# Week 9

Xiaoyuan Xie 谢晓园
xxie@whu.edu.cn
计算机学院E301

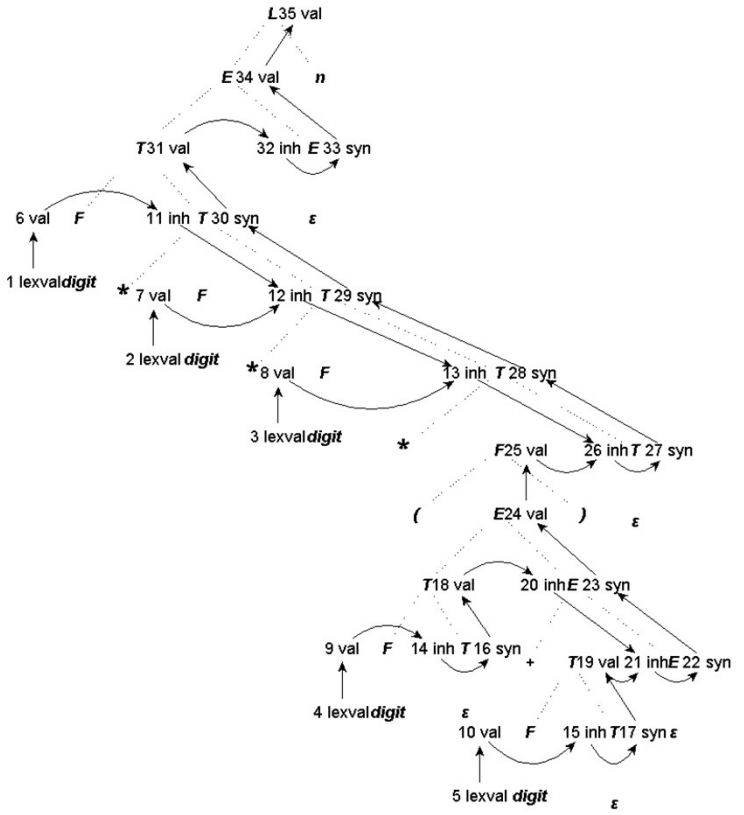
# Week 9 作业

- **PPT：** SDD如下表所示，为表达式 **1 * 2 * 3 * (4 + 5)n** 画出注释语法树和依赖图

| 产生式 | 语法规则 |
| --- | --- |
| L -> En | L.val = E.val |
| E -> TE' | E'.inh = T.val<br>E.val = E'.syn |
| E' -> +$TE_1$' | $E_1$'.inh = E'.inh + T.val<br>E'.syn = $E_1$'.syn |
| E' -> ε | E'.syn = E'.inh |
| T -> FT' | T'.inh = F.val<br>T.val = T'.syn |
| T' -> *$FT_1$' | $T_1$'.inh = T'.inh * F.val<br>T'.syn = $T_1$'.syn |
| T' -> ε | T'.syn = T'.inh |
| F -> (E) | F.val = E.val |
| F -> digit | F.val = digit.lexval |

# Week 9 作业

- 左) 注释语法树
- 右) 依赖图

# Week 9 作业

- **教材P203 5.2.4** 这个文法生成了含“小数点”的二进制数

**S -> L.L|L　　L -> LB|B　　B -> 0|1**

设计一个L属性的SDD来计算S.val，即输入串的十进制数。

其中：

- isLeft 继承属性，表示是否在小数点左边
- len 综合属性，表示二进制串的长度
- val 综合属性

| 产生式 | 语法规则 |
|---|---|
| S -> L_1.L_2 | L_1.isLeft = **true**<br>L_2.isLeft = **false**<br>S.val = L_1.val + L_2.val |
| S -> L | L.isLeft = **true**<br>S.val = L.val |
| L -> L_1B | L_1.isLeft = L.isLeft<br>L.len = L_1.len + 1<br>L.val = L.isLeft **?** (L_1.val * 2 + B.val) **:** (L_1.val + B.val * 2^(-L.len)) |
| L -> B | L.len = 1<br>L.val = L.isLeft **?** B.val **:** B.val/2 |
| B -> 0 | B.val = 0 |
| B -> 1 | B.val = 1 |

# Week 9 作业

- **教材P207 5.3.1** 下面是涉及运算符+和整数或浮点数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。

| 产生式 | 语法规则 |
| --- | --- |
| E -> E_1 + T | (E_1.type ==T.type) **?** (E.type = E_1.type) **:** (E.type = float) |
| E -> T | E.type = T.type |
| T -> num.num | T.type = float |
| T -> num | T.type = int |

# Week 9 作业

- **教材P207 5.3.1** 下面是涉及运算符+和整数或浮点数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**

  - 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。

| 产生式 | 语法规则 |
|---|---|
| E -> E_1 + T | (E_1.type ==T.type) **?** (E.type = E_1.type) **:** (E.type = float)<br>**if** (E_1.type == int **&&** T.type == float) **then** E_1 = **intToFloat**(E_1)<br>**else if** (T.type == int **&&** E_1.type == float) **then** T = **intToFloat**(T)<br>E.val = (E_1.val , T.val , +) |
| E -> T | E.type = T.type<br>E.val = T.val |
| T -> num.num | T.type = float<br>T.val = num.num |
| T -> num | T.type = int<br>T.val = num |

# Week 9 作业

- **教材P207 5.3.2** 给出一个SDD，将一个带有+和*的中缀表达式翻译成没有冗余括号的表达式。比如因为两个运算符都是左结合的，并且 * 的优先级高于 +，所以 ((a*(b+c))*(d)) 可翻译为 a*(b+c)*d
  - 设计产生式如下：

    L -> En

    E -> E + T | T

    T -> T * F | F

    F -> (E) | digit

# Week 9 作业

- 1) 方法一：把表达式全部括号去掉，然后在必要的地方加上括号。

| 产生式 | 语法规则 |
|---|---|
| L -> En | L.expr = E.expr |
| E -> E_1 + T | T.op == plus **?** E.expr = E_1.expr “+” “(” T.expr “)” **:** E.expr = E_1.expr “+” T.expr<br>E.op = plus |
| E -> T | E.expr = T.expr<br>E.op = T.op |
| T -> T_1 * F | **if** F.op == plus **||** F.op == times **then**<br>  **if** T.op == plus **then** T.expr = “(” T_1.expr “)” “*” “(” F.expr “)”<br>  **else** T.expr = T_1.expr “*” “(” F.expr “)”<br>**else if** T.op == plus **then** T.expr = “(” T_1.expr “)” “*” F.expr<br>**else** T.expr = T_1.expr “*” F.expr<br>T.op = times |
| T -> F | T.expr = F.expr<br>T.op = F.op |
| F -> (E) | F.expr = E.expr<br>F.op = E.op |
| F -> digit | F.expr = digit.lexval<br>F.op = id |

其中的属性和值：

expr：表达式

op：运算操作

plus：表示加

times：表示乘

id：表示digit

# Week 9 作业

- 2) 方法二：在叶子节点判断是否需要去掉冗余括号，保留必要括号。

| 产生式 | 语法规则 |
| --- | --- |
| L -> En | E.left_op = 0　　E.right_op = 0<br>L.expr = E.expr |
| E -> E_1 + T | E_1.left_op = E.left_op　　E_1.right_op = 1<br>T.left_op = 1　　T.right_op = E.right_op<br>E.self_op = 1<br>E.expr = E_1.expr “+” T.expr |
| E -> T | T.left_op = E.left_op　　T.right_op = E.right_op<br>E.self_op = T.self_op<br>E.expr = T.expr |
| T -> T_1 * F | T_1.left_op = T.left_op　　T_1.right_op = 2<br>F.left_op = 2　　F.right_op = T.right_op<br>T.self_op = 2<br>T.expr = T_1.expr “*” F.expr |
| T -> F | F.left_op = T.left_op　　F.right_op = T.left_op<br>T_self_op = F.self_op<br>T.expr = F.expr |
| F -> (E) | E.left_op = 0　　E.right_op = 0<br>F.self_op = (F.left_op < E.self_op) **&&** (E.self_op >= F.right_op) **?** E.self_op **:** 3<br>F.expr = (F.left_op < E.self_op) **&&** (E.self_op >= F.right_op) **?** E.expr **:** “(” E.expr “)” |
| F -> digit | F.expr = digit.lexval<br>F.self_op = 3 |

其中几个属性和值：

expr：表达式

left_op：左侧算符优先级

right_op：右侧算符优先级

self_op：自身算符优先级

0：表示左右无运算对象

1、2：分别表示加、乘

3：表示括号或者digit

# Thank you!