D. E. KNUTH, Editor

A Nonrecursive Method of Syntax Specification

JOHN W. CARR III AND JEROME WEILAND
The Moore School, University of Pennsylvania, Philadelphia, Pennsylvania

The use of the Kleene regular expression notation for describing algebraic language syntax, in particular of ALGOL, is described in this paper. A FORTRAN II computer program for carrying out the elimination algorithm of Gorn, similar to Gaussian elimination for linear systems of algebraic equations, is described. This was applied to numerous smaller languages, including some sublanguages of ALGOL. A hand calculation result of the application of the algorithm to all of ALGOL is given, thus expressing the Revised ALGOL 1960 syntax in completely nonrecursive terms, as far as its context-free portion is concerned. This description in many ways is far more intuitively understood than the previous recursive description, it is suggested. The paper also includes results of the machine program, which does not include a simplification algorithm.

The basis and the method to produce a new approach to computer language syntax specification is outlined in this paper. Given a recursive specification for a context-free language in standard so-called "Backus Normal Form" (BNF) [1] a nonrecursive specification can be produced by using the elimination algorithm of Gorn [2]. The elimination algorithm will solve a set of "equations" (i.e., productions) in BNF in a way similar to that of the standard Gaussian elimination. The elimination algorithm will remove all recursion only on linear languages or Chomsky Type 3 languages [3]. (By "linear" we actually mean "one-sided linear.") If the language is not linear the recursion in the linear portions of the language can be removed, thus producing an overall reduction in recursion in the specification of the language. The basis of the elimination algorithm is:

$$\langle a \rangle ::= \langle a \rangle \langle b \rangle \mid \langle c \rangle$$

is replaced by

$$\langle a \rangle ::= \langle c \rangle (\langle b \rangle)*$$

Here the asterisk indicates zero or more occurrences but not necessarily the same strings drawn from the set $\langle b \rangle$.

After applying the elimination algorithm to a linear language the nonrecursive specification produced is in the form of a regular expression as described by Kleene [4]. A program has been written in FORTRAN II which will take as input a BNF specification for a linear language and produce as output the regular expression [5]. An inverse process has been programmed by Roberts [6]. That context-free portion of ALGOL as defined by Naur [7] was used as input to the program. Upon using Section 2.5.1 (⟨number⟩) of Naur as input, a sample of the output from the program is shown in Figure 1.

If the language input to the program is nonlinear the program will go through the elimination procedure an equation (production) at a time until a nonlinear equation is reached. Then the program produces results obtained up to this equation and then stops. The results of the program on ALGOL as well as other hypothetical languages are given by Weiland [5].

The output of the program demonstrates the fact that the regular expressions produced at the end of the back substitution are large and unwieldy to work with except in a computer with a very large memory. The work of Iverson [8] inspired a compact nonrecursive specification of the context-free portion of ALGOL which is presented in Table I. Table I was produced by a hand calculation rather than the machine program, since the program does not include efficient simplification algorithms. Back substitution has not been carried out in general in Table I, in order to keep the specifications compact. Since each equation (production) as originally formulated is im-

```
U I = D I ( D I ) * ,

I N = ( L / 0 / 1 ) D I ( D I ) * ,

D F = 2 D I ( D I ) * ,

E P = 1 0 ( L / 0 / 1 ) D I ( D I ) * ,

D N = ( L / D I ( D I ) * ) 2 D I ( D I ) * / L D I ( D
I ) * ) ,

U N = ( L / ( ( L / D I ( D I ) * ) 2 D I ( D I ) * / L
D I ( D I ) * ) ) 1 0 ( L / 0 / 1 ) D I ( D I ) * / L (
( L / D I ( D I ) * ) 2 D I ( D I ) * / L D I ( D I ) *
) ) ,

N U = ( L / 0 / 1 ) ( ( L / ( ( L / D I ( D I ) * ) 2 D
I ( D I ) * / L D I ( D I ) * ) ) 1 0 ( L / 0 / 1 ) D I
( D I ) * / L ( ( L / D I ( D I ) * ) 2 D I ( D I ) * /
L D I ( D I ) * ) ) ,
```

FIG. 1

TABLE I

Name	*Nonrecursive Specification*
1.1	
EMPTY STRING	$\varphi ::=$
2.1	
LETTER	$\langle LE\rangle ::= a\mid b\mid c\mid \cdots \mid z\mid A\mid B\mid C\mid \cdots \mid Z$
2.2.1	
DIGIT	$\langle DI\rangle ::= 0\mid 1\mid 2\mid 3\mid 4\mid 5\mid 6\mid 7\mid 8\mid 9$
2.2.2	
LOGICAL VALUE	$\langle LV\rangle ::= \mathbf{true}\mid\mathbf{false}$
2.3	
DELIMITERS	(identical with ALGOL 1960 Revised Report)
2.4.1	
IDENTIFIER	$\langle ID\rangle ::= LE\ (LE\mid DI)*$
2.5.1	
UNSIGNED INTEGER	$\langle UI\rangle ::= \langle DI\rangle\langle DI\rangle*$
INTEGER	$\langle IN\rangle ::= (\varphi\mid+\mid-)\langle UI\rangle$
DECIMAL FRACTION	$\langle DF\rangle ::= .\langle UI\rangle$
EXPONENT PART	$\langle EP\rangle ::= {}_{10}\langle IN\rangle$
DECIMAL NUMBER	$\langle DN\rangle ::= (\varphi\mid\langle UI\rangle)\langle DF\rangle\mid\langle UI\rangle$
UNSIGNED NUMBER	$\langle UN\rangle ::= \langle DN\rangle\mid(\varphi\mid\langle DN\rangle)\langle EP\rangle$
NUMBER	$\langle NU\rangle ::= (\varphi\mid+\mid-\mid)\langle UN\rangle$
2.6.1	
PROPER STRING	$\langle PS\rangle ::= \langle AN\rangle\mid\varphi$
ANY SEQUENCE OF BASIC SYMBOLS NOT CONTAINING "OR"	$\langle AN\rangle$
OPEN STRING	$\langle OS\rangle ::= \langle PS\rangle\mid'\langle OS\rangle'\mid\langle OS\rangle\langle OS\rangle$
STRING	$\langle ST\rangle ::= '\langle OS\rangle'$
3.	
EXPRESSION	$\langle EX\rangle ::= \langle BE\rangle\mid\langle AE\rangle\mid\langle DE\rangle$
3.1.1	
VARIABLE IDENTIFIER	$\langle VI\rangle ::= \langle ID\rangle$
SIMPLE VARIABLE	$\langle SV\rangle ::= \langle VI\rangle$
SUBSCRIPT EXPRESSION	$\langle SE\rangle ::= \langle AE\rangle$
SUBSCRIPT LIST	$\langle SL\rangle ::= \langle SE\rangle(,\langle SE\rangle)*$
ARRAY IDENTIFIER	$\langle AI\rangle ::= \langle ID\rangle$
SUBSCRIPTED VARIABLE	$\langle SR\rangle ::= \langle AI\rangle[\langle SL\rangle]$
VARIABLE	$\langle VA\rangle ::= \langle SV\rangle\mid\langle SR\rangle$
3.2.1	
PROCEDURE IDENTIFIER	$\langle PI\rangle ::= \langle ID\rangle$
ACTUAL PARAMETER	$\langle AP\rangle ::= \langle ST\rangle\mid\langle EX\rangle\mid\langle AI\rangle\mid\langle SW\rangle\mid\langle PI\rangle$
LETTER STRING	$\langle LS\rangle ::= \langle LE\rangle\langle LE\rangle*$
PARAMETER DELIMITER	$\langle PD\rangle ::= ,\mid)\langle LS\rangle:($
ACTUAL PARAMETER LIST	$\langle AL\rangle ::= \langle AP\rangle(\langle PD\rangle\langle AP\rangle)*$
ACTUAL PARAMETER PART	$\langle AT\rangle ::= \varphi\mid(\langle AL\rangle)$
FUNCTION DESIGNATOR	$\langle FD\rangle ::= \langle PI\rangle\langle AT\rangle$
3.3.1	
ADDING OPERATOR	$\langle AO\rangle ::= +\mid-$
MULTIPLYING OPERATOR	$\langle MO\rangle ::= \times\mid/\mid\div$
PRIMARY	$\langle PR\rangle ::= \langle UN\rangle\mid\langle VA\rangle\mid\langle FD\rangle\mid(\langle AE\rangle)$
FACTOR	$(FT) ::= \langle PR\rangle(\uparrow\langle PR\rangle)*$
TERM	$\langle TE\rangle ::= \langle FT\rangle(\langle MO\rangle\langle FT\rangle)*$
SIMPLE ARITHMETIC EXPRESSION	$\langle SA\rangle ::= ((\varphi\mid\langle AO\rangle)\langle TE\rangle)(\langle AO\rangle\langle TE\rangle)*$
IF CLAUSE	$\langle IC\rangle ::= \mathbf{if}\ \langle BE\rangle\ \mathbf{then}$
ARITHMETIC EXPRESSION	$\langle AE\rangle ::= (\langle IC\rangle\langle SA\rangle\ \mathbf{else})*\langle SA\rangle$
3.4.1	
RELATIONAL OPERATOR	$\langle RO\rangle ::= <\mid\leq\mid=\mid\geq\mid>\mid\neq$
RELATION	$\langle RE\rangle ::= \langle SA\rangle\langle RO\rangle\langle SA\rangle$
BOOLEAN PRIMARY	$\langle BP\rangle ::= \langle LV\rangle\mid\langle VA\rangle\mid\langle FD\rangle\mid\langle RE\rangle\mid(\langle BE\rangle)$
BOOLEAN SECONDARY	$\langle BS\rangle := (\varphi\mid\neg)\langle BP\rangle$
BOOLEAN FACTOR	$\langle BF\rangle ::= \langle BS\rangle(\wedge\langle BS\rangle)*$
BOOLEAN TERM	$\langle BT\rangle ::= \langle BF\rangle(\vee\langle BF\rangle)*$
IMPLICATION	$\langle IP\rangle ::= \langle BT\rangle(\supset\langle BT\rangle)*$
SIMPLE BOOLEAN	$\langle SB\rangle ::= \langle IP\rangle(\equiv\langle IP\rangle)*$
BOOLEAN EXPRESSION	$\langle BE\rangle ::= (\langle IC\rangle\langle SB\rangle\ \mathbf{else})*\langle SB\rangle$
3.5.1	
LABEL	$\langle LA\rangle ::= \langle ID\rangle\mid\langle UI\rangle$
SWITCH IDENTIFIER	$\langle SW\rangle ::= \langle ID\rangle$
SWITCH DESIGNATOR	$\langle SG\rangle ::= \langle SW\rangle[\langle SE\rangle]$
SIMPLE DESIGNATIONAL EXPRESSION	$\langle SX\rangle ::= \langle LA\rangle\mid\langle SG\rangle\mid(\langle DE\rangle)$
DESIGNATIONAL EXPRESSION	$\langle DE\rangle ::= (\langle IC\rangle\langle SX\rangle\ \mathbf{else})*\langle SX\rangle$
4.1.1	
UNLABELLED BASIC STATEMENT	$\langle UB\rangle ::= \langle AS\rangle\mid\langle GS\rangle\mid\langle DS\rangle\mid\langle PT\rangle$
BASIC STATEMENT	$\langle BA\rangle ::= (\langle LA\rangle:)*\langle UB\rangle$
UNCONDITIONAL STATEMENT	$\langle US\rangle ::= \langle BA\rangle\mid\langle CS\rangle\mid\langle BL\rangle$
STATEMENT	$\langle SM\rangle ::= \langle US\rangle\mid\langle CD\rangle\mid\langle FS\rangle$
COMPOUND TAIL	$\langle CT\rangle ::= (\langle SM\rangle;)*(\langle SM\rangle\ \mathbf{end})$
BLOCK HEAD	$\langle BH\rangle ::= \mathbf{begin}\ \langle DC\rangle(;\langle DC\rangle)*$
UNLABELLED COMPOUND	$\langle UC\rangle ::= \mathbf{begin}\ \langle CT\rangle$
UNLABELLED BLOCK	$\langle UL\rangle ::= \langle BH\rangle;\langle CT\rangle$
COMPOUND STATEMENT	$\langle CS\rangle ::= (\langle LA\rangle:)*\langle UC\rangle$
BLOCK	$\langle BL\rangle ::= (\langle LA\rangle:)*\langle UL\rangle$
PROGRAM	$\langle PR\rangle ::= \langle BL\rangle\mid\langle CS\rangle$
4.2.1	
LEFT PART	$\langle LP\rangle ::= \langle VA\rangle := \mid\langle PI\rangle :=$
LEFT PART LIST	$\langle LL\rangle ::= \langle LP\rangle\langle LP\rangle*$
ASSIGNMENT STATEMENT	$\langle AS\rangle ::= \langle LL\rangle(\langle AE\rangle\mid\langle BE\rangle)$
4.3.1	
GO TO STATEMENT	$\langle GS\rangle ::= \mathbf{go\ to}\ \langle DE\rangle$
4.4.1	
DUMMY STATEMENT	$\langle DS\rangle ::= \varphi$
4.5.1[1]	
IF STATEMENT	$\langle IS\rangle ::= \langle IC\rangle\langle US\rangle$
CONDITIONAL STATEMENT	$\langle CD\rangle ::= (\langle LA\rangle:)*(\langle IS\rangle(\varphi\mid\mathbf{else}\ \langle SM\rangle)\mid\langle IC\rangle\langle FS\rangle)$
4.6.1	
FOR LIST ELEMENT	$\langle FE\rangle ::= \langle AE\rangle(\varphi\mid\mathbf{while}\ \langle BE\rangle\mid\mathbf{step}\ \langle AE\rangle\ \mathbf{until}\ \langle AE\rangle)$
FOR LIST	$\langle FR\rangle ::= \langle FE\rangle(,\langle FE\rangle)*$
FOR CLAUSE	$\langle FC\rangle ::= \mathbf{for}\ \langle VA\rangle ::= \langle FR\rangle\ \mathbf{do}$
FOR STATEMENT	$\langle FS\rangle ::= (\langle LA\rangle:)*(\langle FC\rangle\langle SM\rangle)$
4.7.1	
PROCEDURE STATEMENT	$\langle PT\rangle ::= \langle PI\rangle\langle AT\rangle$
5.	
DECLARATION	$\langle DC\rangle ::= \langle TD\rangle\mid\langle AD\rangle\mid\langle SD\rangle\mid\langle PA\rangle$
5.1.1	
TYPE LIST	$\langle TP\rangle ::= (\langle SV\rangle,)*\langle SV\rangle$
TYPE	$\langle TY\rangle ::= \mathbf{real}\mid\mathbf{integer}\mid\mathbf{Boolean}$
LOCAL OR OWN TYPE	$\langle OW\rangle ::= (\varphi\mid\mathbf{own})\ \langle TY\rangle$
TYPE DECLARATION	$\langle TD\rangle ::= \langle OW\rangle\langle TP\rangle$
5.2.1	
LOWER BOUND	$\langle LB\rangle ::= \langle AE\rangle$
UPPER BOUND	$\langle UR\rangle ::= \langle AE\rangle$
BOUND PAIR	$\langle BD\rangle ::= \langle LB\rangle:\langle UR\rangle$
BOUND PAIR LIST	$\langle BR\rangle ::= \langle BD\rangle(,\langle BD\rangle)*$
ARRAY SEGMENT	$\langle AY\rangle ::= (\langle AI\rangle,)*(\langle AI\rangle[\langle BR\rangle])$
ARRAY LIST	$\langle AA\rangle ::= \langle AY\rangle(,\langle AY\rangle)*$

[1] IF clause and unconditional statement are not repeated here as in report of NAUR [6].

TABLE I (*Continued*)

Name	*Nonrecursive Specification*
ARRAY DECLARATION	⟨AD⟩ ::= (**array**|(⟨OW⟩ **array**) ⟨AA⟩
5.3.1	
SWITCH LIST	⟨SH⟩ ::= ⟨DE⟩(,⟨DE⟩)*
SWITCH DECLARATION	⟨SD⟩ ::= **switch** ⟨SW⟩ := ⟨SH⟩
5.4.1	
FORMAL PARAMETER	⟨FM⟩ ::= ⟨ID⟩
FORMAL PARAMETER LIST	⟨FA⟩ ::= ⟨FM⟩(⟨PD⟩⟨FM⟩)*
FORMAL PARAMETER PART	⟨FO⟩ ::= φ|(**(**⟨FA⟩**)**)
IDENTIFIER LIST	⟨IF⟩ ::= ⟨ID⟩(,⟨ID⟩)*
VALUE PART	⟨VP⟩ ::= **value** ⟨IF⟩;|φ
SPECIFIER	⟨SP⟩ ::= **string**|**array**|⟨TY⟩ (φ|**array**|**procedure**)|**label**| **switch**|**procedure**
SPECIFICATION PART	⟨SF⟩ ::= (φ|⟨SP⟩⟨IF⟩:)(⟨SP⟩ ⟨IF⟩:)*
PROCEDURE HEADING	⟨PH⟩ ::= ⟨PI⟩⟨FP⟩; ⟨VP⟩⟨SF⟩
PROCEDURE BODY	⟨PO⟩ ::= ⟨SM⟩|⟨CO⟩
PROCEDURE DECLARATION	⟨PA⟩ ::= (φ|⟨TY⟩)(**procedure** ⟨PH⟩⟨PO⟩)

mediately solvable in terms of the asterisk notation, a nonrecursive regular expression is produced. This is a special characteristic, not emphasized up until now, of the published context-free part of ALGOL (as is well known for context-free languages in general).

The section numbers in Table I refer to section numbers in the ALGOL report of Naur [7]. Each string class name of the original report is represented in this Table by a symbol composed of two letters. The reason for choosing two letters was that the program was written to accept two characters for each name and that letters could be used as mnemonics for the actual names. No duplicate symbols were allowed; thus the symbol for the name is not always clearly mnemonic. By the use of the distributive law of concatenation over set union the nonrecursive equations have been factored if possible. The names that are defined recursively in the original ALGOL report [7] are easily recognized by the appearance of a * in the nonrecursive specification.

Since the nonrecursive specification requires parentheses as control characters, whenever right or left parenthesis denote themselves in Table I they are in boldface. Section 4.7.1 was not included in its entirety as in [7] because ⟨Procedure statement⟩ has the same specification as 3.2.1 ⟨Function Designator⟩. The symbol φ is used instead of ⟨empty⟩ to conform more closely to the Kleene notation. In addition, Section 2.3 (Delimiters) has been omitted as an exact duplicate of the report.

In Figure 1, because of the limitations of a standard computer printer, the symbol L has been used instead of φ, 0 for +, 1 for −, and 2 for . .

RECEIVED NOVEMBER, 1965

George E. Forsythe, Editor of a *New Education Department* Invites Contributions

Computing and Education come together in two different ways. First, the digital computer can be programmed into a powerful tool in the educational process itself, for example as a teaching machine or as a processor of records about student progress. We might call this *computers in education*, and we welcome contributions in this area. (If they deal with educational data-processing techniques that are mainly the same as data-processing techniques for other purposes, articles should be directed to another department of *Communications*.) The second confluence of Computing and Education might be called *education in computing*. This deals with matters of curriculum, personnel, and organization in formal education at all levels about the computing and information sciences. We welcome contributions in this area also. Reference [1] is an excellent preliminary report on education in computing, and it is criticized in reference [3]. Reference [2] deals with both our subjects, discussing a use of *computers in education about computing*.

With the great growth of interest in teaching machines and the sudden emergence of numerous university departments of computer science (under various titles), there should be a great deal of valuable material for this department. Let's have it!—G. E. FORSYTHE

REFERENCES

1. ACM Curriculum Committee on Computer Science. An undergraduate program in computer science—preliminary recommendations, *Comm. ACM 8* (Sept. 1965), 543–552.
2. FORSYTHE, GEORGE E. AND WIRTH, NIKLAUS. Automatic grading programs. *Comm. ACM 8* (May 1965), 275–278.
3. PARNAS, DAVID L. On the preliminary report of C³S (letter to the Editor) *Comm. ACM 9* (Apr. 1966), 242–243.

◆

REFERENCES

1. BACKUS, J. W. The syntax and semantics of the proposed international algebraic language of the Zurick ACM-GAMM conference: ICIP. Paris, June 1959.
2. GORN, S. Processors for infinite codes of the Shannon-Fano type. Proc. of a Symp. on Mathematical Theory of Automata, Polytechnic Institute of Brooklyn, 1962.
3. CHOMSKY, N. On certain formal properties of grammars. *Inf. Contr. 2* (June 1959), 137–167.
4. KLEENE, S. C. *Representations of Events in Nerve Nets and Finite Automata.* Automata Studies, Shannon, C., and McCarthy, J. (Eds.). Princeton, 1956.
5. WEILAND, J. N. Applications of the elimination algorithm to recursive language specifications. Master's Thesis, U. of Pennsylvania, May 1965.
6. ROBERTS, M. B. A generalized recognizer for finite state languages. Master's Thesis, U. of Pennsylvania, August 1965.
7. NAUR, P. (Ed.) Revised report on the algorithmic language ALGOL 60. *Comm. ACM 6*, 1 (Jan. 1963), 1–17.
8. IVERSON, K. E. A method of syntax specification. *Comm. ACM 7*, 10 (Oct. 1964), 588–589.