2-level grammar를 이용한 교육용 프로그래밍 언어

오솔길† · 박성빈† 1)
† 고려대학교 컴퓨터학과 박성빈

The educational programming language based on 2-level grammar

Sol-Gil Oh† · Seong-Bin Park†
† Dept. of Computer Science and Engineering, Korea University

요 약

W-grammar(Van Wijngaarden grammar)[1].란 형식 문법의 한 종류인 문맥 자유 문법(Context-free grammar, CFG)의 대안으로, 두 개의 CFG가 서로 상호작용하면서 언어를 정의하도록 구성된 문법을 말하며 그 구성상의 특징을 감안하여 2-level grammar라고도 부른다. 현재 많은 프로그래밍 언어는 CFG를 기반으로 구성되어 있는데, 본 연구에서는 W-grammar로 교육용 프로그래밍 언어를 정의하여 CFG 기반이 아닌 W-grammar 프로그래밍 언어의 진입점을 제공함과 동시에 기저의 문법에 따른 계산가능성에 대해서 연구하는 토대를 쌓는 것을 목표로 한다.

1. 서 론

Van Wijngaarden 이 제안한 W-grammar는 해당 문법에 속한 모든 언어를 재귀열거집합(recursively enumerable set)으로 정의할 수 있는 높은 정의 파워를 지닌 문법이다[2].

현재 수많은 프로그래밍 언어의 근간을 이루는 Context Free Grammar(CFG)는 그 계산 가능성에 한계가 있다.[3]. 예를 들어, 가장 널리 알려진 CFG로 표현할 수 없는 language인 non-Context Free Language는 $\{a^n b^n a^n | n \geq 1\}$이다.

반면 이는 W-grammar로는 쉽게 표현 가능한데, 이러한 특징은 CFG 위주의 현 프로그래밍 언어에 새로운 관점을 제시하리라고 예상되어 왔다. 그러나 W-grammar는 Algol 68에서 그 자신을 기반으로 한 프로그래밍 언어를 제안하였으나 W-grammar 특유의 복잡성으로 인해 광범위하게 사용되지는 못하였다.

본 연구에서는 이러한 단점을 보완하기 위하여 접근성이 낮은 교육용 프로그래밍 언어를 제안하며 이러한 접근법이 계산가능성에 대한 연구토대에 기여가 되기를 기대한다.

2. 2-level grammar와 제안하는 예시

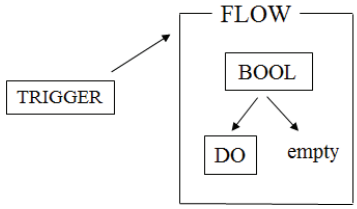
W-grammar는 $G=(V_M, V_P, \Sigma, P_M, P_h, \sigma)$ 로 구성 된다. V_M 은 metavariable을 의미한다. 유한한 집합 V_P는 V_P에 속한 terminal들인 Σ와, protovariable 이라고 불리는 $V_P - \Sigma$로 나누어진다. P_M은 metaproduction이라고 불리는 유한한 집합이고, P_h는 hyperrule이다. 시작 심볼인 σ는 $\sigma \in V_P - \Sigma$이다. W-grammar는 다음과 같은 세가지 조건을 만족시켜야 하는데, 각각의 조건은 다음과 같다. (1) $V_P \cap V_M = \varnothing$ 이며, "<"와 ">"는 $V_P \cup V_M$ 에 속하지 않는다. (2) $A \in V_M$이라고 할 때, $G_A = (V_M, V_H, P_M, A)$ 는 문맥 자유 문법이다. (3) P_h에 속한 각각의 hyperrule은 $Z \rightarrow y$의 형태를 가지는데 $Z \in H \cup (V_P - \Sigma)$이고 $y \in (V_M \cup V_P \cup H)^+$ 이다. 여기에서 $H = \{\langle \alpha \rangle | \alpha \in (V_P \cup V_M)^+\}$ 이며 H 는 hypernotion 이라고 부른다.[4].

Metarules
TRIGGER－>start, mouse_clicked
FLOW －>if
BOOL -> isclicked
DO－> PRINT
VARIABLE－>NUMBER
NUMBER -> {INTEGER}$^+$
EMPTY ->.
Hyperrules

1) 교신저자

start -> <TRIGGER>
<TRIGGER> -> TRIGGER <FLOW>
<FLOW> -> FLOW <BOOL>
<FLOW> -> <FLOW> <FLOW>
<BOOL> -> BOOL <DO> VARIABLE
<BOOL> -> EMPTY

위의 규칙은 프로그래밍 언어의 전반을 2-level grammar를 따르는 형식 언어로 표현한 방식이다. 본 연구에서는, 프로그래밍 언어를 이루는 필수적이고 핵심적인 조건을 세가지를 꼽았다. 해당 세가지 분류는 [그림1]의 흐름도에 등장하는 기능들과 같다. 먼저 TRIGGER는, 일반적인 프로그래밍 언어에서 event를 의미한다. 즉 특정한 조건이 발생했을 때, 이어지는 구문들이 작동하게 된다는 의미를 충족시키기 위해서 핵심적인 조건의 하나로 구분했다. FLOW는 프로그램 전반의 흐름을 나타기 위해서 사용한 nonterminal이다. loop와 if처럼 프로그램 작동구문의 순서를 지정하는 역할을 맡는다. 그림에서와 같이 논리적인 흐름의 전반을 맡고 있으며, 다른 요소들이 그 하위에 포함된다고 본다. BOOL은 일반적으로 사용되는 자료형으로 1과 0으로 참과 거짓을 판별할 수 있는 조건들을 의미한다. 위의 예시에서는 isclicked와 같은 예시를 넣었다. BOOL에서의 판별에 따라, 참일 경우에는 DO로 대표되는 다양한 함수들 중 하나의 기능을 수행하고, 거짓일 경우에는 empty로 향하며 해당 FLOW가 멈춘다.



[그림1] 제안하는 언어의 흐름도

Derivation with metanotion.

1 start -> <mouse_clicked>
2 -> mouse_clicked<if>
3 -> mouse_clicked if <isclicked>
4 -> mouse_clicked if isclicked <print> number

위의 보기는 hyperrule의 진행을 표기한 것으로, mouse_clicked 라는 이벤트가 트리거로서 발생하면 이후 구문들이 시작되고, isclicked라는 bool이 참일 경우 print number로 진행되는 식을 2-level grammar로 작성한 형태이다. 또한, 본 예제에서는 hyperrule의 좌변을 하나의 terminal로 유지하였다. 이는 hyperrule을 문맥 자유롭게 유지할 경우에만 결정론적인 성질을 보증할 수 있기 때문이다[5].

3. 결론 및 논의

본 연구에서는 현재 대부분의 프로그래밍 언어의 토대가 되는 CFG의 계산파워의 한계를 극복하기 위하여 2-level grammar를 기초로 하는 프로그래밍 언어의 기초적인 틀을 제안하고 해당 틀에서 간단한 프로그래밍을 하는 방법을 보였다. 해당 언어의 틀에서는 TRIGGER, FLOW, BOOL을 핵심으로 이하의 기능을 추가하였는데, 이는 현재 산업전반에서 활용되고 있는 SCRATCH [6].나, 이와 유사한 엔트리 등과 유사한 방식을 따랐으며, 따라서 어렵지 않게 블록식 프로그래밍이 가능한 GUI 로 변환할 수 있으리라 생각한다.

참 고 문 헌

[1] van Wijngaarden, A. (1981). Revised report of the algorithmic language algol 68. ALGOL Bulletin, (Sup 47), 1-119.
[2] Sintzoff, M. (1967). Existence of a van Wijngaarden syntax for every recursively enumerable set. ANNALES DE LA SOCIETE SCIENTIFIQUE DE BRUXELLES SERIES 1-SCIENCES MATHEMATIQUES ASTRONOMIQUES ET PHYSIQUES, 81(2), 115.
[3] Hartmanis, J. (1967, December). Context-free languages and Turing machine computations. In Proceedings of Symposia in Applied Mathematics (Vol. 19, pp. 42-51).
[4] Greibach, S. A. (1974, April). Some restrictions on W-grammars. In Proceedings of the sixth annual ACM symposium on Theory of computing (pp. 256-265). ACM.
[5] Edupuganty, B., & Bryant, B. R. (1989). Two-level grammar as a functional programming language. The computer journal, 32(1), 36-44.
[6] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. Communications of the ACM, 52(11), 60-67.