씬택스 컴파일링의 演算子式 分析器를 通한 一般化

논 문
22~6~1

Generalization of Syntax-Directed Compiling by Precedance Analyzer

김 영 택*
(Young Taik Kim)

Abstract

This paper describes a new technique for syntax-directed compiling of algol 60 programming language.

A relatively large size of language specification is devided into two parts, one called language body and the other called language branch.

The language body is compiled by the syntax-directed compiling technique and the other is compiled by the precedance method to minimize the compiling time.

Also the boundary of the portions were studied during this study for the optimization.

1. 序 論

Cheathum과 Sattley[1]가 간단한 數式을 認識하는 Analyzer와 Syntax Type, Structure Table을 作成하여 Syntax-directed compiling을 構成하였다. 그러나 이 槪念을 一般化하여 Algorithmic language 全般에 적용하는 것은 Algorithm 自體가 一般性을 잃고 있기 때문에 無意味하다고 하겠다.

本 論文에서는 Syntax의 Tree Organization에서 기등부분과 가지부분으로 分割하여 기등부분은 Syntax-directed compiling, 가지부분은 Precedance-directed compiling으로 나누어서 Syntax-directed Algorithm의 一般化를 시도하였다.

2. 本 論

아래의 Syntax Spec.에서 이 개념의 實驗을 보기로 한다.

〈Arith expression〉:=〈Simple Ciricth expr〉|
〈if clause〉 〈Simple Arith expr〉 else
〈Arith expr〉

〈if clause〉:=if 〈Boolean expr〉 then
〈Boolean expr.〉:=〈Simple Boolean〉|〈if clause〉
〈Simple Boolean〉 else 〈Boolean expr.〉
〈Simple Boolean〉;=〈implication〉|〈S.B.〉≡〈imp.〉
〈implication〉:=〈B. term〉|〈implication〉⊃
〈B. term〉
〈B. term〉:=〈B. factor〉|〈B. term〉∨〈B. factor〉
〈B. factor〉:=〈B. secondary〉|〈B. factor〉∧
〈B. secondary〉
〈B. secondary〉:=〈B. primary〉|¬〈B. primary〉
〈B. primary〉:=〈B. variable〉|(〈S.B.〉)|
〈logic value〉
〈logic value〉:=L
〈B. variable〉:=B∨
〈Simple a.e.〉:=〈term〉|〈Simple a.e.〉÷〈term〉
〈term〉:=〈factor〉|〈term〉×〈factor〉
〈factor〉:=〈primary〉|〈factor〉|〈primary〉
〈primary〉:=〈integer〉|〈variable〉|(〈s.a.e.〉)
〈variable〉:=∨
〈integer〉:=I

(Syntax Spec I)

Syntax Spec I을 本論文의 對象 Syntax라 하면 이를 本 論文의 槪念대로 축소하면 Syntax Spec II와 같

*정회원 : 서울工大 應用數學科助敎授 (工學博士)

이 된다.

〈Arith. expr.〉:=〈Simple a.e.〉|〈if clause〉

〈simple a.e.〉 else 〈Arith. expr.〉

〈if clause〉:=if〈Boolean expr.〉 then

〈Boolean expr.〉:=〈Simple Boolean〉|〈if clause〉

〈Simple Boolean〉 else 〈Boolean expr.〉

〈Simple arith. expr.〉:=SAE

〈Simple Boolean〉:=SB

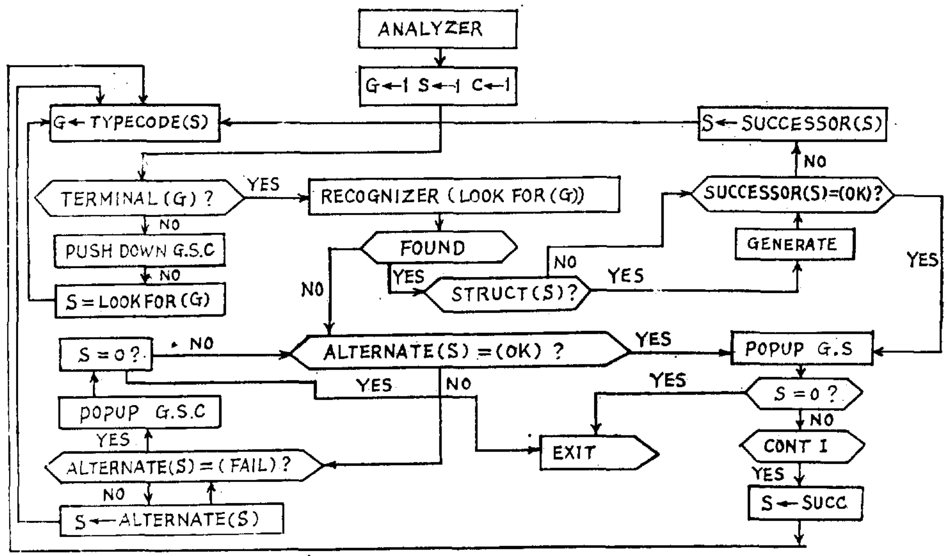
(Syntax Spec Ⅱ)

즉 Syntax Spec Ⅱ는 Syntax Spec Ⅰ의 줄거리 부분이며 그의 가지부분은 SAE나 SB와 같은 Subroutine에 의존하고 있다.

Syntarx Spec Ⅱ를 Compile하기 위하여 Structure Table Ⅰ을 구성하였으며 이의 Analyzing을 위하여 Analyzer Ⅰ을 구성하였다.

SOURCE	SUCC	ALTER	CONTI
1	2	2	No
2	3	F	No
3	4	F	No
4	5	F	No
5	4	F	No
6	7	F	No
7	8	F	Yes
8	OK	F	No
9	10	10	No
10	11	F	No
11	12	F	No
12	13	F	No
13	OK	F	No
14	OK	F	No
15	OK	F	No

(Structure Table Ⅰ)



(Syntax Analyzer Ⅰ)

Syntax Spec Ⅰ에서 Syntax Spec Ⅱ에 나타나지 않는 부분을 가지 부분이라 하며 이를 컴파일 하기 위하여 Precedance Table Ⅰ을 작성하였다.

연산자	(	↑	X ∣ ÷	+ −	=	7	V	V	< \ ≦≧≠	⊐	≡	b
계층순서	1	2	3	4	5	6	7	8	9	10	11	12

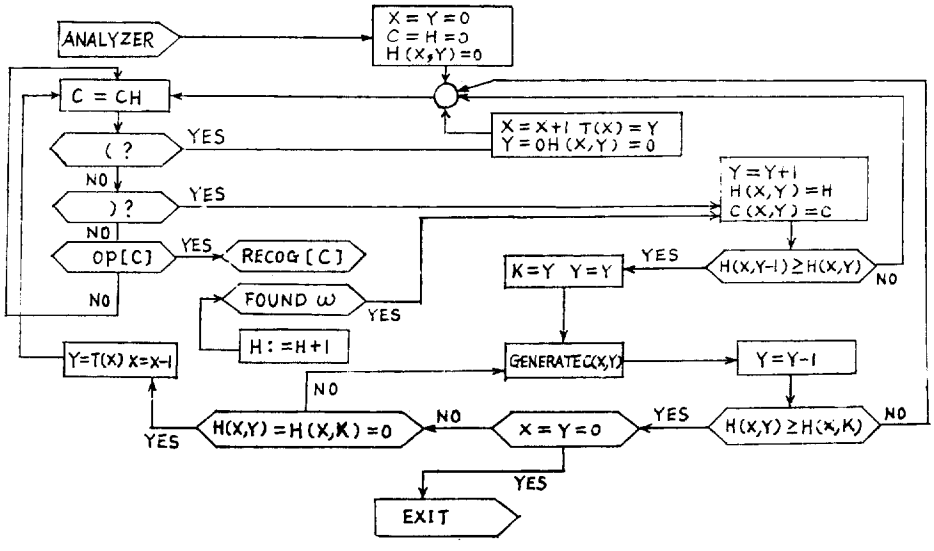
(Precedance Table Ⅰ)

이 Precedance Table을 처리키 위하여 Precedance Analyzer Ⅰ을 작성하였다.

結果的으로 Syntax Spec Ⅰ은 모두 Recognize되며 本來의 複雜性은 이 두가지의 技術의 병용으로 解決되고 있다.

3. 結　論

本 論文에서 가장 중요한 점은 가지부분과 기둥부분의 적절한 구분이다. 이들 區分은 어떠한 조정이나 지도의 槪念이 포함되어야 한다.



(Precedance Analyzer I)

지금 실시한 兩部分의 경계는 operator나 그와 유사한 Algol 單語들에 依하여 실시하였다. 그러므로 더 많은 單語를 利用하면 더많은 Algol 言語의 部分을 處理할수 있지만 precedance operation에서 recursion 槪念의 存在가 위협되므로 적당한 線에서 경계를 짓는 것과 또다른 組織法을 生覺할수도 있다.

참 고 문 헌

1) T.E. Cheathum Jr. and Kirk Sattley, "Syntax Directed Compiling", p.31 Proceedings SJCC 1964.

2) S. Warshall and R.M. Shapiro," A General Purpose Table-driven Compiler", To be Published in the Proceedings SJCC 1964.

3) R.W. Floyd, "Syntactic Anlysis and Operator Precedance; p.316 Jnl ACM Vol. 10. 1963.