stichting mathematisch centrum

MC

AFDELING INFORMATICA (DEPARTMENT OF COMPUTER SCIENCE)   IW 66/76   NOVEMBER

L. AMMERAAL

A W-GRAMMAR FOR THE SEMANTICS OF INTEGER EXPRESSIONS

Prepublication

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS(MOS) subject classification scheme (1970): 68A30

ACM-Computing Reviews-category: 5.23

A W-grammar for the semantics of integer expressions*)

by

L. Ammeraal

ABSTRACT

For parenthesized integer expressions with operators + and * a W-grammar is presented which not only defines their syntactic structure but also prescribes their (semantic) value.

KEY WORDS & PHRASES: *W-Grammar, Van Wijngaarden grammar, semantics.*

*) This report will be submitted for publication elsewhere.

INTRODUCTION

In [1] Van Wijngaarden gave a W-grammar which generates the numerical value of a-b, where a and b are decimally written integer constants. As discussed by Peck [2], W-grammars (also called two-level-grammars or Van Wijngaarden grammars) were used in the Revised Report on ALGOL 68 [3] more extensively and in a more sophisticated way than in the original Report on ALGOL 68. Especially the syntax of predicates developped by L. Meertens proved to be very useful. In the present paper predicates are applied to the evaluation of simple arithmetic expressions. We restrict ourselves to expressions which are in the usual way composed of integer constants, parentheses and operators for addition and multiplication. An example of such an expression is:

$$(348 * (245 + 708) * 61) + 18 * 4711.$$

For those who are not familiar with the subject, we give a brief discussion on W-grammars in general, in terms of some well-known concepts of formal language theory.

In context-free grammars, there is an essentially finite set of production rules. Exactly the same type of production rules are used in W-grammars, but there are infinitely many of them. They are therefore specified in an indirect way, namely by means of a finite set of hyper-rules together with a finite set of metaproduction rules. Hyper-rules can be viewed as skeleton rules with "formal parameters" that have to be replaced by "actual parameters". These "formal parameters" are called <u>metanotions</u> and usually written in capital letters. The "actual parameters" are called <u>protonotions</u> and often written in small letters. The question which protonotions may be chosen for some metanotion is answered by the set of metaproduction rules. Each metanotion acts as a start symbol of a context free grammar whose production rules are the given set of metaproduction rules. Each sentence thus produced by a metanotion may be used as a protonotion to replace that metanotion in a hyper-rule. This substitution should be done "consistently" i.e. the same protonotion must replace all occurrences of the metanotion involved. When all metanotions in a hyper-rule have been consistently replaced by the appropriate protonotions, we have constructed a <u>production rule</u> of the W-grammar. Protonotions for which production rules can be derived are called notions. The substitution process must of course not be interpreted in the sense that the original hyper-rules are destroyed; they can be used more than once. The working of W-grammars is more precisely described in the Revised Report on ALGOL 68 [3]. Baker [4], Greibach [5], and others give more formal definitions of W-grammars.

Construction of the syntax

As in [3], we shall use the capital letters A,B,...,Z as "large syntactic marks" of which metanotions are composed. The ten digits 0,1,...,9 will be used both as "small syntactic marks" and as (terminal) "symbols". This is possible if we do not use single digits as "notions". Other "small syntactic marks" are the small letters a,b,...,z, the parentheses (and) and the operators + and *.

We now consider the following set of metaproduction rules:

```
E :: T; E + T.                                    (M1)
T :: F; T * F.                                    (M2)
F :: C; (E).                                      (M3)
C :: D; CD.                                       (M4)
D :: 0;1;2;3;4;5;6;7;8;9.                         (M5)
```

The expressions under consideration are then obviously all "terminal metaproduction" of E. Our intension is now to introduce notions like

```
value of 3 + 4 * 5 is
```

along with a set of hyper-rules according to which this typical notion has exactly one terminal production namely 23 (which is indeed a sequence of (terminal) symbols in our terminology).

We start with the hyper-rule

```
value of E is: E yields C, C.                     (H1)
```

Here "E yields C" is a typical "predicate"; its task is to ensure that the decimal integer constant C be the value of the expression E. If this has been achieved, the second C in the hyper-rule is the same constant because of the principle of "consistent substitution". Therefore "E yields C" is to vanish after it has fulfilled its task, in other words its terminal production is to be an empty sequence, provided that C represents the numerical value of E. If C does not correspond to E in this way, the hypernotion "E yields C" must not have a terminal production; any attempt to derive such a terminal production then leads to a "blind alley".

Working with computers, the internal representation of numbers is mostly binary, whereas the decimal system is used on input and output. Similarly, the actual "arithmetic" of our grammar will not be done decimally, although it remains true that decimally written constants occur in (terminal metaproductions of) E and C. We shall, however, not use the binary but rather the unary number system. In this system we shall represent an integer n (n≥0) by n consecutive occurrences of the "small syntactic mark" i. So e.g. the decimal constant 12 corresponds to iiiiiiiiiiii. In informal explanations we shall abbreviate the latter by {12}i. So we have for any integer n ≥ 0:

$$n \equiv \underbrace{\mathrm{ii...i}}_{\text{n times}} \equiv \underbrace{\{n\}\mathrm{i}}_{\substack{\text{informal}\\ \text{notation}}}$$

By virtue of the following metaproduction rules, the metanotions X, Y, and Z stand for "unary constants":

```
X ::  ; Xi.                                                    (M6)
Y :: X.                                                        (M7)
Z :: X.                                                        (M8)
```

The next hyper-rule to be explained is

```
E yields C: E eq X, C eq X.                                    (H2)
```

The following production rule can be derived from (H2):

```
3 * 4 yields 12: 3 * 4 eq iiiiiiiiiiii, 12 eq iiiiiiiiiiii.
```

Predicate "E eq X" (of which "C eq X" is a special case) has to ensure that the unary number (produced by) X corresponds to the expression (produced by) E. The following ten hyper-rules are obvious:

0 eq:.	(H3)
1 eq i:.	(H4)
2 eq ii:.	(H5)
3 eq iii:.	(H6)
4 eq iiii:.	(H7)
5 eq iiiii:.	(H8)
6 eq iiiiii:.	(H9)
7 eq iiiiiii:.	(H10)
8 eq iiiiiiii:.	(H11)
9 eq iiiiiiiii:.	(H12)

Notice that (H3)...(H12) list exactly all production rules that correspond to D eq X. They show that D eq X "vanishes" if and only if X stands for the unary constant that corresponds to the decimal constant represented by D.

The following hyper-rule is used for constants consisting of two or more decimal digits:

CD eq XXXXXXXXXXY: C eq X, D eq Y. (H13)

It says that XXXXXXXXXXY is the unary representation of the decimal constant CD if and only if X is the unary representation of the constant C and Y is the unary representation of the decimal digit D. In (H13) the advantage of the unary number system becomes clear: addition becomes concatenation and to this end syntax rules are most appropriate. The following hyper-rule will therefore not come as a surprise:

E + T eq XY : E eq X, T eq Y. (H14)

Multiplication is recursively reduced to addition (i.e. concatenation) in the following way:

T * F eq X: T eq Y, F eq Z, Y * Z res X.	(H15)
* Z res:.	(H16)
Yi * Z res XZ: Y * Z res X.	(H17)

The only purpose of (H15) is to switch to the unary number system. Explained in terms of elementary arithmetic, (H16) stands for $0 \cdot z = 0$. Rule (H16) is used after repeated application of (H17) which says:

$$(y+1)z = x + z \text{ if and only if } yz = x.$$

To deal with parentheses we need:

(E) eq X: E eq X. (H18)

Since we eventually wish to arrive at (terminal) symbols, we decompose a decimal constant by

CD: C,D. (H19)

We now give a complete listing of our W-grammar:

E :: T; E + T.	(M1)
T :: F; T * F.	(M2)
F :: C; (E).	(M3)
C :: D; CD.	(M4)
D :: 0;1;2;3;4;5;6;7;8;9.	(M5)
X :: ; Xi.	(M6)
Y :: X.	(M7)
Z :: X.	(M8)

value of E is: E yields C,C.	(H1)
E yields C : E eq X, C eq X.	(H2)
0 eq:.	(H3)
1 eq i:.	(H4)
2 eq ii:.	(H5)
3 eq iii:.	(H6)
4 eq iiii:.	(H7)
5 eq iiiii:.	(H8)
6 eq iiiiii:.	(H9)
7 eq iiiiiii:.	(H10)
8 eq iiiiiiii:.	(H11)
9 eq iiiiiiiii:.	(H12)

CD eq XXXXXXXXXXY: C eq X, D eq Y. (H13)

E + T eq XY: E eq X, T eq Y. (H14)

T * F eq X: T eq Y, F eq Z, Y * Z res X. (H15)

* Z res:. (H16)

Yi * Z res XZ: Y * Z res X. (H17)

(E) eq X: E eq X. (H18)

CD: C,D. (H19)

An example

We shall now show that the string of symbols 7054 is a terminal production of the notion

value of 34 + 45 * (67+89) is.

(It is left as an exercise to prove that it is the <u>only</u> terminal production of this notion.)

We have

value of 34 + 45 * (67+89) is $\xrightarrow{H1}$ 34 + 45 * (67+89) yields 7054,7054

Since we want only the four symbols 7054 to be the terminal production, the precicate "34 + 45 * (67+89) yields 7054" must vanish:

34 + 45 * (67+89) yields 7054 $\xrightarrow{H2}$ 34 + 45 * (67+89) eq{7054}i, 7054 eq{7054}i.

We now proceed with second of the predicates produces:

7054 eq {7054}i $\xrightarrow{H13}$ 705 eq {705}i, 4 eq iiii $\xrightarrow{H7}$

705 eq {705}i $\xrightarrow{H13}$ 70 eq {70}i, 5 eq iiiii $\xrightarrow{H8}$

70 eq {70}i $\xrightarrow{H13}$ 7 eq iiiiiii, 0 eq $\xrightarrow{H3}$

7 eq iiiiiii $\xrightarrow{H10}$ empty.

Now follows the actual work:

34 + 45 * (67+89) eq {7054}i $\xrightarrow{H14}$ 34 eq{34}i, 45 * (67+89) eq{7020}i.

We now take it for granted that notions like 34 eq 34{i} vanish.

$$45 * (67+89) \text{ eq } \{7020\}i \xrightarrow{H15}$$
$$45 \text{ eq } \{45\}i,\ (67+89) \text{ eq } \{156\}i,\ \{45\}i * \{156\}i \text{ res } \{7020\}i.$$

The first of these three predicates is trivial. We postpone the third because the second is much simpler:

$$(67+89) \text{ eq } \{156\}i \xrightarrow{H18} 67 + 89 \text{ eq } \{156\}i \xrightarrow{H14}$$
$$67 \text{ eq } \{67\}i,\ 89 \text{ eq } \{89\}i.$$

We will now do the multiplication:

$$\{45\}i * \{156\}i \text{ res } \{7020\}i \xrightarrow{H17}$$
$$\{44\}i * \{156\}i \text{ res } \{6864\}i \xrightarrow{H17}$$
$$\{43\}i * \{156\}i \text{ res } \{6708\}i \xrightarrow{H17}$$
$$\vdots$$
$$ii * \{156\}i \text{ res } \{312\}i \xrightarrow{H17}$$
$$i * \{156\}i \text{ res } \{156\}i \xrightarrow{H17}$$
$$* \{156\}i \text{ res } \xrightarrow{H16} \text{empty}.$$

REFERENCES

[1] VAN WIJNGAARDEN, A., *On the boundary between natural and artificial languages*, in: Bruno Visentini et al, Linguaggi nella società e nella tecnica, Milan, 1970.

[2] PECK, J.E.L., *Two-level grammars in action*. Proceedings of the IFIP Congress, Stockholm, 1974, North-Holland Publishing Company, Amsterdam.

[3] VAN WIJNGAARDEN, A., et al (Eds), *Revised Report on the Algorithmic Language Algol 68*, Springer Verlag 1976.

[4] BAKER, J.L., *Grammars with Structured Vocabulary: a Model for the ALGOL 68 Definition*, Inf. and Control 20 (1972) 351-395.

[5] GREIBACH, S.A., *Some Restrictions on W-Grammars*, Intern. J. Comput. Information Sci. 3 (1974) 289-327.