PUSAN NATIONAL UNIVERSITY

REPORT

프로그래밍언어론 과제2

과 목 명 :	프로그래밍언어론
지도교수 :	
학 과 :	전기컴퓨터공학부 정보컴퓨터공학전공
학 번 :	
이 름 :	장 수현
제 출 일 :	2020년 4월 15일

부산대학교

1. Give an unambiguous grammar that generates the same language as S → SS | (S) | ( )

Sol)

주어진 문법에 따라 문자열 "( ) ( ) ( )"을 만들 수 있는데 해당 문자열에 대해서 다음과 같이 두 개의 parse tree가 얻어질 수 있다.

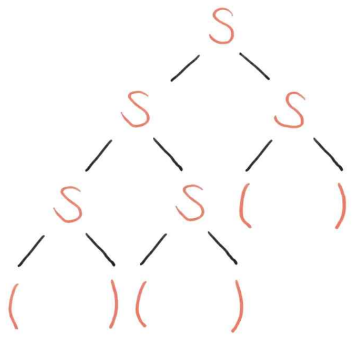

그림 1-1. 문자열 "( ) ( ) ( )"의 parse tree 1.

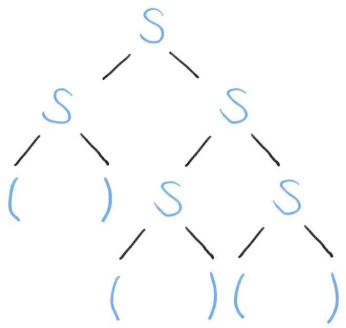

그림 1-2. 문자열 "( ) ( ) ( )"의 parse tree 2.

따라서 이 문법은 모호한 문법(ambiguous grammar)이라고 한다. 모호성을 제거하기 위해 문법을 S → S( ) | (S) | ( ) 로 수정하면 문자열 "( ) ( ) ( )"을 유도할 수 있는 parse tree는 다음으로 유일하게 된다.

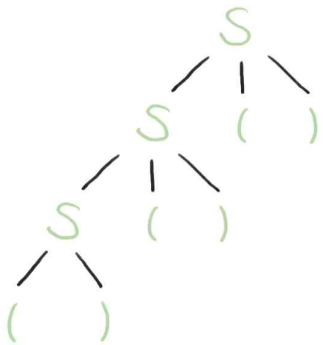

그림 1-3. 문자열 "( ) ( ) ( )"의 유일한 parse tree.

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?

(a) Ape : ba#abababadada#bad#dabbada
(b) Chimp : abdabaadab#ada
(c) Baboon : dad#ad#abaadad#badadbaad

Sol)

(a) ba#abababadada#bad#dabbada
-> ba#abababadada#<stop>a<stop>#dabbada
-> ba#abababadada#<plosive><stop>#dabbada
-> ba#abababadada#<syllable>#dabbada
-> ba#abababadada#<word>#dabbada
-> ba#abababadada#<word>#dabba<stop>a
-> ba#abababadada#<word>#dab<stop>a<stop>a
-> ba#abababadada#<word>#dab<stop>a<plosive>
-> ba#abababadada#<word>#dab<plosive><plosive>
-> ba#abababadada#<word>#dab<plosive><syllable>
-> ba#abababadada#<word>#dab<syllable><syllable>
-> ba#abababadada#<word>#dab<word><syllable>
-> ba#abababadada#<word>#da<stop><word><syllable>
-> ba#abababadada#<word>#<stop>a<stop><word><syllable>
-> ba#abababadada#<word>#<plosive><stop><word><syllable>

-> ba#abababadada#<word>#<syllable><word><syllable>
-> ba#ababadada#<word>#<word>
-> ba#ababa<stop>ada#<word>#<word>
-> ba#aba<stop>a<stop>ada#<word>#<word>
-> ba#aba<plosive><stop>ada#<word>#<word>
-> ba#aba<syllable>ada#<word>#<word>
-> ba#aba<word>ada#<word>#<word>
-> ba#aba<word>a<stop>a#<word>#<word>
-> ba#aba<word>a<plosive>#<word>#<word>
-> ba#aba<word><syllable>#<word>#<word>
-> ba#a<stop>a<word><syllable>#<word>#<word>
-> ba#a<plosive><word><syllable>#<word>#<word>
-> ba#<syllable><word><syllable>#<word>#<word>
-> ba#<word>#<word>#<word>
-> <stop>a#<word>#<word>#<word>
-> <plosive>#<word>#<word>#<word>
-> <word>#<word>#<word>#<word>
-> <sentence>#<word>#<word>#<word>
-> <sentence>#<word>#<word>
-> <sentence>

Ape가 한 말은 start non-terminal인 <sentence>로부터 시작되었으므로 Ape는 monkey이다.

(b) abdabaadab#ada
-> abdabaada<stop>#ada
-> abdabaad<syllable>#ada
-> abdaba<stop><syllable>#ada
-> abdaba<syllable><syllable>#ada
-> abda<stop>a<syllable><syllable>#ada
-> abda<plosive><syllable><syllable>#ada
-> abda<syllable><syllable><syllable>#ada
-> ab<stop>a<syllable><syllable><syllable>#ada
-> ab<plosive><syllable><syllable><syllable>#ada
-> ab<syllable><syllable><syllable><syllable>#ada
-> a<stop><syllable><syllable><syllable><syllable>#ada

-> <syllable><syllable><syllable><syllable><syllable>#ada

-> <syllable><syllable><word><syllable><syllable>#ada

-> <syllable><word><syllable>#ada

-> <word>#ada

-> <sentence>#ada

-> <sentence>#a<stop>a

-> <sentence>#a<plosive>

-> <sentence>#<syllable>

-> <sentence>#<word>

-> <sentence>

Chimp가 한 말은 start non-terminal인 <sentence>로부터 시작되었으므로 monkey이다.

(c) dad#ad#abaadad#badadbaad

-> ... // (a), (b) 와 동일하게 변환 과정을 거친다.

-> <word>#<word>#<syllable><word><syllable>#<syllable><word><syllable><syllable>

-> <word>#<word>#<word>#<word><syllable>

-> <sentence>#<word>#<word>#<word><word>

-> <sentence>#<word>#<word><sentence>

-> <sentence>#<word><sentence>

-> <sentence><sentence>

Baboon이 한 말은 하나의 start non-terminal로부터 시작된 말이 아니다. 따라서 Baboon은 monkey가 아니다.

3. Give regular expression for

(a) Binary strings ending in 01

(b) Decimal integer divisible by 5

(c) C identifiers

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

Sol)

(a)

0으로 시작하는 이진값을 허용한다면,

(0 v 1)* 01

0으로 시작하는 이진값을 허용하지 않는다면,

(1 (0 v 1)*)* 01

(b)

0으로 시작하는 십진값을 허용한다면,

(0 v 1 v 2 v ... v 9)* (0 v 5)

0으로 시작하는 십진값을 허용하지 않는다면,

((1 v 2 v ... v 9) (0 v 1 v 2 v ... v 9)*)* (0 v 5)

(c)

letter (letter v digit)*

(d)

0으로 시작하는 이진값을 허용한다면,

((0*10*10*)* 1 (0*10*10*)*) v ((1*01*01*)* 0 (1*01*01*)*)

0으로 시작하는 이진값을 허용하지 않는다면,

((0*10*10*)* 1 (0*10*10*)*) v (1 (1*01*01*)* 0 (1*01*01*)*)

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 transition

A ⟶ xB

Regular grammar rule

sol)

정규 문법은 정규 언어를 완전히 표현하는 문법으로, 또한 유한 오토마타와 완전히 대응한다. 정규 문법으로부터 만들어지는 모든 정규 표현식은 NFA를 구성할 수 있고, 모든 NFA는 DFA로 변환될 수 있다. 즉, 모든 정규 문법에 대응하는 유한 오토마타가 적어도 하나 있고, 반대로 모든 유한 오토마타에 대응하는 정규 문법이 적어도 하나 존재한다. 따라서 모든 유한 오토마타는 정규 문법으로 나타낼 수 있고, 모든 정규 문법은 유한 오토마타에 의해서 인식될 수 있다. 추가적으로 이 둘은 정규식과도 동치 관계에 있다.

5. Give the finite-state automaton and the regular grammar for the following: (ab ∨ ba)* ∨ (ab)*

Sol)

정규 표현 (ab + ba)* + (ab)*를 NFA로 변환한 뒤, DFA로 변환할 수 있다. 그리고 DFA로부터 정규 문법을 얻을 수 있다.

i) NFA

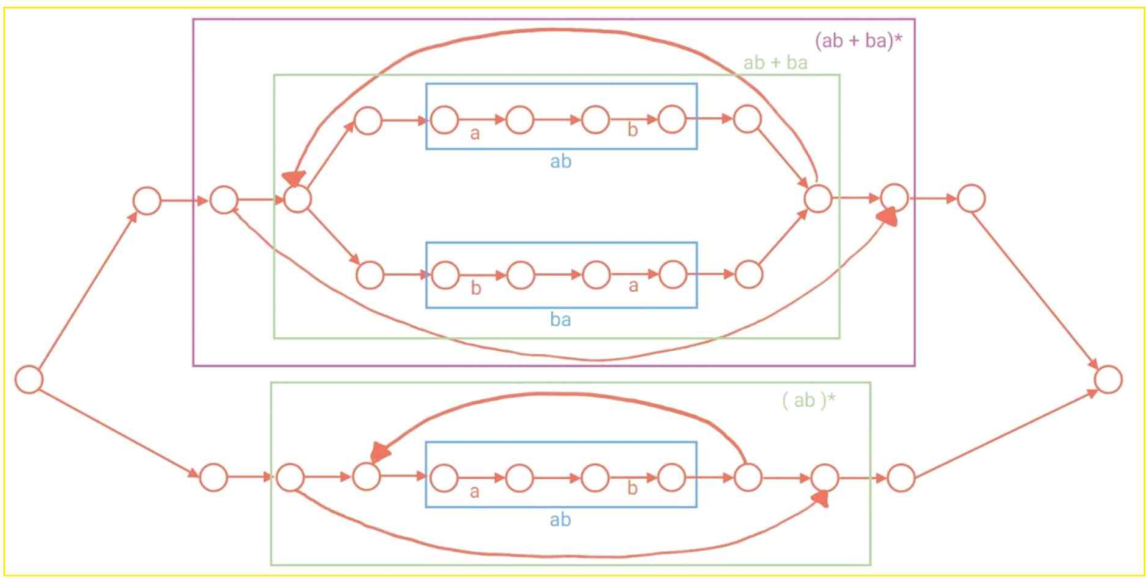

그림 5-1. 정규 표현 (ab + ba)* + (ab)*의 NFA 표현.

ii) DFA

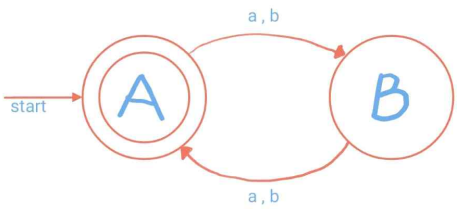

그림 5-2. 그림 5-1을 변환한 DFA 표현.

iii) 정규 문법

A → aB | bB

B → aA | bA