文章编号：1000－5471(2008)05－0134－03

确定有限自动机的逻辑形式定义①

刘　益[1]，　闵　兰[2]

1. 宜宾学院 计算机与信息科学系，四川 宜宾 644007；2. 成都理工大学 信息管理学院，四川 成都 610059

摘要：通过分析确定有限自动机状态转换函数的内在含义，引入有关的原子命题，得到确定有限自动机的逻辑形式定义并证明了状态转换函数表示与逻辑表示之间的等价性.

关　键　词：确定有限自动机；状态转换函数；命题逻辑

中图分类号：TP301.1　　　　**文献标识码：A**

自动机用来描述字母集和状态集之间的关系，主要用于识别语言、电路设计、模型检测以及加密算法等方面[1-2]．为了说明这种关系，一般采用状态转换函数、状态矩阵以及状态转换图等形式表示．为了深入地研究自动机集合之间的关系，常从理论上借助于格值理论和逻辑理论[3-7]．本文使用逻辑理论分析确定有限自动机集合间的内部关系，给出了它的逻辑形式定义，并证明了确定有限自动机的状态转换函数定义形式与逻辑定义形式之间的等价性.

1　确定有限自动机的逻辑形式定义

一般地，一个确定有限自动机由一个五元组构成，$M=(Q,S,f,q_0,F)$.

其中：Q 是状态的有限集合，S 是有限输入字母表，$q_0\in Q$ 是初态，$F\subseteq Q$ 是终止状态集，f：$Q\times S\rightarrow Q$ 是状态转换函数.

由于自动机的本质是集合间的逻辑关系，因此可以采用命题函数表示确定有限自动机.

定义：一个确定有限自动机由一个五元组构成，$M=(Q,S,g,q_0,F)$

其中：Q 是状态的有限集合，S 是有限输入字母表，$q_0\in \mathbf{Q}$ 是初态，$F\subseteq Q$ 是终止状态集，g：$\exists x\exists y(T(x)\wedge R(y))\Rightarrow\exists(x\,W(x))$ 是命题函数.

设 $T(x)$：x 是当前状态，$R(y)$：y 是当前输入字母，$W(x)$：x 是后续状态.

x 的个体域是有限状态集合 Q 中的元素，y 的个体域是有限输入字母表 S 中的元素.

2　状态转换函数表示与逻辑定义表示的等价性

定理　对于每一台采用状态转换函数描述的确定有限自动机 M_S，存在一台采用命题函数描述的确定有限自动机 M_T 与之对应．反之也成立.

证　采用状态转换函数描述的 M_S，$M=(Q,S,f,q_0,F)$

$$f: Q\times S\rightarrow Q$$

状态转换函数 f 的描述是：当自动机 M_S 收到一个输入符号(称为当前输入符号)，根据当前状态，转换到下一个状态(称为后续状态).

设有限状态集合 Q 中的元素是$\{q_1,q_2,\cdots,q_n\}$，有限输入字母表 S 中的元素是$\{a_1,a_2,\cdots,a_m\}$.

---

①　收稿日期：2008－03－11

作者简介：刘　益(1971－)，男，四川宜宾人，讲师，硕士研究生，主要从事自动化控制研究.

引入谓词描述，$T(x)$：x 是当前状态；$R(y)$：y 是当前输入字母；$W(x)$：x 是后续状态. 其中 x 的个体域是有限状态集合 Q 中的元素，y 的个体域是有限输入字母表 S 中的元素. $T(q_1),T(q_2),T(q_3),\cdots,T(q_n)$ 分别表示 $q_1,q_2,q_3,\cdots,q_n$ 的当前状态；$R(a_1),R(a_2),R(a_3),\cdots,R(a_m)$ 分别表示 $a_1,a_2,a_3,\cdots,a_m$ 是当前输入字母；$W(q_1),W(q_2),W(q_3),\cdots,W(q_n)$ 分别表示 $q_1,q_2,q_3,\cdots,q_n$ 的后续状态.

(1) 从 M_S 构造对应的 M_T

设确定有限自动机中：a_i 是当前输入符号，$a_i \in S$；q_j 是当前状态，$q_j \in Q$；q_k 是后续状态，$q_k \in Q$. 对应的一个状态转换函数为：$f(q_j,a_i)=q_k$. 用谓词描述，$T(q_j)$：q_j 是当前状态；$R(a_i)$：a_i 是当前输入符号；$W(q_k)$：q_k 是后续状态.

在确定有限自动机中，后续状态在每次状态转换过程中是唯一的. 但这个状态可以出现在不同的转换过程中. 根据 $f(q_j,a_i)=q_k$ 描述，如果当前状态是 q_j 且输入符号为 a_i 时，自动机将转换到下一个状态 q_k. 借助命题逻辑，可用命题公式表示为：$(T(q_j) \wedge R(a_i)) \to W(q_k)$.

可以看出，对 f：$Q \times S \to Q$，如果要用命题公式描述，需要对状态转换过程分 3 种情况讨论：第一种情况是在所有的状态转换函数中，q_k 作为后续状态出现在多个状态转换函数中；第二种情况是在所有的状态转换函数中，q_k 作为后续状态只出现在一个状态转换函数中；第三种情况是 q_k 作为后续状态不出现在状态转换函数中.

下面就这 3 种情况分析状态转换函数的逻辑形式表示.

① q_k 作为后续状态出现在多个状态转换函数中

不设一般性，设 q_k 是后续状态，出现在 $f(q_1,a_1)=q_k$，$f(q_1,a_2)=q_k$，$f(q_2,a_2)=q_k$ 这 3 个状态转换函数中. 其中：$q_1,q_2,q_k \in Q$，q_1,q_2,q_k 是不同状态；$a_1,a_2 \in S$，a_1,a_2 是不同的输入符号.

对于 $f(q_1,a_1)=q_k$，有 $(T(q_1) \wedge R(a_1)) \to W(q_k)$；对于 $f(q_1,a_2)=q_k$，有 $(T(q_1) \wedge R(a_2)) \to W(q_k)$；对于 $f(q_2,a_2)=q_k$，有 $(T(q_2) \wedge R(a_2)) \to W(q_k)$.

则 $((T(q_1) \wedge R(a_1)) \vee (T(q_1) \wedge R(a_2)) \vee (T(q_2) \wedge R(a_2))) \to W(q_k)$.

由于上式是根据确定的状态转换函数所得，可以判断这个命题公式是重言式. 因此有：

$$((T(q_1) \wedge R(a_1)) \vee (T(q_1) \wedge R(a_2)) \vee (T(q_2) \wedge R(a_2))) \Rightarrow W(q_k)$$

② q_k 作为后续状态出现在一个状态转换函数中

设 q_k 是后续状态只出现在 $f(q_j,a_i)=q_k$ 中，这种情况可以看成是第一种情况的特例. 因此有：

$$(T(q_j) \wedge R(a_i)) \Rightarrow W(q_k)$$

③ q_k 作为后续状态不出现在状态转换函数中

在这种情况下状态转换函数 f：$Q \times S \to Q$ 中没有具体的状态转换函数式描述，也就没有对应的逻辑表示.

设集合 $M=\{T(q_1),T(q_2),T(q_3),\cdots,T(q_n)\}$；$N=\{R(a_1),R(a_2),R(a_3),\cdots,R(a_m)\}$；$L=\{W(q_1),W(q_2),W(q_3),\cdots,W(q_n)\}$. 且集合 M，N，L 中各元素的取值为逻辑值：真、假. 根据笛卡儿乘积有 $M \times N$：

$$M \times N = \{< T(x), R(y) > \mid (T(x) \in M) \wedge (R(y) \in \mathbf{N})\}$$

则有

$$(T(q_1) \wedge R(a_1)) \vee (T(q_1) \wedge R(a_2)) \vee \cdots \vee (T(q_2) \wedge R(a_1)) \vee \cdots \vee (T(q_n) \wedge R(a_m))$$
$$\Rightarrow W(q_1) \vee W(q_2) \vee W(q_3) \vee \cdots \vee W(q_n)$$

根据量词对变元的约束，可以将上式表示为：

$$\exists x \exists y (T(x) \wedge R(y)) \Rightarrow \exists x W(x)$$

因此，对于确定有限自动机的状态转换函数 f：$Q \times S \to Q$ 可以用命题函数 $\exists x \exists y (T(x) \wedge R(y) \Rightarrow \exists x W(x)$ 表示. 即对于 M_S 可以构造出对应的 M_T.

(2) 从 M_T 构造对应的 M_S

根据确定有限自动机的命题函数 $\exists x \exists y (T(x) \wedge R(y)) \Rightarrow \exists x W(x)$ 可知，若存在有限状态集合 Q 中的元素是当前状态，存在有限输入字母表 S 中的元素是当前输入字母，则存在一些有限集合 Q 中的元素是后续状态.

对于任意给定的一个确定有限自动机，$T(x),R(y),W(x)$ 集合中每个元素是确定的. 由逻辑命题

g：$\exists x \exists y(T(x) \wedge R(y) \Rightarrow \exists x W(x)$，可得到一组蕴含式．在这组蕴含式中有两种情况，一种是由多个 $T(x)$ 与 $R(y)$ 合取进行析取后蕴含 $W(x)$；另一种是 $T(x)$ 与 $R(y)$ 合取后蕴含 $W(x)$．

对第一种情况，不设一般性，设有 $((T(q_1) \wedge R(a_1)) \vee (T(q_1) \wedge R(a_2)) \vee (T(q_2) \wedge R(a_2))) \Rightarrow W(q_k)$ 成立．它表示如果 a_1 是当前输入符号，q_1 是当前状态，或者 a_2 是当前输入符号，q_1 是当前状态，或者 a_2 是当前输入符号，q_2 是当前状态，则 q_k 都是后续状态．可以用3个状态转换函数 $f(q_1, a_1) = q_k$，$f(q_1, a_2) = q_k$，$f(q_2, a_2) = q_k$ 描述．

对第二种情况，不设一般性，设有 $T(q_j) \wedge R(a_i) \Rightarrow W(q_k)$ 成立．它表示如果 a_i 是当前输入符号，q_j 是当前状态，则 q_k 是后续状态．用状态转换函数 $f(q_i, a_j) = q_k$ 描述．

从上面的分析可以看出，所有的蕴含式都可以用状态转换函数表示，从而得到用状态转换函数描述的确定有限自动机 M_S．因此，对于确定有限自动机的命题函数描述可以用状态转换函数表示．即对于 M_T 可以构造出对应的 M_S．

3　总　　结

本文依据逻辑理论，结合确定有限自动机输入字母和状态集合间的关系，给出了确定有限自动机的逻辑形式定义．无论是采用状态转换函数定义的确定有限自动机还是采用逻辑形式定义的确定有限自动机，其实质都是说明自动机的工作过程．但采用逻辑形式定义确定有限自动机，阐明了确定有限自动机集合间的内在关系，使通过逻辑运算演算确定有限自动机的状态转换成为可能，同时也使利用计算机推理它的工作过程成为可能，为自动机可计算奠定了理论基础．

参考文献：

[1]　沈　浩，孙永强．自动机，逻辑与博弈 [J]．计算机工程，2003，29(20)：9－11．

[2]　陶仁骥．一种有限自动机公开钥密体制和数字签名 [J]．计算机学报，1985，8(6)：401－409．

[3]　邱道文．基于完备剩余格值逻辑的自动机理论 [J]．中国科学(E辑)，2003，33(2)：138－146．

[4]　李永明．格值自动机与语言 [J]．陕西师范大学学报(自然科学版)，2003，31(4)：1－6．

[5]　雷红轩，潘　超．格值有限自动机及其性质 [J]．内江师范学院学报，2006，2l(4)：9－12．

[6]　李　平，李永明．几类格值自动机的关系 [J]．模糊系统与数学，2005，19(3)：96－100．

[7]　邱道文．基于量子逻辑的自动机和文法理论 [J]．软件学报，2003，14(1)：23－27．

Logic Definition of Deterministic Finite Automaton

LIU　Yi[1]，　MIN　Lan[2]

1. Ddpartment of Computer and Information Science，Yibin University，Yibin Sichuan 644007；

2. College of Information Management，Chengdu University of Technology，Chengdu Sichuan 610059

Abstract：Generally speaking，finite automaton is described by state transition function. The study of finite automaton is just the study of state transition process. In the paper，finite automaton is described by the relation of sets and its work process is analyzed by a logic reasoning process. Through atom proposition state transition function，the logic definition of DFA is established and the equivalence of logic description and state transition function description are proved.

Key words：deterministic finite automaton；state transition function；proposition logic

责任编辑　张　栒