PUC

PRE-EDIÇÃO

SIMPOSIO INTERNACIONAL SOBRE METODOLOGIAS PARA O PROJETO E CONSTRUÇÃO DE SISTEMAS DE SOFTWARE E HARDWARE

PATROCINADOR

CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTIFICO E TECNOLOGICO

28 DE JUNHO A 2 DE JULHO DE 1976

RIO DE JANEIRO
BRASIL

Pontificia Universidade Catolica do Rio de Janeiro
Rua Marquês de São Vicente, 209 — ZC-20
Rio de Janeiro — Brasil

PREPRINTS

INTERNATIONAL SYMPOSIUM ON METODOLOGIES FOR THE DESIGN AND CONSTRUCTION OF SOFTWARE AND HARDWARE SYSTEMS

SPONSOR : CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTIFICO E TECNOLOGICO

CO-SPONSORS: CANADIAN INTERNATIONAL DEVELOPMENT AGENCY
GESELLSCHAFT FUR MATHEMATIK UND DATENVERARBEITUNG
NATIONAL SCIENCE FOUNDATION
IBM DO BRASIL

ORGANIZING COMMITTEE:

PROFESSORS

C.J.P. LUCENA
W. BRAUER
N. MACHADO
A.L. FURTADO

DATES

JUNE 28 TO JULY 2, 1976

NOTE

THESE PREPRINTS HAVE BEEN EDITED. THE AUTHORS ARE KINDLY INVITED TO REQUEST ANY CORRECTIONS OR CHANGES FOR THE FINAL EDITION.

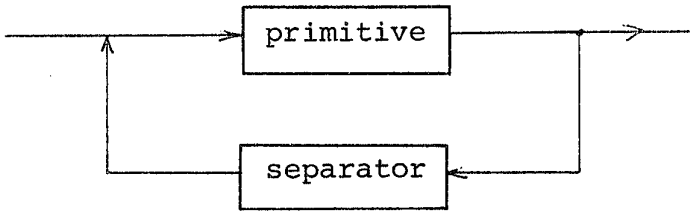
ON GRAMMAR AND LANGUAGE DESIGN FOR IMPROVED ERROR RECOVERY IN LR PARSING

by

Sergio E.R. Carvalho
Departamento de Informática
PUC-RJ

Simplicity of form and meaning is one of the key goals in the design of programming languages. It is generally aggreed that "simple" languages lead naturally to "simple" and "better structured" programs. However, for all practical purposes, a programming language does not exist without its compiler. When designing a language one should be concerned not only with syntax and semantics, but also with other aspects such as run time storage management, error detection and recovery, and code generation. In this paper we explore the conjecture that a simple recovery scheme for syntactic errors can be easily derived from the "simple" syntax of a programming language.

We conjecture that a measure of the simplicity of the syntax of a programming language is the number of "production schemas" in the syntax. To illustrate what we mean by production schema consider an example from Algol 60, the left-recursive schema, which underlies the definitions of <simple arithmetic expression>, <term>, <block head>, <for list>, and others. We can represent the left-recursive schema graphically as follows

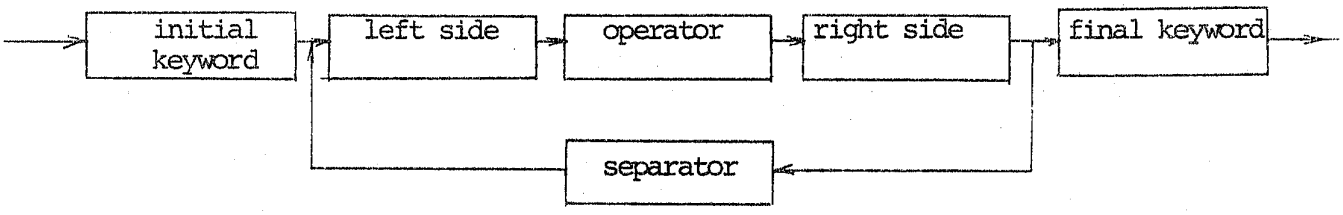


For each nonterminal being defined, primitive and separator take on different meanings; for <term>, the primitive

is <factor> and the separator is <multiplying operator>.

Algol 60 has many production schemas, so its syntax is not "simple".

The following production schema underlies the syntax of most statements in the language we propose.



A triple (initial keyword, operator, final keyword) corresponds unambiguously to some action in the language (declarative or executable).

We expect fewer syntactic errors to be commited in our language due to the fact that there are fewer production schemas. Futhermore, such simple syntax yields a simpler error recovery mechanism, since:

(i) different initial keywords partition the main structure of the syntactic analyzer into a set of smaller analyzers;

(ii) the absence of an initial keyword can be easily corrected by a forward scan until the reserved operator is found;

(iii) the set of delimiters from which "safe" analyzer configurations can be found is considerably increased.