文章编号:1671-4598(2020)08-0032-04 DOI:10.16526/j.cnki.11—4762/tp.2020.08.007 中图分类号:TP934 文献标识码:A

# 基于状态转移图的箭载软件时序控制测试用例生成方法

汪冬瑾[1,2], 张 舒[1,2]
(1. 宇航智能控制技术国家级重点实验室, 北京 100854;
2. 北京航天自动控制研究所, 北京 100854)

**摘要:** 为提升箭载嵌入式软件测试的效率和质量,针对箭载软件时序控制这一测试需求,文章提出了一种基于状态转移图的测试用例自动生成算法设计方法;具体分析步骤为首先分析时序控制特性,其次选取状态转移图来表征,自定义所需图元属性以及图元关系、触发条件全面性与一致性的约束条件从而保证模型合理性;再次采用基路径覆盖、转移对覆盖、条件元覆盖3种覆盖准则相结合方式实现测试用例的自动生成;最后以某箭上软件全飞行周期时序控制为基础,多次结果表明该方法生成的软件测试用例可完全覆盖时序控制功能和性能需求,验证了方法的有效性和稳定性,可为箭载软件自动化测试工程化实践提供一定的借鉴。

**关键词:** 状态转移图;时序控制;测试用例;覆盖准则

# Method for Generating Timeline Control Test Cases Based on State Transfer Graph

Wang Dongjin[1,2], Zhang Shu[1,2]
(1. State Key Laboratory of Aerospace Intelligent Control Technology, Beijing 100854, China;
2. Beijing Aerospace Automatic Control Institute, Beijing 100854, China)

**Abstract:** In order to improve the efficiency and quality of the embedded software test, this paper proposes a design method for automatic generation of test cases based on state transition graph to test rocket-borne software timing control. The timing control characteristics are analyzed firstly. Then the state transition diagram is selected to represent the required primitive attributes, the primitive relationship, the trigger condition comprehensiveness and consistency constraints are customized to ensure the rationality of the model. Finally, automatic generation of test cases are realized by combining the three coverage criteria of base path coverage, transfer coverage and conditional element coverage. Taking the software's full flight cycle timing control as an example, the results show that the software test cases generated by this method can completely cover the timing control function and performance requirements. The results also verify the validity and stability of the method, which can provide some reference for the engineering practice of the automated test of the rocket.

**Keywords:** state transition diagram; timing control; test case; coverage criteria

## 0 引言

箭载软件是航天系统的运行基础,其失效会直接导致发射的失败进而危及人员的安全,因此软件研制质量极为重要,现常采用人工设计专用测试用例的方式来验证软件功能、性能及极限情况下的故障处理能力,从而保证软件的可靠性[1-2]。然而如今型号种类增加、研制周期缩短、软件规模扩大、结构层次复杂却受限于型号的定制化需求,无法直接沿用已有软件,相应的软件测试工作必不可少。

人工设计用例方式受人为因素影响较大且日益不能满足现有需求,因此针对软件功能中的时序控制功能,本文提出了一种基于状态转移图的测试用例自动生成算法设计方法,以期实现时序控制功能测试用例自动生成,提升测试效率和质量,为软件自动化测试提供一定的借鉴。本文首先分析时序控制特性,其次选取状态转移图来表征,自定义所需图元属性以及图元关系、触发条件全面性与一致性的约束条件从而保证模型合理性;再次采用基路径覆盖、转移对覆盖、条件元覆盖3种覆盖准则相结合方式实现测试用例的自动生成;最后以某箭上软件全飞行周期时序控制为例,验证了上述基于状态转移图的测试用例自动生成方法的有效性和稳定性。

## 1 时序控制特性分析

飞行控制软件是火箭控制系统实现的载体,软件将连续的控制系统分成离散的控制周期,以定时中断的方式近似的实现。时序控制承载软件的动态特征,时序切换对软件功能和运行结果的影响贯穿整个软件生命周期[3]。

飞行控制软件一般会分为几个飞行段,在不同的飞行

**收稿日期:**2019-11-14; **修回日期:**2019-12-22。

**作者简介:**汪冬瑾(1991-),女,安徽滁州人,硕士,工程师,主要从事箭载控制系统设计方向的研究。

段中有不同的时序，软件中实现的特点为[4]：

1）软件中的某一个时序，在软件动态运行时一般不会重复进入两次，已经切换到其它时序，则通常不会再回来；

2）时序切换的条件一般为逻辑组合，且判定条件经常需要多种条件共同满足，如：到达一定时间、某一计算功能得到的结果达到某一范围等；

3）某个时序可能切换到不同的后续时序中（主时序和多个备保时序），其切换路径取决于系统满足不同的逻辑组合；

4）时序切换会伴随特定的软件动态行为和输出；

5）时序切换会伴随软件功能和运行参数的变化。

通常对时序控制功能的考核点包括：

1）用边界值的测试方法考核时序切换判定逻辑组合实现的正确性；

2）当被考查时序可能切换到多个后续时序时，考查每一种逻辑组合分别满足时，时序的切换是否与需求一致；考查多个切换逻辑组合同时满足时，系统能否切换到一个确定后续时序中而不发生冲突；

3）考查时序切换伴随的软件动态行为和输出是否正确；

4）时序切换导致的飞行控制软件功能或计算参数的变化是否满足预期要求，以及随着参数和功能变化进行相应的初始化或过度处理。

## 2　状态转移图设计

状态转移图适合描述动态特征，即运行状态切换、应激反应等内容[5]。为了适用于本文应用环境，对状态图内容进行了一些扩展和自定义：在状态转移图中融入一些必要的测试信息；借鉴活动图的表达特点，使状态图能够表达具有原子性和瞬时性的软件功能；借鉴正则语言形式化特点，为状态图设置严格的语法约束，使状态图的语义满足全面性和一致性要求，以避免在测试需求分析过程引入错误。

### 2.1　图元属性定义

状态转移图的图元主要包括状态起点、状态节点、状态终点和状态转移连线，如图 1 所示[6-7]。状态起点为软件运行至该状态图上的初始状态节点；状态节点宏观上用于表示软件运行过程中的阶段、时序，微观上表示软件运行过程中的某个操作进程或处理流程，设置其属性包括名称、关联状态、动作列表、内嵌模型列表、描述等 5 个方面，如表 1 所示。

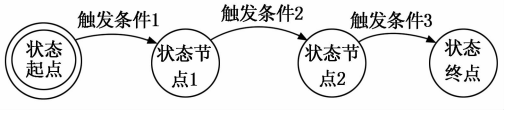


图 1　状态转移图示意图

表 1　WUL 结果及解算时间统计

| 属性 | 描述 |
|---|---|
| 名称 | 显示关联的状态名称(未关联状态则必须手动填写) |
| 关联状态 | 与系统选择状态关联,使该状态节点可以在其他位置被调用 |
| 动作列表 | 系统进入该状态节点时刻完成瞬时性的动作列表 |
| 内嵌模型列表 | 状态节点可嵌入子数据流图/子状态图 |
| 描述 | 由测试人员添加自然语言描述 |

状态终点为状态转移的最后状态节点，理论上，一个状态图可以没有状态终点（软件可以很长甚至无限的生命周期），也可以有多个状态终点（一个终点对应软件运行的某种结果）。然而软件运行过程中的状态图支持在自身的状态节点中嵌套另一个状态图。如果子状态图没有状态终点，则导致软件可能永久停滞在上层状态图的父状态节点中，产生语义错误；并且航天工程软件受硬件资源影响，运行寿命受限，因此设置状态图中必须至少有一个状态终点，且软件运行最终必会到达一个状态终点。

状态转移连线表示软件运行阶段的切换行为，设置其属性包括名称和触发条件，其中触发条件定义为条件元或逻辑表达式，为状态转移发生的前提条件。考虑到被测软件状态节点可能存在瞬时特性，因此定义触发条件可缺省，即当软件运行至该状态节点时，执行进入动作后直接进入连线所指的下一状态节点。

### 2.2　语法约束

为了保证测试模型语义合理性[5,7]，本文还对图元关系、触发条件的全面性与一致性进行了约束。

#### 2.2.1　基本图元关系约束

状态转移图图元间应满足以下几项：

1）状态转移节点名称唯一不可重复；

2）一个状态必须且只能关联一个状态转移节点；

3）一幅状态转移图有且仅有一个状态起点，至少有一个状态终点；

4）状态起点只能与状态转移连线首端相连，状态终点只能与状态转移连线末端相连，状态起点和状态终点不能直接通过状态转移连线相连；

5）状态转移连线首尾必须连接其它图元，可连接同一个状态节点。当指向同一个状态节点的状态转移连线上的出发条件满足时，系统重新执行一次该节点的进入动作。

#### 2.2.2　触发条件全面性与一致性约束

为便于表述，定义触发条件全面性与一致性约束前，首先定义一些数学符号：

1）符号“is _ True（）”表示返回逻辑表达式的真假值，即当前情况下逻辑表达式所述内容是否发生，例如对任意逻辑表达式（Cond），其真假值表示为：is _ True（Cond），且 is _ True（$\phi$）＝＝False。

2）对任意的一个没有定义转移触发条件的状态转移连

线，模型系统自动认定它含有一个触发条件表达式（Cond _ Def），且 is _ Ture（Cond _ Def）＝＝True。

以图 2 为例，示意了触发条件全面性与一致性约束说明。

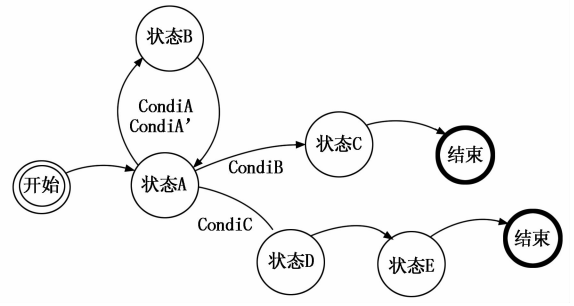


图 2 触发条件全面性与一致性约束示意

a）触发条件全面性约束规定了软件动态运行时不能出现永远停滞在某一个状态节点的情况，并考虑到状态转移连线首尾与一个状态节点连接的情况。即对任意的状态节点（Node），如果从该节点出发有 $n$ 条指向别处的状态转移连线，它们的转移触发条件分别为（Cond1，Cond2，…，CondN）。则要求满足：is _ True（$Cond1 \vee Cond2 \vee \ldots \vee CondN$）！＝ False。

b）触发条件一致性约束规定对任意的状态节点（Node），如果从该节点出发有 $n$ 条状态转移连线，对其中任意的一组触发条件（Condi 和 Condj（$i \neq j$，如果有）），需满足：is _ True（$Condi \wedge Condj$）＝＝False。以图 2 中状态节点 $A$ 为例，如果 $CondiB \wedge CondiC$ 的值不为 False，当 $CondiB \wedge CondiC$ 条件满足时，模型无法确定软件将进入状态 C 或者状态 D，出现二义性。

## 3 测试用例自动生成算法

在上述第 2 章节状态转移图的图元和语法约束设计基础上，采用基路径覆盖、转移对覆盖、条件元覆盖 3 种覆盖准则结合来考核软件时序控制功能，从而实现软件测试用例自动生成[8]，流程图如图 3 所示。首先使用基路径覆盖准则生成基本的测试用例；其次对状态图中的入度大于 1 的状态节点用转移对覆盖准则进行用例的补充和扩展；最后对状态图中由逻辑表达式构成触发条件使用条件元覆盖准则进行用例扩展。

### 3.1 基路径覆盖准则

采用深度优先的算法寻找基路径，并对访问过的路径进行标记以避免重复，对于状态转移图中存在的循环，算法保证每个循环路径在测试用例集中最多出现一次。生成的用例个数为状态图的圈复杂度：$V(G) = e - n + 2p$ 。

### 3.2 转移对覆盖准则

以每个状态转移节点为验证对象，设该状态转移节点的入度为 $m$，出度为 $n$，则按照每一种流入该状态节点的转移连线和从该状态节点留出的转移连线的组合，生成 $m * n$ 个测试用例。

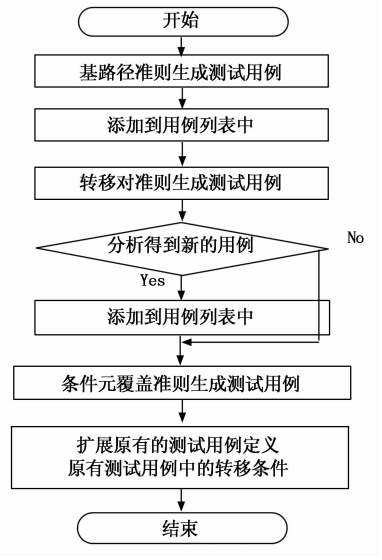


图 3 测试用例生成算法设计流程图

### 3.3 路径相关的状态转移条件判断

对于描述软件应激行为的状态转移图，其状态切换除与外界激励相关外，还可能与之前经历过的状态路径相关，一个典型的例子如图 4 所示，省略了大部分的转移条件，仅保留与转移路径相关的条件。

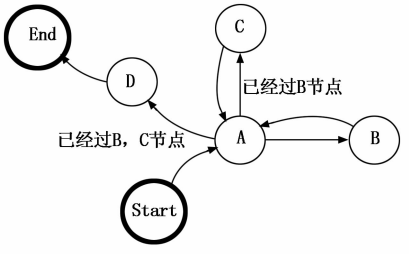


图 4 路径相关的状态转移判定

为了使模型能够表述此类触发条件，可以引入一种特殊的条件元，即 is _ in _ Path（State），该条件元也具有一个布尔返回值。图 4 中由 A 转 C 的触发条件可以用 is _ in _ Path（B）表示，由 A 转 D 的触发条件则用 is _ in _ Path（B）和 is _ in _ Path（C）表示。

生成测试用例时，算法应自动替换不会发生的状态转移路径。按照基路径生成方法可能会生成的测试用例考查的路径为："A－D"、"A－B－D"、"A－C－D"、"A－C－B－D"，这些都是非法的，应被自动剔除，唯一合法的测试路径为 "A－B－C－D"。

### 3.4 条件元覆盖

在基路径覆盖的基础上，对于在每一处转移连线上的条件元或逻辑表达式，按照上一节所述的方法生成考查状态转移逻辑判定的测试用例，用例同时考查转移路径和转移行为的正确性。

## 4　算法结果验证

为验证上述基于状态转移图的测试用例生成算法的合理性、全面性和完备性，以实际某箭上软件的全周期时序控制为例，首先建立状态转移图（见图 5）。

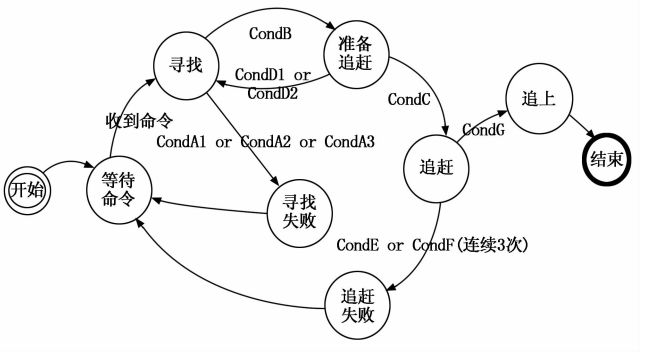


图 5　某型号软件动态特征状态转移图

其次依据状态转移图的描述，依据基路径覆盖准则生成基于路径分析的测试用例 4 个，即 1）等待命令→寻找→准备追赶→追赶→追上；2）等待命令→寻找→准备追赶→追赶→追赶失败→等待命令→寻找→准备追赶→追赶→追上；3）等待命令→寻找→准备追赶→寻找→准备追赶→追赶→追上；4）等待命令→寻找→寻找失败→等待命令→寻找→准备追赶→追赶→追上。

再次依据转移对覆盖准则对生成用例进行扩展，补充了 1 个用例，即等待命令→寻找→准备追赶→寻找→寻找失败→等待命令→寻找→准备追赶→追赶→追上。

最后根据条件元覆盖进行转移条件的定义和扩充，即 1）由“寻找→寻找失败”需有 3 个考察转移条件的用例；2）由“准备追赶→寻找”需有 2 个考察转移条件的用例；3）由“追赶→追赶失败”需有 2 个考察转移条件的用例。最终生成的测试用例如表 2 所示，实测生成时间为 4 s。

其中，1、2、4、5、6、7、8 均与同人工设计测试用例相同。试验结果表明自动生成测试用例软件可稳定自动生成，测试用例可完全覆盖软件动态特征功能和性能需求，覆盖性等于或高于人工设计的测试用例，且缩短用例生成时间至 4 s。

## 5　结束语

本文提出了一种基于状态转移图的测试用例自动生成算法设计方法，以期实现时序控制功能测试用例自动生成，提升测试效率和质量，为软件自动化测试提供一定的借鉴。本文首先分析时序控制特性，其次选取状态转移图来表征，自定义所需图元属性以及图元关系、触发条件全面性与一致性的约束条件从而保证模型合理性；再次采用基路径覆盖、转移对覆盖、条件元覆盖 3 种覆盖准则相结合方式实现测试用例的自动生成；最后以某箭上软件全飞行周期时序控制为例，经以具备较复杂逻辑的被测软件测试用例生成结果表明，软件可稳定自动生成覆盖率较高的测试用例，测试用例可完全覆盖软件动态特征功能和性能需求，从而验证了上述方法的有效性和稳定性。相较与原人工设计用例方式，该方法还提升了测试用例生成效率。

表 2　自动生成用例算法生成的测试用例

| 编号 | 用例描述 |
|---|---|
| 1 | 等待命令→寻找→准备追赶→追赶→追上 |
| 2 | 等待命令→寻找→准备追赶→(追赶→追赶失败，CondE 满足)→等待命令→寻找→准备追赶→追赶→追上<br>注：CondE、CondF 都不满足时，不触发“追赶→追赶失败”的转移 |
| 3 | 等待命令→寻找→准备追赶→(追赶→追赶失败，CondF 满足 3 次)→等待命令→寻找→准备追赶→追赶→追上<br>注：CondF 连续满足小于 3 次时，不触发“追赶→追赶失败”的转移 |
| 4 | 等待命令→寻找→(准备追赶→寻找，CondD1 满足)→准备追赶→追赶→追上<br>注：CondD1、CondD2 都不满足时，不触发“准备追赶→寻找”的转移 |
| 5 | 等待命令→寻找→(准备追赶→寻找，CondD2 满足)→准备追赶→追赶→追上 |
| 6 | 等待命令→(寻找→寻找失败，CondA1 满足)→等待命令→寻找→准备追赶→追赶→追上<br>注：CondA1、CondA2、CondA3 都不满足时，不触发“寻找→寻找失败”的转移 |
| 7 | 等待命令→(寻找→寻找失败，CondA2 满足)→等待命令→寻找→准备追赶→追赶→追上 |
| 8 | 等待命令→寻找→准备追赶→(寻找→寻找失败，CondA3 满足)→等待命令→寻找→准备追赶→追赶→追上 |

**参考文献：**

[1] 赵　斌，等. 软件测试技术经典教程 [M]. 北京：科学出版社，2007.

[2] 王雅文，宫云战，杨朝红. 软件测试工具综述 [J]. 北京化工大学学报（自然科学版），2007（s1）：1－4.

[3] 林　晨. 嵌入式箭载计算机控制软件测试关键技术研究 [D]. 上海：上海交通大学，2014.

[4] 王海燕. 箭载飞行控制软件通用仿真动态测试平台研究和实现 [D]. 上海：同济大学，2007.

[5] 佟　轶，董碧丹. UML 状态图的测试用例自动生成 [J]. 微计算机信息，2011，27（9）：201－203.

[6] 张　卉，高仲合，黄　铭，等. UML 测试用例自动生成方法研究 [J]. 电子技术，2016，45（4）：52－54.

[7] 杨　晶，顾春华. 基于 UML 状态图的测试用例自动生成方法 [J]. 华东理工大学学报（自然科学版），2011，37（3）：346－351.

[8] 黄陈辉. 基于混沌遗传算法的测试用例自动生成研究 [D]. 上海：上海师范大学，2019.