# 形式语言与自动机 期中复习.

by XY

4.18

## Lecture 1. 课程概述及预备知识

**Def.** (字母表)($\Sigma$). 形式符号的非空有限集合.

**Def.** (字)(word, w) 即字符串. $\Sigma$中字符构成的有限序列. 空串用$\varepsilon$表示.
字符串 w 的长度记为 $|w|$.

△ 字符串的运算:

连接 concatenation $xy$. $\Rightarrow (xy)z = x(yz)$. $\varepsilon x = x\varepsilon = x$. $|xy| = |x| + |y|$.

△ 字母表的运算. → 长度为 n 的字符串集合

幂运算 (1) $\Sigma^0 = \{\varepsilon\}$ (2) $x \in \Sigma^{n-1}$, $a \in \Sigma \Rightarrow ax \in \Sigma^n$. (3) $\Sigma^n$ 中元素只能由 (1)(2) 生成.

* 闭包. $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$ 为什么叫闭包?

\+ 闭包. $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$ 即 $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$.

**Def.** (语言)(L). 任何集合 $L \subseteq \Sigma^*$ 是字母表 $\Sigma$ 上的一个语言. (Language).

△ 语言的运算. 空语言中并仅含空串的语言 $\{\varepsilon\}$

(1) 并 (union) $L \cup M = \{w \mid w \in L \vee w \in M\}$

(2) 连接 (concatenation) $L \cdot M = \{w_1 w_2 \mid w_1 \in L \wedge w_2 \in M\}$, 通常记为 $LM$.

(1) 闭包 (closure) $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{i \geq 0} L^i$ (句子集)

其中 $L^0 = \{\varepsilon\}$, $L^1 = L$, $L^n = L^{n-1}L$. (很多个语言里的word连起来)

4.19.

## Lecture 2 上下文无关文法与上下文无关语言.

**Def.** (上下文无关文法) T 终结符的集合. → 有限符号集, 即字母表 (小写)

CFG (context-free grammars)
- V 非终结符的集合. → 有限变量符号的集合 (大写)
- S 开始符号 → 一个特殊的非终结符
- P 产生式的集合. → 形如 <head> → <body>

缩写: $S \to 0 \mid 1 \mid 0S1$

$G = (V, T, P, S)$

**Def.** (归约) 将产生式右边 (body) 的符号串替换为左边 (head) 的符号.

**Def.** (推导) 左 符号 右 符号串. $\Rightarrow$, $\overset{*}{\Rightarrow}$

**Def.** (最左(右)推导) 总是换出现在最左边的非终. $\overset{lm}{\Rightarrow}$, $\overset{*}{\overset{lm}{\Rightarrow}}$ $\overset{rm}{\Rightarrow}$, $\overset{*}{\overset{rm}{\Rightarrow}}$

**Def.** (句型) $S \overset{*}{\Rightarrow} \alpha$ 时, $\alpha \in (V \cup T)^*$ 为 G 的一个句型.

$S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$. 左句型 $S \overset{*}{\underset{rm}{\Rightarrow}}$ 右句型. $\alpha \in T^*$. 句子.

**Def.** (上下文无关文法的语言) $L(G) = \{w \mid w \in T^* \wedge S \overset{*}{\underset{G}{\Rightarrow}} w\}$

**Def.** (上下文无关语言) 若语言 L 是某个 CFG G 的语言, 即 $L(G) = L$, 则 L 是 ~.

例9. P39. 给出上下文无关文法. $L = \{w \mid w \in \{a,b\}^*, \text{count}(w,a) = \text{count}(w,b)\}$.

$S \to aSbS \mid bSaS \mid \varepsilon$. (分情况考虑开头, 左→右扫描分割字符串.)

**Prof.** 给定语言 L 是某个文法 G 的语言

P32 { if $w \in L$ then $w \in L(G)$ 归纳 $|w|$
{ if $w \in L(G)$ then $w \in L$ 归纳推导 w 的步数

例13. P60.
例14. P64 互归纳

Def.(文法) 四元组 $G=(V,T,P,S)$. P70

Chomsky 对于生式施加限制，分为 0.1.2.3型文法. $\alpha\to\beta$

· 0型：$\alpha,\beta\in(V\cup T)^*$. $\alpha$ 中至少包含一个非终结符 ⇒ 图灵机

上下文有关 · 1型：$|\alpha|\le|\beta|$（仅 $S\to\varepsilon$ 例外）. S不出现在右部. ⇒ 线性有界自动机

上下文无关 · 2型：$A\to\beta$. $A\in V$. $\beta\in(V\cup T)^*$ ⇒ 下推自动机

正规 · 3型：$A\to aB$ 或 $A\to a$. $A,B\in V$. $a\in T\cup\{\varepsilon\}$. ⇒ 有限状态自动机

Def.(语法分析树) 归约过程自下而上构造. / 推导自上而下. P77

叶节点可是非终/终/$\varepsilon$ ← 必须是父节点唯一的孩子.

Def.(果实) 语法分析树的叶结点从左到右连接. 句型 ⇔ 果实；句子 ⇔ 只含终的果实

Thm.(归约、推导、分析树的关系). P79

(1) w可归约到非终结符A (2) $A\overset{*}{\Rightarrow}w$ (3) $A\overset{*}{\underset{lm}{\Rightarrow}}w$ (4) $A\overset{*}{\underset{rm}{\Rightarrow}}w$

(5) 存在根为A的分析树，果实为w. (1)—(5) 命题等价.

Def.(文法的二义性) 对于某 $w\in T^*$，存在两棵不同的分析树 / 两个不同的从S到w的最左推导.

Thm. CFG是否二义不可判定.

Def.(固有二义) 如果上下文无关语言L的所有文法都是二义的，则称语言L是~.

△ 消除二义性的几种方法.（了）P90-94

1. 优先级联 2 左结合 3. 消除悬挂性

本质上是找到一种唯一的分割方式（对语法而言）

## Lecture 3. 正规表达式与正规语言.

△ 正规语言的运算.

联合 (union) $L\cup M$

连接 (concatenation) $L\cdot M$

(星)闭包 (closure) $L^*$

△ 正规表达式的语法：定义正规表达式集合R. 则 1. $\varepsilon,\phi\in R$ 2. 若 $a\in\Sigma$. 则 $a\in R$ 3. 任一变量 $L\in R$ 4. 若 $E,F\in R$. 则 $E+F$. $EF$. $(E)$. $E^*\in R$.

△ 正规表达式的语言（递归定义. P77）

算符优先级 $*>\cdot>+$

△ 正规语言的归纳定义. 1. 子集和中是~ 2. 若 $a\in\Sigma$. 则 $\{a\}$ 是~ 3. 若 L.R 是~，则 $L\cup R$. $LR$. $L^*$ 是~

利用正表定义：若存在$\Sigma$上的正表 E. 使 $L(E)=R$，则 R 是~

△ 正规表达式的代数定律 trivial

△ 设计正规表达式. 不允许1相邻：$(10+0)^*(\varepsilon+1)$

△ 正规表达式代数定律的具体化.

Thm. E为正规表达式. $L_1,L_2,\cdots L_m$ 为变量（设E不含非变量）. 将 $L_i$ 替换为符号 $a_i$，得具体表达式 C. 则 E 的任何实例语言 $S_1,S_2,\cdots,S_m$. $L(E)$ 中的任何串 w 可写成 $w=w_1w_2\cdots w_k$，其中 $w_i$ 是某语言 $S_{j_i}$ 中的串. 且串 $a_{j_1}a_{j_2}\cdots a_{j_k}\in L(C)$；反过来也成立.

# Lecture 4 有限状态自动机.



有限自动机的五要素：有限状态集、有限输入符号集、转移函数、一个开始状态、一个终态集合.

DFA Def.(确定有限自动机 DFA) deterministic finite automata

五元组 $A=(Q,\Sigma,\delta,q_0,F)$ ; $\delta: Q\times\Sigma\to Q$ ; $q_0\in Q$. $F\subseteq Q$.

转移图表示、转移表表示.

Def.(扩展转移函数 适合于输入字符串) $\delta'(q,\varepsilon)=q$. $\delta'(q,w\overset{x}{a})=\delta(\delta'(q,x),a)$.

△设计 DFA：不包含 ⇒ 包含设计+验证终/非终集合.

垃圾回收站. 注意 DFA 每个节点对每一字符有性出边.

NFA Def.(非确定有限自动机 NFA) nondeterministic finite automata P47

五元组 $A=(Q,\Sigma,\delta,q_0,F)$. 不同：$\delta: Q\times\Sigma\to 2^Q$ 集合.

Def.(扩展转移函数). $\delta'(q,\varepsilon)=\{q\}$. $w=xa\Rightarrow \delta'(q,w)=\bigcup_{i=1}^{k}\delta(p_i,a)$

$\delta'(q,x)=\{p_1,\cdots,p_k\}$

Def.(NFA的语言). $L(A)=\{w\mid w\in\Sigma^*\wedge\delta'(q_0,w)\cap F\neq\phi\}$

对比 DFA的语言 $L(A)=\{w\mid w\in\Sigma^*\wedge\delta'(q_0,w)\in F\}$.

设 L 是 $\Sigma$ 上的语言，如果存在一个 DFA/NFA A 使 $L(A)=L$. 则可说明 L 是正规语言.

△设计 NFA. P54~P58

DFA = NFA Thm.(DFA和NFA的等价性)

L 是某个 DFA 的语言，当且仅当 L 也是某 NFA 的语言.

① DFA → NFA. $\delta_D(q,a)=p\Rightarrow\delta_N(q,a)=\{p\}$

② NFA → DFA. 子集构造法

ε-NFA Def.(带ε-转移的非确定有限自动机) ε-NFA $\delta: Q\times(\Sigma\cup\{\varepsilon\})\to 2^Q$

Def.(ε-闭包)(closure) 记为 ECLOSE(q). 定义为从 q 经所有 ε 路径可以到达的状态(包括q)

Def.(扩展转移函数) $\delta'(q,\varepsilon)=ECLOSE(q)$. 设 $w=xa$. 设 $\delta'(q,x)=\{p_1,\cdots p_k\}$

令 $\bigcup_{i=1}^{k}\delta(p_i,a)=\{r_1,\cdots,r_m\}$. 则 $\delta'(q,w)=\bigcup_{j=1}^{m}ECLOSE(r_j)$

ε-NFA = DFA Thm.(ε-NFA和DFA的等价性)

① DFA → ε-NFA. trivial

② ε-NFA → DFA 修改的子集构造法

- 顶点：所有ε-闭包. $Q_D$ (元素 S)
- 起点：ECLOSE(q) $q_D$
- 终态集合：$F_D$.(与 $F_E$ 交不为空的S)
- $\delta_D(S,a)$. 原转移函数对S中所有点、转移后的取并、再查闭包.

Def.(DFA状态集合上的一个等价关系)

$pRq$ 当 $\forall w\in\Sigma^*$ ( $\delta'(p,w)\in F\leftrightarrow\delta'(q,w)\in F$).

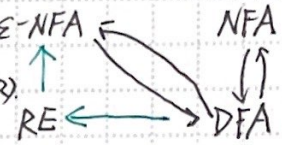
△ 填表法. 计算状态集划分的算法. P106. ① 终态与非终态可区分

② $r\xrightarrow{a}p$, $s\xrightarrow{a}q$ 若 p,q 可区分 ⇒ r,s 可.

△ DFA的最小化. P109

1. 删除所有从开始状态不可到达的状态 2. 填表法找等价类 3. 构造新DFA

Lecture 5. 有限状态自动机 ⟺ 正规表达式.




Thm (正规表达式 → ε-NFA).

L为正表R表示的语言. 则存在一ε-NFA E, 满足 L(E)=L(R).

证: 归纳构造出一满足如下条件的ε-NFA.

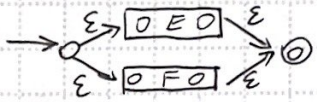
(1) 恰好一个终态. (2) 没有弧进入初态 (3) 没有弧离开终态.

P5 Thompson 构造法. 基础 ① ε. → ○ —ε→ ◎

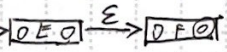
② ϕ → ○ ◎

③ a → ○ —a→ ◎

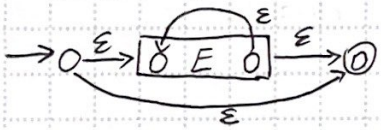
归纳 ① E+F




② EF




③ E*




Thm (DFA → 正规表达式).

(1) 路径迭代法 (Kleene构造法). P18

1) 编号状态 1-n 2) $R_{ij}^{(k)}$ i到j的所有经过顶点编号≤k路径的RE (不包括i,j)

ans: 把所有 $R_{1j}^{(n)}$ 相"+". (1为初态, j为所有终态)

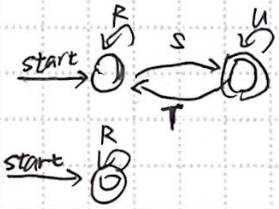
△ k=0. i→j 不存在: ϕ, a: a, 多条 $a_1 \cdots a_m$: $a_1+\cdots+a_m$. i=j时 +ε!!

△ 归纳 $R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \cdot (R_{kk}^{(k-1)})^* \cdot R_{kj}^{(k-1)}$

(2) 状态消去法 P23 对每一对前驱和后继改RE

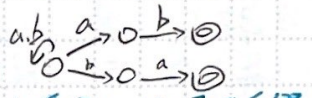
1) 对每个终态 $q_i$, 消到只剩 $q_1$ 和 $q_i$, 最后相"+"

2) 消到剩2个时: $(R+SU^*T)^*SU^*$ or $R^*$



注意事项

1. NFA → DFA 可能有 $\phi, \phi, \phi \cdots$ 的一行. 记得标 → (只有1个) 和 * (可能多个)
2. Thompson 构造法记得标 start → 和 ◎.
3. 构造正则表达式. a次数能除k可以用 $(\underbrace{aa\cdots a}_{k\text{个}a})^*$ 表式. (注意偶数).
4. 构造DFA时要注意试一下各种情况是否能被接收. (空串. 单字. 满足条件后…)
5. DFA最小化时要删掉不可到达的点!!!
6. 没有连续的1: $(0+01)^*0 \Rightarrow (1+\varepsilon)(0+01)^*$ 或者 $(0+10)^*(1+\varepsilon)$
7. ε-NFA 转 DFA 时先写 ECLOSE, 起点为 ECLOSE($q_0$), 每次把得到集合再套一层 ECLOSE. 加之后
   每个顶点的 ECLOSE 不一定在最终 DFA 中.
8. 识别关键字集合 NFA → DFA 书写.
   a) $q_0$ 为初态 → $\{q_0\}$ 是一个状态.
   b) p 是 NFA 的一个状态, 沿 $a_1a_2\cdots a_m$ 路径可从 $q_0$ 走到 p. 先循环 $a_1\cdots a_j$ 再走后缀
      则 $q_0$, p, 还有每一个从 $q_0$ 出发沿 $a_1a_2\cdots a_m$ 的某后缀可达的所有状态.




9. 最小化表格 2 3 4 5 先注意点!
10. 路径迭代法. 列表 (对1 2 3 4 每个k) 填时先标出具体的k, 以避免出错!
    只看 k-1 表格! 不要看错到再之前的. 随时检查图对表 (肉眼正则)
11. 看 Thompson 时不要把 $\circ \xrightarrow{a} \circ \xrightarrow{\varepsilon} \circ \xrightarrow{b} \circ$ 混成 a+b, 这是连接, 是 ab!