# CFG 표기법: BNF와 EBNF

- ( ) 와 택일 기호 '|':
  - BNF
    <exp> ::= <exp>+<exp> | <exp>-<exp> | <exp>*<exp> | <exp>/<exp>
  - EBNF
    <exp> ::= <exp> ( + | - | * | / ) <exp>

- { } : 반복되는 부분 – 위/아래 첨자로 반복횟수를 지정할 수 있음!
  - BNF
    <compound_statement> ::= begin <statement_list> end
    <statement_list> ::= <statement_list>; <statement> | <statement>
  - EBNF
    <compound_statement> ::= begin <statement> {; <statement>} end

- [ ] : 선택적
  - BNF
    <if_stmt> ::= if <condition> then <statement> |
    if <condition> then <statement> else <statement>
  - EBNF
    <if_stmt> ::= if <condition> then <statement> [else <statement>]

# BNF와 EBNF 예제

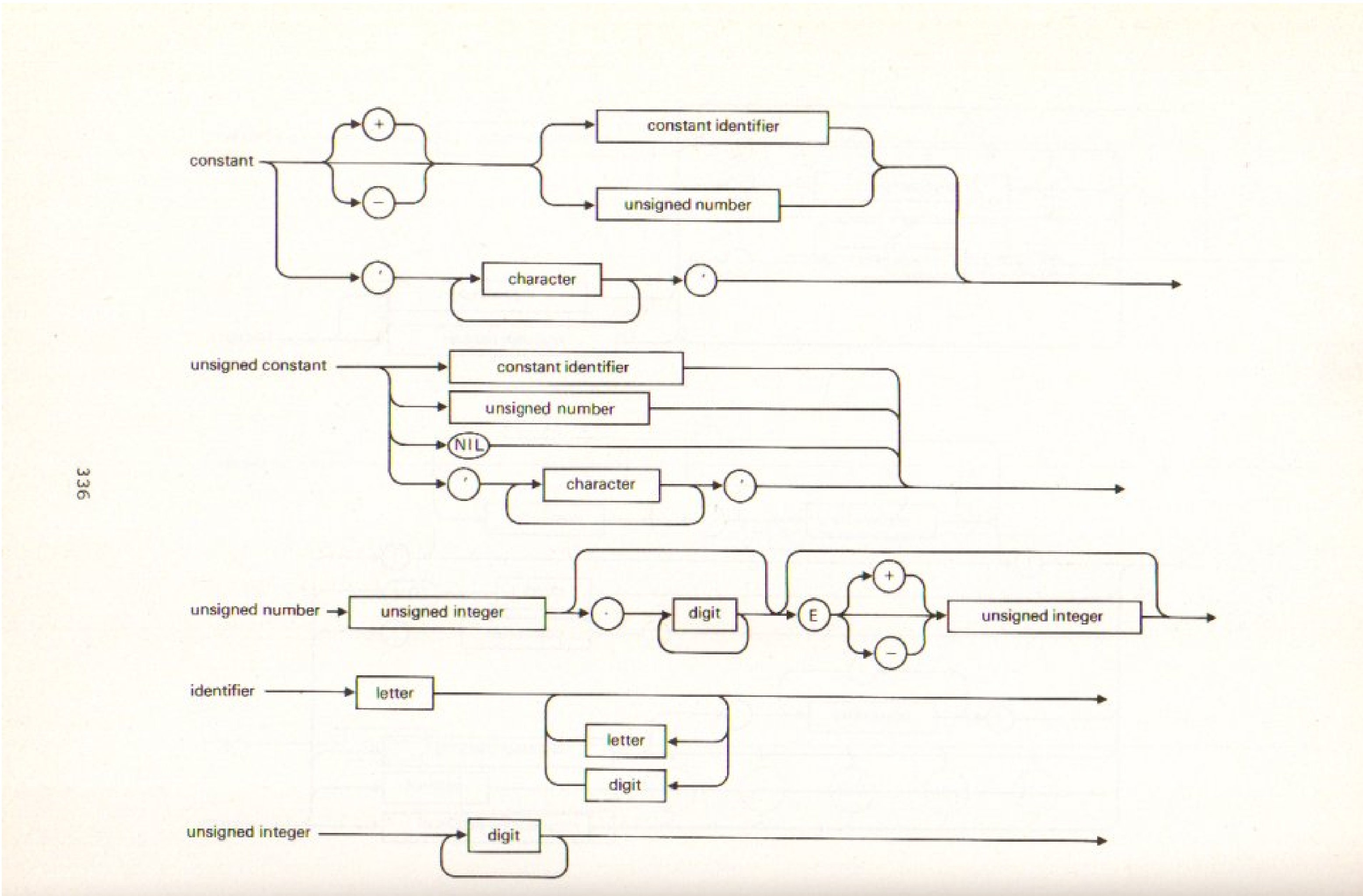
- 예) 명칭의 정의
  <id> ::= <letter> | <id><letter> | <id><digit>
  <letter> ::= a | b | c | … | y | z
  <digit> ::= 0 | 1 | 2 | … | 8 | 9

- Comma로 구분되는 명칭의 정의
  ① BNF
  <id_list> ::= <id_list>, <id> | <id>

  ② EBNF --- 반복되는 부분을 {}로 표현
  <id_list> ::= <id> {, <id> }

- <참고> CFG는 syntax diagram(구문 도표)으로 표현하기도 함

# Pascal Syntax Diagram
## Source: Perter Grogono's "Programming in Pascal"



# Strong LL(k) 문법과 LL(k) 문법

- $FIRST_k(\alpha) = \{ \omega \mid \alpha \overset{*}{\Rightarrow} \omega\beta, |\omega|=k, \text{ or } \alpha \overset{*}{\Rightarrow} \omega \text{ and } |\omega|<k \}$

- Strong LL(k) 문법

  고정된 k에 대하여 2개의 좌단유도 과정이 존재할 때마다 다음을 만족한다.
  ( 생성규칙 $A \rightarrow \alpha \mid \beta$ 에 대하여 )

  $S \overset{*}{\Rightarrow} uA\gamma \Rightarrow u\alpha\gamma \overset{*}{\Rightarrow} ux \in V_T^*$ 와
  $S \overset{*}{\Rightarrow} \underline{vA\gamma \Rightarrow v\beta\gamma \overset{*}{\Rightarrow} vx} \in V_T^*$ 가 존재하여

  *$FIRST_k(x) = FIRST_k(y)$이면 이다.*

Strong LL(k) 조건: 임의의 $A \rightarrow \alpha \mid \beta$ 에 대해
$k\text{-LOOKAHEAD}(A \rightarrow \alpha) \cap k\text{-LOOKAHEAD}(A \rightarrow \beta) = \phi$

- LL(k) 문법

  고정된 k에 대하여 2개의 좌단유도 과정이 존재할 때마다 다음을 만족한다.
  ( 생성규칙 $A \rightarrow \alpha \mid \beta$ 에 대하여 )

  $S \overset{*}{\Rightarrow} uA\gamma \Rightarrow u\alpha\gamma \overset{*}{\Rightarrow} ux \in V_T^*$ 와
  $S \overset{*}{\Rightarrow} \underline{uA\gamma \Rightarrow u\beta\gamma \overset{*}{\Rightarrow} ux} \in V_T^*$ 가 존재하여

  *$FIRST_k(x) = FIRST_k(y)$이면 이다.*

Strong LL(k) 문법과 LL(k) 문법의 차이점
LL(k)는 *A* 앞에 생성된 terminal들이 같아야 하지만
Strong LL(k)는 그렇지 않다.

- 다음 문법은 strong LL(k) 문법인가? strong LL(2)이다.

$S \rightarrow bRS \mid RcSa \mid \varepsilon$
$R \rightarrow acR \mid b$

(1) 1-LOOKAHEAD($S \rightarrow bRS$) = { b }
1-LOOKAHEAD($S \rightarrow RcSa$) = { a, b }

1-LOOKAHEAD($S \rightarrow bRS$) ∩ 1-LOOKAHEAD($S \rightarrow RcSa$) = { b }
∴ **strong LL(1) 문법이 아니다.**

(2) 2-LOOKAHEAD($S \rightarrow bRS$) = { ba, bb }
2-LOOKAHEAD($S \rightarrow RcSa$) = { ac, bc }
2-LOOKAHEAD($S \rightarrow \varepsilon$ ) = { \$, a, aa }
2-LOOKAHEAD($R \rightarrow acR$) = { ac }
2-LOOKAHEAD($R \rightarrow b$) = { b, ba, bb, bc }

2-LOOKAHEAD($S \rightarrow bRS$) ∩ 2-LOOKAHEAD($S \rightarrow RcSa$) ∩ 2-LOOKAHEAD($S \rightarrow \varepsilon$) = $\phi$
2-LOOKAHEAD($R \rightarrow acR$) ∩ 2-LOOKAHEAD($R \rightarrow b$) = $\phi$
∴ **strong LL(2) 문법이다.**

- 다음 문법은 strong LL(2)는 아니지만 LL(2)인 경우이다.
  S → aAaa | bAba
  A → b | ε

  (1) 2-LOOKAHEAD(A → b) = { ba, bb }
  2-LOOKAHEAD(A → ε) = { aa, ba }
  2-LOOKAHEAD(A → b) ∩ 2-LOOKAHEAD(A → ε) = { ba }
  ∴ strong LL(2) 문법이 아니다.

  (2) 유도 과정이 S ⇒ aAaa ⇒ abaa인 경우에 A → b 를 적용
  유도 과정이 S ⇒ bAba ⇒ bba인 경우에 A → 를 적용

- strong LL(1) ⇔ LL(1)
  즉, LL(1) 문법과 strong LL(1) 문법은 동등하다.