通信协议 有限状态机模型 测试

计算机科学1996 Vol. 23 №. 6

(24)

94-96

基于非确定有限状态机模型的测试用例选择

TP393

郑学军 王春森
(复旦大学计算机科学系 上海200433)

摘 要 This paper presents a method of generating automatically test sequences for systems modeled as non-deterministic finite state machines. Generate state charatcterization set for each state in the state machines. Then, generate test sequences to test software. It can also be used for testing dynamic model in object-oriented method.

关键词 Non-deterministic finite state machine, Test sequence, State characterization set.

在异构型系统中相互作用的正确性是通过一致性测试来保证的，自从协议工程领域出现以来，通信协议的一致性测试一直是人们研究的一个课题，一致性测试的目的是保证协议的实现与协议的标准规格说明一致。象通信协议这样的软件系统可以模型化成有限状态机，另外，面向对象方法(如OMT)中的动态模型在一些受限条件下也可以模型化成有限状态机[1]。

测试软件系统的实现与其规格说明是否一致是保证软件功能正确的一项重要活动，目前，有一些方法可以从确定有限状态机模型中产生测试软件的测试序列[2][3][6]。本文提出的方法是从非确定有限状态机模型中产生测试序列，首先根据非确定有限状态机模型产生测试树和每个状态的状态特征集合，然后根据状态特征集合和测试树产生测试序列。

1 基本概念和原理

非确定有限状态机(NFA)M是一个五元式：

$M=(S,I,O,f,g)$。

其中：S是有限集合，它的每一个元素称为状态；I是有限的输入集合；O是有限的输出集合；f是状态转换函数，即从S×I到S的子集的映射；g是输出函数，即从S×I到O的子集的映射。

对非确定有限状态机M的测试就是从M生成测试序列，据此测试M的实现M′。从M产生的测试序列是输入/输出序列对$(i_1/o_1,i_2/o_2,\cdots,i_n/o_n)$，即对于M，如果输入序列$(i_1,i_2,\cdots,i_n)$，期望产生输出序列$(o_1,o_2,\cdots,o_n)$。如果在M的实现M′中输入同一输入序列$(i_1,i_2,\cdots,i_n)$，产生的输出序列不是$(o_1,o_2,\cdots,o_n)$，可能有两种情况造成：(1)M和M′的状态转换函数不同，即下一状态错误；(2)M和M′的输出函数不同，即输出错误。

为了验证输入为i，状态s_k到s_j的转换正确，必须满足：

(1)输入i之前，M′中的状态一定转换到能识别成s_k的状态；

(2)M′中相应于输入i的输出一定在M中已指定；

(3)输入i之后，M′中的状态一定到达能识别成s_j的状态。

因此，测试每次转换正确的关键在于通过状态识别和验证序列来完成起始状态和终止状态的转换，已提出了构造状态识别或验证序列的形式化方法，如可识别序列[5]，状态特征集合[3]，唯一的输入/输出序列及Wp方法和DD方法[4]等。但是这些方法都是基于确定有限状态机模型的，我们在以下几种假设基础上提出了一种基于非确定有限状态机模型的方法：

(1)M是最小的，即M中无两个状态是等价的(总是可以将非最小状态化简为最小状态)；

(2)M可由强连通有向图表示，即每对顶点总有一条路径(为了验证M中的每个状态转换，初始状态总可以到达M中的任意状态)；

(3)M′和M的输入集合相同，即M′的状态数不会超过M的状态数(错误可能改变输出和转换的目标，但不会增加M′的状态数)。

*)郑学军 硕士生，研究兴趣为面向对象和软件测试。王春森 副教授。

下面定义三个概念和给出一个定理：

定义1 S中的两个状态 s_k 和 s_i，对于同一输入i，有不同的输出集合，则称状态 s_k 和 s_i 为单可区别状态对。

定义2 S中的两个状态 s_k 和 s_i，对于同一输入i，如果有公共的输出，对于每个公共的输出o，状态 s_k 和 s_i 分别经相同输入i/输出o后的下一状态对也是多可区别状态对（至少是单可区别状态对），则称状态 s_k 和 s_i 为多可区别状态对。

定义3 状态 s_k 的状态特征集合 $SCS(s_k)=\{\mu_{i/o}(s_k,s_j)|\forall j\neq k,s_k,s_i\in S\}$。其中，$\mu_{i/o}(s_k,s_j)$表示从状态 s_k 到状态 s_j 输入/输出序列，可以用树来表示，即非叶节点表示输入，叶节点表示输入及其输出，边表示输出。$i\#o_1,o_2,\cdots,o_m\{T_1,T_2,\cdots,T_m\}$表示根节点为i，边为 $o_1,o_2,\cdots,o_m$，而 $T_1,T2,\cdots,T_m$ 是输出边相应的子树。

定理 给定非确定有限状态机M，状态特征集合$\{SCS(s_k)|s_k\in S\}$，对于两个状态 s_k 和 s_j，如果满足从 s_k 转换到 s_i 的输入序列与从 s_j 转换到 s_k 的输入序列相同，那么它能识别出M的实现M′的任何状态。

证明：因为对于SCS中的任意一个输入序列，M中的输出集合是唯一的，同样，对于任意状态的SCS的输入序列，M′中无两个状态产生相同的输出集合，除非M′比M的状态多（这与假设相矛盾），或者M′有错误，例如，M′中的状态 s_k 和 s_j 对于从 s_k 转换到 s_i 的输入序列有同样的输出，当识别 s_i 时就会发现错误。□

后面的算法就是先决定所有的单可区别状态对和多可区别状态对，同时生成状态转换的测试树，然后生成满足定理的每个状态的状态特征集合SCS，

2 生成状态特征集合(SCS)的算法

下面提出了一种生成满足定理的状态特征集合和每次状态转换的测试树算法，描述为：

输入：非确定有限状态机M

输出：测试树和每个状态的状态特征集合

步骤：/* 一次替代，决定所有的单可区别状态对 */

```
for k=1 to n-1/* n为状态机M的状态数*/
  for T(sk)中的每个状态sj/*T(sk)={sj,n≥j>k},0<k<n*/
    for I中的每个输入i /*I为输入集合*/
      if g(sk,i)∩g(sj,i)=Φ/*输入为i,状态sk,sj的输出集合不相交*/
      {
        在测试树中加入包含输入i及其输出的叶节点;
        μi/o(sk,sj)=i/g(sk,i);/*状态sk转换到sj的输入/输出序列*/
        μi/o(sj,sk)=i/g(sj,i);/*状态sj转换到sk的输入/输出序列*/
        T(sk)=T(sk)-sj;
        break;
      }
/*多次替代,决定所有的多可区别状态对*/
do
  for k=1 to n-1 /*n为状态机M的状态数*/
    for T(sk)中的每个状态sj/*T(sk)={sj,n≥j>k},0<k<n*/
      for I中的每个输入i /*I为输入集合*/
      {
        /*O为输入i,状态sk和sj的公共输出集合*/
        O={o1,o2,…os}=g(sk,i)∩g(sj,i);
        xm和ym分别是sk和sj经相同的输入i/输出Om的下一状态;
        /*Psk和Psj分别为状态sk和sj的非公共输出集合*/
        Psk和Psj分别为g(sk,i)-O和g(sj,i)-O;
        if ∀Om,1≤m≤z,xm≠ym并且xm,ym是多可区别状态对
        {
          在测试树中加入节点i,加入从节点i到子树μi/o(x1,y1),μi/o(x2,y2),…,μi/o(xz,yz),nil的边o1,o2,…oz,psk;/*产生状态sk转换到sj的输入/输出序列*/
          μi/o(sk,sj)=i#o1,o2,…oz,psk{μi/o(x1,y1),μi/o(x2,y2),…,μi/o(xz,yz),nil};
          在测试树中加入节点i,加入从节点i到子树μi/o(y1,x1),μi/o(y2,x2),…,μi/o(yz,xz),nil的边o1,o2,…oz,psj;
          /*产生状态sj转换到sk的输入/输出序列*/
          μi/o(sj,sk)=j#o1,o2,…oz,psj{μi/o(y1,x1),μi/o(y2,x2),…,μi/o(yz,xz),nil};
          T(sk)=T(sk)-sj;
          break;
        }
      }
while T为空或在替代中无新的可区别状态对产生
/*生成每个状态的状态特征集合*/
for k=1 to n
{
  SCS(sk)={μi/o(sk,sj),∀1≤j≤n,j≠k,sk,sj∈S};
{
```

替代的最大次数为n(n-1)/2，产生的测试树为n(n-1)个，算法的时间复杂性为$O(i_{max}o_{max}n^4)$，这里i_{max}，o_{max}分别表示最大的输入和输出数。

3. 生成测试序列的例子

假如有一个非确定有限状态机M如图1所示，用圆圈表示状态，有向边表示导致状态转换的输入/输出，$S=\{s_0,s_1,s_2,s_3\}$，$I=\{a,b\}$，$O=\{y,z\}$。根据前面的算法，经第一次替代，产生的测试树如图2.1a（$\mu_{i/o}(s_0,s_1)$）和2.1b（$\mu_{i/o}(s_1,s_0)$），经第二次，…，第六次替代，产生的测试树分别如图2.2a（$\mu_{i/o}(s_1,s_2)$）和2.2b（$\mu_{i/o}(s_2,s_1)$），…，2.6a（$\mu_{i/o}(s_2,s_3)$）和2.6b（$\mu_{i/o}(s_3,s_2)$）。同时，根据测试树产生测试状态s_0的状态特征集$SCS(s_0)$为{（$\mu_{i/o}(s_0,s_1)$），（$\mu_{i/o}(s_0,s_2)$），（$\mu_{i/o}(s_0,s_3)$）}，其测试序列为从状态s_0到s_1是{a/y}，从s_0到s_2是{baa/zyy，baaa/zzzy}，从s_0到s_3是{abaa/yzyy，abaaa/yzzzy，a/z}。

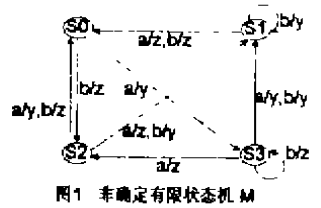
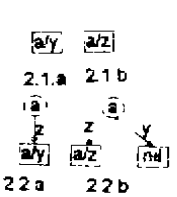
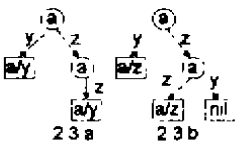
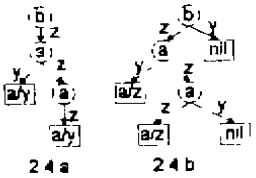


图1 非确定有限状态机 M







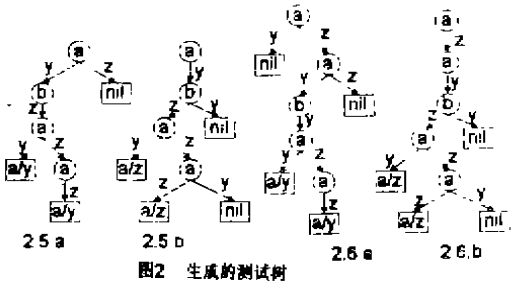


图2 生成的测试树

4. 在面向对象软件测试中的应用

在面向对象方法中，对象的动态模型可以表示成一系列的状态和状态间的转换，但大多数对象并不能构成形式化的确定状态机，文[1]在一些受限条件情况下，利用面向对象的方法可以把软件系统模型化成有限状态机，并且子类的状态机可以继承父类的状态机，从而简化了状态机的构造。利用本文的方法可以测试基于有限状态机模型的面向对象软件，最突出的优点在于子类可以继承父类的测试用例。

参考文献

[1]J. D. McGregor et al., A note on inheritance and state machines, ACM SIGSOFT Software Eng. notes, vol. 18, 1993

[2]T. S. Chow, Testing design modeled by finite-state machines, IEEE Trans. on Software Eng. Vol. 4, 1978

[3]S. Fujiwara et al., Test selection based on finite-state models, IEEE Trans. on Software Eng. Vol. 17, 1991

[4]H. Aboeifotoh, A test generation algorithm for systems modeled as nondeterministic FSMs, Software Eng. Journal, 1993

[5]G. Gonenc, A method for the design of fault detection experiments, IEEE Trans. on Comm., Vol. 19, 1970

[6]T. Ramalingam, etc., On testing and diagnosis of communication protocols based on the FSM model, Computer Communications, Vol, 18, 1995

（上接第97页）

第5期

第六期

（略）