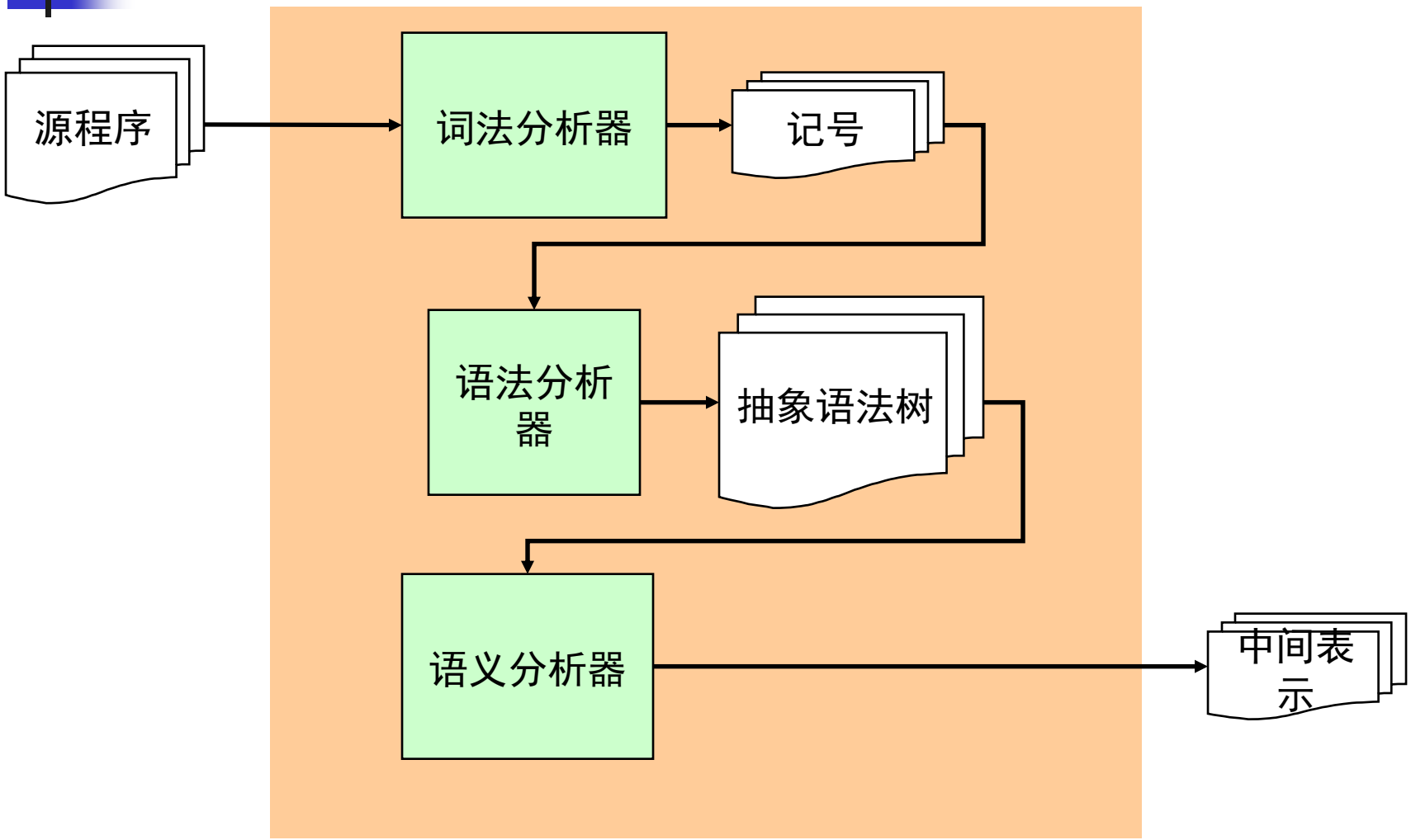
语法分析：递归下降分析

编译原理

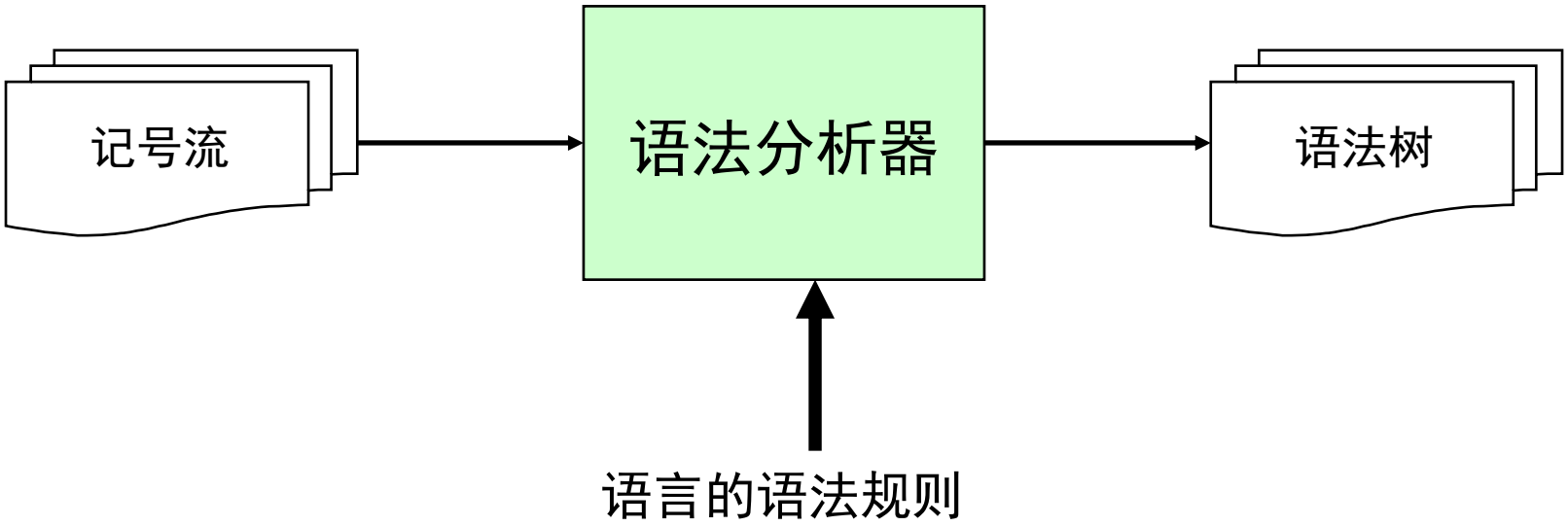
华保健

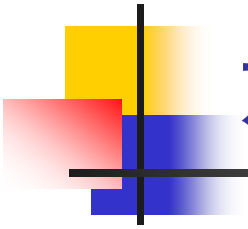
bjhua@ustc.edu.cn

前端



语法分析器的任务





递归下降分析算法

- 也称为预测分析
  - 分析高效（线性时间）
  - 容易实现（方便手工编码）
  - 错误定位和诊断信息准确
  - 被很多开源和商业的编译器所采用
    - GCC 4.0， LLVM，。。。
- 算法基本思想：
  - 每个非终结符构造一个分析函数
  - 用前看符号指导产生式规则的选择

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

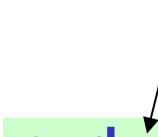
```
parse_S()
  parse_N()
  parse_V()
  parse_N()

parse_N()
  token = tokens[i++]
  if (token==s||token==t||
      token==g||token==w)
    return;
  error("…");

parse_V()
  token = tokens[i++]
  …// leave this part to you ☺
```

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

推导这个句子



```
g d w
```

一般的算法框架

```
X -> β11 … β1i
   | β21 … β2j
   | β31 … β3k
   | …
```

```
parse_X()
  token = nextToken()
  switch(token)
  case …: // β11 … β1i
  case …: // β21 … β2j
  case …: // β31 … β3k
  …
  default: error (“…”);
```

对算术表达式的递归下降分析

```
// a first try
parse_E()
  token = tokens[i++]
  if (token==num)
    ? // E+T or T
  else error("…");
```

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

对这个句子做语法分析

3+4*5

对算术表达式的递归下降分析

```
// a second try
parse_E()
  parse_T()
  token = tokens[i++]
  while (token == +)
    parse_T()
    token = tokens[i++]

parse_T()
  parse_F()
  token = tokens[i++]
  while (token == *)
    parse_F()
    token = tokens[i++]
```

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

对这个句子做语法分析