基于 UIO 测试序列的错误诊断算法

赵保华 钱 兰 郭雄辉
(中国科学技术大学计算机科学技术系 合肥 230027)

摘 要 唯一输入输出(Unique Input Output)测试序列是协议测试中常用的一种测试序列,在一个已有的错误诊断算法基础上，结合 UIO 测试序列的一些特点，该文提出了一种应用于 UIO 测试序列的错误诊断算法。该算法充分利用了 UIO 测试序列给出的判定消息，及测试结果中可能的错误转换后的输入/输出消息，从而能高效完全地诊断单个错误。最后用实验数据给出了该文算法和原始算法之间的比较结果。

关键词 错误诊断，一致性测试，有限状态机，唯一输入输出

中图分类号：TN915.04 **文献标识码**：A **文章编号**：1009-5896(2006)11-2152-05

Fault Diagnosis for UIO Test Sequence

Zhao Bao-hua Qian Lan Guo Xiong-hui
(*Computer Science and Technology Department, USTC, Hefei* 230027, *China*)

Abstract UIO(Unique Input Output) test sequences are widely used in communication protocol testing. In this paper, based on an existing fault diagnosis algorithm, an improved algorithm, which utilizes as much information of UIO test sequences as possible, is proposed. This paper full uses of verdict information given by UIO test sequences and the observed input/output immediately after the potential faulty transition to guarantee the efficient diagnosis of any single fault. Some experiments to compare the algorithm with the original one are conducted and the results show that the algorithm is more efficient.

Key words Fault diagnosis, Conformance testing, Finite state machine, Unique Input Output(UIO)

1 引言

通信协议日趋复杂，如何保证各个厂商的协议实现与协议描述相一致，从而使得各种网络服务能正常运行变得越来越重要。有限状态机可以精确刻画出通信协议的控制流部分，因而在通信协议的一致性测试上基于有限状态机的测试问题得到了广泛的研究，文献[1]给出了一个很好的综述。基于有限状态机的测试序列的生成方法有很多种，常用的有T方法，D 方法，W 方法和UIO 方法[2]，其中UIO 方法由于其生成的测试序列较短且检错能力较强而得到广泛研究和应用。

基于有限状态机的通信协议的一致性测试问题已经得到了广泛的研究。在检测到错误后，如何给出错误诊断信息是一个很重要的问题。Ghedamsi最早研究了这个问题[3]，并在单个错误的前提下给出了错误诊断算法，该算法首先生成了可能的错误转换集，然后枚举出可能的输出或者转换错误，用测试序列执行后所观察到的输出信息来排除其中的一些错误，从而形成最终的错误诊断集。Lee给出了另外一个诊断算法，Lee和Ghedamsi的算法思想非常相似，相关工作参见文献[4]。Miller在文献[5]中利用逆向路径法给出了单个错误诊断集，该算法时间复杂度较低，Guo在文献[6]中给出了文献[5]的两个改进算法，能缩小错误诊断集合，但是存在一些这两个算法诊断不了的错误，因而有其应用限制。

在文献[4]中指出对多个错误进行诊断需要指数时间复杂度的算法才能解决，很少有其实际应用，所以本文假定协议实现中仅存在单个错误。在这个假设条件下Ghedamsi[3]中的算法具有普适性且效率是上面介绍的几种方法中最好的，但是该算法考虑的过于宽泛，没能充分利用测试过程中给出的判定信息和不同的测试序列所具有的一些特点。UIO方法是测试序列生成方法中最常用的，本文假定运行错误诊断前的测试序列是用UIO方法生成的。本文算法的改进之处为：首先利用UIO测试序列给出的判定消息减少可能的错误转换集的大小，然后利用测试序列执行后观察到的可能的错误转换后的输入/输出减少了需要枚举的转换错误。本文的改进算法能极大地降低算法的运行时间，这点通过理论和实验得到了验证。

本文安排如下：第2节是对文献[3]的错误诊断算法的简单介绍；第3节用一个具体的例子来说明文献[3]的算法应用到基于UIO方法生成的测试序列上的不足之处，并在说明中同时给出了本文的算法；第4节给出文献[3]中算法和本文算法的比较结果；最后总结全文。

2 术语定义和已有算法的介绍

定义 1 有限状态机(FSM) M 是一个五元组：$M=(I,O,S,\delta,\lambda)$，其中I是输入符号集，O是输出符号集，S是状态符集，I，O，S均为非空集合；s_0 是初始状态且 $s_0 \in S$；

2005-07-22 收到，2006-01-25 改回
国家自然科学基金重大研究计划项目(90104010)，国家自然科学基金项目(60241004)，国家 973 计划项目和中国科学院计算机科学重点实验室基础研究基金资助课题

$\delta: S\times I\to S$ 是状态转移函数； $\lambda: S\times I\to O$ 是输出函数。$s_i \xrightarrow{x/y} s_j$ 表示一个转换， s_i 称作该转换的头状态， s_j 称作该转换的尾状态，x是该转换的输入，y是该转换的输出。一般将有限状态机M用一个图 $G=(V,E)$ 来表示，图的顶点集合V 代表状态机M的状态集，图中的边集合代表M的转换集合，边上的标号是对应转换的输入/输出。

下面给出错误的定义，也就是错误模型：

(1) 输出错误：当一个转换发生时，输入和尾状态都符合描述，但输出不符合；

(2) 转换错误：当一个转换发生时，输入和输出都符合描述，但到达的尾状态不同；

假定实现中只存在单个错误(单个转换错误或者单个输出错误)，且已有测试套 $\mathrm{TS}=\{\mathrm{tc}_1,\cdots,\mathrm{tc}_p\}$，其中每个 tc_i 构成一个测试序列，包含有 m_i 个输入 $i_{i,1}i_{i,2}\cdots i_{i,mi}$，预期的输出序列记为 $o_i=o_{i,1}o_{i,2}\cdots o_{i,mi}$，其中 $o_{i,j}$ 表示在输入 $i_{i,j}$ 后的预期输出。用该测试套进行实际测试后得到每个测试序列的实际输出序列 $\hat{o}_i=\hat{o}_{i,1}\hat{o}_{i,2}\cdots\hat{o}_{i,mi}$。

定义2　若在 o_i 和 $\hat{o}_i$ 中 $o_{i,j}\neq\hat{o}_{i,j}$ 则称为症状(symptom)，对应描述中的转换 $t_{i,j}$ 则称为可疑转换(symptom transition)。

定义3　如果所有症状对应同一个可疑转换，则将该可疑转换称为唯一可疑转换(ust)，由该唯一可疑转换产生的输出叫做唯一症状输出(uso)。

定义4　观察到某个症状e之前测试序列所经过的描述中给出的转换构成该症状的冲突集记为 CS(e)。

文献[3]中算法一共分 4 步：

第 1 步　生成所有的症状。

第 2 步　对每一个观察到的症状产生冲突集。这一步直观上的意义就是可能存在的错误肯定发生在症状之前。取这些冲突集的交集作为初始候选集(ITC)，并且将 ITC 划分为两个非交子集合，一个子集称为 ustset，仅包含唯一可疑转换(ust)，另外一个子集则称为最终候选集(FTC)。

第 3 步　分开处理 ust 和 FTC。对 ust 的处理如下，遍历测试套 TS 中的所有转换，若该转换是 ust 则检查观察到的输出是否是 uso，否则检查观察到的输出是否和预期的一致，只有在测试套 TS 中的所有转换对应的回答都为是的时候才将 ust 输出作为候选诊断集，直观上看这一步就是在检查 ust 是否能产生所有观察到的输出序列。对 FTC 的处理如下，为 FTC 中的每个转换 t_k 枚举所有可能的转换错误，并检验引入该错误是否能产生所有观察到的输出序列，若回答是的话则将该转换错误对应的尾状态加入 $\mathrm{EndStates}_k$ 集合，最后每一个非空的 $\mathrm{EndStates}_k$ 集合都对应着实现中可能存在的转换错误。

第 4 步　由上面的 3 步可以得到一个可能的错误集合，为了定位出错误，需要再设计一些测试用例来区分这些错误。

3　本文的算法

上节中介绍的算法时间开销主要是在对 FTC 的处理上。原始算法的第 1 步和第 2 步并没有充分利用测试套执行过程中给出的全部信息，而仅仅只利用了症状这一点点信息来生成初始候选集 ITC，这样 ITC 可能包含多余的转换，而第 3 步由于在枚举检验错误，是比较耗时的，由此要求前两步给出的 ITC 应该尽可能的小。本文的算法首先利用 UIO 测试序列给出的判定消息最大程度地减少了 ITC 的数量，接着利用观察到的输入/输出序列中可能的错误转换后的输入/输出减少了需要枚举的转换错误，从而提高了算法的总体执行效率。

下面先给出后文要用到的一些定义。

定义5　若在 o_i 和 $\hat{o}_i$ 中 $o_{i,j}\neq\hat{o}_{i,j}$ 且对任意 $1\le k<j$ 有 $o_{i,k}=\hat{o}_{i,k}$ 则称 $o_{i,j}\neq\hat{o}_{i,j}$ 为测试序列 tc_i 的初始症状。

定义6　标号 x/y 的头状态集$H(x/y)$={转换t_i的头状态|转换t_i的标号是x/y}。

定义7　状态 s_i 的UIO序列，记作 UIO_i，是一个特定的从 s_i 开始的输入输出序列 $\{i_{k,1}/o_{k,1},\cdots,i_{k,r}/o_{k,r}\}$；不存在 $s_j\neq s_i$,使得 UIO_i 可以从 s_j 出发获得。

接下来分析文献[3]的算法的具体不足之处。并在此基础上提出改进算法。

首先，算法第 1 步没有必要生成所有的症状，而只需要生成所有的初始症状即可，这可以通过下面一条定理得出。

定理 1　$\bigcap\limits_{i\text{是症状}}\mathrm{CS}(i)=\bigcap\limits_{j\text{是初始症状}}\mathrm{CS}(j)$。

证明　将所有症状和初始症状按照其所在的测试序列 tc_m 划分为 n 个非交子集，记这样划分后所有症状集 $A=\bigcup\limits_{i=1}^{n}A_i$, 其中$A_i$={症状 j | j 属于测试序列tc_i}，所有初始症状集 $B=\bigcup\limits_{i=1}^{n}B_i$，其中$B_i$={$\mathrm{tc}_i$的初始症状}，每个 B_i 仅包含一个元素，由定义可知初始症状一定是一个症状，所以有 $B_i\subseteq A_i$，再由冲突集的定义知 $\mathrm{CS}(B_i)=\bigcap\limits_{p\in A_i}\mathrm{CS}(p)\lim\limits_{x\to\infty}$。从而有 $\bigcap\limits_{j\text{是初始症状}}\mathrm{CS}(j)=\bigcap\limits_{i=1}^{n}\mathrm{CS}(B_i)=\bigcap\limits_{i=1}^{n}\bigcap\limits_{p\in A_i}\mathrm{CS}(p)=\bigcap\limits_{i\text{是症状}}\mathrm{CS}(i)$，结论得证。

其次，在测试套的执行过程中动态给出的判定信息对排除错误是很有用的，举个例子来说，在图 1 所示的有限状态机上，右边是每个状态的 UIO 序列，则每个转换对应的 UIO 测试序列(如何用 UIO 方法来生成这些测试序列可以参考文献[1])如表 1 所示。

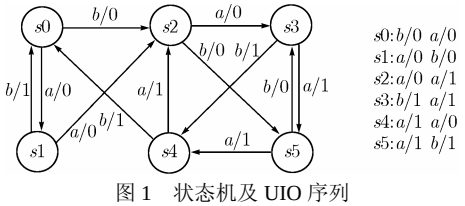

图 1　状态机及 UIO 序列

Fig.1 An example FSM and UIO sequence of each state

表 1 用 UIO 方法生成的图 1 中 FSM 的测试序列

Tab.1 UIO test sequences of the FSM in Fig.1

被测转换	测试序列	预期输出	观察到的输出
$s0\xrightarrow{a/0}s1$	raab	000	000
$s0\xrightarrow{b/0}s2$	rbaa	001	001
$s1\xrightarrow{a/0}s2$	raaaa	0001	0001
$s1\xrightarrow{b/1}s0$	rabba	0100	0100
$s2\xrightarrow{a/0}s3$	rbaba	0011	00$\underline{0}$1
$s2\xrightarrow{b/0}s5$	rbbab	0011	0011
$s3\xrightarrow{a/1}s5$	rbaaab	00111	0011$\underline{0}$
$s3\xrightarrow{b/1}s4$	rbabaa	00110	00$\underline{0}$1$\underline{1}$
$s5\xrightarrow{a/1}s4$	rbbaaa	00110	00110
$s5\xrightarrow{b/0}s3$	rbbbba	00011	00011
$s4\xrightarrow{a/1}s2$	rbabaaa	001101	00$\underline{0}$1$\underline{1}$1
$s4\xrightarrow{b/1}s0$	rbabbba	001100	00$\underline{0}$1$\underline{1}$0

上表同时给出了实际测试时观察到的输出，并已用下划线标出了所有的症状，反斜体标示的是初始症状。在这 12 个测试序列中有 7 条观察到的输出和预期输出一致，通过下面一个定理可以看出这些信息是非常有用的。

定理 2 假定实现中只有单个转换错误，记在描述中该转换为 $r\xrightarrow{e}i$，而在实现中变为 $r\xrightarrow{e}j$，则用 UIO 方法生成的测试该转换 $r\xrightarrow{e}i$ 的测试序列未能检测到该错误的情况当且仅当存在从状态 j 出发的长度大于等于 0 的路径 A，B，并且 A 路径的尾状态为 r，$\mathrm{UIO}_i=A\circ e\circ B$，如图 2 所示。

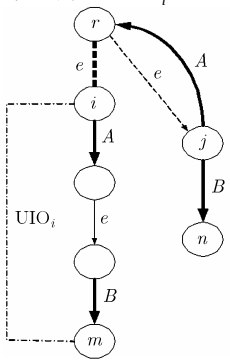

图 2 UIO 方法未能检测的错误

Fig.2 Undetected fault using UIO-method

证明 显然若有图 2 所示的情形，即实现中存在一个转换错误 $r\xrightarrow{e}j$，A和B是长度大于等于 0 的输入/输出序列(每个输入/输出对应有限状态机中的一条边)，e是单个输入/输出，在UIO_i序列中标记为e对应的边的头尾状态可以是任意状态，则测试该转换 $r\xrightarrow{e}i$ 的测试序列给出的输出肯定和预期的一样，该错误未能被检测到。

反之，若测试序列未能检测出该转换错，则根据UIO_i的定义知，这种情况一定是这个转换错误导致了j状态也可产生UIO_i序列，从而使得该错误未能被检测到，而这只能发生在状态j有两条分别标记为A和B的路径，且A路径将状态机从j带到状态r，恰如图 2 所示的那样。综上定理 2 得证。

推论 1 假定实现中只有单个转换错误，若UIO_i序列中不包含标号为e的边，则由UIO方法得到的测试转换 $r\xrightarrow{e}i$ 的测试序列若给出判定通过(观察的输出和预期的一致)，那么该转换在实现中一定是正确的。

该推论直接用反证法由定理 2 即可得出，由推论 1 我们可以从表 1 中给出的判定结果中得知转换 $s0\xrightarrow{b/0}s2$，$s1\xrightarrow{b/1}s0$，$s2\xrightarrow{b/0}s5$，$s5\xrightarrow{b/0}s3$ 在实现中一定是正确的，我们将运用推论 1 能判定出一定正确实现了的转换集合叫做 TTS(True Transition Set)。有了 TTS 集合我们就可以缩小初始候选集(ITC)。在原始算法中给出的 ITC={ $s0\xrightarrow{b/0}s2$，$s2\xrightarrow{a/0}s3$ }，而利用已获知的信息能排除掉 $s0\xrightarrow{b/0}s2$，从而得到只含一个元素的 ITC，当状态数目和转换数目较大时这个改进能使算法的执行效率大大改善。

由上面的讨论知文献[3]中算法并没有利用 UIO 方法生成的测试序列自身所具有的特点(定理 2 和推论 1)，本文考虑到这点后提出改进算法的前两步处理过程为：先生成所有的初始症状并运用推论 1 得出实现中能判断为正确的转换集 TTS，接着对每一个观察到的初始症状产生冲突集，取这些冲突集的交集作为初始候选集(ITC)，并且从 ITC 中删除已在 TTS 中出现的转换，然后将 ITC 划分为两个非交子集合，一个子集称为 ustset，仅包含唯一可疑转换(ust)，另外一个子集则称为最终候选集(FTC)。处理 ust 的算法如下：

```
Procedure ust-processing(ust)
1   For all  tc_m ∈ TS  Do
2       For all  I_{m,n} ∈ tc_m  Do
3           If ( t_{m,n} = ust ) Then
4               If ( ô_{m,n} ≠ uso ) Then
5                   ustset = ∅ ; exit
6           Else If ( o_{m,n} ≠ ô_{m,n} ) Then
7                   ustset = ∅ ; exit
8       End
9   End
```

得出了最终候选集 FTC 后，文献[3]中算法的一个不足之处在于对 FTC 集中的任意一个元素 t_k 它都要尝试所有的可能状态，这是没必要的。可以利用 t_k 的下一个转换所对应的标号的头状态集来改进其处理方式，例如通过统计观察到的输出序列可以得知在测试套中 $s2\xrightarrow{a/0}s3$ 后跟的标号是 $a/1$ 和 $b/0$，对应的头状态集为 $H_1=\{s3,s4,s5\}$ 和 $H_2=\{s0,s2,s5\}$，两者的交集为 $s5$，该交集确定了该转换错误只能是 $s2\xrightarrow{a/0}s3$ 变到 $s2\xrightarrow{a/0}s5$，而在文献[3]的算法中仍然要尝试所有其他的 5 个状态并逐个去排除。其正确性可以由下面一条定理的证明得出。

定理 3 假定实现中只有单个转换错误，t_k 是 FTC 中的一个元素，统计观察到的输出后得到测试序列中第一次出现的 t_k 后跟的标号集合是 $L_{t_k}=\{i_1/o_1,i_2/0_2,\cdots,i_l/0_l\}$，则 $\mathrm{EndStates}_k\subseteq\bigcap_{j=1}^{l}H(i_j/o_j)$。

证明 任取 $\mathrm{EndStates}_k$ 的元素 s'，则在实现中存在转换

错误，使得 t_k 的尾状态变为 s'，若 $s' \notin \bigcap_{j=1}^{l} H(i_j/o_j)$，则说明集合 L_{t_k} 中的某个标号必然不属于以 s' 为头状态的转换的标号集合，这直接与题设矛盾，由此 $\text{EndStates}_k \subseteq \bigcap_{j=1}^{l} H(i_j/o_j)$。证毕

另外在检查 FTC 的元素是否能产生观察到的输出时，文献[3]中算法总是按照固定的顺序去处理测试序列，而没有考虑到某些测试序列未经过 t_k 转换(显然对这种测试序列重新检查是没有必要的)。比如 $s0 \xrightarrow{a/0} s1$ 对应的测试序列根本不经过 $s2 \xrightarrow{a/0} s3$ 这个转换，那么检查 $s2 \xrightarrow{a/0} s3$ 时这条测试序列就不需要重新检查了。综合上面两点本文提出处理 FTC 集中元素的算法如下：

```
Procedure Improved_findendingstates(FTC)
1    For all tc_m ∈ TS Do
2        For all I_{m,n} ∈ tc_m Do
3            If ( t_{m,n} = t_k ∈ FTC ) Then
4                If t_k is the first time occurred in this test sequence Then
5                    {Record I_{m,n+1}/Ô_{m,n+1} into the corresponding set L_{t_k}; exit}
6            End
7    End
8    For all t_k in FTC Do
9        SS = {s | s ∈ S & s 不是t_k的尾状态}
10       For all i/o label in L_{t_k} Do
11           SS = SS ⋂ H(i/o)
12       End
13       EndStates_k = ∅
14       For all state s ∈ SS Do
15           flag = true
16           For all tc_m ∈ TS & t_k ∈ tc_m Do
17               start = s_0
18               For all I_{m,n} ∈ tc_m Do
19                   If ( λ(start, I_{m,n}) ≠ ô_{m,n} ) Then
20                       {flag = false; goto 14}
21                   If ( t_{m,n} = t_k ) Then start = s
22                   Else start = δ(start, I_{m,n})
23               End
24           End
25           If (flag=true) Then
26               EndStates_k = EndStates_k ⋃ {s}
27       End
28   End
```

从以上的分析过程可知，改进算法由以下 4 步组成：

第 1 步　先生成所有的初始症状并运用推论 1 得出实现中能判断为正确的转换集 TTS。

第 2 步　接着对每一个观察到的初始症状产生冲突集，取这些冲突集的交集作为初始候选集(ITC)，并且从 ITC 中删除已在 TTS 中出现的转换，然后将 ITC 划分为两个非交子集合，一个仅包含唯一可疑转换(ust)。

第 3 步　用 ust-processing(ust)来处理 ust，Improved_findendingstates(FTC)来处理 FTC，最后非空的 ustset 代表可能的输出错误，每一个非空的 EndStates_k 集合都对应着实现中可能存在的转换错误。

第 4 步　由上面的 3 步可以得到一个可能的错误集合，为了定位出错误，需要再设计一些测试用例来区分这些错误，相关工作参考文献[3]。

4　实验结果

共做了 2 个实验来比较本文算法和文献[3]中算法，实验 1 是在图 1 给出的有限状态机上给出的，用 UIO 方法生成的测试序列如表 1 所示。实验 2 是在 TCP 协议对应的有限状态机(图 3)上给出的(该图摘自文献[7])，后一个实验是为了验证该方法对实际的通信协议的普适性。

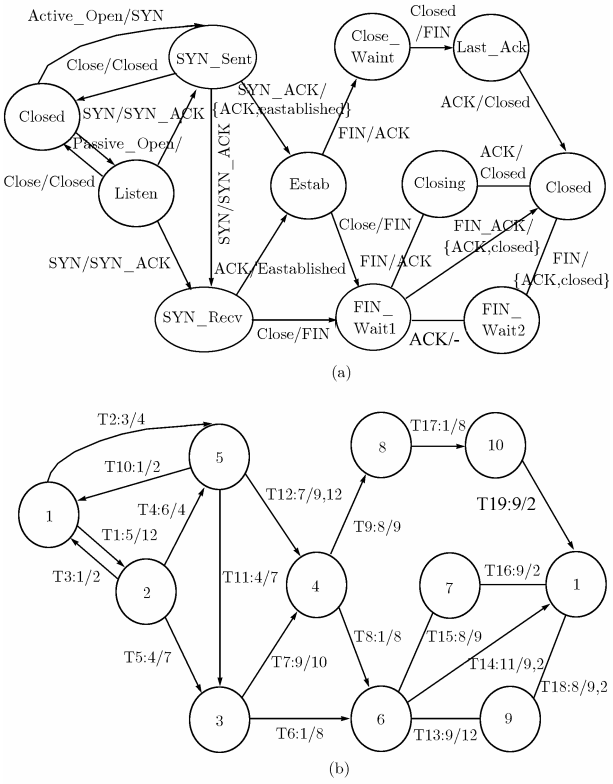

图 3　TCP 协议的 FSM

Fig.3 The TCP FSM

图 3 中每个状态的 UIO 序列见表 2。

表 2　图 3 的 FSM 的 UIO 序列

Tab.2 UIO sequence of each state in Fig.3

状态	UIO 序列	状态	UIO 序列
1	T1:5/12	6	T13:9/12
2	T3:1/2	7	T16:9/2
3	T6:1/8	8	T17:1/8,T19:9/2
4	T9:8/9	9	T18:8/9,2
5	T11:4/7	10	T19:9/2,T1:5/12

用 UIO 方法生成的测试序列如表 3 所示。

表 3 图 3 中 FSM 用 UIO 方法生成的测试序列

Tab.3 UIO test sequences of the FSM in Fig.3

被测转换	测试序列	被测转换	测试序列
T1	T1,T3	T11	T2,T11,T6
T2	T2,T11	T12	T2,T12,T9
T3	T1,T3,T1	T13	T1,T5,T6, T13,T18
T4	T1,T4,T11	T14	T1,T5,T6, T14,T1
T5	T1,T5,T6	T15	T1,T5,T6, T15,T16
T6	T1,T5,T6,T13	T16	T1,T5,T6, T15,T16,T1
T7	T1,T5,T7,T9	T17	T2,T12,T9, T17,T19,T1
T8	T2,T12, T8,T13	T18	T1,T5,T6, T13,T18,T1
T9	T2,T12,T9, T17,T19	T19	T2,T12,T9, T17,T19,T1
T10	T2,T10,T1		

比较的 3 个参数分别是 FTC 集的大小，算 EndStates_k 所需要重新检查的总的转换数目和需要枚举的状态数目。之所以比较这 3 个参数是因为它们是影响效率的关键参数。实验步骤如下：先枚举给定的有限状态机中所有的单个转换错误，针对每个错误都用 UIO 方法生成的测试序列来检测错误，若检测到错误则分别调用本文算法和文献[3]中算法来给出错误诊断。表 4 是本文得到的实验结果(参数都是取平均值之后的结果，符号'/'之前的数值是改进算法得到的，之后的数值是原始算法得到的)。

表 4 实验结果

Tab.4 The experimental results

	FTC 集元素个数	重新检查的转换数目	需要枚举的状态个数
图 1 的 FSM	1.2/2.4	31/225	1.7/5
TCP 的 FSM	1.1/3.0	65/763	5.5/9

从表 4 中可以看出对每一个 UIO 方法能检测出来的错误，改进算法基本上能把候选冲突集限定在单个转换上，这个改进对提高原始算法的效率是最关键的，对 FTC 的处理过程中通过逆向标定使得需要枚举的状态数目减少的改进也是很有效的，而且随着状态机中的状态数目和转换数目的增大，改进效果越明显。

5 结束语

本文充分挖掘了用 UIO 方法生成的测试序列测试过程中给出的判定信息，利用这些信息并结合文献[3]中的算法提出了适用于 UIO 方法生成的测试序列的错误诊断算法，该算法的效率极高，从第 3 节的分析过程和第 4 节的实验结果都能看出其有效性。在用 UIO 方法生成的测试套检测出错误后本文的改进算法能高效完全的定位实现中的单个错误。

参 考 文 献

[1] Lee D, Yannakakis M. Principles and methods of testing finite state machines——A survey. *Proc. IEEE*, 1996, 84: 1090−1126.

[2] 龚正虎. 计算机网络协议工程. 长沙:国防科技大学出版社，1993.12.

[3] Ghedamsi A, Bochmann G Von. Test result analysis and diagnostics for finite state machines. Proceedings of the 12th International Conference on Distributed Computing Systems, Yokohama, Japan, 1992: 244−251.

[4] Lee D, Sabnani K. Reverse-engineering of communication protocols. Proceedings of the International Conference on Network Protocols, San Francisco, California, USA, 1993: 208−216.

[5] Miller R E, Arisha K A. Fault identification in networks by passive testing. Proceedings of the 34th Annual Simulation Symposium, Seattle, WA, USA, 2001: 277−284.

[6] Guo X H, Zhao B H, Qian L. Fault Identification by Passive Testing. Telecommunications and Networking-ICT 2004, Fortaleza, Brazil, 2004, 3124: 826−834.

[7] Kim Myungchul, *et al*.. A dynamic protocol conformance test method. *Journal of Systems and Software*, 2003, 67(1): 31−43.

赵保华： 男，1947 年生，教授，博士生导师，主要研究方向为协议理论工程、无线传感网络.

钱 兰： 女，1981 年生，博士生，研究方向为协议理论与工程.

郭雄辉： 男，1981 年生，博士生，研究方向为协议理论与工程.