구문과 의미론 2

Concepts of Programming Languages

2024년 1학기

한양대학교 인공지능대학원
임덕선

목차

- 파스트리(parse tree)
- 문법의 모호성(Ambiquity)
- 모호하지 않은 문법 (Unambiguous Grammar)
- 연산자 우선순위
- 연산자 결합법칙
- 확장 BNF (EBNF : Extended BNF)

파스트리 (parse tree)

❑ 유도의 계층적 표현

– 문장 "A = B*(A+C)"에 대한 유도 (아래 그림 문법을 사용)

```
<assign> → <id> = <expr>
<id> → A | B | C
<expr> → <id> + <expr>
         | <id> * <expr>
         | ( <expr> )
         | <id>
```

– 그것을 그림으로 표현한 것이 파스트리(parse tree)

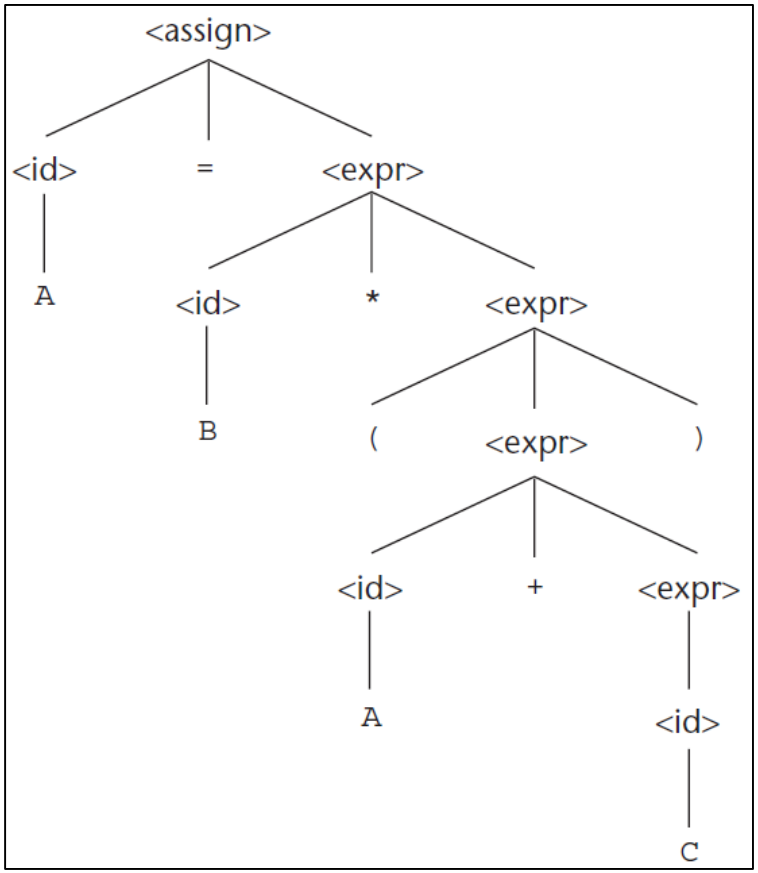
✓ Top-down parsing (leftmost derivation)

✓ Bottom-up parsing (rightmost derivation의 역과정)

파스트리 (parse tree)

❑ 문장 구조에 대한 계층적 표현

- 내부 노드 : 논 터미널
- 잎 노드 : 터미널



파스트리 (parse tree) 예제

❑ 다음 문장에 대한 파스트리를 그릴 것

– Begin A=B+C; B=C end (아래 문법 사용)

```
<program> → begin <stmt_list> end
<stmt_list> → <stmt>
                    | <stmt>; <stmt_list>
<stmt> → <var> = <expression>
<var> → A | B | C
<expression> → <var> + <var>
                    | <var> - <var>
                    | <var>
```

문법의 모호성(Ambiguity)

❑ 정의

- 주어진 문법 G가 2개 이상의 다른 파스 트리를 갖는 문장 형태를 생성하면, G는 모호하다(ambiguous)라고 함

❑ 모호한 문법의 예

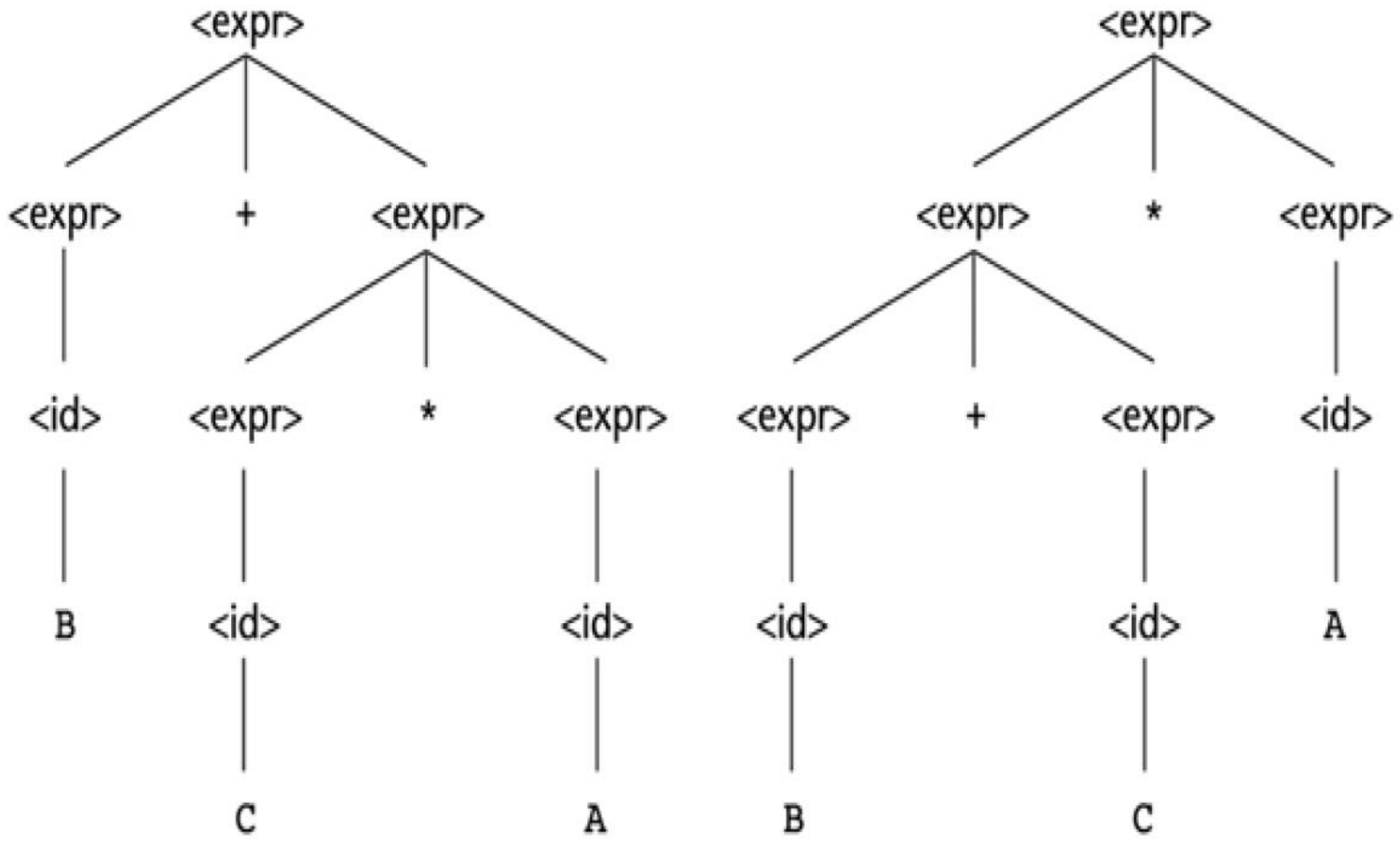
문법 G3

```
<expr> → <expr> + <expr>
         | <expr> * <expr>
         | (<expr>)
         | <id>
<id> → A | B |C
```

- 위의 문법은 모호한가?
  - " B + C * A"의 문장에 대한 parse tree를 구하라

문법의 모호성(Ambiguity)

❑ “ B + C * A”의 문장에 대한 parse tree



모호하지 않은 문법(Unambiguous Grammar)

문법 G3

```
<expr> → <expr> + <expr>
       | <expr> * <expr>
       | (<expr>)
       | <id>
<id> → A | B |C
```

를 종종 E → E + E | E * E | (E) | id
로 표현한다.

- 위의 문법을 Unambiguous Grammar로 고쳐라?

모호하지 않은 문법(Unambiguous Grammar)

문법 G3'

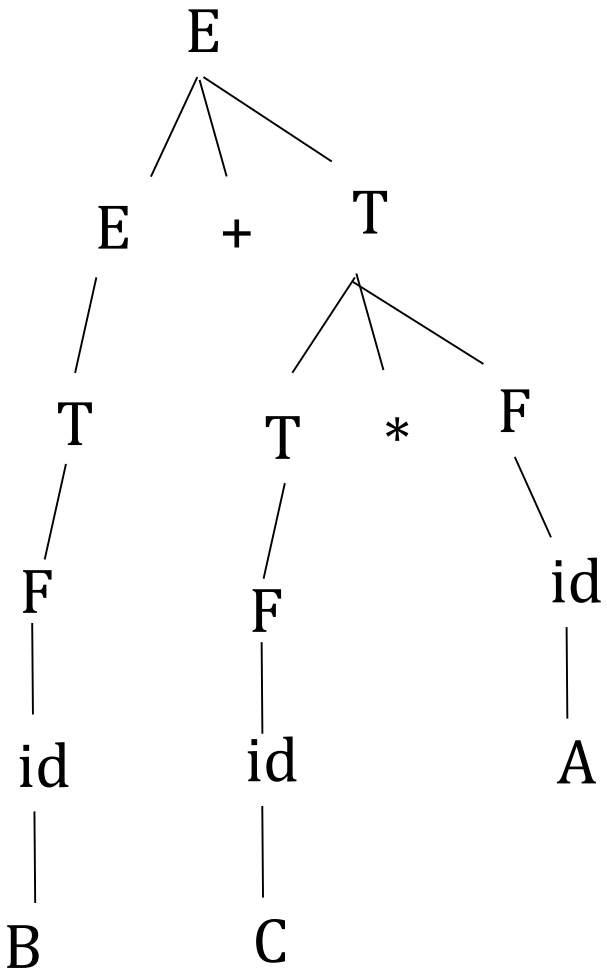
```
E → E + T | T
T → T * F | F
F → ( E ) | id
```

- "B + C * A"의 문장에 대한 parse tree를 구하라

모호하지 않은 문법(Unambiguous Grammar)

E => E + T
=> T + T
=> F + T
=> id + T
=> B + T
=> B + T*F
=> B + T*F
=> B + F*F
=> B + id*F
=> B + C*F
=> B + C*id
=> B + C*A

유도과정(최좌단)



파스트리

문법의 모호성(Ambiguity)

❑ 모호성을 갖는 다른 예제 : if문

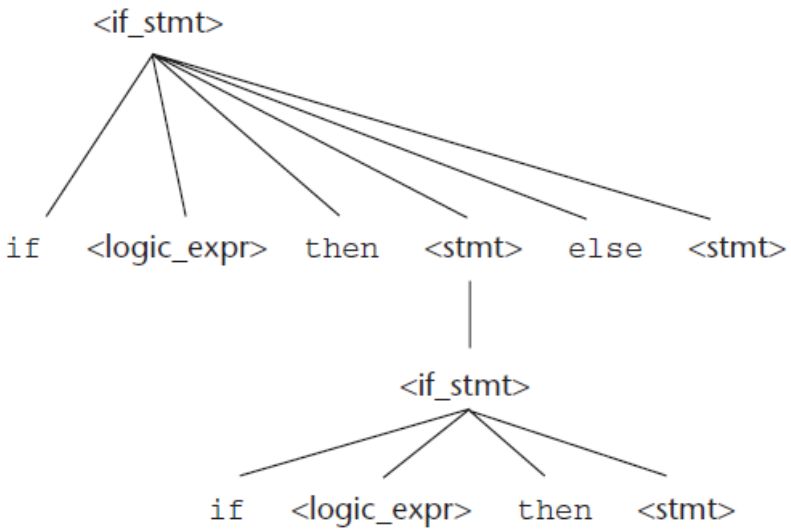
- 문법

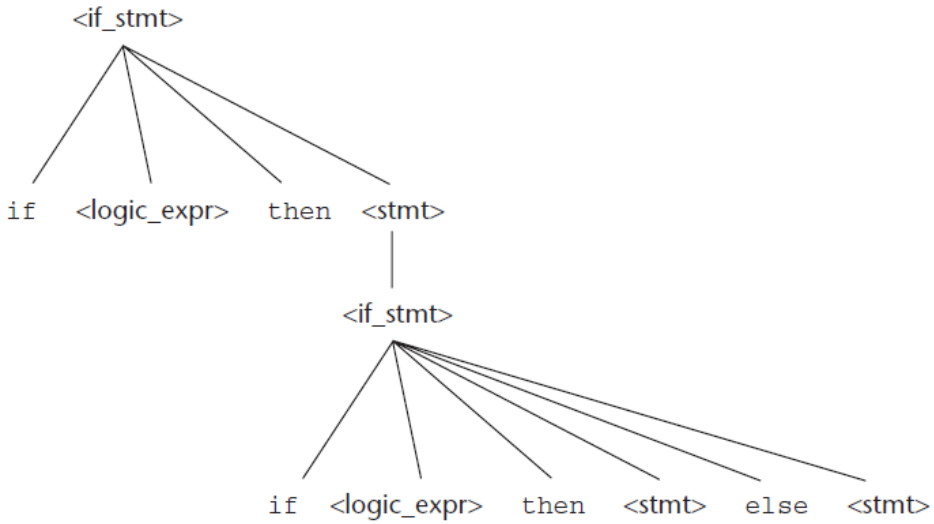
```
<if_stmt> → if <logic_expr> then <stmt>
          | if <logic_expr> then <stmt> else <stmt>
```

- 여기에 <stmt> -> <if_stmt>를 추가하면
- 다음 문장의 파스트리는?

if <logic_expr> **then** **if** <logic_expr> **then** <stmt> **else** <stmt>

문법의 모호성(Ambiguity)





모호성(Ambiguity)을 없앤 if문

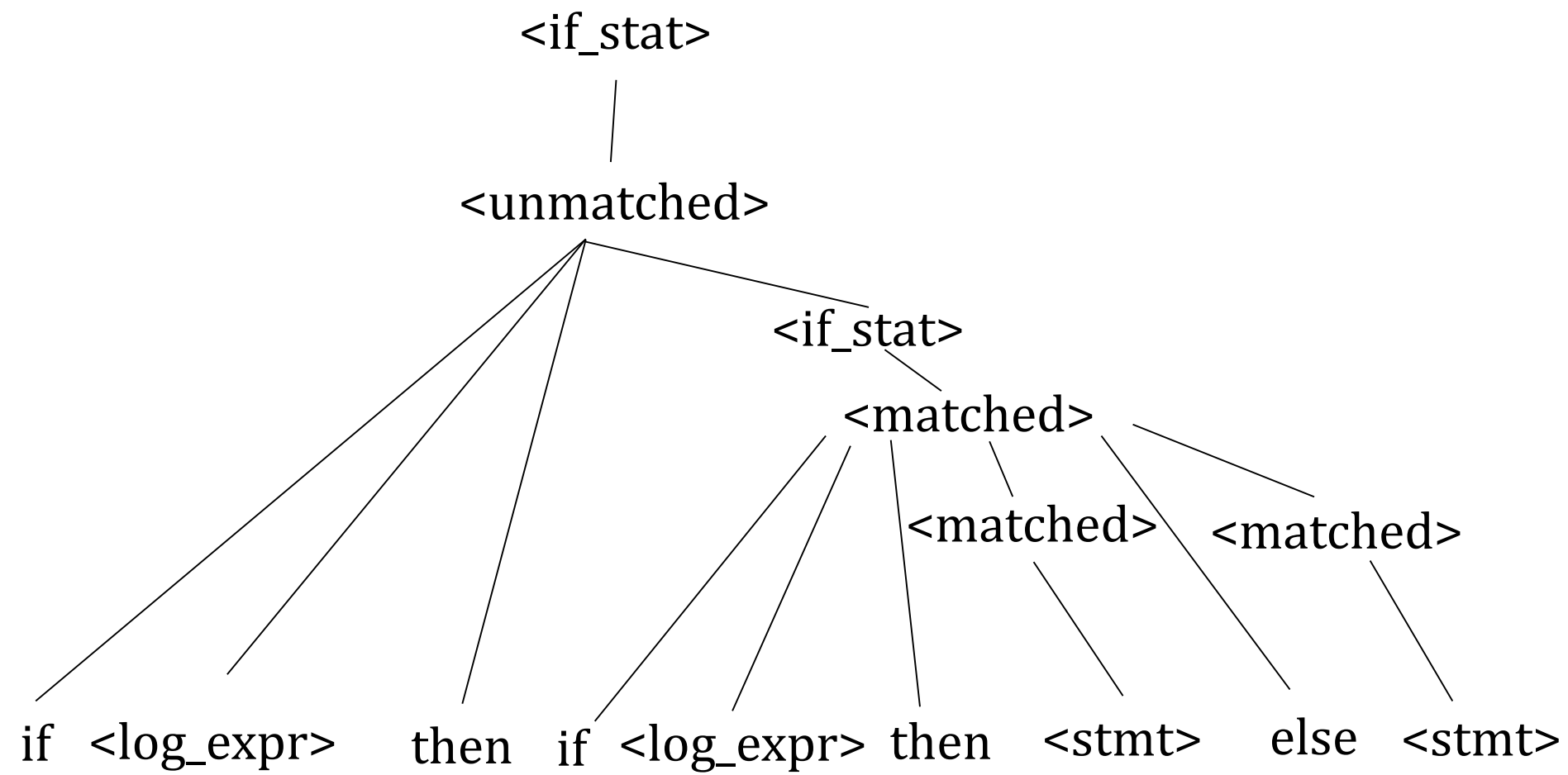
❑ 모호성을 없앤 문법

```
<if_stmt> → <matched> | <unmatched>

<matched > → if <logic_expr> then <matched> else <matched>
      | any non-if statement

<unmatched > → if <logic_expr> then <if_stmt>
      | if <logic_expr> then <matched> else <ummatched>
```

❑ 위 문법에 의한 파스트리는 유일함

if <logic_expr> **then** **if** <logic_expr> **then** <stmt> **else** <stmt>

모호성(Ambiguity)을 없앤 if문



연산자 우선순위

❑ 문법 상에 연산자 우선순위를 부여

- 파스 트리의 낮은 위치의 연산자가 높은 위치의 연산자보다 높은 우선순위를 갖음
- 이와 같이 파스 트리상에 연산자 우선순위가 표현되게 하는 문법을 작성하여 모호성을 제거

연산자 우선순위

❑ 다음에서 *이 +보다 높은 우선순위가 되게 문법을 수정하시오.

```
<assign> → <id> = <expr>
<id> → A | B |C
<expr> → <expr> + <expr>
        | <expr> * <expr>
        | (<expr>)
        | <id>
```

– Hint : 새로운 논터미널 <fact> 와 <term>을 도입

연산자 우선순위

❑ 모호성을 없앤 문법

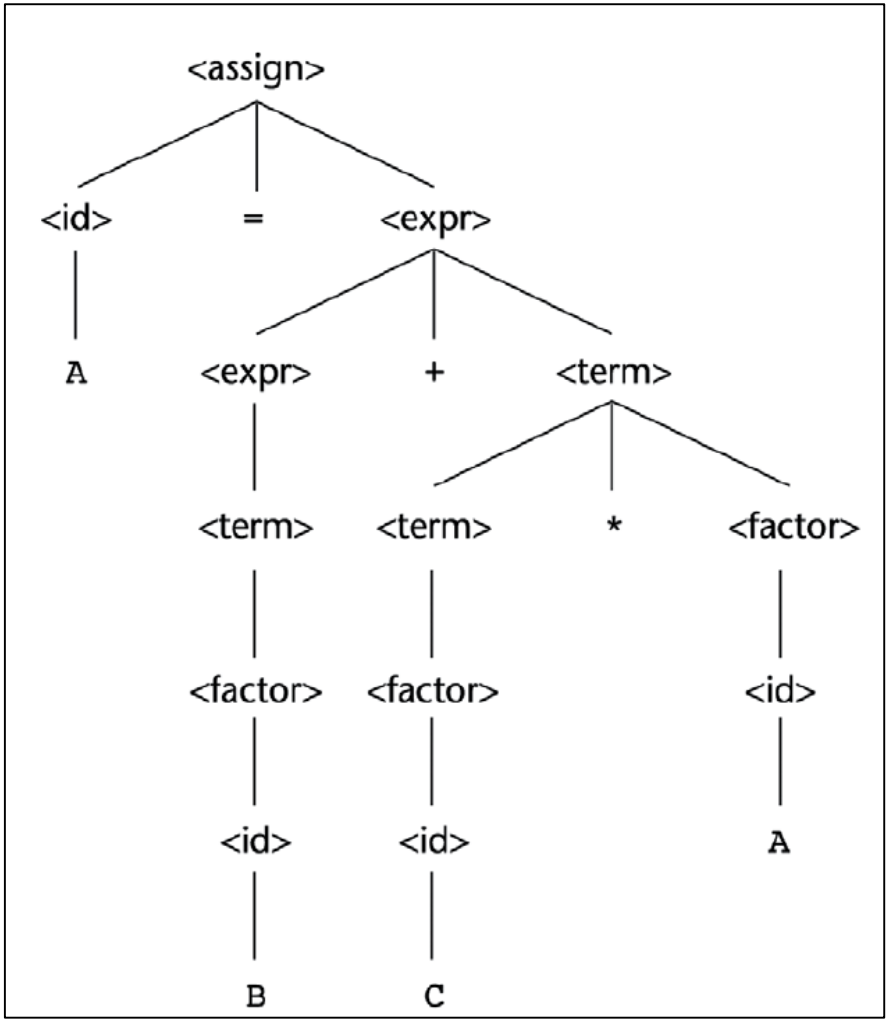
```
<assign> → <id> = <expr>
<id> → A | B |C
<expr> → <expr> + <term> | <term>
<term> → <term> * <fact> | <fact>
<fact> → (<expr>)| <id>
```

혹은

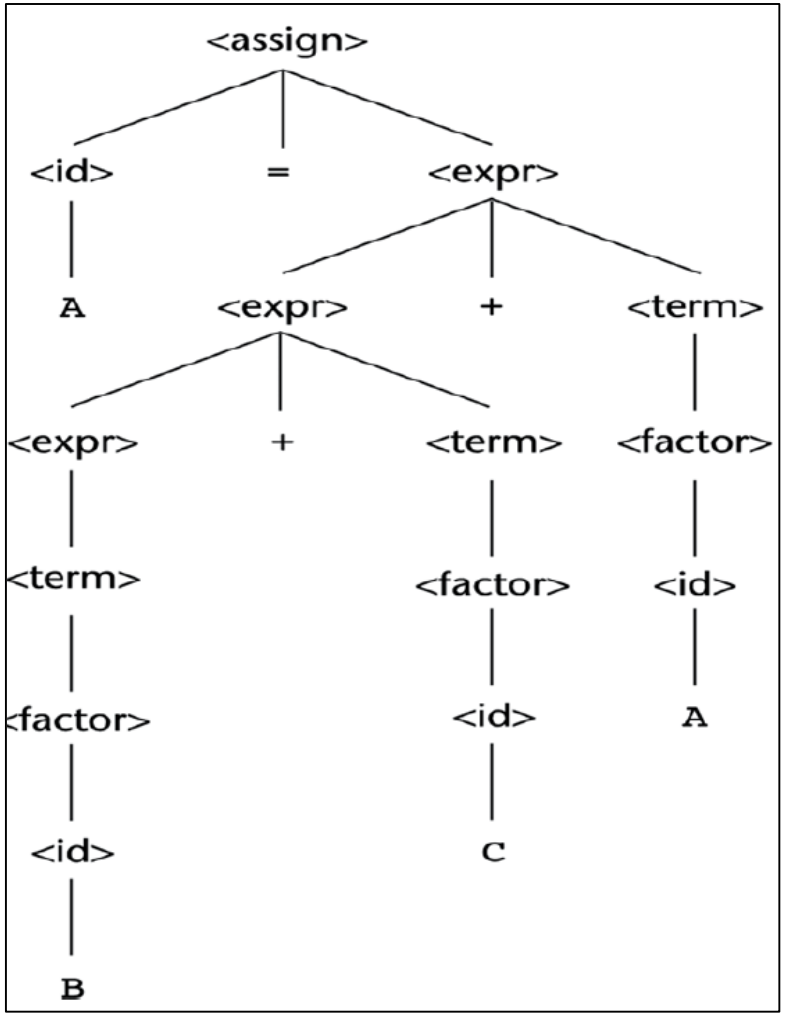
```
E → E + T | T
T → T * F | F
F → ( E ) | id
```

– “A = B + C * A”, “A = B + C + A”에 대한 파스트리는?

연산자 우선순위



A = B + C * A



A = B + C + A

연산자 결합규칙(Association rule)

❑ 연산자 결합규칙도 문법상에 표현할 수 있음

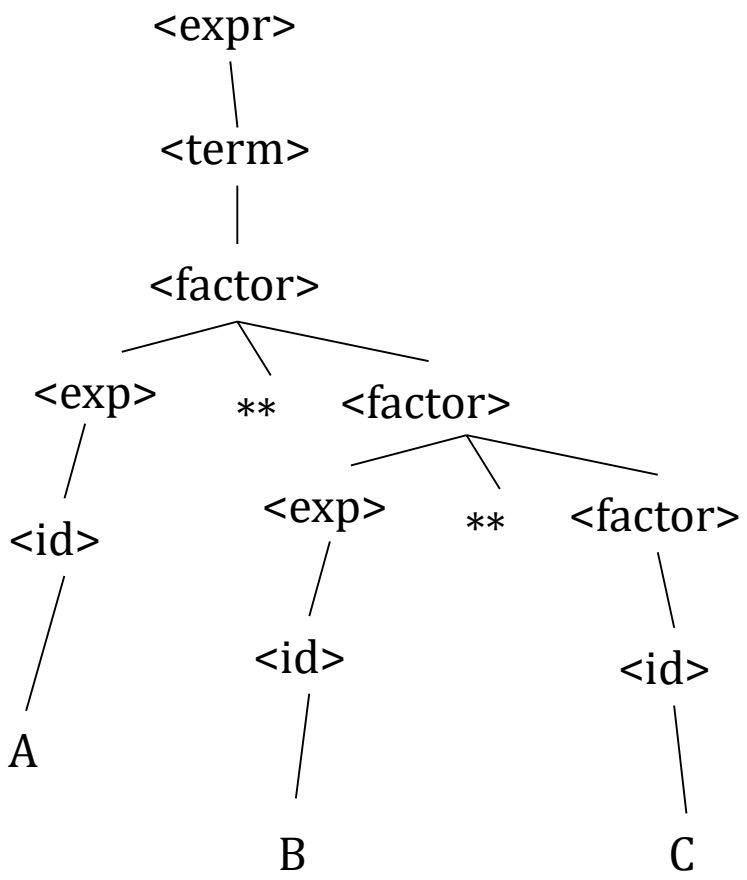
❑ 예

– 다음 문법에서 각 연산자의 결합규칙은?

```
<assign> → <id> = <expr>
<id> → A | B |C
<expr> → <expr> + <term> | <term>
<term> → <term> *<factor>| <factor>
<factor> → <exp> ** <factor> | <exp>
<exp> → (<expr>)| <id>
```

– “A ** B ** C”에 대한 파스 트리는?

연산자 결합규칙(Association rule)



A ** B ** C

연산자 결합규칙(Association rule)

- 규칙에서 **LHS**가 **RHS**의 처음에 오면, 그 규칙은 좌순환적(**left recursive**)이라고 함

- 규칙에서 **LHS**가 **RHS**의 맨 끝에 오면, 그 규칙은 우순환적(**right recursive**)이라고 함

- 좌 순환 규칙은 좌결합 법칙을 명세하고, 우 순환 규칙은 우결합 법칙을 명세함

- 예)
  - 다음 문법에서 각 연산자의 결합 규칙은 무엇인가?

```
E → E + T | T
T → T * F | F
F → P ** F | P
P → ( E )| id
```

확장 BNF (EBNF : Extended BNF)

❑ BNF의 표현력을 향상시키지는 않으나 판독성/작성력을 향상시킴

❑ 확장 사항

1. []: 선택 사항 표현
   ex. <selection> → if (<expr>) <stmt> [else <stmt>]
2. {}: 반복 사항 표현(0 or more)
   ex. <ident_list> → <ident> {, <ident>}
3. (): 다중 선택 표현
   ex. <term> → <term> (* | /) factor
4. + : 한번 이상 반복 표현
   ex. <compound> → begin {<stmt>}$^{+}$ end

확장 BNF (EBNF : Extended BNF)의 예

❑ 다음 BNF를 EBNF로 표현하라

BNF:

```
<expr> → <expr> + <term> | <expr> - <term> | <term>
<term> → <term> * <factor> | <term> / <factor> | <factor>
<factor> → <exp> ** <factor> | <exp>
<exp> → (<expr>)| <id>
```

EBNF:

```
<expr> → <term> {(+ | -)<term>}
<term> → <factor> {(* | /)<factor>}
<factor> → <exp> {**<exp>}
<exp> → (<expr>)| <id>
```

BNF, EBNF의 최근 변경 사항

구분	최근 변경사항
→	:
선택사항 표현 []	아랫첨자 opt 사용 Ex. Proc → name(parameterList$_{opt}$)
다중선택표현 ()	"one of" 사용 Ex. Operator → one of + * - / < >
수직바 '|'	'|' 를 생략한 채, 각 RHS를 단순히 별도의 줄에 표현

Example: in C

statement:
for(expr-1$_{opt}$; expr-2$_{opt}$; expr-3$_{opt}$) statement
if (expression) statement
while (expression) statement
…

Q and A