계산이론

2022년 1학기

이은주

5장 문맥자유 언어
문맥-자유 문법의 예
좌측 우선 유도와 우측 우선 유도
유도 트리
문장 형태와 유도 트리와의 관계
파싱과 소속성
파싱(Parsing)과 모호성(Ambiguity)

Chomsky Hierarchy(촘스키의 분류)

Type	문법(Grammar)	오토마타(Automata)
0	무제한 문법	튜링 기계
1	문맥 의존(연관) 문법(CSG)	선형한계 오토마타
2	**문맥 자유(무관) 문법(CFG)**	**푸시다운 오토마타**
3	**정규 문법** (우선형문법(RLG), 좌선형문법(LLG))	**유한 오토마타**

- Li : type-i 문법이 만들어 내는 언어

$$L0 \supseteq L1 \supseteq L2 \supseteq L3$$

문맥 자유 언어

- 파싱(parsing) : 문장을 문법적인 유도를 통하여 설명
  - 문장의 구조를 표현하는 한 방법
  - 예) 문장의 의미를 이해하는 것을 필요로 할 때 파싱 중요한 역할
  - 번역기(interpreter), 컴파일러(compiler), 다른 프로그램 번역 등

문맥 자유 문법

- 많은 유용한 언어는 정규가 아님
- 문맥 자유 문법 : 프로그래밍 언어의 정의 및 처리
  - 생성 규칙의 오른쪽은 제한이 없는 반면 왼쪽은 단일 변수만 가능
- 문맥 자유 언어 : 문맥 자유 문법으로 생성 된 언어

문맥 자유(무관) 문법(Context-Free Grammar, CFG)

[정의 5.1] 문맥자유 문법 $G = (V, T, S, P)$

V : **논터미널(nonterminal)** 기호의 집합

T: **터미널(terminal)** 기호의 집합

S : **시작 기호(start symbol)** $\in V$

P : $\boldsymbol{A \rightarrow x}$ 형태의 **생성 규칙(production)**의 집합

단, $A \in V, \quad x \in (V \cup T)^*$

- CFG에서의 통상적인 표기
  - A, B, C, D, S 등과 같은 영문자 대문자 : 논터미널 기호
  - 특별한 언급이 없으면 S는 시작 기호를 나타냄
  - a, b, c, d 등과 같은 영문자 소문자 : 터미널 기호

문맥 자유(무관) 문법(Context-Free Grammar, CFG)

[ex] 수식을 생성하는 문법

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow (E)$

$E \rightarrow id$

⇨ $E \rightarrow E + E \mid E * E \mid (E) \mid id$

- 시작 기호 : E
- 논터미널 기호 : E
- 터미널 기호 : + * ( ) *id*

문맥 자유 문법의 예

$G_0 : S \rightarrow aS \mid \lambda$

$L(G_0) = \{ a^n \mid n \geq 0 \}, a^*$

$G_1 : S \rightarrow abS \mid \lambda$

$L(G_1) = \{ (ab)^n \mid n \geq 0 \}, (ab)^*$

$G_2 : S \rightarrow aSb \mid \lambda$

$L(G_2) = \{ a^n b^n \mid n \geq 0 \}$

$G_3 : S \rightarrow AB$

$A \rightarrow 0A \mid \lambda$

$B \rightarrow 1B \mid \lambda$

$L(G_3) = \{ 0^n 1^m, n \geq 0, m \geq 0 \}, 0^* 1^*$

문맥 자유 문법의 예

예제 5.1 (145 page) 다음의 생성 규칙을 갖는 문법
$G = (\{S\}, \{a, b\}, S, P)$는 문맥-자유 문법

$$S \longrightarrow aSa \mid bSb \mid \lambda$$

유도의 예 :

$$S \Longrightarrow aSa \Longrightarrow aaSaa \Longrightarrow aabSbaa \Longrightarrow aabbaa$$
$$S \Longrightarrow bSb \Longrightarrow baSab \Longrightarrow baab$$

문법 G에 의해 정의되는 언어

$$L(G) = \{ wwR: w \in \{ a, b \}^* \}$$

(in other words, even-length palindromes in $\{ a, b \}^*$)

문맥 자유(무관) 문법(Context-Free Grammar, CFG)

예제 5.2 (146 page)

$$G = (\{S, A, B\}, \{a, b\}, P, S)$$

$$P: S \rightarrow abB$$

$$A \rightarrow aaBb$$

$$B \rightarrow bbAa$$

$$A \rightarrow \lambda$$

$$S \Longrightarrow abB \Longrightarrow abbbAa \Longrightarrow abbba$$

$$S \Longrightarrow abB \Longrightarrow abbbAa \Longrightarrow abbbaaBba \Longrightarrow abbbaabbAaba \Longrightarrow abbbaabbaba$$

$$L(G) = \{ ab(bbaa)^n bba(ba)^n \mid n >= 0 \}$$

문맥 자유(무관) 문법(Context-Free Grammar, CFG)

예제 5.3 (146 page)

$$L = \{ a^n b^m \mid n \neq m \}$$

1) $\{ a^n b^m \mid n > m \}$ 인 경우

$S \rightarrow AP$

$P \rightarrow aPb \mid \lambda$

$A \rightarrow aA \mid a$

2) $\{ a^n b^m \mid n < m \}$ 인 경우

$S \rightarrow PB$

$P \rightarrow aPb \mid \lambda$

$B \rightarrow bB \mid b$

1) 과 2)를 합치면 $S \rightarrow AP \mid PB$

$P \rightarrow aPb \mid \lambda$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

문맥 자유(무관) 문법(Context-Free Grammar, CFG)

예제 5.4 (147 page)

$$G : S \rightarrow aSb \mid SS \mid \lambda$$

- Sample derivations:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

$$S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$$

$L(G) = \{ ab, aabb, abaabb, \cdots \}$

$= \{ w \in \{a,b\}^* \mid n_a(w) = n_b(w)$이고 w의 어떤 접두사 v에 대해 $n_a(v) \geq n_b(v) \}$

- a, b를 각각 $(\ ,\ )$로 바꾸면 괄호 사용에 대한 규칙이 된다.

$$G' : S \rightarrow (S) \mid SS \mid \lambda$$

$$L(G') = \{ (\ ), (\ (\ )\ (\ )\ ), (\ (\ )\ )\ (\ ), \cdots \}$$

문맥 자유(무관) 문법(Context-Free Grammar, CFG)

$$S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \lambda$$

$L(G) = \{\lambda, 010, 010010, \cdots\}$

= { 회문(palindrome) of 0 & 1}

(영문 회문의 예) eye, Madam, I'm Adam 등

(국문 회문의 예) 자기 자, 아 좋다 좋아, 다시 합창합시다, 다 큰 도라지일지라도 큰다 등

좌측 우선 유도와 우측 우선 유도

- 문맥-자유 문법에 대한 유도에서
  - 두 개 이상의 변수를 포함하고 있는 문장 : 여러 변수들을 어느 순서로 대체할 것인가 선택
  - 예) 다음의 생성 규칙을 갖는 문법 G = ({A, B, S}, {a, b}, S, P)에서

1. $S \rightarrow AB$
2. $A \rightarrow aaA$
3. $A \rightarrow \lambda$
4. $B \rightarrow Bb$
5. $B \rightarrow \lambda$

$$L(G) = \{a^{2n}b^{m} : n \geq 0, m \geq 0\}$$

  - 문자열 aab에 대한 두 가지 유도

$$S \overset{1}{\Rightarrow} AB \overset{2}{\Rightarrow} aaAB \overset{3}{\Rightarrow} aaB \overset{4}{\Rightarrow} aaBb \overset{5}{\Rightarrow} aab$$

$$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} ABb \overset{2}{\Rightarrow} aaABb \overset{5}{\Rightarrow} aaAb \overset{3}{\Rightarrow} aab$$

좌측 우선 유도와 우측 우선 유도

[정의 5.2] (148 page)

- 좌측우선 유도(leftmost derivation) : 유도 과정의 각 단계에서 각 문장 형태의 가장 좌측 변수가 대체되는 유도
- 우측우선 유도(rightmost derivation) : 가장 우측 변수가 대체되는 유도

좌측 우선 유도와 우측 우선 유도

예제 5.5 (148 page)

$$V = \{S, A, B\}, T = \{a, b\}, and\ productions$$

$$S \rightarrow aAB$$

$$A \rightarrow bBb$$

$$B \rightarrow A \mid \lambda$$

- 문자열 abb 의 유도 :
  - 좌측 우선 유도 : $S \Longrightarrow aAB \Longrightarrow abBbB \Longrightarrow abbB \Longrightarrow abb$
  - 우측 우선 유도 : $S \Longrightarrow aAB \Longrightarrow aA \Longrightarrow abBb \Longrightarrow abb$

유도(derivation)와 언어(language)

- $\boldsymbol{x\alpha y \implies x\beta y}$

  생성규칙 $\alpha \to \beta$ 를 이용하여 $x\,\alpha y$ 로부터 $x\,\beta\,y$ 를 **유도(derive)**한다.

  $\implies$: 직접(1번에) 유도한다.

  $\overset{*}{\Rightarrow}$: 여러 번(0번 이상)에 걸쳐 유도한다.

유도(derivation)와 언어(language)

- **문장형(sentential form)** of $G = (V, T, S, P)$
  시작 기호 S로부터 시작하여 유도 과정에 나타나는 모든 스트링

- **문장(sentence)** : 터미널 기호만으로 구성된 스트링

- **언어(language)** L(G)
  문법 G로부터 만들어진 문장(터미널 스트링)들의 집합

$$\boldsymbol{L(G) = \{ w \mid w \in \sum * \ and \ S \overset{*}{\Rightarrow} w \}}$$

유도(derivation)와 언어(language)

[ex] 재귀문법(Recursive Grammar)

① $CFG\ \ G = (N, T, P, S)$에서 $N = \{S\}, T = \{a, b\}$,

$P = \{S \to aSb, S \to ab\}$일 때 $L(G)$는?

풀이) 첫번째 생성규칙을 $n-1$번 적용하고

두 번째 생성 규칙을 최종적으로 적용

$$S \Longrightarrow aSb \Longrightarrow aaSbb \Longrightarrow a^3Sb^3 \Longrightarrow \cdots \Longrightarrow a^{n-1}Sb^{n-1} \Longrightarrow a^nb^n$$

$$L(G) = \{\, a^nb^n \mid n \geq 1 \,\}$$

② $S \to aSd \mid aAd$

$A \to bAc \mid bc$

$L(G) = \{\, a^mb^nc^nd^m \mid m, n \geq 1 \,\}$

유도(derivation)와 언어(language)

[ex] 재귀문법(Recursive Grammar) - 계속

② $S \rightarrow aSd \mid aAd$

$A \rightarrow bAc \mid bc$

$S \Rightarrow aSd \Rightarrow aaSdd \Rightarrow a^3Sd^3 \Rightarrow \cdots \Rightarrow a^{m-1}Sd^{m-1} \Rightarrow a^{m-1}aAdd^{m-1}$

$\Rightarrow a^mAd^m \Rightarrow a^mbAcd^m \Rightarrow a^mbbAccd^m \Rightarrow a^m\ b^3Ac^3d^m \Rightarrow \cdots$

$\Rightarrow a^m\ b^{n-1}Ac^{n-1}d^m \Rightarrow a^m\ b^{n-1}bcc^{n-1}d^m \Rightarrow a^m\ b^nc^nd^m$

$L(G) = \{ a^mb^nc^nd^m \mid m, n \geq 1 \}$

유도(derivation)와 언어(language)

[ex] 수식을 생성하는 문법

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

$T \rightarrow T * F \mid T / F \mid F$

$F \rightarrow (E) \mid a$

① $(a + a)$

$E \Rightarrow T \Rightarrow F \Rightarrow (E) \Rightarrow (E + T) \Rightarrow (T + T) \Rightarrow (F + T) \Rightarrow (a + T) \Rightarrow (a + F) \Rightarrow (a + a)$

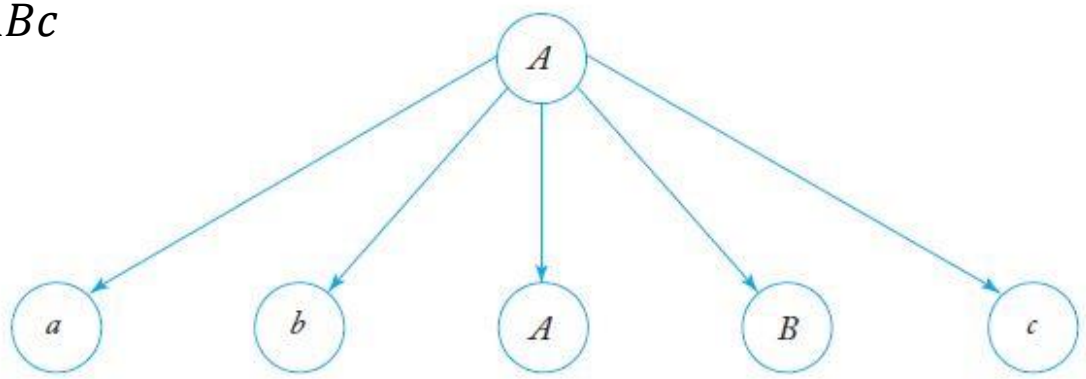
② $a * a - a$

$E \Rightarrow E - T \Rightarrow T - T \Rightarrow T * F - T \Rightarrow F * F - T \Rightarrow a * F - T \Rightarrow a * a - T \Rightarrow a * a - F$
$\Rightarrow a * a - a$

유도 트리 (Derivation Trees), 파스 트리(parse tree)

- 유도 트리(derivation tree) : 유도 과정을 보이는 또 다른 방법
  - 사용된 생성규칙들의 순서와는 무관
  - 유도 트리 : 순서 트리(ordered tree)
    - 부모 노드 : 생성규칙의 좌변에 있는 변수, 라벨이 주어짐
    - 자식 노드 : 대응되는 우변에 있는 심벌들을 표현
  - 예) 다음의 생성 규칙에 대한 유도 트리의 한 부분

$$A \rightarrow abABc$$



유도 트리(Derivation Tree)

[정의 5.3] (149 page) $CFG \quad G = (V, T, S, P)$의 **유도 트리**

(1) 루트노드의 표식(label) : S(시작변수)

(2) 모든 잎(leaf, terminal) 노드 : ($T \cup \{\lambda\}$)의 표식

(3) 모든 중간(nonterminal) 노드 : V 의 표식

(4) 중간 노드 A가 $a_1, a_2, \ldots, an$ 인 자식 노드를 가지면

생성규칙 P는 $A \rightarrow a_1 a_2 \ldots an$ 을 포함

(5) λ의 표식을 가진 노드는 형제 노드가 없음

- **부분 유도트리(partial derivation tree)**
  - 조건 (3), (4), (5) 를 만족하고, (2) 대신에 조건 (2)'를 갖는다.
  - (2)' 모든 잎 노드는 ($V \cup T \cup \{\lambda\}$) 의 표식을 갖는다.

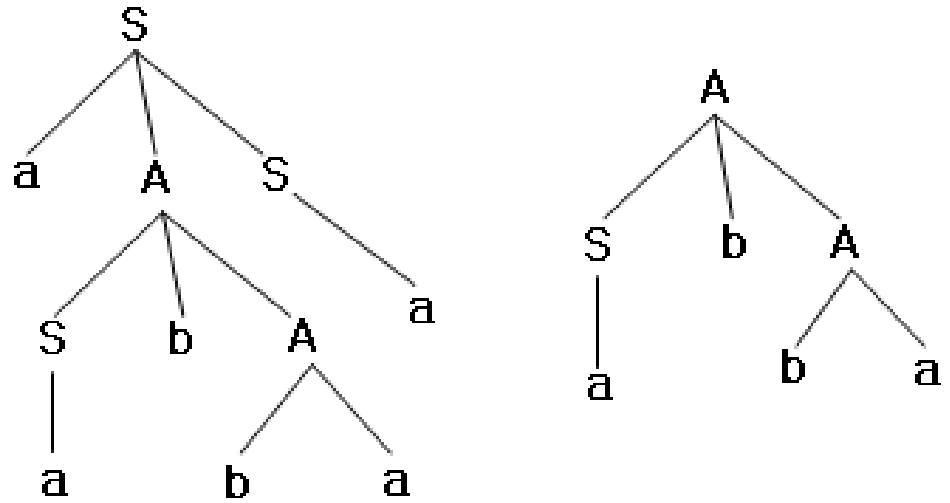
- 트리의 **생성물(yield)**
  - 유도트리의 잎 노드를 왼쪽으로부터 접속하여 만들어진 스트링

유도 트리(Derivation Tree)

[ex]

$$S \rightarrow aAS \mid a$$

$$A \rightarrow SbA \mid SS \mid ba$$



유도트리 *부분유도트리*

- 생성물(yield) : $aabbaa$

유도 트리(Derivation Tree)

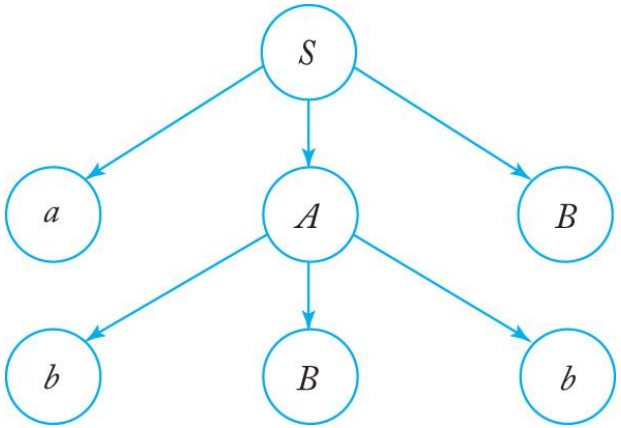
예제 5.6 (150 page)

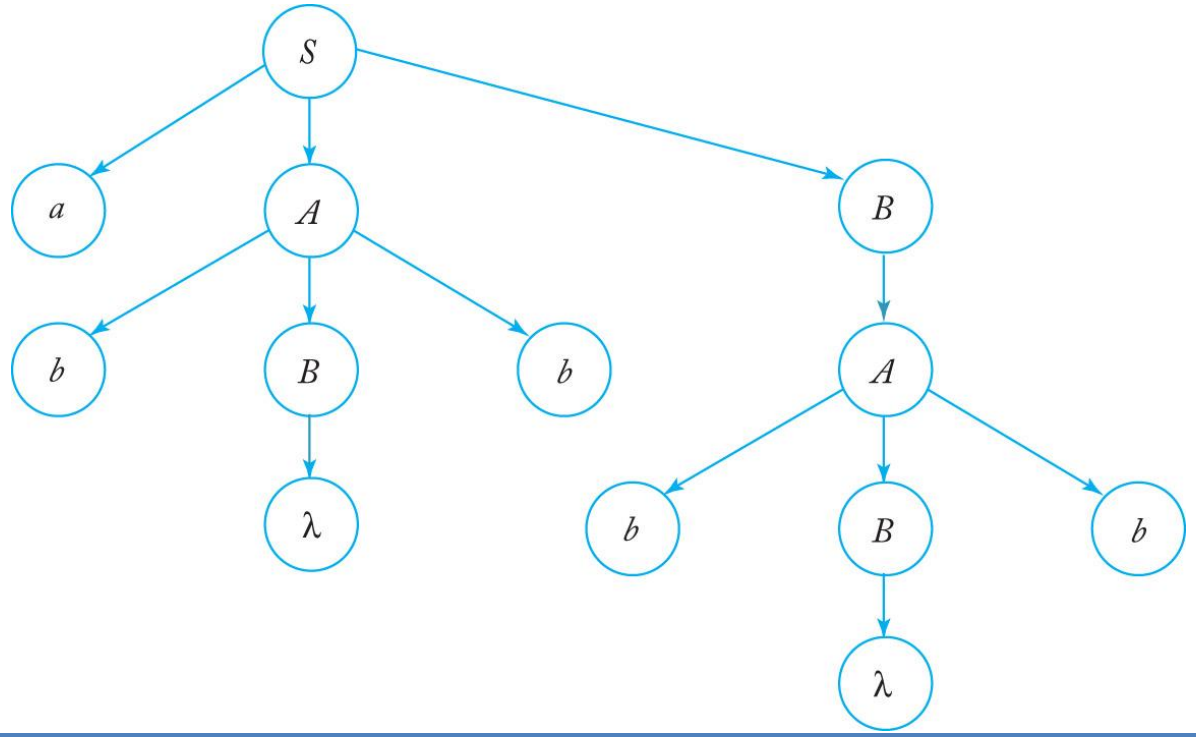
$S \rightarrow aAB$

$A \rightarrow bBb$

$B \rightarrow A \mid \lambda$

- $abBbB$
- $abbbb$





문장 형태와 유도 트리와의 관계

[정리 5.1] (151 page) 문맥-자유 문법 $G = (V, T, S, P)$에서

- 모든 $w \in L(G)$에 대해, 생성물이 w인 G의 유도 트리 존재
- 역으로, 모든 유도 트리의 생성물은 $L(G)$에 속함
- t_G가 루트 노드가 S인 G의 부분 유도 트리이면, t_G의 생성물은 G의 문장 형태

문장 형태와 유도 트리와의 관계

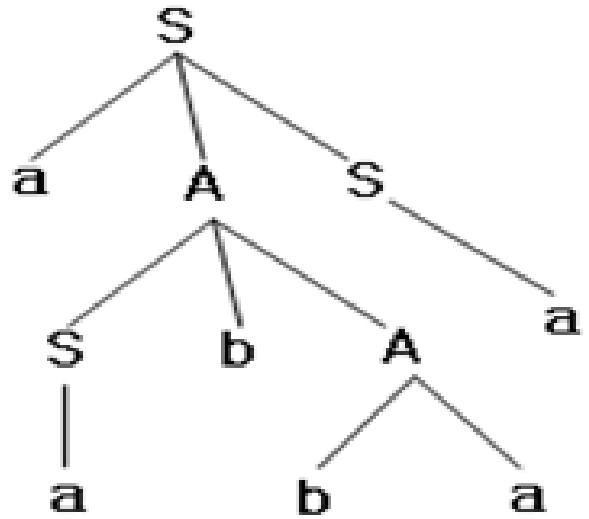
[ex]

$$S \rightarrow aAS \mid a$$

$$A \rightarrow SbA \mid SS \mid ba$$

좌측유도: $S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$

우측유도: $S \Rightarrow aAS \Rightarrow aAa \Rightarrow aSbAa \Rightarrow aSbbaa \Rightarrow aabbaa$



파싱과 모호성

- 문법의 생성적인 측면 : 문법 G가 주어졌을 때 G로부터 유도될 수 있는 문자열들의 집합
  - G가 생성하는 언어
- 실제 응용에서
  - 주어진 단말들의 문자열 w가 $L(G)$에 속하는 지의 여부를 알고자 하는 문법의 분석적인 측면
    - 만약 $w \in L(G)$, w에 대한 유도
- 소속성(membership) 알고리즘 : w가 $L(G)$에 속하는지 혹은 아닌지를 판별하는 알고리즘
- 파싱(parsing)의 의미 : $L(G)$에 속하는 w가 유도되는데 사용된 일련의 생성규칙들을 찾는 것이라 할 수 있음

파싱과 소속성

- 주어진 문자열 $w \in L(G)$에 대한 단순한 파싱 방법
  - 체계적으로 (systematically) 가능한 모든 (예, 좌측우선) 유도들을 구성
    - 그 중에 w와 일치하는 것이 있는지를 알아보는 것
  - 첫 번째 라운드 : 시작 심벌을 좌변으로 갖는 모든 생성규칙들을 살펴보기

$$S \rightarrow x$$

    - 시작 심벌 S로부터 한 단계에 유도될 수 있는 모든 문장 형태 x 찾기
  - 두 번째 라운드 : 만약 어느 x도 w 와 일치하지 않으면, 각 x의 가장 왼쪽 변수에 적용될 수 있는 모든 생성규칙들 적용
    - 새로운 문장형태들의 집합 생성, 일부는 w로 유도될 수 있음

파싱과 소속성

- 계속되는 각 라운드에서, 가장 왼쪽 변수를 취해서 가능한 모든 생성 규칙들 적용
- 생성된 문장 형태들 가운데 w로 유도될 수 없는 문장 형태는 제외
- 매 라운드마다 문장 형태들의 집합이 남게 됨
  - 첫 번째 라운드 종료 후 : 생성 규칙을 한 번 적용하여 유도될 수 있는 문장 형태들의 집합이 남게 됨
  - 두 번 째 라운드 종료 후 : 두 단계에 유도될 수 있는 문장 형태들의 집합이 남게 됨
- w가 $L(G)$에 속하면, 반드시 이에 대한 유한한 길이의 좌측 우선 유도가 존재하게 됨 : w의 좌측 우선 유도를 만들어 냄

파싱과 소속성

- 철저한 탐색 파싱(exhaustive search parsing)
  - w의 유도를 찾기 위해 시작 심벌 S로부터 모든 가능한 유도들을 생성
  - 하향 식 파싱(top-down parsing)의 한 형태

파싱과 소속성

예제 5.7 (156 page)

$w = aabb$의 유도 여부

$$S \rightarrow SS \mid aSb \mid bSa \mid \lambda$$

- 첫 라운드 : 다음의 네 가지 유도 가능

  1. $S \Rightarrow SS$

  2. $S \Rightarrow aSb$

  3. $S \Rightarrow bSa$

  4. $S \Rightarrow \lambda$

  3과 4의 어느 것도 $aabb$로 유도될 수 없기 때문에 고려 대상에서 이들을 제외시킬 수 있음

파싱과 소속성

예제 5.7 (156 page) 계속 $S \rightarrow SS \mid aSb \mid bSa \mid \lambda$

- 두 번째 라운드 : 1의 문장 형태 SS의 가장 좌측 변수 S에 생성 규칙들을 적용

$$S \Rightarrow SS \Rightarrow SSS$$

$$S \Rightarrow SS \Rightarrow aSbS$$

$$S \Rightarrow SS \Rightarrow bSaS$$

$$S \Rightarrow SS \Rightarrow S$$

  - 2의 문장형태로부터 다음의 문장형태들이 산출됨

$$S \Rightarrow aSb \Rightarrow aSSb$$

$$S \Rightarrow aSb \Rightarrow aaSbb$$

$$S \Rightarrow aSb \Rightarrow abSab$$

$$S \Rightarrow aSb \Rightarrow ab$$

  - 새로운 문장 형태들 중 일부를 제외시킬 수 있음

파싱과 소속성

예제 5.7 (156 page) 계속 $S \rightarrow SS \mid aSb \mid bSa \mid \lambda$

- 다음 라운드 : $aabb$의 유도를 찾을 수 있음

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

  - $aabb$는 주어진 문법에 의해 생성되는 언어에 속함

파싱과 소속성

- 철저한 탐색 파싱의 몇 가지 문제점
  - 너무 지루함 : 모든 가능한 유도 과정들을 다 고려하기 때문에 효율적인 파싱을 요구하는 경우에는 사용되기 어려움
  - $L(G)$에 속한 문자열은 항상 파싱을 하지만, $w \notin L(G)$인 경우에는, 종료되지 않을 수도 있음
    - 예제 5.7의 문법 G에서 $w = abb$인 경우, $S \rightarrow SS \mid aSb \mid bSa \mid \lambda$
      - $w \notin L(G)$

- 철저한 탐색 방법에 종료하는 방법을 추가하지 않으면,
  - 철저한 탐색 방법은 w와 일치하는 문장을 찾기 위해 모든 가능한 문장형태들을 계속 무한히 생성하게 됨

파싱과 소속성

- 철저한 탐색 파싱의 비종료 문제 해결
  - 다음 두 개의 생성 규칙 배제

$$A \rightarrow \lambda$$
$$A \rightarrow B$$

파싱과 소속성

예제 5.8 (156 page) 다음의 생성 규칙을 갖는 문법 G

$$S \rightarrow SS \mid aSb \mid bSa \mid ab \mid ba$$

- $A \rightarrow \lambda$와 $A \rightarrow B$ 형태의 생성 규칙이 없음
- 예제 5.7의 문법이 생성하는 언어에서 λ 를 제외한 것과 동일한 언어 생성
- 문자열 $w \in \{a, b\}^+$에 대하여 철저한 탐색 파싱 방법
  - 항상 $|w|$ 단계 이내에 모든 파싱 과정이 종료하게 됨 : 각 유도 단계에서 나타나는 문장 형태가 최소한 한 개의 심벌이 증가되기 때문
  - $|w|$ 단계 후 : 파싱을 만들어내거나 w가 $L(G)$에 속하지 않는다는 판정을 내리게 됨

파싱과 소속성

[정리 5.2] (158 page) 문맥-자유 문법 $G = (V, T, S, P)$

$$A \rightarrow \lambda \text{ 혹은 } A \rightarrow B \ (A, B \in V)$$

- 철저한 탐색 파싱 : 모든 w 에 대해 w ∈ Σ*의 파싱을 산출 or 파싱이 불가능하다고 알려주는 알고리즘이 될 수 있음

증명) w의 길이 혹은 w를 구성하는 단말 심벌들의 개수 : $|w|$를 초과할 수 없음

- 유도 과정의 총 단계 수 : $2|w|$를 초과할 수 없음
- 그 시점에 성공적인 파싱을 얻거나 아니면 w는 주어진 문법에 의해 생성될 수 없음을 알게 됨

파싱과 소속성

- 문장 형태의 총수

$$M = |P| + |P|^2 + \ldots + |P|^{2|w|} = O(|P|^{2|w|+1})$$

  - 좌측 우선 유도로 제한
    - 첫 단계 후 : 최대 $|P|$개의 문장 형태
    - 두 번째 단계 후 : 최대 $|P|^2$개의 문장 형태
    - 계속해서 각 단계 후에 생성될 수 있는 문장 형태들의 최대 개수를 계산할 수 있음

파싱과 소속성

[정리 5.3] (159 page) 모든 문맥-자유 문법에 대하여

임의의 $w \in L(G)$를 $|w|^3$에 비례하는 수의 단계 내에 파싱하는 알고리즘 존재

- 선형 시간(linear time) 파싱 알고리즘 : 문자열의 길이에 비례하는 시간이 걸리는 파싱 방법

파싱과 소속성

[정리 5.4] (159 page)

문맥-자유 문법 $G = (V, T, S, P)$의 모든 생성규칙들이 다음과 같은 형태이면 단순 문법(simple grammar) 혹은 s-문법(s-grammar)

$$A \rightarrow ax$$

$A \in V, a \in T, x \in V^*$이고 임의의 쌍 (A, a)는 P에서 많아야 한번 나타남

파싱과 소속성

예제 5.9 (159 page)

- $S \rightarrow aS \mid bSS \mid c$
  - s –문법
- $S \rightarrow aS \mid bSS \mid aSS \mid c$
  - s –문법이 아님

∵ 쌍 (S, a)가 생성 규칙 $S \rightarrow aS$와 $S \rightarrow aSS$에서 나타나기 때문

파싱과 소속성

- G가 s −문법이면 $L(G)$에 속한 모든 문자열 w가 $|w|$에 비례하는 노력으로 파싱될 수 있음
  - 문자열 $w = a_1 a_2 \ldots a_n$에 대한 철저한 탐색 방법
    - 좌변에 S가 있고 우변이 a_1으로 시작하는 생성 규칙이 많아야 하나만 있을 수 있음

$$S \Rightarrow a_1 A_1 A_2 \ldots A_m$$

  - 그 다음에 변수 A_1을 치환

$$S \overset{*}{\Rightarrow} a_1 a_2 B_1 B_2 \ldots A_2 \ldots A_m$$

각 단계마다 하나의 단말 심벌 생성

- 전체 과정이 $|w|$단계 내에 완료

문법과 언어에서의 모호성

- 모호성(Ambiguity)
  - $w \in L(G)$ 인 한 문자열에 대하여 여러 개의 다른 유도 트리들이 존재할 가능성이 있음

[정리 5.5] (160 page)

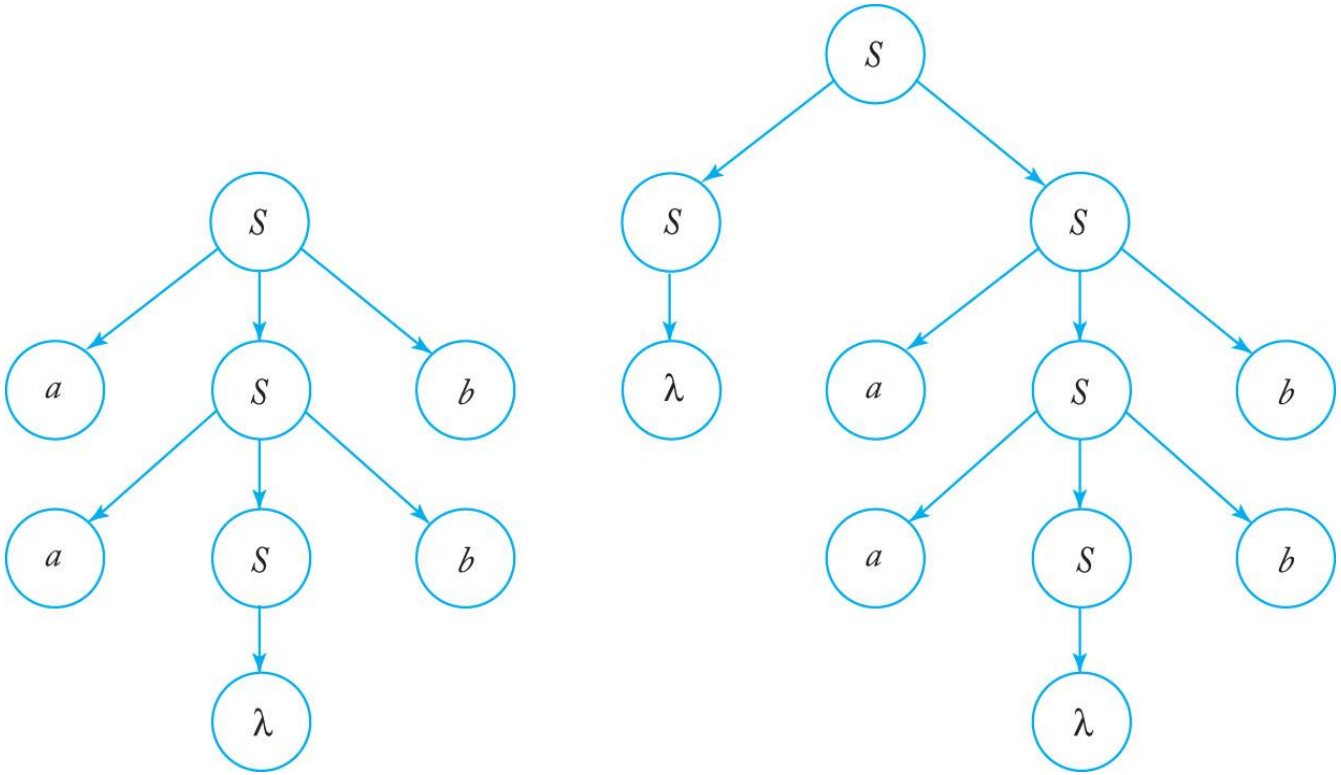
- 문맥-자유 문법 G에서 두 개 이상의 서로 다른 유도 트리를 갖는 문자열 w가 존재하면, 문법 G는 모호(ambiguous)
- 모호성은 어떤 문자열 w에 대해 두 개 이상의 좌측 우선 혹은 우측 우선 유도가 존재하는 것을 의미

문법과 언어에서의 모호성

예제 5.10 (161 page)

생성규칙 $S \rightarrow aSb \mid SS \mid \lambda$를 갖는 예제 5.4의 문법 G는 모호하다.

- 두 개의 유도 트리들을 갖는 문자열 $aabb$가 존재하기 때문



문법과 언어에서의 모호성

- 모호성 : 자연언어의 일반적인 특징
- 프로그래밍 언어 : 각 문장이 정확히 하나의 의미로 해석되어야 하므로 가능한 한 모호성을 제거
- 모호성 제거 : 동치이면서 모호하지 않은(unambiguous) 다른 문법으로 다시 구성함

파싱(Parsing)과 모호성(Ambiguity)

[ex]

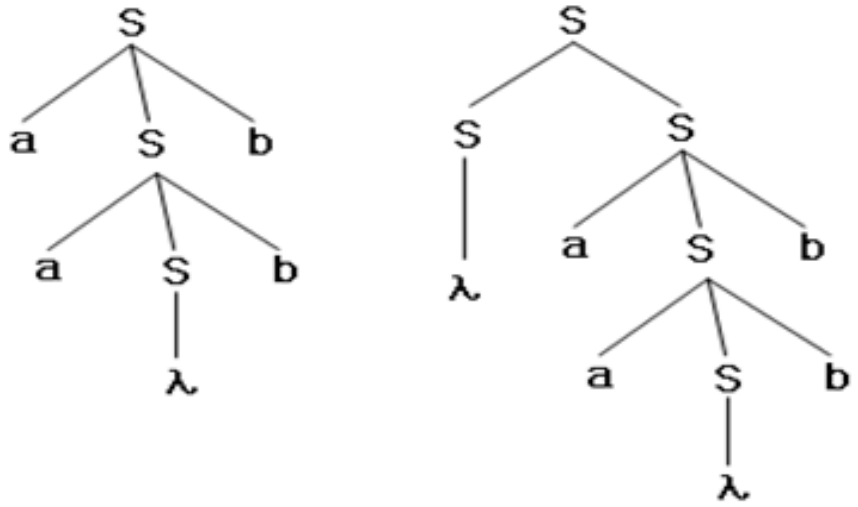
$$S \rightarrow aSb \mid SS \mid \lambda$$

스트링 $aabb$에 대해 서로 다른 2개의 유도트리가 존재한다.

∴ **모호한 문법**이다.

좌측유도 : $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$

좌측유도 : $S \Rightarrow SS \Rightarrow S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$



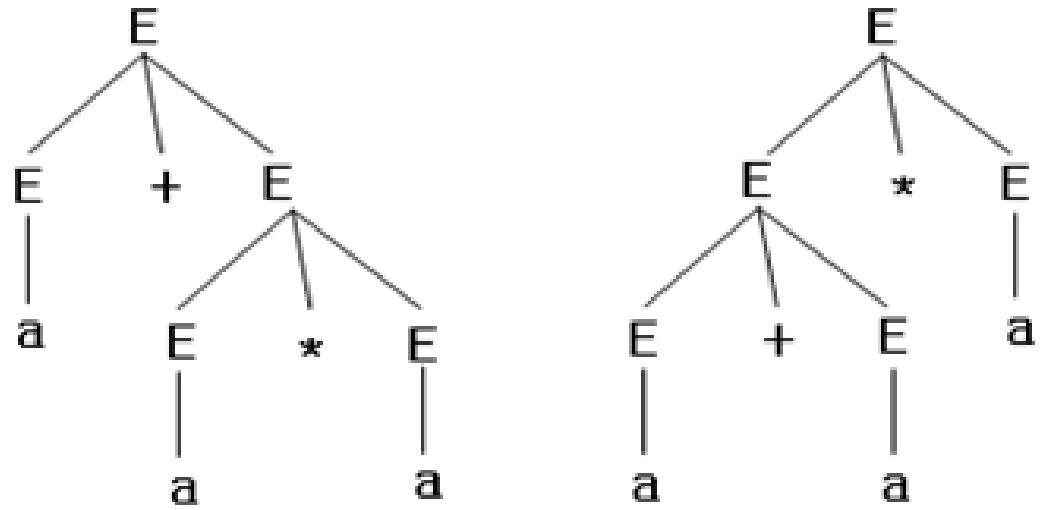
파싱(Parsing)과 모호성(Ambiguity)

[ex]

$$E \rightarrow E + E \mid E * E \mid a$$

스트링 $a + a * a$ 에 대한 유도트리

∴ **모호한 문법**이다.

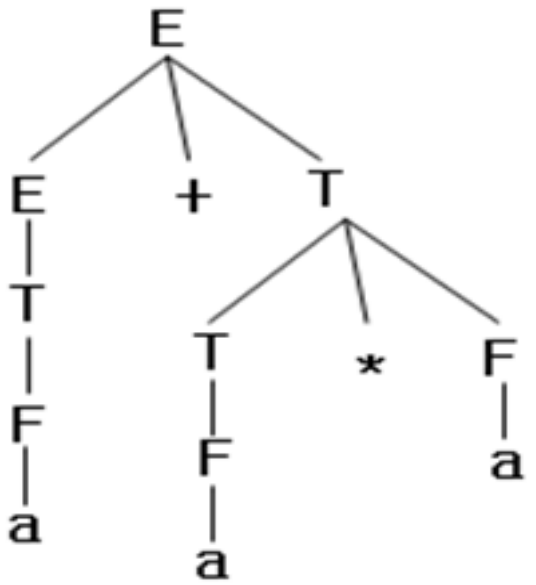


파싱(Parsing)과 모호성(Ambiguity)

[ex]

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid a$$

$a + a * a$ 에 대한 유도트리 ∴ **모호하지 않은 문법**이다.

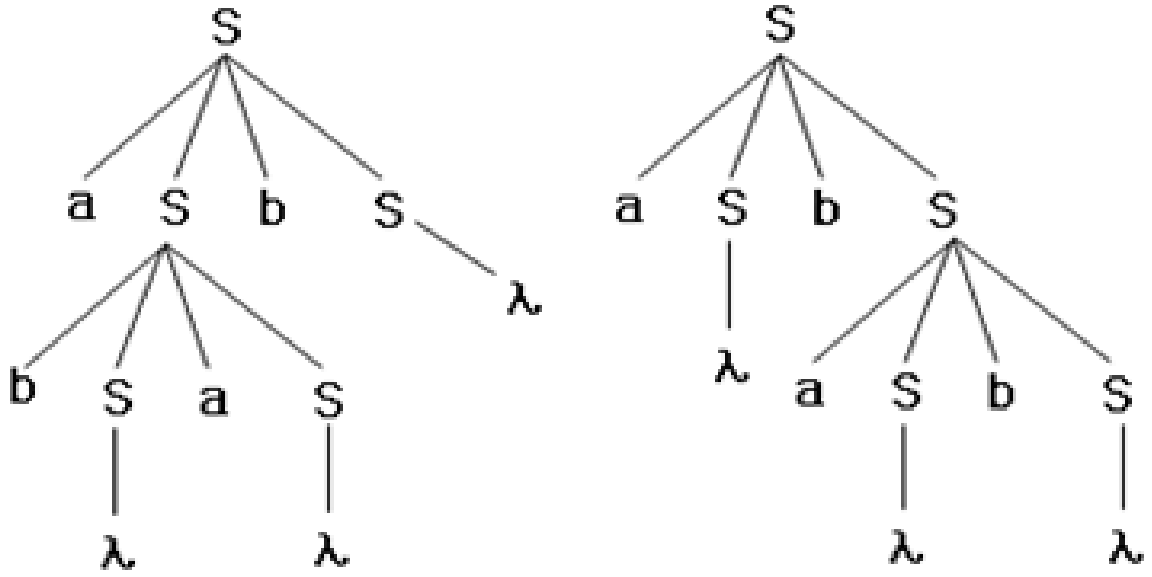


파싱(Parsing)과 모호성(Ambiguity)

[ex] $S \rightarrow aSbS \mid bSaS \mid \lambda$

스트링 $abab$ 에 대한 유도트리

∴ **모호한 문법**이다.



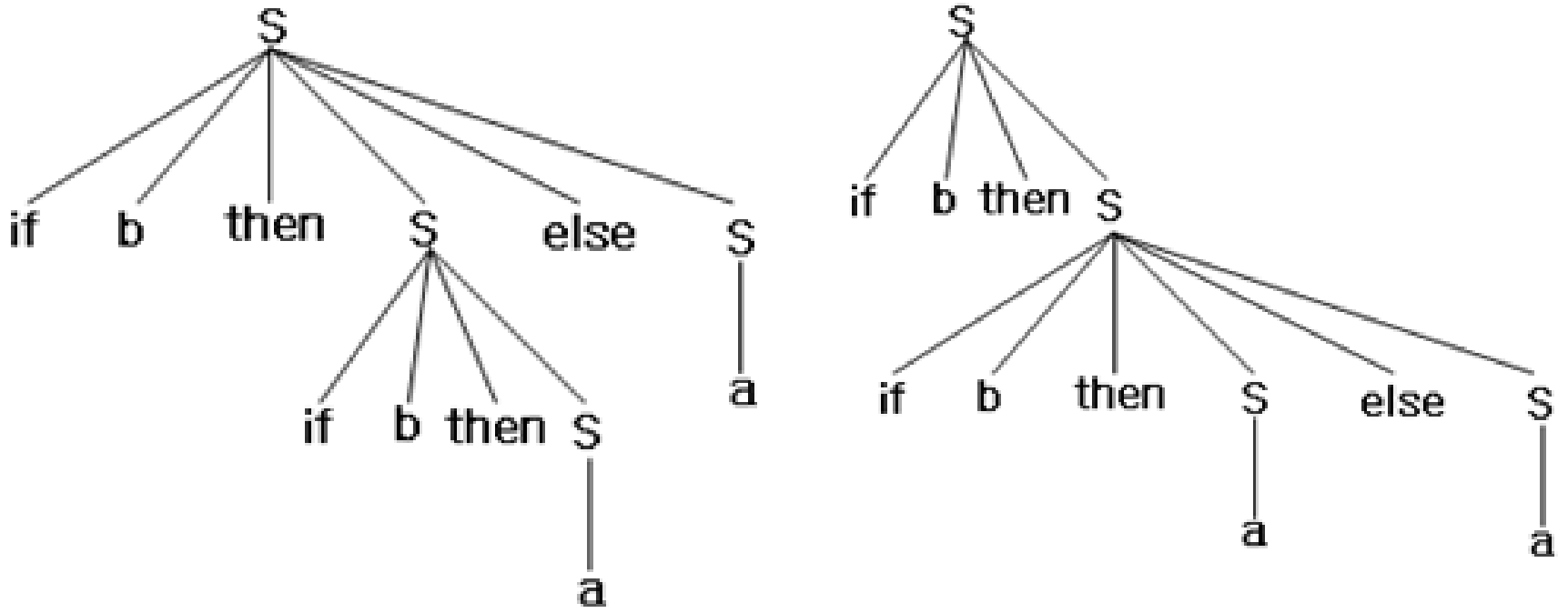
파싱(Parsing)과 모호성(Ambiguity)

[ex]

S → *if b then S else S* | *if b then S* | a

문장 ***if b then if b then a else a*** 에 대한 유도트리

∴ **모호한 문법**이다.



파싱(Parsing)과 모호성(Ambiguity)

- 모호성을 해결하는 방법

$S \rightarrow \textit{if } b \textit{ then } S \textit{ else } S \mid \textit{if } b \textit{ then } S \mid a$

✔ *if* 문을 block으로 처리하는 방법

✔ *else* 문은 가장 가까운 *if* 문의 else로 해석

✔ (*if b then a)* 문을 허용하지 않는 방법.