E. Satterthwaite
January 1968
CGTM # 39

B5500 Extended Algol to Algol W Translation

1. Introduction

ALGOL W is an extension of ALGOL 60 based upon suggestions of N. Wirth and C.A.R. Hoare [2]. The main modifications to ALGOL 60 are:

(1) the enrichment of the set of elementary data types to: integer, real, long real, complex, long complex, logical (Boolean), bits, string, reference.

(2) the addition of general plex processing facilities, based on the concepts of record classes and references.

(3) the ability to use any complete subarray (e.g., any complete row or column of a matrix) as a procedure actual parameter.

(4) the inclusion of result parameters, which avoid the use of the name parameter mechanism for "output parameters".

(5) the modification of the for statement so that a local integer control variable, to which explicit assignments are prohibited, is used.

A compiler for ALGOL W was written at Stanford by H. Bauer, S. Becker, L. Bumgarner, and S. Graham under the direction of N. Wirth. The language and the use of the compiler are described in the document ALGOL W [1]. The compiler is available on the SLAC 360/75 system.

A program to aid in the conversion of Burroughs B5500 Extended ALGOL to ALGOL W has been written and is available for experimental use. The program is primarily a character set translator; however, several simple syntactic transformations are also made. In most cases, output will require further hand translation; many questionable constructs are flagged by the program. Suggestions for such hand conversion are included in section 4.

The general philosophy in the design of the translator was to produce output having the correct syntactic form so that meaningful diagnostics can be obtained from the ALGOL W compiler. Several ALGOL W constructs, most notably the for statement, are syntactically similar to the corresponding B5500 ALGOL constructs but semantically quite different. In general, the translator neither flags these constructs nor attempts to provide a different, equivalent construct. Although the B5500 ALGOL for-statement can be rewritten as ALGOL W assignment and goto statements, for example, such rewriting is in practice seldom required and would lose the efficiency the ALGOL W for-statement was specifically designed to provide. For alternate approaches to the problem of translation between algorithmic languages, see [4, 5, 6].

The translator was written in PL360 to allow use of the string manipulation capabilities of the System 360 hardware, particularly the special string scanning and translating instructions. It runs under the PL360-ALGOL W OS subsystem [3].

2. Use of the Program

The following steps will normally be required to translate a B5500 source deck to a syntactically correct ALGOL W source file:

(1) Use the translation program on the B5500 source deck to obtain a card deck or disk file.

(2) Correct cards flagged in the translator output.

(3) Use the ALGOL W compiler to attempt to compile the corrected program.

(4) If any statements are flagged by the compiler, correct them and repeat from (3).

a. Translation Options

Any cards with a $ in column 1 in the input deck are not translated and do not appear in the output. Such cards may be punched on an 029 keypunch. Option keywords are placed on such cards; they must be punched beginning in column 2.

(1) Sequencing
The printed output is always sequenced. A keyword of SEQ begins sequencing of the card-image output; a keyword of NOSEQ terminates it. The initial option is SEQ. It is recommended that punched cards be sequenced and card files unsequenced.

(2) Precision
A keyword of LONG causes the translator to enter long precision mode; a keyword of SHORT causes the translator to enter short precision mode. The initial option is SHORT. In long precision mode. LONG is prefixed to every occurence of REAL in the translator output; in short mode, it is not.

b. Deck Set-Up

See Appendix III for the deck setup currently required at SLAC.

3. Symbol Mapping

With the exception of the special constructs and symbols below, input symbols are written directly into the output stream (with suitable translation of internal and card-punch codes to provide identical graphics).

a. Symbol Translation

B5500 Symbol	ALGOL W Symbol	Notes
←	:=	
[	(	
]	)	
⊗	*	
*	**	exponentiation only
@	'	
:	::	array declarations only
≠	¬=	
≤	<=	
≥	>=	
ARRAY	REAL ARRAY	if not explicitly typed
BOOLEAN	LOGICAL	
NOT	¬	
LABEL	COMMENT LABEL	
ALPHA	STRING(6)	

b. Additional Transformations

(1) Procedure declaration headings are rewritten according to the ALGOL W syntax.

(2) Array declarations are rewritten, if necessary, so that only one bound pair list occurs per declaration.

(3) DEFINEd identifiers are expanded to the defined text.

(4) LONG REAL is substituted for REAL in long precision mode. Note that the precision mode at the time of expansion, not definition, is used for DEFINEd text.

4. Translation Flags

Detection of certain constructs in the B5500 program will cause the translator to flag the corresponding output card image by placing a "?" in column 73. These flags are associated with key words or symbols as listed below

1. PROCEDURE

a. Typed procedures must terminate with an expression which has the desired value. A safe, but usually non-optimal, conversion algorithm is the following:

(1) Enclose the procedure body by BEGIN and END if it is not already so enclosed;

(2) Declare in the procedure body a local identifier with the name and type of the procedure;

(3) Place a semicolon followed by that identifier immediately before the END terminating the block constituting the procedure body.

Example

B5500:
```
REAL PROCEDURE AVG (A,N); VALUE N; REAL ARRAY A[0]; INTEGER N;
BEGIN    INTEGER I; REAL SUM;
   SUM ← 0;
   FOR I ← 1 STEP 1 UNTIL N DO    SUM ← SUM + A[I];
   AVG ← SUM/N
END
```

ALGOL W:
```
REAL PROCEDURE AVG (REAL ARRAY A(*); INTEGER VALUE N);
BEGIN   INTEGER I; REAL SUM;
   REAL AVG;
   SUM := 0;
   FOR I := 1 STEP 1 UNTIL N DO    SUM := SUM + A(I);
   AVG := SUM/N
   ;AVG
END
```

ALGOL W: (better)
```
REAL PROCEDURE AVG (REAL ARRAY A(*); INTEGER VALUE N);
BEGIN REAL SUM;
   SUM := 0;
   FOR I := 1 UNTIL N DO   SUM := SUM + A(I);
   SUM/N
END
```

b. Labels (as well as switches, files, formats, etc.) may not be used as procedure parameters in ALGOL W. Transfer statements may, however, be used as parameters of type procedure.

Example

B5500:
```
PROCEDURE RECIPSUMSQ (X, Y, R, L); VALUE X, Y;
REAL X, Y, R; LABEL L;
BEGIN REAL T;
   T ← X*2 + Y*2;
   IF T = 0 THEN GO TO L;
   R ← 1/T
END
```

ALGOL W:
```
PROCEDURE RECIPSUMSQ (REAL VALUE X, Y; REAL R; PROCEDURE S);
BEGIN REAL T;
   T := X**2 + Y**2;
   IF T = 0 THEN S;
   R := 1/T
END
```

[call:
```
RECIPSUMSQ(A,B,C, GO TO M) ]
```

2. OWN

No provision for OWN variables is made in ALGOL W. Declarations of such variables should be moved to an outer block.

3. AND, OR, ¬

The relative precedences of these operators and the relational operators have been changed in ALGOL W. In the following examples, the extra parentheses are required to maintain equivalence.

Examples

B5500: `A ⩽ B AND C > 0 OR VAR`
`(A ≠ 0 OR B ≠ 0) AND C > 0`

ALGOL W: `(A <=B) AND (C>0) OR VAR`
`((A¬=0) OR (B¬=0)) AND (C>0)`

4. READ, WRITE, FORMAT, FILE

ALGOL W provides only one (card-image) input file and only one (line-image) output file. No provision for specification of format is provided. Note, however, that string literals may be included in ALGOL W output expression lists.

Example

B5500: `WRITE(<"A= ", E14.8>, A)`

ALGOL W: `WRITE("A= ", A)`

5. DOUBLE

ALGOL W does not provide DOUBLE statements, but variables may be declared as LONG REAL and used in arithmetic expressions, in which all operands should be of type LONG REAL or INTEGER.

Example

B5500:
```
REAL PROCEDURE INNERPROD (I,L,U,V1,V2); VALUE L,U;
INTEGER I,L,U; REAL V1,V2;
BEGIN   REAL SUMHI, SUMLO;
     SUMHI ← SUMLO ← 0;
     FOR I ← L STEP 1 UNTIL U DO
        DOUBLE (V1, 0, V2, 0, ⊗, SUMHI, SUMLO, +,←, SUMHI, SUMLO);
     INNERPROD ← SUMHI
END
```

ALGOL W:
```
REAL PROCEDURE INNERPROD
   (INTEGER I; INTEGER VALUE L,U; REAL V1,V2);
BEGIN   LONG REAL SUM;
   SUM := 0; I := L;
   WHILE I <= U DO
   BEGIN  SUM := SUM + V1*V2; I := I + 1;
   END;
   SUM
END
```

6. SWITCH

ALGOL W does not allow switch lists or designational expressions; however, the CASE statement may often be used to achieve the same effect.

Example

```
B5500:      LABEL L1, L2, L3, L;        . . .
            SWITCH SW ← L1, L2, L3;     . . .
            GO TO SW [I];
   L1:      A ← P1(X,Y); GO TO L;
   L2:      A ← P2(X,Y); GO TO L;
   L3:      A ← 0; B ← TRUE;
   L:        . . .

ALGOL W:    CASE I OF
            BEGIN
                 A := P1(X,Y);
                 A := P2(X,Y);
                 BEGIN A := 0; B := TRUE;
                 END
            END
```

7. FILL

ALGOL W does not provide a comparable statement.

8. ALPHA, STREAM, Partial Word Designators

Programs using these constructs extensively will generally require substantial rewriting to use ALGOL W STRING or BITS variables. Note that the substring designator is quite similar to the B5500 partial word designator.

Example

```
B5500:      ALPHA A, B;
            A ← "STRING";
            B.[12:6]← "$"; B.[18:30] ← A.[12:30];

ALGOL W:    STRING(6) A,B ;
            A := "STRING" ;
            B(0|1) := "$"; B(1|5) := A(0|5);
```

9. Octal Constants

ALGOL W does not provide octal constants. Bit sequences are specified as hexadecimal constants; however, there is currently no clean method of converting between a real number and its BITS representation.

5. Other Problem Areas

Most additional incompatibilities between B5500 ALGOL and ALGOL W will be detected when an attempt is made to compile the translated text. At least two subtle difficulties will not be detected, however.

1. For Variable

In ALGOL W, the controlled variable of a FOR statement is implicitly declared and is unknown outside the controlled statement. The translator, however, does not delete the B5500 explicit declaration; hence references to that variable outside the controlled statement will not be detected as errors by the compiler but will be references to generally unpredictable values.

Example

B5500:
```
PROCEDURE LOOKUP (ARG, TABLE, INDEX); VALUE ARG;
INTEGER ARG, INDEX; INTEGER ARRAY TABLE [0];
BEGIN  INTEGER I; LABEL L;
     FOR I ← 1 STEP 1 UNTIL TABLE [0] DO
       IF ARG = TABLE [I]  THEN GO TO L;
L:   INDEX ← I
END
```

ALGOL W:
```
PROCEDURE LOOKUP (INTEGER VALUE ARG; INTEGER ARRAY TABLE(*);
  INTEGER INDEX);
BEGIN
     FOR I := 1 UNTIL TABLE (0) DO IF ARG = TABLE(I) THEN
     BEGIN INDEX := I; GO TO L
     END;
L:   END
```

2. Labels

In ALGOL W, labels are implicitly declared within the nearest enclosing BEGIN - END pair, i.e., a compound statement may be implicitly a block. In certain cases, GO TO statements will have different effects after translation.

Example

B5500:
```
BEGIN  LABEL L, L1;      . . .
    L: L1:    . . .
    BEGIN LABEL L, L2;
       . . .
       BEGIN
       L: L2:  . . .
       END;
       GO TO L;
    END;
END
```

In the B5500 version execution of the GO TO statement will result in transfer to the point labeled L2. In translation, the label declarations are converted to comments. Execution of the GO TO statement in the ALGOL W version will result in transfer to the point labeled L1.

References

[1] Henry R. Bauer, Sheldon Becker, and Susan L. Graham, ALGOL W, Computer Science Department, Stanford (January 1968).

[2] Niklaus Wirth and C.A.R. Hoare, "A contribution to the development of ALGOL", Comm. ACM 9 (June 1966), pp. 413-432.

[3] Niklaus Wirth (Ed.), "The PL360 System", Technical Report CS68, Computer Science Department, Stanford (June 1967).

[4] Ira Pohl, "Translation from B5500 extended ALGOL to OS/360 ALGOL", CGTM 23, SLAC (September 1967).

[5] Wayne Wilner, "ALGOL to FORTRAN translation", CGTM 15, SLAC (June 1967).

[6] Lanse Leach, "FORTRAN to PL/1 translator", Document No. 33-78-2, Stanford Computation Center (August 1967).

Appendix III

The translation program is available as a PL360 object deck from E. Satterthwaite. The following deck setup is currently required for use of the program at SLAC.

```
         {Job  Card}
//JØBLIB   DD   DSNAME=PUB.EHS.PL360,UNIT=2314,VOLUME=SER=PUB001,  X
//              DISP=(OLD,PASS)
//PL360    EXEC PGM=ALGØLW,PARM=C32760
//DEVICE1  DD   SYSØUT=A,DCB=(BLKSIZE=133,BUFNØ=1)
//DEVICE3  DD   SYSØUT=B,DCB=(BLKSIZE=80,BUFNØ=1)
//DEVICE4  DD   DSNAME=PUB.EHS.$PL360,UNIT=2314,VOLUME=SER=PUB001, X
//              DCB=(RECFM=UT),DISP=(ØLD,KEEP)
//DEVICE2  DD   *
%LØAD
           {translator object deck}
%EØF
           {B5500 ALGOL source deck}
%EØF
/*
```