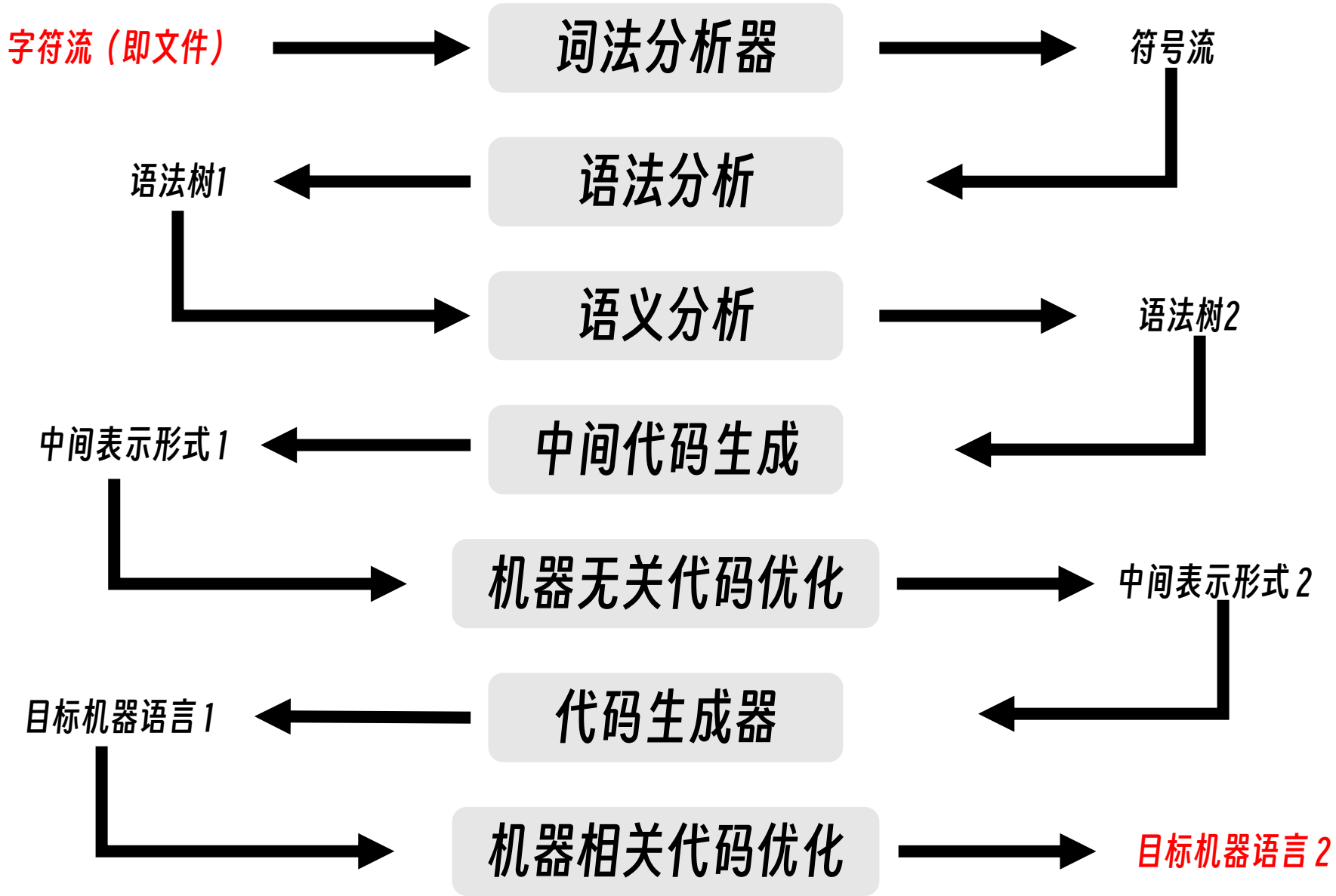
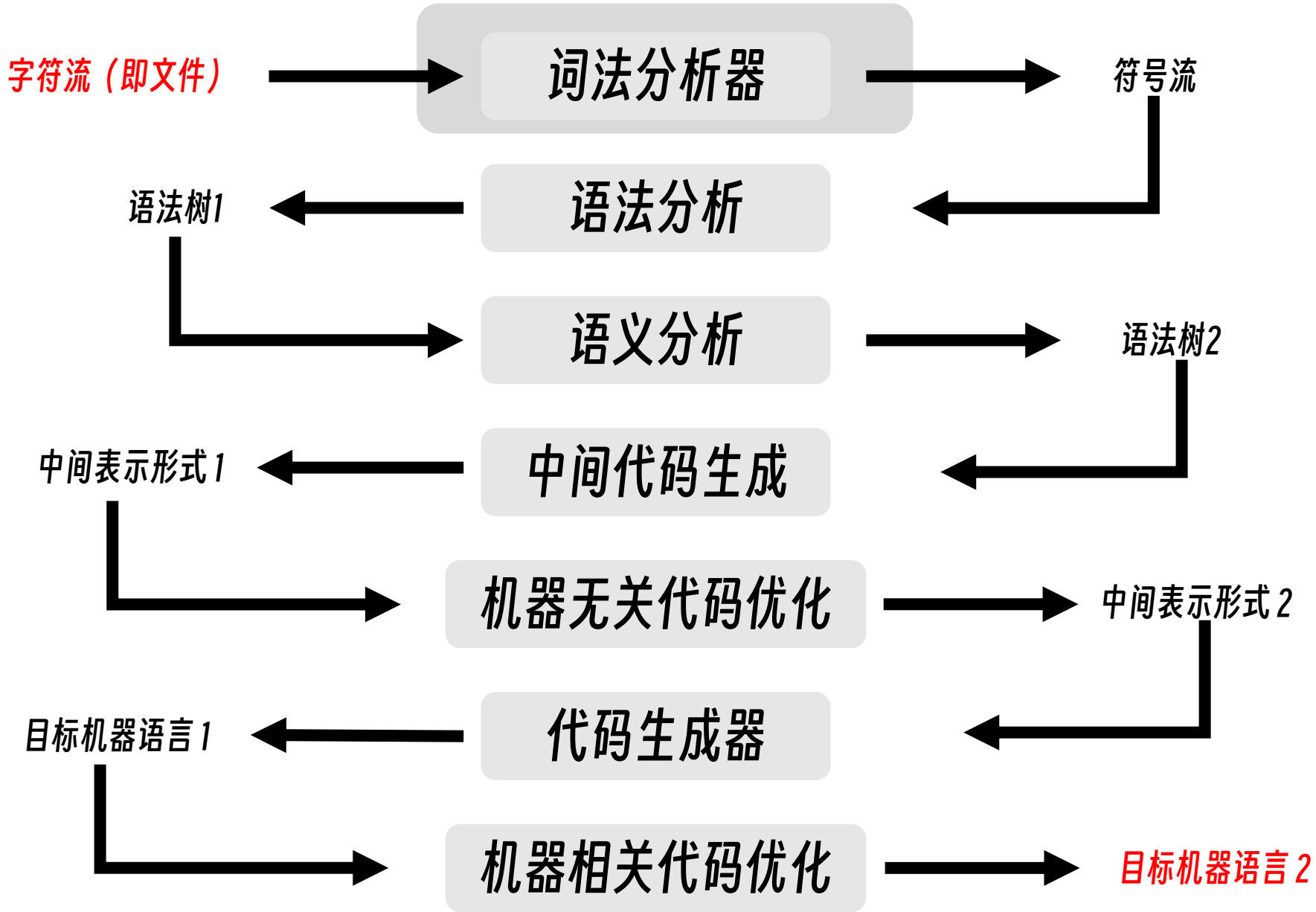
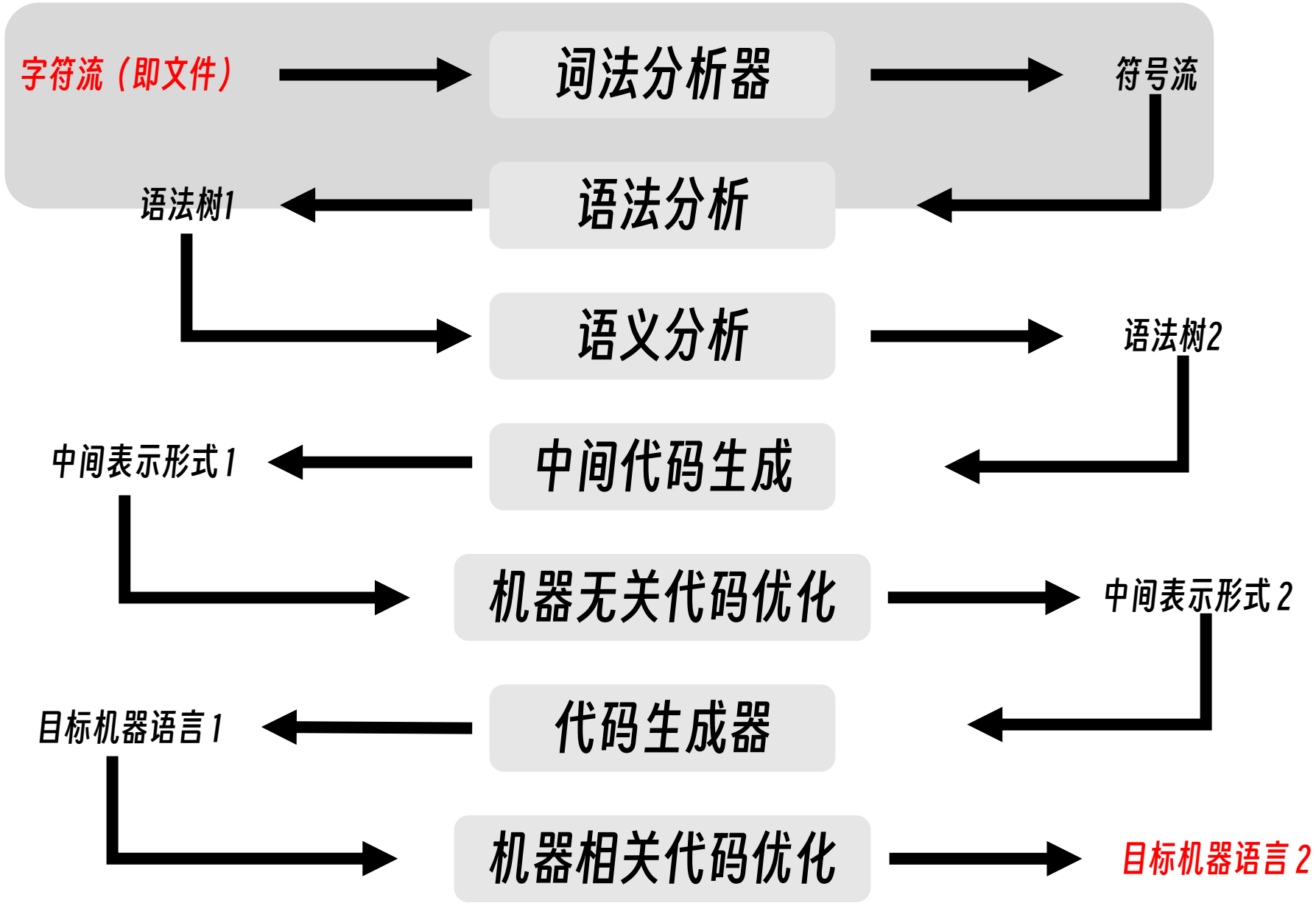
编译之词法分析

汇报人：皮昊旋


字符流（即文件）
词法分析器
符号流
语法树1
语法分析
语义分析
语法树2
中间表示形式1
中间代码生成
机器无关代码优化
中间表示形式2
目标机器语言1
代码生成器
机器相关代码优化
目标机器语言2


字符流（即文件）
词法分析器
符号流
语法分析
语法树1
语义分析
语法树2
中间代码生成
中间表示形式1
机器无关代码优化
中间表示形式2
代码生成器
目标机器语言1
机器相关代码优化
目标机器语言2

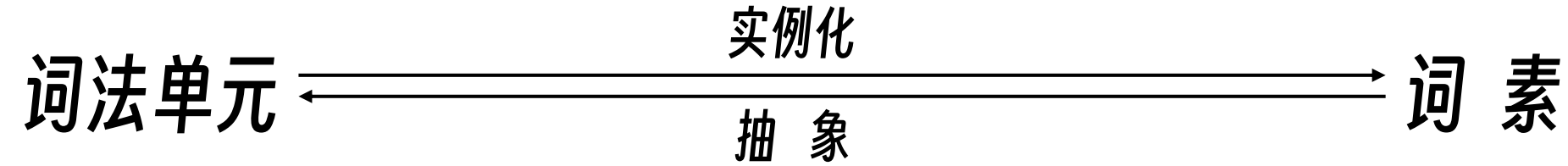

字符流（即文件）
词法分析器
符号流
语法分析
语法树1
语义分析
语法树2
中间代码生成
中间表示形式1
机器无关代码优化
中间表示形式2
代码生成器
目标机器语言1
机器相关代码优化
目标机器语言2

<id> ---------- xiao、systemInfo

<number> ---------- 1、2、3.14、-5.87

<comparison> ---------- !=、>=、<=

<if> ---------- if



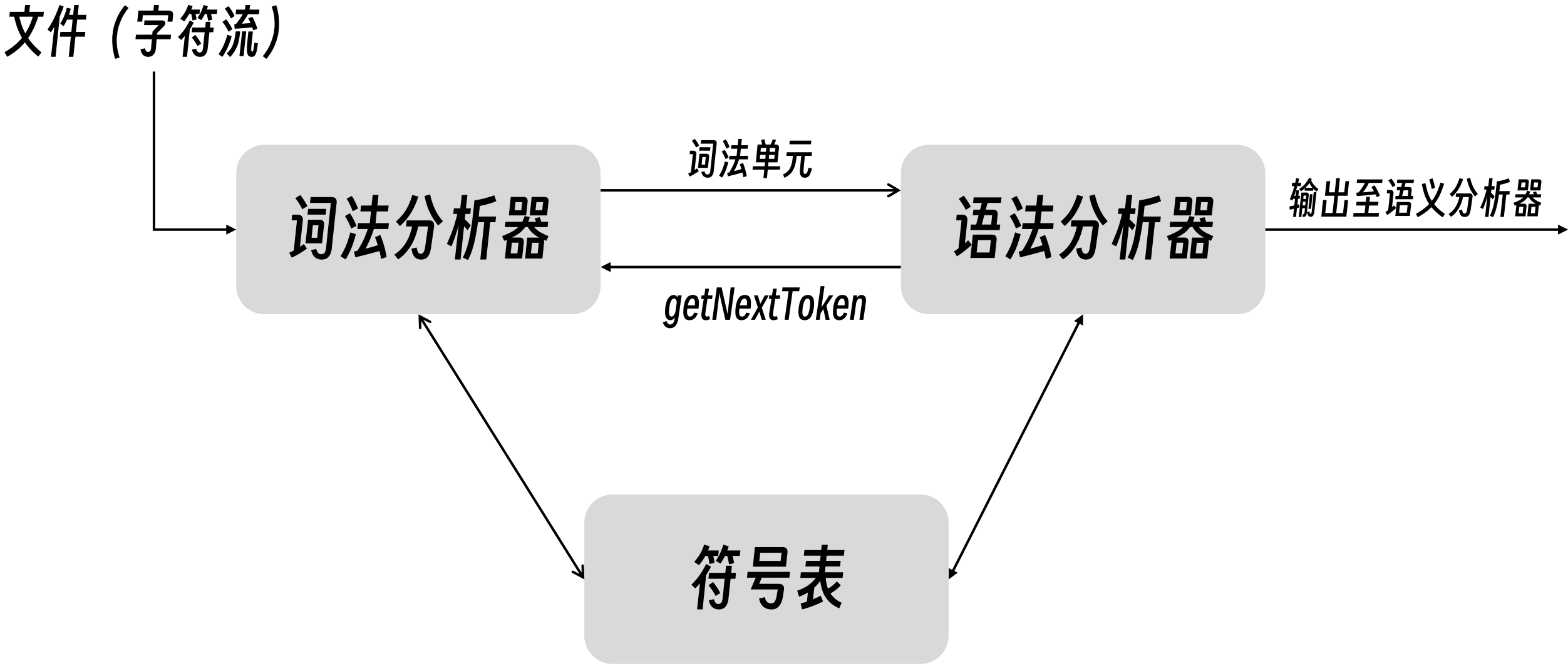
<id, p1> ---------- xiao

<id, p2> ---------- systemInfo

<number, static p0> ---------- 0

<assign_op> ---------- =

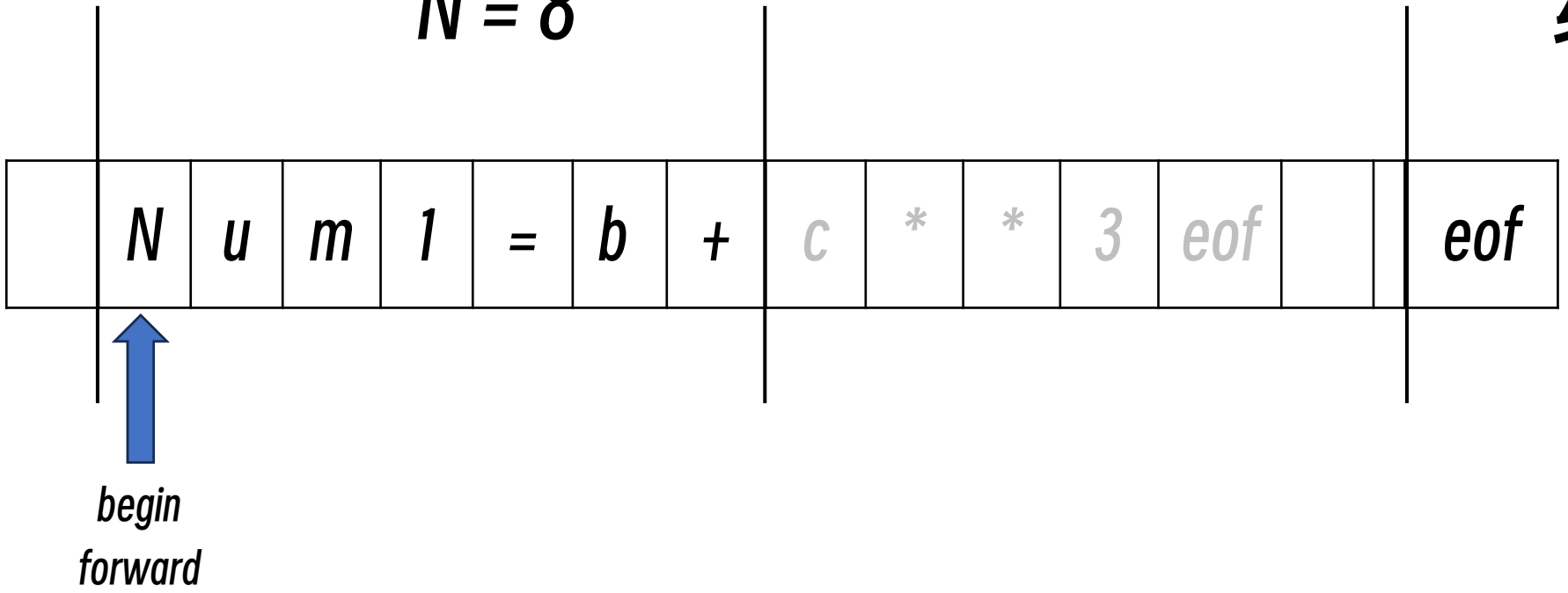
<if> ---------- if


文件（字符流）
词法分析器
词法单元
语法分析器
输出至语义分析器
getNextToken
符号表

读取输入

N = 8

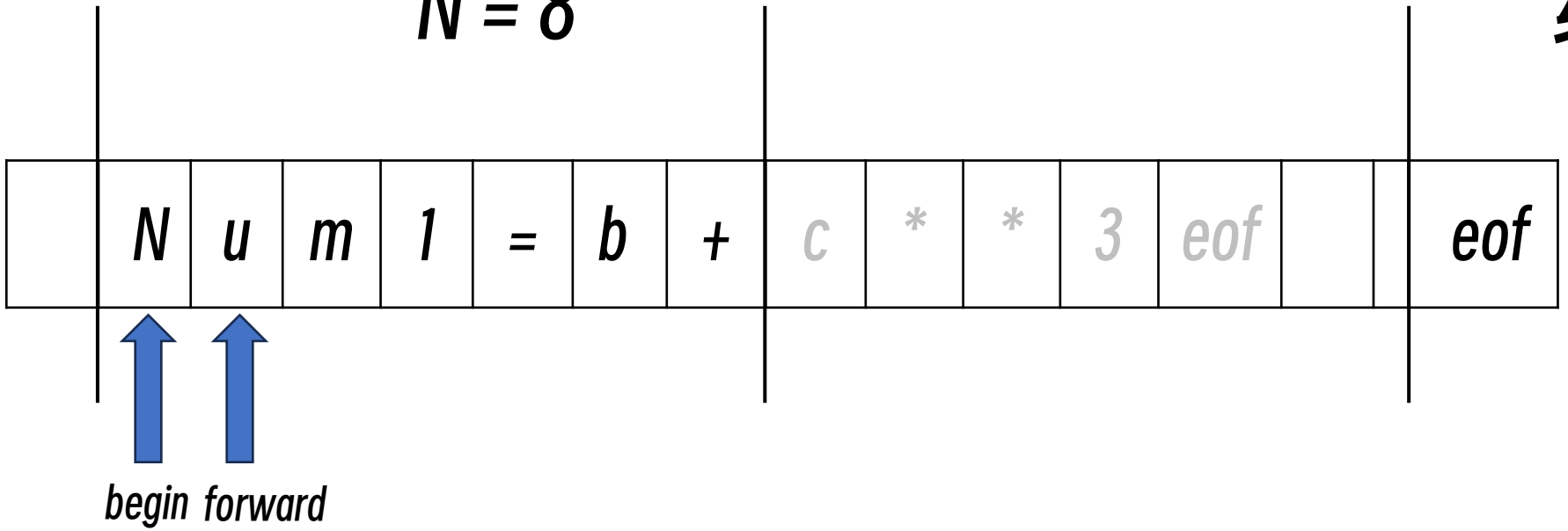
缓冲区对



读取输入

N = 8

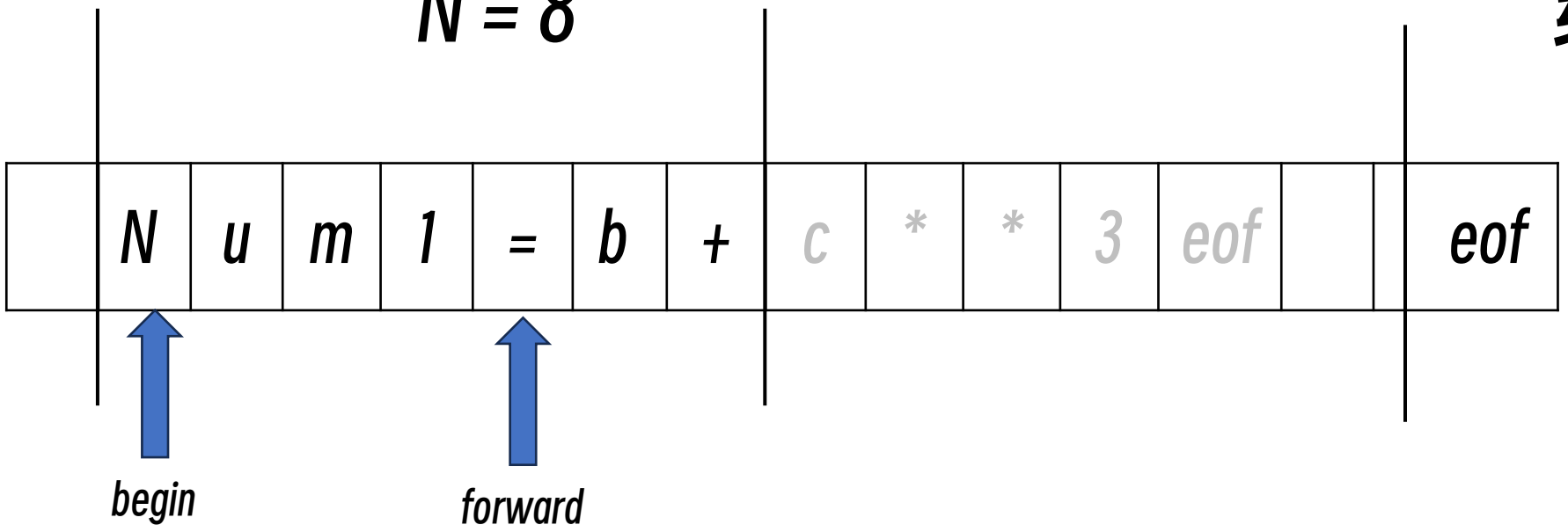
缓冲区对



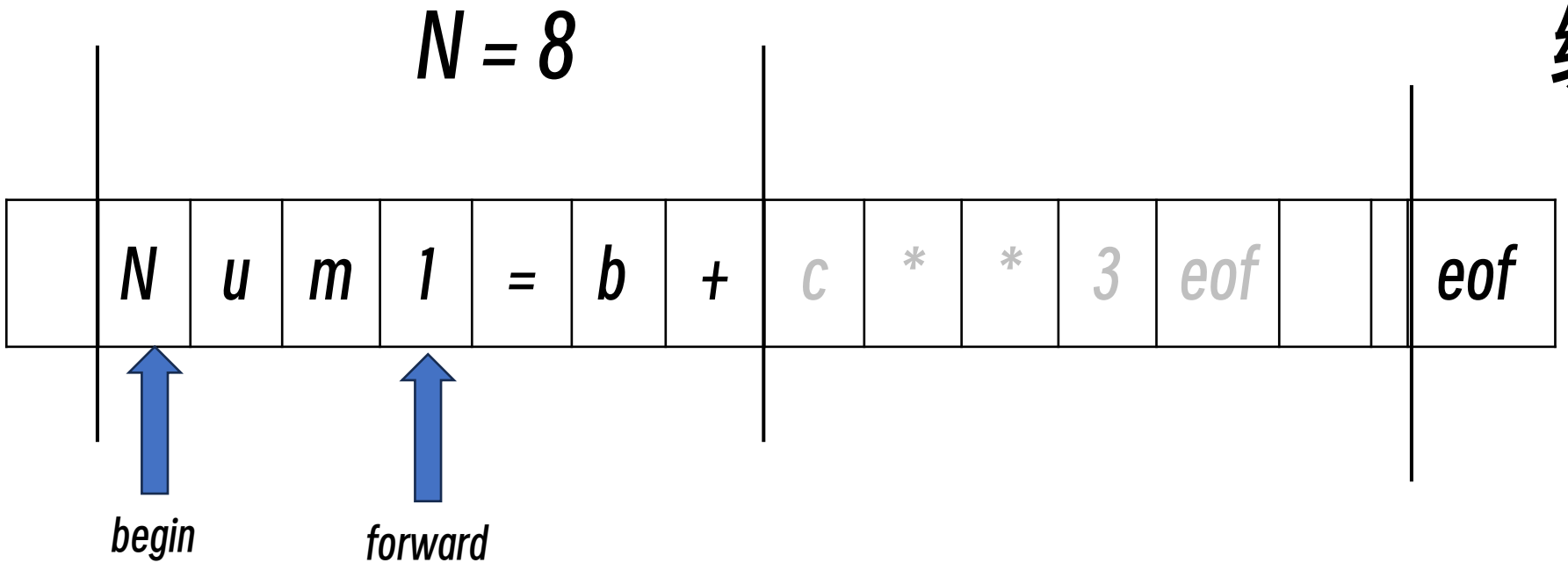
读取输入

N = 8

缓冲区对

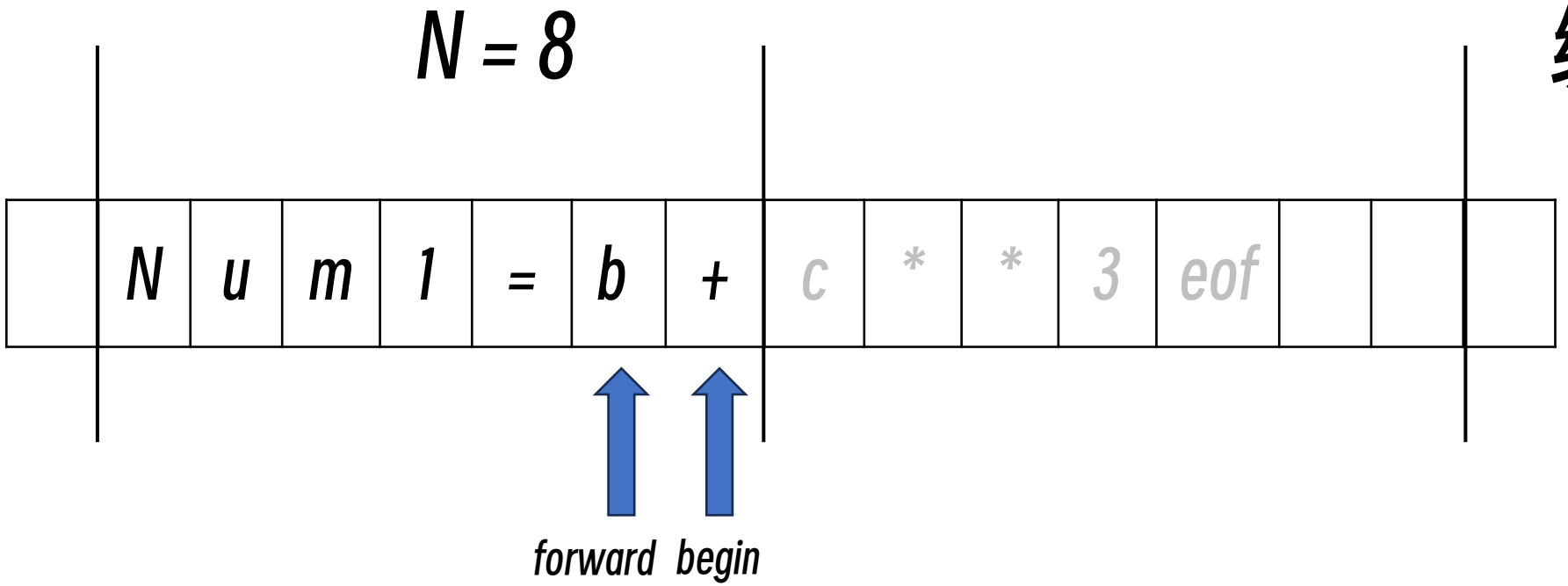


读取输入



<Num1> ⟶ 符号表 ⟶ <Num1,p10086> ⟶ 语法分析器

读取输入

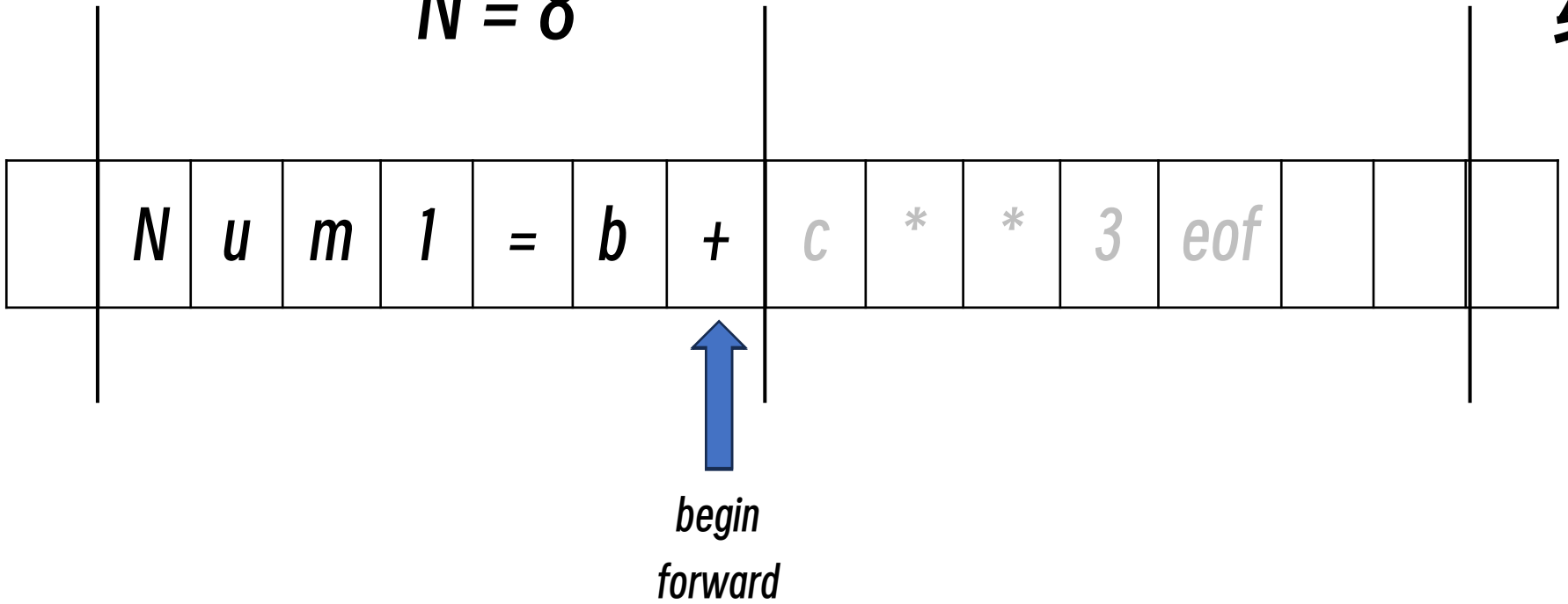


缓冲区对

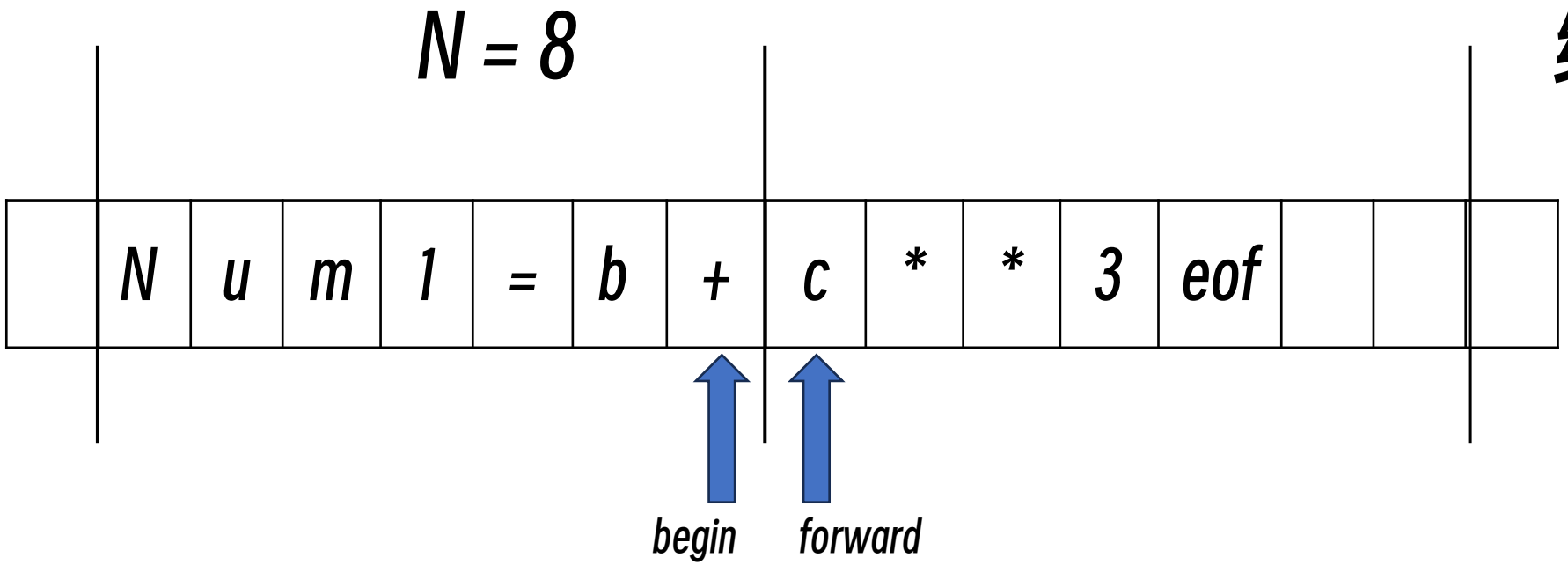
读取输入

N = 8

缓冲区对



读取输入



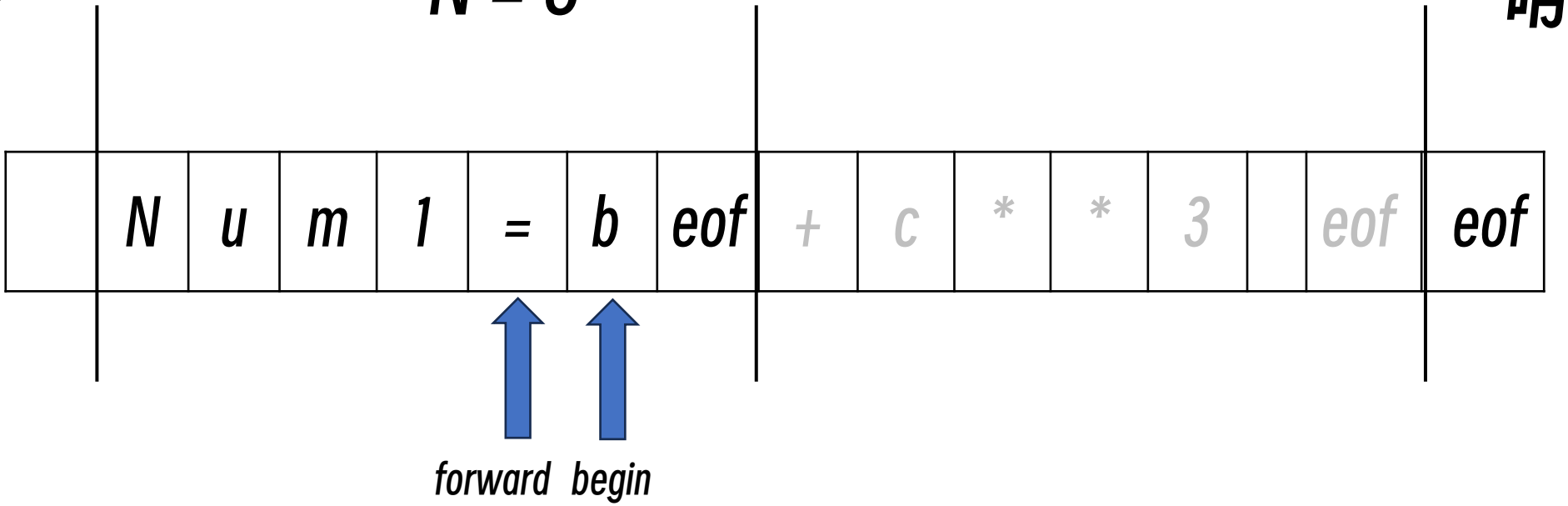
缓冲区对

forward指针每前移一次，都需要判断是否能前移（即是否到达了缓冲区末尾）

读取输入

N = 8

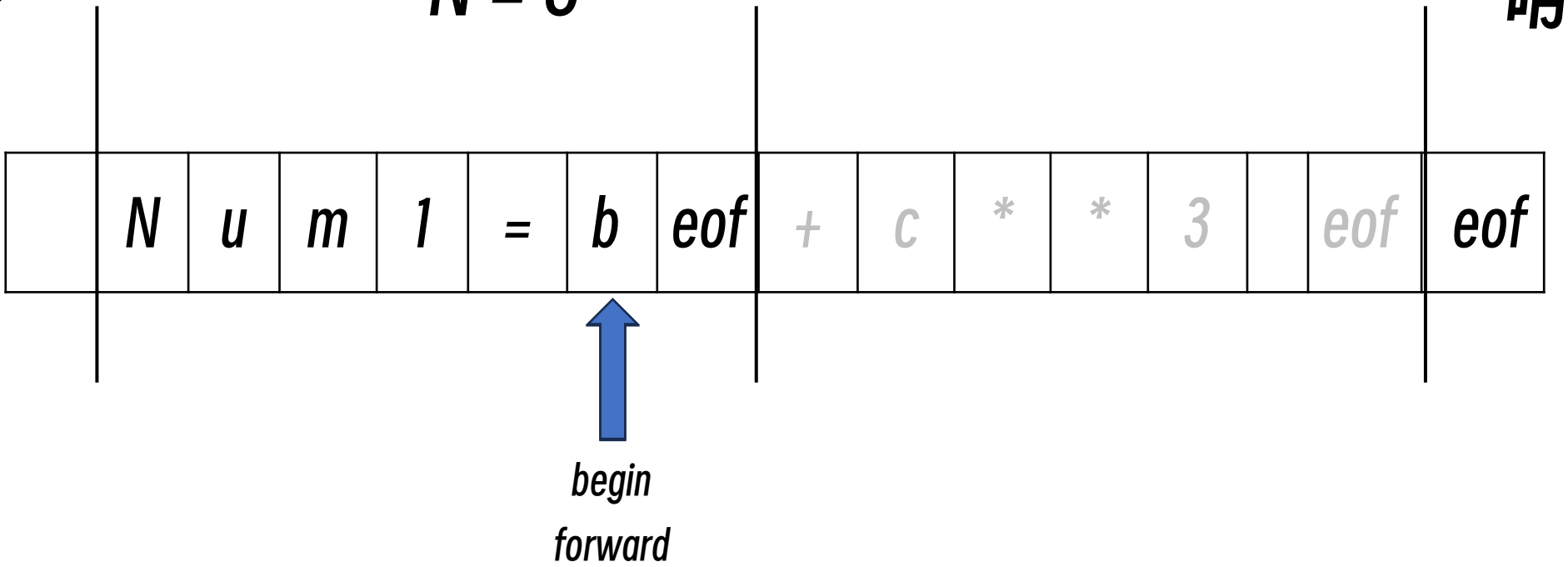
哨兵标记



读取输入

N = 8

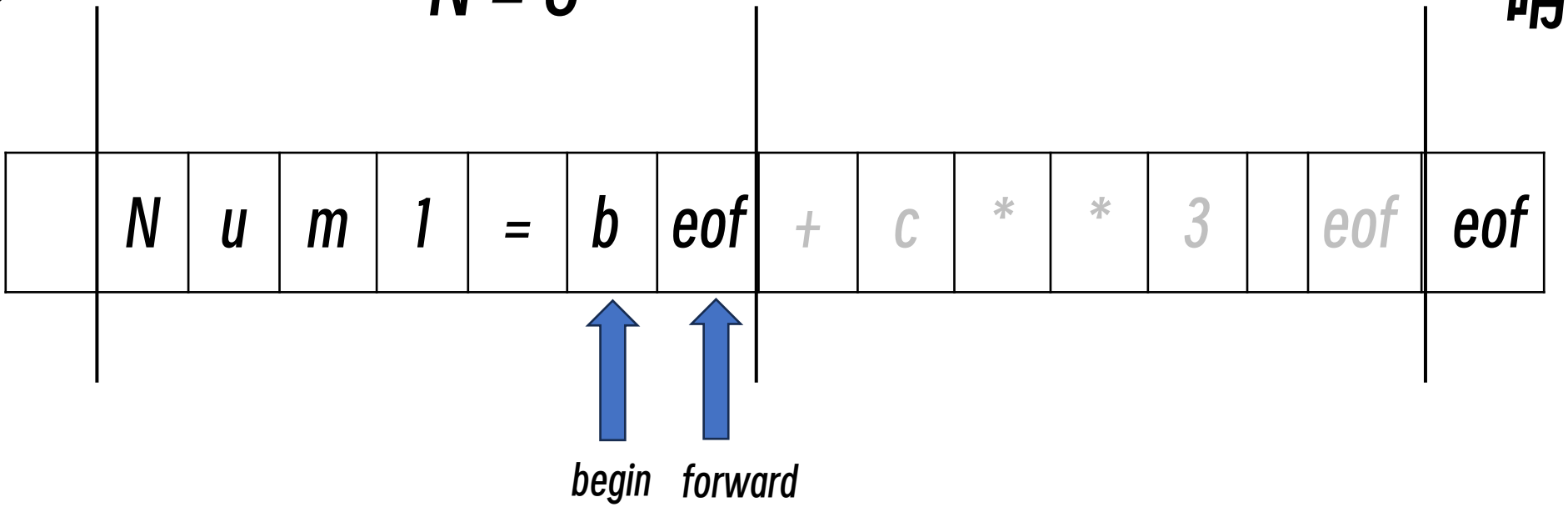
哨兵标记



读取输入

N = 8

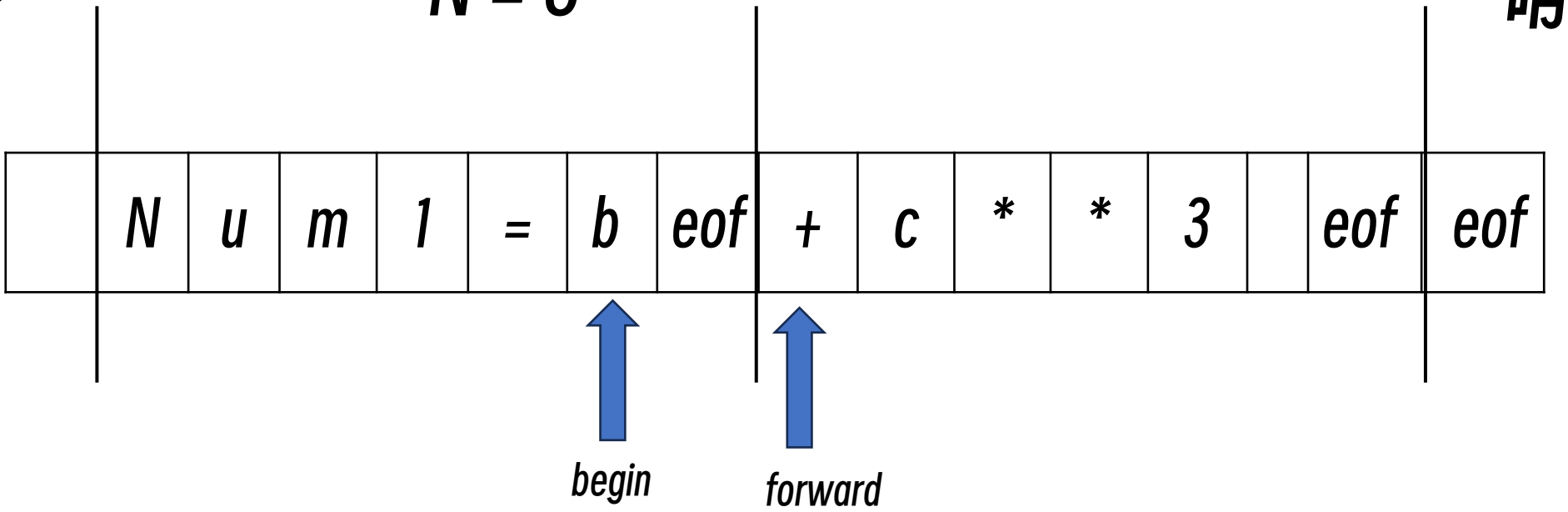
哨兵标记



读取输入

N = 8

哨兵标记

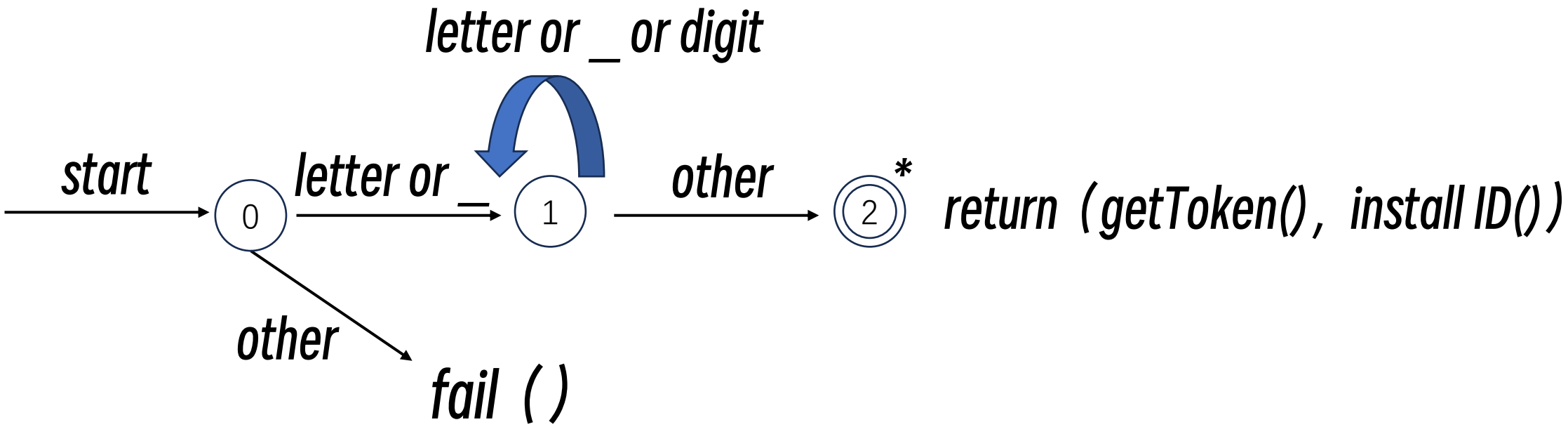


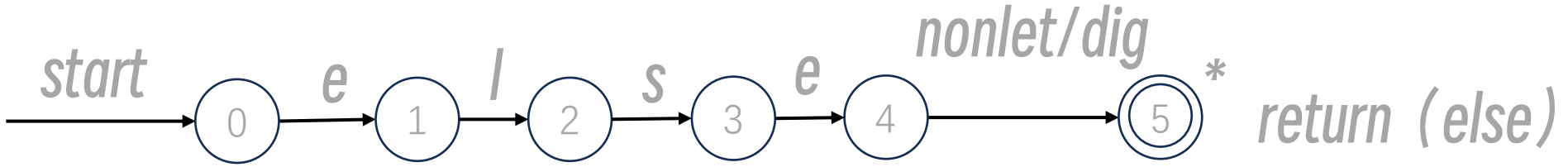
注意：

如果该缓冲区最后一个eof刚好是整个输入的eof，则需要额外判断一次

如果不在缓冲区的末尾出现了eof，则标志为所有输入读取完成

状态转换图





if(i==1) v=1;else return; ⟶ if(i==1)v=1;elsereturn;

状态转换图

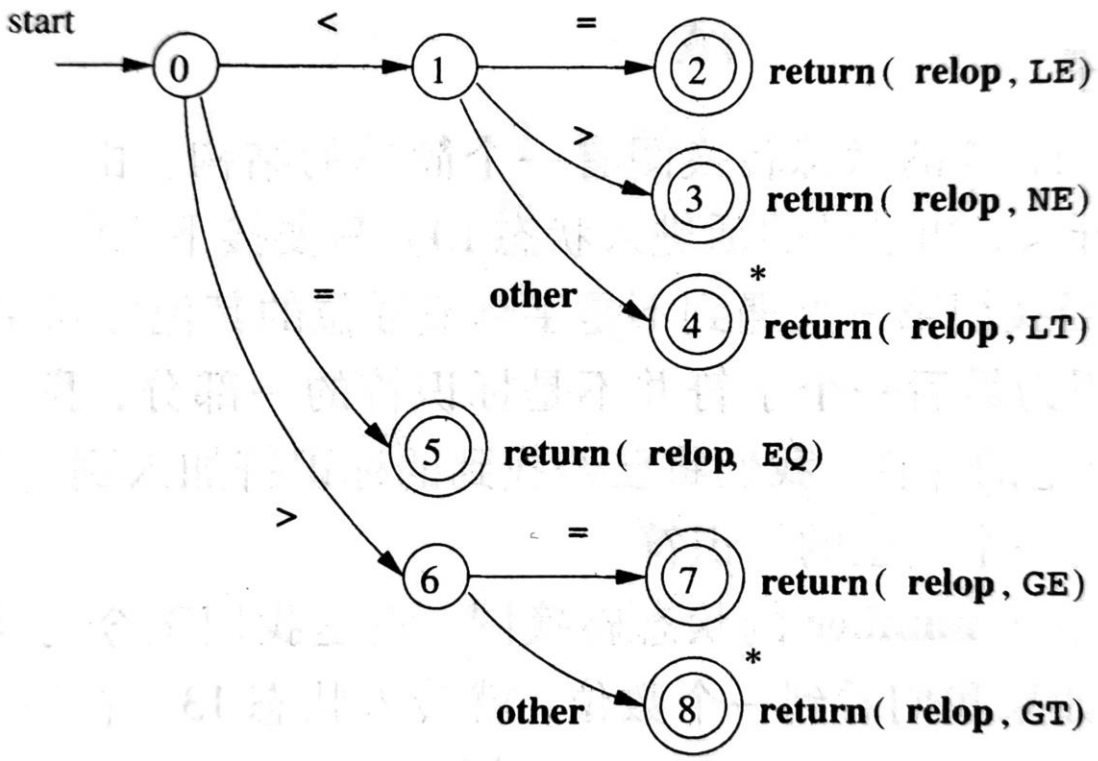


图 3-13　词法单元 **relop** 的状态转换图

```
1  while(1){
2      switch (state){
3          case 0:
4          if(c == '<') state = 1;
5          else if(c == '=') state = 5;
6          else if(c == '>') state = 6;
7          else fail();
8          break;
9      }
10 }
```

状态转换图

1、串行尝试状态图
2、并行运行状态图

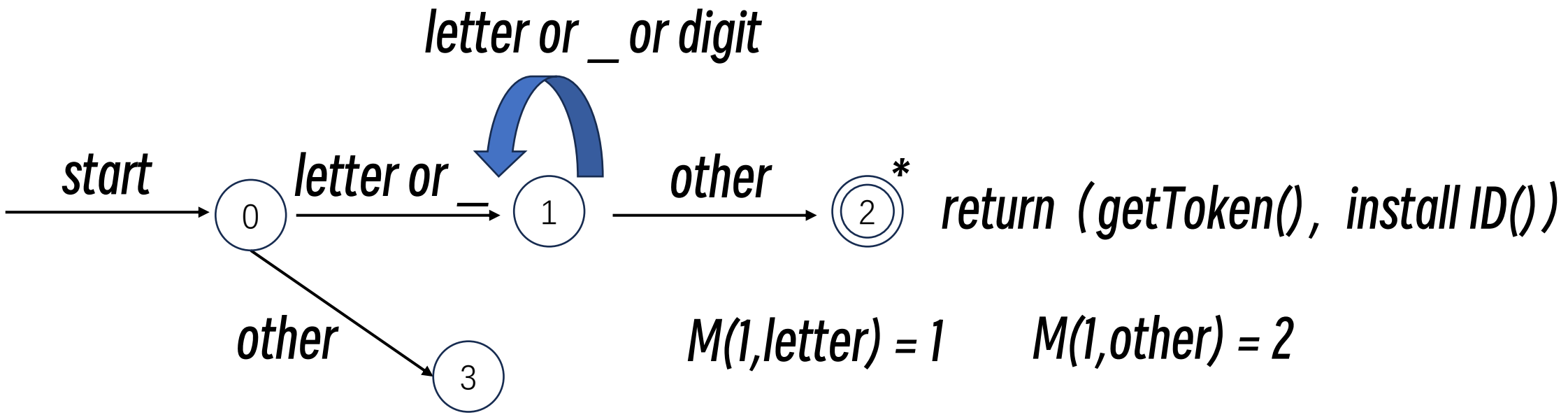
合并所有状态图

Lex语言

```
%{
    /*....*/
%}

/*声明部分*/
delim    [ \t\n]

ws       {delim}+

letter   [A-Za-z]

digit    [0-9]

id       {letter}({letter}|{digit})*

number   (digt)+(\.{digit}+)?(E[+-]?{digit}+)?

%%
/*转换规则*/



/*......*/

%%


/*辅助函数*/

int installId(){
    /*.......*/
}
```

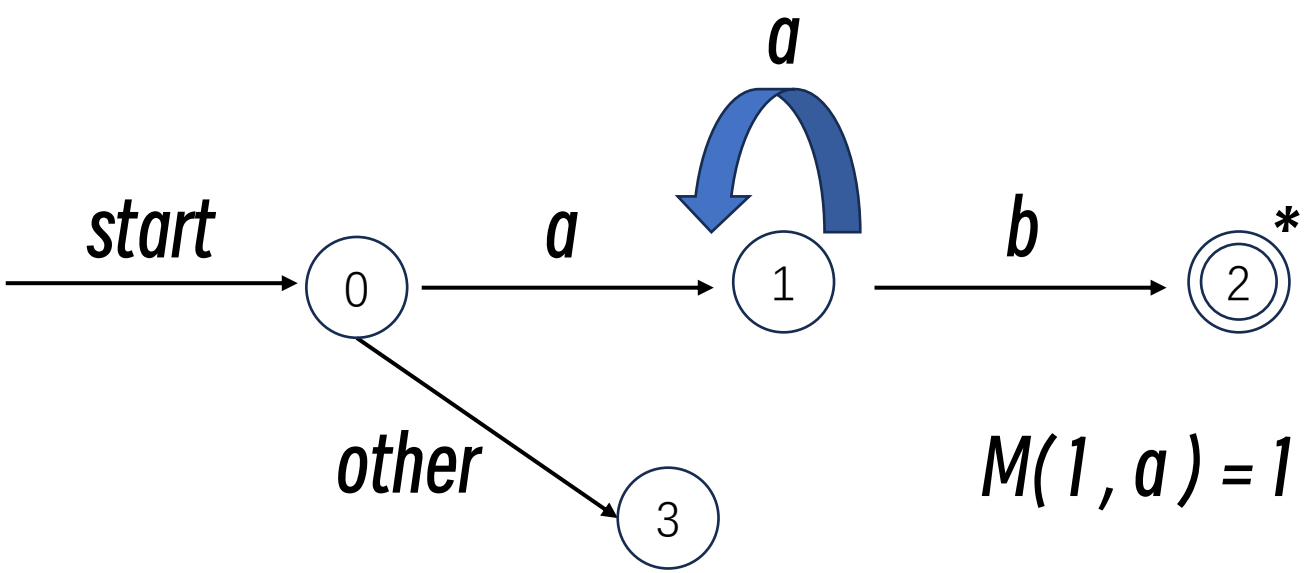
有穷自动机（NFA、DFA）

	NFA $N(S, \Sigma, M, s_0, F)$	DFA $D(S, \Sigma, M, s_0, F)$
S	有穷状态集合	
Σ	输入的字母表	
M	映射关系	
	非确定的，一对多的	确定的，一对一的
s_0	初始状态	
F	终止状态集合	



有穷自动机（NFA、DFA）

	NFA $N(S, \Sigma, M, s_0, F)$	DFA $D(S, \Sigma, M, s_0, F)$
S	有穷状态集合	
Σ	输入的字母表	
M	映射关系	
	非确定的，一对多的	确定的，一对一的
s_0	初始状态	
F	终止状态集合	



return（getToken()，install ID()）

$M(1, a) = 1$　　$M(1, a) = 2$

字符串高效处理

基于正则表达式的DFA(NFA)

DFA模式匹配优化

状态最小算法等等

```
if ( elseabc == 1) v=1;else ifa = 2;
```

```
if(elseabc==1)v=1;elseifa =2;
```

谢谢大家