第 3 章 有穷状态自动机

第 2 章从文法的角度，也就是从语言生成的角度研究语言。除了文法之外，也可以用不同的识别模型，从识别的角度研究语言。本章讨论的是正则语言的识别模型——有穷状态自动机（FA）。主要内容有确定的有穷状态自动机（DFA），不确定的有穷状态自动机（NFA），带空移动的有穷状态自动机（ε-NFA），带输出的有穷状态自动机，以及双向有穷状态自动机。

关于 DFA，介绍如何从实际问题抽象出 DFA，DFA 的直观物理模型。按照这种抽象，给出 DFA 的形式定义，它接受的句子、语言状态转移图等重要概念。如何构造识别给定语言的 DFA 是本章难点之一。本章通过例子介绍 DFA 的构造。

NFA 是 DFA 的一种变形。为了构造方便，从基本定义上对 DFA 进行了扩充。但是在扩充之后，它仍然与 DFA 等价。

ε-NFA 是对 NFA 的进一步扩充。扩充后得到的 ε-NFA 与 DFA 也是等价的。

DFA，NFA，ε-NFA 是 FA 的 3 种等价形式，它们是正则语言的识别器。在 3.5 节中，专门讨论正则文法（RG）与 FA 的等价性及其相互转换方法。

带输出的 FA 和双向 FA 是 FA 的另一些变形。其中，带输出的 FA 可以根据状态或者移动输出信息，而双向 FA 既允许 FA 的读头向右移动，又允许读头向左移动。

由于和 FA 相关的内容都是以 DFA 的概念为基础的，而且让初学者把一个模型——形式化模型当成一个自动机去理解，在心理上会遇到较大的障碍，所以，在上述所列的内容中，DFA 的概念不仅是重点，而且还是难点。对 DFA 概念的突破，是对其他内容理解的重要基础。本章的第二个重点内容为 DFA，NFA，ε-NFA，RG 之间的等价转换思路与方法。这些内容，不仅再次讨论了形式模型之间的等价转换问题，而且体现了对这些形式模型更深层的理解，甚至能使读者体验到这些形式模型之间等价变换思想的闪光，并且体验大师们当初发现这些等价转换方法尤其是 FA 与 RG 的等价转换方法所获得的乐趣。

除了对 DFA 概念的理解之外，DFA，RG 的构造方法，RG 与 FA 的等价性证明等也是比较难以掌握的。

3.1 语言的识别

知识点

(1) 推导和归约中的回溯问题将对系统的效率产生极大的影响。

(2) 作为语言的识别模型,需要5个要素,该模型的每个移动由3个节拍组成:读入它正注视的字符,根据当前状态和读入的字符改变有穷控制器的状态,将读头向右移动一格。

(3) 相应的物理模型为:一个右端无穷的输入带,一个有穷状态控制器,一个读头。

3.2 有穷状态自动机

1. 知识点

(1) FA是一个五元组 $M=(Q,\Sigma,\delta,q_0,F)$;由于对任意的 $(q,a)\in Q\times\Sigma$,Q 中有唯一的状态 p 与 $\delta(q,a)$ 对应,所以称之为DFA。

(2) 虽然 δ 的定义域 $Q\times\Sigma$ 是 $\hat{\delta}$ 的定义域 $Q\times\Sigma^*$ 的真子集,但对于任意的 $(q,a)\in Q\times\Sigma$,$\hat{\delta}$ 和 δ 有相同的值,所以不用区分这两个符号。

(3) $L(M)=\{x\mid x\in\Sigma^*$ 且 $\delta(q,x)\in F\}$ 为 M 接受(识别)的语言。

(4) FA的状态转移图。

(5) DFA的等价。

(6) DFA的ID及ID之间的关系 $\vdash_M$。

(7) DFA的状态只具有有穷记忆能力。

(8) DFA的每个状态记忆集合 $set(q)=\{x\mid x\in\Sigma^*,\delta(q_0,x)=q\}$,这使得DFA $M=(Q,\Sigma,\delta,q_0,F)$ 自然地对应于等价关系 R_M,该等价关系将 Σ^* 分成有穷多个等价类,对于每一个可达状态 q,$set(q)$ 就是 q 对应的等价类。

(9) 陷阱状态。

(10) 将要记忆的内容用作状态的标识,有利于一类DFA的表示和构造。

2. 主要内容解读

定义3-1 有穷状态自动机(finite automaton,FA)M 是一个五元组:

$$M=(Q,\Sigma,\delta,q_0,F)$$

其中,

Q——状态的非空有穷集合。$\forall q\in Q$,q 称为 M 的一个**状态**(state)。

Σ——**输入字母表**(input alphabet)。输入字符串都是 Σ 上的字符串。

δ——**状态转移函数**(transition function),有时又叫做状态转换函数或者移动函数。δ:$Q\times\Sigma\to Q$,对 $\forall(q,a)\in Q\times\Sigma$,$\delta(q,a)=p$ 表示 M 在状态 q 读入字符 a,将状态变成 p,并将读头向右移动一个带方格而指向输入字符串的下一个字符。

q_0——M 的**开始状态**(initial state),也可叫做初始状态或者启动状态,$q_0 \in Q$。

F——M 的**终止状态**(final state)集合,$F \subseteq Q$。$\forall q \in F$,q 称为 M 的**终止状态**。终止状态又称为**接受状态**(accept state)。

对此定义的理解,需要注意以下几个方面:

(1) 要求状态集 Q 是非空有穷的,这和要求一个字母表是非空有穷的原因是类似的。实际上,希望 FA 描述的是一个语言的处理系统,而每个系统至少要有一个状态。当然,如果该系统具有无穷多个状态的话,它就不是当代计算机系统可以"模拟"的了。因此 Q 必须是有穷的。

(2) 状态转移函数 $\delta: Q \times \Sigma \to Q$,表明对任意的 $(q,a) \in Q \times \Sigma$,$Q$ 中有唯一的状态 p 与 (q,a) 对应,所以,为了和后面定义的其他类型的 FA 相区别,才称这里定义的 FA 为 DFA (deterministic finite automaton)。初学者最容易忽略的问题之一是当检查用状态转移图表示的 FA 是否为 DFA 时,有可能忽略掉 $\delta(q,a)$ 不对应任何状态这一不满足要求的情况。尤其是在后面讲过 NFA 后,更容易错误地认为,只要不存在 $(q,a) \in Q \times \Sigma$,使得 $|\delta(q,a)| > 1$,相应的 FA 就是 DFA 了。

(3) 开始状态就是 FA 在正常情况下,开始处理一个输入串的状态。在实际系统的实现中,当需要此 FA 再处理另外一个输入串时,它必须重新从这个状态启动,而不管它在处理完上一个串后处于哪个状态。

(4) 虽然 F 中的状态称为终止状态,并不是说 M 一旦进入这种状态就终止了,而是说,一旦 M 在处理完输入字符串时到达这种状态,M 就接受当前处理的字符串。在 FA 处理一个输入串的过程中,它可能"经过"某些终止状态,但是,如果它在处理完这个输入串后未停在终止状态,则此串不被 FA"认可"。也就是说,FA 在决定是否"认可"一个输入串时,并不考虑它是否曾经在中途经过某一个终止状态,而是根据它处理完该字符串时的当前状态来决定的。据此,将这类状态称为接受状态也许更恰当。

例 3-1 给出了两个 FA,一方面用来作为两个实例,另一方面是给出 FA 的状态转移函数的表达方式。由于现在还没有定义什么是 FA 接受的语言,所以,现在还不能讨论这两个 FA 能识别什么语言。但是,如果在给出 FA 接受的语言的定义后再行举例,新的概念就显得在一起堆积得太多,不利于学习。

由于语言是由句子,也就是满足一定条件的串组成的,所以,为了定义 FA 识别的语言,必须先将 δ 的定义域从 $Q \times \Sigma$ 扩充到 $Q \times \Sigma^*$ 上,以便于讨论 FA 是如何处理字符串的。然后,考虑如何表示组成语言的句子所需要满足的条件就容易了。对此扩展,读者需要弄明白为什么不在一开始就直接用 δ 表示扩展的状态转移函数,而是要在证明了对于任意的 $(q,a) \in Q \times \Sigma$,$\hat{\delta}$ 和 δ 有相同的值后方可以用同一个符号表示。

将 δ 扩充为 $\hat{\delta}: Q \times \Sigma^* \to Q$ 使用的是递归定义,这也是与 FA 处理字符串的过程一致的,从而也给今后研究 FA 处理字符串做了一定的准备。这个定义为:对任意的 $q \in Q$,$w \in \Sigma^*$,$a \in \Sigma$:

(1) $\hat{\delta}(q,\varepsilon) = q$。

(2) $\hat{\delta}(q,wa) = \delta(\hat{\delta}(q,w),a)$。

其中,第一条是基础,表示 FA 在不读入任何字符时将保持在原来的状态不变;第二条表示

对一个串 wa,FA 必须处理完 w 后再处理 a,而且处理 a 时所处的状态是处理完 w 后所处的状态。进一步对 w 展开这个式子,它表示 FA 是从左到右处理输入串中的字符,该 FA 从状态 q 开始,首先在状态 q 下处理第一个字符,然后进入一个新状态,再在这个新状态处理下一个字符,如此下去,直到串中所有字符都被处理完。设 $x=a_1a_2\cdots a_n$,以此形式,这个过程可以表示为

$$\begin{aligned}&\delta(q,a_1a_2\cdots a_n)\\&=\delta(\delta(q,a_1a_2\cdots a_{n-1}),a_n)\\&=\delta(\delta(\delta(q,a_1a_2\cdots),a_{n-1}),a_n)\\&\vdots\\&=\delta(\delta(\delta(\delta(\delta(q,a_1),a_2),\cdots),a_{n-1}),a_n)\end{aligned}$$

不妨设

$$\delta(q,a_1)=q_1,\delta(q_1,a_2)=q_2,\cdots,\delta(q_{n-2},a_{n-1})=q_{n-1},\delta(q_{n-1},a_n)=q_n$$

则

$$\begin{aligned}&\delta(q,a_1a_2\cdots a_n)\\&=\delta(q_1,a_2\cdots a_n)\\&=\delta(q_2,a_3\cdots a_n)\\&\vdots\\&=\delta(q_{n-2},a_{n-1}a_n)\\&=\delta(q_{n-1},a_n)\\&=q_n\end{aligned}$$

如果按照 ID 的表示方法,这个过程可以表示为

$$qa_1a_2\cdots a_n\vdash a_1q_1a_2\cdots a_n\vdash a_1a_2\ q_2a_3\cdots a_n\vdash\cdots\vdash a_1a_2\cdots q_{n-2}a_{n-1}a_n\vdash a_1a_2\cdots a_{n-1}q_{n-1}a_n\vdash a_1a_2\cdots a_n\ q_n$$

实际上,q_i 左侧的字符串是不会再被扫描的,所以,它们是可以不在 ID 中反映的。另外,被 FA 处理过的 $a_1a_2\cdots a_n$ 的前缀 $a_1a_2\cdots a_i$ 的对应后缀 $a_{i+1}\cdots a_n$ 正好是 FA 从状态 q_i 开始待处理的内容。

定义 3-2 设 $M=(Q,\Sigma,\delta,q_0,F)$ 是一个 FA。对于 $\forall x\in\Sigma^*$,如果 $\delta(q,w)\in F$,则称 x 被 M 接受,如果 $\delta(q,w)\notin F$,则称 M 不接受 x。

$$L(M)=\{x\mid x\in\Sigma^*\text{ 且 }\delta(q,w)\in F\}$$

称为由 M 接受(识别)的语言。

这个定义也可以叙述为:能引导 FA 从启动状态出发,最终到达终止状态的字符串是 FA 识别的语言的句子。所有这样的字符串组成的集合为 FA M 接受的语言,记作 $L(M)$。

定义 3-3 设 M_1,M_2 为 FA。如果 $L(M_1)=L(M_2)$,则称 M_1 与 M_2 等价。

这个定义说明,判定两个 FA 是否等价的标准是它们接受的语言是否相同。这和判定两个文法是否等价一样。另外,表示一个语言的文法可以有多个,识别一个语言的 FA 也可以有多个:如果有一个 FA 能识别语言 L,那么也可能存在“许许多多”的识别 L 的 FA。

例 3-2 是真正按照规定的语言构造的第一个 DFA,这个 DFA 的构造虽然比较简单,但是却表现出构造 DFA 的一个非常重要的方法,这就是 DFA 的每一个状态表达不同的意思。当 DFA 到达不同的状态时,表明 DFA 当前扫描过的字符串的前缀已经具有语言要求

的句子的某种结构。由于 DFA 只有有穷个状态，所以，DFA 的状态只具有有穷记忆能力。这就预示着，如果一个语言的结构特征中只有有穷多种需要记忆的信息，则能构造出接受它的 DFA 来。

状态转移图是 DFA 的一个非常直观的表示，当熟悉这种表示时，通过画出接受一个语言的 DFA 的状态转移图来完成这个 DFA 的构造是最为方便的。由于状态转移图具有直观性，而这种直观性非常便于人们对其表达内容的理解，所以，对人们来说，使用这种表示方法进行“规模”不是特别大的 DFA 的构造是最为方便和有效的。所以，往往只画出 DFA 的状态转移图就可以了。FA 的状态转移图的定义如下。

定义 3-4 设 $M=(Q,\Sigma,\delta,q_0,F)$为一个 FA，满足如下条件的有向图被称为 M 的状态转移图(transition diagram)：

(1) $q\in Q \Leftrightarrow q$ 是该有向图中的一个顶点。

(2) $\delta(q,a)=p\Leftrightarrow$图中有一条从顶点 q 到顶点 p 的标记为 a 的弧。

(3) $q\in F\Leftrightarrow$标记为 q 的顶点被用双层圈标出。

(4) 用标有 S 的箭头指出 M 的开始状态。

状态转移图又可以叫做状态转换图。有时，状态转移图中会存在一些并行的弧：它们从同一顶点出发，到达同一个顶点。对这样的并行弧，可以用一条有多个标记的弧表示。

通过例 3-3 DFA 的构造，总结出了以下几点注意事项：

(1) 定义 FA 时，只给出 FA 相应的状态转移图就可以了。

(2) 对于 DFA 来说，并行的弧按其上标记字符的个数计算，对于每个顶点来说，它的出度恰好等于输入字母表中所含字符的个数。这一点提醒读者，第一，当完成一个 DFA 的构造后，至少需要检查一下，看每个顶点的出度是否正确；第二，在构造 DFA——画 DFA 的状态转移图时，需要逐个状态地考虑该状态对应字母表中的每个字母的转移方向，这在许多时候会给构造提供启发。

(3) 字符串 x 被 FA M 接受的充分必要条件是在 M 的状态转移图中存在一条从开始状态到某一个终止状态的有向路，该有向路上从第一条边到最后一条边的标记依次并置而构成的字符串 x。此路的标记简称为 x。

(4) 一个 FA 可以有多个终止状态。实际上，每个终止状态代表语言中句子的不同分类。例如，主教材图 3-5 的终止状态 q_3 和 q_4 分别代表语言$\{x000|x\in\{0,1\}^*\}\cup\{x001|x\in\{0,1\}^*\}$的以 000 结尾的句子组成的类和以 001 结尾的句子组成的类。

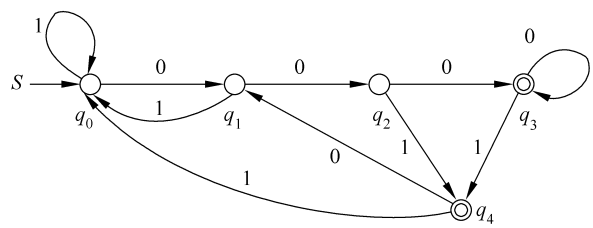

图 3-5 接受语言$\{x000|x\in\{0,1\}^*\}\cup\{x001|x\in\{0,1\}^*\}$的 DFA

FA 的即时描述及其相互之间的关系⊢提供了描述 FA 识别一个输入串的更为方便的过程，其定义如下。

定义 3-5 设 $M=(Q,\Sigma,\delta,q_0,F)$为一个 FA,$x,y\in\Sigma^*$,$\delta(q_0,x)=q$,$xqy$ 称为 M 的一个**即时描述**(instantaneous description,ID)。它表示 xy 是 M 正在处理的一个字符串,x 引导 M 从 q_0启动并到达状态 q,M 当前正指向 y 的首字符。

如果 $xqay$ 是 M 的一个即时描述,且 $\delta(q,a)=p$,则

$$xqay \vdash_M xapy$$

表示 M 在状态 q 时已经处理完 x 并且正指向输入字符 a,此时它读入 a 并转入状态 p,将读头向右移动一格指向 y 的首字符。

显然,与⇒是文法的变量集和终极符号集的并集的克林闭包上的与产生式相关的二元关系类似,$\vdash_M$为 M 的所有 ID 集上的二元关系,它与 M 定义的移动有关,而且可用 $\vdash_M^n$,$\vdash_M^*$,$\vdash_M^+$ 分别表示$\left(\vdash_M\right)^n$,$\left(\vdash_M\right)^*$,$\left(\vdash_M\right)^+$,当意义明确时,也可以直接用 $\vdash$,$\vdash^n$,$\vdash^*$,$\vdash^+$表示$\left(\vdash_M\right)^n$,$\left(\vdash_M\right)^*$,$\left(\vdash_M\right)^+$。

$\alpha \vdash_M^n \beta$:表示 M 从 ID α 经过 n 次移动到达 IDβ。即 M 存在 ID 序列 $\alpha_1,\alpha_2,\cdots,\alpha_{n-1}$,使得 $\alpha \vdash_M \alpha_1,\alpha_1\vdash_M \alpha_2,\cdots,\alpha_{n-1}\vdash_M \beta$。

当 $n=0$ 时,有 $\alpha=\beta$。即 $\alpha \vdash_M^0 \alpha$。

$\alpha \vdash_M^+ \beta$:表示 M 的 ID 从 α 经过至少一次移动变成 β。

$\alpha \vdash_M^* \beta$:表示 M 的 ID 从 α 经过若干步移动变成 β。

需要注意,在第 9 章图灵机中,也定义有类似的 ID,只不过在图灵机中,q_i左侧的字符串(即 $a_1a_2\cdots a_n$的前缀)是有可能被再次扫描的,而在这里它们是不会被再次扫描的。

定义 3-6 设 $M=(Q,\Sigma,\delta,q_0,F)$为一个 FA,对 $\forall q\in Q$,能引导 FA 从开始状态到达 q 的字符串的集合为

$$set(q)=\{x\mid x\in\Sigma^*,\delta(q_0,x)=q\}$$

这个定义的引入,是为了明确地将 DFA 的状态和 DFA 所确定的关于 Σ^* 的等价类对应起来,以深入理解 DFA。根据这种对应关系,任意一个 FA $M=(Q,\Sigma,\delta,q_0,F)$就自然有了等价关系 R_M。

对 $\forall x,y\in\Sigma^*$,$xR_My \Leftrightarrow \exists q\in Q$,使得 $x\in set(q)$和 $y\in set(q)$同时成立。

之所以说利用这个关系可以将 Σ^* 划分成不多于$|Q|$个等价类,主要是因为 DFA 中可能有不可达状态。但是,对 NFA,这个结论是不成立的。

例 3-4 给出了一种根据“主体框架”构造 DFA 的思路,同时引入了陷阱状态的概念及其用法。

通过例 3-4 和例 3-5 进一步表明如何利用状态的有穷存储功能构造 DFA,尤其是在例 3-5 中,先分析出可以按照同余类来分等价类这一关键点,然后将等价类(同余类)与状态对应起来,并考虑用 3 所确定的输入串所处的等价类及其等价类之间的变换关系,来确定相应的 DFA 的状态之间应该如何转移。

例 3-6 将要记忆的内容用作对状态的标识，这种方法在表示一类 DFA 及其构造思想时是非常有效的。

$q[\varepsilon]$——M 还未读入任何字符。

q_t——陷阱状态。

$q[a_1a_2\cdots a_i]$——M 记录有 i 个字符，$1\leqslant i\leqslant 5$。其中，$a_1,a_2,\cdots,a_i\in\{0,1\}$。

3.3 不确定的有穷状态自动机

3.3.1 作为对 DFA 的修改

知识点

希望对于所有的$(q,a)\in\Sigma\times Q$，$\delta(q,a)$对应的是 Q 的一个子集。并认为这种扩展了的“FA”在于允许它在任意时刻可以处于有穷多个状态，这样一来，则有

(1) 并不是对于所有的$(q,a)\in\Sigma\times Q$，$\delta(q,a)$都有一个状态与它对应。

(2) 并不是对于所有的$(q,a)\in\Sigma\times Q$，$\delta(q,a)$只对应一个状态。

Q 的有穷性保证了 2^Q的有穷，使得这种“FA”仍然是具有有穷个状态的。

也可以将这种扩充看成是允许“FA”具有“智能”，该“智能”足以使 FA 在一个状态下，根据当前读入的字符在集合 $\delta(q,a)$中选择一个正确的状态。

3.3.2 NFA 的形式定义

1. 知识点

(1) NFA 与 DFA 相比，只是状态转移函数的值域由 Q 变成了 2^Q。

(2) DFA 是 NFA 的特例。

(3) x 被 M 接受，当且仅当 $\delta(q_0,x)\cap F\neq\varnothing$。

2. 主要内容解读

定义 3-7 不确定的有穷状态自动机(non-deterministic finite automaton，NFA)M 是一个五元组：

$$M=(Q,\Sigma,\delta,q_0,F)$$

其中，

Q,Σ,q_0,F 的意义同 DFA。

δ——**状态转移函数**(transition function)，又叫**状态转换函数**或者**移动函数**。$\delta:Q\times\Sigma\to 2^Q$，对 $\forall(q,a)\in Q\times\Sigma$，$\delta(q,a)=\{p_1,p_2,\cdots,p_m\}$表示 M 在状态 q 读入字符 a，可以有选择地将状态变成 $p_1,p_2,\cdots$，或者 p_m，并将读头向右移动一个带方格而指向输入字符串的下一个字符。

原来为 DFA 定义的状态转移图、状态对应的等价类、即时描述等对 NFA 都是有效的。

与 DFA 相同，将 δ 扩充为 $\hat{\delta}:Q\times\Sigma^*\to 2^Q$。对任意的 $q\in Q$，$w\in\Sigma^*$，$a\in\Sigma$：

(1) $\hat{\delta}(q,\varepsilon)=\{q\}$。

(2) $\hat{\delta}(q,wa)=\{p\mid \exists r\in\hat{\delta}(q,w)$,使得 $p\in\delta(r,a)\}$。

而且由于对于任意的 $q\in Q,a\in\Sigma$:

$$\hat{\delta}(q,a)=\delta(q,a)$$

所以,可以用 δ 代替 $\hat{\delta}$。

由于对 $\forall(q,w)\in Q\times\Sigma^*$,$\delta(q,w)$是一个集合,因此,为了叙述方便,才进一步扩充 δ 的定义域:$\delta:2^Q\times\Sigma^*\to 2^Q$。对任意的 $P\subseteq Q,w\in\Sigma^*$,

$$\delta(P,w)=\bigcup_{q\in P}\delta(q,w)$$

考虑到对 $\forall(q,w)\in Q\times\Sigma^*$:

$$\delta(\{q\},w)=\delta(q,w)$$

所以,可以不区分 δ 的第一个分量是一个状态还是一个状态集合。于是,对任意的 $q\in Q,w\in\Sigma^*,a\in\Sigma$,可以得到

$$\delta(q,wa)=\delta(\delta(q,w),a)$$

从而,对输入字符串 $a_1a_2\cdots a_n$有

$$\delta(q,a_1a_2\cdots a_n)=\delta((\cdots\delta(\delta(q,a_1),a_2),\cdots),a_n)$$

定义 3-8 设 $M=(Q,\Sigma,\delta,q_0,F)$是一个 NFA。对于 $\forall x\in\Sigma^*$,如果 $\delta(q_0,x)\cap F\neq\varnothing$,则称 x 被 M 接受,如果 $\delta(q_0,x)\cap F=\varnothing$,则称 M 不接受 x。

$$L(M)=\{x\mid x\in\Sigma^*\text{ 且 }\delta(q_0,x)\cap F\neq\varnothing\}$$

称为由 M 接受(识别)的语言。

注意:对于 $\forall x\in\Sigma^*$,$\delta(q_0,x)$ 是一个集合,所以,在判定字符串 x 是否被 M 接受时,要看 $\delta(q_0,x)$ 中是否含有终止状态,即判定 $\delta(q_0,x)\cap F\neq\varnothing$是否成立。按照状态转移图来说,就是要看在 M 的状态转移图中是否存在从 q_0 出发到达某一个终止状态的标记为 x 的路。

3.3.3 NFA 与 DFA 等价

1. 知识点

(1) NFA 与 DFA 等价是指这两种模型识别相同的语言类。

(2) DFA 对 NFA 的模拟的关键是 DFA 用一个状态去对应 NFA 的一组状态:$[p_1,p_2,\cdots,p_m]\Leftrightarrow\{p_1,p_2,\cdots,p_m\}$。

(3) 不可达状态是无意义的。

(4) 根据 NFA 构造等价的 DFA 时,可以只给出 DFA 的所有可达状态,从而可以直接略去关于不可达状态相关的计算。

2. 主要内容解读

定理 3-1 NFA 与 DFA 等价。

证明要点:

(1) 构造。

设 NFA $M_1=(Q,\Sigma,\delta_1,q_0,F_1)$。

取 DFA $M_2=(Q_2,\Sigma,\delta_2,[q_0],F_2)$。

$Q_2=2^Q$，只是暂时将用“{”和“}”把集合的元素汇集成整体的习惯写法改为用“[”和“]”把集合的元素汇集成整体。

$F_2=\{[p_1,p_2,\cdots,p_m]\mid\{p_1,p_2,\cdots,p_m\}\subseteq Q$ 且 $\{p_1,p_2,\cdots,p_m\}\cap F_1\neq\varnothing\}$

$\delta_2([q_1,q_2,\cdots,q_n],a)=[p_1,p_2,\cdots,p_m]\Leftrightarrow\delta_1(\{q_1,q_2,\cdots,q_n\},a)=\{p_1,p_2,\cdots,p_m\}$

这实际上给出了构造给定 NFA 的等价 DFA 的一般方法。显然，这种转化方法是可以“被自动化”的。由于这种构造方法的一般性，所以，在新构造出的 DFA 中，可能存在不可达的状态。例 3-7 给出了相应的实例。

由于不可达状态对语言的识别是无用的，所以，需要将它们删除。最好是直接避免因不可达状态所导致的无用计算。方法是从开始状态$[q_0]$出发，考察 Σ 中的所有符号，逐步地计算出所有可达的状态及其相应的转移。例 3-7 给出的是逐步在表中填写表达状态转移函数值的方法。

(2) 对任意 $x\in\Sigma^*$，施归纳于$|x|$，证明 $\delta_1(q_0,x)=\{p_1,p_2,\cdots,p_m\}\Leftrightarrow\delta_2([q_0],x)=[p_1,p_2,\cdots,p_m]$。

(3) 证明 $L(M_1)=L(M_2)$。

(2)和(3)两步表明了(1)中所给的构造方法的正确性。因此，今后可以直接用此方法构造给定 NFA 的等价 DFA，不用再对构造的结果进行证明。

例 3-7 给出的直接省去不可达状态相关计算的思路，可以用到直接根据 NFA 的状态转移图构造 DFA 的状态转移图中。在这里，所有的与不可达状态相关的顶点和弧，都不用在状态转移图中出现。

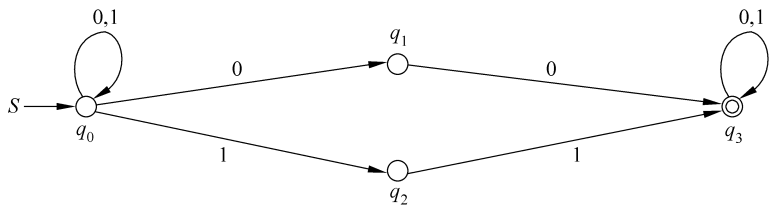

图 3-9　希望是接受$\{x\mid x\in\{0,1\}^*$，且 x 含有子串 00 或 11$\}$的 FA

下面仍然以图 3-9 所示的 NFA 为例，直接画出与其等价的 DFA 的状态转移图。相应的构造过程如下：

(1) 先画出只含开始状态$[q_0]$的图 3-9(a)。

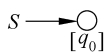

图 3-9(a)　从开始状态$[q_0]$开始

(2) 考察状态$[q_0]$关于 0 和 1 的转移。

因为 $\delta(\{q_0\},0)=\{q_0,q_1\}$，所以，在图中增加标记为$[q_0,q_1]$的顶点，并从标记为$[q_0]$的顶点到标记为$[q_0,q_1]$的顶点画一条标记为 0 的弧。

因为 $\delta(\{q_0\},1)=\{q_0,q_2\}$，所以，在图中增加标记为$[q_0,q_2]$的顶点，并从标记为$[q_0]$的顶点到标记为$[q_0,q_2]$的顶点画一条标记为 1 的弧。此时可得图 3-9(b)。

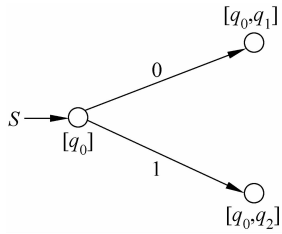

图 3-9(b)　增加在$[q_0]$状态下的移动

(3) 考察状态$[q_0,q_1]$关于 0 和 1 的转移。

因为 $\delta(\{q_0,q_1\},0)=\{q_0,q_1,q_3\}$,所以,在图中增加标记为$[q_0,q_1,q_3]$的顶点,并从标记为$[q_0,q_1]$的顶点到标记为$[q_0,q_1,q_3]$的顶点画一条标记为 0 的弧。

因为 $\delta(\{q_0,q_1\},1)=\{q_0,q_2\}$,所以,在图中标记为$[q_0,q_1]$的顶点到标记为$[q_0,q_2]$的顶点画一条标记为 1 的弧。此时,可得图 3-9(c)。

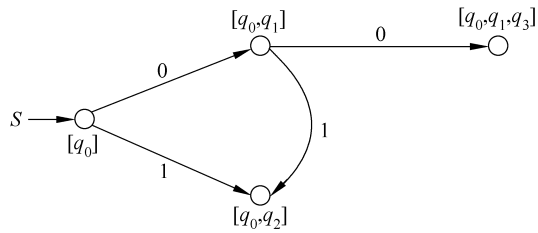

图 3-9(c)　增加在$[q_0,q_1]$状态下的移动

(4) 考察状态$[q_0,q_2]$关于 0 和 1 的转移。

因为 $\delta(\{q_0,q_2\},0)=\{q_0,q_1\}$,所以,从标记为$[q_0,q_2]$的顶点到标记为$[q_0,q_1]$的顶点画一条标记为 0 的弧。

因为 $\delta(\{q_0,q_2\},1)=\{q_0,q_2,q_3\}$,所以,将标记为$[q_0,q_2,q_3]$的顶点添入图中,并从标记为$[q_0,q_2]$的顶点到标记为$[q_0,q_2,q_3]$的顶点画一条标记为 1 的弧。此时,可得图 3-9(d)。

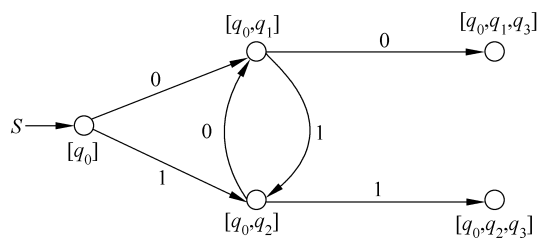

图 3-9(d)　增加在$[q_0,q_2]$状态下的移动

(5) 考察状态$[q_0,q_1,q_3]$关于 0 和 1 的转移。

因为 $\delta(\{q_0,q_1,q_3\},0)=\{q_0,q_1,q_3\}$,所以,从标记为$[q_0,q_1,q_3]$的顶点画一条到自身的标记为 0 的弧。

因为 $\delta(\{q_0,q_1,q_3\},1)=\{q_0,q_2,q_3\}$,所以,从标记为$[q_0,q_1,q_3]$的顶点到标记为$[q_0,q_2,q_3]$的顶点画一条标记为 1 的弧。此时,可得图 3-9(e)。

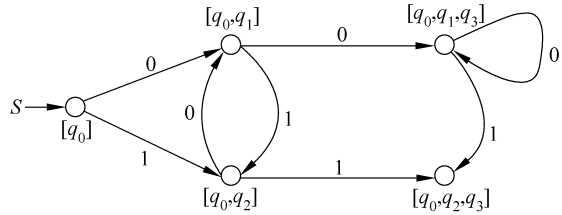

图 3-9(e)　增加在$[q_0,q_1,q_3]$状态下的移动

(6) 考察状态$[q_0,q_2,q_3]$关于 0 和 1 的转移。

因为$\delta(\{q_0,q_2,q_3\},0)=\{q_0,q_1,q_3\}$，所以，从标记为$[q_0,q_2,q_3]$的顶点到标记为$[q_0,q_1,q_3]$的顶点画一条标记为 0 的弧。

因为$\delta(\{q_0,q_2,q_3\},1)=\{q_0,q_2,q_3\}$，所以，从标记为$[q_0,q_2,q_3]$的顶点画一条到自身的标记为 0 的弧。此时，可得图 3-9(f)。

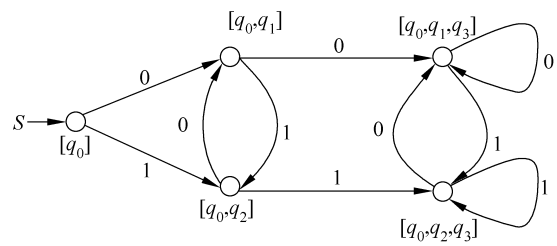

图 3-9(f)　增加在$[q_0,q_2,q_3]$状态下的移动

注意到q_3是图 3-9 所示的 NFA 的终止状态，所以，在图 3-9(f)中，状态$[q_0,q_1,q_3]$和$[q_0,q_2,q_3]$是等价的 DFA 的终止状态。用双圈将它们标出，从而得到图 3-11。

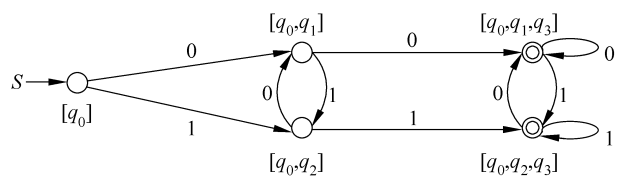

图 3-11　图 3-9 所示 NFA 的等价 DFA

3.4　带空移动的有穷状态自动机

1. 知识点

(1) 允许 ε-NFA 在某一状态下不读入字符——不移动读头，而只改变状态。

(2) 在 ε-NFA 中，$\hat{\delta}(q,\varepsilon)\neq\delta(q,\varepsilon)$，$\hat{\delta}(q,a)\neq\delta(q,a)$。所以，需要严格区分 $\hat{\delta}$ 与 δ。

(3) 在 ε-NFA 中，$\hat{\delta}$ 与 δ 需要分别进行扩展。

(4) NFA 是 ε-NFA 的特例。

(5) 由于 DFA，NFA，ε-NFA 是等价的，所以，统称它们为 FA。

2. 主要内容解读

定义 3-9　带空移动的不确定的有穷状态自动机(non-deterministic finite automaton with ε-moves,ε-NFA)M 是一个五元组:

$$M=(Q,\Sigma,\delta,q_0,F)$$

其中,

Q,Σ,q_0,F 的意义同 DFA。

δ——状态转移函数,又叫状态转换函数或者移动函数,$\delta:Q\times(\Sigma\cup\{\varepsilon\})\to 2^Q$。

对 $\forall(q,a)\in Q\times\Sigma$,$\delta(q,a)=\{p_1,p_2,\cdots,p_m\}$ 表示 M 在状态 q 读入字符 a,可以有选择地将状态变成 $p_1,p_2,\cdots$,或者 p_m,并将读头向右移动一个带方格而指向输入字符串的下一个字符。

对 $\forall q\in Q$,$\delta(q,\varepsilon)=\{p_1,p_2,\cdots,p_m\}$ 表示 M 在状态 q 不读入任何字符,可以有选择地将状态变成 $p_1,p_2,\cdots$,或者 p_m。也可以叫做 M 在状态 q 做一个空移动(或者 ε 移动),并且选择地将状态变成 $p_1,p_2,\cdots$,或者 p_m。

同样地,将 δ 扩充为 $\hat{\delta}:Q\times\Sigma^*\to 2^Q$。对任意的 $q\in Q,w\in\Sigma^*,a\in\Sigma$:

(1) ε-CLOSURE(q)=$\{p\mid$从 q 到 p 有一条标记为 ε 的路$\}$。

(2) $\varepsilon\text{-CLOSURE}(P)=\bigcup_{p\in P}\varepsilon\text{-CLOSURE}(p)$。

(3) $\hat{\delta}(q,\varepsilon)=\varepsilon\text{-CLOSURE}(q)$。

(4) $\hat{\delta}(q,wa)=\varepsilon\text{-CLOSURE}(P)$。

$$\begin{aligned}P&=\{p\mid\exists r\in\hat{\delta}(q,w),\text{使得 }p\in\delta(r,a)\}\\&=\bigcup_{r\in\hat{\delta}(q,w)}\delta(r,a)\end{aligned}$$

进一步扩展 $\delta:2^Q\times\Sigma\to 2^Q$。对任意 $(P,a)\in 2^Q\times\Sigma$:

(5) $\delta(P,a)=\bigcup_{q\in P}(q,a)$。

再进一步扩展 $\hat{\delta}:2^Q\times\Sigma^*\to 2^Q$。对任意 $(P,w)\in 2^Q\times\Sigma^*$:

(6) $\hat{\delta}(P,w)=\bigcup_{q\in P}\hat{\delta}(q,w)$。

定义 3-10　设 $M=(Q,\Sigma,\delta,q_0,F)$ 是一个 ε-NFA。对于 $\forall x\in\Sigma^*$,如果 $\hat{\delta}(q_0,x)\cap F\neq\varnothing$,则称 x 被 M 接受;如果 $\hat{\delta}(q_0,x)\cap F=\varnothing$,则称 M 不接受 x。

$$L(M)=\{x\mid x\in\Sigma^*\text{ 且 }\hat{\delta}(q_0,x)\cap F\neq\varnothing\}$$

称为由 M 接受(识别)的语言。

定理 3-2　ε-NFA 与 NFA 等价。

证明要点:

(1) 等价构造。

设 ε-NFA $M_1=(Q,\Sigma,\delta_1,q_0,F)$。

取 NFA $M_2=(Q,\Sigma,\delta_2,q_0,F_2)$。其中,

$$F_2=\begin{cases}F\cup\{q_0\} & \text{如果 } F\cap\varepsilon\text{-CLOSURE}(q_0)\neq\varnothing\\ F & \text{如果 } F\cap\varepsilon\text{-CLOSURE}(q_0)=\varnothing\end{cases}$$

对 $\forall(q,a)\in Q\times\Sigma$，使 $\delta_2(q,a)=\hat{\delta}_1(q,a)$。

关键点：

① 让 NFA 用非空移动去代替 ε-NFA 的含有一系列移动，其中含若干个空移动和相应的一个非空移动。

② 对 $F\cap\varepsilon\text{-CLOSURE}(q_0)\neq\varnothing$ 的情况，要进行特殊处理。

(2) 对 $\forall x\in\Sigma^+$，施归纳于 $|x|$，证明 $\delta_2(q_0,x)=\hat{\delta}_1(q_0,x)$。

(3) 证明对 $\forall x\in\Sigma^+$，$\delta_2(q_0,x)\cap F_2\neq\varnothing\Leftrightarrow\hat{\delta}_1(q_0,x)\cap F\neq\varnothing$。

(4) 证明 $\varepsilon\in L(M_1)\Leftrightarrow\varepsilon\in L(M_2)$。

3.5 FA 是正则语言的识别器

3.5.1 FA 与右线性文法

1. 知识点

(1) 右线性文法推导句子的过程，实际上可以与一个“适当的”FA 处理该句子的过程相对应。这个对应的关键是右线性文法的语法变量与 FA 状态的对应。

(2) FA 接受的语言是 RL。

(3) 在构造与所给的 DFA 等价的右线性文法时，不需要考虑与该 DFA 中的陷阱状态直接相关的产生式。

(4) RL 可以由 FA 接受。

(5) 当构造右线性文法对应的 FA 时，构造出来的一般是 NFA，并且需要增加一个终止状态。

(6) 定理 3-3 给出如何根据 DFA 构造右线性文法。从相应的证明可以知道，这个构造方法对 NFA 也是有效的。定理 3-4 给出了如何根据正则文法构造等价的 NFA。

2. 主要内容解读

定理 3-3 FA 接受的语言是正则语言。

证明要点：

由于曾经将正则语言定义为正则文法产生的语言，所以要证明此定理，只要证明对于任意给定的 DFA，存在等价的正则文法就可以了。

(1) 构造。

构造的基本思想是让正则文法的派生对应 DFA 的移动。

设 DFA $M=(Q,\Sigma,\delta,q_0,F)$。

取右线性文法 $G=(Q,\Sigma,P,q_0)$，其中，$P=\{q\to ap\mid\delta(q,a)=p\}\cup\{q\to a\mid\delta(q,a)=p\in F\}$。满足 $L(G)=L(M)-\{\varepsilon\}$。

这里的关键问题是对 $\{q\to a\mid\delta(q,a)=p\in F\}$ 所表示的产生式的理解，并且在实际的构

造过程中不要被忽略掉。

(2) 证明 $L(G)=L(M)-\{\varepsilon\}$。

对于 $a_1a_2\cdots a_{n-1}a_n\in\Sigma^+$,有如下等价关系式:

$q_0\overset{+}{\Rightarrow}a_1a_2\cdots a_{n-1}a_n\Leftrightarrow q_0\to a_1q_1,q_1\to a_2q_2,\cdots,q_{n-2}\to a_{n-1}q_{n-1},q_{n-1}\to a_n\in P$

$\Leftrightarrow\delta(q_0,a_1)=q_1,\delta(q_1,a_2)=q_2,\cdots,\delta(q_{n-2},a_{n-1})=q_{n-1},\delta(q_{n-1},a_n)=q_n$,且 $q_n\in F$

$\Leftrightarrow\delta(q_0,a_1a_2\cdots a_{n-1}a_n)=q_n\in F$

$\Leftrightarrow a_1a_2\cdots a_{n-1}a_n\in L(M)$

(3) 根据定理 2-5 和定理 2-6 考虑 ε 句子的问题。

例 3-9 不仅给出了定理 3-3 中所给出的构造方法的具体应用,而且这个例子还表明,在构造所给的 DFA 的等价 RG 时,不需要考虑与该 DFA 中的陷阱状态直接相关的产生式——因为陷阱状态对应的语法变量是"无用符号"。所以,应该在将 DFA 转换成 RG 之前,先将所有陷阱状态删除。另外,不可达状态对应的变量将不可能出现在相应文法的任何句型中,所以它们对应的产生式也是无用的,因此,也应该事先删除所有的不可达状态。有关"无用符号"将在 6.2 节中具体讨论。

定理 3-4 正则语言可以由 FA 接受。

证明要点:

(1) 构造。

基本思想是让 FA 模拟 RG 的派生。

设右线性文法 $G=(V,T,P,S)$,且 $\varepsilon\notin L(G)$。

取 FA $M=(V\cup\{Z\},T,\delta,S,\{Z\})$,$Z\notin V$。对 $\forall(a,A)\in T\times V$,

$$\delta(A,a)=\begin{cases}\{B\mid A\to aB\in P\}\cup\{Z\} & \text{如果 } A\to a\in P\\ \{B\mid A\to aB\in P\} & \text{如果 } A\to a\notin P\end{cases}$$

在此用 $B\in\delta(a,A)$与产生式 $A\to aB$ 对应,用 $Z\in\delta(a,A)$与产生式 $A\to a$ 对应。

(2) 证明 $L(M)=L(G)$。

首先,对于 $a_1a_2\cdots a_{n-1}a_n\in T^+$:

$a_1a_2\cdots a_{n-1}a_n\in L(G)\Leftrightarrow S\overset{+}{\Rightarrow}a_1a_2\cdots a_{n-1}a_n$

$\Leftrightarrow S\Rightarrow a_1A_1\Rightarrow a_1a_2A_2\Rightarrow\cdots\Rightarrow a_1a_2\cdots a_{n-1}A_{n-1}\Rightarrow a_1a_2\cdots a_{n-1}a_n$

$\Leftrightarrow S\to a_1A_1,A_1\to a_2A_2,\cdots,A_{n-2}\to a_{n-1}A_{n-1},A_{n-1}\to a_n\in P$

$\Leftrightarrow A_1\in\delta(S,a_1),A_2\in\delta(A_1,a_2),\cdots,A_{n-1}\in\delta(A_{n-2},a_{n-1}),Z\in\delta(A_{n-1},a_n)$

$\Leftrightarrow Z\in\delta(S,a_1a_2\cdots a_{n-1}a_n)$

$\Leftrightarrow a_1a_2\cdots a_{n-1}a_n\in L(M)$

其次,根据定理 2-6 补充说明 $\varepsilon\in L(G)$的情况。

推论 3-1 FA 与正则文法等价。

这个推论是作为对定理 3-3 和定理 3-4 的总结。

3.5.2 FA 与左线性文法

1. 知识点

(1) 将左线性文法对句子进行归约的过程与 FA 处理该句子的过程相对应。这个对应

总体上可以通过语法变量与状态的对应表现出来。

(2) 由于 FA 与左线性文法等价,所以左线性文法也是 RL 的一种描述,故将左线性文法和右线性文法统称为 RG 是合理的。

(3) 在构造与给定的 DFA 等价的左线性文法时,也不需要考虑该 DFA 中的陷阱状态和不可达状态直接相关的产生式。

(4) 根据 DFA 构造左线性文法。

(5) 根据 DFA 构造 RG 的方法也可以用于根据 NFA 构造 RG。

(6) 当构造左线性文法对应的 FA 时,一般构造出来的是 NFA,并且需要增加一个开始状态,文法的开始符号对应的状态为终止状态。

(7) 根据左线性文法构造 FA 的方法。

2. 主要内容解读

定理 3-5 左线性文法与 FA 等价。

证明要点:

(1) 证明文法中的归约与 FA 中的移动可以相互模拟。

(2) 根据 DFA 构造左线性文法的方法。

方法 1:

① 删除 DFA 的陷阱状态和不可达状态(包括与之相关的弧)。

② 在图中加一个识别状态 Z。

③ "复制"一条原来到达终止状态的弧,使它从原来的起点出发,到达新添加的识别状态 Z。

④ 如果 $\delta(A,a)=B$,则有产生式 $B\rightarrow Aa$。

⑤ 如果 $\delta(A,a)=B$,且 A 是开始状态,则有产生式 $B\rightarrow a$。

⑥ Z 为文法的开始符号。如果开始状态还是终止状态,则有产生式 $Z\rightarrow\varepsilon$。

方法 2:

① 删除 DFA 的陷阱状态和不可达状态(包括与之相关的弧)。

② 如果 $\delta(A,a)=B$,则 $B\rightarrow Aa$。

③ 如果 $\delta(A,a)=B$,且 A 是开始状态,则有产生式 $B\rightarrow a$。

④ 如果 $\delta(A,a)=B$,且 B 是终止状态,则有产生式 $Z\rightarrow Aa$。

⑤ 如果 $\delta(A,a)=A$,且 A 既是开始状态又是终止状态,则有产生式 $Z\rightarrow a$。

⑥ Z 为文法的开始符号。如果开始状态还是终止状态,则有产生式 $Z\rightarrow\varepsilon$。

(3) 根据左线性文法构造 FA 的方法。

① 增加状态 Z 为开始状态。

② 如果 $A\rightarrow Ba\in P$,则 $\delta(B,a)=A$。

③ 如果 $A\rightarrow a\in P$,则 $\delta(Z,a)=A$,$a\in\Sigma\cup\{\varepsilon\}$。

④ 文法的开始符号对应的状态为 FA 的终止状态。

有了本节的结论以及给出的等价转化方法,对于一个给定的 RL,可以用自己感觉比较方便的方法构造一个描述,然后再将其转化成要求的形式。一般来讲,由于 FA 的状态转移图比较直观,所以,画出接受给定 RL 的 FA 的状态转移图是比较方便的,当画出状态转移

图之后,再将其转换成RG。

3.6 FA的一些变形

本节内容可以只作为了解内容来阅读。

3.6.1 双向有穷状态自动机

1. 知识点

(1) 2DFA允许读头前进、后退、不动。

(2) 一个输入串被2DFA M 接受的充要条件是 M 从启动状态出发,最终到达某个终止状态,并且读头处于输入串的最后一个字符的右侧。

(3) 2DFA是正则语言的识别器。

(4) 2NFA在一个状态下读到一个符号,可以像NFA那样,选择进入某一个状态,并使读头前进、后退、不动。

(5) 2NFA与DFA等价。

2. 主要内容解读

定义3-11 确定的双向有穷状态自动机(two-way deterministic finite automaton, 2DFA)M 是一个五元组:

$$M=(Q,\Sigma,\delta,q_0,F)$$

其中,

Q,Σ,q_0,F 的意义同DFA。

δ——状态转移函数,又叫做状态转换函数或者移动函数,$\delta:Q\times\Sigma\to Q\times\{L,R,S\}$。对 $\forall(q,a)\in Q\times\Sigma$,

① 如果 $\delta(q,a)=\{p,L\}$,表示 M 在状态 q 读入字符 a,将状态变成 p,并将读头向左移动一个带方格而指向输入字符串中的前一个字符;

② 如果 $\delta(q,a)=\{p,R\}$,表示 M 在状态 q 读入字符 a,将状态变成 p,并将读头向右移动一个带方格而指向输入字符串的下一个字符;

③ 如果 $\delta(q,a)=\{p,S\}$,表示 M 在状态 q 读入字符 a,将状态变成 p,读头保持在原位不动。

关于FA的即时描述的定义对2DFA仍然有效。

定义3-12 设 $M=(Q,\Sigma,\delta,q_0,F)$ 为一个2DFA,M 接受的语言

$$L(M)=\{x\mid q_0x\overset{*}{\vdash}xp \text{ 且 } p\in F\}$$

定理3-6 2DFA与FA等价。

证明要点:

使用穿越序列。具体参考John E. Hopcroft, Jeffrey D. Ullman撰写的《Introduction to Automata Theory, Languages, and Computation》,该书由Addison-Wesley Publishing Company于1979年出版。或者参阅John E. Hopcroft, Jeffrey D. Ullman合著,莫绍揆等

翻译的《形式语言及其与自动机的关系》。

定义 3-13　不确定的双向有穷状态自动机（two-way nondeterministic finite automaton，2NFA）M 是一个五元组：

$$M=(Q,\Sigma,\delta,q_0,F)$$

其中，

Q,Σ,q_0,F 的意义同 DFA。

δ——状态转移函数，又叫做状态转换函数或者移动函数，$\delta:Q\times\Sigma\to 2^{Q\times\{L,R,S\}}$。对 $\forall(q,a)\in Q\times\Sigma,\delta(q,a)=\{(p_1,D_1),(p_2,D_2),\cdots,(p_m,D_m)\}$ 表示 M 在状态 q 读入字符 a，可以有选择地将状态变成 p_1，同时按 D_1 实现读头的移动，或者将状态变成 p_2 同时按 D_2 实现读头的移动；……；或者将状态变成 p_m，同时按 D_m 实现读头的移动。其中，$D_1,D_2,\cdots,D_m\in\{L,R,S\}$，表示的意义与定义 3-11 中表示的意义相同。

定理 3-7　2NFA 与 FA 等价。

3.6.2　带输出的 FA

1. 知识点

（1）Moore 机和 Mealy 机为带输出的 FA，与基本 FA 只输出是否接受输入串不同，这两个机器将根据输入串输出更多的内容。

（2）Moore 机对应处理输入串的过程中经过的每个状态，都有一个字符输出。

（3）Mealy 机对应处理输入串的过程中的每一个移动，都有一个字符输出。

（4）在约定的意义上，Moore 机与 Mealy 机等价。

2. 主要内容解读

定义 3-14　Moore 机是一个六元组：

$$M=(Q,\Sigma,\Delta,\delta,\lambda,q_0)$$

其中，

Q,Σ,q_0,δ 的意义同 DFA。

Δ——**输出字母表**（output alphabet）。

λ——$\lambda:Q\to\Delta$ 为**输出函数**。对 $\forall q\in Q,\lambda(q)=a$ 表示 M 在状态 q 时输出 a。

显然，对于 $\forall a_1a_2\cdots a_{n-1}a_n\in\Sigma^*$，$M$ 的输出串为：

$$\lambda(q_0)\lambda(\delta(q_0,a_1))\lambda(\delta(\delta(q_0,a_1),a_2))\cdots\lambda(\delta((\cdots\delta(\delta(q_0,a_1)a_2)\cdots),a_n))$$

设

$$\delta(q_0,a_1)=q_1,\delta(q_1,a_2)=q_2,\cdots,\delta(q_{n-2},a_{n-1})=q_{n-1},\delta(q_{n-1},a_n)=q_n$$

则 M 的输出可以简单表示为

$$\lambda(q_0)\lambda(q_1)\lambda(q_2)\cdots\lambda(q_n)$$

这是一个长度为 $n+1$ 的串。

定义 3-15　Mealy 机是一个六元组：

$$M=(Q,\Sigma,\Delta,\delta,\lambda,q_0)$$

其中，

Q,Σ,q_0,δ 的意义同 DFA。

Δ——输出字母表。

λ——$\lambda:Q\times\Sigma\to\Delta$ 为输出函数。对 $\forall(q,a)\in Q\times\Sigma,\lambda(q,a)=d$ 表示 M 在状态 q 读入字符 a 时输出 d。

对于 $\forall a_1a_2\cdots a_{n-1}a_n\in\Sigma^*$，$M$ 的输出串为

$$\lambda(q_0,a_1)\lambda(\delta(q_0,a_1),a_2)\cdots\lambda(\delta(\cdots\delta(\delta(q_0,a_1),a_2)\cdots),a_n)$$

设

$$\delta(q_0,a_1)=q_1,\delta(q_1,a_2)=q_2,\cdots,\delta(q_{n-2},a_{n-1})=q_{n-1},\delta(q_{n-1},a_n)=q_n$$

则 M 的输出可表示成长度为 n 的串：

$$\lambda(q_0,a_1)\lambda(q_1,a_2)\cdots\lambda(q_{n-1},a_n)$$

定义 3-16 设 Moore 机

$$M_1=(Q_1,\Sigma,\Delta,\delta_1,\lambda_1,q_{01})$$

Mealy 机

$$M_2=(Q_2,\Sigma,\Delta,\delta_2,\lambda_2,q_{02})$$

对于 $\forall x\in\Sigma^*$，当下式成立时，称它们是等价的。

$$T_1(x)=\lambda_1(q_0)T_2(x)$$

其中，$T_1(x)$和 $T_2(x)$分别表示 M_1 和 M_2 关于 x 的输出。

定理 3-8 Moore 机与 Mealy 机等价。

证明要点：

Moore 机处理输入 x 时每经过一个状态，就输出一个字符，使得输出字符和状态一一对应。所以，它的输出的长度为 $|x|+1$。

Mealy 机处理输入 x 时的每一个移动输出一个字符，使得输出字符和移动一一对应。所以，它的输出长度为 $|x|$。

无论是根据 Moore 机构造等价的 Mealy 机，还是根据 Mealy 机构造 Moore 机，可以让它们具有相同的状态和相同的移动函数：$Q_1=Q_2,\delta_1=\delta_2$。对输入串 x，无论是 Moore 机还是 Mealy 机，都将经过 $|x|+1$ 个状态，这些状态分别构成两个相同的状态序列，除了它们各自的最后一个状态对应相等之外，其他的状态都对应于相同的转移，所以，忽略 Moore 机在启动时对应于启动状态的输出，令 Moore 机对应的状态的输出等于 Mealy 机做相应的到达此状态的移动对应的输出就可以了。

3.7 小　　结

本章主要讨论正则语言的识别器——FA，包括 DFA，NFA，ε-NFA。讨论 RG 与 FA 的等价性，并且简单介绍带输出的 FA 和双向 FA。

(1) FA M 是一个五元组，$M=(Q,\Sigma,\delta,q_0,F)$，它可以用状态转移图表示。

(2) M 接受的语言为 $\{x\mid x\in\Sigma^*$ 且 $\delta(q,x)\in F\}$。如果 $L(M_1)=L(M_2)$，则称 M_1 与 M_2 等价。

(3) FA 的状态具有有穷存储功能。这一特性可以用来构造接受一个给定语言的 FA。

(4) NFA 允许 M 在一个状态下读入一个字符时，有选择地进入某一个状态，对于 $\forall x \in \Sigma^*$，如果 $\delta(q_0,x) \cap F \neq \varnothing$，则称 x 被 M 接受；如果 $\delta(q_0,x) \cap F = \varnothing$，则称 M 不接受 x。M 接受的语言 $L(M)=\{x \mid x \in \Sigma^* \text{ 且 } \delta(q_0,x) \cap F \neq \varnothing\}$。

(5) ε-NFA 是在 NFA 的基础上，允许直接根据当前状态变换到新的状态而不考虑输入带上的符号。对 $\forall q \in Q, \delta(q,\varepsilon)=\{p_1,p_2,\cdots,p_m\}$ 表示 M 在状态 q 不读入任何字符，可以有选择地将状态变成 $p_1,p_2,\cdots$，或者 p_m。这称为 M 在状态 q 做一个空移动。

(6) NFA 与 DFA 等价，ε-NFA 与 NFA 等价，统称它们为 FA。

(7) 根据需要，可以在 FA 中设置一种特殊的状态——陷阱状态。但是，不可达状态却应该无条件地删除。

(8) FA 是正则语言的识别模型。分别按照对推导和归约的模拟，可以证明 FA 和右线性文法、左线性文法等价。

(9) 2DFA 是 FA 的又一种变形，它不仅允许读头向前移动，还允许读头向后移动。通过这种扩充，2DFA 仍然与 FA 等价。2DFA 进一步扩展所得 2NFA 也与 FA 等价。

(10) Moore 机和 Mealy 机是两种等价的带输出的 FA。Moore 机根据状态决定输出字符，Mealy 机根据移动决定输出字符。

3.8 典型习题解析

2. 构造识别下列语言的 DFA(给出相应 DFA 的形式描述或者画出它们的状态转移图)。

(2) $\{0,1\}^+$。

解：本题的关键是保证接受的串的长度至少为 1，相应的 DFA 如图 3-18(a)所示。

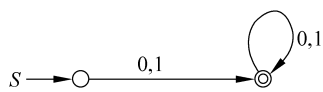

图 3-18(a)　题 2(2)的 DFA

(3) $\{x \mid x \in \{0,1\}^+$ 且 x 中不含形如 00 的子串$\}$。

解：构造要点是，自动机启动并读入一个字符后，就将“精力”集中在考察是否出现 00 子串上，一旦发现子串 00，就进入陷阱状态。构造结果如图 3-18(b)所示。

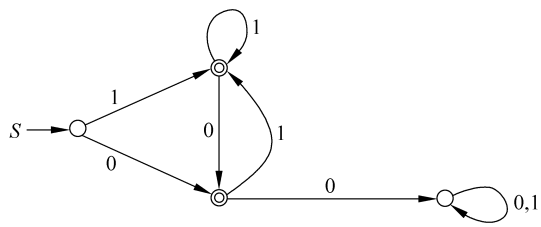

图 3-18(b)　题 2(3)的 DFA

(7) $\{x \mid x \in \{0,1\}^+$ 且当把 x 看成二进制数时，x 模 5 与 3 同余，要求当 x 为 0 时，$|x|=1$，且当 $x \neq 0$ 时，x 的首字符为 1$\}$。

解:构造要点如下。

① 以0开头的串都不能被此DFA接受,包括字符串0。所以,如果DFA在启动状态读入的符号为0,则直接进入陷阱状态。

② 该DFA共有7个状态:开始状态、陷阱状态、终止状态各一个,在相应的状态转移图中,终止状态和其余4种状态构成最大的强连通子图。

③ 其余部分请参考主教材例3-5。

(9) $\{x|x\in\{0,1\}^+$且x以0开头以1结尾$\}$。

解:构造要点如下。

① 启动时,只要考虑开始符号是否为0。

② 以1开头的字符串都是不可接受的。

③ 在读入一个字符0之后,当读到1时,可将这个1先当成结尾的1,如果是,就停止并接受;如果不是,就继续向前扫描。

④ 被接受串的长度至少为2。

构造结果如图3-18(c)所示。

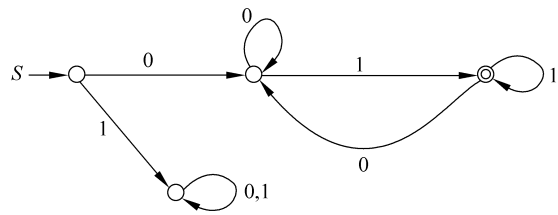

图3-18(c)　题2(9)的DFA

(11) $\{x|x\in\{0,1\}^*$且如果x以1结尾,则它的长度为偶数;如果x以0结尾,则它的长度为奇数$\}$。

解:构造要点如下。

① 语言根据句子长度的奇偶性对句子的结尾符号提出要求,因此,它将$\{0,1\}^+$中的字符串分成4个等价类,这4个等价类依次为:

[奇0]——$\{x|x\in\{0,1\}^+$不仅长度为奇数,而且以0结尾$\}$。

[奇1]——$\{x|x\in\{0,1\}^+$不仅长度为奇数,而且以1结尾$\}$。

[偶0]——$\{x|x\in\{0,1\}^+$不仅长度为偶数,而且以0结尾$\}$。

[偶1]——$\{x|x\in\{0,1\}^+$不仅长度为偶数,而且以1结尾$\}$。

这里直接用[奇0]、[奇1]、[偶0]、[偶1]分别标示这4个等价类对应的状态,这样理解起来就比较方便。显然,[奇0]、[偶1]对应的状态为终止状态。

② ε不属于上述任何一个等价类,所以,它自己独立地构成一个等价类,而且它是语言的句子,该等价类对应的状态为终止状态。可以用[ε]表示此等价类及其对应的状态。

③ $M=(\{[\varepsilon],[奇0],[奇1],[偶0],[偶1]\},\{0,1\},\delta,[\varepsilon],\{[\varepsilon],[奇0],[偶1]\})$,其中$\delta$由下面状态转移表表示。

	[ε]	[奇0]	[奇1]	[偶0]	[偶1]
0	[奇0]	[偶0]	[偶0]	[奇0]	[奇0]
1	[奇1]	[偶1]	[偶1]	[奇1]	[奇1]