

4주차 syntax 2

2022년 3월 23일 수요일 오후 12:18

Arithmetic Expression Grammar

예) $2 - (3 + 4)$ 의 유도

$$E \rightarrow E + E \mid E \rightarrow E - E$$

$$E \rightarrow (E)$$

$$E \rightarrow 0 \mid \dots \mid 9$$

$$E \Rightarrow E - E$$

$$\Rightarrow 2 - E$$

$$\Rightarrow 2 - (E)$$

$$\Rightarrow 2 - (E+E)$$

$$\Rightarrow 2 - (3 + E)$$

$$\Rightarrow 2 - (3+E)$$

$$\Rightarrow 2 - (3+4)$$

연습

$$(5 - 3) - (2 - (3 + 4)) + 6$$

$(S \rightarrow) - (2 - (3 + 4)) + 6$ 유도 연습

$$E \Rightarrow E - E \Rightarrow (E) - E \Rightarrow (E-E) - E$$

$$\Rightarrow (S \rightarrow) - E + E$$

$$\Rightarrow (S \rightarrow) - (E) + E$$

$$\hookrightarrow (E-E) \Rightarrow (E-E) \Rightarrow (2 - (3 + 4)) \Rightarrow (2 - (3 + 4))$$

$$(S \rightarrow) - (2 - (3 + 4)) + E \Rightarrow (S \rightarrow) - (2 - (3 + 4)) + 6$$

화면 캡처: 2022-03-23 오후 10:55

Parse Trees

문장 (sentence)의 유도단계를 표현한 트리

유도 관계를 트리로 표현

- root node는 start symbol
- Internal node는 nonterminal symbol
- Leaf node는 terminal symbol
- 생성된 string은 모든 leaf nodes를 연결하여 얻는다 (left -> right)

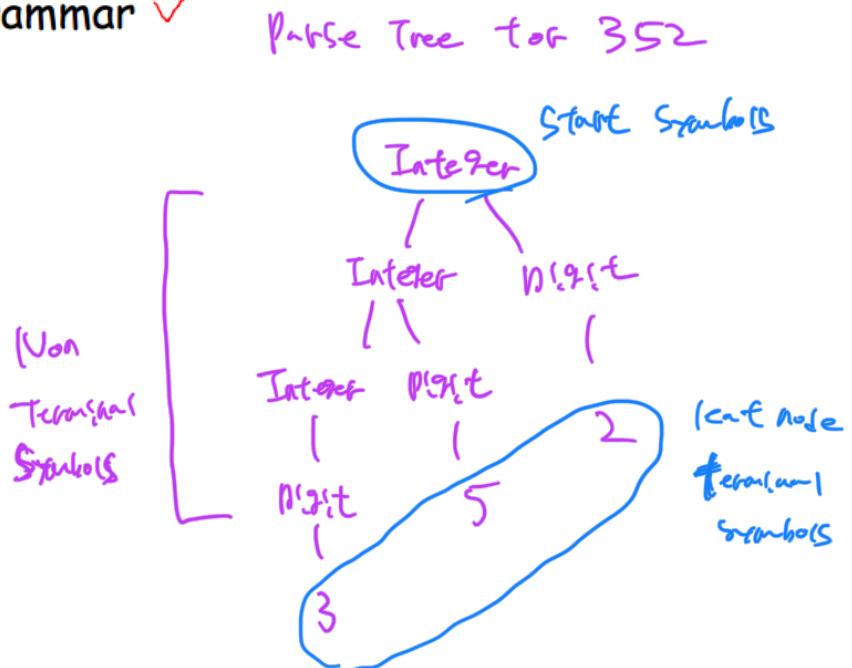
Integer Grammar ✓

예) 352 ✓

$\text{Integer} \rightarrow \text{Integer Digit} \mid \text{Digit}$ ✓
 $\text{Digit} \rightarrow 0 \mid \dots \mid 9$

$\text{Integer} \Rightarrow \text{Integer Digit}$
 $\Rightarrow \underline{\text{Integer Digit Digit}}$
 $\Rightarrow \text{Digit Digit Digit}$
 $\Rightarrow 3 \text{ Digit Digit}$
 $\Rightarrow 3 5 \text{ Digit}$
 $\Rightarrow 3 5 2$

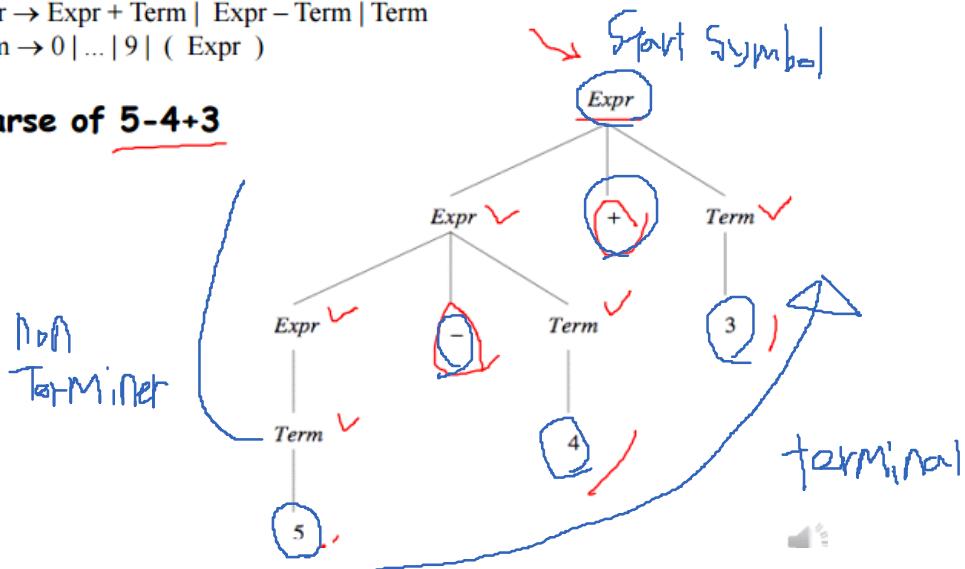
화면 캡처: 2022-03-23 오후 10:57



Arithmetic Expression Grammar

$\text{Expr} \rightarrow \text{Expr} + \text{Term} \mid \text{Expr} - \text{Term} \mid \text{Term}$
 $\text{Term} \rightarrow 0 \mid \dots \mid 9 \mid (\text{ Expr })$

Parse of 5-4+3



화면 캡처: 2022-03-23 오후 11:01

Associativity and Precedence (결합성과 우선순위)

계산 진행 방향 : 좌결합

*/가 +-보다 우선순위 높다

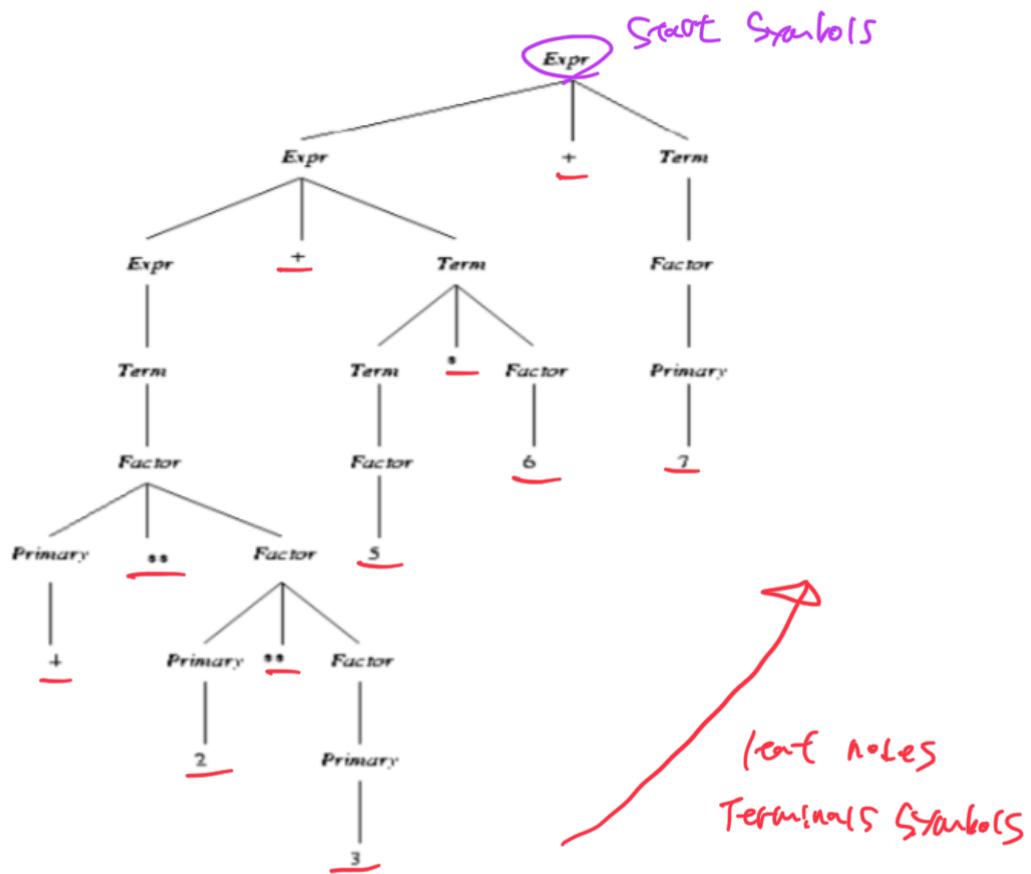
Grammar G_4 :

Expr \rightarrow Expr + Term | Expr - Term | Term
 Term \rightarrow Term * Factor | Term/Factor | Term % Factor | Factor
 Factor \rightarrow Primary ** Factor | Primary
 Primary \rightarrow 0|....|9 | (Expr)

Parse Tree of Grammar G_4

화면 캡처: 2022-03-23 오후 11:08

parse tree for $4^{**}2^{**}3+5\times6+7$



화면 캡처: 2022-03-23 오후 11:08

Operation 우선 순위 있다

**은 우결합하고 좌결합 진행한다

Parse tree는 left most derivation인지 right most derivation인지 알 수 없다

Parse tree에서 아래 쪽에 있을수록 우선 순위가 높아진다

같은 심볼이 있는 방향에 따라 recursion과 associativity 결정된다

Left associativity results from left-recursion

$A \rightarrow AB$ - left recursion

$F \rightarrow P^*F$ - right recursion

Ambiguous Grammar

같은 sentence에 대해 두가지 이상의 parse tree 가지는 문법

동치 표현

- 문장 A가 두개의 다른 parse tree 가진다
- \Leftrightarrow 문장 A가 두개의 다른 leftmost derivations를 가진다
- \Leftrightarrow 문장 A가 두개의 다른 rightmost derivations를 가진다

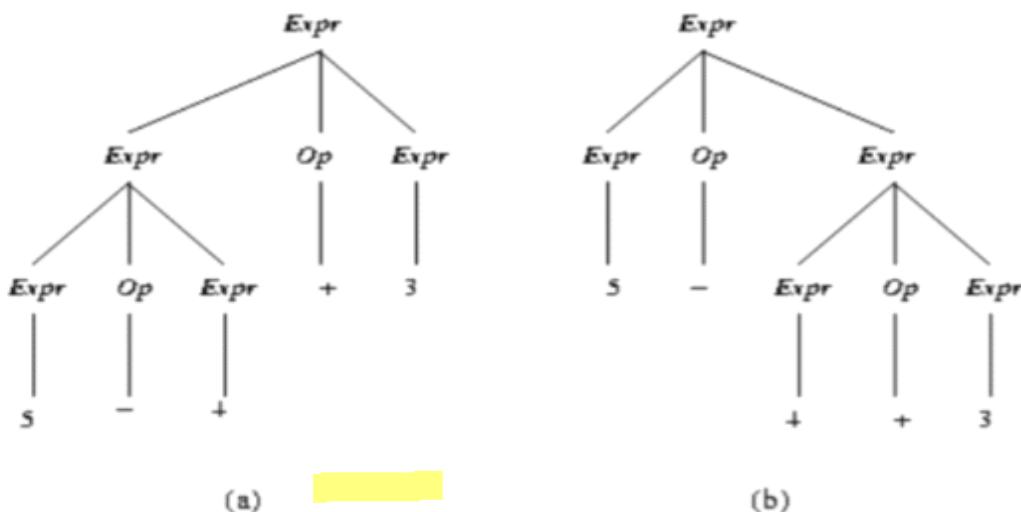


Leftmost derivations와 rightmost derivations 같이 쓰면 안된다

예: G_3 : $\text{Expr} \rightarrow \text{Expr} \text{ Op } \text{Expr} \mid (\text{Expr}) \mid 0 \mid 1 \mid 2 \mid \dots \mid 9$
 $\text{Op} \rightarrow + \mid - \mid * \mid / \mid \% \mid **$

5 - 4 + 3의 파스 트리는?

서로 다른 2개의 Parse Tree가 있다.



5 - 4 + 3의 두 파스 트리

$$a = 4 / b = -2$$

화면 캡처: 2022-03-23 오후 11:18

If and the Dangling Else

조건

Rule 1

Rule 2

IfStatement \rightarrow if (Expression) Statement | if (Expression) Statement else Statement

Statement \rightarrow Assignment | IfStatement | Block

Block \rightarrow {Statement_2}

Statement_2 \rightarrow Statement_2 Statement | Statement

EX)

문장 : If(e1) if (e2) S1 else S2

- If (e_1)을 rule 1으로 보고 나머지를 rule2로 적용하는 방법
 $\rightarrow \text{If}(e_1) \underline{\text{if}}(e_2) \underline{S_1} \underline{\text{else}} S_2$
- If (e_1) else S_2 를 rule2로 적용하고 나머지를 rule1으로 적용하는 방법
 $\rightarrow \underline{\text{If}(e_1) \underline{\text{if}}(e_2) \underline{S_1} \underline{\text{else}} S_2}$

모호성 발생



어떤걸 선택해야하나?

자바에서는 가장 가까운 if문에 걸리게 함 \Rightarrow 즉 else 은 if (e_2)와 짝을 맞는다.

Ambiguity의 해결 방법

- 문법을 처음부터 모호성이 없도록 설계 한다

- 문법이 길고 복잡해질수있다
- EX) G3 : Expr \rightarrow Expr OP Expr | (Expr)
- OP \rightarrow + | - | * | % | **

G4 : Expr \rightarrow Expr + Term | Expr-Term|Term
Term \rightarrow Term * Factor | Term/Factor|Term % Factor | Factor
Factor \rightarrow Primary**Factor | Primary
Primary \rightarrow I | (Expr)

- $G_3 \Leftrightarrow G_4$ 등식지만
 G_3 은 모호한 문법이다

$G_3 \Leftrightarrow G_4$ 등식지만
 G_3 은 모호한 문법이다

- Grammar 외에 추가 규칙을 사용한다

- 항상 직전에 있는 if문에 속한다
- 필요할 경우 {}로 if문의 경계 명확히 함



Extended BNF (EBNF)

BNF : 반복문을 위한 재귀 있음

EBNF : 추가 기호를 넣어 재귀를 없앰

{ } : 0번 이상 반복 될수있어야함 - iteration

() : 문조건 하나를 골라야함 - Choice

[] : 하나를 골라도 되고 안골라도 됨 - Option

EX)

BNF : Expr \rightarrow Expr + Term | Expr-Term | Term

-> Expr은 +/-로 구분되는 하나이상의 Term이다

EBNF : Expr \rightarrow Term $\{ (+|-) \text{ Term} \}$ (+|-) 티타-증정수-Choice

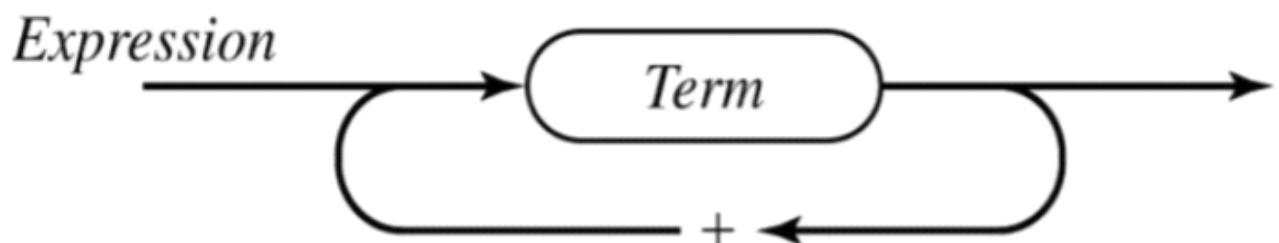
↳ 0번이상 반복

If statement

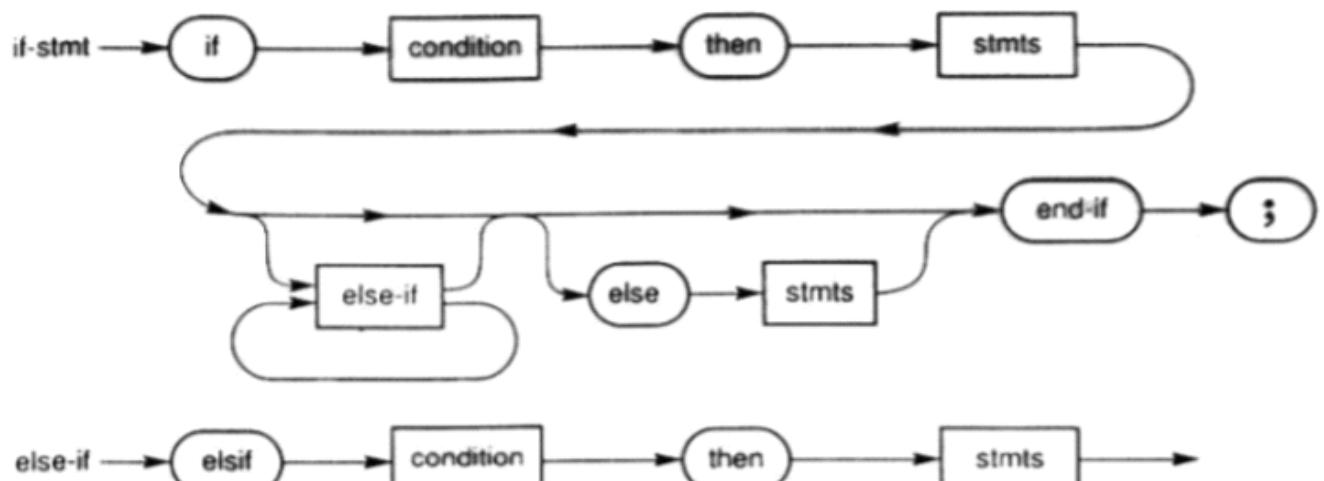
BNF: IfStatement \rightarrow if (Expression) Statement | if (Expression) Statement else Statement

EBNF: if (Expression) Statement [else Statement]

EBNF to BNF : 서로 상호 변환 가능하고 표현력이 같다



Syntax Diagram for Expressions with Addition



Syntax Diagram for if-stmt

화면 캡처: 2022-03-23 오후 11:44

동그라미 : Terminal symbol

사각형 : Non Terminal symbol

An Example program written in myFunction

과제 실습 4를 같이 참고하면서 봐라

BNF와 EBNF 표현을 같이 볼것이다

unary -, !
exponentiation **
multiplication *, /, %
addtion +, -
Relation >, <, >=, <=
equality ==, !=
Conjunction &&

↑ **우선순위**

연산자 우선순위로 올라갈수록 높아진다

My Function in BNF

BNF 표현식

Program -> int main() {Declarations Statement_2}

Declarations -> Declarations Decl | ε

Decl -> Type IdList

IdList -> IdList Id | Id

Statement_2 -> Statement_2 Statement | Statement

Statement -> ; | Assignment | IfStmt | Block

Assignment -> Id = Expr

IfStmt -> if (Expr) Statement | if (Expr) Statement else Statement

Block -> {Statement_2}

Type -> int | float

Expr -> Expr && RelExpr | RelExpr

RelExpr -> AddExpr RelOp AddExpr | AddExpr

AddExpr -> AddExpr + Term | AddExpr-Term | Term

Term -> Term *Factor | Term /Factor | Factor

Factor -> Primary **Factor | Primary | unaryOp Primary

Primary -> Num | Id | (Expr)

RelOp -> < | > | <= | == | !=

unaryOp -> - | !

Id -> Letter LetterDigit

LetterDigit -> LetterDigit Letter | LetterDigit Digit | ε

Letter -> a | b | ... | z

Digit -> 0 | ... | 9

Num -> Integer | Float

Integer -> Integer Digit | Digit

Float -> Integer . Integer

2.3 EBNF for myC

Program -> int main () { Declarations Statements }

Declarations -> { Decl }

Decl -> Type Id { , Id }

Statements -> Statement { Statement }

Statement -> ; | Assignment | IfStmt | Block

Assignment -> Id = Expr;

IfStmt -> if (Expr) Statement [else Statement]

Block -> { Statements }

Type -> int | float

Expr -> RelExpr { && RelExpr }

RelExpr -> AddExpr RelOp AddExpr

AddExpr -> Term { (+ | -) Term }

Term -> Factor { (* | /) Factor }

Factor -> Primary { ** Primary } | UnaryOp Primary

Primary -> Num | Id | (Expr)

RelOp -> < | > | <= | == | !=

UnaryOp -> - | !

화면 캡처: 2022-03-24 오전 12:00

Lexical Level

```
Id -> Letter { Letter | Digit }
Letter -> a | b | ... | z | A | B | ... | Z
Digit -> 0 | 1| ... | 9
Num -> Integer | Float
Integer -> Digit { Digit }
Float -> Integer . Integer
```

화면 캡처: 2022-03-24 오전 12:00