTAIN THE FOLLOWING REPRS:

ALL OCCURENCES OF NODES AND NOPREV WILL HAVE THE REPR SET(→B1). A AND B AT LINE 3 WILL BE UNBASED, BUT HAVE THE REPR →B1 AT LINES 4,5,6. HENCE, LOCATE INSTRUCTIONS OF A AND B INTO B1 WILL BE INSERTED JUST BEFORE LINE 4 AND LINE 5 RESPECTIVELY. NO OTHER LOCATE INSTRUCTIONS ARE NECESSARY. CESOR AT LINE 6 WILL HAVE THE MULTIPLE REPR SET(→B2) AND SET(PAIR(→B1,→B1)). HOWEVER, THE FIRST REPR WILL BE DROPPED AT PHASE 3 (CESOR IS THE ONLY COMPOSITE OBJECT SUPPORTED BY B2), AND THE SECOND REPR IS ESSENTIALLY EQUIVALENT TO MAP(→B1)→B1 (WE TREAT THE ≠MAP≠ TYPE RATHER LOOSELY AT PRESENT, FOR THE EXACT DETAILS OF HOW TO DETERMINE THAT AN OCCURENCE HAS THE TYPE MAP, ARE NOT COMPLETELY RESOLVED YET). ALL OCCURENCES OF NUMPREV ARE REPRED MAP(→B1)INT. OCCURENCES OF N AND M GET THE REPR →B1. N AT LINE 16 IS NOT ANTI-BASING (FOR SORTED IS OF UNAMBIGUOUS TYPE), AND SO SORTED AT LINES 13 AND 16 GETS THE REPR HTUP(→B1) (HTUP STANDS FOR HOMOGENEOUS TUPLE). HOWEVER, THE IMPLIED IVARIABLE OF THE ≠PRINT≠ STATEMENT AT LINE 7 IS ANTI-BASING, SO THAT THE PROPAGATION OF THE REPR OF ≠SORTED≠ WILL HALT AT THE ARGOUT ASSIGNMENT FOLLOWING THE CALL TO TOPSORT AT LINE 7 (SO THAT THERE WILL BE A CONVERSION TO TO A HOMOGENEOUS TUPLE BEFORE THE PRINT).

NOTE THAT WE DID NOT REQUIRE ANY USER SUPPLIED REPR TO DERIVE ALL THIS INFORMATION. THE BEST THAT SUCH A REPR COULD DO IS TO CHANGE THE FORM OF B1 FROM BASE OF GENERALS (WHICH IS THE FORM OUR ALGORITHM WOULD PICK UP) TO BASE OF ATOMS, SAY, BUT THIS CHANGE WILL HAVE NO EFFECT ON EXECUTION EFFICIENCY.

POSTSCRIPT:

THE APPROACH DESCRIBED IN THIS NEWSLETTER IS NOW BEING WEIGHED AGAINST ANOTHER APPROACH, WHICH BORROWS MORE HEAVILY IDEAS FROM ED SCHONBERG≠S ALGORITHM AND USES THEM TO ELIMINATE ALTOGETHER BASE PROPAGATION ACROSS INSTRUCTIONS, AND TO REDUCE BASE-PROPAGATION TO A MINIMUM, ALL THIS AT THE EXPENSE OF A PRE-PROCESSING OF THE CODE AND A MORE DELICATE BASE ADJUSTMENT PHASE (CF. THE REMARK AT THE END OF PHASE (2) OF THE ALGORITHM). THIS NEW APPROACH WILL BE DESCRIBED IN A COMING NEWSLETTER.