

What Can We Do about the Unnecessary Diversity of Notation for Syntactic Definitions?

Niklaus Wirth
Federal Institute of Technology (ETH), Zürich, and
Xerox Palo Alto Research Center

Key Words and Phrases: syntactic description
language, extended BNF
CR Categories: 4.20

The population of programming languages is steadily growing, and there is no end of this growth in sight. Many language definitions appear in journals, many are found in technical reports, and perhaps an even greater number remains confined to proprietary circles. After frequent exposure to these definitions, one cannot fail to notice the lack of "common denominators." The only widely accepted fact is that the language structure is defined by a syntax. But even notation for syntactic description eludes any commonly agreed standard form, although the underlying ancestor is invariably the Backus-Naur Form of the Algol 60 report. As variations are often only slight, they become annoying for their very lack of an apparent motivation.

Out of sympathy with the troubled reader who is weary of adapting to a new variant of BNF each time another language definition appears, and without any claim for originality, I venture to submit a simple notation that has proven valuable and satisfactory in use. It has the following properties to recommend it:

1. The notation distinguishes clearly between meta-, terminal, and nonterminal symbols.
2. It does not exclude characters used as metasymbols from use as symbols of the language (as e.g. "|" in BNF).
3. It contains an explicit iteration construct, and thereby avoids the heavy use of recursion for expressing simple repetition.

Copyright © 1977, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

Author's present address: Xerox Corporation, Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304.

4. It avoids the use of an explicit symbol for the empty string (such as $\langle \text{empty} \rangle$ or ϵ).
5. It is based on the ASCII character set.

This meta language can therefore conveniently be used to define its own syntax, which may serve here as an example of its use. The word *identifier* is used to denote *nonterminal symbol*, and *literal* stands for *terminal symbol*. For brevity, *identifier* and *character* are not defined in further detail.

```
syntax      = {production}.
production = identifier "=" expression ".".
expression = term {"|" term}.
term       = factor {factor}.
factor     = identifier | literal | "(" expression ")" |
           "[" expression "]" | "{" expression "}".
literal    = " " " " character {character} " " " " .
```

Repetition is denoted by curly brackets, i.e. $\{a\}$ stands for $\epsilon | a | aa | aaa | \dots$. Optionality is expressed by square brackets, i.e. $[a]$ stands for $a | \epsilon$. Parentheses merely serve for grouping, e.g. $(a|b)c$ stands for $ac | bc$. Terminal symbols, i.e. literals, are enclosed in quote marks (and, if a quote mark appears as a literal itself, it is written twice), which is consistent with common practice in programming languages.

Received January 1977; revised February 1977