Chapter 3

1. 형식 언어와 정규 표현

목차

형식 언어 (formal language)

- **언어 : 알파벳**으로부터 생성되는 문자열들의 **부분집합**
- **문법 : 문법**은 언어를 **정의**
- **인식기 :** 언어는 **인식기**에 의해 **인식**

N. Chomsky 분류	형식 언어	recognizer = automata
문법	언어	인식기
type 0(무제약 문법)	재귀 열거 언어	튜링 기계(turing machine)
type 1(문맥인식 문법)	문맥인식 언어	선형한계 오토마타(linear-bounded automata)
type 2(문맥자유 문법)	문맥자유 언어	푸시다운 오토마타(push-down automata)
type 3(정규 문법)	정규 언어	유한 오토마타(finite automata)

How are tokens defined?

- 영어 사전에서 찾을 수 없는 단어는?

  gallant, gallivan, gallic, galumph, galvanic

- 변수 이름으로 쓸 수 없는 것은?

  i, sum, 2user

  - *How do you know that?*

Alphabet ∑

- **집합(set) : a collection of *unique* elements**
  - The elements may be *listed in any order*.
  - It may contain *infinite* elements.

    예: {**girl**, animal, boy, **girl**} ➔ {boy, girl, animal}

- **공집합 (empty set): { } 또는 Φ**
  - contains *no* elements

- **알파벳 (Alphabet), ∑ : 기호**(*Symbol*, 문자)들의 유한 집합

  ∑ = {ㄱ, ㄴ, ㄷ, ㄹ, …, ㅎ}

  ∑ = { *a* , *b* , *c* , …, *z* }

  ∑ = { 0 , 1}

String

- **문자열 (String of characters)**
  - A list of **ordered** characters from a *given alphabet* $\sum$
  - The elements of a string need *not* be *unique*.
  - 문자들이 나열된 순서가 중요

[예] $\sum = \{ a, b, c \}$ 일 때

문자열 : $a,\ \ ca,\ ccba,\ aabbcc, \ldots$

단, $abc \neq cba,\ abb \neq ab$

String length

■ 문자열 길이

- 문자열 w 에 포함된 문자들의 개수: $|w|$ 로 표시

  $w = v_1 v_2 v_3 \cdots v_k$ 일 때 $|w| = k$

[예] $\sum = \{ a, b, c \}$ 일 때, $w_1 = abc$, $w_2 = abab$

➔ $|w_1| = 3$, $|w_2| = 4$

Empty string

빈 문자열(*empty string*): ε

- $|\varepsilon| = 0$ ➔ 문자열의 길이가 0인 문자열
  - null string.
  - ε 은 "입실론"으로 발음. λ(람다)로도 표기.

- ε 연산
  - $u\,\varepsilon = u = \varepsilon\,u$
  - $u\,\varepsilon\,v = u\,v$

- a 가 n 번 발생 : $a^1 = a, a^2 = aa, \ldots, a^n = aa \cdots a$
- a 가 한 번도 발생하지 않음 : $a^0 = \varepsilon$

Concatenation

■ 문자열 연결(string concatenation)

- 두 개 이상의 string 을 연결해서 하나의 string으로 만듦
- $u = a_1 a_2 \cdots a_n$, $v = b_1 b_2 \cdots b_m$ 일 때

  ➔ $u \cdot v = uv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m$
- 교환 법칙은 성립하지 않는다. ➔ $uv \neq vu$

[예] u = dog, v = house 일 때

➔ $u \cdot v$ = **dog**house, $v \cdot u$ = house**dog**

$\therefore uv \neq vu$

- **접두사(prefix) 및 접미사(suffix)**
  - 문자열 w = ***u v*** 에 대해 **u**가 접두사, ***v*** 가 접미사

  [예] w = house 일 때,

  접두사 : $h, ho, hou, hous, house$

  접미사 : $e, se, use, ouse, house$

Language

- **언어(Language)** → A set of *strings* ***from a given alphabet***
  - 문자열들의 집합

Σ = {0, 1}

1. {0, 10, 1011}
2. {ε, 0, 00, 000, 0000, 00000, . . . }
3. the set of strings having an ***even number of ones***.
   {11, 0101, 1001, 10101010, . . . }

Σ = { ***characters available on a computer keyboard*** }

1. {0, +123, -123.456e+10}
2. { *if, for, while, case, switch*, … }
3. { 인천대학교, 정보기술대학, 컴퓨터공학부, … }

언어 연산 : Union and Concatenation

- **두 개의 언어 *L* 과 *M* 에 대한 연산**
  - ***L, M* 은 집합이므로 언어 연산은 집합(set) 연산**
- **합(Union) 연산 : $L \cup M$**
  - $L \cup M = \{s | s \in L \cup s \in M\}$

  [예] $L = \{a, ba, bbb\}$, $M = \{aaa, bbb, aba, bba\}$ 일 때

  ➔ $L \cup M = \{a, ba, bbb, aaa, aba, bba\}$

- **접속(Concatenation) 연산 : $L \cdot M$**
  - $L \cdot M = \{s \cdot t \mid s \in L \cap t \in M\}$
    - 분배 법칙을 적용

  [예] $L = \{aa, bb\}$, $M = \{ab, ba\}$ 일 때

  ➔ $L \cdot M = \{aaab, aaba, bbab, bbba\}$

언어 연산 : L^n

▪ 거듭제곱 연산 : L^n

- $L^0 = \{ \varepsilon \}$
- $L^1 = LL^0 = L \cdot \{ \varepsilon \} = L$
- $L^2 = LL^1, \ldots$
- $L^n = LL^{n-1}$, 단, $n \geq 1$

[예] $L = \{ a,\ ba \}$ 일 때,

➔ $L^0 = \{ \varepsilon \}$

➔ $L^1 = L = \{ a,\ ba \}$

➔ $L^2 = LL^1 = \{a, ba\}\{a, ba\} = \{aa, aba, baa, baba\}$

➔ $L^3 = LL^2 = \{a, ba\}\{aa, aba, baa, baba\}$

$= \{aaa, aaba, abaa, ababa, baaa, baaba, babaa, bababa\}$

Kleene Closure * 와 Positive Closure +

- **Kleene Closure :** L^* (L-스타로 읽음)

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

- **Positive Closure :** L^+ (L-대거(dagger)로 읽음)

$$L^+ = L^1 \cup L^2 \cup \cdots \cup L^n \cup \cdots = \bigcup_{i=1}^{\infty} L^i = L^* - L^0 = L^* - \{\varepsilon\}$$

[예] $L = \{0,\ 1\}$ 일 때,

➔ L^* : ε **포함.** 0과 1로 만들어지는 모든 문자열. **생략 가능.**

➔ L^+ : ε **제외.** 0과 1로 만들어지는 모든 문자열. **반드시 1번은 발생.**

Regular Expression and Regular Language

정규 언어 (Regular Language)

- 정규 표현에 의해 정의되는 언어

정규 표현 (Regular Expression, RE)

알파벳 Σ 에 대해

1. $a \in \Sigma$ 는 정규 표현
2. Φ(공집합) 와 ε 도 정규 표현
3. r, s 가 각각 정규 언어 L_r, L_s를 정의하는 정규 표현일 때
   (1) $r \mid s$ 는 $L_r \cup L_s$를 나타내는 정규 표현
   (2) $r \cdot s$ 는 $L_r \cdot L_s$ 를 나타내는 정규 표현
   (3) r^*는 $(L_r)^*$를 나타내는 정규 표현

Quiz : { } 와 { ε }은 같은가 다른가?

RE : Choice (선택)

- **r , s 가 정규 표현이면, $r \mid s$ 도 정규 표현**
  - 예 : $\sum = \{ a,\ b \}$ 일 때
    - a, b 는 정규 표현 ➔ $a \mid b$ 도 정규 표현
    - $a \mid b$ 가 생성하는 언어는?
      - $L(a \mid b) = L(a) \cup L(b) = \{a\} \cup \{b\} = \{a, b\}$
    - a, ε 은 정규 표현 ➔ $a \mid \varepsilon$ 도 정규 표현
    - $a \mid \varepsilon$ 이 생성하는 언어는?
      - $L(a \mid \varepsilon) = \{ a, \varepsilon \}$
  - 다중 선택으로 확장 가능

$$L(a \mid b \mid c \mid d) = \{ a, b, c, d \}$$

RE : Concatenation (연결)

- **r , s 가 정규 표현이면, $r \cdot s$ 도 정규 표현**
  - 예 : $\sum = \{ a,\ b,\ c \}$ 일 때
    - a, b 는 정규 표현 ➔ $\boldsymbol{a \cdot b}$ 도 정규 표현($\cdot$ 를 없애고 $\boldsymbol{ab}$로 사용)

      $\boldsymbol{a\ b}$ 가 생성하는 언어는?

      $L(a\ b\ ) = L(a)L(b) = \{a\}\{b\} = \{\ ab\ \}$

    - 정규 표현 $\boldsymbol{(a|b)c}$

      ➔ $L(\ (a|\ b\ )c) = L((a|b)) \cdot L(c) = \{a, b\} \cdot \{c\} = \{\ ac, bc\ \}$

    - 정규 표현 $\boldsymbol{a(b|c)}$

      ➔ $L(a(b|c)) = L(a) \cdot L((b|c)) = \{a\} \cdot \{b, c\} = \{\ ab, ac\ \}$

  - 다중 연결로 확장가능

    $L(\ a \cdot b \cdot c \cdot d) = L(abcd) = \{\ abcd\ \}$, **한 개의 원소**

RE : Repetition (반복)

- $\sum = \{a,\ b\}$ 일 때
  - $bb,\ a|bb$ 는 정규 표현 ➔ $(\boldsymbol{a|bb})^*$ 도 정규 표현
  - $(\boldsymbol{a|bb})^*$가 생성하는 언어는?

$$r^* = r^0 \cup r^1 \cup r^2 \cup r^3 \cup \cdots = \bigcup_{i=0}^{\infty} r^i$$

$$r^0 = (a \mid bb)^0 = \{\varepsilon\}$$

$$r^1 = (a \mid bb)^1 = \{a, bb\}$$

$$r^2 = (a \mid bb)^2 = \{a, bb\}\{a, bb\} = \{aa, abb, bba, bbbb\}$$

$$r^3 = (a \mid bb)^3 = (a \mid bb)^1 (a \mid bb)^2 = \{a, bb\}\{aa, abb, bba, bbbb\}$$

$$= \{aaa, aabb, abba, abbbb, \ldots\}$$

a또는 bb로 이루어진 모든 문자열. **생략도 가능.**

RE : 우선 순위

- $a \mid b^*$는 어떻게 해석하는 것이 맞을까?

  $(a \mid b)^*$ 또는 $a \mid (b)^*$

- RE 연산 기호의 우선 순위
  - *repetition* > *concatenation* > *choice*
    - $a \mid b\, c^* = a \mid (b \cdot (c^*))$
    - $ab \mid c^* d = (ab) \mid ((c^*) \cdot d)$

- 우선순위를 바꾸고 싶으면 괄호를 사용

  $a \mid b\, c$ ➔ $(a \mid b)\, c$

Practice #1

- 정규 표현 $(a \mid bb)^*$ 이 생성할 수 없는 문자열을 모두 고르시오.
  - a
  - $aaaaaaaaaaaaa$
  - $abbaaabbbbaaa$
  - $abbabaaa$
  - $bbbbabb$
  - $bbba$

Practice #2

- $\sum = \{ a,\ b \}$ 이다.
- 아래 정규 표현이 생성하는 언어는?
  1. $(aa \mid b\ )\ (a \mid bb\ )$
  2. $a^*(a \mid b\ )$
- 아래 2개의 정규 표현이 생성하는 언어는 같은가 다른가?
  1. $(a|bb)^*$ 와 $a \mid bb^*$
  2. $a\ (ba)^*$ 와 $(ab)^*a$

Practice #3

- $\sum = \{ a,\ b,\ \boldsymbol{c} \}$ 일 때, 한 개의 b 만을 포함하는 문자열을 정의하시오.

$$(a|c)^*\ b\ (a|c)^*$$

- $\sum = \{ a,\ b,\ \boldsymbol{c} \}$ 일 때, 기껏해야 한 개의 b 만을 포함할 수 있는 문자열을 정의하시오.

$$(a|c)^*\ |\ (a|c)^*\ b\ (a|c)^*$$

$$(a|c)^*\ \ (b|\varepsilon)\ \ (a|c)^*$$

Regular Definition

- 정규 표현으로 한 개 이상의 숫자열을 정의

  (0| 1 | 2 | … | 9) (0| 1 | 2 | … | 9)*

- 정규 정의를 사용하여 digit pattern을 정의

  digit digit*

  digit = 0 | 1 | 2 | … | 9

- 어느 방식이 사용하기 쉽고 이해하기 쉬운가?
- 어느 방식이 확장하기 쉬운가?

Extension to Regular Expressions(1/2)

- ***r+*** : ***one*** **or** ***more*** **repetitions (ε 은 제외)**

  '+'는 Dagger로 읽음.

  (0| 1) (0| 1)* ➔ (0| 1)+

- **. : any character**

  .*b.* ➔ all strings that contain *at least* one *b*

- **[_-_] : a range of characters**

  a | *b* | *c* |...| *z* ➔ [a-z]

  1|2|…|7 ➔ [1-7]

  a | *b* | *c* = [abc] ➔ [a-c]

  [A – Za – z] ≠ [A-z]

Extension to Regular Expressions(2/2)

- **~ : any character *not* in a given set**

  ~(*a* | *b* | *c*) ➔ 알파벳에서 *a, b, c*를 뺀 임의의 문자

  - Lex에서는 ^(*caret*)를 사용

    [^a] [^abc]

- **? : optional sub-expressions**

  - 자연수 정의에서

    natural = [0-9]+

    signedNatural = natural | + natural | - natural

  - ? 메타 기호를 사용하여 다시 표현하면

    signedNatural = (+|-)? natural

Regular Expressions for P. L. Tokens

Numbers

nat = [0-9]+

$signedNat$ = (+|-)? nat

number = $signedNat$("." nat)?(E $signedNat$)?

Meta symbol이 아닌 소수점 기호

Reserved words

reserved = if | while | do | ...

Identifiers

letter = [a − zA − Z]

digit = [0-9]

identifier = letter (letter | digit)*

Scanning 과정에서의 Ambiguity 해결

- **Some strings can be matched by *several* REs.**
  - **문제 1 :** `if` 는 키워드 일 수도, identifier일 수도 있다.
  - **문제 2 :** `< >` 는 2개의 token(<, >)이거나 1개의 token(≠)일 수 있다.

- ***Disambiguating* rules (모호성 해결 원칙)**
  - **문제 1 해결방법:** *keyword* interpretation is generally *preferred*.
  - **문제 2 해결방법:** the principle of ***longest substring***
    - Matching 되는 길이가 가장 긴 token 정의를 먼저 적용

Ambiguity : 애매함

Example

If I were a bird, I can fly.

Reserved = "If" | "I"

ID = letter (letter | digit)*

- Q1: If 는 키워드일까, ID일까? Why?

If (a <> b) a = 100;

If (a > b) a = 100;

relational_op = "<>" | ">" | "<"

- Q2: <>는 <, >, <> 중 어느 것으로 인식될까?

Comments (주석)

■ 주석 처리

- 1개의 *delimiter* (구분기호)에 의해 둘러싸인 경우

  `{ This is a Pascal comment }` ➔ `{(~})*}`

- 특정 문자로 시작하여 해당 줄의 끝까지가 주석

  `; This is a Scheme comment` ➔ `;(~newline)*`

  `-- This is an Ada comment` ➔ `--(~newline)*`

- 1개 이상의 *delimiter*에 의해 둘러싸인 경우
  - `/* This is a C comment */`
  - *ba* … (no appearance of ab) … *ab* 의 형태

    ba`(~(`*ab*`))*` *ab*