编译原理第五章第二次作业

李昊宸 2017K8009929044

5.3.1下面是涉及运算符 + 和整数或浮点运算分量的表达式的文法。区分浮点数的方法是看它有无小数点。

E -> E + T | T

T -> num.num | num

1）给出一个 SDD 来确定每个项 T 和表达式 E 的类型

2）扩展这个得到的 SDD，使得它可以把表达式转换成为后缀表达式。使用一个单目运算符 intToFloat 把一个整数转换为相等的浮点数。

答：

1）

|  |  |
| --- | --- |
| PRODUCTION | SEMANTIC RULES |
| E -> E1 + T | E.type = (E1.type == float) || (T.type == float) ? float : int |
| E -> T | E.type = T.type |
| T -> num.num | T.type = float |
| T -> num | T.type = int |

2)

|  |  |
| --- | --- |
| PRODUCTION | SEMANTIC RULES |
| S -> E | print(E.code) |
| E -> E1 + T | if( (E1.type == float) || (T.type == float) )  {  E.type = float  if(E1.type == int)  {  E1.type = float  E1.val =intToFloat(E1.val)  E1.code = E1.val  }  if(T.type == int)  {  T.type = float  T.val =intToFloat(T.val)  T.code = T.val  }  E.code = E1.code||T.code|| “+” |
| E -> T | E.type = T.type  E.val = T.val  E.code = T.code |
| T -> num.num | T.type = float  T.val = num.num.lexval  T.code = num.num.lexval |
| T -> num | T.type = int  T.val = num.lexval  T.code = num.lexval |

5.4.2改写下面的 SDT:

A -> A {a} B | A B {b} | 0

B -> B {c} A | B A {d} | 1

使得基础文法变成非左递归的。其中a、b、c和d是语义动作，0和1是终结符。

答：

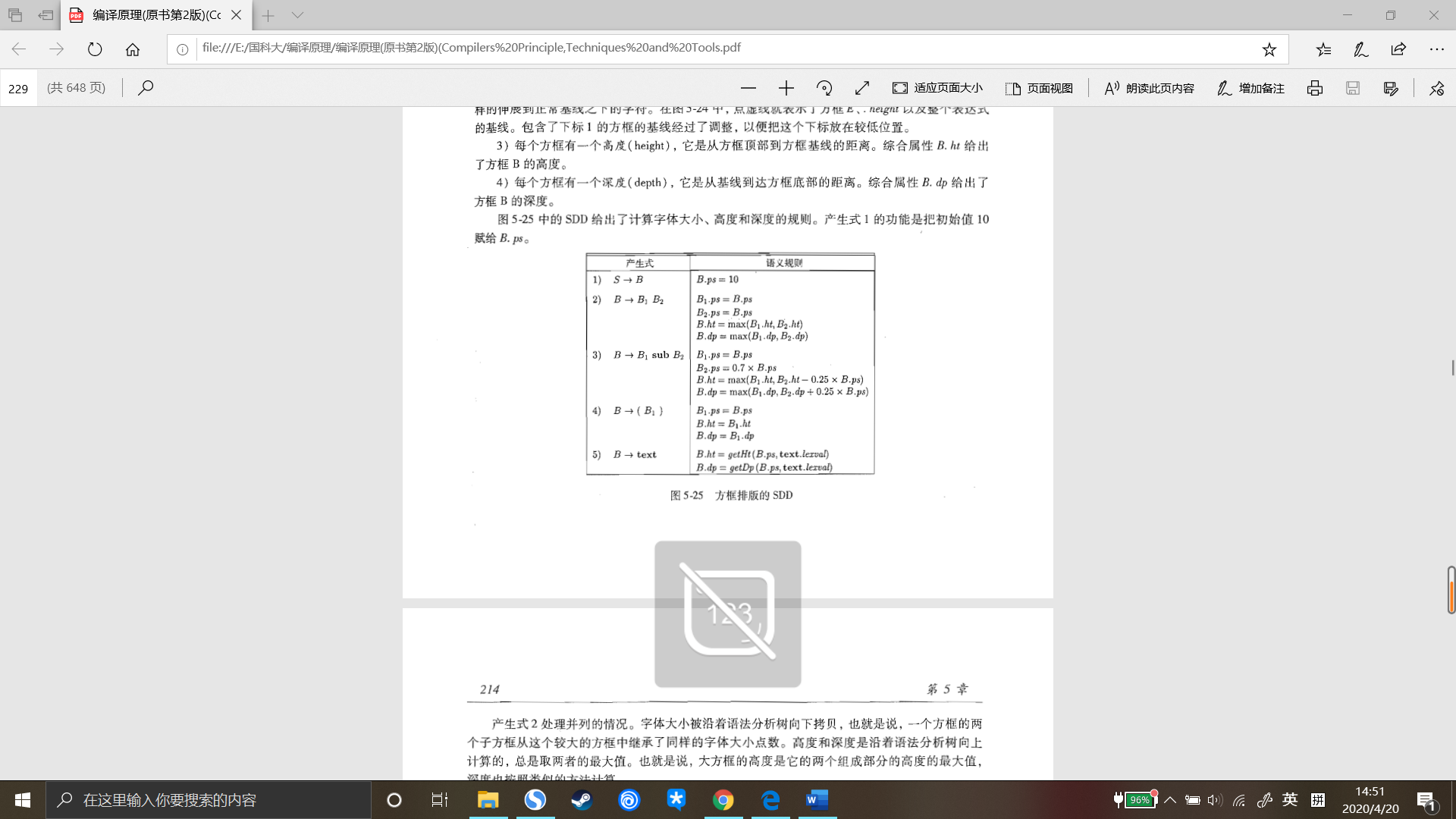
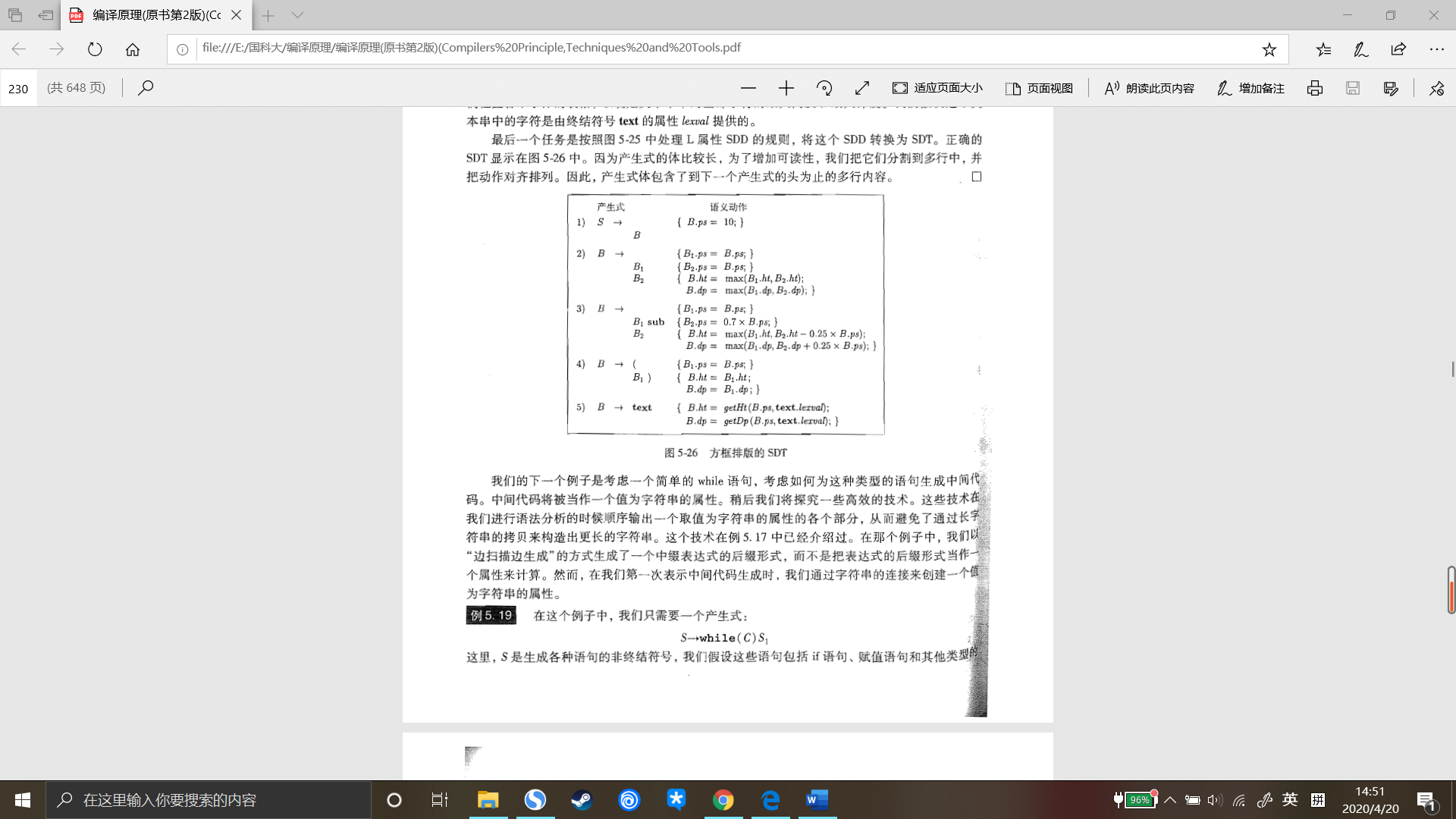
A -> 0 C

C -> {a} B C | B {b} C | ε

B -> 1 D

D -> {c} A D | A {d} D | ε

5.4.6修改图 5.25 中的 SDD，使它包含一个综合属性 B.le，即一个方框的长度。两个方框并列后得到的方框的长度是这两个方框的长度和。然后把你的新规则加入到图 5.26 中 SDT 的合适位置上。



答：

新增加的规则：

|  |  |
| --- | --- |
| PRODUCTION | SEMANTIC RULES |
| S -> B |  |
| B -> B1 B2 | B.le = B1.le + B2.le |
| B -> B1 sub B2 | B.le = B1.le + B2.le |
| B -> ( B1 ) | B.le = B1.le |
| B -> text | B.le = getLe(B.ps , text.lexval) |

对于每一个产生式新增加的规则，由于全部都是计算B的综合属性le，故新增加的规则在SDT表中添加到每个产生式对应的语义动作的最末端即可（也就是跟在计算B.dp的后面）。