컴파일러의 기초: Homework 2 for Problem Solving

가능한 간단하게 답을 기술하시오. Due:

Problem 1. (25)
다음의 ambiguous grammar 에 대하여 답하시오.

- $E \rightarrow E\#E$
- $E \rightarrow @E$
- $E \rightarrow (E)$
- $E \rightarrow a$

(a) 위의 grammar가 ambiguous 함을 증명하기 위해 어떠한 string 에 대해 두가지 leftmost derivation 을 보이고 그 parse tree를 그리시오 (5).
(b) 위의 grammar를 equivalent한 unambiguous grammar로 rewrite 하되 # 는 right-associative 하고 @ 보다 lower precedence를 갖도록 한다. rewrite된 grammar를 이용 하여 다음 string의 parse tree를 그리시오: "$@a\#a\#@a$" (Dragon pp. 30-32, 174-175 참조) (20)

Problem 2. (20)
Alphabet {a, b, c, d, e} 에서 정의된 다음의 grammar 에 대하여 답하시오. (A: start symbol)

- $A \rightarrow aB$
- $A \rightarrow Db$
- $B \rightarrow \epsilon$
- $B \rightarrow cA$
- $D \rightarrow dBD$
- $D \rightarrow eD$
- $D \rightarrow \epsilon$

(a) 각 Nonterminal 에 대하여 $Nullable()$, $FNE()$, $FOLLOW()$ set 을 구하시오 (5).
(b) LL(1) parse table을 construct 하시오 (10).
(c) 다음의 string을 parse 할 경우 stack 과 input의 상태를 순서적으로 보이시오: "$acdcaeb$" (5)

Problem 3. (30)
다음의 Language 에 대하여 답하시오: $\{a^i b^j | i \geq j\}$
(a) 이 Language 를 accept 하는 간단한 grammar는 다음과 같다.

- $S \rightarrow aS$
- $S \rightarrow aSb$
- $S \rightarrow \epsilon$

이 grammar가 ambiguous함을 보이시오. (5)
(b) 위의 grammar를 left-factoring 시키면 다음과 같다.

- $S \rightarrow aST$
- $S \rightarrow \epsilon$
- $T \rightarrow b$
- $T \rightarrow \epsilon$

위의 grammar가 여전히 ambiguous 함을 보이시오. 위의 grammar에 대해 LL(1) parse table 을 construct 하면 이 table 에 conflict 가 있음을 찾을 수 있을 것이다. (즉, 두가지 production 이 있는 table entry 가 있음; no wonder since it is ambiguous). 이 두 가지 production 중 하나를 제거 하여 parse table을 modify 하면, 어떻게 위의 language를 올바로 LL(1) parsing 할 수 있는지 설명하시오. (15)
(c) 다음의 grammar는 위의 language를 accept하는 unambiguous version 이다.

- $S \rightarrow aS$
- $S \rightarrow T$
- $T \rightarrow aTb$
- $T \rightarrow \epsilon$

이 grammar에 대하여 LL(1) parse table을 construct 하시오. (여전히 conflict가 있음). (b) 에서 처럼 conflicting한 production 중 하나를 제거하면, 이경우에는 incorrect한 LL(1) parser가 됨을 보이시오.(10)

NOTE: 위의 language 는 많은 programming language 에 포함된 `if..then..else..` 구조에서 `else` 부분이 optional 인 경우를 표현한다. ("a"는 `if` $< expr >$ `then` $< stmt >$를 나타내고 "b"는 `else` $< stmt >$를 나타냄). 위에서 보듯, 이 language 는 LL(1) 이 아니며 conflicting entry를 제거함으로서 working parser를 만들수 있으나, 언제나 가능한 것은 아니다. LR parsing은 이를 간단히 다룰 수 있다.