SETL NEWSLETTER 199

NON-PROPAGATION OF ERRORS - A MODIFIED TYPE-FINDING ALGORITHM

MICHA SHARIR
5 JAN 1978

THE GLOBAL OPTIMIZATION ANALYSES PERFORMED BY THE SETL OPTIMIZER PRODUCE INFORMATION WHICH CAN BE QUITE USEFUL DURING DEBUGGING. HOWEVER, THE OPTIMIZER ALGORITHMS AS THEY NOW STAND ARE ORIENTED TOWARD CODE IMPROVEMENT RATHER THAN TOWARD THE PRODUCTION OF HELPFUL DIAGNOSTIC MESSAGES. THIS SUGGESTS THAT IT MIGHT BE USEFUL TO DESIGN AN ADDITIONAL ≠DEBUG - ORIENTED≠ VERSION OF THE PRESENT OPTIMIZER, BY ADAPTING SEVERAL EXISTING ALGORITHMS TO THE DEBUGGING, DIAGNOSTIC PRODUCING MODE. NOTE THAT MOST OF THE OPTIMIZER ALGORITHMS ARE NOT NEEDED IN SUCH A MODE; THE MOST USEFUL ALGORITHM IN SUCH A MODE IS THE TYPE FINDER.

IN THIS NEWSLETTER WE PROPOSE A SLIGHT MODIFICATION OF THE TYPE FINDER, WHICH WILL MAKE IT MORE ADVANTAGEOUS THAN THE CURRENT ALGORITHM IN THE DEBUGGING, ERROR-DETECTING MODE OF OPTIMIZATION, AND WILL HAVE NO EFFECT ON OPTIMIZATION OF CORRECT, PRODUCTION PROGRAMS.

PRESENTLY, THE SMALLEST (ZERO) ELEMENT IN THE TYPE LATTICE IS BOTH THE ERROR TYPE AND THE UNDEFINED TYPE. BY DEFAULT, IF AN OCCURENCE IS NOT YET ANALYZED DURING TYPE FINDING, THEN ITS TYPE IS STILL UNDEFINED, AND MAY INDEED BE INTERPRETED AS A TEMPORARY ERROR-TYPE, IN THE SENSE THAT IF THE TYPE OF THAT OCCURENCE REMAINS UNDEFINED AT THE END OF TYPE-FINDING (AS WITH A VARIABLE THAT IS USED BEFORE IT IS EVER DEFINED), THEN INDEED WE HAVE AN ERRONEOUS OCCURENCE.

A MORE FREQUENT CAUSE FOR ASSIGNING THE ERROR TYPE TO AN OCCURENCE IS THE THE APPEARANCE OF AN ILLEGAL TYPE FOR AN ARGUMENT IN AN INSTRUCTION. FOR EXAMPLE

```
(1)     X := ≠A≠;
(2)     Y := X + 1;
```

HERE WE ASSIGN THE ERROR TYPE TO Y2 DURING FORWARD PROPAGATION, THIS TIME EXPLICITLY AND NOT BY DEFAULT.

IN THE CURRENT ALGORITHM, THIS ERROR-TYPE WILL BE PROPAGATED FURTHER, TO SUBSEQUENT USES OF Y, WHICH ARE LINKED IN THE DATA FLOW TO THE OCCURENCE OF Y IN (2) AND TO NO OTHER OCCURENCE OF Y. THIS MIGHT BE ALL RIGHT FOR PRODUCTION OPTIMIZATION, WHERE WE MIGHT WANT TO INFORM THE PROGRAMMER WHAT PARTS OF HIS PROGRAM MIGHT BECOME INVALID DUE TO THIS ERROR. HOWEVER, DURING DEBUGGING, PROPAGATION OF ERROR-TYPES WILL NOT CONTRIBUTE ANYTHING, BUT RATHER MAKE THINGS MORE OBSCURE, AND MAY PREVENT THE DETECTION OF MORE ERRORS. FOR EXAMPLE, IN A CODE SEQUENCE LIKE

```
X1 := CONSTANT;
X2 := OP1(X1);
X3 := OP2(X1, X2);
      .
      .
      .
```

IF THE COMPUTATION OF, SAY, X2 IS ERRONEOUS, THEN THIS ERROR WILL PROPAGATE THROUGH THE WHOLE SEQUENCE, AND ALL THE OVARIABLES WILL BE ASSIGNED AN ERROR-TYPE, REGARDLESS OF WHETHER THE CODE CONTAINS ADDITIONAL ERRORS OR NOT.

FOR DEBUGGING PURPOSES, IT IS THEREFORE REASONABLE TO MODIFY THE TYPE-FINDING ALGORITHM, SO THAT THIS UNNEEDED PROPAGATION OF ERROR TYPES IS SUPPRESSED; THIS WILL ALLOW THE OPTIMIZER TO DETECT AS MANY ≠INDEPENDENT≠ ERRORS AS POSSIBLE, AND IS DONE AS FOLLOWS;

CONSIDER FIRST THE FORWARD PROPAGATION PHASE OF THE TYPE FINDER.

LET US ASSUME, FOR THE SAKE OF SIMPLICITY, THAT OUR ANALYSIS IS INTRA-PROCEDURAL (AN ANALOGOUS APPROACH WOULD HOLD IN THE INTER-PROCEDURAL CASE), AND LET ≠TYP≠ BE A MAP ON OCCURENCES MAPPING EACH OCCURENCE TO ITS TYPE, AS COMPUTED SO FAR.

SUPPOSE THAT AT THE END OF THE FIRST PHASE OF THE TYPE FINDER, THERE EXIST OCCURENCES OI, FOR WHICH TYP(OI) = T¬Z (THE ERROR, ZERO TYPE). THE Q1¬CODE, AS PRODUCED BY THE SEMANTIC PASS, AND ARTIFICIALLY MODIFIED BY THE INITIAL PHASES OF THE OPTIMIZER, IS SUCH THAT EVERY VARIABLE IS SOMEHOW DEFINED BEFORE BEING USED (LOCAL VARIABLES ARE SET TO OM AT THE ENTRY OF EACH PROCEDURE, AND GLOBAL VARIABLES ARE SET TO OM AT THE BEGINNING OF THE MAIN PROGRAM, OR HAVE A DUMMY DEFINITION IN A SIMULATED ≠BLACK-BOX≠ MAIN PROGRAM). IT THUS FOLLOWS THAT THERE MUST EXIST INSTRUCTIONS FOR WHICH TYP OF THE OVARIABLE IS T¬Z, BUT TYP OF EACH IVARIABLE IS NOT T¬Z. LET US CALL SUCH INSTRUCTIONS ERRONEOUS. WE MODIFY THE CODE BY REPLACING THE OPCODE OF EACH ERRONEOUS INSTRUCTION BY AN AUXILIARY OPCODE Q1¬COR, WHOSE SEMANTICS ARE - ASSIGN A GENERAL VALUE TO THE OVARIABLE, REGARDLESS OF THE TYPES OF THE IVARIABLES. NOW WE RE-APPLY THE FORWARD PROPAGATION PHASE ONCE MORE, WITH THE FOLLOWING INITIAL VALUE OF ≠TYP≠:

TYP(OI) := MAXIMAL TYPE OF OI, IF OI IS AN OVARIABLE OF AN ERRONEOUS INSTRUCTION (I.E. A GENERAL TYPE, OR A REPRED TYPE IF DECLARATIONS GIVING INFORMATION ABOUT OI HAVE BEEN MADE). TYP(OI) IS UNCHANGED FOR ALL OTHER OCCURENCES OI.

THE WORKPILE DRIVING THE ADDITIONAL FORWARD TYPEFINDING PHASE IS INITIALIZED TO CONTAIN ALL ERRONEOUS OVARIABLES.

IF, AT THE END OF THIS ADDITIONAL PHASE, THERE ARE NEW ERRONEOUS INSTRUCTIONS, WE REPEAT THE WHOLE PROCEDURE ABOVE ONCE MORE, AND KEEP ON DOING SO TILL NO NEW ERRONEOUS INSTRUCTIONS ARE OBTAINED.

THE HEURISTICS OF THIS APPROACH ARE: THE EXECUTION OF ANY ERRONEOUS INSTRUCTION WILL CERTAINLY CAUSE A PROGRAM ABORT. SUPPOSE, HOWEVER, THAT SOMEHOW THIS ERROR HAS BEEN CORRECTED AND THE PROGRAMMER WANTS US TO PROCEED WITH TYPE ANALYSIS. SINCE WE HAVE NO IDEA WHAT THE CORRECT VALUE OF THE OVARIABLE IS SUPPOSED TO BE, WE ASSUME THAT IT IS OF A GENERAL TYPE, AND PROCEEDS WITH FORWARD PROPAGATION ONCE MORE.

IT IS EASY TO SHOW THAT THE ABOVE PROCESS CONVERGES. INDEED, (IN THE TERMINOLOGY OF TENENBAUM≠S THESIS), EACH APPLICATION OF THE FORWARD PROPAGATION PHASE OF THE TYPE FINDER IS MONOTONIC, IN THE SENSE THAT EACH PROPAGATION STEP INCREASES THE TYPE VECTOR OF ALL VARIABLE OCCURENCES (THIS IS ESSENTIALLY THE ≠TYP≠ MAP). THE ≠INTERFACES≠ BETWEEN ANY TWO SUCCESSIVE APPLICATIONS OF THE FIRST PHASE, AS SKETCHED ABOVE, ARE ALSO MONOTONIC INCREASING, SINCE THEY REPLACE SOME T→Z COMPONENTS BY T→G COMPONENTS. HENCE, THE WHOLE PROCESS IS MONOTONIC, AND THEREFORE CONVERGES, YIELDING A FINAL TYP VECTOR WHICH IS OBVIOUSLY LARGER THAN THE ONE COMPUTED BY THE STANDARD FIRST PHASE.

AFTER THIS PROCESS TERMINATES, WE ISSUE A FATAL ERROR MESSAGE FOR EACH ERRONEOUS INSTRUCTION.

NOTE THAT THE ADDED APPLICATIONS OF THE FIRST TYPE-FINDING PHASE WILL GENERALLY BE MUCH FASTER THAN THE FIRST ONE. THEY WILL ONLY MODIFY THE TYPES OF OCCURENCES THAT DEPEND UPON AN ERRONEOUS OVARIABLE, AND SO WILL BE MAINLY OF A LOCAL NATURE (UNLESS, OF COURSE, THE PROGRAM WAS WRITTEN BY SOME CLEVER APE, OR THE HUMAN EQUIVALENT THEREOF).

REMARK: THE REASON FOR PERFORMING THE CODE MODIFICATION AND RESETTING THE ERROR TYPE OF ERRONEOUS OVARIABLES OFF-LINE, INSTEAD OF EMBEDDING IT INTO THE FORWARD PROPAGATION PHASE, AND THUS AVOIDING RE-ITERATIONS OF THE WHOLE PHASE, IS TWOFOLD:

(1) THE FIRST PHASE IS TYPE-INCREASING, AND SO, OBTAINING A T→Z TYPE FOR AN OVARIABLE DURING THE EXECUTION OF THIS PHASE DOES NOT IMPLY A DEFINITE ERROR. PERHAPS WE SIMPLY HAVE NOT YET CONSIDERED ALL POSSIBLE LINKS. CONSIDER THE FOLLOWING EXAMPLE:

```
(1)     Y := 1;
(2)     IF COND1 THEN Y := ≠1≠; END IF;
(3)     IF COND2 THEN X := Y + 1; END IF;
```

IF WE PROPAGATE THE TYPE OF Y2 FIRST, WE OBTAIN T¬Z FOR X3, BUT AT THE END OF FORWARD PROPAGATION, TYP(X3) := INTEGER. AT THE END OF THE TYPE-FINDING ANALYSIS, WE SHALL ISSUE A WARNING, NON-FATAL, MESSAGE THAT THE STATIC LINK BETWEEN Y2 AND Y3 SHOULD NEVER MATERIALIZE IN RUN-TIME.

THUS, IT IS DANGEROUS AND MISLEADING TO INTERPRET T¬Z AS ERRONEOUS, UNTIL THE FIRST PHASE IS OVER.

(2) IF WE DO NOT MODIFY THE CODE, OR EVEN FLAG INSTRUCTIONS AS ERRONEOUS, DURING THE FIRST PHASE, BUT ONLY RESET T¬Z TO T¬G FOR OVARIABLES, WE RUN INTO PROBLEMS OF NON-MONOTONICITY, WHICH MIGHT CAUSE THE FORWARD PHASE TO DIVERGE, AS INDEED HAPPENS IN THE FOLLOWING EXAMPLE:

```
(1)     X := ≠1≠;
        (ᵕ)
(2)         Y := X + 1;
(3)         Z := Y + ≠1≠;
(4)         X := Z + NL;
        END ᵕ;
```

ASSUMING THAT THE ORDER OF TYPE PROPAGATION IS THE SAME AS THE ORDER OF OCCURENCES IN THE CODE, WE OBTAIN THE FOLLOWING TYPE-VECTORS, AFTER EACH FULL ITERATION THROUGH THE CODE:

CI	X1	X2	Y2	Y3	Z3	Z4	X4
TYP1(OI)	T¬C	T¬C	T¬Z(T¬G)	T¬G	T¬C	T¬C	T¬Z(T¬G)
TYP2(OI)	T¬C	T¬G	T¬I	T¬I	T¬Z(T¬G)	T¬G	T¬SET
TYP3(OI)	T¬C	T¬SET .DIS T¬C	T¬Z(T¬G)	T¬G	T¬C	T¬C	T¬Z(T¬G)

THUS THIS APPROACH IS UNACCEPTABLE SINCE IT EVENTUALLY CAUSES THE TYPE VECTOR TO ALTERNATE BETWEEN TYP2 AND TYP3.

AT THIS MOMENT, WE HAVE NO IDEA HOW TO OVERCOME THESE PROBLEMS, WHICH IS WHY WE HAVE SUGGESTED PERFORMING THE ERROR ADJUSTMENTS ≠OFF-LINE≠, RATHER THAN AS AN INTEGRAL PART OF THE TYPE-FINDING ALGORITHM.

NEXT, CONSIDER THE SECOND, BACKWARD PROPAGATION PHASE OF THE TYPE FINDER, AGAIN, FOR DEBUGGING PURPOSES, IT IS POINTLESS TO PROPAGATE ERRORS UNCOVERED AT THIS PHASE, BUT IN THIS CASE WE HAVE A MUCH SIMPLER SOLUTION, NAMELY - WHENEVER AN OCCURENCE IS FOUND TO BE ERRONEOUS, FLAG IT AS BEING SUCH, BUT LEAVE ITS TYPE UNCHANGED, SO THAT NO TYPE PROPAGATION WILL BE REQUIRED.

TO SEE THAT THIS MODIFICATION STILL PRESERVES CONVERGENCE, NOTE THAT THE SECOND TYPE-FINDING PHASE HAVE THE FOLLOWING PROPERTY:

LET X BE A TYPE VECTOR, SHOWING THE TYPE OF ALL OCCURENCES, AND LET F BE SOME TRANSFORMATION OF X, USED DURING THE BACKWARD ANALYSIS PHASE, SO THAT F(X) IS THE TYPE VECTOR OBTAINED BY APPLYING A SINGLE PROPAGATION STEP. THEN F CAN BE REPRESENTED AS F(X) := X .CON G(X), WHERE G IS SOME OTHER TRANSFORMATION, AND CONJUNCTION IS TAKEN COMPONENTWISE. HENCE, WITHOUT ANY MONOTONICITY ASSUMPTIONS ON G, WE SEE THAT F(X) <= X, FOR ALL X AND F. THUS, EACH PROPAGATION STEP IN THE SECOND PHASE DECREASES THE TYPE VECTOR, FROM WHICH CONVERGENCE IS OBVIOUS. IT REMAINS TO OBSERVE THAT THE MODIFIED SECOND PHASE WHICH WE HAVE SUGGESTED JUST ABOVE, HAS THIS SAME FORM, BUT WITH G REPLACED BY ANOTHER TRANSFORMATION G1, WHERE G1(X) := G(X) IF X .CON G(X) /= T¬Z. AND G1(X) := X OTHERWISE.

WE SEE THEREFORE THAT THE MODIFIED SECOND PHASE WILL ALWAYS CONVERGE, TO A LARGER TYPE VECTOR THAN THE ONE COMPUTED BY THE NON-MODIFIED PHASE. HOWEVER, UNIQUENESS OF THAT VECTOR IS NO LONGER ENSURED, AND SO THE PROGRAMMER SHOULD BE AWARE THAT THE TYPE VECTOR COMPUTED FOR AN ERRONEOUS PROGRAM CAN BE SOMEWHAT ARBITRARY.

TO SEE THE SIGNIFICANCE OF THIS, CONSIDER THE FOLLOWING EXAMPLE:

```
(1)    READ Z;
(2)    X := Z + ≠1≠;
(3)    Y := Z + 1;
```

HERE WE HAVE THE FOLLOWING POSSIBLE TYPE VECTORS:

	Z1	Z2	X2	Z3	Y3
AT THE END OF PHASE 1	T¬G	T¬G	T¬C	T¬G	T¬I
PROPAGATING (2) TO (3)	T¬G	T¬C	T¬C	T¬C	T¬Z(T¬I)
PROPAGATING (3) TO (2)	T¬G	T¬I	T¬Z(T¬C)	T¬I	T¬I

SO THAT Z2 AND Z3 ARE BOTH ASSIGNED EITHER T¬I OR T¬C.

AT THE END OF THE BACKWARD TYPEFINDING PHASE WE ISSUE A FATAL ERROR MESSAGE FOR EACH FLAGGED ERRONEOUS OCCURENCE.