编译原理习题课（一）

课外学习资料

- **龙书核心作者所授课程（STU，CMU）**
  - STU-CS143，CS243，CS343 逐步深入

- **CS143 - Compilers（Instructor: Alex Aiken）**
  - Compiler基础知识，完整介绍了COOL语言的实现过程
  - COOL语言：面向教学的编程语言，同时兼有OO特性
  - Lecture：课程官网
  - Video：B站有部分熟肉，edX有完整生肉

课外学习资料

- **CS243 - Advanced Compilers**
  - 编译器的结构都十分类似，Fortran开始结构比重变化：
  - 工作重心：从解析器（Parser）到优化（Optimization）
  - Lecture-21: Program Analysis and Optimization
  - Lecture-06:（Ullman）: Advanced Compiling Techniques
  - 缺少视频，21版讲义过于简单，06版适用于自学
- **CS343 - Advanced Topics in Compilers**
  - 领域经典论文，以不同Topic为章节
  - 选用论文截止2014年

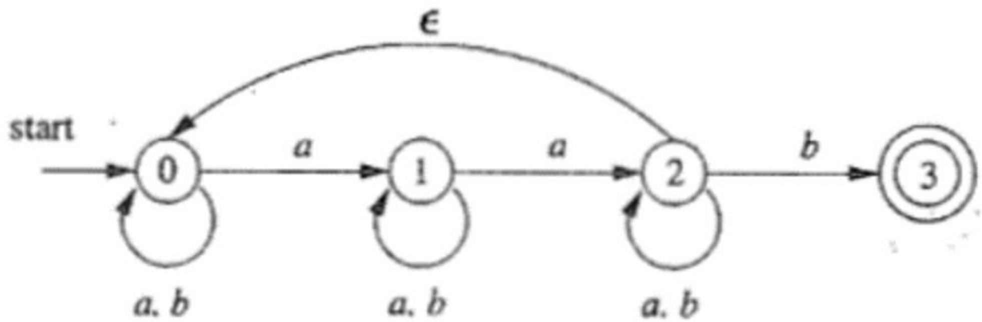
1. 试描述该正则表达式定义的语言：

$$\underbrace{(a|b)^*}_{\text{a和b组成的字符串}}\underbrace{a}_{\text{倒数第三位一定是a}}\underbrace{(a|b)(a|b)}_{\text{最后两位是a和b组成的字符串}}$$

由a和b组成的字符串，且倒数第三个字符为a。

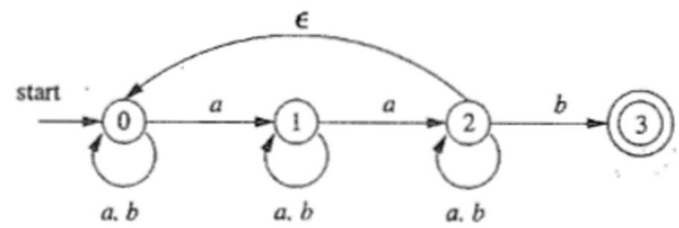
- 解不唯一，回答尽量简洁
- 书图3-5

2. 找出下图NFA中所有标号为aabb的路径，这个NFA接受aabb吗？



标号为aabb的路径为：00000, 00111, 01111, 01222, 01223, 012000, 012200。
接受aabb，如路径01223。

3. 将下图中的NFA转化为DFA：



（书算法3.20）首先构造子集：

A: ε-closure(0)={0}

B: Dtran[A,a]=ε-closure(move(A,a))=ε-closure(0,1)={0,1}

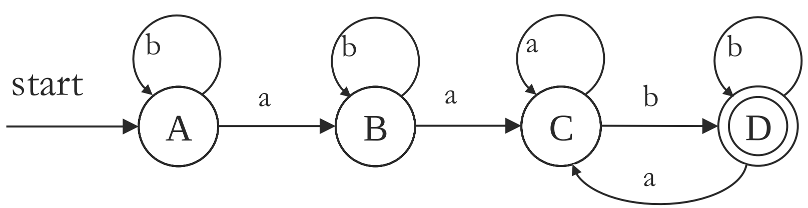
C: Dtran[B,a]=ε-closure(move(B,a))=ε-closure(0,1,2)={0,1,2}

D: Dtran[C,b]= ε-closure(move(C,b))=ε-closure(0,1,2,3)={0,1,2,3}

构造状态转换表：

NFA状态	DFA状态	Input-a	Input-b
0	A	B	A
0, 1	B	C	B
0, 1, 2	C	C	D
0, 1, 2, 3	D	C	D

DFA：



课后作业一 (Sec1-4)

- **Syntax Analysis 语法分析（Parsing）**
  - 语法分析的输入是词法分析后的tokens，输出是程序的解析树
  - CFG上下文无关语法
    - 解析树–语法的推导过程
      - 最左推导
      - 最右推导
      - 有歧义的语法（语法的一个语句可以用多棵解析树表示）
  - Top–Down Parsing 自上而下解析
    - 递归下降解析
      - 局限性（左递归）
    - 预测解析法
      - LL(K) 语法
      - LL(1) 算法（借助解析表）
      - LL(1) 解析表的构建
        - Fisrt集
        - Follow集
        - 构建算法
      - LL(1) 算法的限制性
  - Bottom–Up Parsing 自下而上解析
    - Shift Reduce Parsing 位移规约解析
      - 位移&规约
      - 句柄
      - 句柄的识别
        - 识别可行前缀
        - 合法项
    - SLR Parsing
      - LR(0) 解析
      - Simple LR 解析
      - LR(K) 解析
      - LALR 解析

课后作业一 (Sec1-4)

4. 考虑上下文无关文法：$S \rightarrow SS+ \mid SS^{*} \mid a$ 以及串 $aa+a^{*}$

（1）给出这个串的一个最左推导：

lm: $S \Rightarrow \underline{S}S^{*} \Rightarrow \underline{S}S+S^{*} \Rightarrow a\underline{S}+S^{*} \Rightarrow aa+S^{*} \Rightarrow aa+a^{*}$

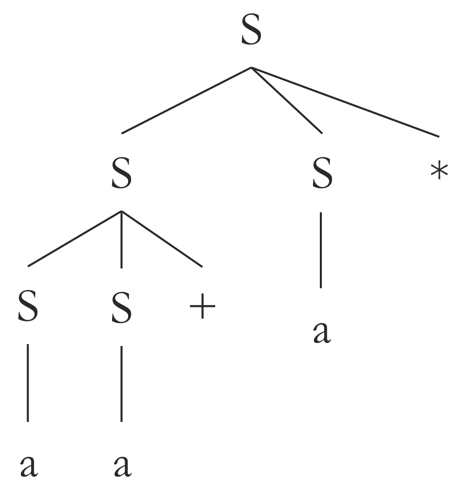
（2）给出这个串的一个最右推导：

rm: $S \Rightarrow S\underline{S}^{*} \Rightarrow \underline{S}a^{*} \Rightarrow S\underline{S}+a^{*} \Rightarrow \underline{S}a+a^{*} \Rightarrow aa+a^{*}$

1. 考虑上下文无关文法：$S \rightarrow SS+ \mid SS^* \mid a$ 以及串$aa+a^*$

（3）给出这个串的一颗语法分析树；

以最左推导为例：lm: $S \Rightarrow \underline{S}S^* \Rightarrow \underline{S}S+S^* \Rightarrow a\underline{S}+S^* \Rightarrow aa+S^* \Rightarrow aa+a^*$



2. 考虑上下文无关文法：S→0S1|01

给出该文法的预测分析表（需要先提取左公因子并消除左递归）

提取左公因子（书算法4.10）：

S → 0 A

A → S 1 | 1

该文法没有左递归（书算法4.8，把S的右部带入）：

S → 0 A

A → 0 A 1 | 1

预测分析表：

非终结符号	输入符号		
	0	1	$
S	S → 0 A		
A	A → 0 A 1	A → 1	

3. 考虑上下文无关文法：S→SS+|SS*|a 以及串aa+a*

（1）给出该文法的Fisrt集和Follow集；

First(S) = {a} Follow(S) = {+, *, a}

（2）指出下列最右句型的句柄:

SSS+a*+ 句柄: SS+

SS+a*a+ 句柄: SS+

aaa*a++ 句柄: a

句型：包含非终结符和终结符的串，可以是空串。
最右句型：包含了一个句柄，可以完成一次最右推导。
句柄：和某个产生式体匹配的子串（非正式）

（3）给出串aaa*a++自底向上的解析过程；

3. 考虑上下文无关文法：$S \to SS+ \mid SS^* \mid a$ 以及串 $aa+a^*$

（3）给出串 aaa^*a++ 自底向上的解析过程:

栈	输入	句柄	动作
\$	$aaa^*a++\$$		移入a
\$a	$aa^*a++\$$	a	规约：$S \to a$
\$S	$aa^*a++\$$		移入a
\$Sa	$a^*a++\$$	a	规约：$S \to a$
\$SS	$a^*a++\$$		移入a
\$SSa	$^*a++\$$	a	规约：$S \to a$
\$SSS	$^*a++\$$		移入*
$\$SSS^*$	a++\$	SS^*	规约：$S \to SS^*$
\$SS	a++\$		移入a

栈	输入	句柄	动作
\$SSa	++\$	a	规约：$S \to a$
\$SSS	++\$		移入+
\$SSS+	+\$	SS+	规约：$S \to SS+$
\$SS	+\$		移入+
\$SS+	\$	SS+	规约：$S \to SS+$
\$S	\$		接受

关于句柄的总结:
1. 句柄永远出现在**栈顶**；
2. 句柄永远不会出现在非终结符的左侧；
3. 自下而上解析完全基于**句柄的识别**；

1. 扩展右图中的SDD，使他可以像左图所示那样处理表达式。

产生式	语义规则
1) $L \to E\ \mathbf{n}$	$L.val = E.val$
2) $E \to E_1 + T$	$E.val = E_1.val + T.val$
3) $E \to T$	$E.val = T.val$
4) $T \to T_1 * F$	$T.val = T_1.val \times F.val$
5) $T \to F$	$T.val = F.val$
6) $F \to (E)$	$F.val = E.val$
7) $F \to \mathbf{digit}$	$F.val = \mathbf{digit}.\mathrm{lexval}$

产生式	语义规则
1) $T \to F\,T'$	$T'.inh = F.val$ $T.val = T'.syn$
2) $T' \to * F\,T_1'$	$T_1'.inh = T'.inh \times F.val$ $T'.syn = T_1'.syn$
3) $T' \to \epsilon$	$T'.syn = T'.inh$
4) $F \to \mathbf{digit}$	$F.val = \mathbf{digit}.lexval$

观察可以发现，左图的SDD只包含综合属性，右图的SDD包含继承属性和综合属性。因此扩展即为将左图的+和*改为继承左运算分量的形式。

继承属性的语义规则（书例5.8）：

产生式	语义规则
$T \to FT'$	$T'.inh = F.val$ $T.val = T'.syn$
$T' \to *FT_1'$	$T_1.inh = T'.inh * F.val$ $T.syn = T_1.syn$

1. 扩展右图中的SDD，使他可以像左图所示那样处理表达式。

	产生式	语义规则
1)	$L \to E\ \mathbf{n}$	$L.val = E.val$
2)	$E \to E_1 + T$	$E.val = E_1.val + T.val$
3)	$E \to T$	$E.val = T.val$
4)	$T \to T_1 * F$	$T.val = T_1.val \times F.val$
5)	$T \to F$	$T.val = F.val$
6)	$F \to (E)$	$F.val = E.val$
7)	$F \to \mathbf{digit}$	$F.val = \mathbf{digit}.\text{lexval}$

	产生式	语义规则
1)	$T \to F\,T'$	$T'.inh = F.val$ $T.val = T'.syn$
2)	$T' \to * F\,T'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$
3)	$T' \to \epsilon$	$T'.syn = T'.inh$
4)	$F \to \mathbf{digit}$	$F.val = \mathbf{digit}.lexval$

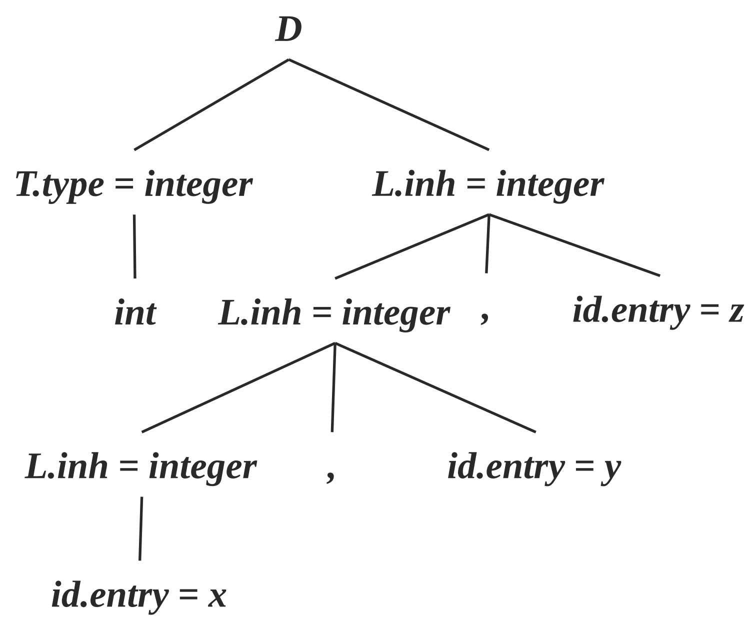
	产生式	语法规则
1)	$L \to En$	$L.val = E.val$
2)	$E \to TE'$	$E'.inh = T.val$ $E.val = E'.syn$
3)	$E' \to +TE_1'$	$E_1.inh = E'.inh + T.val$ $E.syn = E_1.syn$
4)	$E' \to \varepsilon$	$E'.syn = E'.inh$
5)	$T \to FT'$	$T'.inh = F.val$ $T.val = T'.syn$
6)	$T' \to *FT_1'$	$T_1.inh = T'.inh * F.val$ $T.syn = T_1.syn$
7)	$T' \to \varepsilon$	$T'.syn = T'.inh$
8)	$F \to (E)$	$F.val = E.val$
9)	$F \to digit$	$F.val = digit.lexval$

2. 对于图中的SDD，给出int x, y, z对应的注释语法分析树。

产生式	语义规则
1) $D \rightarrow T\ L$	$L.inh = T.type$
2) $T \rightarrow \textbf{int}$	$T.type = \text{integer}$
3) $T \rightarrow \textbf{float}$	$T.type = \text{float}$
4) $L \rightarrow L_1\ ,\ \textbf{id}$	$L_1.inh = L.inh$ $addType(\textbf{id}.entry, L.inh)$
5) $L \rightarrow \textbf{id}$	$addType(\textbf{id}.entry, L.inh)$

L属性定义的SDD，依赖图的边一定是从左到右的！

（书图5-9）的简单改写：



副作用为哑属性，不出现在树中

3. 图中的SDT计算了一个由0和1组成的串的值，它把输入的符号串当做按照正二进制数来解释。改写这个SDT，使得基础文法不再是左递归的，但仍然可以计算出整个输入串的相同的B.val的值:

$$
\begin{array}{lll}
B & \rightarrow & B_1\ 0\ \{B.val = 2 \times B_1.val\} \\
 & | & B_1\ 1\ \{B.val = 2 \times B_1.val + 1\} \\
 & | & 1\ \{B.val = 1\}
\end{array}
$$

SDT消除左递归（书5.4.4节）：

如果不涉及属性值计算，将动作看作终结符进行处理；

如果涉及属性值计算，则通用解决方案为：

产生式	语义规则
$T \rightarrow FT'$	$T'.inh = F.val$ $T.val = T'.syn$
$T' \rightarrow *FT_1'$	$T_1.inh = T'.inh * F.val$ $T.syn = T_1.syn$

- 假设
  - $A \rightarrow A_1 Y \{ A.a = g(A_1.a, Y.y) \}$
  - $A \rightarrow X \{ A.a = f(X.x) \}$
- 那么
  - $A \rightarrow X \{ R.i = f(X.x) \}\ R \{ A.a = R.s \}$
  - $R \rightarrow Y \{ R_1.i = g(R.i, Y.y) \}\ R_1 \{ R.s = R_1.s \}$
  - $R \rightarrow \varepsilon \{ R.s = R.i \}$

3. 图中的SDT计算了一个由0和1组成的串的值，它把输入的符号串当做按照正二进制数来解释。改写这个SDT，使得基础文法不再是左递归的，但仍然可以计算出整个输入串的相同的B.val的值:

$$
\begin{array}{lll}
B & \rightarrow & B_1\ 0\ \{B.val = 2 \times B_1.val\} \\
 & | & B_1\ 1\ \{B.val = 2 \times B_1.val + 1\} \\
 & | & 1\ \{B.val = 1\}
\end{array}
$$

- 假设
  - $A \rightarrow A_1 Y\ \{\ A.a = g(A_1.a,\ Y.y)\ \}$
  - $A \rightarrow X\ \{\ A.a = f(X.x)\ \}$
- 那么
  - $A \rightarrow X\ \{\ R.i = f(X.x)\ \}\ R\ \{\ A.a = R.s\ \}$
  - $R \rightarrow Y\ \{\ R_1.i = g(R.i,\ Y.y)\ \}\ R_1\ \{\ R.s = R_1.s\ \}$
  - $R \rightarrow \varepsilon\ \{\ R.s = R.i\ \}$

基础文法为：$B \rightarrow B_1 0 \mid B_1 1 \mid 1$

不提取左公因子：

消除左递归：$B \rightarrow 1\ A$

$A \rightarrow 0\ A_1 \mid 1\ A_1 \mid \varepsilon$

改写后的SDT：$B \rightarrow 1\ \{A.i = 1\}\ A\ \{B.val = A.val\}$

$A \rightarrow 0\ \{A_1.i = 2 \times A.i\}\ A_1\ \{A.val = A_1.val\}$

$\mid 1\ \{A_1.i = 2 \times A.i + 1\}\ A_1\ \{A.val = A_1.val\}$

$\mid \varepsilon\ \{A.val = A.i\}$

3. 图中的SDT计算了一个由0和1组成的串的值，它把输入的符号串当做按照正二进制数来解释。改写这个SDT，使得基础文法不再是左递归的，但仍然可以计算出整个输入串的相同的B.val的值:

$$
\begin{aligned}
B \quad &\rightarrow \quad B_1\ 0\ \{B.val = 2 \times B_1.val\} \\
&\mid \quad B_1\ 1\ \{B.val = 2 \times B_1.val + 1\} \\
&\mid \quad 1\ \{B.val = 1\}
\end{aligned}
$$

基础文法为：$B \rightarrow B_1\,0 \mid B_1\,1 \mid 1$

提取左公因子的SDT:

$B \rightarrow B_1$ digit $\{B.val = 2\times B_1.val + digit\}$

$\mid 1\ \{B.val = 1\}$

digit $\rightarrow 0\ \{digit.val = 0\}$

$\mid 1\ \{digit.val = 1\}$

消除左递归的SDT:

$B \rightarrow 1\ \{A.i = 1\}\ A\ \{B.val = A.val\}$

$A \rightarrow$ digit $\{A_1.i = 2\times A.i + digit\}\ A_1\ \{A.val = A_1.val\} \mid \varepsilon\ \{A.val = A.i\}$

digit $\rightarrow 0\ \{digit.val = 0\} \mid 1\ \{digit.val = 1\}$

- 假设
  - $A \rightarrow A_1Y\ \{\ A.a = g(A_1.a,\ Y.y)\ \}$
  - $A \rightarrow X\ \{\ A.a = f(X.x)\ \}$
- 那么
  - $A \rightarrow X\ \{\ R.i = f(X.x)\ \}\ R\ \{\ A.a = R.s\ \}$
  - $R \rightarrow Y\ \{\ R_1.i = g(R.i,\ Y.y)\ \}\ R_1\ \{\ R.s = R_1.s\ \}$
  - $R \rightarrow \varepsilon\ \{\ R.s = R.i\ \}$

随堂测试

将下列正则表达式转换成DFA，并将DFA最小化:

(a* | b*)*