10. 형식언어

충북대학교

이 재 성

학습내용

- 형식 언어 정의 방법
- 형식 문법 정의 방법
- 언어 및 문법 계층 구조

알파벳과 스트링

알파벳과 스트링

- 잘 정의된 언어는 문장으로 구성됨
- 알파벳은 문장을 이루는 기본적인 심볼

(1) 알파벳(alphabet)

– 심벌들의 유한 집합

– ex) T_1 = {ㄱ,ㄴ,ㄷ,...,ㅎ,ㅏ,ㅑ, … ,ㅡ,ㅣ}

T_2 = {A,B,C, … ,Z}

T_3 = {begin, integer, … ,end}

(2) 스트링(string) (또는 문장, 단어)

– 알파벳 T에 속하는 하나 이상의 심벌들의 나열

(3) 길이(length)

– 스트링을 이루는 심벌들의 개수

– **$|\omega|$ 로 표시**

꼭 기억해야 할 세 가지 개념

1. 언어의 정의
2. 문법의 정의 및 개념
3. 인식기의 의미

(4) empty 스트링(empty string)

- 스트링의 길이가 0인 것
- ε(epsilon) 또는 λ(lambda)로 표시

(5) T^* 알파벳 T에 대하여 empty 스트링을 포함한 모든 스트링의 집합

$$T^+ = T^* - \{\varepsilon\}$$

$T^+ = \varepsilon$ 제외한 set of ∀ string.

T^* : T star

T^+ : T dagger

$T^* =$ set of ∀ string

(6) 접속(concatenation)

– 스트링을 연속으로 연결한 것

– $u = a_1a_2a_3...a_n$, $v = b_1b_2b_3..b_m$, $u \bullet v = a_1a_2a_3...a_nb_1b_2b_3...b_m$

– $u \bullet v$를 보통 uv로 표기.

– $u\varepsilon = u = \varepsilon u$

– $\forall u,v \in T^*, uv \in T^*$. set of ∀string

– $|uv| = |u| + |v|$

(7) a^n

– n개의 a를 나타낸다. ex) $a^3 = aaa$

– $a^0 = \varepsilon$

(8) 문자ω의 **반전**은 문자ω에 반전 표시를 $(\omega^R)^R=\omega$로 표시한다.

i.e., if $\omega = a_1a_2...a_n$ then $\omega^R = a_na_{n-1}...a_1$.

순서 반전...?

언어

- 잘 정의된 언어는 문장으로 구성됨
- 알파벳은 문장을 이루는 기본적인 심볼
- 알파벳 T는 항상 유한 집합, 반면에 T^*는 항상 무한집합

언어(Language): 알파벳 T에 대하여 언어 L 은 T^*의 부분집합

$$\mathbf{L \subseteq T^*}$$

- 언어는 T^*에 속하는 스트링 중 특정 형태만 모은 것
- 유한 언어: 스트링 수가 유한개일 때
- 무한언어: 스트링 수가 무한개일 때

언어 연산

(1) 언어 곱(product)

$LL' = \{xy \mid x \in L \text{ 그리고 } y \in L'\}$

(2) 언어 L의 거듭제곱(powers)은 순환적(recursive)으로 정의된다.

$L^0 = \{\varepsilon\}$

$L^n = LL^{n-1}$ for $n \geq 1$.

(3) L^* : 재귀 전이 클로저(reflexive transitive closure)

$= L^0 \cup L^1 \cup L^2 \cup \ldots \cup L^n \cup \ldots = \bigcup_{i=0}^{\infty} L^i$

☆

(4) L^+ : 전이 클로저(transitive closure)

$= L^1 \cup L^2 \cup \ldots \cup L^n \cup \ldots$

$= L^* - L^0$

언어 표현의 문제점

(1) 어떻게 언어를 표현할 것인가?

유한 언어는 쉽게 표현할 수 있다.

무한 언어는 유한 표현법이 필요하다.

* **무한 언어의 유한 표현**

1. 집합 표현으로 조건제시법
2. 언어 생성 시스템인 문법 : Scheme의 생성
3. 언어의 인식기 : Scheme의 인식

유한 L → easy expression

무한 L → need for finite expression

method of

(2) 모든 언어는 유한한 표현을 가지고 있나?

– 항상 모든 언어에 대하여 유한 표현이 존재하는 것은 아니다.

문법

문법

- 무한 언어의 유한 표현 방법

문법 요소

- 단말기호(terminal) : 정의된 언어의 알파벳
- 비단말기호(nonterminal) :
  - 문법에서 스트링을 생성하는데 사용되는 중간 과정의 기호
  - 언어의 구조를 정의하는데 사용
- 문법 기호: terminal + non-terminal
  - 단말기호와 비단말기호를 합한 것으로 보통 V(vocabulary)로 표시
- 생성규칙:
  - $\alpha \rightarrow \beta$
  - α 를 β 구조로 정의한다는 뜻으로 유도과정에서 α 가 β 로 대체됨

■ $G = (V_N, V_T, P, S)$

- V_N : 비단말(nonterminal) 기호들의 유한 집합
- V_T : 단말(terminal) 기호들의 유한 집합
  - $V_N \cap V_T = \varnothing$, $V_N \cup V_T = V$
- P : 생성 규칙들의 유한 집합 & all
  - $\alpha \rightarrow \beta$, $\alpha \in V^+$, $\beta \in V^*$ all
  - α 는lhs, β는 rhs
- S : 시작 심벌(문장 심벌)

V_N = non-terminal 유한집합

V_T = terminal 유한집합.

P = 생성 규칙의 유한집합.

S = start symbol

[예] $G = (\{S, A\}, \{a, b\}, P, S)$

P : $S \rightarrow aAS$ $\quad S \rightarrow a$

$A \rightarrow SbA$ $\quad A \rightarrow ba$ $\quad A \rightarrow SS$

$\Rightarrow$ $S \rightarrow aAS \mid a$

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

$S \rightarrow aAS \rightarrow abaS$

$\rightarrow abaa$

$\alpha \ \beta \ \gamma \ \delta \ \varepsilon$

■ 유도(derivation)

$\alpha \rightarrow \beta$

1. $\Rightarrow$: "직접 생성"또는"직접 유도"

   if $\alpha \rightarrow \beta \in P$ and $\gamma, \delta \in V^*$ then

   $\gamma \alpha \delta \Rightarrow \gamma \beta \delta$

2. $\overset{*}{\Rightarrow}$: $\alpha_1, \alpha_2, \ldots, \alpha_n \in V^*$ 와 $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$, 라고 가정하면

   $\alpha_1 \overset{*}{\Rightarrow} \alpha_n$

   (0 번 이상의 유도)

3. $\overset{+}{\Rightarrow}$: 한 번 이상의 유도

참고) $\rightarrow$: 생성 규칙에서 사용.

"대치할 수 있음"

$\Rightarrow$: 유도 과정에서 사용한다.

■ L(G) : 문법G에 의해 생성되는 언어

$$L(G) = \{\omega \mid S \overset{*}{\Rightarrow} \omega, \omega \in V_T^*\}$$

☞ ω 는 G의 문장 형태 S 일 때 $\overset{*}{\Rightarrow}$ ω 와 ω ∈ V^*. ⇒ $V_N \cup V_T$

ω 는 G의 문장 S 일 때 $\overset{*}{\Rightarrow}$ ω 와 ω ∈ V_T^*.

P : $S \rightarrow aA \mid bB \mid \varepsilon$
$A \rightarrow bS$
$B \rightarrow aS$

$S \Rightarrow aA$ (생성규칙 $S \rightarrow aA$)
$\Rightarrow abS$ (생성규칙 $A \rightarrow bS$)
$\Rightarrow abbB$ (생성규칙 $S \rightarrow bB$)
$\Rightarrow abbaS$ (생성규칙 $B \rightarrow aS$)
$\Rightarrow abba$ (생성규칙 $S \rightarrow \varepsilon$)

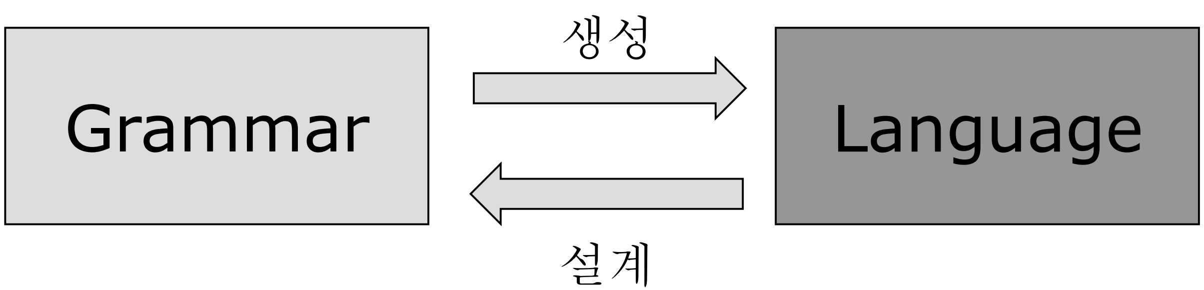
$S \overset{*}{\Rightarrow}$ *abba* 유도 과정

G_1 = ({S}, {a}, P, S) 을 이용하여 $L(G_1)$

$P : S \rightarrow a \mid aS$

$L(G_1) = \{ a^n \mid n \geq 1 \}$

언어 설계



■ $G = (\{A, B, C\}, \{a, b, c\}, P, A)$

$$
\begin{array}{llll}
P: & A \rightarrow abc & & A \rightarrow aBbc \\
 & Bb \rightarrow bB & & Bc \rightarrow Cbcc \\
 & bC \rightarrow Cb & & aC \rightarrow aaB \\
 & aC \rightarrow aa & &
\end{array}
$$

$$L(G) = \{ a^n b^n c^n \mid n \geq 1 \}$$

다양한 문법 예: ex1)

$S \Rightarrow 01$
$S \Rightarrow 0S1 \Rightarrow 0011$
$S \Rightarrow S1 \Rightarrow 00S11 \Rightarrow 000111$

$L(G) = \{0^n 1^n \mid n \ge 1\}$

ex1) $S \rightarrow 0S1 \mid 01$

$S = c$
$S \Rightarrow aSb \Rightarrow acb$
$\quad \Rightarrow aaSbb \Rightarrow aacbb$

$L(G) = \{x \mid a^n c b^n, n \ge 0\}$

ex2)

ex2) $S \rightarrow aSb \mid c$

ex3) $A \rightarrow aB$
$B \rightarrow bB \mid b$

ex3) $A \Rightarrow aB \Rightarrow ab$
$\quad \Rightarrow abB \Rightarrow abb$
$\quad \Rightarrow abbB \Rightarrow abbb$

$L(G) = \{x \mid x \subset ab^n \mid n \ge 1\}$

~~ex4)~~ $A \rightarrow abc$
$Bb \rightarrow bB$
$bC \rightarrow Cb$
$aC \rightarrow aa$

$A \rightarrow aBbc$
$Bc \rightarrow Cbcc$
$aC \rightarrow aaB$

문법 설계

- $L = \{ a^n \mid n \geq 0 \}$ 일 때 문법 :

  $\mathbf{A \rightarrow aA \mid \varepsilon}$ → 0 포함.

- $L = \{ a^n \mid n \geq 1 \}$ 일 때 문법 :

  $\mathbf{A \rightarrow aA \mid a}$ → 1 부터.

- **임베디드** 생성

  $\mathbf{A \rightarrow aAb \mid ab}$

ex1) $L_1 = \{ a^n b^n \mid n \geq 0 \}$

ex2) $L_2 = \{ 0^i 1^j \mid i \neq j,\ i, j \geq 1 \}$

☆

촘스키 계층 (Chomsky Hierarchy)

- 노암 촘스키(Noam Chomsky)

$\alpha \in V^+ \quad \beta \in V^*$

- 생성 형식에 따라

$aBa \rightarrow xByZ$

$$\alpha \rightarrow \beta \in P$$

- Type 0 : 제한 없음(unrestricted grammar, UG) (제한되지 않은 문법)
- Type 1 : 문맥 의존 문법(context-sensitive grammar, **CSG**).

$$\alpha \rightarrow \beta, \quad |\alpha| \leq |\beta|$$

- Type 2 : 문맥 자유 문법(context-free grammar, **CFG**).

$$A \rightarrow \alpha, \text{ 여기서 } A : \text{nonterminal}, \alpha \in V^*.$$

- Type 3 : 정규 문법(regular grammar, **RG**).

$$A \rightarrow tB \text{ or } A \rightarrow t, \text{ (우선형, right-linear)}$$

$$A \rightarrow Bt \text{ or } A \rightarrow t, \text{ (좌선형, left-linear)}$$

$$\text{여기서, } A, B : \text{nonterminal}, \ t \in V_T^*.$$

■ 형식 언어

- T_0 ● REL (Recursively Enumerable Language)
- T_1 ● CSL(Context Sensitive Language)
- T_2 ● CFL (Context Free Language)
- T_3 ● RL(Regular Language)

$A \to aAa \mid bAb \mid c$

$A \to aAa \to aca$

좌우 대칭 $\to abAba$

$\to abcba$

회문 언어

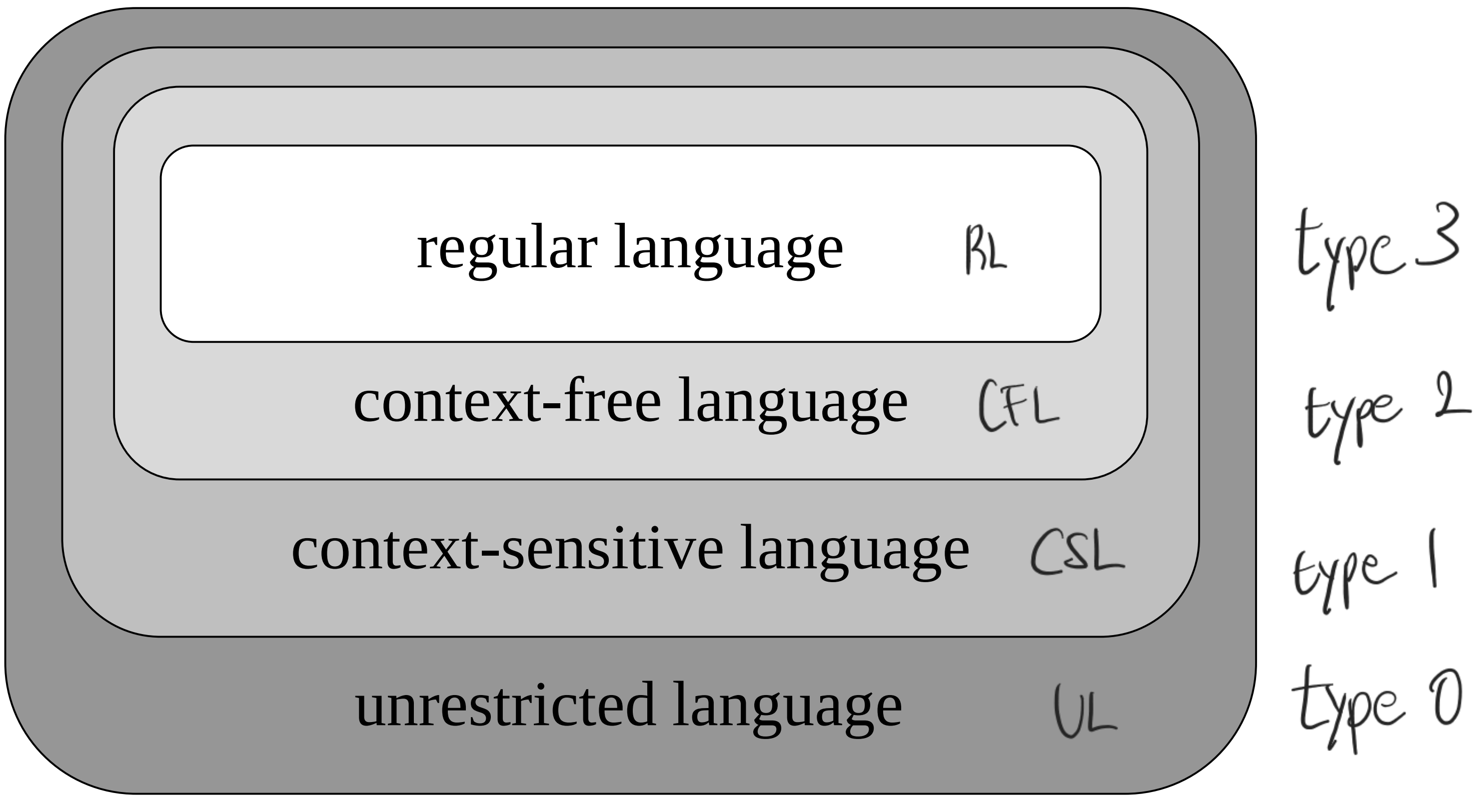
■ 형식 언어의 예

단순 매칭 언어	: $L_m = \{a^n b^n \mid n \geq 0\}$	CFL
중복 매칭 언어	: $L_{dm} = \{a^n b^n c^n \mid n \geq 0\}$	CSL
좌우 대칭 언어	: $L_{mi} = \{\omega\omega^R \mid \omega \in V_T{}^*\}$	CFL
회문 언어	: $L_r = \{\omega \mid \omega = \omega^R\}$	CFL
괄호 언어	: $L_p = \{\omega \mid \omega\text{: balanced parenthesis}\}$	CFL

→ 앞 뒤 같음

ex) 우영우, 토마토

■ 촘스키의 언어 계층


regular language
RL
context-free language
CFL
context-sensitive language
CSL
unrestricted language
UL
type 3
type 2
type 1
type 0

■ 언어 & 인식기

Grammar	Language	Recognizer
type 0 (unrestricted)	recursively enumerable set	Turing machine
type 1 (context-sensitive)	context-sensitive language	linear bounded automata
type 2 (context-free)	context-free language	pushdown automata
type 3 (regular)	regular language	finite automata

automata 에서는 shift/reduce X.

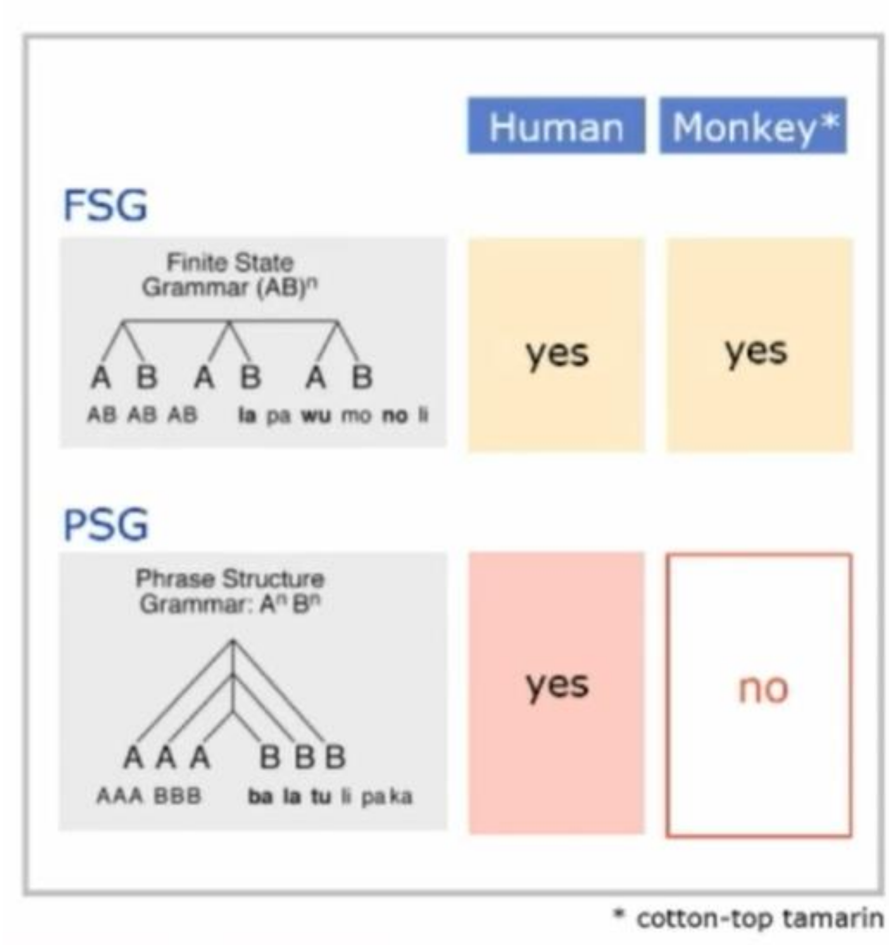
= 어휘 분석기

인간과 원숭이의 언어 인식 능력 차이

- 사람 언어 인식
  - PSG – type 2 language CFG
  - FSG – type 3 language RG
- 원숭이의 언어 인식
  - FSG – type 3 language

- FSG: finite state grammar
- PSG: phrase structure grammar

- Ref: Fitch and Hauser, Computational Constraints on Syntactic Processing in a Nonhuman Primate, Science Jan. 2004.



참고 문헌

- Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, “Compilers – Principles, Techniques, and Tools,” Bell Telephone Laboratories, Incorporated, 1986.
- 오세만, “컴파일러 입문”, 정익사, 2004.