



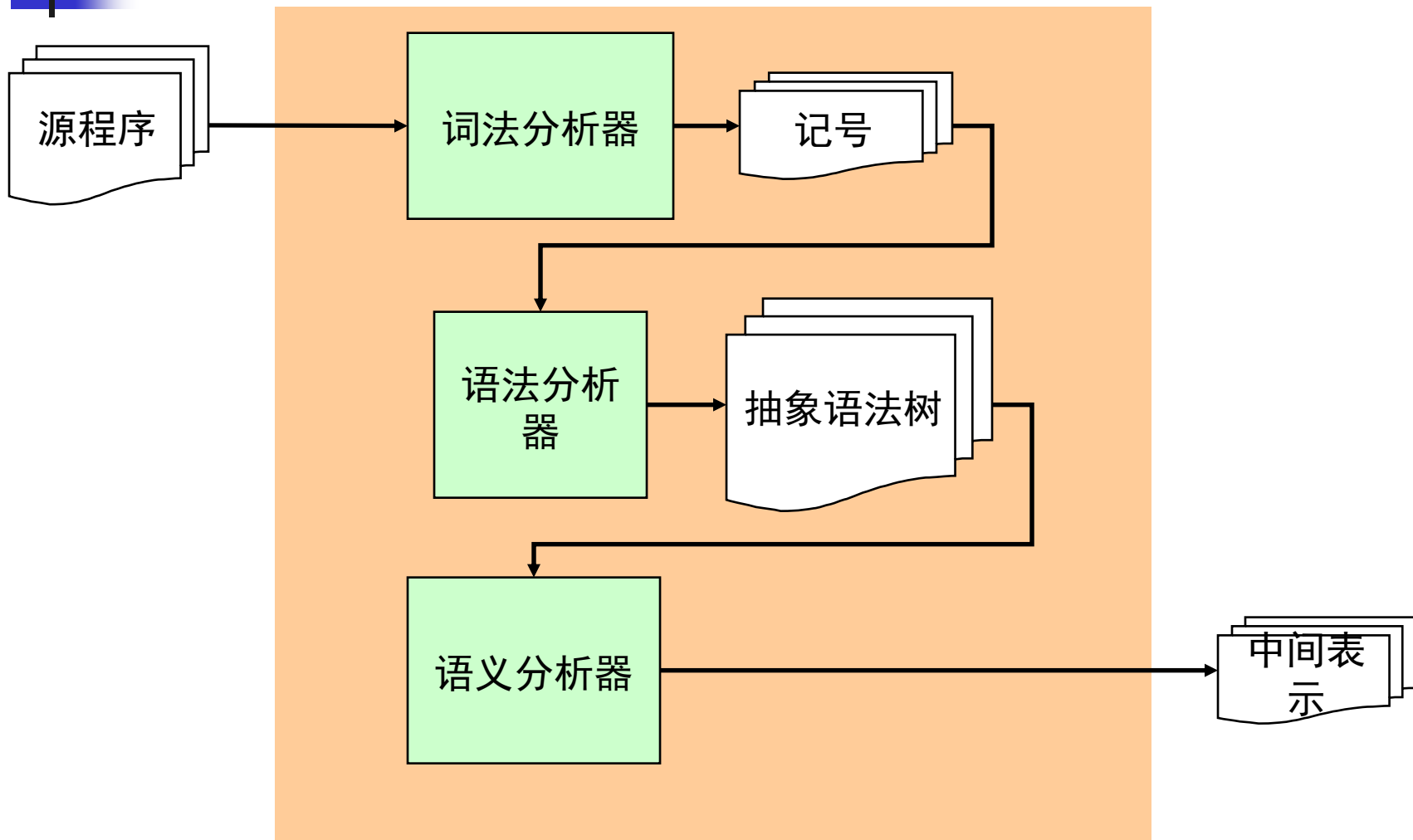
语法分析： 自顶向下分析

编译原理

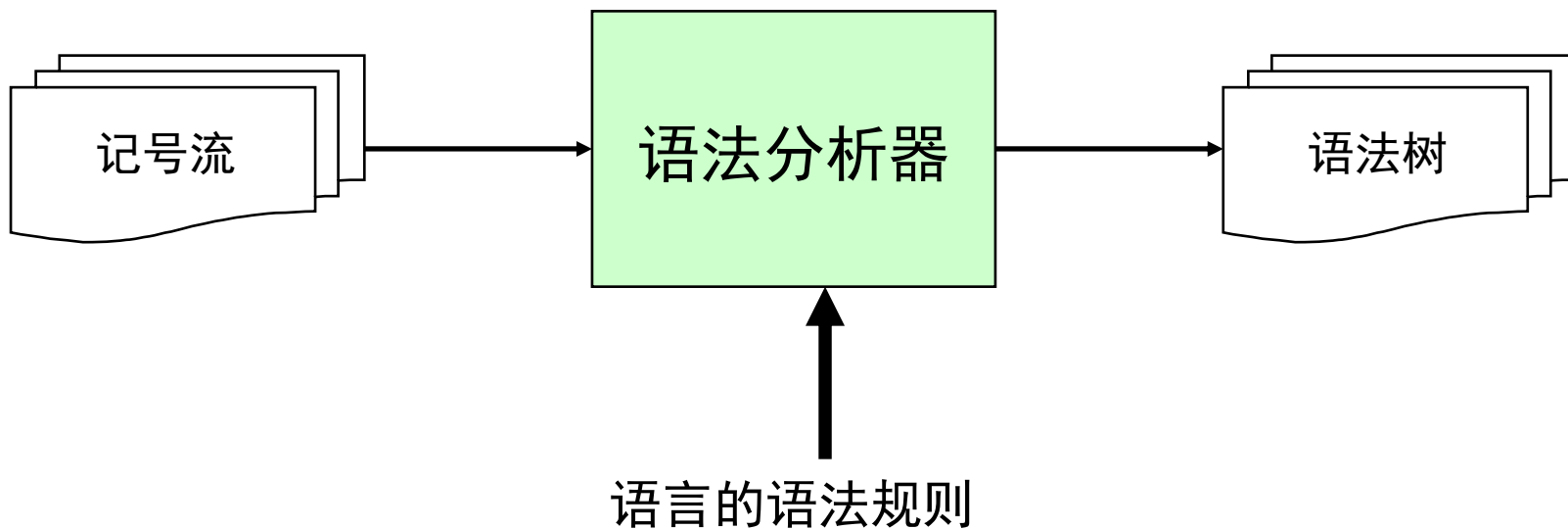
华保健

bjhua@ustc.edu.cn

前端



语法分析器的任务





自顶向下分析的算法思想

- 语法分析：给定文法 G 和句子 s ，回答 s 是否能够从 G 推导出来？
- 基本算法思想：从 G 的开始符号出发，随意推导出某个句子 t ，比较 t 和 s
 - 若 $t=s$ ，则回答“是”
 - 若 $t \neq s$ ，则？
- 因为这是从开始符号出发推出句子，因此称为自顶向下分析
 - 对应于分析树自顶向下的构造顺序



示例

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

推导这个句子

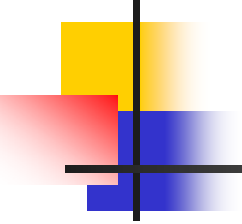


g d w



算法

```
tokens[];    // all tokens
i=0;
stack = [S]  // s是开始符号
while (stack != [])
    if (stack[top] is a terminal t)
        if (t==tokens[i++])
            pop();
        else backtrack();
    else if (stack[top] is a nonterminal T)
        pop(); push(the next right hand side of T)
```



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

```
tokens[];    // holding all tokens
i=0;
stack = [S]  // s是开始符号
while (stack != [])
    if (stack[top] is a terminal t)
        if (t==tokens[i++])
            pop();
        else backtrack();
    else if (stack[top] is a nonterminal T)
        pop(); push(the next right hand side of T)
```

推导这个句子



g d w



算法的讨论

- 算法需要用到回溯
 - 给分析效率带来问题
- 而就这部分而言（就所有部分），编译器必须高效
 - 编译上千万行的内核等程序
- 因此，实际上我们需要线性时间的算法
 - 避免回溯
 - 引出递归下降分析算法和LL(1)分析算法



重新思考示例

- 用前看符号避免回溯

S -> N V N

N -> s

| t

| g

| w

V -> e

| d

推导这个句子



g d w