形式化 B 描述测试序列自动生成研究①

丁岳伟，彭金梅

(上海理工大学 光电信息与计算机工程学院，上海 200093)

摘 要：基于严格数学理论的软件形式化规格说明，经过逐层精化，不仅可以让软件开发过程更加有效精准，而且为测试用例测试序列的自动生成提供了最原始可靠的依据。通过 B 抽象机操作的规范型，依据测试理论，可以将原操作等价于多个效用谓词（effect predication）的形式。按照路径覆盖得到状态转换图，并对状态图做了确定性处理；运用基于状态图的测试准则，生成有效的测试序列，提高测试的有效性和效率。

关键词：形式化描述；测试序列生成；状态转换图；B 方法；效用谓词

Automating the Generation of Test Sequences from B Formal Specification

DING Yue-Wei, PENG Jin-Mei

(School of Optical-electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

Abstract: Formal specification, based on critical mathematics, not only make the process of software development more effective and precise, but also contain a great deal of information that can be as the original reliable basis for the generation of test sequences. The work reported here base on the standard B notation, splitting the operation to the equal effect predications and then generate state transition diagram according to path coverage criterion, and also solving the indeterminism of state transition. Generating the test sequences according to the test criterion on the state diagram, which can prove an effective test, has also been presented.

Key words: formal specification; test sequences generation; state transition diagram; B method; effect predication

一直以来，软件测试在软件生产过程中占有很大比重，而测试用例是软件测试的关键与难点，其质量直接决定软件测试的科学性和有效性。传统的用例设计方法是由测试人员手工生成，其质量直接依赖于软件测试人员的经验和专业水平，具有一定的局限性。为此，人们研究在基于形式化规格的基础上，利用形式规格语义的一致性和无二义性，自动生成测试用例，从而可以节约人力、提高测试质量和效率。

1 形式化规格生成测试用例

1.1 形式化方法

形式化方法始于 20 世纪 60 年代，它是从精确的数学语义出发来开发计算机软件的严格方法,被誉为是克服软件危机、提高软件可靠性的革命性途径。形式化方法以一种精确并严格的方法描述软件系统的定义与需求。它建立在严格的数学基础上，通过集合、谓词逻辑、函数映射等数学方法, 使软件系统从规格说明到代码生成每阶段的形式变换都有严格的数学推演和正确性证明, 从而保证其语义不变。

理论上通过形式化方法开发出来的软件是可以保证 100%的正确性，但是由于形式规格与实际需求还可能会存在不一致性，加上形式化方法的复杂性以及实际开发中不同的编译器、操作系统以及运行环境等因素，测试在形式化开发中仍然不可或缺。

1.2 形式化规格生成测试用例

一般来说，基于形式化规格生成测试用例的方法一般遵循如下步骤：

① 首先是通过形式化语言对需求进行描述，得到

① 收稿时间:2011-08-25;收到修改稿时间:2011-10-03

形式化模型。

② 通过对形式化模型进行词法分析，语法分析，提取相关的测试上下文，包括输入变量、类型检查、条件分析、操作代换的提取等。

③ 按照一定的约定形式将测试上下文抽取成相应的规格，得到原始的测试用例。

④ 按照相应的覆盖准则，简化用例集，得到最终用例。

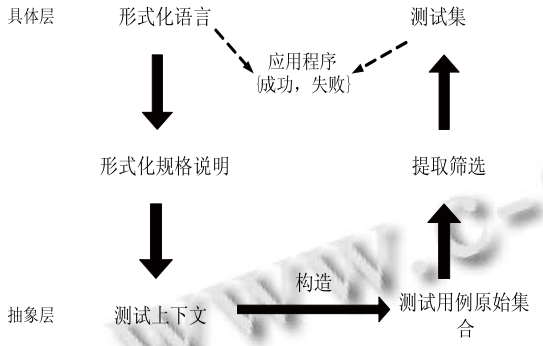

图 1　形式化规格生成测试用例基本步骤

假如需要将测试过程进一步自动化，可以通过随机数生成器或者约束求解器，根据测试用例生成满足条件的测试数据。结合测试脚本，将测试用例映射到测试脚本中，即可将测试过程进一步自动化。

CASTING[3]支持任何形式的输入，将输入按照预定规则转化为系统支持的语义，通过分裂树的形式将语义划分为确定的测试用例，包括条件、输入、输出，再根据测试用例构造具体的 FSA，其缺点是不支持非确定性的规格说明。BZ-TT 框架兼容 B 和 Z 两种规格的输入，结合边界值和因果测试策略，将系统状态的最大最小边界值定义为边界目标，利用约束逻辑求解技术，生成满足边界目标的测试用例[1]。而基于 VDM—SL 的方法是利用一系列的重写规则对形式化描述进行等价变换，把形式化描述中的前置条件转换成析取范式。该方法的缺点是难以保证输入空间和有效输入空间相同[2]。本文提出一种基于 B 规格的测试序列自动生成的方法。通过将 B 规格的操作转化为一种规范型，提取其前-后谓词，将原操作划分为多个确定的效用谓词。根据这些效用谓词建立态转换图，并在得到分枝树的过程中以具体状态代替抽象状态，在一定程度上解决非确定性问题。

2　从B描述生成测试用例

2.1 B 方法

B 方法是一种覆盖了从规格说明到编码的所有阶段的开发方法。建立在 Zermelo-Franke 集合论的基础上。B 方法以“抽象机器”的结构化机制描述需求，以广义代换来描述系统的动态特征，使得不变式的证明义务维持在集合论的范畴之内。

根据 B 的抽象机理论，任何一个广义代换 S 都可以化成一种规范型：S=P | @ x′ . (Q==>x:= x′)，其中 P 和 Q 是谓词，Q 依赖于 x，x′, x′ 是一个与 x 不同的变量，在 P 中不自由出现。它说明了该操作所造成的状态改变必须满足的所有特定的相关特性。上述规范型的语义是：在满足谓词 P 的前条件下，选择任意满足后条件谓词 Q 的 x′，使得原变量 x 可以用 x′ 替代。这一规范型为操作的前-后谓词的表示形式提供了依据。详见参考文献 6 第六章。

Prdx(S)是 B 前-后谓词的语法定义，表示只要 S 不会导致代换之后变量 x 和变量 x′ 不同，那么后值 x′ 就将与前值 x 一样。表示一个操作规范型的前-后谓词。对于 B 抽象机中的每一个操作，沿着操作的分支语句(包括 if..else.., choice 语句等)，提取出相应的条件语句，使得每一个判定真假值各取一次。以合取形式表示，最终可以将此操作分为多个包含前-后谓词的效用谓词，如(1)所示。效用谓词之间以?符号分隔，这有别于 DNF 方法中的析取符号，能更好地处理抽象机中的∨运算。

$$EP1 \square EP2 \cdots \square EPi \quad (1)$$

$$EP\ i = inv \wedge pre \wedge condition_i \wedge Prd_x(S)$$

其中，$1 \leqslant i \leqslant N$, inv, pre, $condition_i$ 分别表示效用谓词 EPi 的不变式，前条件，以及所有变量的取值。由于抽象机中的不变式是对所有操作的约束，在此引入是以保证操作的正确性和完整性。若每个 EPi 中的 conditioni 表示的子域不相交，则由 EPi 表示的等价类的集合就构成了该操作谓词的定义域。

本文中的方法需要基于以下假设进行：

① 每个操作都有明确的前条件；

② 只支持有限的数据类型，即每一种数据类型都可以用穷举的方式给出。

2.2 效用谓词的构造示例

下面通过例子来描述效用谓词的划分。

MACHINE

EXAMPLE
VARIABLES
variable1
INVARIANT
Inv
OPERATIONS
OP = pre
THEN
IF T1 THEN
Sub1
ELSE
Sub2
END
END

上述 B 方法的抽象机刻画，将其划分为效用谓词得到该操作的测试用例为：

inv ∧ pre ∧ T1 ∧ Prd(Sub1) □ inv ∧ pre ∧ !T1 ∧ Prd(Sub2)

3 测试序列的自动生成

3.1 构造状态转换图

我们已经为某个独立的操作生成了确定的测试用例。但是，在某些基于状态的系统描述上，由于系统之间状态的相互关系，仅仅是单个操作的测试用例并不能直接激活。此时，我们可以基于效用谓词生成测试系列，以测试系统的连续行为。基本思路是：

① 将所有的效用谓词表示为集合 E，其中包括初始状态；

② 以初始状态为起点，激活所有满足了其前条件的效用谓词，到达下一个状态。若 E 中所有的效用谓词均被覆盖，转 4, 否则转 3；

③ 以当前状态为起始状态，转 2；

④ 根据测试准则得到测试序列。如果状态转化图中存在行为的不确定性，则在生成测试序列的过程将其确定化并简化；

下面通过一个计算机进程调度的例子来说明。

MACHINE
SCHEDULER
SETS
PID = {p1, p2, p3}
VARIABLES
active, ready, waiting
INVARIANT
active⊆ PID∧ready ⊆ PID∧waiting ⊆ PID∧ready∩waiting = ∅∧ready∩active=∅∧waiting∩active=∅∧active∩(ready∪waiting)=∅∧card(active)≤1∧(active =∅) ⇒ (ready=∅)
INITIALIZATION
active := ∅
ready := ∅
waiting := ∅
OPERATIONS
NEW(pp)
PRE
pp∈PID∧pp ∉ (active∪ready∪waiting)
THEN
waiting := (waiting∪{pp})
END;
READY(rr)
PRE
rr ∈ waiting
THEN
waiting := (waiting − {rr})
IF (active = ∅) THEN
active := {rr} ELSE
ready := ready ∪ {rr} END
END;
SWAP
PRE
active = ∅
THEN
waiting := waiting ∪ active
IF (ready = ∅) THEN active := ∅
ELSE
ANY pp WHERE pp ∈ ready
THEN
active := {pp}
ready := ready − {pp}
END
END
END;

计算机进程共有三个状态:等待，就绪，运行，分

别通过操作 new，ready，swap 进行进程的状态转换。按照第 2 节介绍的效用谓词构造方法，得到以下的 5 组效用谓词：

new(pp): inv ∧ pp ∈ PID ∧ pp∉(active ∪ ready ∪ waiting) ∧ active′=active ∧ ready′=ready ∧ waiting =waiting∪{pp}

ready1(qq): inv ∧ qq ∈ waiting ∧ active=∅ ∧ active={qq}∧ready= ∅∧waiting=waiting-{qq}

ready2(qq): inv ∧ qq ∈ waiting ∧ active≠∅ ∧ active′ =active∨ready=ready∪{qq}∧waiting=waiting-{qq}

swap1: inv ∧ active≠∅ ∧ ready=∅ ∧ active′=∅ ∧ ready=∅∧waiting=waiting∪active

swap2: inv ∧ active≠∅ ∧ ready≠∅ ∧ (∃pp ∈ ready| waiting=waiting ∪ active ∧ ready′=ready-{pp} ∧ active′={pp})

以每个变量取值为空或非空为依据，则系统总共有以下 6 个状态：

表 1　进程的状态

1.active= ∅, ready= ∅, waiting=∅	2.active= ∅, ready= ∅, waiting ≠∅	3.active ≠∅, ready= ∅, waiting= ∅
4.active ≠∅, ready= ∅, waiting ≠∅	5.active ≠∅, ready ≠∅, waiting=∅	6.active ≠∅, ready ≠∅, waiting ≠∅

根据 3.1 描述的算法，得到的状态转换图如下：

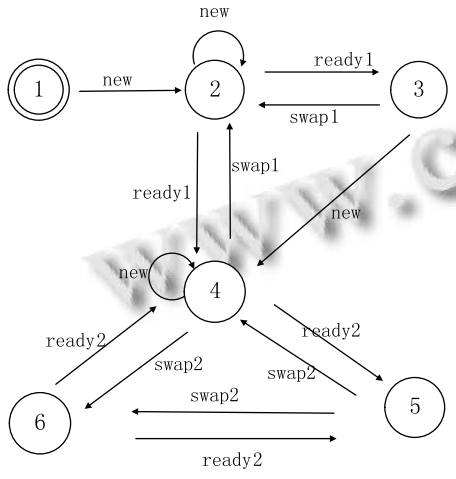

图 2　进程状态转换图

其中状态 1 是起始状态。由图 2 可以看出，状态图中存在非确定性。比如从状态 2，经过 ready1 操作，分别到达了状态 3 和状态 4，对于状态 4 的 ready2 操作以及状态 5 的 swap2 操作也是如此，所以需要对这些状态之间的转换进行确定化。

3.2 测试序列生成

定义 1.(抽象状态/具体状态)：假设状态 S 由变量集合{vi}表示，其中 i=1,2,..n。若∃v∈{vi}且 v 已经被赋予具体值或明确的逻辑运算关系，则称状态 S 为具体状态，反之，则称 S 为抽象状态。

定义 2.（非确定状态转换）：S 是状态集合，Σ为有限操作集，它有效用谓词构成，δ：S×Σ→S 确定下一个状态。给定 s, s′, s″∈S, a∈Σ且 s′ !=s″，若∃δ(s, a) = s′ 且δ(s, a) = s″，则称状态 s 在操作 a 上的转换是非确定的。

对于非确定状态转换，可以通过在构造分枝树的过程中引入具体状态，这在一定程度上可以消除状态转换的非确定性。本例中通过分别引入和状态 2，4，5 相应的具体状态在某程度上取代抽象状态，以达到状态转移确定性的目的，同时抽象状态仍需保留。

表 2　增加的具体状态

2′.active=∅, ready=∅, #waiting=1	2″.active=∅, ready=∅, #waiting>1	4′.active≠∅, ready=∅, #waiting=1

4″.active≠∅,
ready=∅,
#waiting>1
	5′.active≠∅,
	#ready=1,
	waiting=∅
		5″.active≠∅,
		#ready>1,
		waiting=∅

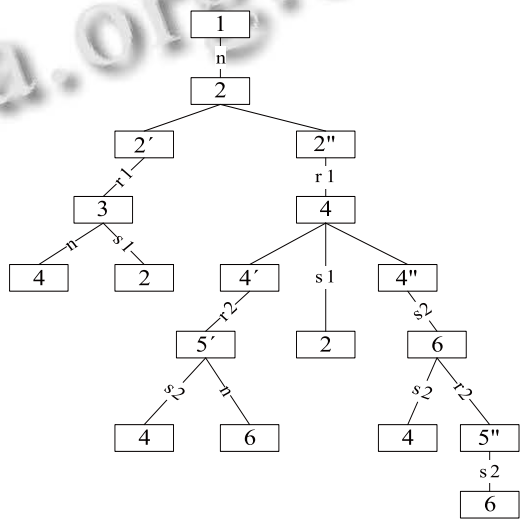

图 3　分枝树

其中#表示取集合的势运算，即集合元素的个数。

基于状态图的测试主要有以下 5 种覆盖准则：(1)状态覆盖(2)转移覆盖(3)转移对覆盖(4)ZOT 循环覆盖准则(5)全 ZOT 路径覆盖准则[4]。限于篇幅，在这里采用转移覆盖准则。依据状态转换图以及引入的具体状态，得到如下的分枝树。

取分枝树的每一条分枝，即是满足转移覆盖准则的测试序列：

1--2′ --3--4

1--2′ --3--2

1--2″ --4′ --5′ --4

1--2″ --4′ --5′ --6

1--2″ --4--2

1--2″ --4″ --6--4

1--2″ --4″ --6--5″ --6

3.3 结果分析

一般运用状态转换图进行问题的分析通常需要考虑状态爆炸的问题。本例在只有 3 个进程的假设前提下对非确定的状态转换，通过引入具体状态而进行确定化，以生成有效的测试序列。事实上，在具体执行的过程中，所有的状态均为具体状态，状态转换图也可以作为测试结果的验证依据。

4 结语

本文对基于形式化 B 规格进行测试用例自动生成的研究，给出了效用谓词的生成过程与其充分性分析，同时对基于集合的状态模型转换为状态转换图，并对非确定的状态图进行确定性转换，按照转移覆盖原则生成测试序列。下一步的研究工作是结合测试脚本，真正实现测试过程自动化。

参考文献

1 Legeard B, Peureux F, Utting M. Automated Boundary Testing from Z and B. Proc. of the International Conference on Formal Methods Europe (FME'02). Copenhagen, Denmark, July 2002. LNCS 2391, Springer-Verlag, 2002: 21–40.

2 Dick J, Faivre A. Automating the generation and sequencing of test cases from model-based specifications. FME'93: Industrial-Strength Formal Methods. LNCS 670, Springer-Verlag, April 1993: 268–284.

3 van Aertryck L, Benveniste M, Le Metayer D. CASTING: a formally based software test generation method. 1st IEEE International Conference on Formal Engineering Methods (ICFEM'97). 1997: 99–112.

4 李鹏,彭祥伟,周喜,等.基于状态图的测试路径自动生成.计算机工程,2011,37(2):25–29.

5 马亮,张刚.测试用例自动生成方法的现状及研究.现代电子技术,2008.

6 Abrial JR.裘宗燕译.B 方法.北京:电子工业出版社,2004.

7 Legeard B, Peureux F. Generation of functional test sequences from B formal specifications-presentation and industrial case-study. 16th IEEE International Conference on ASE2001. 2001: 377–381.

(上接第 76 页)

参考文献

1 Desmedt Y, Frankel Y. Shared Generation of Authenticators and Signatures. In: Feigenbaum J, ed, Advances in Cryptology-Crypto'91 Proc. LNCS 576, Berlin: Springer-Verlag, 1992:457–469.

2 Xie Q. Cryptanalysis and improvement of two threshold signature schemes. Journal of Communications, 2005,26(7): 123–128.

3 徐光宝,姜东焕.抗合谋攻击的门限签名方案分析与改进.计算工程,2010,36(20):155–156,166.

4 高炜,于晓冬.对一个无可信中心的(t,n)门限签名方案的改进,2010,46(1):84–86.

5 Li CM, Hwang T, Lee NY. Remark on the threshold RSA signature scheme. In: Stinson DR, ed, Advances in Cryptology-Crypto'93 Proc. LNCS773, Berlin: Springer-Verlag, 1994: 413-420.

6 王贵林,卿斯汉.几个门限群签名方案的弱点.软件学报,2000,11(10):1326–1332.

7 Waters B. Efficient identity-based encryption without random oracles. Advances in Cryptology-Eurocrypt 2005. LNCS 3494, Berlin: Springer-Verlag, 2005: 114–127.

8 Gennaro R, Jarecki S, Krawczyk H, Rabin T. Secure distributed key generation for discrete-log based cryptosystems. Advances in Cryptology -EUROCRYPT 1999. LNCS 1592, Berlin: Springer-Verlag, 1999, 295–310.

9 辛向军,肖国镇.几种具有附加性质的数字签名体制的研究[博士学位论文].西安:西安电子科技大学,2007.