HW05

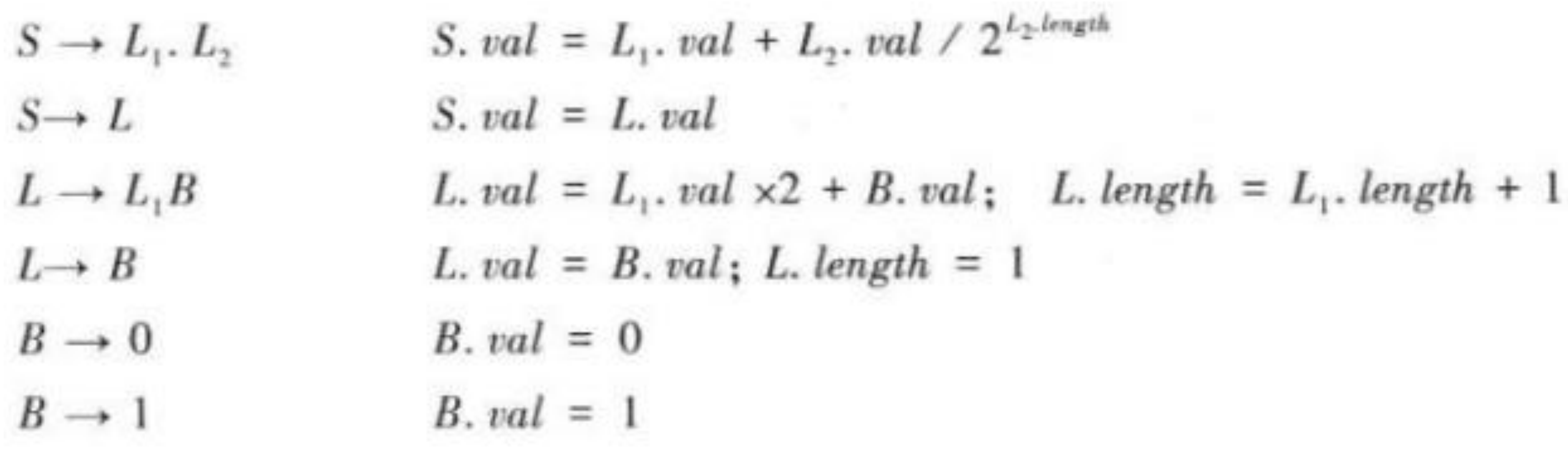
（1）习题4.9， 并分别给出（a）和（b）两个语法制导定义的属性栈代码实现（非yacc代码）。

文法:

S -> L.L | L

L -> LB | B

B -> 0 | 1



$S \rightarrow L_1.L_2$	$S.val = L_1.val + L_2.val / 2^{L_2.length}$
$S \rightarrow L$	$S.val = L.val$
$L \rightarrow L_1B$	$L.val = L_1.val \times 2 + B.val;\quad L.length = L_1.length + 1$
$L \rightarrow B$	$L.val = B.val;\ L.length = 1$
$B \rightarrow 0$	$B.val = 0$
$B \rightarrow 1$	$B.val = 1$

属性栈代码

● S->L1.L2	stack[ntop].val=stack[top-2].val+stack[top].val/stack[top].length
● L->L1B	stack[ntop].val=stack[top-1].val*2+stack[top].val
	stack[ntop].length=stack[top-1].length+1
● L->B	stack[ntop].length=1
● B->0	stack[ntop].val=0
● B->1	stack[ntop].val=1

(b)——翻译方案

S → L.R	L.w = 1; R.w = 0.5; S.val = L.val + R.val;
S → L	L.w = 1; S.val = L.val
L → L1B	L1.w = L.w * 2; B.w = L.w; L.val = B.c + L1.val
L → B	B.w = L.w; L.val = B.c;
R → BR1	B.w = R.w; R1.w = R.w / 2; R.val = B.c + R1.val
R → B	B.w = R.w; R.val = B.c;
B → 0	B.c = 0;
B → 1	B.c = B.w;

(b)

S->M{L.w=M.w}L.N{R.w=N.w}R	S.val=L.val+R.val
M->ε	M.w=1
N->ε	N.w=0.5
S->P{L.w=P.w}L	S.val=L.val
P->ε	P.w=1
L->{Q.win=L.w}Q{L1.w=Q.wout}L1{T.win=L.w}T{B.w=T.wout}B{L.val=B.c+L1.val}	
Q->ε	Q.wout=Q.win*2
T->ε	T.wout=T.win
L->{B.w=R.w}B	R.val=B.c

(b)

- R->{U.win=R.w}U{B.w=U.wout}B{V.win=R.w}V{R1.w=V.wout}R1{R.val=B.c+R1.val}
- U->ε U.wout=U.win
- V->ε V.wout=V.win/2
- R->{B.w=R.w}B R.val=B.c`
- B → 0 B.c = 0
- B → 1 B.c = B.w

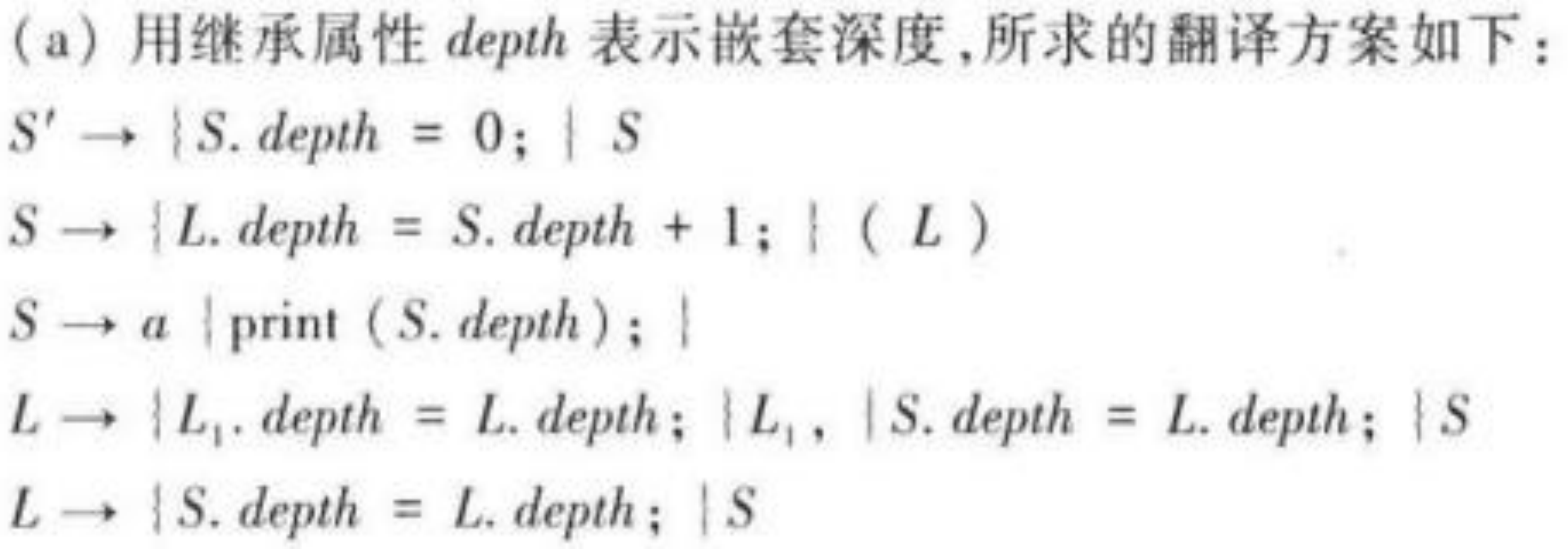
（b）——属性栈代码

S->ML.NR	stack[ntop].val=stack[top].val+stack[top-3].val
M->ε	stack[ntop].w=1
N->ε	stack[ntop].w=0.5
S->PL	stack[ntop].val=stack[top].val
P->ε	stack[ntop].w=1
L->QL1TB	stack[ntop].val=stack[top].c+stack[top-2].val
Q->ε	stack[ntop].w=stack[top].w*2
T->ε	stack[ntop].w=stack[top-2].w
L->B	stack[ntop].val=stack[ntop].c

（b）——属性栈代码

- R->UBVR1 stack[ntop].val=stack[top].val+stack[top-2].c
- U->ε stack[ntop].w=stack[top].w
- V->ε stack[ntop].w=stack[top-2].w/2
- R->B ` stack[ntop].val=stack[ntop].c
- B->0 stack[ntop].c=0
- B->1 stack[ntop].c=stack[top-1].w

4.12 （a）



(a) 用继承属性 *depth* 表示嵌套深度,所求的翻译方案如下：

$S' \to \{S.depth = 0;\}\ S$

$S \to \{L.depth = S.depth + 1;\}\ (\ L\ )$

$S \to a\ \{\mathrm{print}\ (S.depth);\}$

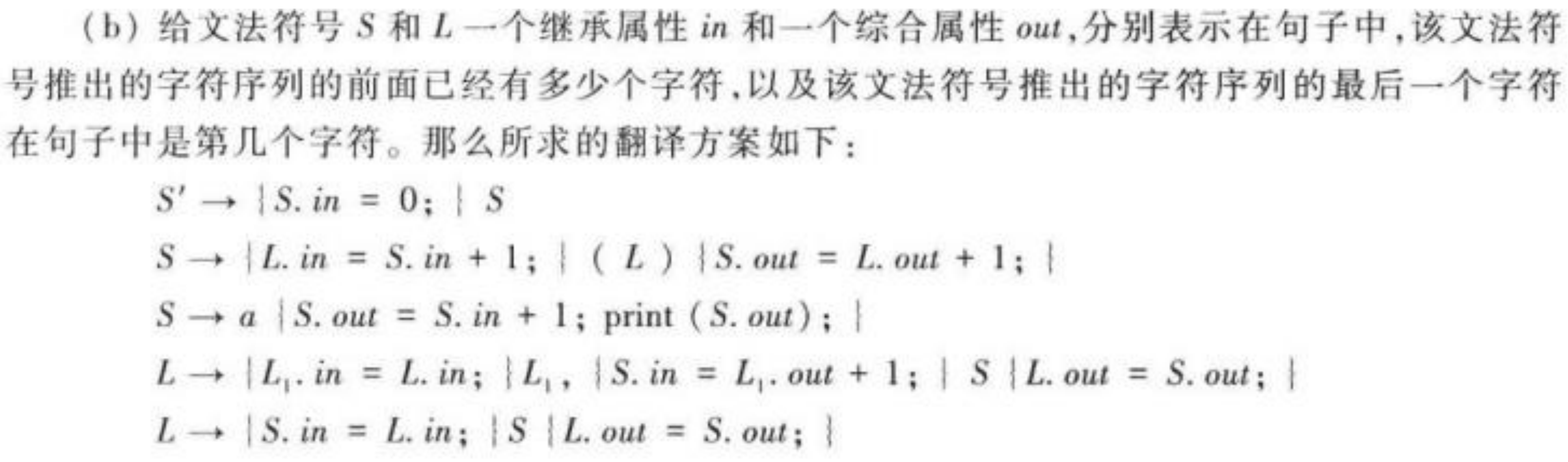
$L \to \{L_1.depth = L.depth;\}\ L_1\ ,\ \{S.depth = L.depth;\}\ S$

$L \to \{S.depth = L.depth;\}\ S$

4.12(a)

- S’→M{S.depth = M.s;}S
- M→ε{M.s = 0;} val[ntop]= 0;
- S→({N.i = S.depth;}N{L.depth = N.s;}L)
- N→ε{N.s = N.i+1;} val[ntop] = val[top - 1]+ 1;
- S→a{print(S.depth);} print (val[top - 1]);
- L→{L_1.depth = L.depth;}L_1,{P.i = L.depth;}P{S.depth = P.s;}S
- P→ε{P.s = P.i;} val[ntop] = val[top - 2]
- L→{S.depth = L.depth;}S

4.12(b)



(b) 给文法符号 S 和 L 一个继承属性 *in* 和一个综合属性 *out*，分别表示在句子中，该文法符号推出的字符序列的前面已经有多少个字符，以及该文法符号推出的字符序列的最后一个字符在句子中是第几个字符。那么所求的翻译方案如下：

$$S' \rightarrow \{S.in = 0;\}\ S$$

$$S \rightarrow \{L.in = S.in + 1;\}\ (\ L\ )\ \{S.out = L.out + 1;\}$$

$$S \rightarrow a\ \{S.out = S.in + 1;\ \text{print}\ (S.out);\}$$

$$L \rightarrow \{L_1.in = L.in;\}\ L_1,\ \{S.in = L_1.out + 1;\}\ S\ \{L.out = S.out;\}$$

$$L \rightarrow \{S.in = L.in;\}\ S\ \{L.out = S.out;\}$$

4.12(b)

- S’→M{S.in = M.s;}S
- M→ε{M.s = 0;}
- S→({N.i = S.in;}N{L.in = N.s;}L){S.out = L.out + 1;}
- N→ε{N.s = N.i+1;}
- S→a{S.out = S.in+1; print(S.out);}
- L→{L_1.in = L.in;}L_1,{P.i = L_1.out+1;}P{S.in = P.s;}S{L.out = S.out}
- P→ε{P.s = P.i;}
- L→{S.in = L.in;}S{L.out = S.out}

4.12(b)

S’→MS	
M→ε	stack[ntop].in = 0;
S→(NL)	stack[ntop].out = stack[top - 1].out + 1;
N→ε	stack[ntop].in = stack[top - 1].in + 1;
S→a	stack[ntop].out = stack[top - 1].in + 1; print(stack[ntop].out);
L→L1,PS	stack[ntop].out = stack[top].out
P→ε	stack[ntop].in = stack[top - 1].out + 1
L→S	stack[ntop].out = stack[top].out

3

移进(	(	
移进id	(id	
F->id归约	(F	print(6)
T->F归约	(T	print(4)
E->T归约	(E	print(2)
移进+	(E+	
移进id	(E+id	
F->id归约	(E+F	print(6)
T->F归约	(E+T	print(4)
E->E+T归约	(E	print(1)
移进)	(E)	
F->(E)归约	F	print(5)
T->F归约	T	print(4)
移进*	T*	
移进id	T*id	
F->id归约	T*F	print(6)
T->T*F归约	T	print(3)
E->T归约	E	print(2)

4-4.3-递归下降语法分析函数

```
● void S(){
    if(lookahead()==’(‘){match(‘(‘);L();match(‘)’)’;}
    else if (lookahead()==’a’){match(‘a’);}
    else error()
}
  void L(){
    S();
    while(lookahead()==’,’){match(‘,’);S();}
    }
```

4-4.3-预测翻译器

- 消除左递归:

S’->S	print(S.val)
S->(L)	S.val=L.val+1
S->a	S.val=0
L->ST	L.val=S.val+L.val
T->,ST_1	T.val=S.val+T_1.val
T->ε	T.val=0

4-4.3-预测翻译器

```
● void S’(){print(S());}
● int S(){
      int val;
      if (lookahead()==’(‘){match(‘(‘); val=L()+1; match(‘)’);}
      else {match(‘a’); val=0;}
      return val;
      }
  int L(){
      int val; val=S()+T();return val;
}
```

4-4.3-预测翻译器

- int T()

```
{
    int val=0;
    if(lookahead()==’,’) {match(‘,’);val=S()+T();}
    return val
}
```

（也可以将一个nodeptr作为参数和返回值，将属性设置为nodeptr的属性即可）

4-4.12-预测翻译器

S’->S	S.depth=0
S->(L)	L.depth=S.depth+1
S->a	print(S.depth)
L->ST	S.depth=L.depth;T.depth=L.depth
T->,ST_1	S.depth=T.depth;T_1.depth=T.depth
T->ε	

4-4.12-预测翻译器

```
● void S’(){S(0);}
● void S(int depth){
      int mydep;
      if (lookahead()==’(‘)
      {match(‘(‘); mydep=depth+1; L(mydep); match(‘)’);}
      else
      {match(‘a’);print(depth);}
}
● void L(int depth){int mydep=depth;S(mydep);T(mydep);}
```

4-4.12-预测翻译器

- void T(int depth)
- {

```
    int mydep=depth;

    if(lookahead())==’,’){match(‘,’);S(mydep);T(mydep);}

    }
```

谢谢！