HW5

林宸昊

4.3 $S \to (L) \mid a$ (a) 输出括号对数

$L \to L,S \mid S$ (b) 输出嵌套深度

(a) 拓广该文法：提供用生输出括号对数的属性 *.pair

$S' \to S$ => print(S.pair)

$S \to (L)$ => S.pair = L.pair + 1

$S \to a$ => S.pair = 0 (a不提供括号)

$L \to L_1, S$ => L.pair = L_1.pair + S.pair

$L \to S$ => L.pair = S.pair

(b) 拓广如上，提供描述深度的属性 *.depth

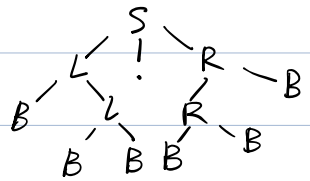
$S' \to S$ => print(S.depth)

$S \to (L)$ => S.depth = L.depth + 1

$S \to a$ => S.depth = 0

$L \to L_1, S$ => L.depth = max(L_1.depth, S.depth)

$L \to S$ => L.depth = S.depth.



4.9(b) 同样拓文法，但使用L属性运用.

$S \to L.R$ => S.val = L.val + R.val

$S \to L$ => S.val = L.val

$L \to BL_1$ => B.in = L.c = L_1.c × 2 ; L.val = B.c + L_1.val.

$L \to B$ => B.c = L.c = 1 ; L.val = B.c

$R \to R_1 B$ => B.in = R.c = R_1.c / 2 ; R.val = R_1.val + B.c

$R \to B$ => B.in = R.c = 0.5 ; R.val = B.c (单个R最终封顶)

$B \to 0$ => B.c = 0 (0不产生贡献)

$B \to 1$ => B.c = B.in (B失去了val，自左向右的计算会由R为B提供其贡献).

4.12

(a) a 的嵌套深度可仅由左部括号决定，即支持L属性而使用，又a本身无法预知其深度，换言之，需要继承。

S'->S => S'->{S.depth=0} S

S->(L) => S->{L.depth=S.depth+1}(L)

S->a => S->a{print(S.depth)} 遇到a即输出。

L->L1,S => L->{L1.depth=L.depth}L1,{S.depth=L.depth}S

L->S. => L->{S.depth=L.depth}S

(b) 同样仅从a或()无法得知其为第n个字符，则引入继承属性in，综合属性num。其中in指当前文法符号之前已生现的字符数，num指当前文法符号展开为字符序列后最终字符的位置。

S'->{S.in=0} S

S->{L.in=S.in+1}(L){S.num=L.num+1}

S->a{S.out=S.in+1; print(S.out)}

L->{L1.in=L.in} L1,{S.in=L1.out+1} S{L.out=S.out}

L->{S.in=L.in} S{L.out=S.out}