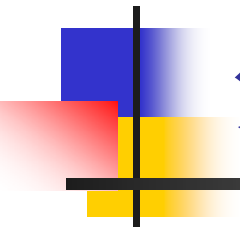


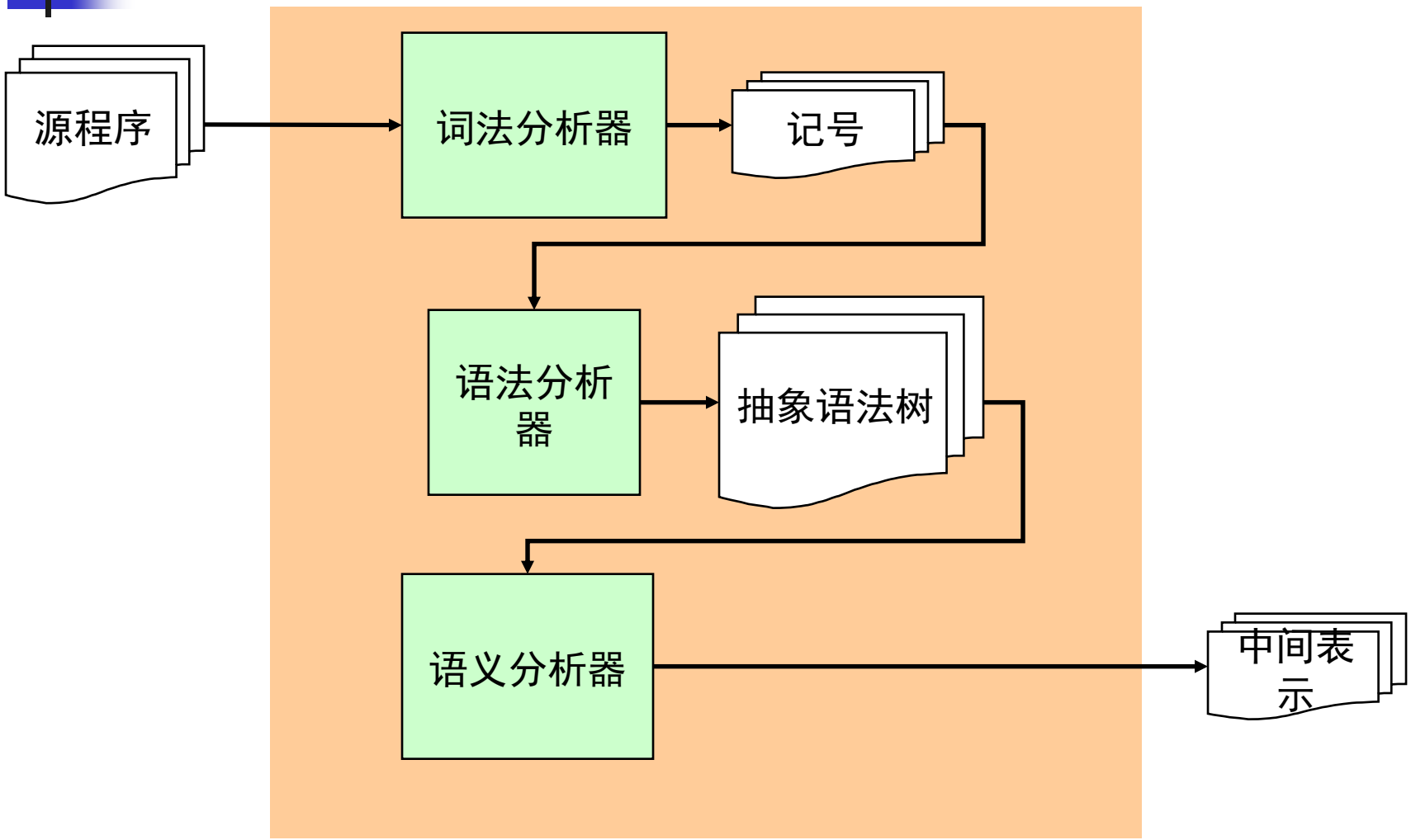
语法分析：分析树与二义性

编译原理

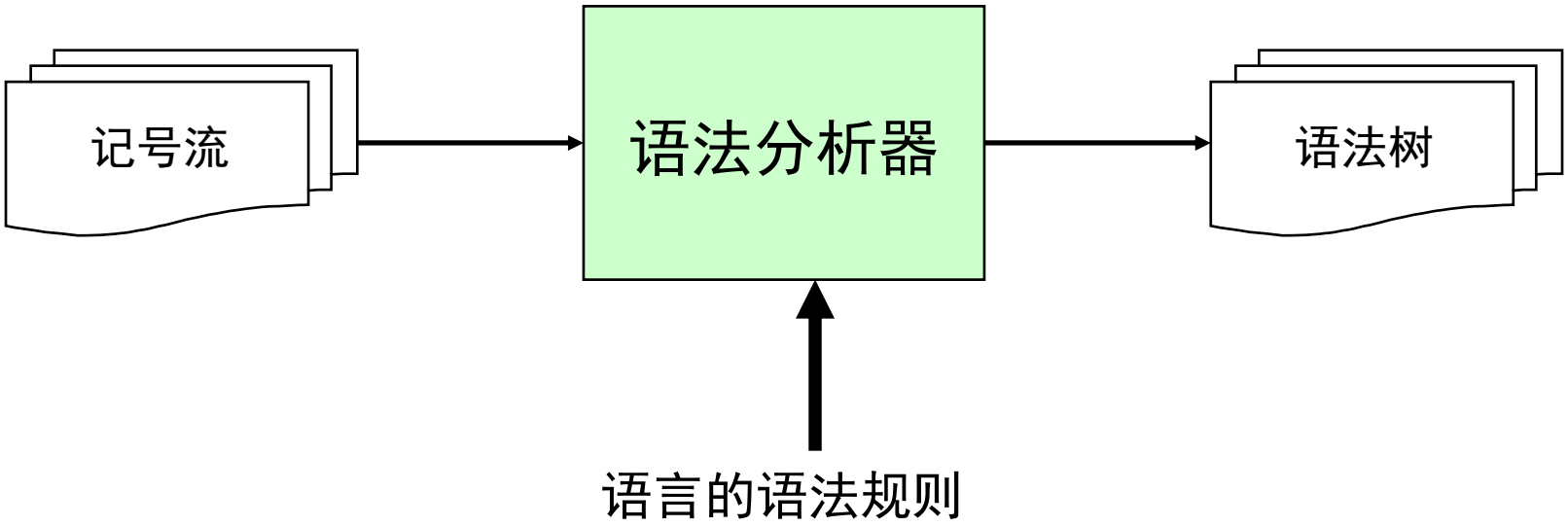
华保健

bjhua@ustc.edu.cn

前端

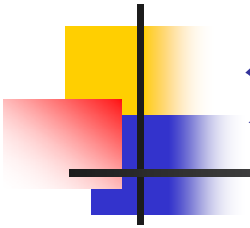


语法分析器的任务



推导与分析树

```
S -> N V N
N -> s
   | t
   | g
   | w
V -> e
   | d
```



分析树

- 推导可以表达成树状结构
  - 和推导所用的顺序无关（最左、最右、其他）
- 特点:
  - 树中的每个内部节点代表非终结符
  - 每个叶子节点代表终结符
  - 每一步推导代表如何从双亲节点生成它的直接孩子节点

表达式的例子

```
E -> num
   | id
   | E + E
   | E * E
```

```
E -> E + E
  -> 3 + E
  -> 3 + E * E
  -> 3 + 4 * E
  -> 3 + 4 * 5
E -> E * E
  -> E + E * E
  -> 3 + E * E
  -> 3 + 4 * E
  -> 3 + 4 * 5
```

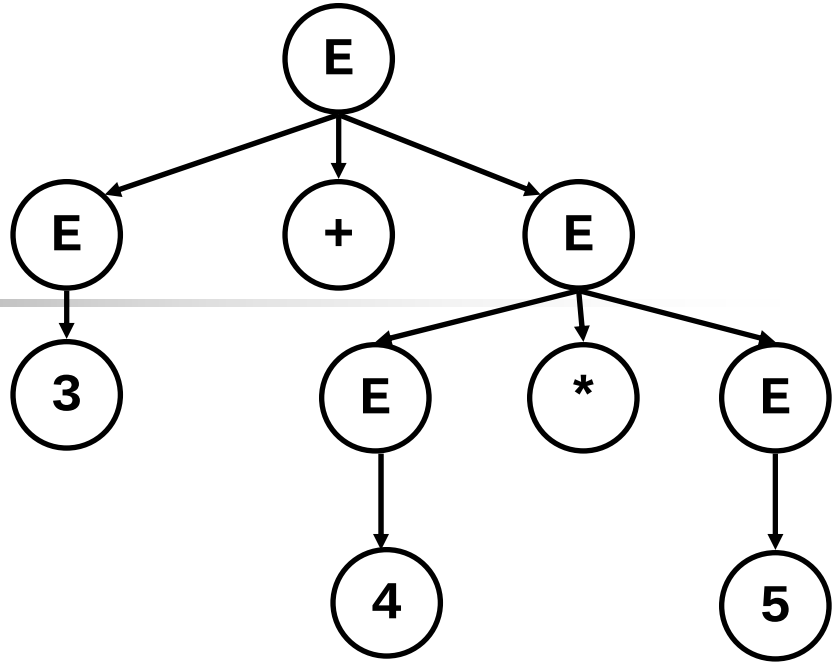
推导这个句子

3+4*5

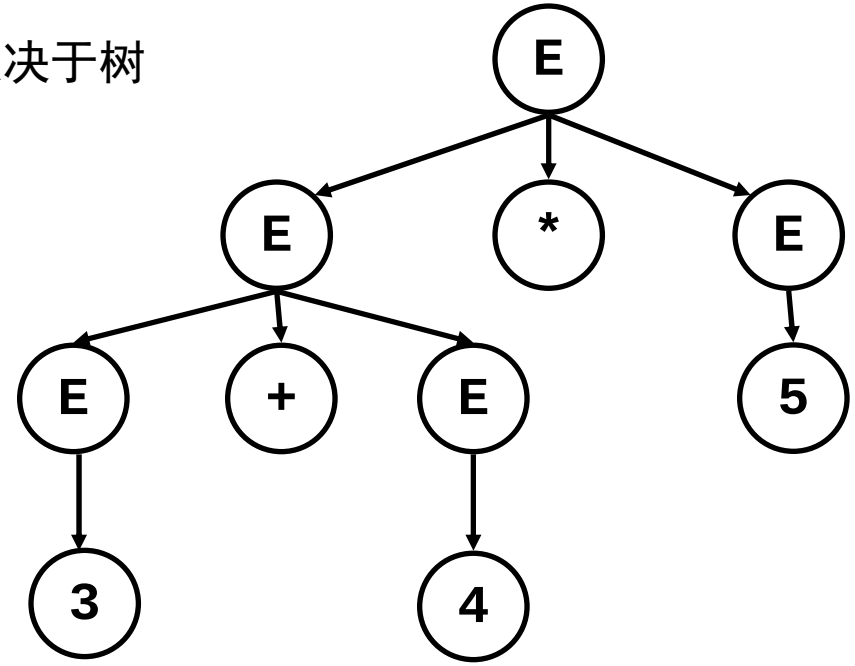
分析树

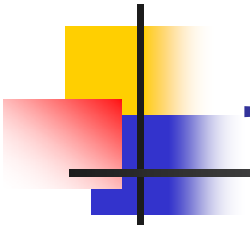
```
E -> num
   | id
   | E + E
   | E * E
```

```
E -> E + E
  -> 3 + E
  -> 3 + E * E
  -> 3 + 4 * E
  -> 3 + 4 * 5
E -> E * E
  -> E + E * E
  -> 3 + E * E
  -> 3 + 4 * E
  -> 3 + 4 * 5
```



分析树的含义取决于树的后序遍历。





二义性文法

- 给定文法G，如果存在句子s，它有两棵不同的分析树，那么称G是二义性文法
- 从编译器角度，二义性文法存在问题：
  - 同一个程序会有不同的含义
  - 因此程序运行的结果不是唯一的
- 解决方案：文法的重写

表达式文法的重写

```
E -> E + T
   | T
T -> T * F
   | F
F -> num
   | id
```

```
E -> E + T
  -> T + T
  -> F + T
  -> 3 + T
  -> 3 + T * F
  -> 3 + F * F
  -> 3 + 4 * F
  -> 3 + 4 * 5
```

推导这个句子

3+4*5

表达式文法的重写

```
E -> E + T
   | T
T -> T * F
   | F
F -> num
   | id
```

```
E -> E + T
  -> E + T + T
  -> T + T + T
  -> F + T + T
  -> 3 + T + T
  -> 3 + F + T
  -> 3 + 4 + T
  -> 3 + 4 + F
  -> 3 + 4 + 5
```

推导这个句子

3+4+5