XYZ/AE 描述程序性质的探讨*)

左春华[1] 张广泉[1,2] 戎 玫[3]
(苏州大学计算机科学与技术学院 苏州 215006)[1]
(重庆师范大学数学与计算机科学学院 重庆 400047)[2] (暨南大学深圳旅游学院 深圳 518053)[3]

摘 要 为保证程序的正确性,程序在投入使用前需要检验其是否满足规定的性质,那么程序的性质需要用一种语言来描述。XYZ/AE是时序逻辑系统XYZ的一个子语言,用此语言描述程序性质有很多优点。XYZ/AE不仅能描述所有的程序性质且简单易懂,另外它能与可执行语言XYZ/EE结合,描绘程序中间程度的抽象性,具有很好的扩展性。文章从上述几方面探讨了XYZ/AE描述程序性质的能力。

关键词 程序性质,XYZ/AE,XYZ/E

Discussion of Describing Program's Property with XYZ/AE

ZUO Chun-Hua[1] ZHANG Guang-Quan[1,2] RONG Mei[2]
(School of Computer Science and Technology, Soochow University, Suzhou 215006)[1]
(School of Mathematics and Computer Science, Chongqing Normal University, Chongqing 400047)[2]
(Shenzhen Tourism College, Jinan University, Shenzhen 518053)[3]

Abstract In order to guarantee the procedure accurate, it is necessary to examine whether it satisfies program's property before used. XYZ/AE is one sub-language of the temporal logical system XYZ. Describing program's property with XYZ/AE has superiority comparable to other temporal logical languages. XYZ/AE can express all program's properties and it is easy to understand. Moreover it can combine with executable language XYZ/EE to describe middle abstract program. It has very good extension. The paper discusses the description ability of XYZ/AE from these aspects.

Keywords Program's property, XYZ/AE, XYZ/EE

1 引言

XYZ系统是中科院软件研究所唐稚松院士提出的一阶线性多类型逻辑系统,其特征是能在统一的时序逻辑框架下既表示程序的静态语义又表示程序的动态语义,时序逻辑语言XYZ/E是这个系统的核心语言。XYZ/E又可分为两类子语言:XYZ/EE和XYZ/AE,XYZ/EE称为可执行子语言,是一种常见的命令式过程性算法语言,可以在计算机上有效执行;XYZ/AE称为抽象描述子语言,用来表示程序应该满足的性质。

目前开发软件系统,花在开发过程中的时间比用在修改维护上的时间少得多,于是如何保证软件的正确性受到了很大的关注。保证软件正确常用的方法就是测试,可是测试只能证明程序有错误,不能证明程序是正确的。对于一些对安全性要求很高的系统,用测试来保证程序的正确性显然不能满足要求。形式化的验证方法能保证程序的正确性,顺序程序一般采用HOARE逻辑进行证明,而并发程序常采用模型检测的方法,可是不管哪种方法都需要描述程序的性质,然后再运用各种方法证明程序代码满足性质。程序的性质一般由程序员自己输入,为了使得形式化的验证方法得到推广,必须选择一个简单易懂,表达能力强的语言来描述程序的性质。XYZ/AE是XYZ系统中用来描述程序性质的,其形式灵活,简单易懂,可扩充性很大,能够描述程序的所有性质,适合用于程序的验证过程中。

文[1]给出一实例,用XYZ/AE描述程序性质,XYZ/EE描述可执行代码,再证明可执行代码满足程序性质。为了推广XYZ/AE作为描述语言的使用范围,我们深入研究了该语言的表达能力,总结了一些优点。

2 XYZ/AE简介[2]

XYZ/AE是XYZ/E的子语言,本文先简单介绍XYZ/AE的基本元素,再给出该语言的基本结构。

2.1 时序逻辑算子

XYZ/E是一个一阶线性多类型的时序逻辑语言,其包含的部分时序逻辑算子介绍如下。

(1)一阶逻辑中常见的逻辑连接词,如:否定词表示成"~",合取词表示成"∧",析取词表示成"∨",不可兼析取词表示成"$∨",蕴涵词表示成"→",等值词表示成"==",全称量词表示成"$A",存在量词表示成"$E"。

(2)将来时序算子,如:下一时刻算子表示成"$O";必然算子"[]",表示从所指时刻起以后所有的时刻;终于算子"<>",表示从所指时刻起某一时刻;直到算子"$U",表示左式真直

*)基金项目:江苏省高校自然科学基金(批准号:05KJB520119)、重庆市自然科学基金项目(编号:CSTC,2006BB2259)、重庆市教委科学技术研究项目(合同号:040803)。**左春华** 硕士研究生,研究方向为程序验证与编译器的结合;**张广泉** 教授,博士,CCF高级会员,主要研究方向为软件工程与形式化方法;**戎 玫** 博士,副教授,主要研究方向为软件工程与电子商务。

到右式真；除非算子“$W”，表示左式真除非右式真。

用 XYZ/AE 描写程序性质时，用得较多的是将来时序算子，与之对应的过去时序算子在这里不作介绍。

2.2 状态转换等式

状态转换等式如式(2.1)所示，用于将状态转换机制表示在直言式逻辑中，此处 $O 为下一时刻算子，v 表示一个时序变量，e 表示一个类型与 v 相同的表达式，这就是常见高级语言的赋值语句，不同的是(2.1)式表示的是变量 v 下一时刻的值为 e。当变量 v 恒为一特殊的系统变量“LB”时，它恒取执行标号为值，其形式如式(2.2)，表示“下一时刻程序执行标号为 y 的语句”，即常见高级语言的跳转语句。(2.3)式表示“当前执行标号 y”。

$$\$Ov=e \tag{2.1}$$

$$\$OLB=y \tag{2.2}$$

$$\$LB=y \tag{2.3}$$

2.3 条件元

利用上面的表示状态转换机制的等式，可表示出 XYZ/E 中一种基本的命令形式如：式(2.4)及(2.5)，我们称之为“条件原子式”，或简称“条件元”。其中，“@”表示“$O”(下一时刻)或是“<>”(终于)；“R”，“Q”为一阶逻辑公式，分别称为条件元的“条件部分”与“动作部分”。

$$LB=y\wedge R\rightarrow @(Q\wedge LB=z) \tag{2.4}$$

$$LB=y\wedge R\rightarrow \$O(v1,v2\cdots vk)=(e1,e2\cdots ek)\wedge \$OLB=z \tag{2.5}$$

2.4 XYZ/AE 描述语言的形式

XYZ/AE 是 XYZ/E 的抽象子语言，其特征是程序中出现式(2.4)形式的条件元。此外，每一单元可以带有约束部分。用此语言表示程序的性质时，不但可以用一阶逻辑公式表示其前置条件(2.4)式中 R)及后续条件(2.4)式中 Q)，而且可以将其用在约束部分所定义的特殊谓词里。XYZ/AE 由如下语句构成。

[][LB=START∧R→@(Q∧LB=STOP)]

WHERE []SPECIFICATION

此处约束部分 SPECIFICATION 可以单独定义，由(2.4)式或是逻辑式组成。可执行语言 XYZ/EE 就是由条件元(2.5)式组成。

3 用 XYZ/AE 描述程序性质

程序性质一般指的是安全性和活性。根据一种非形式的解释，所谓安全性是指“坏情况不会发生”，所谓活性是指“好情况最终将发生”[1]。安全性强调执行过程中不会有不安全的情况发生，而活性则是强调正确的结果最终会产生。并发程序执行顺序不确定，情况很复杂，需要同时关注执行过程中的性质及执行结果，所以对安全性及活性都有严格要求，因此以之为例来研究 XYZ/AE 的描述能力比较有代表性。安全性包括部分正确性、无死锁性、互斥性[4]；活性包括终止性、完全正确性、无活锁性、响应性[3~5]。由于完全正确性包含终止性，因此在讨论活性时不单独讨论终止性。本节先从安全性和活性方面研究 XYZ/AE 的描述能力，接着从状态的转换方面来研究。

3.1 安全性的描述

下面首先讨论安全性，分别研究部分正确性、无死锁性、互斥性等性质的描述。

(1) 部分正确性

部分正确性指的是若程序的前置断言正确，如果程序终止，一定会有后置断言的正确。实例如下：

(3.1.1)
```
#pragma begin safety_property sp1
    int s=1;
void process1(){
  P(s);     lb1: //关键代码
  V(s);}
void process2(){
  P(s);     lb2: //关键代码
  V(s);}
#pragma end safety_property sp1
```

程序表示的是两个进程争用一临界资源的情况。s 是临界资源变量，其初值为 1，表示临界资源最多只能被一个进程占用。某一时刻，process1 执行代码 p(s)，如果此时 s>0，表示临界资源未被占用，则 process1 抢占临界资源成功。p(s)是加锁操作，执行结果是 s 值减 1。由 s 初始值为 1。可知，此时 s=0。process1 占用临界资源后开始执行关键代码 lb1。使用完后释放临界资源，执行 v(s)解锁操作，执行结果是 s 的值加 1，整个过程执行结束后，s 的值还是 1。如果 process1 执行 p(s)操作时，s<=0 表示已有进程占用临界资源，则 process1 只能等待其他进程释放资源。process2 的执行过程与 process1 相似。

部分正确性用 XYZ/AE 描述为：

sp1=[][LB=START∧s==1∧→[](LB=STOP→s==1)]

上式表示程序开始时 s=1，恒有如果程序结束可以推出变量 s 还是为 1。

(2)无死锁性

无死锁性即不会发生各进程相互死等的状态。实例如下：

(3.1.2)
```
#pragma begin safety_property sp1
      int s=1,t=1;
process1(){
   lb1:p(s);     lb2: //代码
   p(t);         lb3; //代码
   v(t);         v(s);}
process2(){
   lb4:p(t);     lb5: //代码
   p(s);         lb6; //代码
   v(s);         v(t);}
#pragma end safety_property sp1
```

程序描述的是两进程争用临界资源的情况，这里的两进程需要同时占用两个临界资源才能完成相应任务。s 是临界资源 A 的变量，初值为 1，表示临界资源 A 最多能被一个进程占用。t 是临界资源 B 的变量，t 初值为 1，表示资源 B 最多能被一个进程占用。进程 process1 首先需要使用临界资源 A，再占用资源 B，才能完成整个任务。而 process2 首先需要使用临界资源 B，再占用资源 A 才能完成任务。由于并发程序顺序的不确定性，很容易造成进程 process1 已经占有临界资源 A，正等待资源 B，而 process2 已占用临界资源 B，正等待资源 A 的情况。此时，进程 process1 不能完成任务，也不能释放资源 A，只能等待资源 B，而 process2 刚好相反。这样，两进程相互死等，就会造成死锁。

对于程序无死锁性的性质可表示为：

sp1=[][LB=START∧s==1∧t==1→<>(LB=

STOP∧s==1∧t==1)]

WHERE[[](~(LB1=lb1∧LB2=lb4)∧~(LB1=lb1∧LB2=lb5)∧~(LB1=lb2∧LB2=lb4))]

上式表示的含义为程序执行开始时变量 s=1、t=1，到某一时刻结束时有变量 s=1，t=1，只不过运行时两进程需满足一定的顺序。当 process1 执行代码 lb1 时，process2 不能执行代码 lb4 和 lb5；当 process1 执行代码 lb2 时，process2 不能执行代码 lb4。这样，就能避开死锁，从而描述了程序的无死锁性。

(3)互斥性

互斥性表示两个不同的进程不会在同一时间进入临界区。

以程序(3.1.1)为例，该程序互斥性可表示如下：

sp1=[][LB=START∧s==1→<>(LB=STOP∧s==1)]

WHERE[][~(LB1 = lb1∧ LB2=lb2)]

上式表示程序执行开始时变量 s=1，结束时变量 s 还是为 1，执行过程中不可能出现进程 process1 执行 lb1 而 process2 执行 lb2 的时刻。

3.2 活性的描述

程序的活性包括完全正确性、无活锁性和响应性。

(1)完全正确性

完全正确性即若程序的前置断言正确，则程序开始执行后一定能终止，并有后置断言正确。

对于程序(3.1.1)的完全正确性可描述为：

sp1=[][LB=START∧s==1→<>(LB=STOP∧s==1)]

上式表示程序执行开始时变量 s=1，且程序一定会终止，终止时变量 s 还是为 1。

(2)无活锁性

活锁就是指某个事务永远处于等待状态，得不到执行的现象。

实例如下：

(3.2.1) #pragma begin active_property sp1

```
        int t=1;
process1(){
    p(t);     lb1;//代码
    v(t);}
process2(){
    p(t);     lb2;//代码
    v(t);}
process3(){
    p(t)      lb3; //代码
    v(t);}·····
#pragma end active_property sp1
```

程序表示的是多个进程争用一临界资源的情况。某一时刻临界资源最多只能被一个进程占有。如果进程很多，使得某一个进程一直处于等待状态，就会造成活锁。

无活锁性可描述为：

sp1=[][LB=START∧t==1→<>(t==1∧LB=STOP)]

WHERE[][$E LB1=lb1∧$E LB2=lb2∧$E LB3=lb3∧·····]

上式表示程序开始时变量 t=1，某一时刻结束时变量 t 为 1，执行过程中进程 process1 能执行到代码 lb1，process2 能执行到代码 lb2，process3 能执行到代码 lb3，……。

(3)响应性

对于永不终止的连续运行程序(如操作系统)，终止性和完全正确性是没有意义的。通常期望这类程序具有的性质是响应性，即一个服务请求最终得到响应。

以程序(3.2.1)为例，响应性可描述为：

sp1=[][$ELB1=lb1∧$ELB2=lb2∧$ELB3=lb3∧·····]

上式表示意义为进程 process1 能执行代码 lb1，进程 process2 能执行代码 lb2，process3 能执行代码 lb3，等。

以上是从安全性和活性等方面来研究 XYZ/AE 语言的描述能力。从上可以看到，XYZ/AE 描述性质的能力强大，不论是安全性还是活性，XYZ/AE 都能准确地描述程序应该满足的性质，且表达形式简单易懂，均是通过一个条件元和 WHERE 引导的约束语句来表达。对于较复杂的性质，可以通过 WHERE 语句嵌套表达。在实际使用过程中，往往是安全性和活性等多个性质一起描述。

3.3 用 XYZ/AE 描述程序性质的实例

上面讨论的都是 XYZ/AE 描述并发程序的性质，这里给出一个关于状态转换的顺序程序片断，可以用 XYZ/AE 描述程序的转换过程。通过此实例，可以看出 XYZ/AE 描述顺序程序的能力。

```
Do{
 KeAcquireSpinLock();
 NPacketsOld=NPackets;
 If(request){
     Request=request->next;
     KeRealeaseSpinLock();
     NPackets++;}
}while(Npackets! =NPacketsOld);
KeReleaseSpinLock();}
```

此程序片断满足一定规范，用图 1 表示如下。

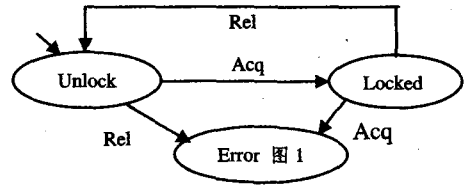

图 1

这是非形式化的描述方法。为了判断程序的正确与否，需要对其进行验证。但是，不论采用哪种验证方法，我们都需要用形式化的方法来描述程序应该满足的性质。用 XYZ/AE 描述如下：

```
%Var[
state:{Unlocked,Locked,Error}
infoType:{Rel,Acq}]
Scene1=(state=Unlocked∧(chin? infoType∧infoType=Acq))→$Ostate=Locked;
Scene2=(state=Locked∧(chin? infoType∧infoType=Rel))→$Ostate=Unlocked;
sp1=[][LB=START∧state==Unlocked→LB=STOP]
WHERE[[]Scene1∧[]Scene2]
```

(下转第 278 页)

即较重要的顾客需求上。这与质量改进和互动机制的出发点和目标是一致的，也符合质量经济性的要求。

开发 QFD 矩阵主要就是为了促使企业同顾客进行接触，了解他们的需求，同时帮助企业在有限的资源约束条件下确定产品的技术需求以使顾客满意度最大。

QFD 矩阵建立起来后，根据顾客要求找出工程特征、确定其指标值并满足之，这样才能引起顾客对产品质量的满意和信任。实现这些指标值的难度是不同的，它与资源约束、技术力量、目标(指工程指标)值的精度等有关。作为企业管理者而言，实现顾客需求的目标为绝对目标，在此基础上，企业要实现效益最佳，就是要对产品成本、交货时间及有限资源进行综合平衡。利用数学语言表示，就是对下列模型进行求解：

$$Q_p(c,t,e)\geqslant Q_r \tag{1}$$

$$S.T.\ \min T=\min \Sigma a_i t_i \tag{2}$$

$$\min C=\min \Sigma b_i c_i \tag{3}$$

$$f(t_1,t_2,\cdots,t_m)\leqslant T_0 \tag{4}$$

$$g(c_1,c_2,\cdots,c_m)\leqslant C_0 \tag{5}$$

$$h((c_1,t_1,e_1),\cdots,(c_m,t_m,e_m)) \tag{6}$$

其中式(1)为目标函数，表示产品质量 Q_p(是一个与成本 C、交货时间 t、工程指标 e 有关的函数)应至少不低于顾客期望的质量目标 Q_r；式(2)表示使交货时间最短；式(3)表示使目标成本最少；式(4)和式(5)表示可利用的资源(时间、资金)有限；式(6)反映工程指标与成本、时间之间的相关关系，如：指标值越好，所花成本或时间可能越多。系数 a_i，b_i 与实现指标的难度系数有关，m 为工程指标个数，T_0 为顾客需要的交货时间，C_0 为产品目标成本。

在选择应优先配置的技术需求时，应综合考虑技术需求的重要度、顾客竞争性评估、技术竞争性评估、技术实施难度和成本、技术需求相关矩阵等各种因素。其中技术需求重要度是一个关键因素，可通过顾客需求重要度(权重)进行矩阵运算得到。设 W_i 为第 i 个顾客需求的权重，F_{ij} 为第 i 个顾客需求和第 j 个技术需求之间的相关系数，W_{Tj} 为第 j 个技术需求的重要度，则：

$$W_{Tj}=\Sigma w_i \cdot F_{ij}\ \ j=1,2,\cdots,m \tag{7}$$

其中 n 为顾客需求总数，m 为技术需求总数。

其他因素也很重要。如果顾客竞争性评估和技术竞争性评估表明某产品的某项技术需求在现有生产条件下已能使顾客满意，可考虑在质量改进中和互动机制中把注意力更多地放在其他顾客不太满意的技术需求上，以提高质量改进和互动机制的有效性和效率。

4. 其他信息对企业实施互动机制的指导作用

互动质量屋中的顾客对需求的满意度矩阵 P 可形象地揭示顾客现在对何种需求满意度最高，何种需求仍需提高顾客满意度，可使企业有目的地通过互动引导和影响顾客注重于满意度高的需求上，并从对同类竞争产品的满意度比较中识别质量改进的机会。如顾客对现有市场上所有同类产品的某项需求都不太满意，表明在此存在一个质量改进的机会，并在质量改进后通过互动将其作为营销重点，使本企业的产品区别于其他竞争对手的同类产品。而通过质量改进和实施互动机制前后 P 矩阵的比较，也可清晰地看出质量改进与互动机制实施对顾客满意度的影响，以作为评价互动机制有效性和效率的最好依据。

互动质量屋中的顾客对技术需求(工程特征)的满意度矩阵 E，可令企业通过改进前后的 E 矩阵比较，直观地看出改进某项技术需求后对顾客满意度的影响，并从与其他竞争产品同一项技术引起的顾客满意度比较识别改进机会，制定明确的改进目标。如矩阵 E 和转换质量屋中的技术竞争性评估矩阵显示对某产品的某一项技术需求，该公司领先于竞争者，且顾客对其也很满意，则改进该项技术需求不会获得明显的经济效益；如显示顾客对所有现有产品某项技术需求均不满意，则存在改进机会并在互动机制中将其作为互动的重点；如显示顾客对其他竞争产品的该项技术需求均不太满意，而认为本公司的最差，则应马上改进，并与顾客互动了解顾客的需求，确定高于现有产品质量水平的改进目标值。

由此可见，互动质量屋(I-HOQ)对于企业建立和实施质量改进与顾客互动机制具有重要的指导作用，是企业实现互动机制、进行正确决策的有效工具。

参 考 文 献

1 吴伟强．面同制造企业的质量改进与顾客互动机制[C]：[浙江大学硕士学位论文]．2001

2 滕晓林，赵新力，任守榘．一种改进的质量功能部署模型[J]．系统工程理论与实践，1995(3)：14～19

3 Day R G. Quality Function Deployment[M]. ASQC Press, Milwaukee, Wisconsin, 1993

4 Cohen L. Quality Function Deployment: How to work for you [M]. Addison Wesley Publishing Company, Massachusetts, 1995

5 Kogure M, Akao Y. Quality Function Deployment and CWQC in Japan[M]. Quality Progress, Oct. 1983

(上接第 270 页)

通过验证可发现，该程序片断不符合此性质。如果 request 为真时，进入条件语句执行范围，执行完后 state＝Unlocked。当整个 while 语句执行结果，再一次执行释放资源，即 infoType＝Rel。显然，程序性质中不存在这条路径，进入错误状态。

结论 XYZ/AE 语言以时序逻辑为基础，但它不仅仅只具有时序逻辑语言的特点，还有另外一些优点：第一，能表示程序的所有性质且形式简单易懂；第二，XYZ 系统是一个统一的时序逻辑框架，既能描述可执行的语句，也能描述抽象的性质，且 XYZ/AE 能与可执行语言 XYZ/EE 结合，用来表示中间程度的抽象性[2]，能够为以后的验证过程提供方便。

为了使得形式化的验证方法得到推广，在对描述程序性质研究的时候，还应该注意被描述语言以及所描述程序的特点。总结这些特点，可以得到比较通用的描述程序性质的语言。

参 考 文 献

1 官荷卿，郭亮．程序性质的描述与证明，计算机科学，2003，30(3)：146～148

2 唐稚松，等．时序逻辑程序设计与软件工程．北京：科学出版社，2002

3 丁志军，蒋昌俊．并发程序验证的时序 PETRI 网方法．计算机学报，2002，25(5)：467～475

4 蒋屹新，等．基于线性时态逻辑的 PETRI 网模型检测．系统仿真学报，2003，15(8)：6～10

5 张广泉，戎玫．时态逻辑与并发程序．重庆大学学报，1999，22(1)：47～50

6 Hoare T. The Verification Grand Challenge. http://www.csl.sri.com/users/shankar/VGC05/

7 Pnueli A. Looking Ahead. http://www.csl.sri.com/users/shankar/VGC05/