耦合条件的 MC/DC 测试用例集生成算法①

谢祥南, 魏延栋

(中国航空无线电电子研究所, 上海 200241)

摘 要: MC/DC 是一套航电 A 级软件的测试覆盖性准则, 可以有效减少测试用例量. 针对如何快速获取尽可能小的测试用例集这一难点展开研究, 重点关注于具有耦合条件的逻辑表达式, 提出了两套解决方案, 分别用于解决零耦合/弱耦合条件和强耦合条件问题, 并给出了示例证明. 结果表明, 灵活使用两套算法, 可以全面解决一般逻辑表达式的 MC/DC 测试用例集的快速生成问题.

关键词: MC/DC 覆盖; 测试用例; 条件; 判定; 耦合; 逻辑

Test Case Generation Algorithm Based on MC/DC with Coupling Conditions

XIE Xiang-Nan, WEI Yang-Dong

(China National Aeronautical Radio Electronics Research Institute, Shanghai 200241, China)

Abstract: MC/DC is a coverage criterion for the verification of avionics software of Level A, which can massively lower down the number of test cases. This paper focuses on the research of the logical expression with coupling conditions, and studies how to obtain the test case set as small as possible. It proposes two solutions which can be used respectively to solve the problems under the conditions of zero-coupling/weak-coupling and strong-coupling, and related examples are showed. The result shows that flexible usage of the two algorithms can solve the problems of rapid generation of MC/DC test case set for general logic expression.

Key words: MC/DC; test case; condition; decision; coupling; logic

航空电子软件等级由系统安全性评估过程中予以确定的软件对潜在失效状态的影响程度决定. 不同等级软件对应不同的测试覆盖性要求, 其中, A 级软件测试应满足 MC/DC(Modified Condition/Decision Coverage)覆盖[1].

MC/DC, 即修订的条件/判定覆盖, 是有别于语句覆盖、条件覆盖、判定覆盖等的一种覆盖性测试准则[2].

MC/DC 要求: 程序中, 每一个入口点和退出点至少被引用一次; 每一个判定的所有可能输出结果至少产生一次; 判定中, 每一个条件的所有可能输出结果至少产生一次, 且能独立影响该判定的输出[1].

如何快速获得满足 MC/DC 要求的最小测试用例集, 国内已有不少学者作了研究, 例如袁军的“经验”法[3], 朱晓波等的最小真值表法[4], 葛汉强的递归分块矩阵生成算法[5], 以及段飞雷等的快速生成算法[6]. 前三种算法均通过对最简表达式的测试用例集进行递归解析实现, 最后一种算法则给出了一套规则, 通过不断规约获得最终测试用例集. 算法各有优势, 但无一例外地(袁军有涉及, 但未详细展开)规避了对存在耦合条件的逻辑表达式的 MC/DC 测试用例集生成情况的分析.

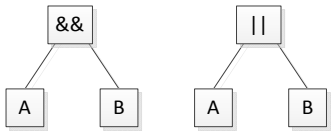
本文首先从最小判定入手, 通过扩展逐步总结出初步的生成算法. 然后分别通过分析零耦合条件、弱耦合条件和强耦合条件逻辑表达式示例, 对原算法进行优化或修正, 得出了最终的两套生成算法.

1 研究对象

条件(Condition): 指逻辑表达式中不包含与、或、非等布尔操作符的部分.

判定(Decision): 指含两个及以上条件的逻辑关系式. 其中, 仅由两个条件组成的逻辑表达式为最小判定, 可用图 1 所示语法树表示.

① 收稿时间:2016-09-29;收到修改稿时间:2016-11-03 [doi:10.15888/j.cnki.csa.005793]



(a) 与树 (b) 或树

图 1 最小语法树

耦合条件是指, 逻辑表达式中, 同一条件重复出现, 或某条件输出值制约其它条件的输出值.

由 MC/DC 要求可知, 每个耦合条件均必须满足其所有可能输出结果至少产生一次, 且能独立影响表达式的输出. 另规定, 在本文中, *A* 与!*A* 视为两个互斥条件.

关系运算(“>”、“<”)需要考虑多值情况, 可以在提出的算法基础上进行衍生研究, 本文不展开讨论. 本文主要针对仅含基本的与、或、非的一般逻辑表达式.

2 算法设计

2.1 基本定义

独立条件: 当前独立影响逻辑表达式输出结果的条件.

受控条件: 当前除独立条件以外的条件.

比较条件: 当前准备处理的受控条件.

独立判定: 当前独立影响逻辑表达式输出结果的判定, 在逻辑表达式中至少有一个.

受控判定: 当前除独立判定以外的判定.

比较判定: 当前准备处理的受控判定, 即含比较条件的判定.

同父判定: 同时包含独立条件与比较条件的判定中含最少条件的判定.

一级独立子判定: 同父判定下一级包含独立条件的判定.

一级比较子判定: 同父判定下一级包含比较条件的判定.

首条件: 判定的第一个条件.

决定条件: 决定当前受控判定输出结果的条件, 通常为首条件.

前置逻辑符: 逻辑表达式中出现在该条件或判定前面的逻辑符号(“&&”, “||”).

补集: 某条测试用例的独立条件及其重复条件值取反, 受控条件值保持不变.

2.2 通用算法

对于图 1 的最小判定, 当逻辑符为“&&”时, 比较条件必须设为 True, 独立条件的值才能独立影响同父判定的结果. 同理, 当逻辑符为“||”时, 比较条件则应设为 False.

因此, 可得与树的最小测试用例集如表 1 所示.

表 1 “与树”最小测试用例集

No.	*A*	*B*	Output
1	F	T	F
2	T	F	F
3	T	T	T

进一步地, 在“与树”基础上增加一个“或树”, 如图 2 所示.

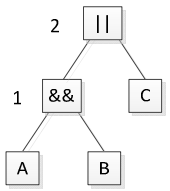


图 2 衍生语法树 1

图 2 的测试用例集可通过下述步骤获取:

a) *A* 为独立条件, *B*、*C* 为受控条件: 首先, 设 *A* 为 False; 然后, 选 *B* 为比较条件, 判定 1 为 *A*、*B* 的同父判定, 由于 *A* 应独立影响判定 1 的输出结果, 因此 *B* 应设为 True; 选 *C* 为比较条件, 判定 2 为判定 1 与 *C* 的同父判定, 为保证 A 独立影响判定 2 的输出, *C* 应设为 False; 整理可得表 2.

表 2 衍生语法树 1 第一组测试用例真值表

A	*B*	*C*	Output
F	T	F	F

b) *B* 为独立条件, *A*、*C* 为受控条件: 为保证与第一步测试用例不一样, 设 *B* 为 False; 同第一步, *A* 应设为 True, *C* 应设为 False; 整理可得表 3.

表 3 衍生语法树 1 第二组测试用例真值表

A	*B*	*C*	Output
T	F	F	F

c) *C* 为独立条件, *A*、*B* 为受控条件: 同上, *C* 设为 True; *C* 应独立影响判定 2 的输出结果, 因此, 判定 1 的输出应为 False, 此时 *A*、*B* 任一设为 False 即可, 同时考虑到最小测试用例集要求, 优先选择 *A* 作为判定 1 的决定条件, 因此 *A* 应设为 False, *B* 设为 True, 如表 3 所示; 整理可得表 4.

表 4　衍生语法树 1 第三组测试用例真值表

A	*B*	*C*	Output
F	T	T	T

d) a)、b)、c)的测试用例只能决定当前独立条件的一种输出, 因此需增加各自测试用例的补集, 整理可得如表 5 所示测试用例真值初表.

表 5　衍生语法树 1 测试用例真值初表

No.	*A*	*B*	*C*	Output
1	F	T	F	F
2	T	F	F	F
3	F	T	T	T
4	T	T	F	T
5	T	T	F	T
6	F	T	F	F

e) 去除所有重复条件, 即得到最终测试用例真值表, 如表 6 所示.

表 6　衍生语法树 1 测试用例真值表

No.	*A*	*B*	*C*	Output
1	F	T	F	F
2	T	F	F	F
3	F	T	T	T
4	T	T	F	T

再进一步, 将图 2 中“或树”的右子树替换为判定 3, 且判定 3 的首条件与判定 1 的首条件一致.

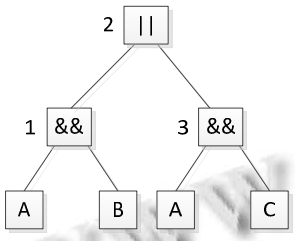


图 3　衍生语法树 2

图 3 的测试用例集可通过下述步骤获取:

a) 第一个 *A*(简称 *A_1*, 下同)为独立条件, *B*、第二个 A(简称 *A_2*, 下同)、*C* 为受控条件: 首先, 设 *A_1* 为 False; 然后, 选 *B* 为比较条件, 判定 1 为 *A_1*、*B* 的同父判定, 由于 *A_1* 应独立影响判定 1 的输出结果, 因此 *B* 应设为 True; *A_2* 与 *A_1* 为同一条件(输出值相同), 因此亦设为 False; 选 *C* 为比较条件, 由于判定 3 的原决定条件 *A_2* 受 *A_1* 的值限制, 因此 *C* 转为该判定的决定条件, 设 *C* 为 False; 整理可得表 7.

表 7　衍生语法树 2 第一组测试用例真值表

A	*B*	*A*	*C*	Output
F	T	F	F	F

b) *B* 为独立条件, *A_1*、*A_2*、*C* 为受控条件: 为保证与第一步测试用例不一样, 设 B 值为 False; 同第一步, *A_1*、*A_2* 应设为 True, *C* 应设为 False; 整理可得表 8.

表 8　衍生语法树 2 第二组测试用例真值表

A	*B*	*A*	*C*	Output
T	F	T	F	F

c) *A_2* 为独立条件, *A_1*、*B*、*C* 为受控条件: 同上, *A_1*、*A_2* 值设为 False; 同第一步, *B* 为判定 1 的决定条件, 因此 *B* 设为 False. *A_2* 应独立影响判定 3 的输出结果, 因此, *C* 设为 True; 整理可得表 9.

表 9　衍生语法树 2 第三组测试用例真值表

A	*B*	*A*	*C*	Output
F	F	F	T	F

d) *C* 为独立条件, *A_1*、*B*、*A_2* 为受控条件: 同上, *C* 设为 False, 且应独立影响判定 3 的输出结果, 因此 *A_1*、*A_2* 值设为 True; 同 c), *B* 为判定 1 的决定条件, 因此 *B* 设为 False; 整理可得表 10.

表 10　衍生语法树 2 第四组测试用例真值表

A	*B*	*A*	*C*	Output
T	F	T	F	F

e) a)、b)、c)、d)的测试用例只能决定当前独立条件的一种输出, 因此需增加各自测试用例的补集, 同上, 整理可得表 11.

表 11　衍生语法树 2 测试用例真值初表

No.	*A*	*B*	*A*	*C*	Output
1	F	T	F	F	F
2	T	F	T	F	F
3	F	F	F	T	F
4	T	F	T	F	F
5	T	T	T	F	T
6	T	T	T	F	T
7	T	F	T	T	T
8	T	F	T	T	T

f) 去除所有重复条件, 即得到最终测试用例真值表, 如表 12 所示.

表 12 衍生语法树 2 测试用例真值表

No.	*A*	*B*	*A*	*C*	Output
1	F	T	F	F	F
2	T	F	T	F	F
3	F	F	F	T	F
4	T	T	T	F	T
5	T	F	T	T	T

依此类推, 测试用例集生成步骤为:

第一步: 选取逻辑表达式的第一个条件 C_1 作为独立条件, 设值为 False; 依次选取后续受控条件为比较条件, 设比较条件为 $C_i(i>=2)$, 找到所有比较判定 D_1~$D_n(n>=1)$, 其中, D_1 为同父判定, $D_2(n>=2)$为一级比较子判定; 从 D_n 开始往前遍历, 判断 C_i 是否为当前判定的决定条件, 如果是, 则继续向前查找, 直到找到决定条件不为 C_i 的比较判定 $D_m(m>=1)$, 则 C_i 的设定值由 $D_{m-1}(m>=2$, 当 $m=1$ 时, 则为 $C_i)$的前置逻辑符决定; 如果遍历到 D_2, C_i 依然为决定条件, 则如果比较条件出现在独立条件之前, 则其值由一级独立子判定(如果没有, 则为 C_1)的前置符号决定, 否则, 则由 D_2(当 $n=1$ 时, 则为 C_i)的前置逻辑符决定. 然后依次获取其余受控条件的设定值.

某判定的决定条件指: 该判定从左至右第一个未曾在当前测试用例中设定过, 或者设定的判断依据为当前判定的条件.

对于重复条件或反条件, 则设值由当前测试用例中的前次设定值为依据.

第二步: 依次选取后续条件为独立条件, 设值与上一条测试用例中设值相反. 然后依次选取受控条件作为比较条件, 设值方法参照第一步.

第三步: 逐条添加前两步产生的测试用例的补集, 并剔除所有的重复测试用例, 即得到最终测试用例集.

3 示例介绍及论证

为验证上述算法, 本文择取三个示例——零耦合条件、弱耦合条件、强耦合条件逻辑表达式进行分析.

3.1 零耦合条件

零耦合条件逻辑表达式是指, 式中所有条件有且仅出现一次, 且无关联条件存在.

以*((A&&B)||C)&&((D||E)&&F)*为例, 式中所有条件均可取 T/F 两种值, 且相互间不会产生影响.

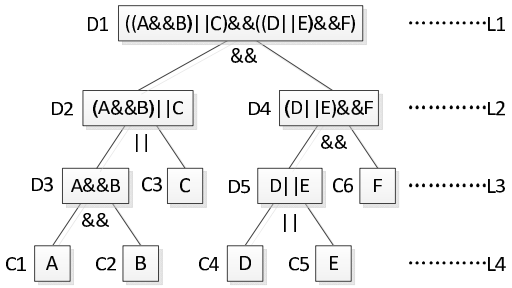


图 4 零耦合条件逻辑表达式示例

按照第 2 节算法, 可得如表 13 所示的测试用例真值表.

表 13 零耦合条件示例逻辑测试用例真值表

No.	*A*	*B*	*C*	*D*	*E*	*F*	Output
1	F	T	F	T	F	T	F
2	T	F	F	T	F	T	F
3	F	T	T	T	F	T	T
4	T	T	F	F	F	T	F
5	T	T	F	F	T	T	T
6	T	T	F	T	F	F	F
7	T	T	F	T	F	T	T

根据 MC/DC 理论, 可得各条件的测试用例对如表 14 所示.

表 14 零耦合条件示例逻辑测试用例对

Condition/Output	T	F
A	7	1
B	7	2
C	3	1
D	6	4
E	5	4
F	7	6

可见, 生成的测试用例集满足 MC/DC 理论, 且满足 n+1 的最小用例集准则.

经过多次试验, 不难发现, 对于零耦合条件, 第 2 节方法中的第三步, 可以改为: 增加第一条测试用例的补集, 即可得到最终的测试用例集.

3.2 弱耦合条件

弱耦合条件是指, 逻辑表达式中部分条件重复出现, 但任一判定的所有条件不能均在之前的判定中作为决定条件出现过.

以*((A&&B&&C)||(A&&(B&&D)))||(M&&N)*为例, 其中条件 *A* 和条件 *B* 均出现两次. 首先分解成图 5 所示树形结构.

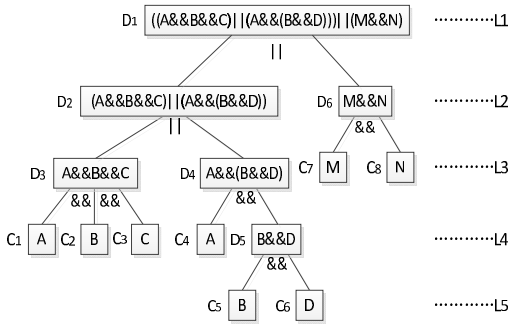


图 5　弱耦合条件逻辑表达式示例

按照第 2 节算法, 可得如表 15 所示的测试用例真值表.

表 15　弱耦合条件示例逻辑测试用例真值表

No.	*A*	*B*	*C*	*D*	*M*	*N*	Output
1	F	T	T	F	F	T	F
2	T	F	T	F	F	T	F
3	T	T	F	F	F	T	F
4	F	T	F	T	F	T	F
5	T	F	F	T	F	T	F
6	F	T	T	F	T	T	T
7	F	T	T	F	T	F	F
8	T	T	T	F	F	T	T
9	T	T	F	T	F	T	T

根据 MC/DC 理论, 可得各条件的测试用例对如表 16 所示.

表 16　弱耦合条件示例逻辑测试用例对

Condition/Output	T	F
A_1	8	1
B_1	8	2
C	8	3
A_2	9	4
B_2	9	5
D	7	6
M	6	1
N	9	3

表 16 中, *A_1* 与 *A_2*, *B_1* 与 *B_2* 的测试用例对是不一样的, 这是因为在逻辑式中, 它们分属不同的判定.

将每个条件视为不同条件, 则得到的测试用例集依然满足 *n*+1 规则.

3.3 强耦合条件

强耦合条件是指, 逻辑表达式中部分条件重复出现, 且存在某一判定的所有条件均在之前的判定中作为决定条件出现过.

以*(A&&B)||(A&&C)||(A&&D)||(B&&C)||(B&&D)||(C&&D)* 为例, 其中 *(B&&D)*判定中的 *B* 和 *D* 均在之前的判定中作为决定条件出现过. 首先分解成图 6 所示树形结构.

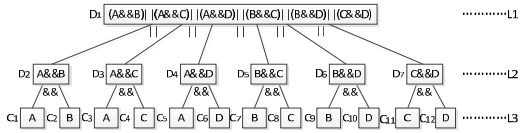


图 6　强耦合条件逻辑表达式示例

按照上述算法, 可得如表 17 所示的测试用例真值表.

表 17　强耦合条件示例逻辑测试用例真值表 1

No.	*A*	*B*	*C*	*D*	Output
1	F	T	F	F	F
2	T	F	F	F	F
3	F	F	T	F	F
4	F	F	F	T	F
5	F	T	T	F	T
6	F	T	T	T	T
7	T	T	F	F	T
8	T	F	T	F	T
9	T	F	F	T	T
10	F	T	F	T	T

经检查, 该测试用例集不满足 MC/DC 要求, 可见, 第 2 节的算法不适用于强耦合条件的逻辑表达式的测试用例生成问题.

因此原算法需要进行适当调整, 调整后的具体步骤为:

第一步: 选取逻辑表达式的第一个条件 C_1 作为独立条件, 设值为 False; 依次选取后续受控条件为比较条件, 设比较条件为 $C_i(i>=2)$, 找到所有比较判定 D_1~$D_n(n>=1)$, 则 D_1 为同父判定. 如果存在一级比较子判定 D_2, 则 C_i 的值由该判定的前置符号决定, 如果不存在, 则由 C_i 的前置符号决定.

第二步: 依次选取后续条件为独立条件, 设值与上一条测试用例相反. 然后依次选取受控条件作为比较条件, 如果比较条件出现在独立条件之前, 则比较条件的值由一级独立子判定决定(如果不存在, 则由独立条件的前置符号决定); 如果当前比较条件出现在独立条件之后, 则按照第一步方法取值.

第三步: 逐条添加前两步产生的测试用例的补集,

并剔除所有重复的测试用例, 即得到最终测试用例集.

按照上述方法, 可得如表 18 所示的测试用例真值表.

表 18 强耦合条件示例逻辑测试用例真值表 2

No.	A	B	C	D	Output
1	F	T	F	F	F
2	T	F	F	F	F
3	F	F	T	F	F
4	F	F	F	T	F
5	F	T	T	F	T
6	F	F	T	T	T
7	T	T	F	F	T
8	T	F	T	F	T
9	T	F	F	T	T
10	F	T	F	T	T

根据 MC/DC 理论, 可得各条件的测试用例对如表 19 所示.

表 19 强耦合条件示例逻辑测试用例对

Condition/Output	T	F
A_1	7	1
B_1	7	2
A_2	8	3
C_1	8	2
A_3	9	4
D_1	9	2
B_2	5	3
C_2	5	1
B_3	10	4
D_2	10	1
C_3	6	4
D_3	6	3

可见, 测试用例集严格满足 MC/DC 要求, 且数量远小于全遍历情况(24 个).

经检验发现, 该算法同样适用于零耦合条件和弱耦合条件情况, 但生成的均非最小测试用例集.

4 结论

本文共提出了两套算法: 第一套可快速有效地处理零耦合条件和弱耦合条件逻辑的解析问题, 并保证生成最小测试用例集; 第二套则主要用于解决强耦合条件问题, 且生成的测试用例数远远低于全遍历的情况.

通过以上算法, 可以快速生成所需的测试用例集, 有利于减轻测试工程师的工作量, 提高工作效率.

参考文献

1 DO-178B. Software Consideration in Airborne Systems and Equipment Certification. USA: RTCA,1992.

2 NASA. A Practical Tutorial on Modified Condition/Decision Coverage. USA: NASA, 2001.

3 段飞雷,吴晓,张凡等.MC/DC 最小测试用例集快速生成算法.计算机工程,2009,35(17):40–45.

4 朱晓波,杨伟民,叶芯.更改条件/判定覆盖最小真值表生成算法及其应用.上海理工大学学报,2007,29(1):84–88.

5 葛汉强.MC/DC 最小测试用例集递归分块矩阵生成算法.计算机系统应用,2011,20(7):195–198.

6 袁军.基于 MC/DC 最小测试用例集设计方法研究.航空电子技术,2010,41(3):51–54.