重庆三峡学院学报——JOURNAL OF CHONGQING THREE GORGES UNIVERSITY
2005 年第 3 期 第 21 卷——No. 3. 2005 Vol.21.

基于 KMP 算法的有限自动机的确定化

李 峰

（重庆三峡学院数学与计算机科学学院，重庆万州 404000）

摘 要：一般非确定有限自动机转化为确定的有限自动机，其时间复杂度是指数函数级。对于小规模的，以输入串为识别语言的非确定的有限自动机，可采用本文介绍的方法加以确定化，其效率有极大的提高。

关键词：有限自动机；确定化；KMP 算法

中图分类号：TP301.1 文献标识码：A 文章编号：1009-8135(2005)03-0034-03

有限自动机是具有离散输入和输出系统的一种数学模型，在计算机科学中，有许多有限状态系统的例子，可以用自动机加以解决。如开关线路设置、文稿编辑、图书资料查找、结构型数据翻译等等。

1.有限自动机理论

有限自动机和现实的计算机一样都有一个固定的、能力有限的“中心处理装置”，它接收输入，没有输出，只给出是否接收这个输入的符号的结论。

1．1 确定的有限自动机（DFA）

定义 1 一个确定的有限自动机 M 是一个五元式[1]

$$M=（S，\Sigma，f，s_0，Z）$$

其中：S 是一个有限集，它的每个元素称为一个状态；

Σ是一个有限字母表，它的每个元素称为一个输入符号；

f 是一个从 S×Σ至 S 的（单值）部分映射。f(s,a)=s’意味着：当现行状态为 s，输入字符为 a 时，将转移到下一个状态 s’。

s_0∈S，是唯一的一个初态；

Z□S，是一个终态集（可空）。

一个确定型的有限自动机由一个输入带（输入串放在上面）、一个有穷控制器（含有有穷个不同的内部状态）和一个可单向移动的读头组成。DFA 的运行是这样的：开始时，读头在输入带最左端（指向第一个符号），控制器处于一个指定的初始状态。每隔一定时间有限自动机从输入带上读一个符号，然后进入一个唯一的新状态（即所谓确定的）。在读入一个符号之后，读头在输入带上向右移动一个字符位置，即指向下一个要识别的符号。这个过程重复，最后，读头达到输入端的结尾，若此时控制器处于终结状态，则表示该输入串被接受，否则不被自动机识别。机器接受的语言是它接受的所有字串的集合。

1．2 非确定型有限自动机（NFA）

定义 2 一个非确定的有限自动机 M 是一个五元式

$$M=（S，\Sigma，f，s_0，Z）$$

其中：S 是一个有限集，它的每个元素称为一个状态；

Σ是一个有穷字母表，它的每个元素称为一个输入符号；

收稿日期：2005-02-20

作者简介：李峰（1966-），女，四川内江人，重庆三峡学院数学与计算机科学学院讲师。

f 是一个从 $S\times\Sigma$ 至 2^S 的映射。

$s_0\in S$，是唯一的一个初态；

Z□S，是一个终态集（可空）。

非确定的有限自动机的形式定义与 DFA 相似，不同之处在于转移函数 f 的值域不同。对于 DFA 来说，$f(s_i,a)$是一个确定的状态 s_j，而对 NFA 来说，$f(s_i,a)$可以是空（Φ），也可以是一个状态子集$\{s_{i1},s_{i2},\cdots\cdots\}\in 2^S$。对于 NFA，当前状态和输入符号只能起部分决定作用。也即，对于给定的当前状态和输入符号，允许几个可能的" 下一个状态 "，这就是所谓非确定性。在本质上，这体现了一种改变状态的能力，而这一能力，可以极大的简化这些自动机的描述。因此，设计 NFA 要比 DFA 方便得多。

1．3 DFA 与 NFA 的等价性

定理 1　设 L 是被一个非确定有限自动机接受的集合，那么存在一个确定的有限自动机接受这个集合。[2]

[证]　设 M=（S，Σ，f，s_0，Z）是一个 NFA，它接受 L。定义一个 DFA，M'=（S'，Σ，f'，s'_0，Z'）如下：M'的状态是 M 的状态集合的所有子集，即 $S'=2^S$。Z 是 S'中所有包含 M 的一个终结状态的那些状态组成的集合。S'的元素记作$[s_1,s_2,...,s_i]$，其中 $s_1,s_2,...,s_i$ 都在 S 中，可以看到，$[s_1,s_2,...,s_i]$是 DFA 的一个单独的状态，它对应于 NFA 的一个状态集合。注意，$s_0'=[s_0]$。

我们定义

$$F'([s_1,s_2,...,s_i],a)=[p_1,p_2,\ldots,p_j]$$

当且仅当

$$F(\{s_1,s_2,...,s_i\},a)=\{p_1,p_2,\ldots,p_j\}$$

这就是说，在应用 f'到 S'的元素$[s_1,s_2,...,s_i]$上时，f'的值是通过将 f 用到由$[s_1,s_2,...,s_i]$表示出的 S 中的每一个状态上去的方法计算出来的。应用 f 到每个 $s_1,s_2,...,s_i$ 上，再取并集，由此，我们得到某个新状态集合 $p_1,p_2,\ldots,p_j$。这个新状态集合在 S'中有一个代表，即$[p_1,p_2,\ldots,p_j]$。这个元素就是 $f'([s_1,s_2,...,s_i],a)$的值。

用关于输入字符串 x 长度的归纳法，容易证明

$$f'(s_0',x)=[s_1,s_2,...,s_i]$$

当且仅当

$$f(s_0,x)=\{s_1,s_2,...,s_i\}$$

① 对于|x|=0，结论是明显的，因为 $s_0'=[s_0]$，且 x 必是 ε 。

② 假定对于长度小于等于 m 的输入，题设是对的。设 xa 是一个长度为 m+1 的字符串，a 在Σ中，那么

$$f'(s_0',xa)=f'(f'(s_0',x),a)$$

根据归纳假设

$$f'(s_0',x)=[p_1,p_2,\ldots,p_j]$$

当且仅当

$$f(s_0,x)=\{p_1,p_2,\ldots,p_j\}$$

根据 f'的定义

$$f'([p_1,p_2,\ldots,p_j],a)=[r_1,r_2,\ldots,r_k]$$

当且仅当

$$f(\{p_1,p_2,\ldots,p_j\},a)=\{r_1,r_2,\ldots,r_k\}$$

因此

$$f'(s_0',xa)=[r_1,r_2,\ldots,r_k]$$

当且仅当

$$f(s_0,xa)=\{r_1,r_2,\ldots,r_k\}$$

这就证明了归纳假设。

因为 $f'(s_0',x)$在 Z'中，恰好当 $f(s_0,x)$包含 S 中的一个在 Z 中的状态，于是 L(M)=L(M')。

可见，对任一个 NFA M，都有一个与之等价的 DFA M'。将 NFA 转化为 DFA 称为 NFA 的确定化，下面给出确定化算法。

[算法]

S'= ε -closure(s_0)　　//给出 M'的初始状态 s_0，S'={s_0}

for　S'中含有尚未标记的状态

{　　标记 s_i；　　//这里 $s_i=\{s_{i1},\ldots,s_{im}\}$, $s_{ii}\in S$;

　　for　a∈Σ

　　{　T=f(s_i,a);

　　　s_j= ε -closure(T);

　　　if　not(s_j□ S')　　then

　　　{　S'=S'∪ s_j;

　　　　将转换关系 $f'(s_i,a)=s_j$ 加入到 M';}

　　}

}

for　$s_{ii}\in s_i$ 且 $s_{ii}\in Z$　　//得到 M'的终态集 Z'

{　Z'=Z'∪s_i;}

该算法的时间复杂度为 O（$2^{|S|}|S|^3|\Sigma||f||S|^2$）[1]，约为输入规模的指数函数（$2^{|S|}$）。

2.KMP 算法用于 NFA 的确定化

KMP 算法是字符串模式匹配算法的一种高效算法，其时间复杂度可达 O（m+n）[3]，其中 m、n 分别为子串和主串的长度。该算法是由 D.E.Kunth 与 V.R.Pratt 和 J.H.Morris 同时发现。其基本思想是：每当一趟匹配过程中出现字符比较不等时，不需回溯主串读指针，而是利用已经得到的" 部分匹配 "

的结果将模式向右“ 滑动 ”尽可能远的一段距离后，继续比较。为此，设计一个 next 数组，next[i]中记录模式串中第 i 个字符失配时向右“ 滑动 ”的距离。

```
Get _next( T, next[ ] )     //计算 next 数组的值
{   i=1;   next[i]=0;   j=0;
    while   (i<T 的长度)
      {
        if   (j==0)||T[i]==T[j])
              { ++i; ++j; next[i]=j; }
        else     j=next[j];
      }
}
```

用 KMP 算法实现确定化的步骤是：首先设计一个以输入串为其识别语言的 NFA，然后利用 KMP 算法的 next 函数将 NFA 转化为 DFA。时间复杂度为 O（m）。

```
typedef   struct
  {   int   ch1,ch2;
  } elem;
elem   *DFA ;       // DFA 的存储结构
char   NFA[]=”abababb” ;       // NFA 的结构
NFA_len=strlen(NFA);
DFA=(elem ) malloc(sizeof(elem)*NFA_len);
Get_next(NFA, trans);   //求 KMP 的 next[ ]值
if   (NFA[0]==’a’ )         //0 状态的确定化
  { DFA[0].ch1=1;   DFA[0].ch2=0;}
else
  {DFA[0].ch2=1;   DFA[0].ch1=0;}
// NFA 的第 i 状态的确定化
for   (i=1; i<NFA_len; i++)
 {
   if   (NFA[i].ch1=’a’ )
     {      DFA[i].ch1=i+1;
            DFA[i].ch2=trans[i+1];
     }
   else
     {      DFA[i].ch2=i+1;
            DFA[i].ch1=trans[i+1];
     }
```

如，以“ ababaab ”作为输入串，得到的 NFA 和 DFA 如图 2 所示。

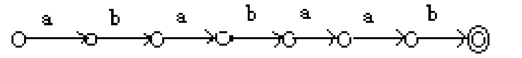

图 2（a） NFA

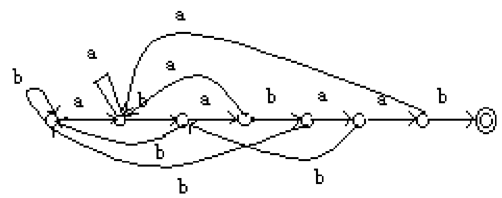

图 2（b） DFA

参考文献：

[1]Harry R.Lewis, Christos H.Papadimitriou.张立昂.刘田译.计算理论基础[M]，清华大学出版社，2000.7

[2]John E. Hopcroft, Jeffery D. Ullman，徐美瑞译，自动机理论、语言和计算导引[M].科学出版社，1986.9.

[3]严蔚敏.吴伟民.数据结构（C 语言版）[M].清华大学出版社，1999.2.

（责任编辑：郑宗荣）

Finite Automation Determined Based on KMP Algorithms

LI Feng

(Dept. of Computer Science, Chongqing Three Gorges University, Wanzhou 404000)

Abstract: Non-deterministic finite automaton is translated into deterministic finite automaton. The time complexity is an exponent function. On small scale, a NFA witch recognize input string, is determined by the method in the paper, then the efficiency is raised

Key Words: finite automaton; deterministic; KMP algorithms