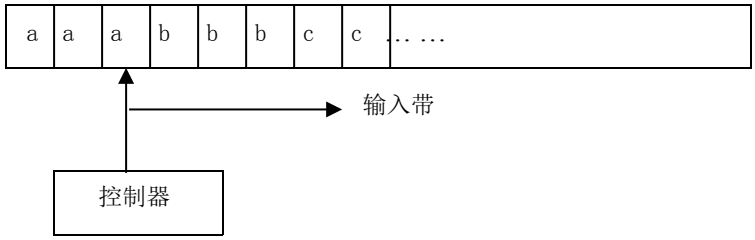
实验 2

名称	不确定有穷自动机的确定化

一、背景资料

有穷自动机（FA）可以看作是由一个带有读头的有穷控制器和一条字符输入带组成，如图所示。



FA 的示意图

控制器的读头从左至右顺次扫描输入带，每当从输入带上读到一个符号时，便引起控制器状态的改变，同时读头右移一个符号位。

控制器包括有穷个状态，状态和状态之间存在转换关系。当处于某个状态，读入一个字符时，则使状态改变为另一个状态，从而形成状态转换，改变后的状态称为后继状态。状态转换后的后继状态有三种可能情况：（1）后继状态为自身；（2）后继状态为一个；（3）后继状态为若干个。

某个有穷自动机，如果每次状态转换的后继状态都是惟一的，则称它是确定有穷自动机（DFA）；如果转换后的后继状态并不都是惟一的，则称它是不确定有穷自动机（NFA）。

有穷自动机的开始工作状态称为初始状态，结束工作的状态称为终止工作状态或接收状态。如果把上一小节中的状态转换图的各个结点看成是某一个状态，初始结点为初始状态，终止结点为终止状态，并且每一条边表示一个转换关系，这样一个有穷自动机的工作状态就可以采用状态转换图来描述了，从而可以把前面的图看成是一个有穷自动机。

对于上图，有穷自动机处在初始状态 0，当读入符号 a 后，自动机便从状态 0 转换到后继状态 1 中，再读入一个符号 b 后，自动机便从状态 1 转换到后继状态 2。当自动机读入一个符号串，自动机则从初始状态开始，经过一系列状态转换，最终若能够到达终止状态，则称这一符号串被该自动机所接收或识别，否则不能被该自动机所接收。

二、实验目的要求

输入： 非确定有穷（穷）状态自动机。

输出： 确定化的有穷（穷）状态自动机

三、实验原理

一个确定的有穷自动机（DFA）M 可以定义为一个五元组，M＝（K，∑，F，S，Z），其中：

（1）K是一个有穷非空集，集合中的每个元素称为一个状态；

（2）Σ是一个有穷字母表，Σ中的每个元素称为一个输入符号；

（3）F是一个从K×Σ→K的单值转换函数，即F（R，a）＝Q，（R，Q∈K）表示当前状态为R，如果输入字符a，则转到状态Q，状态Q称为状态R的后继状态；

（4）S∈K，是惟一的初态；

（5）Z⊆K，是一个终态集。

由定义可见，确定有穷自动机只有惟一的一个初态，但可以有多个终态，每个状态对字母表中的任一输入符号，最多只有一个后继状态。

对于DFA M，若存在一条从某个初态结点到某一个终态结点的通路，则称这条通路上的所有弧的标记符连接形成的字符串可为DFA M所接受。若M的初态结点同时又是终态结点，则称 ε 可为M所接受（或识别），DFA M所能接受的全部字符串（字）组成的集合记作L（M）。

一个不确定有穷自动机（NFA）M可以定义为一个五元组，M＝（K，Σ，F，S，Z），其中：

（1）k是一个有穷非空集，集合中的每个元素称为一个状态；

（2）Σ是一个有穷字母表，Σ中的每个元素称为一个输入符号；

（3）F是一个从K×Σ→K的子集的转换函数；

（4）S⊆K，是一个非空的初态集；

（5）Z⊆K，是一个终态集。

由定义可见，不确定有穷自动机NFA与确定有穷自动机DFA的主要区别是：

（1）NFA的初始状态S为一个状态集，即允许有多个初始状态；

（2）NFA中允许状态在某输出边上有相同的符号，即对同一个输入符号可以有多个后继状态。即DFA中的F是单值函数，而NFA中的F是多值函数。

因此，可以将确定有穷自动机DFA看作是不确定有穷自动机NFA的特例。和DFA一样，NFA也可以用矩阵和状态转换图来表示。

对于NFA M，若存在一条从某个初态结点到某一个终态结点的通路，则称这条通路上的所有弧的标记（ε 除外）连接形成的字符串可为M所接受。NFA M所能接受的全部字符串（字）组成的集合记作L（M）。

由于DFA是NFA的特例，所以能被DFA所接受的符号串必能被NFA所接受。

设M_1和M_2是同一个字母集Σ上的有穷自动机，若L（M_1）＝L（M_2），则称有穷自动机M_1和M_2等价。

由以上定义可知，若两个自动机能够接受相同的语言，则称这两个自动机等价。DFA是NFA的特例，因此对于每一个NFA M_1总存在一个DFA M_2，使得L（M_1）＝L（M_2）。即一个不确定有穷自动机能接受的语言总可以找到一个等价的确定有穷自动机来接受该语言。

NFA确定化为DFA

同一个字符串 α 可以由多条通路产生，而在实际应用中，作为描述控制过程的自动机，通常都是确定有穷自动机DFA，因此这就需要将不确定有穷自动机转换成等价的确定有穷自动机，这个过程称为不确定有穷自动机的确定化，即NFA确定化为DFA。

下面介绍一种NFA的确定化算法，这种算法称为子集法：

（1）若NFA的全部初态为S_1，S_2，…，S_n，则令DFA的初态为：

$S=[S_1, S_2, \cdots, S_n]$，

其中方括号用来表示若干个状态构成的某一状态。

（2）设DFA的状态集K中有一状态为$[S_i, S_{i+1}, \cdots, S_j]$，若对某符号$a\in\Sigma$，在NFA中有$F(\{S_i, S_{i+1}, \cdots, S_j\}, a)=\{S_i', S_{i+1}', \cdots, S_k'\}$

则令$F(\{S_i, S_{i+1}, \cdots, S_j\}, a)=\{S_i', S_{i+1}', \cdots, S_k'\}$为DFA的一个转换函数。若$[S_i', S_{i+1}', \cdots, S_k']$不在K中，则将其作为新的状态加入到K中。

（3）重复第2步，直到K中不再有新的状态加入为止。

（4）上面得到的所有状态构成DFA的状态集K，转换函数构成DFA的F，DFA的字母表仍然是NFA的字母表Σ。

（5）DFA中凡是含有NFA终态的状态都是DFA的终态。

对于上述NFA确定化算法——子集法，还可以采用另一种操作性更强的描述方式，下面我们给出其详细描述。首先给出两个相关定义。

假设I是NFA M状态集K的一个子集（即$I\in K$），则定义ε-closure（I）为：

（1）若$Q\in I$，则$Q\in$ ε-closure（I）；

（2）若$Q\in I$，则从Q出发经过任意条 ε 弧而能到达的任何状态Q’，则$Q'\in$ ε-closure（I）。

状态集 ε-closure（I）称为状态I的 ε 闭包。

假设NFA M＝（K，Σ，F，S，Z），若$I\in K$，$a\in\Sigma$，则定义$I_a=$ε-closure（J），其中J是所有从 ε-closure（I）出发，经过一条a弧而到达的状态集。

NFA确定化的实质是以原有状态集上的子集作为DFA上的一个状态，将原状态间的转换为该子集间的转换，从而把不确定有穷自动机确定化。经过确定化后，状态数可能增加，而且可能出现一些等价状态，这时就需要简化。

四、注意事项

⑴ 实现计算闭包closure（I）的算法；

⑵ 实现转换函数move（q，a）的算法；

⑶ 输出界面如下：

NFA的 图形形式
DFA的图形形式