A PRECOMPILE FOR THE
FORMULA MANIPULATION SYSTEM TRIGMAN

WILLIAM H. JEFFERYS
Dept. of Astronomy, Univ. of Texas at Austin, Austin, Tex., U.S.A.

(Received 30 November, 1971)

Abstract. A translator has been written which simplifies the programming of problems with the formula manipulation system TRIGMAN. It allows for the introduction of a new data type, SERIES, into a FORTRAN program, and translates the user’s program into legal FORTRAN. It should not be difficult to adapt it for other formula manipulation systems now being used in Celestial Mechanics.

1. Introduction

A number of systems for the automated manipulation of algebraic expressions in Celestial Mechanics have now been developed (Barton, 1966; Chapront and Mangeney-Ghertzman, 1968; Rom, 1970, 1971; Jefferys, 1970a, b). They have been applied to a variety of interesting problems. All of these systems are programmed for an application in the same basic way: A series of calls to the appropriate subroutines in the formula manipulation package is written, and the program is then compiled (e.g., with the local FORTRAN compiler) to produce an object program for execution.

For example, a program to generate Legendre polynomials in the function $E \cdot \cos(U)$ might be written for the TRIGMAN subroutine package as follows:

```fortran
DIMENSION P(11)
CALL INIT (0,1,1)
CALL TERM (1.0, 2HE., 3HCOS, 2HU., X)
CALL CONST (1.0, P(1))
CALL TRIGADD (ZERO, X, P(2))
DO 1 I=2, 10
   CALL RATDEF (2*I-1, I, COEFF)
   CALL TRIGMPY (COEFF, X, P(I), TEMP)
   CALL RATDEF (I-1, I, COEFF)
   CALL TSCMPY (COEFF, P(I-1), TEMP2)
   CALL TRIGSUB (TEMP, TEMP2, P(I+1))
1   CALL OUTPUT (1HP, P(I+1))
END
```

In this program, INIT initializes the routines, TERM creates a series, as specified, CONST creates a constant series, RATDEF creates a rational multiplier, and the rest of the program should be fairly intelligible as a prescription for the recursion relationship for Legendre polynomials, if it is remembered that because FORTRAN
arrays start with a subscript 1 instead of 0, it has been necessary to depart from the standard labelling for Legendre polynomials.

As shown by the example, a typical program written for the TRIGMAN system consists of a sequence of calls to the needed subroutines, plus a few additional statements to control the flow of the program. It is cumbersome to write, more difficult than need be to debug, and the meaning of the individual steps is greatly obscured. It is not intolerable to have to live with these drawbacks, but a preferable route would be to invent a higher-level language in which it is easier to express ones problem, and to write a translator or precompiler which takes a program written in this higher level language and rewrites it so that it will be acceptable to the local compiler.

Such a program has now been written for use with the TRIGMAN system. It is relatively short (about 500 lines of code) and is written in the string processing language SNOBOL. Since SNOBOL is available on many modern machines, it is to be expected that the precompiler could easily be adapted for other formula manipulation systems.

To demonstrate the improved appearance and readability of a program, one can now write the example above as follows:

```fortran
POLY  E
TRIG  U
SERIES P(11), X
P(1) = 1
P(2) = E*COS(U)
X = P(2)
DO 1 I = 2, 10
   P(I+1) = (2*I-1)/I*P(I)*X - (I-1)/I*P(I-1)
1  OUTPUT P(I+1)
END
```

The first two lines declare the variable letters E and U, while the third declares the variable P to be an array whose values are series. The rest of the program is self-explanatory.

2. A Brief Description of the Translator

The precompiler for this system is called TRIGRUN; it accepts a program written in the FORTRAN-like language of the previous example and produces a program acceptable to the local FORTRAN compiler. The language is actually a superset to FORTRAN, that is, every statement which is legal in FORTRAN is also legal in this language, and some additional statements are also acceptable. The new statements fall into three categories:

1) Declaration statements (POLY, TRIG and SERIES);
2) Input/Output statements for series;
3) Assignment statements involving series.

A method has also been provided so that the user can, with a single control card,