204433 การแปลภาษาโปรแกรม
การบ้านที่ 4

ให้นิสิตตอบคำถามต่อไปนี้

1. เขียน CFG ที่ accept สตริงที่มีจำนวนตัวอักษร a เท่ากับจำนวนตัวอักษร b

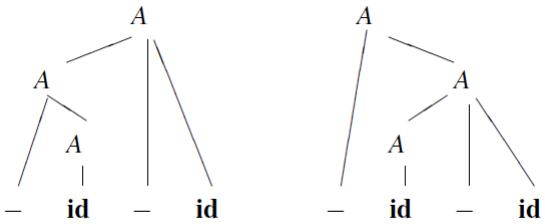
   S -> ε
   S -> aSbS
   S -> bSaS

2. จงแสดงว่า grammar ต่อไปนี้

   A -> - A
   A -> A - id
   A -> id

   มี ambiguity โดยการหาสตริงหนึ่งตัวที่มี parse tree ได้มากกว่าหนึ่งรูปแบบ แสดงสตริงที่ได้และ parse tree ทั้งสอง จากนั้นให้เขียน grammar ใหม่โดยกำจัด ambiguity ทิ้ง และแสดง parse tree ที่ได้จากการ derive สตริงที่เราหามาในตอนแรกเพื่อพิสูจน์ว่า grammar มี ambiguity

   พิจารณาสตริง: - 5 - 4
   มี parse tree ได้สองรูปแบบ



   ด้านซ้ายให้ผลลัพธ์ - 5 - 4 = - 9 ส่วนด้านขวาให้ผลลัพธ์ - (5 - 4) = 1

   แก้ grammar เพื่อกำจัด ambiguity โดยให้ unary - มี precedence มากที่สุด ได้ grammar ดังต่อไปนี้
   A -> A - B
   A -> B
   B -> -B
   B -> id

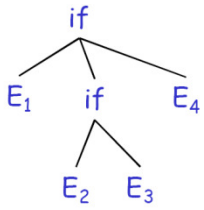
3. พิจารณา grammar ของประโยค if-then-else ต่อไปนี้

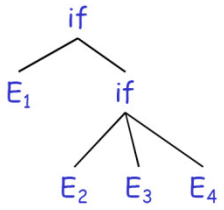
   S -> if E then S | if E then S else S | OTHER

โดย S แทน non-terminal ที่เป็น statement ของโปรแกรม E แทน non-terminal ที่เป็น expression ของโปรแกรม และ OTHER เป็น non-terminal หรือ terminal ที่ไม่เกี่ยวข้องกับประโยค if-then-else ที่เรากำลังพิจารณา

- อธิบายว่าทำไม grammar นี้ ambiguous (คำใบ้: ลองนึกถึงที่เราได้คุยกันในชั้นเรียนว่า ambiguous grammar คือ grammar ที่เราสามารถสร้าง parse tree ได้มากกว่าหนึ่งแบบเวลาที่เราทำ derivation เพื่อตรวจจับสตริง)

ประโยคต่อไปนี้ if E1 then E2 then E3 else E4 **มี** parse tree **ได้มากกว่าหนึ่งรูปแบบดังต่อไปนี้**





4. กำจัด left recursion ใน grammar ต่อไปนี้ให้หมดสิ้น

$$
\begin{aligned}
Exp &\rightarrow Exp + Exp \\
Exp &\rightarrow Exp - Exp \\
Exp &\rightarrow Exp * Exp \\
Exp &\rightarrow Exp / Exp \\
Exp &\rightarrow \mathbf{num} \\
Exp &\rightarrow (Exp)
\end{aligned}
$$

เขียน grammar **ใหม่ที่กำจัด** left recursion **ออกได้ทั้งหมดดังต่อไปนี้**

$$
\begin{aligned}
Exp &\rightarrow \mathbf{num}\, Exp_1 \\
Exp &\rightarrow (\, Exp\,)\, Exp_1 \\
Exp_1 &\rightarrow + Exp\, Exp_1 \\
Exp_1 &\rightarrow - Exp\, Exp_1 \\
Exp_1 &\rightarrow * Exp\, Exp_1 \\
Exp_1 &\rightarrow / Exp\, Exp_1 \\
Exp_1 &\rightarrow \varepsilon
\end{aligned}
$$

5. กำจัด left recursion ใน grammar ต่อไปนี้ และนำผลลัพธ์ของ grammar ที่ได้มาทำ left factoring

E -> E E+

E -> E E *

E -> num

กำจัด left-recursion ได้โดยการเขียน grammar ด้านบนใหม่ดังต่อไปนี้

$$
\begin{aligned}
E &\rightarrow \mathbf{num}\, E' \\
E' &\rightarrow E + E' \\
E' &\rightarrow E * E' \\
E' &\rightarrow \varepsilon
\end{aligned}
$$

ทำการ left-factor หลังจากกำจัด left-recursion ได้โดยการเขียน grammar ใหม่ดังต่อไปนี้

$$
\begin{aligned}
E &\rightarrow \mathbf{num}\, E' \\
E' &\rightarrow E\, Aux \\
E' &\rightarrow \varepsilon \\
Aux &\rightarrow + E' \\
Aux &\rightarrow * E'
\end{aligned}
$$

6. คำนวณหา FIRST และ FOLLOW ของ non-terminal และ terminal ใน grammar ต่อไปนี้

   Z -> d

   Z -> XYZ

   Y -> **ε**

   Y -> c

   X -> Y

   X -> a

   จากนั้นสร้างตาราง predictive parsing และบอกว่า grammar นี้เป็น LL(1) หรือไม่ โดยให้ X Y และ Z เป็น non-terminal ส่วน a c และ d เป็น terminal

FIRST(X) = {a, c, **ε**}

FIRST(Y) = {c, **ε**}

FIRST(Z) = {a, c, , d}

FIRST(a) = {a}

FIRST(c) = {c}

FIRST(d) = {d}

FOLLOW(X) = {a, c, d}

FOLLOW(Y) = {a, c, d}

FOLLOW(Z) = {}

FOLLOW(a) = FOLLOW(X) = {a, c, d}

FOLLOW(c) = FOLLOW(Y) = {a, c, d}

FOLLOW(d) = FOLLOW(Z) = {}

ตาราง predictive parsing เป็นดังต่อไปนี้

	a	c	d
X	$X \to a$ $X \to Y$	$X \to Y$	$X \to Y$
Y	$Y \to \varepsilon$	$Y \to \varepsilon$ $Y \to c$	$Y \to \varepsilon$
Z	$Z \to XYZ$	$Z \to XYZ$	$Z \to d$ $Z \to XYZ$

และจะได้ว่า grammar นี้ไม่ใช่ LL(1) เนื่องจากในตาราง parsing ในบางช่องของคู่ non-terminal และ terminal เช่น X กับ a ที่มี production ที่เป็นไปได้มากกว่าหนึ่งอยู่