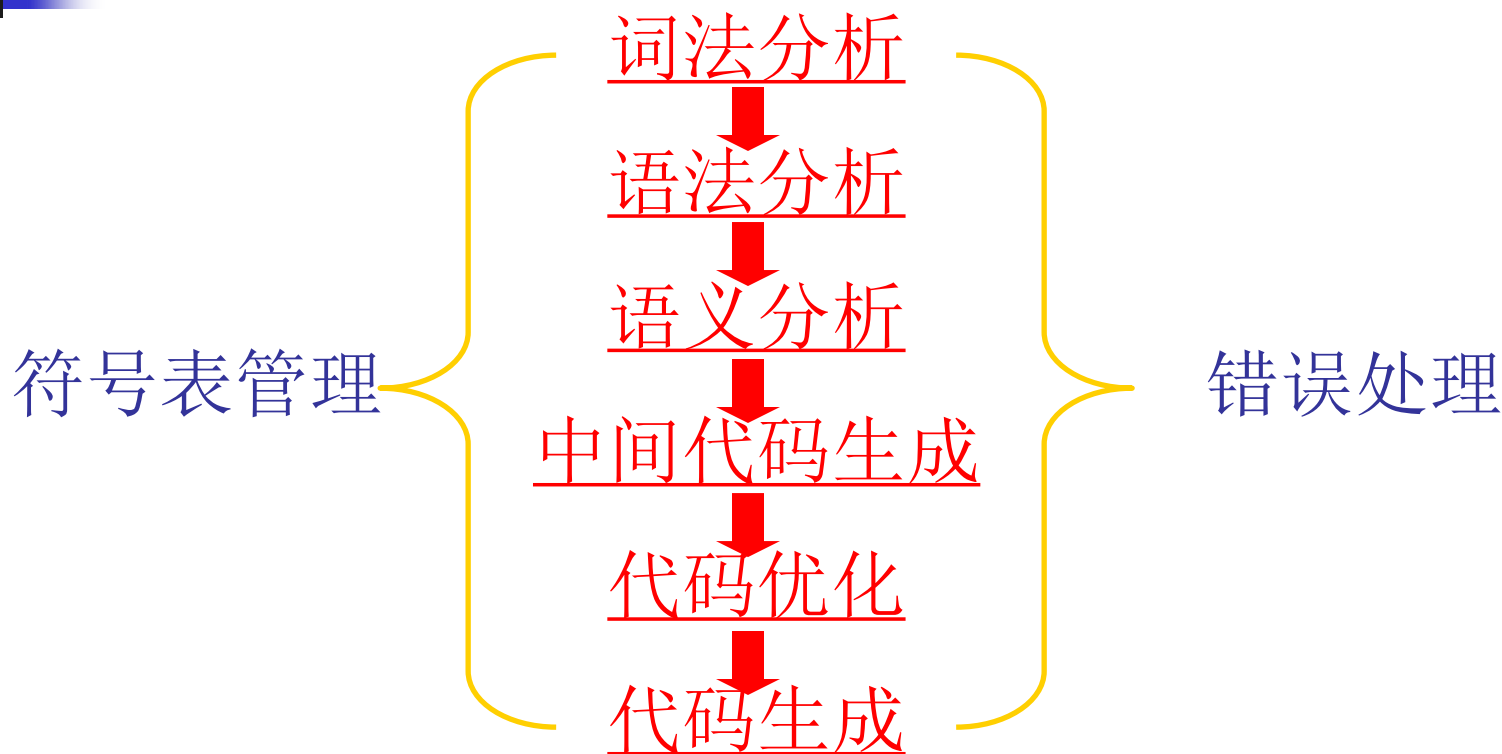


复习要点

编译器六个阶段



各个阶段做什么事？

倒过来问：某个事情在哪个阶段做？

几个阶段如何组织？

编译器报告“缺少运算符”错误是在____阶段。

- ☐ A 词法分析
- ☒ B 语法分析
- ☐ C 语义分析
- ☐ D 代码生成
- ☐ E 代码加载
- ☐ F 代码执行

提交

编译器对常量进行类型转换是在_____阶段。

- ☐ A 词法分析
- ☐ B 语法分析
- ☒ C 语义分析
- ☐ D 代码生成
- ☐ E 代码加载
- ☐ F 代码执行

提交

将可重定位机器码中相对地址修改为绝对地址是在_____阶段。

- ☐ A 词法分析
- ☐ B 语法分析
- ☐ C 语义分析
- ☐ D 代码生成
- ☒ E 代码加载
- ☐ F 代码执行

提交

早期BASIC语言是源程序逐条语句分析执行，因此它是一种_____。

- ☐ A 预处理器
- ☐ B 编译器
- ☐ C 链接器
- ☒ D 解释器

提交

PASCAL程序执行方式是源程序转换为目标平台的可执行程序再执行，因此它是一种_____。

- ☐ A 预处理器
- ☒ B 编译器
- ☐ C 链接器
- ☐ D 解释器



提交



词法分析

- 技术路线（总体知识点）
 - 用正则表达式描述单词
 - 模拟有限状态自动机执行来识别单词
 - 正则表达式与自动机间的桥梁——转换算法
 - 正则表达式→NFA，Thompson构造法
 - NFA→DFA，子集构造法
 - DFA最小化



基本概念

○ 符号串集合!!!

- 单词：类别，符号串分组
- 模式：单词 \leftrightarrow 词素
- 词素：实例

○ 正则式——模式

- 用简单语言（符号串集合）的运算描述复杂语言
- 正则式：语言运算的描述方法



正则表达式基本概念

- 字母表、符号串、语言
- 符号串（集合）的运算
- 正则表达式递归定义：|、连接、*、()
- 正则式运算法则，等价
- 正则定义
- 符号简写：+、?、[]

$\text{FIRST}(\alpha) = \{\epsilon\}$ 中的 ϵ 表示_____。

- ☐ A 字母表中符号
- ☒ B 长度为0的符号串
- ☐ C 空集
- ☐ D 包含一个长度为0的符号串的集合

提交

s, t, p 是相同字母表上的正则表达式，则下面正确的是_____。

- ☐ A $st = ts$
- ☒ B $(s \mid t)p = sp \mid tp$
- ☐ C $(st) \mid p = (s \mid p)(t \mid p)$
- ☐ D $(s \mid t)^* = s^* \mid t^*$

提交



设计正则表达式，接受首尾符号不同的0、1串

$(0(0^*1)^+) \mid (1(1^*0)^+)$



有限自动机基本概念

- 非确定有限自动机, NFA
 - 五元组, $M = \{ S, \Sigma, \delta, s_0, F \}$
 $\delta: S \times \Sigma \cup \{\varepsilon\} \rightarrow 2^S$
 - 工作方式
- 确定有限自动机, DFA
 - $\delta: S \times \Sigma \rightarrow S$
 - 工作方式
- 表示方式
 - 五元组
 - 状态转换图: 节点 (状态)、边的含义
符号串集合!
 - 状态转换矩阵

我们倾向于使用DFA而非NFA构造词法分析器，是因为_____。

- ☐ A DFA空间占用优于NFA
- ☒ B DFA时间复杂性优于NFA
- ☐ C 以上皆对
- ☐ D 以上皆错

提交



设计DFA，接受能被3整除的八进制数

S: 除3余0

A: 除3余1

B: 除3余2

$\text{move}(S, 0/3/6)=S$

$\text{move}(S, 1/4/7)=A$

$\text{move}(S, 2/5)=B$

$\text{move}(A, 1/4/7)=S$

$\text{move}(A, 0/3/6)=B$

$\text{move}(A, 2/5)=A$

$\text{move}(B, 2/5)=S$

$\text{move}(B, 1/4/7)=B$

$\text{move}(B, 0/3/6)=A$

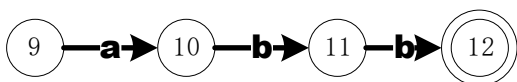
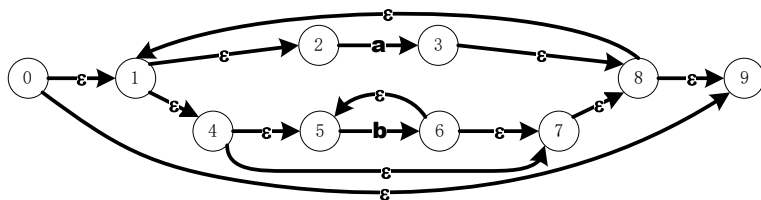
S为终态



正则式→NFA

- 正则式——静态描述→自动机——动态识别；DFA时间复杂性更优，NFA转换更方便——中间桥梁
- Thompson构造法
 - 语法制导翻译方法
 - 基本正则式→基本NFA
 - 正则式操作→NFA组合构造

对正则表达式 $(a \mid b^*)^*abb$ ，用Thompson构造法转换为NFA



NFA \rightarrow DFA——子集构造法

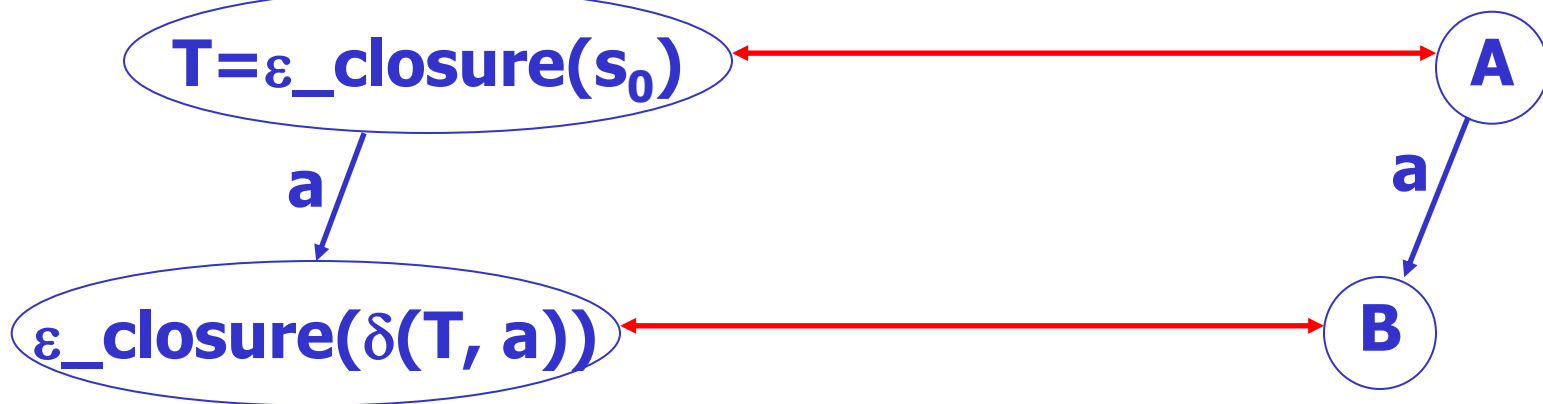
○ 得到的NFA与原DFA等价 \rightarrow

任何符号串 x 在DFA识别结果（单个状态） \equiv 在NFA识别结果（状态集合） \rightarrow

用NFA状态子集表示DFA来实现DFA构造 \rightarrow

不可能穷举所有“ x ”——由简单到复杂：

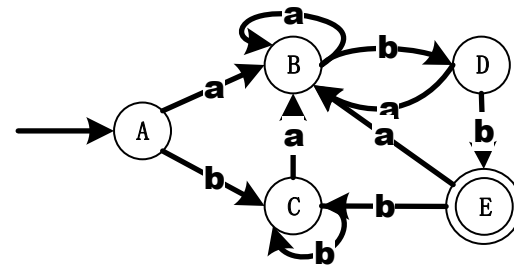
长度为0、长度为1、...，长度为 $k+1$ 的符号串看作长度为 k 的拼接一个符号



对正则表达式 $(a \mid b^*)^*abb$ ，将Thompson构造法得到的NFA用子集构造法转换为DFA，用其识别abbaabb

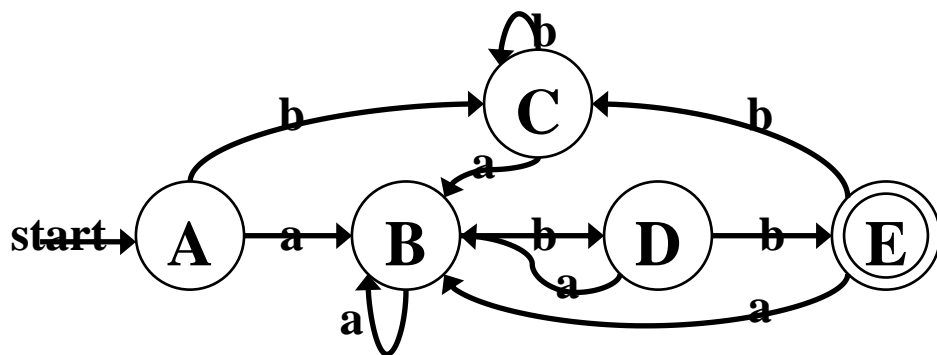
$A = e_closure(\{0\}) = \{0, 1, 2, 4, 5, 7, 8, 9\}$
 $\epsilon_closure(d(A, a)) = \{1, 2, 3, 4, 5, 7, 8, 9, 10\} = B$
 $\epsilon_closure(d(A, b)) = \{1, 2, 4, 5, 6, 7, 8, 9\} = C$
 $\epsilon_closure(d(B, a)) = B$
 $\epsilon_closure(d(B, b)) = \{1, 2, 4, 5, 6, 7, 8, 9, 11\} = D$
 $\epsilon_closure(d(C, a)) = B$
 $\epsilon_closure(d(C, b)) = C$
 $\epsilon_closure(d(D, a)) = B$
 $\epsilon_closure(d(D, b)) = \{1, 2, 4, 5, 6, 7, 8, 9, 12\} = E$
 $\epsilon_closure(d(E, a)) = B$
 $\epsilon_closure(d(E, b)) = C$

识别过程：A → B → D → E → B → B → D → E



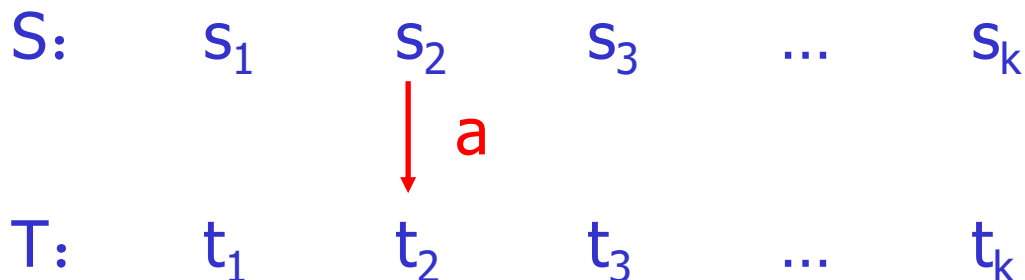
最小化DFA

- 核心思想：“区分”概念 \rightarrow 不可区分的状态合并
- 实际算法：从状态全集开始，符号串 x 可区分 \rightarrow 状态集分裂
- 不可能穷举所有符号串，由简单到复杂，长度 $k+1$ 看作单个符号拼接长度 k
“区分”判定 \rightarrow 状态转移



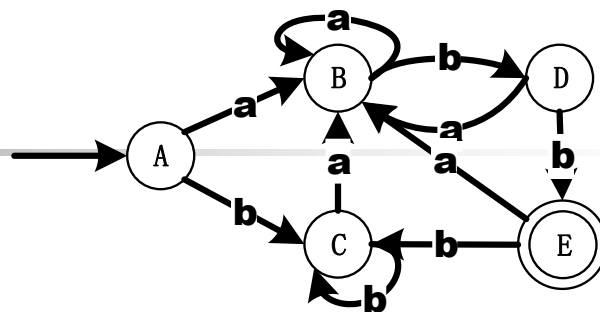
算法

- 初始：{ 终态 }、{ 非终态 }



- t不同组，存在符号串x可区分它们→
符号串ax可区分s→
s按t的分组方式分裂

将前面得到的DFA最小化

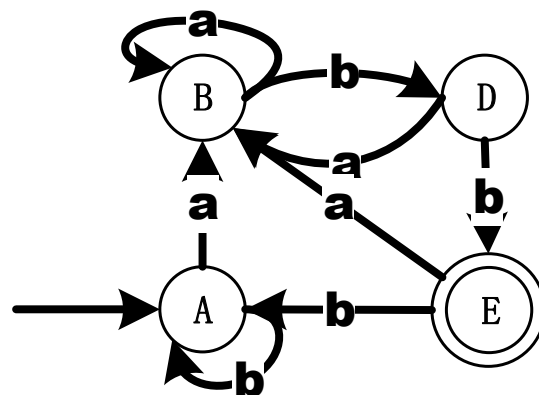


初始划分 $\{A, B, C, D\} \{E\}$

$\{A, B, C, D\} \xrightarrow{b} \{C, D, C, E\}$,
因此划分为 $\{A, B, C\} \{D\}$

$\{A, B, C\} \xrightarrow{b} \{C, D, C\}$, 因此
划分为 $\{A, C\} \{B\}$

$\{A, C\}$ 不可再分, 最终状态分
组为 $\{A, C\} \{B\} \{D\} \{E\}$





语法分析基本概念

- 上下文无关文法, CFG, 四元式(V_T, V_N, S, P)
 $A \rightarrow \alpha$
- 推导、语言、句型、句子
- 最左推导、最右推导
- 语法树、二义性文法
- CFG的等价



与词法分析的联系

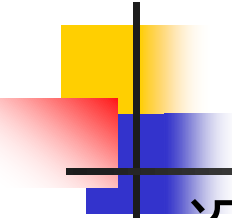
○ CFG和正则式

- 正则式、NFA/DFA改写为等价的文法（正则文法，3型文法）
- 正则式可描述的语言
- 正则式不能而CFG能描述的语言
- 均不能描述的



CFG的设计

- 消除左递归：直接、间接
- 消除 ϵ 产生式
- 消除回路
- 提取左公因子
- 为什么要做这些改写？



设计接受语言 $\{a^i b^j a^k b^l \mid i+j=k+l, i, j, k, l \geq 0\}$
的上下文无关文法

A: 中央的a比b多

B: 中央的b比a多


M: 中央的a和b一样多

$S \rightarrow aSb \mid A \mid B \mid M$

$A \rightarrow aAa \mid M$

$B \rightarrow bBb \mid M$

$M \rightarrow bMa \mid \varepsilon$



设计上下文无关文法接受以两个0结尾的
01串

$S \rightarrow A00$

$A \rightarrow BA \mid B$

$B \rightarrow 0 \mid 1$



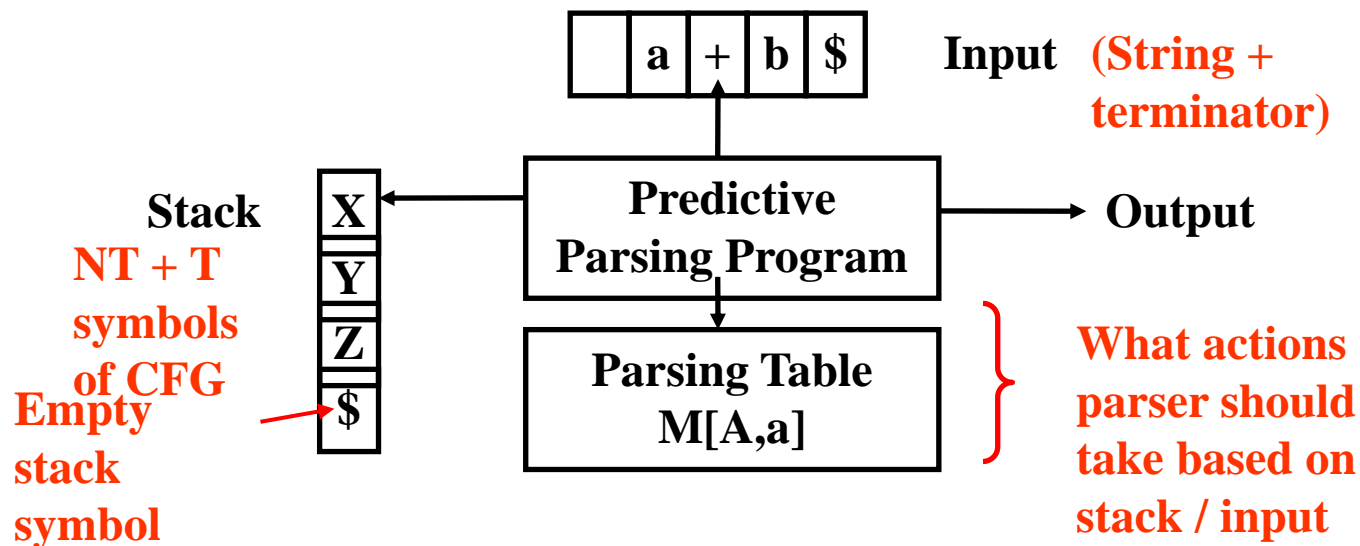
自顶向下语法分析（重点）


- 寻找最左推导
语法树构造根→叶
由整体→局部
- 预测分析法
 - 当前输入符号+待扩展NT→避免回溯
- 递归下降法构造预测分析器：
NT→递归函数

非递归实现

○ FIRST、FOLLOW

→ 预测分析表, LL(1)文法





对下面CFG，指出终结符集合、非终结符集合、开始符号

$S \rightarrow \%aT \mid U!$

$T \rightarrow aS \mid baT \mid e$

$U \rightarrow \#aTU \mid e$

终结符集合 $\{\%, a, !, b, \#\}$

非终结符集合 $\{S, T, U\}$

开始符号 S

对下面CFG，构造预测分析表，分析句子

#abaa%a!

$S \rightarrow \%aT \mid U!$

$T \rightarrow aS \mid baT \mid \epsilon$

$U \rightarrow \#aTU \mid \epsilon$

$FIRST(S) = \{\%, \#, !\}$

$FIRST(T) = \{a, b, \epsilon\}$

$FIRST(U) = \{\#, \epsilon\}$

$FOLLOW(S) = \{\#, !, \$\}$

$FOLLOW(T) = \{\#, !, \$\}$

$FOLLOW(U) = \{!\}$

	a	b	#	%	!	\$
S			$S \rightarrow U!$	$S \rightarrow \%aT$	$S \rightarrow U!$	
T	$T \rightarrow aS$	$T \rightarrow baT$	$T \rightarrow \epsilon$		$T \rightarrow \epsilon$	$T \rightarrow \epsilon$
U			$U \rightarrow \#aTU$		$U \rightarrow \epsilon$	

栈	输入	动作
\$S	#abaa%a!\$	$S \rightarrow U!$
\$!U	#abaa%a!\$	$U \rightarrow \#aTU$
\$!UTa#	#abaa%a!\$	
\$!UTa	abaa%a!\$	
\$!UT	baa%a!\$	$T \rightarrow baT$
\$!UTab	baa%a!\$	
\$!UTa	aa%a!\$	
\$!UT	a%a!\$	$T \rightarrow aS$
\$!USa	a%a!\$	
\$!US	%a!\$	$S \rightarrow \%aT$
\$!UTa%	%a!\$	
\$!UTa	a!\$	
\$!UT	!\$	$T \rightarrow \epsilon$
\$!U	!\$	$U \rightarrow \epsilon$
\$!	!\$	
\$	\$	accept



自底向上语法分析

- 寻找最右推导
语法树构造叶 \rightarrow 根
由局部 \rightarrow 整体
- 句柄
 - $S \xRightarrow{*}_{rm} \alpha A w \xRightarrow{*}_{rm} \alpha \beta w$
- “移进—归约”分析方法
 - 基本操作：移进、归约、接受、错误



算符优先分析方法

- 算符文法
- 算符优先级：低于、高于、等于——
算符优先文法
- 句柄的确定
 - < = ... = >

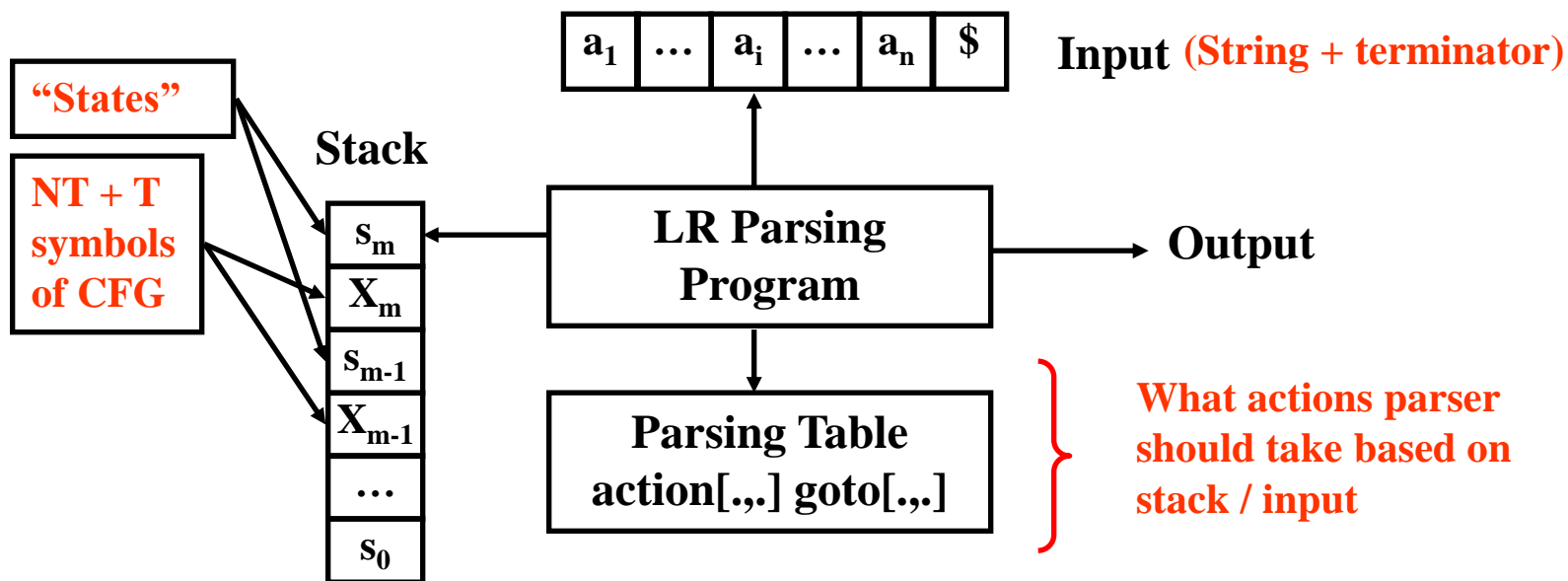


LR分析方法

- 活前缀
- 核心思想：构造识别活前缀的DFA

LR分析方法

- LR分析器：状态、action表、goto表
- 格局



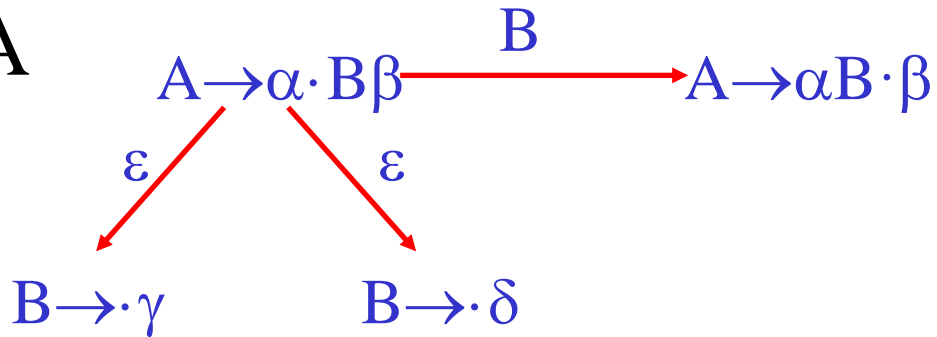


SLR分析法（重点）

- LR(0)项目： $A \rightarrow X \cdot YZ$
 - 已分析得到X，期待分析得到YZ
- 构造识别活前缀的DFA
 - LR(0)项目集——DFA状态
 - LR(0)项目集规范族 \rightarrow DFA
 - closure函数：存在 $A \rightarrow \alpha \cdot B\beta \rightarrow$ 添加 $B \rightarrow \cdot \gamma$
 - goto函数： $\text{goto}(I, X) \rightarrow A \rightarrow \alpha \cdot X\beta \rightarrow A \rightarrow \alpha X \cdot \beta$

SLR分析法

- 用LR(0)项目可直接构造识别活前缀的NFA



- 转换为DFA——计算LR(0)项目集规范族
- 构造SLR分析表
- SLR(1)文法



规范LR分析法

- SLR分析能力不够
- LR(1)项目: $[S \rightarrow aA \cdot Be, c]$



规范LR分析法

- closure函数
 - 存在 $[A \rightarrow \alpha \cdot B \beta, a]$ ——添加 $[B \rightarrow \cdot \eta, b]$,
 $b \in \text{FIRST}(\beta a)$
- 规范LR分析法的构造
- LR(1)文法



LALR分析法

- 介于SLR和规范LR之间
- 简单LALR分析表构造
 - 构造LR(1)项目集规范族
 - 同心（LR(0)项目部分）集合并
- LALR(1)文法
- 如何用LR算法分析二义性文法

对下面上下文无关文法，下列说法不正确的是

$S \rightarrow aA$ $A \rightarrow Bb$ $B \rightarrow Ba \mid a$ $C \rightarrow Ab$

- ☐ A C是无用的
- ☐ B 与 aa^+b 对应相同的语言
- ☐ C 是算符文法
- ☒ D $aaab$ 是其活前缀

提交

aAA不是下面上下文无关文法的活前缀，原因是

$S \rightarrow aA\overset{\circ}{B}e$

$A \rightarrow Abc \mid b$

$B \rightarrow d$

- ☒ A 不存在最右句型，其前缀是aAA
- ☐ B aAA是某个最右句型的前缀，但它在句柄左侧
- ☐ C aAA是某个最右句型的前缀，但它的末尾超过了句柄末尾
- ☐ D 以上皆错

提交

题型示例（续）

四、（11 分）给定上下文无关文法 ϵ

$E \rightarrow id \epsilon$

$E \rightarrow id (E) \epsilon$

$E \rightarrow E + id \epsilon$

ϵ

拓广文法，计算 LR(0)项目集规范族，构造 SLR 分析表，它是 SLR(1)文法吗？ ϵ

解： ϵ

拓广文法 ϵ

(0) $S \rightarrow E \epsilon$

(1) $E \rightarrow id \epsilon$

(2) $E \rightarrow id (E) \epsilon$

(3) $E \rightarrow E + id \epsilon$

题型示例（续）

计算 LR(0)项目集规范族[↵]

$\text{closure}(\{S \rightarrow \cdot E\}) = \{S \rightarrow \cdot E, E \rightarrow \cdot \text{id}, E \rightarrow \cdot \text{id} (E), E \rightarrow \cdot E + \text{id}\} = I_0$ [↵]

$\text{goto}(I_0, E) = \{S \rightarrow E \cdot, E \rightarrow E \cdot + \text{id}\} = I_1$ [↵]

$\text{goto}(I_0, \text{id}) = \{E \rightarrow \text{id} \cdot, E \rightarrow \text{id} \cdot (E)\} = I_2$ [↵]

$\text{goto}(I_1, +) = \{E \rightarrow E + \cdot \text{id}\} = I_3$ [↵]

$\text{goto}(I_2, () = \{E \rightarrow \text{id} (\cdot E), E \rightarrow \cdot \text{id}, E \rightarrow \cdot \text{id} (E), E \rightarrow \cdot E + \text{id}\} = I_4$ [↵]

$\text{goto}(I_3, \text{id}) = \{E \rightarrow E + \text{id} \cdot\} = I_5$ [↵]

$\text{goto}(I_4, E) = \{E \rightarrow \text{id} (E \cdot), E \rightarrow E \cdot + \text{id}\} = I_6$ [↵]

$\text{goto}(I_4, \text{id}) = I_2$ [↵]

$\text{goto}(I_6, +) = I_3$ [↵]

$\text{goto}(I_6,)) = \{E \rightarrow \text{id} (E) \cdot\} = I_7$ [↵]

[↵]

$\text{FIRST}(E) = \{\text{id}\}$ [↵]

$\text{FOLLOW}(E) = \{\$,), +\}$ [↵]

题型示例（续）

构造 SLR 分析表

	action					goto
	id	()	+	\$	E
0	s2					1
1				s3	acc	
2		s4	r1	r1	r1	
3	s5					
4	s2					6
5			r3	r3	r3	
6			s7	s3		
7			r2	r2	r2	

分析表没有冲突，所以它是 SLR(1) 文法。





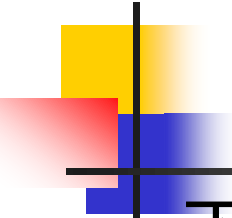
语法制导翻译

- 综合属性、继承属性
- S-属性定义
 - 仅综合属性
- L-属性定义
 - 继承属性依赖父结点，或左兄弟结点
- 语法制导定义/翻译模式的设计

综合属性计算的依赖关系是父节点依赖孩子节点，所以综合属性的计算_____。

- ☐ A 容易与预测分析法相结合
- ☒ B 容易与算符优先分析算法相结合
- ☐ C 以上皆对
- ☐ D 以上皆错

提交



下面文法描述了正则表达式，设计语法制导定义实现构造正则表达式对应的表达式树。假设已有辅助函数mkleaf(char) (及mkleaf_epsilon())和mknnode(op, child1, child2)分别为基本正则表达式和正则表达式运算创建叶节点和内部节点

$$R \rightarrow \text{char} \mid ' \epsilon ' \mid R \mid R \mid R . R \mid R * \mid (R)$$

```
R → char    { R.p = mkleaf(char.lexme); }  
R → ' ε '   { R.p = mkleaf_epsilon(); }  
R → R1 '|' R2 { R.p = mknnode(UNION, R1.p,  
R2.p); }  
R → R1 . R2 { R.p = mknnode(CON, R1.p, R2.p);  
}  
R → R1 * { R.p = mknnode(CLOSURE, R1.p,  
NULL); }  
R → ( R1 ) { R.p = R1.p; }
```



类型检查

○ 类型表达式

- 基本类型表达式: `int`、`...`、`void`、`type_error`、类型名
- 类型构造符: 数组、指针、函数、结构、...

○ 类型表达式等价

- 结构等价: 语法树等价——内在
- 名字等价: 形式等价——外在

关于下面类型表达式，正确说法是_____。
(pointer(char)→int)→pointer(char)

- ☐ A C语言对这种类型的等价判定采用名字等价方式
- ☐ B Pascal语言对这种类型的等价判定采用结构等价方式
- ☒ C 在C语言中，这种类型会引发类型错误
- ☐ D 以上皆错

提交





中间代码生成

- 表达式翻译
 - 临时名字重用
- 布尔表达式和控制流语句的翻译
 - 真假值出口

题型示例

9. 将下面表达式转换为三地址码, 采用临时变量重用算法, 可将临时变量数目减少到_____个。

$$a + (a + a + a * (a + a + a))$$

A. 2

B. 3

C. 4

D. 5

题型示例（续）

9. 将下面表达式转换为三地址码，采用临时变量重用算法，可将临时变量数目减少到_____A_____个。

$$a + (a + a + a * (a + a + a))$$

A. 2

B. 3

C. 4

D. 5





代码优化

- 基本块、流图、循环、下次引用
- 基本优化方法
 - 消去公共子表达式
 - 复制传播
 - 无用代码删除
 - 循环优化
 - 代码外提
 - 强度削弱
 - 归纳变量删除



目标代码生成

- 目标代码生成算法
 - 基本块内局部最优
 - 计算结果尽量保存在寄存器内
- 寄存器分配优化
 - 着眼于循环，选定某些变量一直放于寄存器

题型示例

9. 我们能在_____时完成数组越界判定。↵
- A. 类型检查 B. 中间代码生成↵
- C. 目标代码生成 D. 以上均不对↵

得 分
↵

五、（10 分）对下面的三地址码程序划分基本块，画出流图，并计算每个基本块末尾的活动变量集合。↵

```
1 m := 0                      9 x := M[r]↵
2 v := 0                      10 s := s+x↵
3 if v >= n goto 15           11 if s <= m goto 13↵
4 r := v                      12 m := s↵
5 s := 0                      13 r := r + 1↵
6 if r < n goto 9              14 goto 6↵
7 v := v + 1                15 return m↵
8 goto 3↵
```





试题分布

○ 知识点分布

- 概述10%、词法分析35%、
语法分析35%、语义分析及翻译20%

○ 题型分布

- 单项选择20%左右（基本概念）、设计20%
左右、问答60%左右（算法应用）

○ 难易分布

- 简单40%、中等55%、较难5%



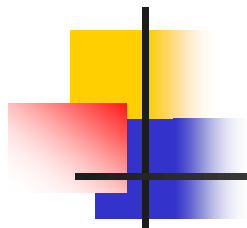
一种复习方式

- 预习作业（及反馈解答）和复习中的选择题→基本概念（期末考试选择题）
 - 如完成不是很好，结合讲义、预习情况分析
及课堂习题强化基本概念的复习
- 平时书面作业→算法应用（期末考试设计题、问答题）
 - 如完成不是很好，结合平时书面作业、教材
课后习题相应强化掌握欠佳的算法
- 然后做一套往年试卷，根据完成情况继续强化前两步的复习，如此往复



关于考试

- 时间1月11日14:00~15:40
- 认真复习，把精力用在考试前
- 诚信作答、不要作弊



祝同学们考试顺利！