基于扩展有限状态机测试中测试输入数据自动选取的研究

张 涌 钱乐秋 王渊峰

(复旦大学计算机科学系 上海 200433)

摘 要 扩展有限状态机(EFSM)模型是有限状态机(FSM)模型的一个扩展,它在 FSM 模型的基础上增加了变量、操作以及状态迁移的前置条件,通过 EFSM 我们可以更加精确地刻画软件系统的动态行为.基于 EFSM 的测试可以应用到许多领域,因此具有重要的研究价值和实际意义.许多研究人员已经提出了基于 FSM 测试的测试输入序列的构造方法,但基于 EFSM 的测试与 FSM 相比由于变量和状态迁移的前置条件的引入,增加了构造其测试输入的复杂性.我们认为基于 EFSM 测试的测试输入应该包含两个部分:即测试输入序列以及该输入序列上包含的输入变量的确定值(测试输入数据).手工选取这些测试数据的工作十分繁琐,极大地增加了测试的花费,因此自动选取这些测试数据可以大大提高实际测试工作的效率.该文提出一种基于 EFSM 测试的测试数据自动选取方法,该方法利用两个关键的步骤:①区间削减和②分段梯度最优下降算法来自动选取测试数据.实验表明利用该方法可以自动选取大部分的测试数据,并且收敛速度较快;在某些无法得到确定解的情况下,区间削减也可以为测试人员提供一个较小的输入变量取值区间,方便了测试人员从中手工选择测试数据.

关键词 软件测试;扩展有限状态机;测试输入数据自动选取;区间削减;分段梯度最优下降算法

中图法分类号 TP311

Automatic Testing Data Generation in the Testing Based on EFSM

ZHANG Yong QIAN Le-Qiu WANG Yuan-Feng

(*Department of Computer Science, Fudan University, Shanghai* 200433)

Abstract Extended Finite State Machine (EFSM) is an extension of Finite State Machine (FSM). It add variables, operation and the precondition of transitions on the basis of FSM, and we can characterize the dynamic behavior of software system more precisely using EFSM. Testing based on EFSM, which can be used in many fields, have great research value and practical significance. Many researchers have proposed many testing sequence generation methods based on FSM, but this problem is more complicated in EFSM context. We consider that the testing input of the testing based on EFSM should include two parts: testing sequences and the determinate input variables' value in the sequences, which is called testing data. The work of selection testing data by hand is very tedious and may increase the cost of testing. If we can generate those testing data automatically, the efficiency of testing can be improved and the cost can be reduced also. In this paper, we propose an automatic testing data generation method in the testing based on EFSM. In this method, we use two key phases, interval narrowing and subsection gradient optimal descent algorithms to generate testing data automatically. In the interval-narrowing phase, we design a group of interval narrowing operators to reduce the variables' value scope. If the precondition in the state transitions is simple, we can get the testing data in this phase. In the second

收稿日期:2001-08-03;修改稿收到日期:2002-02-08.**张 涌**,男,1973 年生,博士研究生,研究方向为软件测试及其自动化、面向对象软件技术.E-mail: yongzhang@etang.com.**钱乐秋**,男,1942 年生,教授,博士生导师,研究领域包括构件技术、软件构架、软件测试等.**王渊峰**,男,1974 年生,博士研究生,研究方向为构件技术、软件构件库.

phase, we use subsection gradient optimal descent algorithms to generate testing data from reduced variables' value scope. The experiment shows that we can generate most of testing data automatically using this method, and has speedy convergence. In the situation of having no solution of input variables, interval narrowing can provide small intervals of the value of input variables for tester, which would be convenient for tester to select testing data from those small intervals.

Keywords software testing; extended finite state machine; automatic testing data generation; interval narrowing; subsection gradient optimal descent algorithm

1 引 言

扩展有限状态机(EFSM)模型是对有限状态机(FSM)模型的一个扩展,它在 FSM 模型的基础上增加了变量、操作以及状态迁移的前置条件,通过 EFSM 我们可以更加精确地刻画软件系统的动态行为,因此它可以被广泛应用于通信协议软件、嵌入式系统、面向对象软件中对象行为及对象之间的交互行为建模中去,同时在面向对象设计领域已经被广泛使用的 UML 中的状态图也与 EFSM 十分相似,因此基于 EFSM 的测试技术可以应用到许多领域,具有重要的研究价值.基于 FSM 的测试问题已经有了大量的研究工作[1~4],我们可以对基于 FSM 的测试方法进行调整来解决基于 EFSM 模型测试的问题,但是这种调整并不像表面那样简单.首先,状态迁移的行为与变量相互关联;其次,状态迁移之间通过迁移上的操作以及变量而产生交互作用.我们认为基于 EFSM 的测试输入应该包括两个部分:(1)测试输入序列.它由一系列的输入/输出消息序列组成;(2)施加该测试输入序列之后所导致的被执行的状态迁移上的输入变量值.我们称之为测试数据.在应用了某些方法产生了基于 EFSM 的测试输入序列之后,因为这些测试输入序列的执行还与输入变量相关,这些测试数据必须要满足该测试输入序列所引发的状态迁移上的前置条件才能保证测试输入序列在测试工作中的执行.因此为了使这些测试输入序列能够应用在实际测试工作中,我们必须要确定测试输入序列中包含的每个状态迁移上的输入变量的值.手工确定这些输入变量值的工作十分繁琐,耗费巨大,因此如何利用机器自动选取输入变量值是一个很有实际意义的研究课题.

本文提出一种基于 EFSM 得到的测试输入序列中包含的输入变量值的自动选取方法——ADS 方法,利用该方法可以自动选取满足输入测试序列所引发的状态迁移上的约束条件的输入变量的值,若在利用该方法无法自动确定输入变量值的情况下,利用该方法也可以得到经削减的输入变量的取值区间,以便于测试人员从较小的取值区间中选取合适的值使测试输入序列得以执行.

本文的研究基础是基于 EFSM 的测试输入序列已经确定,并且这些测试输入序列的可执行性已经在选取时得到确认,即只要选择了合适的输入变量值,这些测试输入序列就一定能够得到执行.

2 相关研究工作

一些研究人员已经对基于 EFSM 的测试输入序列的产生方法进行了研究.基于 EFSM 的测试输入序列的产生的难点是要进行可执行性分析,对于可执行性问题的解决方法主要分为两种:直接法和间接法.Ramaligom 等人[5]提出使用上下文独立的单一输入序列(CIUS)的概念来间接地解决可执行性问题,并根据 CIUS 来构造测试输入序列.Naik[6]等人提出扩展单一输入输出序列(EUIO)的概念来解决可执行性问题,该方法与 CIUS 方法相似.这些方法的缺点是 EFSM 中有些目标状态不一定存在 CIUS 和 EUIO,因此阻碍了其应用范围.Huang 等人[7]提出了从 EFSM 构造测试输入序列的方法 UIO_E,该方法直接考虑到变量以及前置条件对测试输入序列产生的影响,并利用状态迁移的可执行性分析产生测试输入序列,因此是一种直接方法,但该方法对 EFSM 进行了简化,在该文所述的 EFSM 模型中仅存在环境变量(即某些内部变量),这些环境变量的初始值都是常数,因此可以用符号执行的方法进行可执行性分析,而没有考虑到各状态迁移中所存在的输入变量.国内也有一些学者对 EFSM 测试输入序列的产生方法进行了研究[8,9].

以上这些研究都是关于如何构造基于 EFSM 测试的测试输入序列.本文的研究内容是关于测试

输入的第二个部分——测试数据的产生,即在测试输入序列产生之后,如何自动选取测试输入序列所引发的状态迁移上的输入变量的值,只有在这些输入变量的值确定下来之后,才能构造出完整的测试用例来对系统进行测试.

对于如何产生满足程序中某条路径上的约束条件的输入值的问题,国内外已经有了一些研究成果,例如,Korel[10]提出使用函数最小化方法来自动产生测试数据;Gupta[11]提出使用松弛技术来产生测试数据;王志言等人[12]利用区间算术的方法来产生测试数据;北京航空航天大学在他们的 Ada 程序测试工具中使用了函数最小化方法产生测试数据[13];兰毓华等人[14]根据 Z 语言的规约说明导出线性不等式组,然后对该不等式组进行求解来产生测试数据.这些研究工作所解决的问题与本文所要解决的问题存在某些的相似性.由于在本文所研究的问题中,测试输入序列可以引发多个状态迁移的执行,故存在多次变量输入并且约束条件的数目较多且约束条件也较复杂,因此本文讨论的问题更为复杂.

3　EFSM 模型、基于 EFSM 测试的基本思想以及所要解决的问题

EFSM 是对 FSM 的扩展,它在 FSM 的基础上增加了变量、状态迁移的前置条件以及状态迁移所引起的操作.

定义 1.　一个 EFSM 是一个六元组$\langle S,S_0,I,O,T,V\rangle$;其中 S 是一个非空的状态集合,S_0 是初始状态,I 是一个非空的输入消息集合,O 是一个非空的输出消息集合,T 是一个非空的状态迁移集合,V 是变量的集合.T 中的每一个元素 t 是一个六元组$\langle Head(t),Tail(t),I(t)(input\ variables\ list),P(t),operation,O(t)(output\ variables\ list)\rangle$,其中 $Head(t)$是状态迁移 t 的出发状态,$Tail(t)$是状态迁移 t 的到达状态,$input\_message$ 或者是 I 中包含的 EFSM 的输入消息或者为空;$P(t)$是状态迁移 t 执行的前置条件,它可能为空;$operation$ 是状态迁移中的操作,它由一系列的变量赋值语句或输出语句组成,其中不包含分支;$output\_message$ 或者是 O 中包含的 EFSM 的输出消息或者为空.EFSM 中的变量是一个三元组$\langle IV,CV,OV\rangle$,其中 IV 代表输入变量集合,输入变量可以由测试人员控制;OV 代表输出变量集合,CV 代表环境变量集合.既不是输入变量也不是输出变量的变量我们就称之为环境变量,环境变量就是某些局部变量或全局变量.OV 和 CV 不受测试人员控制,它的值由状态迁移中的操作来确定.

由于添加了状态迁移的前置条件,当 EFSM 处于某个状态迁移 t 的出发状态 $Head(t)$时,接收到特定的消息输入 $I(t)$,并且 $P(t)$=true 时,该状态迁移才会执行,并且状态转换到下一个状态 $Tail(t)$.

基于 EFSM 的测试是一种典型的黑盒测试方法,我们把系统的实现看作一个黑盒,通过对系统的实现施加一定的输入,观察其输出并与预期的输出相比较来判断系统功能是否已经正确实现.测试输入的选取可以随机选取,但是这样选取的输入对于错误的发现较为低效;另一种测试输入的选取方法是要求其满足一定的覆盖准则,对于 EFSM 测试来说常用的覆盖准则是状态全部覆盖或状态迁移全部覆盖,这样构造的测试输入针对性较强,能够覆盖大部分的错误.基于 EFSM 测试的输入应该包含两个部分:(1)测试输入序列集合(TS).它是一个以 EFSM 中的初始状态为起始点的输入消息序列的集合;(2)TS 中的每一条测试输入序列所引发的状态迁移上的输入变量的确定值.这些输入变量的值必须满足状态迁移上的前置条件,在本文中我们把状态迁移上输入变量的值叫做测试输入数据.利用测试输入序列我们可以获知 EFSM 中的哪些从初始状态出发的路径被用于测试;在得到了这些被测试的路径之后,为了使测试输入可以在实际测试工作中由被测程序执行,我们必须确定该路径上包含的状态迁移上的输入变量的值,否则该测试路径不能够被实际执行.输入变量值的选取可以归结为对如下问题的求解:给定输入变量的向量 $\boldsymbol{X}$ 以及其取值区间的向量 $\boldsymbol{R}$,求解满足约束条件集合 $C=(C_1,C_2,\cdots,C_n)$的输入变量的值,其中约束条件为该测试路径中包含的各状态迁移上的前置条件.本文主要讨论的内容是在得到一条测试输入序列之后如何自动选取该测试输入序列所引发的各状态迁移上的输入变量的值,对构造测试输入序列的方法不做讨论.

4　输入变量值的自动选取

对于从 EFSM 得到的测试输入序列,由于其所引发的状态迁移上可能存在有输入变量,因此我们必须要确定各个输入变量的值,才能把测试输入序列应用到实际的测试工作中去.本文提出一种输入

变量值自动选取的方法——ADS 方法.

ADS 方法分为 3 个步骤:

1. 初始化:把测试输入序列引发的各状态迁移中包含的表达式转化为正规表达式.

2. 区间削减:利用测试序列上的状态迁移中包含的前置条件对输入变量的取值区间进行削减,以利于后面的分段梯度最优下降算法能够快速收敛,并且如果在第 3 步中无法正确产生输入变量值的情况下,可以给测试人员一个较小的输入变量的取值空间,便于利用它来手工产生测试输入数据.

3. 输入变量值的确定:利用分段梯度最优下降算法自动选取测试数据.

下面我们对该方法进行详细的介绍.

4.1 初始化

定义 2. 对于赋值语句若其右部为只包含输入变量和(或)常量的表达式,对于条件语句若其左部和右部的表达式都仅包含输入变量和(或)常量,我们称这样的表达式形式为正规表达式.

在状态迁移中出现的变量种类包括输入变量、环境变量和输出变量,其中输出变量和环境变量的值都可以归结为由某些输入变量和(或)常量来确定,为了便于对输入变量的区间进行削减以及输入变量值的确定,我们首先要把测试输入序列引发的状态迁移包含的表达式转化为正规表达式.把表达式转化为正规表达式之后,我们就可以根据输入变量的取值空间确定状态迁移上的任意表达式所代表的取值区间.

此外,同一输入变量符有可能在不同的状态迁移的输入变量表中出现,为了表示它们代表的值不同,我们需要把每个状态迁移上的输入变量改写为 v_{t_i} 的形式以示区分,其中 v 为输入变量的标识符,t_i 是某个状态迁移的标识符.例如,若某个输入变量 x 在两个状态迁移 t_1 和 t_2 的输入变量列表中同时出现,则该变量代表的实际值可能会不相同,为了区分该变量的不同输入值,我们在初始化阶段把该变量分别改写为 x_{t_1},x_{t_2}.

在把表达式转化为正规表达式之前,我们首先要确定某个输入变量的影响范围,即使用该变量的替换到哪里结束.影响范围的确定要利用对测试输入序列引发的状态迁移进行数据流分析的结果.数据流分析的主要目的是找到输入变量的定义点以及使用点.输入变量的定义点就是输入变量出现在赋值表达式的左部的那些语句,记为 $def(x)$;变量的使用点可分为以下几类,若该变量出现在赋值语句的右部,则我们称该变量为计算使用,记为 *C-use*(x),若该变量出现在逻辑表达式中,则称该变量为谓词使用,记为 *P-use*(x),若该变量出现在状态迁移的输入变量列表中,则我们称该变量为输入使用,记为 *I-use*(x);我们对某一个变量的影响范围的确定基于如下思想:从该测试输入序列所引发的第一个状态迁移开始对每一个输入变量进行搜索,若在搜索过程中发现了该变量的 *I-use*,则对该变量的搜索结束,同时把所有该变量的出现替换为 v_{t_i} 的形式,然后对下一个输入变量进行搜索,在对第一个状态迁移的处理完成之后,取下一个状态迁移重复以上步骤;直到所有的状态迁移都已经处理完.

在完成这种输入变量的不同出现的转换之后,我们就可以进行状态迁移中出现的表达式向正规化表达式的转化.转换为正规形式的算法:在算法过程中,我们使用一个临时表 VDT,形如:

变量(V)	表达式(E)
被定义的变量	仅包含输入变量和(或)常量的表达式

该表包含了每一个变量在每次定义点处的定义表达式,在转化过程中,我们动态选取该表.我们从第一个状态迁移开始转化,若遇到的是操作(即赋值语句),则分析其右部的表达式,若其中包含变量,则对每一个变量后向查找表 VDT(因为后加入的表项代表该变量的最新的定义),找到该变量的定义表达式后,用该表达式替换该变量,在赋值语句的右部表达式完成了所有这样的替换之后,把该赋值语句的左部(变量标识)和右部(仅包含输入变量和常量的表达式)填入 VDT 表的对应项目中去;若遇到的是状态迁移的前置条件,则对其左部和右部的表达式进行类似以上说明的替换.

以下是该初始化算法伪代码:

```
Initialization(TT):
输入:
   TT: 保存候选测试输入序列所引发的状态迁移的
       链表
输出:
   NT: 转化后的状态迁移表
局部变量:
   VDT: 保存所有变量的定义表达式的表
   T: 指向 TT 中某个结点的指针
Begin
   T=TT.head;
   While T ! =null
      For each input variables x_i in T
         Effect_Range(x_I);//确定输入变量 x_I 的影
```

```
        响范围
        Replace x_I with x_{i_(t_i)}
        Update TT;
      End For
      T=T.next ;
    End While
    T=TT.head;
    While T! =null
      For each operation and precondition of T
        If operation
          For each variable in the right part of assignment statement
            Look up the VDT for it's definition expression E from back to front;
            Replace this variable with E;
            Add this assignment statement to VDT;
          End For
        Else
          For each variable in the left part and right part of logical expression
            Look up the VDT for it's definition expression E from back to front;
            Replace this variable with E;
          End for
        End if
      End for
      T=T.next;
    End While
  End
```

4.2　区间削减

在完成了初始化的工作之后,该测试输入序列所引发的状态迁移中的所有操作以及前置条件都已经转换为正规表达式,为了加快分段梯度最优下降算法的收敛速度,减小输入变量值取值区间,我们利用各状态迁移上的前置条件对输入变量的取值区间进行削减.前置条件为一系列的关系表达式或其合取或析取的形式.关系表达式形如:$d_1.opl.d_2$,其中 d_1,d_2 为算术表达式,opl 为关系运算符,可能存在的关系运算符包括:>,>=,<,<=,=,!=.我们设计了一套区间削减算子,利用该算子可以有效地削减输入变量的取值区间.在介绍区间削减算子之前,我们先介绍一下区间算术的基本概念.在区间算术中,任何一个数或者变量都可以用一个区间来表示,区间算术的基本运算+,-,*,/定义如下:

设有三个变量 z,x,y 分别表示为 $[z.bottom, z.top]$,$[x.bottom, x.top]$,$[y.bottom, y.top]$

若 $z=x+y$,则 $z.bottom=x.bottom+y.bottom$,$z.top=x.top+y.top$;

若 $z=x-y$,则 $z.bottom=x.bottom-y.top$,$z.top=x.top-y.bottom$;

对于乘除运算,我们要分区间定义:

若 $z=x*y$,

(1) 若 $x.bottom>=0$, $y.bottom>=0$,
则 $z.bottom=x.bottom*y.bottom$,
$z.top=x.top*y.top$;

(2) 若 $x.top<0$, $y.bottom>=0$,
则 $z.bottom=x.bottom*y.top$,
$z.top=x.top*y.bottom$;

(3) 若 $x.top<0$, $y.top<0$,
则 $z.bottom=x.top*y.top$,
$z.top=x.bottom*y.bottom$;

(4) 若 $x.bottom<0$, $x.top>0$ 并且 $y.bottom>=0$,
则 $z.bottom=x.bottom*y.top$,
$z.top=x.top*y.top$;

(5) 若 $x.bottom<0$, $x.top>0$ 并且 $y.top<=0$,
则 $z.bottom=x.top*y.bottom$,
$z.top=x.bottom*y.bottom$;

(6) 若 $x.bottom<0$, $x.top>0$ 并且 $y.bottom<0$, $y.top>0$,则

$$z.bottom=\begin{cases} x.bottom*y.top, & 若 x.bottom*y.top<x.top*y.bottom \\ x.top*y.bottom, & 否则 \end{cases};$$

$$z.top=\begin{cases} x.bottom*y.bottom, & 若 x.bottom*y.bottom>x.top*y.top \\ x.top*y.top, & 否则 \end{cases}.$$

对于除法的定义与乘法相似,需要分区间来讨论.

区间削减:我们可以根据某些约束条件对输入变量的取值区间进行削减.例如,若存在两个输入变量 $x_1\in[37,60]$,$x_2\in[50,70]$,约束条件为 $x_1>x_2$,则我们可以分别把 x_1,x_2 的区间根据该约束条件削减为 $x_1\in[51,60]$,$x_2\in[50,59]$.我们根据区间算术设计了一套区间削减算子,该区间削减算子利用状态迁移上的前置条件对输入变量的区间进行削减,这些算子根据关系表达式中的关系运算符分情况对输入变量的取值区间进行削减;对于关系表达式的左部和右部的算术表达式中都包含两个以上的输入变量的情况,由于左部和右部的算术表达式会出现算术运算符组合的情况,对于区间削减情况的讨论会变得异常复杂,因此对于这样的逻辑表达式

在本步骤中不作处理.由于篇幅所限,本文无法把所有的区间削减算子一一列出.

4.3 输入变量值的确定

在该步骤中,我们利用分段梯度最优下降算法来从测试输入序列中确定其引发的每个状态迁移中的输入变量的值.文献[10,13]中采用函数最小化方法来确定满足程序的某一条路径执行条件的输入变量的值.但是函数最小化方法是一种简单的搜索算法,其存在如下缺点:

(1) 在利用函数最小化方法求解时,每次只考虑一个约束条件,求出满足该约束条件的变量值后,再在当前变量值的基础上搜索满足下一个约束条件的变量值,这种搜索过程由于每次只考虑一个约束条件,因此存在多次回溯的概率较大;

(2) 利用函数最小化方法求解满足某一约束条件的输入变量的值时,每次只考虑一个变量,即改变该变量的值而使目标函数向正确的方向靠近,而其它变量的值保持不变,然后得到满足约束条件的一个解,再用该解作为求解下一个约束条件的解的搜索过程的起始点,因为每次只考虑一个变量以及一个约束条件,因此在利用上一个输入变量的解作为求解满足当前约束条件的变量值的搜索过程的起始点时,可能会存在符合下一个约束条件的解永远无法达到的情况,其收敛性较差.

因此我们认为函数最小化方法不适合求解复杂的约束问题.

在本文中我们使用约束条件来表示某条测试输入序列所引发的各状态迁移上的所有前置条件.

利用分段梯度最优下降算法我们首先需要确定目标函数,然后确定该函数在某个初始点梯度下降最快的方向并确定最优步长,利用梯度下降的方向以及最优步长对输入变量的值进行调整,然后判断新的输入变量的值是否满足所有的约束条件,若当前输入变量的值已经满足所有的约束条件则求解成功结束,否则利用当前的输入变量的值重新确定目标函数在该点处的梯度下降最快的方向并确定最优步长,对输入变量值进行调整,若这些变量的值满足所有的约束条件则成功结束,否则重复执行以上的迭代过程.

4.3.1 目标函数的确定

在确定目标函数时,我们借用 Korel[10] 提出的分支函数的思想.分支函数是一个实值函数,它是分支谓词到实值的一个映射.分支谓词相当于本方法中的状态迁移上的前置条件.分支函数如表 1 所示.当分支谓词为真时,分支函数 $f(x)$ 的值小于(或等于)零,当分支谓词为假时,$f(x)$ 的值大于零.在把分支谓词转换为分支函数时我们不考虑分支谓词中的逻辑运算符为"="的情况,因为我们可以根据该等式把某个输入变量用其它的输入变量来表示,并代入到其它包含该变量的不等式中.

表 1 分支函数

分支谓词	分支函数
$E_1(x)>E_2(x)$ $E_1(x)\geq E_2(x)$	$f(x)=E_2(x)-E_1(x)$
$E_1(x)<E_2(x)$ $E_1(x)\leq E_2(x)$	$f(x)=E_1(x)-E_2(x)$
$E_1(x)\neq E_2(x)$	$f(x)=-abs(E_1(x)-E_2(x))$
$RE_1(x)$ and $RE_2(x)$	$f(x)=\max(f_1(RE_1(x)),f_2(RE_2(x)))$
$RE_1(x)$ or $RE_2(x)$	$f(x)=\min(f_1(RE_1(x)),f_2(RE_2(x)))$
not $(RE_1(x))$	$f(x)=-f_1(RE_1(x))$

其中 E_1 和 E_2 代表算术表达式,RE_1 和 RE_2 代表关系表达式,$f_1(RE_1(x))$ 和 $f_2(RE_2(x))$ 分别代表 RE_1 和 RE_2 的分支函数.

我们可以把每一个状态迁移上的前置条件转化为分支函数,然后根据这些分支函数来确定目标函数 $F(\boldsymbol{X})$:

$$F(\boldsymbol{X})=\sum_{i=1}^{n} f_i(\boldsymbol{X}),$$

其中 $\boldsymbol{X}$ 代表输入变量组成的向量,n 为约束条件的数目.

根据分支函数的特性我们可以看到,当所有的约束条件全部都满足时,该目标函数的值必为小于或者等于零的值,更一般化,目标函数的值应小于零.注意目标函数小于零是有解的必要条件,该必要条件指导了我们向解值区间搜索的方向,即经过迭代后的输入变量值与前一次的输入变量值相比,应该使目标函数不断地向其最小值的方向前进.输入变量值迭代调整的依据是使目标函数的值不断地下降,直到达到某一点,使所有的约束条件都满足.利用目标函数进行满足约束条件的输入变量值的搜索过程如图 1 所示.

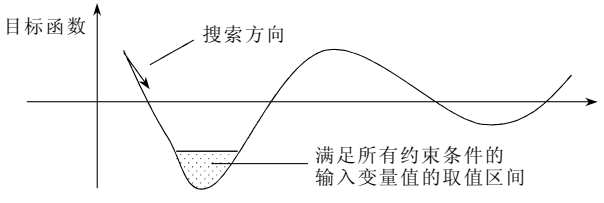

图 1 满足某些约束条件的输入变量值的确定过程

4.3.2 梯度下降方向的确定

某个函数的梯度可以定义为

$$\nabla F(\boldsymbol{X}) = \frac{\partial F}{\partial \boldsymbol{X}} = \left[\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \cdots, \frac{\partial F}{\partial x_n}\right]^{\mathrm{T}}.$$

由上式可见,梯度是一个 n 维向量.梯度方向就是函数 $F(\boldsymbol{X})$ 的法线方向,根据函数梯度的特性可知,在某一点附近,沿梯度方向函数的变化率最大.为了使目标函数值下降,我们应该选取负梯度方向作为搜索方向,使目标函数值下降最快,以更快地达到收敛.

负梯度方向的单位向量为

$$g = -\frac{\nabla F(\boldsymbol{X})}{\|\nabla F(\boldsymbol{X})\|},$$

其中 $\|\nabla F(\boldsymbol{X})\|$ 为梯度向量的模.

$$\|\nabla F(\boldsymbol{X})\| = \sqrt{\left(\frac{\partial F}{\partial x_1}\right)^2 + \left(\frac{\partial F}{\partial x_2}\right)^2 + \cdots + \left(\frac{\partial F}{\partial x_n}\right)^2} \tag{1}$$

4.3.3　最优步长的确定

在确定了梯度下降的方向之后,我们需要确定选择一个最优的步长,使搜索步骤减少以尽快达到目标函数的收敛.梯度下降法的迭代公式为

对于第 $k+1$ 次迭代,$\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{k} - h\nabla F(\boldsymbol{X}^{(k)})$ (2)

根据上式,从某点 $\boldsymbol{X}^{(k)}$ 出发,沿负梯度方向,取步长参数 h,下一步可以到达点 $\boldsymbol{X}^{(k+1)}$.根据式(1)我们可以把式(2)改写为

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{X}^{(k)} + h\|\nabla F(\boldsymbol{X}^{(k)})\| g^{(k)} = \boldsymbol{X}^{(k)} + \lambda g^{(k)} \tag{3}$$

$\lambda = h\|\nabla F(\boldsymbol{X}^{(k)})\|$ 为从 $\boldsymbol{X}^{(k)}$ 出发沿负梯度方向进行一维搜索的步长.由式(3)可得

$$F(\boldsymbol{X}^{(k+1)}) = F(\boldsymbol{X}^{(k)} + \lambda g^{(k)}) \tag{4}$$

设沿负梯度方向求使 $F(\boldsymbol{X}^{(k+1)})$ 为最小的步长 λ_k,则有

$$\begin{aligned}\min_{\lambda \geq 0} F(\boldsymbol{X}^{(k+1)}) &= \min_{\lambda \geq 0} F(\boldsymbol{X}^{(k)} + \lambda g^{(k)}) \\ &= F(\boldsymbol{X}^{(k)} + \lambda_k g^{(k)})\end{aligned} \tag{5}$$

利用分段梯度最优下降法进行输入变量值的迭代计算时,每一次迭代过程都要对步长 λ_k 进行适当的选取.由式(5)可知求最优步长 λ_k 时,实际上是对函数 $F(\boldsymbol{X}^{(k+1)})$ 求最小值,一般我们可以对式(4)求导且令其等于零,即

$$\frac{\mathrm{d}F(\boldsymbol{X}^{(k+1)})}{\mathrm{d}\lambda_k} = \frac{\mathrm{d}F(\boldsymbol{X}^{(k)} + \lambda_k g^{(k)})}{\mathrm{d}\lambda_k} = 0 \tag{6}$$

对于任意函数 $F(\boldsymbol{X})$,如果在点 $\boldsymbol{X}^{(k)}$ 附近展开成泰勒级数并且取二次近似,则有

$$F(\boldsymbol{X}^{(k+1)}) = F(\boldsymbol{X}^{(k)}) + \nabla^{\mathrm{T}} F(\boldsymbol{X}^{(k)})\Delta\boldsymbol{X} + 1/2(\Delta\boldsymbol{X})^{\mathrm{T}}\boldsymbol{A}\Delta\boldsymbol{X} \tag{7}$$

式中,$\Delta\boldsymbol{X} = \boldsymbol{X}^{(k+1)} - \boldsymbol{X}^{(k)}$,$A = \nabla^2 F(\boldsymbol{X}^{(k)})$ 为 Hessian 矩阵,Hessian 矩阵形如:

$$\boldsymbol{A} = \frac{\partial^2 F}{\partial \boldsymbol{X}^2} = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}.$$

将式(4)代入式(7),则有

$$F(\boldsymbol{X}^{(k)} + \lambda_k g^{(k)}) = F(\boldsymbol{X}^{(k)}) + \nabla^{\mathrm{T}} F(\boldsymbol{X}^{(k)})\lambda_k g^{(k)} + 1/2(\Delta\boldsymbol{X})^{\mathrm{T}}\boldsymbol{A}(\lambda_k g^{(k)}) \tag{8}$$

对式(8)求导且令其倒数等于零,则可得

$$\frac{\mathrm{d}F(\boldsymbol{X}^{(k)} + \lambda_k g^{(k)})}{\mathrm{d}\lambda_k} = \nabla^{\mathrm{T}} F(\boldsymbol{X}^{(k)}) g^{(k)} + (g^{(k)})^{\mathrm{T}}\boldsymbol{A}g^{(k)}\lambda_k = 0 \tag{9}$$

在确定的某点处,目标函数的梯度以及梯度的单位向量都是确定的值,因此我们可以求解式(9)中的方程式得到最优步长,该步长不用考虑梯度的方向性,因此其值应大于零,若得到的解为负数,应对其取绝对值.同时我们要考虑到输入变量的取值区间,当某一次迭代过程中,若出现了迭代后的某个输入变量值超出了其取值区间时,则该变量的值应该约束在其取值区间内.即

$$x_i^{(k+1)} = \begin{cases} x_i.bottom, & x_i^{(k+1)} < x_i.bottom \\ x_i.top, & x_i^{(k+1)} > x_i.top \end{cases}.$$

4.3.4　分段梯度最优下降算法

综上所述,我们可以得到求解满足所有约束条件的输入变量值的算法.在利用梯度下降法进行求解的过程中,如果函数存在多峰的情况,由于初始点的不合适选择,可能会得到某个局部最优点(即局部最小值点),若该局部最优点能够使所有的约束条件得到满足,那么我们可以得到确定的满足所有约束条件的输入变量的解,但若该局部最优点无法使所有的约束条件得到满足,那么我们无法在该方向上进一步迭代得到输入变量的解,因此初始点的选取对算法的收敛性有很大的影响.为了提高算法收敛的概率,我们对输入变量取值区间进行分割,从每一个被分割后的取值区间中选取一个初始点,然后从该初始点进行迭代,若某个区间无法得到正确解,则选取下一个区间进行迭代,实验表明这种方法提高了算法收敛的概率.我们把每个输入变量的区间平均分割为四个部分,即分割为$[x_i.bottom, 1/4 * x_i.top]$,$[1/4 * x_i.top, 1/2 * x_i.top]$,$[1/2 * x_i.top, 3/4 *$

$x_i.top]$，$[3/4*x_i.top, x_i.top]$，然后从每个输入变量的相应区间中选取一个数值组成初始点处的向量.

整个算法的步骤如下：

1. 选取一个未处理过的输入变量向量的取值区间，并在该区间内随机选取一个初始点 $\boldsymbol{X}^{(0)}$；若所有的取值区间都已经被处理过，则算法失败并退出；

2. 计算在向量 $\boldsymbol{X}^{(k)}(k=0,1,\cdots,n)$ 处的负梯度的单位向量 $\boldsymbol{g}^{(k)}$ 以及最优步长 λ_k；

3. 令 $\boldsymbol{X}^{(k+1)}=\boldsymbol{X}^{(k)}+\lambda\boldsymbol{g}^{(k)}$，并使迭代后的新的输入变量的值约束在其取值区间之内，即

$$x_i^{(k+1)}=\begin{cases}x_i.bottom, & x_i^{(k+1)}<x_i.bottom\\ x_i.top, & x_i^{(k+1)}>x_i.top\end{cases};$$

4. 若此时 $\boldsymbol{X}^{(k+1)}$ 能够满足所有的约束条件，则输出该向量并退出；若此时 $|\boldsymbol{X}^{(k+1)}-\boldsymbol{X}^{(k)}|<\eta$（$\eta$ 为任取的一个小的正数），则我们认为其梯度下降已经接近谷底，已经没有多少调整的空间，转(1)；否则，转(2)进一步迭代；

5 方法讨论及实验结果

本文提出的 ADS 方法在对输入变量值的求解过程中存在两个关键性的步骤：(1)区间削减；(2)分段梯度最优下降算法求解满足约束的输入变量值. 利用状态迁移上的前置条件进行区间削减可以为后面的梯度下降算法提供一个较小的输入变量的取值区间，从而加快其收敛速度；同时，在区间削减时如果前置条件中存在等式并且表达式比较简单，我们可以得到某些输入变量的确定的值，在梯度下降算法中把这些变量的值代入目标函数中，可以简化梯度下降算法的搜索过程. 对于非线性约束条件的求解问题还不存在有效的方法能够达到完全可解，因此在无解的情况下，区间削减可以为测试人员提供一个较小的变量取值区间，以便于测试人员手工选取测试数据，降低工作的强度.

对于梯度下降算法中所用到的导数，为了计算的方便性，我们可以对函数的导数取近似值来计算，即利用如下公式来进行计算：

$$\frac{\partial F}{\partial x_i}=[F(x_1,x_2,\cdots,x_i+\delta x_i,\cdots,x_n)-F(x_1,x_2,\cdots,x_i-\delta x_i,\cdots,x_n)]/2\delta x_i,$$

其中 δx_i 是一个大于零的微小值.

由于目前存在众多的数学函数库，因此在我们的实验系统中直接调用 MathCAD 中的库函数对导数进行计算. 此外，若规约说明中规定某些输入变量必须是整形时，我们对迭代后得到的变量值取整.

对于分段梯度最优下降算法我们讨论算法的终止性及收敛性. 在梯度下降算法中存在两种情况的执行终止：(1) 成功得解后终止；(2) 不成功时终止. 对于不成功的情况，我们在算法中添加了条件 $|\boldsymbol{X}^{(k+1)}-\boldsymbol{X}^{(k)}|<\eta$，也就是说在搜索过程中如果变量向量的改变量已经很微小时，表明该处的梯度很平缓，再进行搜索对于目标函数值的影响不会很大，因此就放弃在该方向上继续搜索. 对于任意的函数，它必存在某些点是局部(或全局)最小点，在这些点附近，变量向量的改变量很微小，因此对于求解不成功的情况该算法也必能够终止.

若算法成功得解，我们认为该算法收敛. 对于算法收敛的证明，由于函数的多样性，目前还不存在有效的算法对于所有的函数都是收敛的. 因此我们通过实验来说明其收敛情况并与函数最小化方法以及单纯的梯度下降法进行比较.

我们选择了 100 条测试输入序列进行实验，这些测试输入序列包含的平均约束条件数目为 6，其中既包含线性约束也包含非线性约束. 对这些测试输入序列我们分别使用函数最小化方法、松弛算法、单纯的梯度下降法，先进行区间削减然后使用单纯的梯度下降法，单独使用分段梯度最优下降法以及本文提出的 ADS 方法(区间削减＋分段梯度最优下降)对其自动生成测试数据，并对实验结果通过收敛情况、收敛时平均迭代次数以及收敛时的平均处理时间三个方面进行比较，实验数据如表 2 所示.

表 2 实验数据

算法	收敛情况	收敛时平均迭代次数	收敛时平均处理时间(ms)
函数最小化方法	61	33	831
松弛算法	57	40	811
梯度下降(未采用取值区间分段选择初始点)	70	20	712
区间削减＋梯度下降(未采用取值区间分段选择初始点)	80	9	650
分段梯度最优下降	81	12	684
ADS(区间削减＋分段梯度最优下降)	92	5	412

注：实验平台：CPU：PIII667，内存：128M.

从实验数据可知，与其它的方法相比，采用 ADS 方法得到的收敛概率最大，并且迭代次数最小，平均处理时间也最小. 函数最小化方法和松弛算法的实验结果基本相同并且结果不令人满意，并且松弛算法需要经过预测切片、识别输入依赖集合、计算预测余数等多个阶段，其执行代价也较多. 对于文献[14]中提出的算法来说，由于是利用线性方程组求解得到测试数据，可能存在大量的非线性的约束条件，因

此该方法的适应性不强.

6 结　论

手工选取测试数据的工作不仅繁琐,而且效率低下,使得软件测试过程中的花费大大增加,利用机器自动选取测试数据可以改变这种状况,因此该问题是一个很有实际意义的研究课题.本文提出一种基于 EFSM 测试的测试输入数据的自动选取方法,该方法通过区间削减和分段梯度最优下降算法自动选取测试输入所需的测试数据.实验表明,该方法在大多数情况下都能够得到一组确定的解,并且收敛速度较快.在某些无法得解的情况下,区间削减可以为测试人员提供一个较小的输入变量取值区间,便于测试人员从中手工选取测试数据,从而降低了测试开销.

参　考　文　献

1 Inna K, Ural H. Efficient checking sequences for testing finite state machines. Information and Software Technology, 1999, 41(11/12):799～812

2 Rezaki A, Ural H. Construction of checking sequences based on characterization sets. Computer Communications, 1995, 18(12):911～920

3 LIU Ji-Ren, Du Jun. Protocol conferemance test generation based upon multiple UIO sequence. Journal of Software, 1995, 6(Supplement):52～55(in Chinese)
(刘积仁,都　军. 基于多 UIO 序列的协议一致性测试生成. 软件学报, 1995,6(增刊):52～59)

4 Fujiwara S, Bochmann G V, Khendek F, Amalou M, Ghedamsi A. Test selection based on finite state models. IEEE Transaction on Software Engineering, 1991, 17(6):591～603

5 Ramligom T, Thulasiraman K, Das A. Context independent unique sequences generation for protocol testing. In: Proceedings of IEEE INFOCOM'96, San Francisco, CA, USA, 1996. 1141～1148

6 Naik K. Efficient computation of unique input/output sequences in finite-state machines. IEEE/ACM Transactions on Networking, 1997,5(4):585～599

7 Huang C M, Chiang M S, Jang M Y. UIO_E: A protocol test sequence generation method using the transition executability analysis(TEA). Computer Communication, 1998, 21(16): 1462～1475

8 Pang Qi-Xiang, Cheng Shi-Duan, Jin Yue-Hui. Equivalent transformation of EFSM and protocol conformance testing. Journal of China Institute of Communications, 1997, 18(4):37～42(in Chinese)
(庞其祥, 程时端, 金跃辉. EFSM 的等价转换和通信协议一致性测试. 通信学报,1997,18(4):37～42)

9 Zhou Xiao-Yu, Qu Yu-Gui, Zhao Bao-Hua. Using invertibility to reduce the length of test sequences in EFSM Model. Journal of China Institute of Communications, 2000,21(11):48～55(in Chinese)
(周晓煜,屈玉贵,赵保华. 利用逆向判定性缩短 EFSM 的测试序列长度. 通信学报,2000,21(11):48～55)

10 Korel B. Automated software test data generation. IEEE Transactions on Software Engineering, 1990,16(8): 870～879

11 Gupta N, Mathur A P, Soffa M L. Automated test data generation using iterative relaxation method. In: Proceedings of Foundations of Software Engineering, Orlando, Florida, USA, 1998. 231～244

12 Wang Zhi-Yan, Liu Chun-Nian. The application of interval computation in software testing. Journal of Software, 1998,9(6):438～443(in Chinese)
(王志言,刘椿年. 区间算术在软件测试中的应用. 软件学报, 1998,9(6):438～443)

13 Xi Hong-Yu, Xu Hong, Gao Zhong-Yi. ADA software test case generation tool. Journal of Software, 1997,8(4):297～302(in Chinese)
(奚红宇,徐　红,高仲仪. Ada 软件测试用例生成工具. 软件学报, 1997,8(4):297～302)

14 Lan Yu-Hua, Mao Fa-Yao, Cao Hua-Gong. Automated test case generation from Z specification. Chinese Journal of Computers, 1999,22(9):963～969(in Chinese)
(兰毓华,毛法尧,曹化工. 基于 Z 规格说明的软件测试用例自动生成. 计算机学报, 1999,22(9):963～969)

ZHANG Yong, born in 1973, Ph. D. candidate. His research interests include automatic software testing and object-oriented software technique.

QIAN Le-Qiu, born in 1942, professor, Ph. D. supervisor. His research interests include software component, software architecture and software testing.

WANG Yuan-Feng, born in 1974, Ph. D. candidate. His research interests include software component technique and component repository.