프로그래밍 언어론

Assignment #2

전기컴퓨터공학부 정보컴퓨터공학전공

201524582 정희석

1. Give an unambiguous grammar that generates the same language as

S → SS | (S) | ( )

A. 먼저 위의 문법이 ambiguous 한 이유를 찾는다

S -> SS => (S)S => (())() - (1)

=> S(S) => ()(()) – (2)

위의 S 는 (1)과 (2)로 두가지 결과가 나오게 된다. 이를 해결하기 위해서

문법을 수정하면 2 가지 문법이 나온다. 이 때 phaser 에서 원하는 결과가 나오는 문법을 결정하게 한다.

-1) S -> ( )S | (S) | ()

-2) S -> S( ) | (S) | ()

2. The syntax of the monkey language is quite simple, yet only monkeys can speak it without making mistakes. The alphabet of the language is {a, b, d, #}, where # stands for a space. The grammar is

<stop>::= b | d

<plosive> ::= <stop>a

<syllable> ::= <plosive> | <plosive><stop> | a <plosive> | a <stop>

<word> ::= <syllable> | <syllable><word><syllable>

<sentence> ::= <word> | <sentence>#<word>

Which of the following speakers is the secrets agent masquerading as a monkey?

Ape : ba#ababadada#bad#dabbada

Chimp: abdabaadab#ada

Baboon: dad#ad#abaadad#badadbaad

A.

<Rule>

Stop = b, d | Plosive = ba, da

Syllable = ba, da | bab, bad, dab, dad| aba, ada| ab, ad

Word = syllable로만 이루어져 있어야한다. <word>::=<syllable>|<syllable><word><syllable>

Sentence = word로만 이루어져 있어야한다, 즉 syllable로만 이루어져 있으면 됨. 그리고 syllable는 무조건 홀수 개. <sentence>::=<word>|<sentence>#<word>

Ape: ba#aba/bad/ada#bad#dab/ba/da
=> <sentence> => <sentence>#<word> => <sentence>#<word>#<word>
=> <sentence>#<word>#<word>#<word> => <sentence>#<word>#<word>#<word>#<word>
-> <word>#<word>#<word>#<word>#<word>
-> <syllable>#<syllable><word><syllable>#<syllable>#<syllable><word><syllable>
-> <plosive>#<syllable><syllable><syllable>#<syllable>#<syllable><syllable><plosive>
-> <plosive>#<syllable><syllable><syllable>#<syllable>#<syllable><plosive><plosive>
=> O.K

Chimp: ab/da/ba/ad/ab#ada
=> <sentence> => <sentence>#<word> => <word>#<word>
-> <syllable><word><syllable>#<syllable>
-> <syllable><syllable><word><syllable><syllable>#<syllable>
-> <syllable><plosive><syllable><syllable><syllable>#<syllable>
-> <syllable><plosive><plosive><syllable><syllable>#<syllable>
=> O.K

Baboon: dad/#ad/#aba/ad/ad/#ba/dad/ba/ad
=> <sentence> => <sentence>#<word> => <sentence>#<word>#<word>
=> <sentence>#<word>#<word>#<word> => <word>#<word>#<word>#<word>
-> <syllable>#<syllable>#<syllable><word><syllable>#<syllable><word><syllable>
->
<syllable>#<syllable>#<syllable><syllable><syllable>#<plosive><syllable><syllable><syllable>
<syllable?> => error발생! 구문 오류: Syllable missing!
따라서 spy는 Baboon이다!

3. Give regular expression for

(a) Binary strings ending in 01

A. L((0|1)*01)

(b) Decimal integer divisible by 5

A. L([0-9]*(5|0))

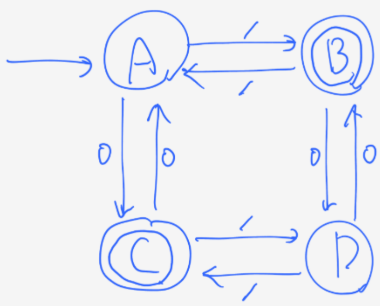
(c) C identifiers

A. L((_ | [a-zA-Z])(_ | [a-zA-Z0-9])*)

(d) Binary strings consisting of either an odd number of 1s or an odd number of 0s

A. L(1(1(1+0)+0(0+1))+0(0(1+0)+1(0+1))) = L(1(0+1)*+0(0+1)*)

DFA



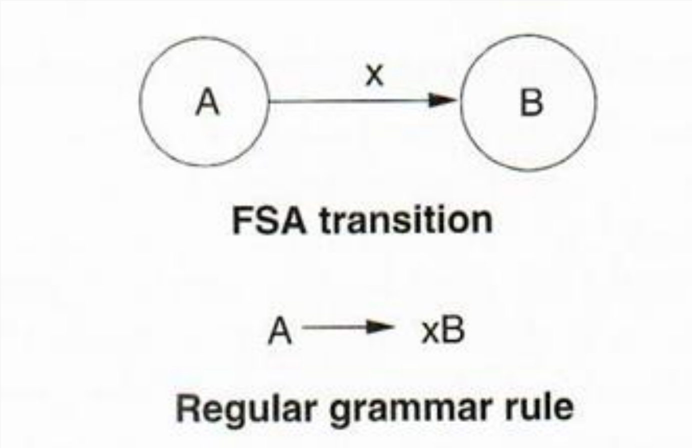
✓ $A = 1B + 0C$

$B = 1A + 0D$ $\quad B = 1(1B+0C)+0(0B+1C)$

$C = 0A + 1D$ $\quad C = 0(1B+0C)+1(0B+1C)$

$D = 0B + 1C$

4. Show that any FSA can be represented by a regular grammar and any regular grammar can be recognized by an FSA. The key is to associate each nonterminal of the grammar with a state of the FSA. For example, the transformation of the below figure becomes the rule A →xB. (How do you handle final states?)



FSA => NFA, DFA 가 존재

Regular grammar => NFA 를 먼저 증명 – (1)

(1) 먼저 Regular grammar G 를 선언 G 는 아무런 규칙이 없다고 가정, 정규표현식을 나타내면 L(G) = {} 이므로 이 grammar 로 동작하는 machine 는 -> ○이다.

다음으로 G 는 ε 를 규칙으로 가지고 있다고 가정, 정규표현식으로 나타내면 L(G) = {ε} 이므로 이 grammar 로 동작하는 machine 는 -> ◎이다.

다음으로 G 는 a 를 규칙으로 가지고 있다고 가정, 정규표현식으로 나타내면 L(G) = {a} 이므로 이 grammar 로 동작하는 machine 는 ->○-a->◎이다.

위의 machine 을 FA 로 보면 S->aF 로 NFA 가 되는 것을 알 수 있다. 따라서 모든 Regular grammar 는 NFA 로 변환이 가능하다.

NFA => DFA 를 증명 – (2)

모든 NFA 는 ε-NFA 를 통하여 DFA 로 변환이 가능하며 DFA 는 NFA 에 포함되어 있으므로 NFA => DFA 이다.

(1)과 (2)에 의해 Regular grammar 은 FSA 의해 recognize 될 수 있음을 증명한다.

5. Give the finite-state automaton and the regular grammar for the following:

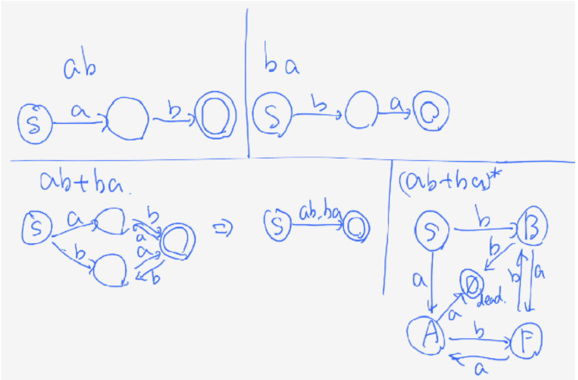
(ab ∨ ba)* ∨ (ab)*

A. ∨: 논리합으로 둘 중에 1개이상 있는가 => +에 해당!

즉. (ab + ba)* + (ab)* 이다!

=> +: 둘 중 하나 1개 이상, 0개 이상 반복 가능

=> ε, ab, ba, abab, baab, abba, abbaab... => (ab+ba)*와 동일하다!



(ab+ba)* => G=({S,A,B,F},{a,b}, P, S) 이 때의 P를 찾아야한다.

위의 그림의 FA를 참조하면

FA M = ({S,A,B,F},{a,b},δ,S,F)

δ (S,a)=A, δ (S,b)=B, δ (A,b)=F, δ (B,a)=F, δ (F,a)=A, δ (F,b)=B

S → aA, S → bB, A → bF, B → aF. F → aA, F → bB, F → ε

=>P : S → aA | bB

A → bF

B → aF

F → aA | bB | ε

따라서 주어진 논리식(ab ∨ ba)* ∨ (ab)*은

Finite Automata M = ({S,A,B,F},{a,b},δ,S,F),

δ (S,a)=A, δ (S,b)=B, δ (A,b)=F, δ (B,a)=F, δ (F,a)=A, δ (F,b)=B 이고

정규 문법 G = ({S,A,B,F},{a,b}, P, S),

P:

S → aA | bB

A → bF

B → aF

F → aA | bB | ε

로 나타낼 수 있다.