

Required Exercises

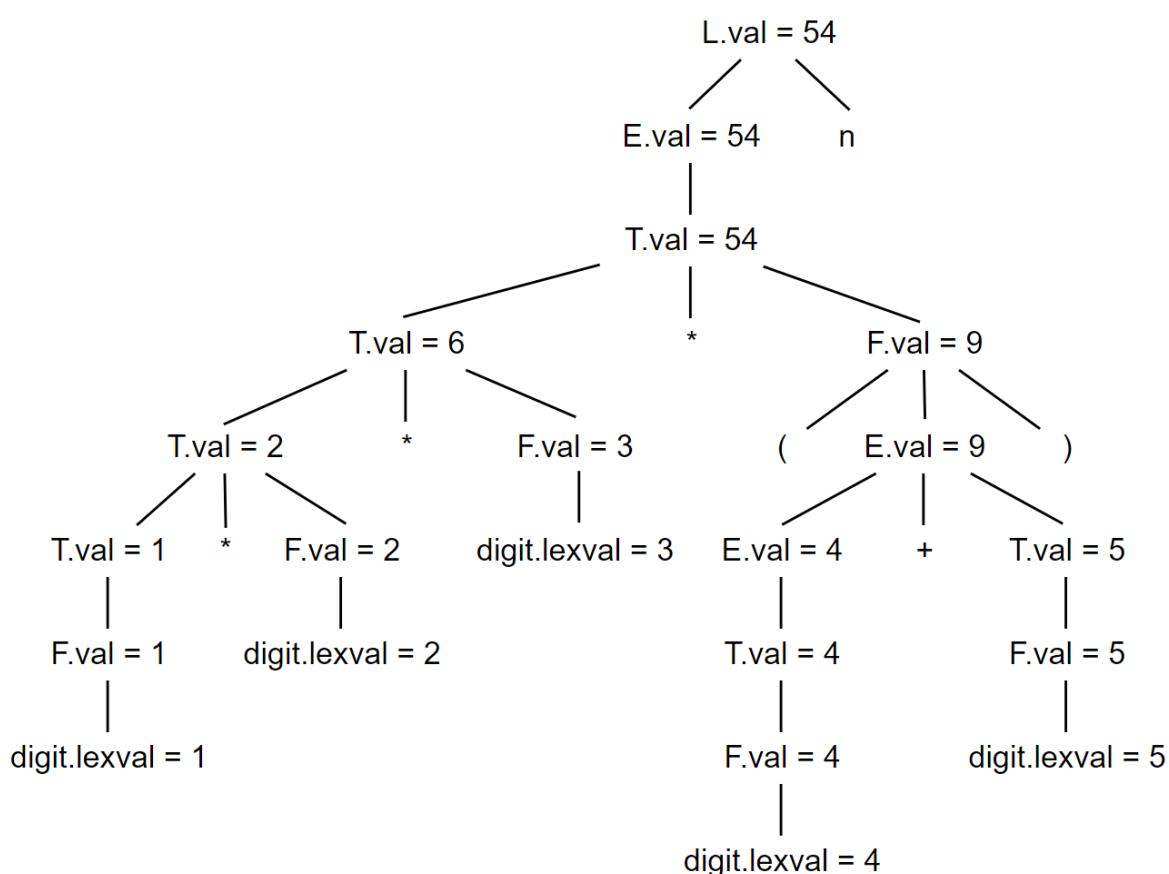
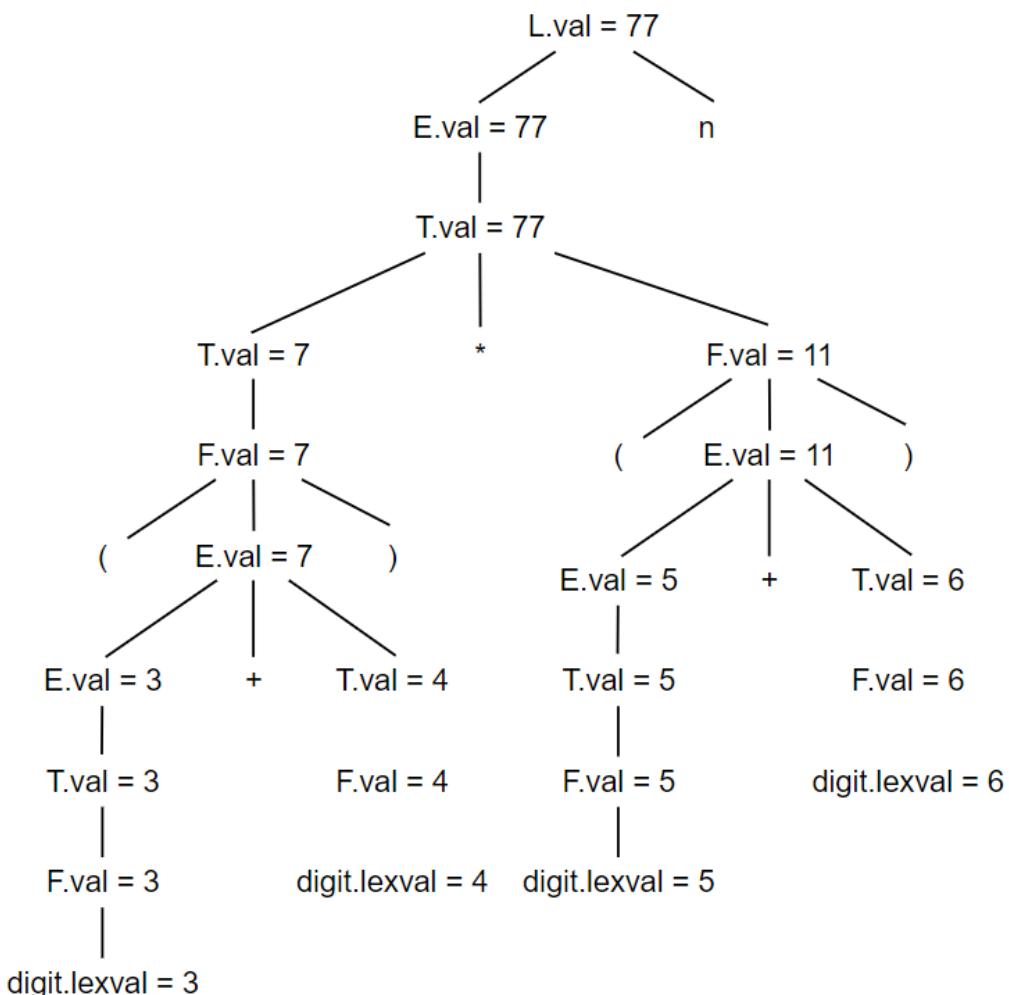
Exercise 1

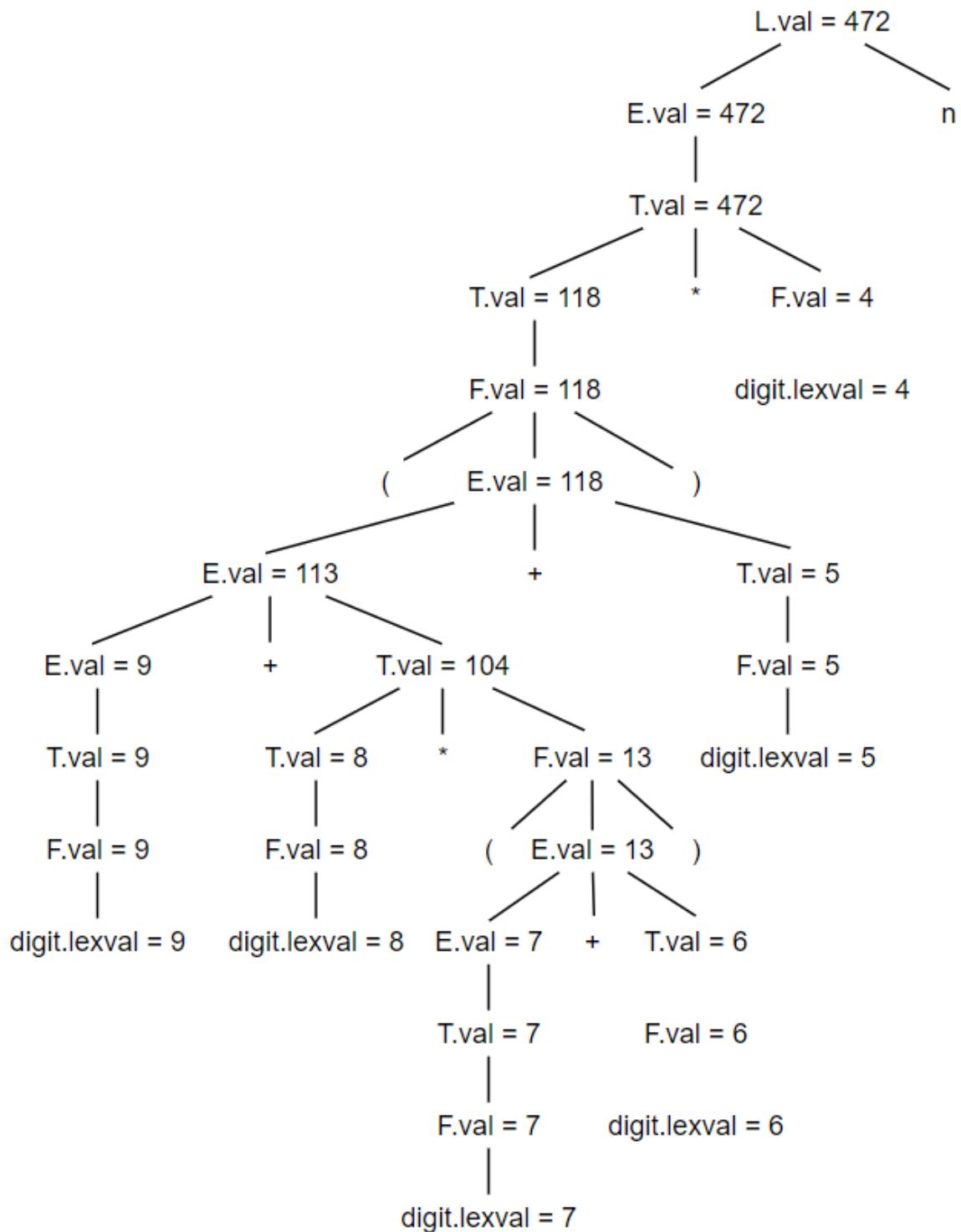
For the SDD in Figure 1, give annotated parse trees for the following expressions:

1. $(3 + 4) * (5 + 6)n$ [20 points]
2. $1 * 2 * 3 * (4 + 5)n$ [20 points]
3. $(9 + 8 * (7 + 6) + 5) * 4n$ [20 points]

PRODUCTION		SEMANTIC RULES
1)	$L \rightarrow E \ n$	$L.val = E.val$
2)	$E \rightarrow E_1 \ + \ T$	$E.val = E_1.val + T.val$
3)	$E \rightarrow T$	$E.val = T.val$
4)	$T \rightarrow T_1 \ * \ F$	$T.val = T_1.val \times F.val$
5)	$T \rightarrow F$	$T.val = F.val$
6)	$F \rightarrow (\ E \)$	$F.val = E.val$
7)	$F \rightarrow \text{digit}$	$F.val = \text{digit}.lexval$

Figure 1: Syntax-directed definition of a simple desk calculator





Exercise 2

What are all the topological sorts for the dependency graph of Figure 2?

One sort mentioned during lecture is 1, 2, 3, ..., 9. [20 points]

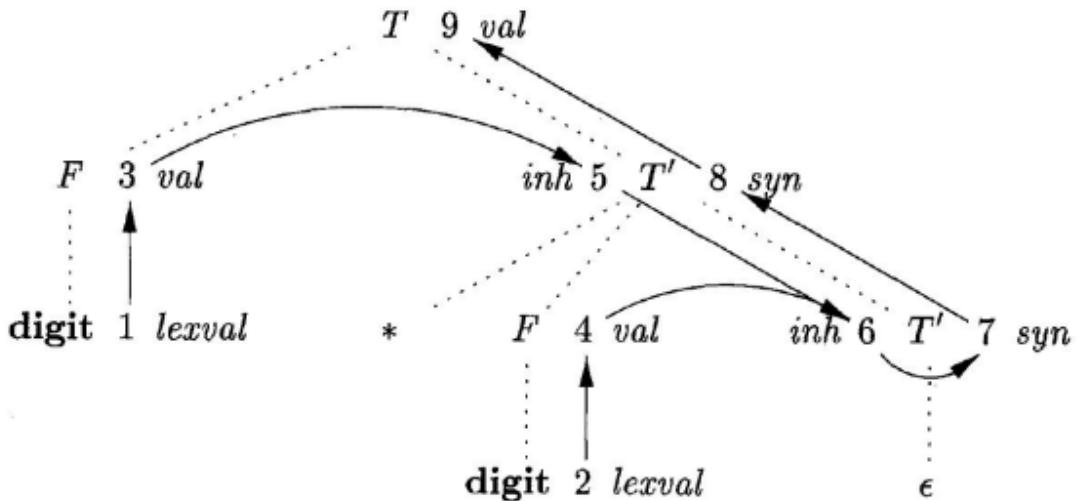


Figure 2: A dependency graph

图2中涉及的文法的产生式和语义规则为

PRODUCTION	SEMANTIC RULES
1) $T \rightarrow F T'$	$T'.inh = F.val$ $T.val = T'.syn$
2) $T' \rightarrow * F T'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$
3) $T' \rightarrow \epsilon$	$T'.syn = T'.inh$
4) $F \rightarrow \text{digit}$	$F.val = \text{digit}.lexval$

经过分析，我们可以得到属性计算的先后顺序如下，(1, 3)表示1必须于3之前。

```
(1, 3)
(2, 4)
(3, 5)
(5, 6)
(4, 6)
(6, 7)
(7, 8)
(8, 9)
```

为了找出1-9的所有符合以上要求的排列组合，我们编写了一个python程序如下：

```
import itertools

# 给定列表
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]

#给定先后顺序
list_order = [(1, 3),
               (2, 4),
               (3, 5),
               (5, 6),
               (4, 6),
               (6, 7),
               (7, 8),
               (8, 9)]
```

```

        (2, 4),
        (3, 5),
        (5, 6),
        (4, 6),
        (6, 7),
        (7, 8),
        (8, 9),
    ]
}

# 使用permutations生成所有排列顺序
permutations = itertools.permutations(my_list)

# 将排列结果转换为列表
permutations_list = list(permutations)

# 打印所有排列顺序
for perm in permutations_list:
    print_flag = True
    for order in list_order:
        former = order[0]
        latter = order[1]
        if perm.index(former) > perm.index(latter):
            print_flag = False
            break
    if print_flag:
        print(perm)

```

输出结果为：

```

(1, 2, 3, 4, 5, 6, 7, 8, 9)
(1, 2, 3, 5, 4, 6, 7, 8, 9)
(1, 2, 4, 3, 5, 6, 7, 8, 9)
(1, 3, 2, 4, 5, 6, 7, 8, 9)
(1, 3, 2, 5, 4, 6, 7, 8, 9)
(1, 3, 5, 2, 4, 6, 7, 8, 9)
(2, 1, 3, 4, 5, 6, 7, 8, 9)
(2, 1, 3, 5, 4, 6, 7, 8, 9)
(2, 1, 4, 3, 5, 6, 7, 8, 9)
(2, 4, 1, 3, 5, 6, 7, 8, 9)

```

Exercise 3

Below is a grammar for expressions involving operator + and integer or floating-point operands. Floating-point numbers are distinguished by having a decimal point. Give an SDD to determine the type of each term T and expression E. [20 points]

$$E \rightarrow E + T | T$$

$$T \rightarrow \text{num} \cdot \text{num} | \text{num}$$

PRODUCTION	SEMANTIC RULES
1) $E \rightarrow E + T$	$E.type = (E.type == float?float : T.type)$
2) $E \rightarrow T$	$E.type = T.type$
4) $T \rightarrow num. num$	$T.type = float$
4) $T \rightarrow num$	$T.type = int$