SETL NEWSLETTER 190 ART GRAND
THE IMPLEMENTATION OF BACKTRACKING MAY 5, 1977

THIS NEWSLETTER DISCUSSES THE IMPLEMENTATION OF THE BACKTRACKING SCHEME PROPOSED IN NEWSLETTER 168. WE BEGIN OUR DISCUSSION WITH THE RUN TIME IMPLEMENTATION, THEN DISCUSS THE CHANGES WHICH MUST BE MADE IN THE COMPILER.

FOR PURPOSES OF DISCUSSION, WE ASSUME THAT BACKTRACKING INVOLVES THE FOLLOWING PRIMITIVES. THE TRANSLATION OF SETL SOURCE INTO THESE PRIMITIVES IS DISCUSSED LATER.

OK:	SAVE THE CURRENT STATE OF THE BACKTRACK VARIABLES.
OKOK:	A SPECIAL OK.
FAIL:	RETREAT TO PREVIOUSLY SAVED ENVIRONMENT.
ACCEPT:	FINALIZE ALL CHANGES TO ENVIRONMENT SINCE LAST OKOK.
REJECT:	REJECT ALL CHANGES SINCE LAST OKOK.
ARB:	NONDETERMANISTIC ARB SELECTION

ENVIRONMENT BLOCKS

BACKTRACKING IS BASED ON VARIOUS OPERATIONS WHICH SAVE AND RESTORE PARTS OF THE ENVIRONMENT. ENVIRONMENTS ARE SAVED BY PUSHING ≠ENVIRONMENT BLOCKS≠ ONTO THE STACK AND RESTORED BY POPPING THEM. AN ENVIRONMENT BLOCK CONSISTS OF A SERIES OF SAVED VALUES TOGETHER WITH A POINTER TO A PROCEDURE TO POP THE BLOCK.

FOR EXAMPLE, THE OK PRIMITIVE PUSHES THE VALUES OF ALL ≠BACK≠ VARIABLES TOGETHER WITH A POINTER TO A PROCEDURE WHICH WILL POP THE VALUES. THE FAIL PRIMITIVE THEN POPS THE PROCEDURE POINTER AND EXECUTES IT TO RESTORE THE ENVIRONMENT.

THE STACK CONSISTS OF A SERIES OF ENVIRONMENT BLOCKS, STACKED IN THE ORDER THEY WERE SAVED. THE FAIL PRIMITIVE DESCRIBED ABOVE MUST ACTUALLY RESTORE ALL ENVIRONMENTS SAVED SINCE THE LAST OK. IT DOES THIS BY POPPING ENVIRONMENT BLOCKS FROM THE STACK UNTIL IT POPS AN OK BLOCK. EACH POP IS DONE BY EXECUTING THE PROCEDURE FOUND ON TOP OF THE STACK.

ENVIRONMENT BLOCKS ARE MADE OF STANDARD SETL VALUE SPECIFIERS. THE TOP TWO SPECIFIERS IN EACH BLOCK HAVE A STANDARD SIGNIFICANCE.

THE TOP SPECIFIER ALWAYS HAS

TYPE T↛SUBR
VALUE POINTER TO CODE FRAGMENT TO POP BLOCK

THE SPECIFIER BELOW IT HAS

TYPE T↛INT
VALUE LENGTH OF BLOCK

THERE ARE FOUR TYPES OF ENVIRONMENT BLOCKS.

ENTRY BLOCKS: THESE ARE BUILT BY ROUTINE PROLOGUES AND POPPED BY ROUTINE EPILOGS.

EXIT BLOCKS: THESE BLOCKS ARE BUILT WHEN WE EXECUTE A STATEMENT SUCH AS

IF OK THEN RETURN;;

AND ARE REMOVED BY EXECUTING FAIL, ACCEPT, OR REJECT. RETURNS LIKE THE ONE ABOVE ARE CALLED CONDITIONAL RETURNS AND ARE DESCRIBED BELOW.

OK BLOCKS: THESE ARE BUILT BY OK-S AND POPPED BY FAILS.

OKOK BLOCKS: THESE ARE BUILT BY OKOK-S AND POPPED BY FAILS.

ALL ENTRY BLOCKS ARE LINKED TOGETHER WITH A SPECIFIER WITH

.TYPE T↛INT
VALUE POINTER TO PREVIOUS ENTRY BLOCK

THE GLOBAL ≠LAST↛ENTRY≠ CONTAINS A POINTER TO THE MOST RECENT ENTRY BLOCK.

ALL OKOK BLOCKS ARE LINKED IN A SIMILAR FASHION, WITH THE GLOBAL ≠LAST↛OKOK≠ POINTING TO THE MOST RECENT BLOCK. BOTH LISTS MUST BE ADJUSTED WHEN THE GARBAGE COLLECTOR REQUESTS MORE SPACE FROM THE OPERATING SYSTEM, SINCE THIS WILL FORCE IT TO MOVE THE STACK. OTHEWISE THEY ARE IGNORED BY THE GARBAGE COLLECTOR.

CONDITIONAL RETURNS

A CONDITIONAL RETURN IS A RETURN MADE WHILE WE ARE IN A BACKTRACKING ENVIRONMENT, FOR EXAMPLE,

```
IF OK THEN RETURN;;
```

IF THE RETURN EVENTUALLY LEADS TO A FAIL, IT WILL BE NECESSARY TO UNDO THE RETURN, THAT IS TO RESTORE THE VALUES OF ALL PARAMETERS AND STACKED VARIABLES PRIOR TO THE RETURN.

WHEN WE ENTER A PROCEDURE WE SAVE THE VALUES OF ALL PARAMETERS AND STACKED VARIABLES ON THE STACK. WHEN WE MAKE A NORMAL RETURN, WE RESTORE THESE VALUES AND POP THE STACK. WHEN WE MAKE A CONDITIONAL RETURN WE CANNOT POP THE STACK; INSTEAD WE POP THE CURRENT VALUES OF THE STACKED VARIABLES WITH THOSE IN THE ENTRY BLOCK. THIS ALLOWS US TO REPEAT THE SWAP AND UNDO THE RETURN LATER.

WHEN WE MAKE A CONDITIONAL RETURN, WE PUSH A THREE WORD EXIT BLOCK ONTO THE STACK. THIS BLOCK CONTAINS:

1. A POINTER TO A PROCEDURE TO UNDO THE RETURN. SUCH A PROCEDURE IS CALLED A RESTORE PROCEDURE.

2. THE SIZE OF THE EXIT BLOCK, NAMELY 3.

3. A POINTER TO THE ENTRY BLOCK.

SIMULATED RETURNS

SUPPOSE WE EXECUTE THE CODE SEQUENCE SUCH AS:

```
(1)     IF OK THEN
            P(X);
        END IF;
            .
            .
        DEFINE P(Y);
            .
(2)         FAIL;
            .
        END P;
```

WHEN WE REACH THE FAIL AT (2) THERE ARE TWO BLOCKS ON THE STACK, AN OK BLOCK AND AN ENTRY BLOCK. BEFORE WE CAN RESTORE THE VALUES SAVED AT (1), WE MUST RESTORE THE VALUES STACKED ON ENTRY TO P. THIS IS DONE BY SIMULATING A RETURN FROM P WITHOUT ACTUALLY JUMPING TO THE RETURN ADDRESS.

WHEN WE PUSH AN ENTRY BLOCK ONTO THE STACK, THE TOP SPECIFIER CONTAINS A POINTER TO A PROCEDURE WHICH WILL SIMULATE A RETURN FROM THE ROUTINE.

TREATMENT OF ARB

THE ARB PRIMITIVE HAS TWO OPERANDS, AN ITERATOR AND A SET(TUPLE, ETC.). THE OPERATION X = ARB S SETS X TO THE FIRST ELEMENT OF S AND BUILDS A SPECIAL OK BLOCK ON THE STACK. THIS BLOCK IS UNUSUAL IN THAT IT POINTS TO A SPECIAL ≠ADVANCE≠ PROCEDURE WHICH NOT ONLY RESTORES THE SAVED VARIABLES WHEN A FAIL IS ENCOUNTERED, BUT ALSO ADVANCES X. IT ONLY POPS THE OK BLOCK IF X HAS REACHED THE END OF S.

ACCEPT AND REJECT

ACCEPT IS TREATED SOMEWHAT DIFFERENTLY FROM THE OTHER PRIMITIVES. ITS ACTION IS TO COMPACT THE TOP OF THE STACK, REMOVING ALL OK, AND EXIT BLOCKS SINCE THE LAST OKOK. THERE IS A STANDARD ACCEPT PROCEDURE IN THE LIBRARY WHICH IS DRIVEN BY THE SIZE SPECIFIER CONTAINED IN EACH ENVIRONMENT BLOCK.

REJECT PERFORMS A SERIES OF FAILS UNTIL IT REACHES AN OKOK BLOCK. THESE FAILS ARE UNUSUAL IN THAT THEY RETURN TO THE POINT OF THE REJECT RATHER THAN TO THE POINT OF THE OK(WHICH HAS BEEN SAVED AS PART OF THE OK BLOCK).

THE CODE FOR FAIL OPERATIONS CHECKS WHETHER IT IS RESTORING AN ENVIRONMENT SAVED BY AN OK OR ONE SAVED BY AN OKOK. IN THE LATTER CASE, IT ALWAYS RETURNS TO THE POINT OF THE OKOK. IN THE FORMER CASE, IT CHECKS THE GLOBAL FLAG IS↛REJECT TO SEE WHETHER WE ARE SIMULATING FAILS AS PART OF A REJECT. IF SO, IT RETURNS TO THE POINT OF THE REJECT. OTHERWISE IT RETURNS TO THE POINT OF THE OK.

SYSTEM SUCCESS FLAG

THERE IS A GLOBAL SYSTEM SUCCESS FLAG WHICH IS SET TO TRUE BY THE OK AND OKOK PRIMITIVES AND SET TO FALSE BY FAIL AND REJECT. THIS FLAG IS RETURNED AS THE VALUE OF OK AND OKOK.

CODE FOR BACKTRACKING

THE BACKTRACKING PRIMITIVES ARE IMPLEMENTED AS AUXILLIARY PROCEDURES WHICH ARE COMPILED FOR EACH ROUTINE CONTAINED IN THE SOURCE PROGRAM. AN INDIVIDUAL OK IS COMPILED AS A CALL TO THE AUXILLIARY OK PROCEDURE OF THE CURRENT ROUTINE; A FAIL IS IMPLEMENTED AS A SERIES OF CALLS TO THE PROCEDURES FOUND ON THE TOP OF THE STACK.

THE BASIC CALL AND RETURN NUBBINS ARE VERY LOW LEVEL. THE CALL MERLEY SAVES THE RETURN ADDRESS IN SOME GLOBAL LOCATION AND JUMPS TO ITS ARGUMENT. THE RETURN JUMPS TO THE PREVIOUSLY SAVED RETURN ADDRESS.

THE CODE SEQUENCES FOR LINKAGE, ROUTINE PROLOGUES, OK, ETC. ARE:

1. CALLS

   CALLS CONSIST OF A PUSH NUBBIN TO PUSH ITS ARGUMENTS, A CALL NUBBIN, AND A POP NUBBIN TO POP THE ARGUMENTS.

2. ROUTINE PROLOGUES

   A ROUTINE PROLOG CONSISTS OF A NUBBIN CALLED Q2ᵖENTRY WHICH SAVES THE RETURN ADDRESS, STACKED VARIABLES, ETC. THIS NUBBIN IS FOLLOWED BY CODE TO REINITIALIZE STACKED BASES AND PERFORM ANY RESULTING BASE ASSIGNMENTS.

3. ROUTINE EPILOGUES

   A ROUTINE EPILOGUE CONSISTS OF THE ROUTINES RETURN CODE PLUS LOCAL PROCEDURES FOR OK, OKOK, FAIL, RESTORE, SIMULATED RETURN, ARB, AND ADVANCE.

   THE RETURN CODE CONSISTS OF A Q2ᵖEXIT NUBBIN WHICH PERFORMS ALL THE STACK MANIPULATION, FOLLOWED BY A RETURN NUBBIN.

4. RETURNS

A RETURN IS COMPILED AS A JUMP TO THE ROUTINE EPILOGUE.

5. OK AND OKOK

THESE PRIMITIVES ARE COMPILED AS CALLS TO THE ROUTINE≠S OK AND OKOK PROCEDURES RESPECTIVELY.

6. FAIL

A FAIL IS REALIZED BY AN INFINITE LOOP WHICH POPS PROCEDURES FROM THE STACK AND EXECUTES THEM. THESE PROCEDURES WILL EITHER BE RESTORES AND SIMULATED RETURNS, WHICH WILL JUMP BACK TO THE LOOP, OR FAILS, WHICH WILL JUMP BACK TO THE POINT OF THE LAST OK.

7. REJECT

A REJECT IS SIMILAR TO A FAIL, EXCEPT THAT BEFORE ENTERING THE LOOP IT SETS THE FLAG IS≠REJECT TO TRUE. WHEN THIS FLAG IS SET WE WILL KEEP POPPING BLOCKS UNTIL WE REACH AN OKOK.

8. ACCEPT

ACCEPT IS A CALL TO A STANDARD LIBRARY ROUTINE.

9. ARB

WE GENERATE THE FOLLOWING INLINE CODE FOR ARB:

A. ASSUME WE ARE DOING X = ARB S. BEGIN BY PUSHING SYMBOL TABLE POINTERS TO X AND S ONTO THE STACK.

B. CALL THE ROUTINES ARB PROCEDURE. THE ARB PROCEDURE WILL SET X TO THE FIRST ELEMENT OF S AND BUILD AN OK BLOCK ON THE STACK. THIS OK BLOCK WILL CONTAIN A POINTER TO THE ROUTINES ADVANCE PROCEDURE.

C. AFTER RETURNING FROM THE ARB PROCEDURE, SEE IF X IS OMEGA. IF SO THEN FAIL.

WHEN THE ADVANCE PROCEDURE IS CALLED, IT WILL ADVANCE X OVER S, THEN PROCEED TO RESTORE BACKTRACK VARIABLES AS IF WE WERE EXECUTING A FAIL. HOWEVER IT WILL ONLY POP THE OK BLOCK IF X HAS REACHED THE END OF S.

NOTE THAT SINCE THE OK BLOCK CONTAINS POINTERS TO X AND S, THE SAME ADVANCE PROCEDURE CAN BE USED FOR ALL THE ARB OPERATIONS OF A SINGLE ROUTINE.

NUBBINS

IN THE NEXT TWO SECTIONS WE DETAIL THE NUBBINS FOR BACKTRACKING AND LINKAGE.

WE CONTINUE TO MAKE CERTAIN ASSUMPTIONS ABOUT THE ORDER OF THE SYMBOL TABLE, NAMELY THAT THE FORMAL PARAMETERS FOR EACH ROUTINE ARE STORED CONTIGUOUSLY, AS ARE THE STACKED VARIABLES FOR EACH ROUTINE. THIS ALLOWS US TO GENERATE CODE IN Q2 WHICH MOVES ENTIRE BLOCKS AT A TIME. THE ACTUAL MACHINE CODE FOR THESE LOOPS MAY OR MAY NOT BE DONE AS A LOOP.

SOME OF THE NUBBINS PRESENTED HERE HAVE MORE THAN THREE ARGUMENTS. AS USUAL, WE REPRESENT THESE NUBBINS BY A SERIES OF QUADRUPLES. THE OPCODE AND FIRST THREE ARGUMENTS ARE CONTAINED IN THE FIRST QUADRUPLE, AND THE REMAINING ARGUMENTS ARE PACKED INTO SUCCESSIVE QUADRUPLES.

BASIC LINKAGE NUBBINS

WE ASSUME THAT THERE IS SOME GLOBAL LOCATION KNOWN AS RETADDR WHICH CONTAINS THE RETURN ADDRESS. THIS WILL BE A HARDWARE REGISTER IN MOST IMPLEMENTATIONS.

THE BASIC LINKAGE NUBBINS ARE

1. Q2↑CALL
   SET RETADDR TO THE NEXT LOCATION THEN JUMP TO ARG3

2. Q2↑SCALL
   AS ABOVE, BUT CALL THE PROCEDURE GIVEN BY THE TOP STACK SPECIFIER

3. Q2↑VCALL
   AS Q2↑CALL, BUT THE THIRD ARGUMENT IS A SYMBOL TABLE POINTER TO A PROCEDURE VARIABLE.

4. Q2↑RETURN
   GO TO RETADDR.

ENVIRONMENT NUBBINS

THIS SECTION CONTAINS NUBBINS FOR PROCEDURE ENTRY AND EXIT, OK AND FAIL, ETC. THEY ARE ALL QUITE LONG AND COMPLEX.

THE ENTRY NUBBIN IS PLACED AT THE BEGINNING OF EACH ROUTINE AND IS FOLLOWED BY THE BODY OF THE ROUTINE. THE REMAINING NUBBINS ARE PLACED AT THE END OF THE ROUTINE AND ARE ESSENTIALLY LOCAL PROCEDURES WHICH ARE CALLED FROM THE BODY OF THE ROUTINE. THUS THE INLINE CODE GENERATED FOR OK IS A CALL TO THE OK-PROCEDURE LOCATED AT THE END OF THE ROUTINE, ETC. THESE LOCAL CALLS ARE MADE WITH THE STANDARD LINKAGE QUADRUPES DESCRIBED ABOVE

THE NUBBINS FOR ENTRY, EXIT, OK, AND OKOK PUSH ENVIRONMENT BLOCKS ONTO THE STACK. THUS THE TOP WORD OF THE STACK ALWAYS CONTAINS A SPECIFIER FOR A ROUTINE TO POP THE MOST RECENTLY SAVED ENVIRONMENT.

1. Q2↛ENTRY

   THIS NUBBIN IS THE ENTIRE ROUTINE PROLOGUE. IT ACTUALLY CONSISTS OF TWO QUADRUPLES SO THAT IT CAN HAVE MORE THAN THREE ARGUMENTS. ITS ARGUMENTS ARE:

   ARG1: SYMBOL TABLE POINTER TO FIRST FORMAL PARAMETER
   ARG2: NUMBER OF FORMAL PARAMETERS
   ARG3: SYMBOL TABLE POINTER TO FIRST STACKED VARIABLE
   ARG4: NUMBER OF STACKED VARIABLES
   ARG5: SYMBOL TABLE POINTER FOR THE ROUTINE
   ARG6: SYMBOL TABLE POINTER TO SPECIFIER WITH:

      TYPE: T↛SUBR
      VALUE: POINTER TO SIMULATED RETURN CODE

   ARG5 IS USED TO ACCESS THE PRINT NAME OF THE ROUTINE FOR DEBUGGING PURPOSES.

THE SEMANTIC OF THIS NUBBIN ARE

1. SWAP THE ARGUMENTS ON THE STACK WITH THE FORMAL PARAMETERS IN THE SYMBOL TABLE.

2. RESERVE ARG4 + 4 WORDS OF STACK SPACE.

3. COPY THE SPECIFIERS FOR ALL STACKED VARIABLES ONTO THE STACK AND SET THEIR ISFUNDEF BITS IN THE SYMBOL TABLE.

4. PUSH RETADDR, LASTFENTRY, AND THE SIZE OF THE BLOCK.

5. PUSH THE SPECIFIER FOR THE RETURN CODE

6. SET LASTFENTRY = TOP OF STACK

2. Q2FEXIT

THIS NUBBIN HANDLES THE ENTIRE RETURN CODE. ITS ARGUMENTS ARE LIKE THOSE OF Q2FENTRY EXCEPT THAT ARG6 POINTS TO A SPECIFIER FOR THE ROUTINES RESTORE PROCEDURE. ITS ACTIONS ARE

1. RESTORE THE PREVIOUS VALUE OF LASTFENTRY AND POP THE RETURN ADDRESS.

2. SEE IF THIS IS AN UNCONDITIONAL RETURN. THE RETURN IS UNCONDTIONAL IF THERE HAVE BEEN NO OK-S SINCE THE ROUTI ROUTINE WAS CALLED, I.E. IF THE ENTRY BLOCK IS THE TOP ENVIRONMENT BLOCK.

3. IF THE RETURN IS UNCONDITIONAL THEN

   A. RESTORE ALL STACKED VARIABLES
   B. SWAP ALL PARAMETERS WITH THEIR SAVED VALUES
   C. POP THE ENTRY BLOCK.

4. OTHERWISE THE RETURN IS CONDITIONAL. SWAP THE SYMBOL TABLE ENTRIES FOR ALL STACKED VARIABLES AND PARAMETERS WITH THE CORRESPONDING STACK ENTRIES, THEN BUILD AN EXIT BLOCK ON THE STACK. THIS CONSISTS OF:

   A. A POINTER TO THE ENTRY BLOCK. THIS POINTER IS STORED AS A SHORT INTEGER WHOSE VALUE IS THE OFFSET BETWEEN THE ENRTRY BLOCK AND THE EXIT BLOCK SO THAT IT CAN BE IGNORED BY THE GARBAGE COLLECTOR.

   B. THE LENGTH OF THE BLOCK, NAMELY 3.

   C. A SPECIFIER FOR THE ROUTINE=S RESTORE PROCEDURE.

A Q2FEXIT QUADRUPLE IS ALWAYS FOLLOWED BY A Q2FRETURN.

3. Q2↗RESTORE

THIS QUADRUPLE RESTORES A CONDITIONAL RETURN. ITS FIRST FOUR ARGUMENTS ARE LIKE Q2↗ENTRY; ARG5 AND ARG6 ARE UNUSED. ITS ACTIONS ARE:

1. RECONSTRUCT A POINTER TO THE LAST ENTRY BLOCK AND RESET LAST↗ENTRY.

2. SWAP THE ROUTINE↗S PARAMETERS AND STACKED VARIABLES WITH THE CORRESPONDING VALUES IN THE LATEST ENTRY BLOCK

3. POP THE EXIT BLOCK.

THIS QUADRUPLE IS ALWAYS FOLLOWED BY A RETURN.

4. Q2↗OK

THIS NUBBIN COMPRISES THE ROUTINE↗S OK PROCEDURE. IT HAS A VARIABLE NUMBER OF ARGUMENTS DEPENDING ON THE NUMBER OF SAVED VARIABLES, AND USUALLY CONSISTS OF SEVERAL QUADRUPLES.

THE FIRST QUADRUPLE HAS:

ARG2 POINTER TO SPECIFIER FOR FAIL PROCEDURE
ARG3 NUMBER OF SAVED VARIABLES

THE I-TH ADDITIONAL QUADRUPLE CONTAINS:

ARG3 SYMBOL TABLE POINTER TO I-TH SAVED VARIABLE

THE ACTION OF THIS NUBBIN IS

1. GET A BLOCK OF ARG3 + 3 WORDS.

2. ITERATE OVER THE LIST OF SAVED VARIABLES. SET THE SHARE BIT OF EACH SYMBOL TABLE ENTRY AND STORE IT AT HEAP(T-2-I). (T IS THE TOP OF THE STACK.)

3. PUSH RETADDR AND THE FAIL PROCEDURE.

THIS NUBBIN IS ALWAYS FOLLOWED BY A Q2↗RETURN.

6. Q2↗OKOK

THIS NUBBIN HANDLES OKOK. IT IS IDENTICAL TO Q2↗OK EXCEPT THAT IT THREADS THE NEW ENVIRONMENT BLOCK INTO THE LIST OF OKOK BLOCKS.

5. Q2↑FAIL

THIS NUBBIN DEFINES A LOCAL ROUTINE FOR HANDLING FAIL AND REJECT. THE ROUTINE PERFORMS ONE OF TWO ACTIONS BASED ON THE SETTING OF THE GLOBAL FLAG IS↑REJECT, WHICH INDICATES WHETHER WE ARE DOING A FAIL OR A REJECT.

THE NUBBIN CONSISTS OF SEVERAL QUADRUPLES. THE FIRST QUADRUPLE H

ARG3 NUMBER OF VARIABLES BEING RESTORED

THERE IS AN ADDITIONAL QUADRUPLE FOR EACH VARIABLE BEING RESTORED. THE I-TH ADDITIONAL QUADRUPLE CONTAINS:

ARG3 SYMBOL TABLE POINTER TO I-TH VARIABLE BEING RESTORED

THE I-TH VARIABLE IS RESTORED FROM HEAP(T-1-I).
(T POINTS TO THE TOP OF THE STACK.)

THE ACTION OF THIS NUBBIN IS

1. AT THIS POINT THE TOP ENVIRONMENT IS EITHER THE LAST OK BLOCK OR THE LAST OKOK BLOCK; WE CAN TELL WHICH BY COMPARING THE TOP OF THE STACK WITH LAST↑OKOK.

2. IF WE ARE PROCESSING AN OKOK BLOCK POP THE PREVIOUS VALUE OF LAST↑OKOK. POP THE RETURN ADDRESS AND SET IS↑REJECT TO NO.

3. IF WE ARE PROCESSING AN OK BLOCK, WE MUST CHECK WHETHER WE ARE ACTUALLY PERFORMING A FAIL OR A REJECT. WE ONLY POP THE RETURN ADDRESS IN THE FORMER CASE.

4. RESTORE EACH OF THE BACKTRACK VARIABLES.

5. POP THE TOP ENVIRONMENT BLOCK.

THIS NUBBIN IS ALWAYS FOLLOWED BY A Q2↑RETURN.

7. Q2↑ACCEPT

THIS NUBBIN CALLS A LIBRARY ROUTINE TO COMPACT THE STACK. THE ROUTINE DELETES ALL OK BLOCKS, EXIT BLOCKS, AND CONDITIONAL RETURN BLOCKS CREATED SINCE THE LAST OKOK. IT IS DRIVEN BY THE SIZE SPECIFIER CONTAINED IN EACH ENVIRONMENT BLOCK.

8. Q2↑ARB

THIS NUBBIN IS A LOCAL PROCEDURE FOR THE NONDETERMINISTIC ARB OPERATION. ITS ARGUMENTS ARE JUST LIKE THOSE OF Q2↑OK. ITS ACTION IS:

1. ASSUME WE ARE DOING X = ARB S. THEN ON ENTRY TO THIS NUBBIN THE TOP STACK ENTRY IS A POINTER TO X, AND THE ENTRY BELOW IT IS A POINTER TO S. USE THESE TO INITIALIZE X TO POINT TO T THE FIRST ELEMENT OF S.

2. BUILD AN OK BLOCK AS DESCRIBED FOR THE Q2↑OK NUBBIN. RATHER THAN PUSHING A POINTER TO THE ROUTINES FAIL PROCEDURE, PUSH A POINTER TO ITS ADVANCE PROCEDURE.

THIS NUBBIN IS ALWAYS FOLLOWED BY A Q2↑RETURN NUBBIN.

9. Q2↑ADVANCE

THIS NUBBIN IS A LOCAL PROCEDURE TO ADVANCE AN ITERATOR FOR A NONDETERMINISTIC ARB. ITS ARGUMENTS ARE LIKE THOSE OF A FAIL NUBBIN. ITS SEMANITCS ARE:

1. GET POINTERS TO X AND S FROM THE STACK. ADVANCE X TO THE NEXT ELEMENT OF S.

2. RESTORE ALL ≠BACK≠ VARIABLES AS IF WE WERE DOING A FAIL.

3. IF X IS OMEGA, POP THE TOP ENVIRONMENT BLOCK.

CHANGES TO THE COMPILER

IN THIS SECTION WE DISCUSS THE CHANGES WHICH MUST BE MADE TO THE COMPILER. WE BEGIN BY REVIEWING THE BASIC COMPILER STRUCTURE.

THE FIRST PHASE OF THE COMPILER IS A COMBINATION SCANNER - PARSER.

THE SECOND PHASE OF THE COMPILER IS CALLED THE SEMANTIC PASS. IT WALKS THE TREE PRODUCED BY THE PARSER, PROCESSING DECLARATIONS AND EXPANDING VARIOUS HIHG LEVEL CONSTRUCTS. THE OUTPUT OF THE SEMANTIC PASS IS AN ANNOTATED SYMBOL TABLE AND A SERIES OF QUADRUPLES KNOWN AS Q1.

THE THIRD PHASE OF THE COMPILER IS CALLED THE CODE GENERATOR. IT EXPANDS THE QUADRUPLES PRODUCED BY THE SEMANTIC PASS INTO A MORE DETIALED SET OF QUADRUPLES KNOWN AS Q2. THESE QUADRUPLES CORRESPOND TO INDIVIDUAL NUBBINS, AND ARE EITHER CONVERTED TO MACHINE CODE OR USED BY THE INTERPRETER.

PARSING

WE WILL NOT DISCUSS THE CHANGES WHICH MUST BE MADE TO THE PARSER SINCE THE BACKTRACKING SYNTAX IS STILL BEING FINALLIZED.

SEMANTIC PROCESSING

IN THIS SECTION WE DISCUSS CHANGES TO THE SEMANTIC PASS WHICH MUST BE MADE TO IMPLEMENT BACKTRACKING.

BACKTRACKING ADDS BOTH STATIC DECLARATIONS AND EXECUTABLE STATEMENTS TO THE LANGUAGE. USUALLY THE SEMANTIC PASS PROCESSES DECLARATIONS BY SETTING VARIOUS FIELDS IN THE SYMBOL TABLE. HOWEVER, FOR VARIOUS TECHNICAL REASONS IT WILL PROCESS THE ≠BACK≠ AND ≠NOBACK≠ DECLARATIONS BY GENERATING SPECIAL QUADRUPLES WHICH INSTRUCT THE CODE GENERATOR TO MAKE TABLE ENTRIES.

THE SEMANTIC PASS SEES BACKTRACKING AS CONSISTING OF THE FOLLOWING PRIMITIVES, EACH OF WHICH IS REPRESENTED BY A NEW OPCODE.

BACK V1, ..., VN:	ADD V1 THROUGH VN TO THE LIST OF BACKTRACKED VARIABLES
NOBACK V1,...VN:	REMOVE V1 THROUGH VN FROM THE LIST OF BACKTRACKED VARIABLES.
OK:	SAVE THE CURRENT STATE OF THE BACKTRACK VARIABLES.
OKOK:	A SPECIAL OK.
FAIL:	RETREAT TO PREVIOUSLY SAVED ENVIRONMENT.
ACCEPT	FINALIZE ALL CHANGES TO ENVIRONMENT SINCE LAST OKOK.
REJECT	REJECT ALL CHANGES SINCE LAST OKOK.
ARB	NONDETERMINISTIC ARB

THE CORRESPONDING OPCODES ARE

Q1⇗BACK	ADD A1, A2, AND A3 TO LIST OF BACKTRACK VARIABLES.
Q1⇗NOBACK	REMOVE A1, A2, AND A3 FROM THE LIST
Q1⇗ARB	A1 = ARB A2
Q1⇗OK	
Q1⇗OKOK	
Q1⇗FAIL	
Q1⇗ACCEPT	
Q1⇗REJECT	

NOTE THAT ONLY THE FIRST THREE OPCODES HAVE OPERANDS.

HIGHER LEVEL CONSTRUCTS

THE DEFERED EVALUATION OPERATOR

```
X = ARB. TH. <E1, E2, ... EN>
```

IS TREATED AS A SYNTACTIC MACRO FOR:

```
CASE ARB. N OF
    (1): X = E1;
    (2): X = E2;
          .
          .
          .
    (N): X = EN;
END CASE;
```

THIS IS EXPANDED BY THE SEMANTIC PASS.

CODE GENERATION

THE PHASE OF THE COMPILER MOST EFFECTED BY BACKTRACKING IS THE CODE GENERATOR. THE CODE GENERATOR WILL HAVE TO BE MODIFIED IN THREE AREAS:

1. THE CODE GENERATOR MUST PROCESS THE BACK AND NOBACK DECLARATIONS BY SETTING VARIOUS FLAGS IN THE SYMBOL TABLE WHENEVER A BACK OR NOBACK QUADRUPLE IS SEEN. UNLIKE MOST INFORMATION WHICH IS PICKED UP FROM DECLARATIONS, THE SETTING OF THESE FLAGS WILL VARY DURING CODE GENERATION. AS A RESULT THEY CANNOT BE SET STATICLY BY THE SEMANTIC PASS.

2. ROUTINE PROLOGUES ARE SOMEWHAT MORE COMPLEX THAN PREVIOUSLY. THE BASIC CHANGE HERE IS IN THE WAY THE Q2→ENTRY QUADRUPLE IS EMITTED.

3. ROUTINE EPILOGUES MUST CONTAIN NOT ONLY CODE FOR RETURNS, BUT FOR OKS, FAILS, ETC.