AT_Midterm

Three Basic Concepts

Language

Alphabet을 가지고 만든 임의의 String들의 Subset. $L = \{a^n b^n, n \geq 0\}$

Alphabet

$\Sigma = \{a, b\}$ String을 만드는 데 사용하는 Terminal

String

$w = abab, aaabb...$ alphabet을 0개 이상 사용하여 만든 문자열

Substring

특정 String에 속하는 String

- Prefix : $w = abab$, prefix $p = \{a, ab, aba, abab\}$
- Suffix : $w = abab$, suffix $s = \{b, ba, bab, baba\}$

Symbols Dealing With Language, Alphabet and String

$w = a_1 a_2 a_3 ... a_n$ $v = b_1 b_2 b_3 ... b_m$

- $wv = a_1 a_2 a_3 ... a_n b_1 ... b_m$
- $w^R = a_n a_{n-1} a_{n-2} ... a_1$
- $|w| = n$
  - $|\lambda| = 0$
- w^n
  - $w^0 = \lambda$
  - $w^1 = w$
  - $w^2 = ww$
- Σ^*
  - Σ안에 있는 모든 alphabet을 써서 만들 수 있는 모든 String의 집합
    - λ포함
- $\Sigma^+ = \Sigma^* - \{\lambda\}$

$L = \{a, aa, aab\}$

- a, aa, aab 각각을 sentence라고 한다.
- $\overline{L} = \Sigma^* - L$
- $L^R = \{w^R : w \in L\}$
- $|L| = 3$
- $L^n = LLLLL....$
- $L^0 = \{\lambda\} \neq \{\}$
- $L^* = L^0 \cup L^1 \cup L^2 \cup ...$
- $L^+ = L^1 \cup L^2 \cup L^3 \cup ....$

$L_1 = \{a, aa\}$ $L_2 = \{b, bb\}$

- $L_1 L_2 = \{ab, abb, aab, aabb\}$

$|L_1| = n, (n \geq 0)$ $|L_2| = m, (m \geq 0)$

- $|L_1 L_2| \leq nm$
  - 중복되는 문자열이 있을 수 있기 때문이다.

Grammer

$G = (V, T, S, P)$

- Variables
- Terminal Symbols
- Start variables
- Productions

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

- $S \to aSb$
- $S \to \lambda$

Derivation

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

- $\Rightarrow$ - Derivation Arrow
- $aaSbb$ - 이러한 꼴을 Sentential Form이라고 한다.
- $aabb$ - Terminal을 다 소모한 상태로 Derivation이 완료되었으므로, 이를 Sentence라 하며, 이 Sentence는 Language에 포함된다.

Definition of Language with Grammer

$L(G) = \{w \in T^* : S \Rightarrow^* w\}$

- Grammer G에 의해 생성되는 Language L은,
- Start Symbol S로부터 만들수 있는 모든 string w의 집합이며,
- w는 Terminal T의 원소로만 이뤄진다.

Language to Grammer

$L = \{a^n b^{n+1} : n \geq 0\}$ 을 Grammer로 표현하려면?

- $L = \{a^n b^n, n \geq 0\}$을 표현하는 Grammer의 뒤에 b하나만 추가하면 된다.
- P :
  - $S \to Ab$ `여기서 b 추가함`
  - $A \to aAb|\lambda$ `기존 Grammer`

하나의 Languague를 표현하는 서로 다른 여러 Grammer가 있을 수 있다.

Deterministic Finite Accepters (DFA)

$M = (Q, \Sigma, \delta, q_0, F)$

- Q - Set of States
- Σ - Input alphabet
- δ - $Q \times \Sigma \to Q$ (Transition Function)
- q_0 - Initial State
- $F \subset Q$ - Final States

Example

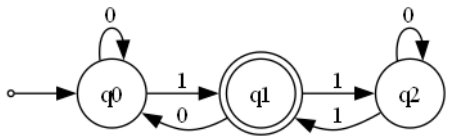
$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0\{q_1\})$

- $\delta(q_n, t) = q_m$
  - t is terminal
- $\delta^*(q_0, w) = q_2$
  - w is string

Init	transition	Destination
q_0	0	q_0
q_0	1	q_1
q_1	0	q_0
q_1	1	q_2
q_2	0	q_2
q_2	1	q_1

- 모든 Input alphabet 에 대한 Edge가 존재해야 함.
- Input alphabet에 따라 다음 State가 하나로 결정되기 때문에 **deterministic** 하다.
- State가 유한한 갯수이므로 **Finite** 하다.
- 특정 String을 Accept/Unaccept 하기 때문에 **Accepter**이다.

DFA Transition Graph



Regular Languages & NonDeterministic Finite Accepters

Equivalence of Deterministic and Nondeterministic Finite Accepter

Reduction of the Number of States in Finite Automata

Reduction of the Number of States in Finite Automata & Regular Expressions

Regular Expressions & Connection Between Regular Expressions and Regular Languages

Connection Between Regular Expressions and Regular Languages

Regular Grammars

Closure Properties of Regular Languages

Closure Properties of Regular Languages and Elementary Questions about Regular Languages

Identifying Nonregular Languages Part 1

Identifying Nonregular Languages

$L = \{a^n b^n : n \geq 0\}$

- $a^m b^m$ - $|y| = k$ 일 때, $a^{m-k} b^m$ 역시 L에 포함되어야 한다. - 그러나 $a^{m-k} b^m \notin L$, 따라서 가정은 모순이다. - L is not regular. $L = \{ww^R : w \in \Sigma^*\}$

- $a^m b^m b^m a^m$ - $|y| = k$ 일 때, $a^{m-k} b^m b^m a^m$ 역시 L에 포함되어야 한다. - 그러나 $a^{m-k} b^m b^m a^m \notin L$, 따라서 가정은 모순이다. - L is not regular.
  $L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$
- $a^m b^{m+1}$ - $|y| = k$ 일 때, $a^{m+k} b^{m+1}$ 역시 L에 포함되어야 한다. - $m + k \geq m + 1$ - 따라서 가정은 모순이다. - L is not regular
  $L = \{(ab)^n a^k : n > k, k \geq 0\}$
- $(ab)^{m+1} a^m$
  - $|y| = k$ 일 때, $(ab)^{m+1-k} a^m$ 역시 L에 포함되어야 한다.
    - y에 b가 포함된 경우 - 모순
    - y에 b가 포함되지 않은 경우 - 패턴 생성 불가능, 모순
  - L is not regular

> pumping lemma를 적용하는 위치는 반드시 string의 앞일 필요는 없다. string의 중간, 끝에서도 사용할 수 있다.

$L = \{a^{n^2} : n \geq 0\}$

- a^{m^2}
  - $|y| = k \leq m$일 때, $a^{m^2} + k$ 역시 L에 포함되어야 한다.
    - $a^{m^2} + k \neq a^{(m+1)^2}$
  - 이는 불가능하므로, 가정은 모순이다.
  - L is not regular

$L = \{a^n b^k c^{n+k} : n \geq 0, k \geq 0\}$

- $a^m b^m c^{2m}$
  - $|y| = k$일 때, $a^{m-k} b^m c^{2m}$역시 L에 포함되어야 한다.
    - $2m - k \leq 2m$
  - 이는 불가능하므로, 가정은 모순이다.
  - L is not regular

Homomorphism을 이용한 증명

$h(a) = a\ h(b) = a\ h(c) = c\ \Gamma(L) = \{a^{n+k} c^{n+k} : n \geq 0, k \geq 0\}$

Homomorphism은 Regular language에 의해 닫혀있으므로, $\Gamma(L)$ 이 regular 하지 않으면, L 역시 regular 하지 않다.

Complement를 이용한 증명

$L = \{a^n b^l : n \neq l\}$ $L' = \{a^n b^l : n = l\}$. which is not regular.

$L \cup L' = L(a^* b^*)$

- $\overline{L} \cap L(a^* b^*) = L'$
  - Since every operation we did is closed in regular languages,
    - L is regular when L' is regular.
- But L' is not regular, so L is also not regular.

Context-free Grammer

> Stronger grammer than regular grammer

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

- $S \to aSa$
- $S \to bSb$
- $S \to \lambda$
- Why it is context free? : 문맥에 영향을 받지 않기 때문.

    - 왼쪽에 Terminal이 나오지 않는다.
- Derivation
    - $S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aabbaa$
- Context free language로는 Regular Language가 표현하지 못하는 언어도 표현할 수 있다.

Opposite : **Context-sensitive**

- $aSb \rightarrow acb$
- $aSc \rightarrow adc$

다음 Grammer가 만드는 Language는?

$S \rightarrow abB\ A \rightarrow aaBb\ B \rightarrow bbAa\ A \rightarrow \lambda$

- $B \rightarrow bba|bbaaBba$
    - $\{L = ab(bbaa)^n bba(ba)^n, n \geq 0\}$

다음 Language가 만드는 Grammer는?

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

> a의 갯수와 b의 갯수가 다르다는 것은, a의 갯수보다 b의 갯수가 많거나, 혹은 그 반대를 의미한다.

Step 1) a의 갯수가 b의 갯수보다 많은 Language를 생성하는 Grammer $S \rightarrow AS_1\ S_1 \rightarrow aS_1b|\lambda\ A \rightarrow aA|a$

Step 2) 이를 바탕으로, a의 갯수와 b의 갯수가 서로 다른 Language를 생성하는 Grammer $S \rightarrow AS_1|S_1B\ S_1 \rightarrow aS_1b|\lambda\ A \rightarrow aA|a\ B \rightarrow bB|b$

다음 Grammer가 생성하는 Language는?

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

> a의 갯수와 b의 갯수가 같은 Language.

만약 이 Grammer를 다음과 같이 바꾼다면? $S \rightarrow aSb|SS|\lambda$

이 Grammer는 아래의 string을 만들 수 없다.

- ba (b로 시작하는 모든 language)
- $abba$

정의하자면, $n_a(w) = n_b(w)$ `1. a의 갯수와 b의 갯수가 같다.` $n_a(v) \geq n_b(v)$ `3. a의 갯수가 항상 b의 갯수보다 크거나 같다.` v is any prefix of w `2. 이 Grammer로 인해 만들어지는 String의 prefix v는`

만약 a가 열린 괄호, b가 닫힌 괄호라면, 이 Grammer은 legal 한 괄호 표현법을 나타내는 Grammer이다.

증명

aSb 가 terminal을 생성하는 유일한 문법이므로, a와 b의 갯수는 항상 같다 (`1.`) a가 항상 b보다 먼저 나오기 때문에, Prefix항상 a의 갯수가 많다. (`2.3.`)

Left-most Derivation and Right-most Derivation

특정 Grammer가 주어졌을 때, Sentential form의 왼쪽 Variable부터 순차적으로 Derivation 하는 것을 **Left-most Derivation**이라고 한다.

Example

$S \rightarrow aAB\ A \rightarrow bBb\ B \rightarrow A|\lambda$

Derivation (**Left-Most**) $S \Rightarrow aAB \Rightarrow abBbB \Rightarrow abAbB \Rightarrow abbBbbB \Rightarrow abbbbB \Rightarrow abbbb$

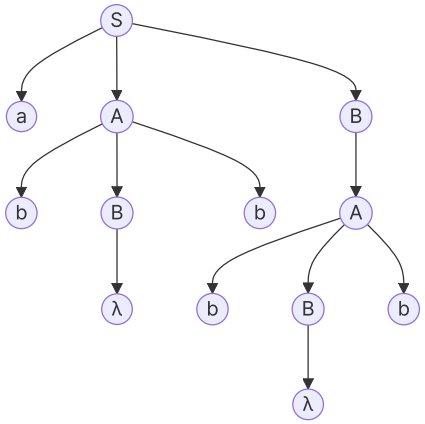
Derivation (**Right-Most**) $S \Rightarrow aAB \Rightarrow aAA \Rightarrow abBbA \Rightarrow abBbbBb \Rightarrow abBbbb \Rightarrow abbbb$

같은 String을 Derivation 하는 **여러가지 Derivation** 방법이 있다.

Parse Tree

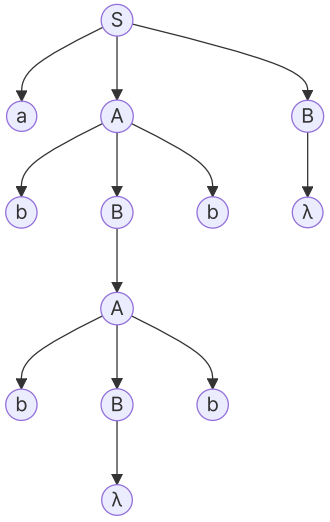
Derivation 과정을 Child Node로 표현한 것

$S \Rightarrow aAB \Rightarrow abBbB \Rightarrow abBbA \Rightarrow abbBbbBb \Rightarrow abbbBb \Rightarrow abbbb$



> Parse Tree 역시 derivation의 갯수 많큼 많아질 수 있다.

$S \Rightarrow aAB \Rightarrow abBbB \Rightarrow abAbB \Rightarrow abbBbbB \Rightarrow abbbbB \Rightarrow abbbb$



> Parse Tree 가 여러개 나오면 그 문법을 Ambiguous 하다고 하는데, 이는 지양해야 한다.