一种基于动态故障树的SBDD方法

张晓策 燕雪峰 周 勇
(南京航空航天大学计算机科学与技术学院 南京 211106)

摘 要 在分析基于Pandora的动态故障树时,SBDD方法未考虑各底事件间复杂的关系,造成生成的SBDD中存在无效分支,即计算的不交化割集中存在无效割集。针对该问题,提出了一种基于动态故障树的SBDD方法,可以动态删除无效节点,避免无效分支的产生。该方法主要包括两个方面:基于结构式排序方法的关系式排序方法和动态优化SBDD生成算法。关系式排序方法的基本思想是利用故障树的结构关系和底事件间的关系给底事件赋予不同的排序优先级。在底事件排序队列的基础上,使用动态优化SBDD生成算法来生成SBDD。在计算过程中,该算法动态删除无效的节点,使结果中不存在无效割集。实验结果表明,在相近的时间内,使用基于动态故障树的SBDD方法生成的SBDD规模更小,不交化割集数目更少且不存在无效割集。

关键词 Pandora,动态故障树,SBDD方法,动态优化SBDD生成算法,关系式排序方法
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.09.037

Method of SBDD Based on Dynamic Fault Tree

ZHANG Xiao-ce YAN Xue-feng ZHOU Yong
(College of Computer Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing 211106, China)

Abstract When analyzing the dynamic fault tree of Pandora, the method of SBDD doesn't take into account the complex relations between events. It causes that the generated SBDD has invalid branches where there are invalid cut sets in the non-intersect minimum cut sets. Aiming at this problem, an improved method of SBDD was proposed in this paper which can remove invalid node dynamically and avoid invalid branches. The improved method of SBDD mainly includes two aspects, the relation sorting method based on the structure sorting method and the dynamic optimization generation algorithm of SBDD. The basic idea of the relation sorting method is giving the events different sorting priority based on the structure relations of the fault tree and the relationship between events. The dynamic optimization generation algorithm of SBDD is used to calculate the SBDD based on the event sequencing queue. In the process of calculation, the algorithm removes invalid node dynamically and makes the results don't contain invalid cuts. Experiments show that the SBDD generated by the improved method of SBDD is smaller and the non-intersect minimum cut sets are less and don't contain invalid sets in a approximate time.

Keywords Pandora, Dynamic fault tree, Method of SBDD, Generation algorithm of SBDD, Relation sorting method

1 引言

Martin Walker于2005年首次提出Pandora[1]。Pandora通过引入新的时间门来更加完整地描述底事件的顺序关系。同时,Pandora添加了新的规则和算法,使得Pandora动态故障树得到解析[2]。

SBDD方法是一种基于二元决策图(Binary Decision Diagram,BDD)的动态故障树分析方法。在分析基于Pandora的动态故障树时,SBDD方法未考虑底事件间的关系,从而导致使用SBDD方法建立的SBDD图中存在无效的分支。这些分支不仅占用存储空间,并且使计算的不交化割集中存在无效的割集,这些无效割集影响了故障树的定性分析和定量分析。针对该问题,本文提出一种基于动态故障树的SBDD方法,用于分析基于Pandora的动态故障树。该方法采用基于结构式排序方法的关系式排序方法进行动态故障树的变量排序,并且通过使用基于SBDD生成算法的动态优化生成算法,在SBDD的建立过程中动态删除无效分支。

2 SBDD方法

SBDD(Sequential Binary Decision Diagram)是受二元决策图(BDD)影响的动态故障树分析方法[3]。SBDD的核心思想是:将动态故障树中的每个动态门看作一个连续的底事件,并使用一个对应的序列代替它;然后通过生成算法将取代后的动态故障树转化成相应的SBDD。

到稿日期:2016-08-16 返修日期:2016-12-05 本文受十三五重点基础科研项目(JCKY2016206B001),江苏省六大人才高峰项目(XXRJ-004)资助。

张晓策(1991-),男,硕士生,主要研究领域为系统建模与仿真,E-mail:zhang_xiao_ce@163.com;燕雪峰 教授,主要研究领域为软件工程方法论、系统建模与仿真等;周 勇(1973-),男,博士,主要研究领域为人工智能、专家系统、智能推理等。

SBDD 是一种有向无环图，可以表示为布尔函数 $f:\{x_1,x_2,\cdots,x_n\}\rightarrow\{0,1\}$[4]。SBDD 的叶节点为 0 和 1，其他节点用变量 x_i 表示，并且该节点还有两个叶子节点。左右子分支分别代表在 $x_i=0$ 和 $x_i=1$ 的情况下的函数。

$f(x_1,x_2,\cdots,x_n)=x_1\wedge f(1,x_2,\cdots,x_n)+\bar{x}_1\wedge f(0,x_2,\cdots,x_n)$。其中，$x_1$ 是故障树的顶节点，$f(1,x_2,\cdots,x_n)$ 和 $f(0,x_2,\cdots,x_n)$ 是 x_1 的孩子函数。递归地使用该函数关系，直到所有的变量转化成 SBDD 节点，形成完整的 SBDD。构建 SBDD 使用 Shannon 公式[5-7]扩展构建顶节点。图 1 给出一个基于 Pandora 的动态故障树，图 2 给出使用 SBDD 方法生成的 SBDD。

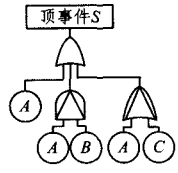

图 1 基于 Pandora 的动态故障树

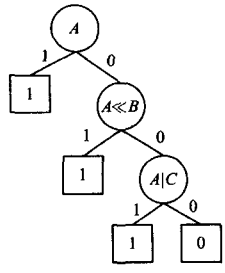

图 2 SBDD

3 基于动态故障树的 SBDD 方法

在基于 Pandora 的动态故障树中，由于重复事件的出现，各底事件不是完全相互独立的，各底事件之间存在着复杂的关系。假设存在两个变量 x_i 和 x_j，当 $x_i=0$ 时，$x_j=0$。SBDD 是故障树中所有节点的全排序，则一定存在这样一个序列：$x_1,x_2,\cdots,\bar{x}_i,\cdots,x_j,\cdots,x_n$。由已知条件可知，上述序列是 SBDD 中的无效分支。这种分支不仅在运算过程中占用大量空间，而且使求得的不交化割集中存在无效割集。

SBDD 中存在无效节点的原因在于 SBDD 方法未考虑各底事件间的相关性。为了解决该问题，本文综合考虑了底事件间的关系，提出了基于动态故障树的 SBDD 方法。该方法主要对 SBDD 方法中的结构式排序方法和生成算法进行改进。在建立 SBDD 的过程中，基于动态故障树的 SBDD 方法动态地删除无效节点，以避免无效分支的生成。

3.1 关系式排序方法

在将故障树转化为 BDD 的过程中，存在一个预处理过程，需要先对底事件进行排序[8]。故障树的 BDD 排序方法主要可以分为两类：结构式和加权式。结构式排序方法的基本思想是：从故障树的顶端从上而下、从左往右选择各底事件并依次排序。当已经被选择的底事件再次出现时，不再重复排序[9]。但是，该方法仅考虑了故障树的结构，未考虑重复底事件之间的关系。

本文提出的基于结构式排序方法的关系排序方法继承了结构式方法的特点，着重强调了动态底事件和静态底事件间的关系。关系排序方法先根据故障树的结构特点和重复底事件的次数来确定部分底事件的次序，然后根据底事件间的包含关系来确定其他底事件的排序。

关系式排序方法遵循下述规则。

规则 1 静态底事件位于动态底事件之前。

规则 2 对于静态底事件而言，静态底事件越靠近故障树顶事件，则需要排在越靠前的位置。

规则 3 当静态底事件处于同层时，优先排出现次数多的静态底事件。当出现次数相同时，优先排与已排序的静态底事件在同一个静态门的静态底事件。若无，则按照从左到右的顺序进行排序。

规则 4 动态底事件的排序方法为：3 个门按照优先排列 SAND 门(A&B)、POR 门(A|B)和 PAND 门(A->B)的先后顺序进行排列。

对基于 Pandora 的动态故障树运用关系式排序方法的具体步骤如下：

1)第一次遍历转换后的故障树。统计故障树中各底事件出现的次数，其中动态底事件也是由静态底事件组合而成的，因此也需要统计动态底事件中的静态底事件数量。

2)第二次采用层序方式遍历转换后的动态故障树。每一层的底事件按照规则 3 和规则 4 进行排序，得到排序集合 A_i $(1<i<n)$。将每层的排序结合 A_i 合并到总排序集合 A 中。层序遍历完动态故障树，得到的集合 A 就是动态故障树的底事件排序队列。

例 1 基于图 1 中的故障树，使用基于结构式排序方法的关系式排序方法求解该故障树的排序队列。

图 1 中出现的底事件分别为 $A,B,C,A<B$ 和 $A|B$，各底事件出现的次数如表 1 所列。

表 1 底事件出现次数

底事件	A	B	C	$A<B$	$A\|B$
次数	3	1	1	1	1

该故障树只有一层，直接应用排序规则对其进行排序，得到底事件序列为：$A<(A|B)<(A<B)$。

3.2 动态优化 SBDD 生成算法

基于 SBDD 生成算法的动态优化生成算法主要是将原 SBDD 生成公式[10]进行扩展，使其可以在递归过程中保持各底事件间的关系，动态判断并删除无效的节点。

动态优化 SBDD 生成算法即扩展的 Shannon 公式如下：

$$A\langle op\rangle B=ite(x,A_1,A_0)\langle op\rangle ite(y,B_1,B_0)$$

$$=\begin{cases}ite(x,A_1\langle op\rangle B_{|x=1},A_0\langle op\rangle B_{|x=0}).x<y\\ite(x,A_1\langle op\rangle B_1,A_0\langle op\rangle B_0).x=y\\ite(y,B_1\langle op\rangle A_{|y=1},B_0\langle op\rangle A_{|y=0}).x>y\end{cases}$$

A 和 B 是 ite 形式的布尔函数。$\langle op\rangle$ 对应于故障树中门类型的布尔操作，包括与门和或门。

无效分支产生在使用 Shannon 公式计算 $A\langle op\rangle B$ 的过程中，由于各底事件的重新组合，A 或 B 中出现与上层底事件冲突的节点，这些节点即为无效节点。在扩展的 Shannon 公式中，当 $x<y$ 时，无效节点可能出现在 $B|_{x=1}$ 和 $B|_{x=0}$ 中。在计算 $A_1\langle op\rangle B|_{x=1}$ 和 $A_0\langle op\rangle B|_{x=0}$ 之前，分别使用下述规则在 $x=1$ 或 $x=0$ 的情况下判断和删除 B 中的无效节点，避免运算结果中的无效分支。

在基于 SBDD 生成算法的动态优化 SBDD 生成算法中，$X_{|y=k}$（X 为布尔函数，y 为底事件，$k=0$ 或 1）采用下列 3 条计算规则进行无效节点的判断和删除。

已知：$X=\text{ite}(x_i\langle\text{op}\rangle x_j,M,N)$，$x_i$ 和 x_j 为静态底事件，$x_i\langle\text{op}\rangle x_j$ 是含有 A 的动态底事件。

计算规则一：

$\text{ite}(x_i\langle\text{op}\rangle x_j,M,N)_{|x_i=0}=N_{|x_j=0}$，$\langle\text{op}\rangle\in\{\&,<,|\}$

计算规则二：

$\text{ite}(x_i\langle\text{op}\rangle x_j,M,N)_{|x_i\&x_j=1}=N_{|x_i\&x_j=1}$，$\langle\text{op}\rangle\in\{<,|\}$

计算规则三：

$\text{ite}(x_i\langle\text{op}\rangle x_j,M,N)_{|(x_i|x_j=0)}=N_{|(x_i|x_j=0)}$，$\langle\text{op}\rangle\in\{<\}$

在扩展的 Shannon 公式中，规则一用来化简类似 $A_0\langle\text{op}\rangle B_{|x=0}$ 的式子。

A_0 表示为 $\bar{x}\prod_{i=1}^{i\leqslant n}x_i\wedge f(x_1,\cdots,x_n)$，其中，$\bar{x}\prod_{i=1}^{i\leqslant n}x_i$ 表示 A_0 包含的以 $\bar{x}$ 开始的底事件的全排序。

$B:\prod_{j=1}^{j\leqslant n}y_j\wedge g(y_1,\cdots,y_n)$

$A_0\langle\text{op}\rangle B:\bar{x}\prod_{i=1}^{i\leqslant n}x_i\prod_{j=1}^{j\leqslant n}y_j\wedge g(y_1,\cdots,y_n)\wedge f(x_1,\cdots,x_n)$

假设 B 中存在一个变量 y_m（$1\leqslant m\leqslant n$），当 $\bar{x}=0$ 时，$y_m=0$。对于任意 $y_m=1$ 的分支，$A_0\langle\text{op}\rangle B$ 计算后的分支为 $\bar{x}\cdot y_m\prod_{i=1}^{i\leqslant n}x_i\prod_{j=1}^{j<n\&j\neq m}y_j$，由于该分支中 $\bar{x}=0$ 和 $y_m=1$ 相互冲突，因此该分支函数

$$\bar{x}\cdot y_m\prod_{i=1}^{i\leqslant n}x_i\prod_{j=1}^{j<n\&j\neq m}y_j\wedge g(y_1,\cdots,y_n)\wedge f(x_1,\cdots,x_n)=0$$

则：

$$\begin{aligned}&\bar{x}\prod_{i=1}^{i\leqslant n}x_i\prod_{j=1}^{j\leqslant n}y_j\wedge g(y_1,\cdots,y_n)\wedge f(x_1,\cdots,x_n)\\&=\bar{x}\cdot y_m\prod_{i=1}^{i\leqslant n}x_i\prod_{j=1}^{j<n\&j\neq m}y_j\wedge g(y_1,\cdots,y_n)\wedge f(x_1,\cdots,x_n)+\\&\quad\bar{x}\cdot\bar{y}_m\prod_{i=1}^{i\leqslant n}x_i\prod_{j=1}^{j<n\&j\neq m}y_j\wedge g(y_1,\cdots,y_n)\wedge f(x_1,\cdots,x_n)\\&=\bar{x}\cdot\bar{y}_m\prod_{i=1}^{i\leqslant n}x_i\prod_{j=1}^{j<n\&j\neq m}y_j\wedge g(y_1,\cdots,y_n)\wedge f(x_1,\cdots,x_n)\end{aligned}$$

即 $A_0\langle\text{op}\rangle B=A_0\langle\text{op}\rangle B_{|x=0}$。这说明使用规则 1 对原公式进行无效节点删除后的结果与原结果一致。同理，其他两个规则对原结果也无影响。

例 2　基于图 1 中的故障树，在上节中已有的排序队列：$A<(A|B)<(A<B)$ 的基础上，采用基于 SBDD 生成算法的动态优化生成算法进行 SBDD 建立的过程如下。

第 1 步　A 和 $A<B$ 进行或操作。

$$\begin{aligned}&\text{ite}(A,1,0)\vee\text{ite}(A|B,1,0)\\&\quad=\text{ite}(A,1\vee\text{ite}(A|B,1,0)_{|A=1},0\vee\text{ite}(A|B,1,0)_{|A=0})\\&\quad=\text{ite}(A,1,0)\end{aligned}$$

第 2 步　$A|B$ 和第一步的结果进行或操作。

$$\begin{aligned}&\text{ite}(A,1,0)\vee\text{ite}(A<B,1,0)\\&\quad=\text{ite}(A,1\vee\text{ite}(A<B,1,0)_{|A=1},0\vee\text{ite}(A<B,1,0)_{|A=0})\\&\quad=\text{ite}(A,1,0)\end{aligned}$$

采用动态优化生成算法建立的 SBDD 如图 3 所示。

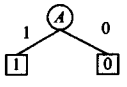

图 3　修改后的 SBDD

由图 2 求出的不交化割集为：A，$\bar{A}\cdot A<B$ 和 $\bar{A}\cdot\overline{A<B}\cdot A|B$，其中有效割集为 A，其他均为无效割集。由图 3 求出的不交化割集为 A，与图 2 的有效割集相同。

3.3　基于动态故障树的 SBDD 方法的实现步骤

基于动态故障树的 SBDD 方法用于分析基于 Pandora 的动态故障树，具体分为 4 步：替换动态故障树中的动态门，使用关系排序算法对动态故障树中的底事件排序，使用动态优化生成算法构建 SBDD 和计算动态故障树的不交化割集。

1）替换动态门

将动态故障树中的每个动态门替换为相应的割序集。例如，在图 1 中底事件 A，B 和与它们相连的 PAND 门转为割序集：$A<B$，表示 A 在 B 之前发生的情况。

2）底事件排序

底事件排序使用本文提出的关系式排序方法。该方法主要根据各底事件在动态故障树中的位置、出现次数进行排序以及根据底事件间的关系进行排序。其重要伪代码如下所示。

```
SeqFTA() //计算叶节点顺序的算法
Begin
  Input 故障树头节点
  //**第一次遍历故障树，计算叶节点数量**//
  while (队列中不为空) do
    q:=队列中首节点
    if (q 为叶节点)
      将 q 计数，加入 record_Num[]中
    else
      将 q 的子节点加入队列
    Endif
  Endwhile
//**第二次遍历故障树，计算叶节点的排序队列**//
  while (该层不为空) do
    for 该层所有元素
      if (该元素是叶子节点)
      if (该元素是静态变量)
        该元素按照出现次数从大到小的顺序插入该层排序队列
        else
          按照门的类型顺序插入该层排序队列
      Endif
    Endif
Endfor
    该层排序队列插入总的排序队列中
    读入故障树下一层
Endwhile
  Return 各叶节点的排序队列
End
```

代码主要由两部分组成：第一次遍历过程和第二次遍历过程。第一次遍历过程是通过队列存储节点的形式层序遍历该故障树，并将各门的数量记录到 *record_Num*[]中。第二次遍历过程是通过队列辅助进行的层序遍历。为每一层的节点建立一个排序队列。在该层访问结束后，该层排序队列按照排序规则插入全局排序队列中。

3）构建 SBDD

构建 SBDD 使用的是动态优化 SBDD 生成算法。基于

SBDD 生成算法的动态优化生成算法主要是根据改进后的计算公式不断地迭代动态故障树进行 SBDD 树的建立。使用动态优化生成算法的流程主要包括 3 部分：对动态故障树的循环遍历、对 $A\langle op\rangle B$ 的运算和对运算前式子的化简。在算法开始时，输入动态门被替换掉的动态故障树和一个排序队列。在程序中，自顶向下使用扩展 SBDD 算法对各层节点进行运算。在对两个子式 A 和 B 进行计算时，首先判断 A 和 B 中首个底事件的排序情况，若 A 的底事件在 B 的底事件之前，则采用扩展的 Shannon 公式中 $x<y$ 的情况进行计算。在计算 $A_1\langle op\rangle B_{|x=1}$ 和 $A_0\langle op\rangle B_{|x=0}$ 的过程中，使用计算规则对 B 进行无效节点的判断和删除。若 A 的底事件在 B 的底事件之后，计算过程同上。由于扩展 SBDD 算法的香农公式主要是同层的子式进行计算，因此流程中当顶事件的兄弟节点都被遍历计算完毕时，该动态故障树的 SBDD 树建立完成。该算法的流程图如图 4 所示。

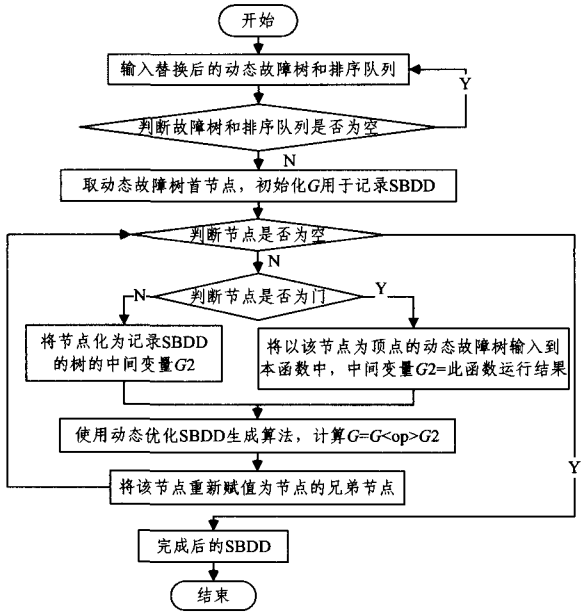

图 4 改进的 SBDD 生成算法流程图

4)计算不交化割集

在 SBDD 中，不交化割集表示为从根节点到 1 节点路径的集合。每个节点的左右分支分别表示该节点上事件发生与不发生两种情况。在计算过程中，每条路径表示为路径经历的每个节点上事件发生或不发生情况的集合。例如，图 3 中 SBDD 的一个不交化割集为 A。

4 实验分析

为了对比 SBDD 方法和基于动态故障树的 SBDD 方法的优劣性，本文将两种方法分别用 C 语言编程实现[11]，比较其分析随机树时生成的 SBDD 的节点数目和时间。文献[12]提出了随机树生成算法[12]。随机树生成算法是自顶向下地生成随机树，包括随机生成门节点和有重复事件的叶节点序列两部分。由于本文需要生成基于 Pandora 的故障树，因此在原随机树生成算法中加入了新的门节点，包括 PAND 门、POR 门和 SAND 门。对生成的随机树分别使用 SBDD 方法和基于动态故障树的 SBDD 方法进行计算，得到生成 SBDD 的总节点数和不交化割集数目。总节点数目的对比结果如图 5 所示，不交化割集数目的对比结果如图 6 所示，运行时间的对比结果如图 7 所示。

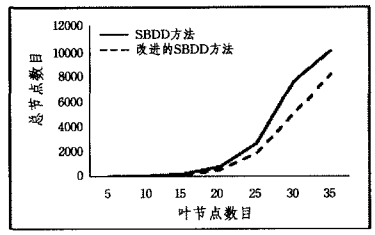

图 5 总结点数目对比图

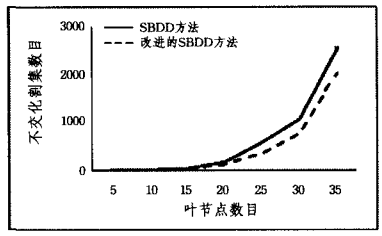

图 6 不交化割集数目对比图

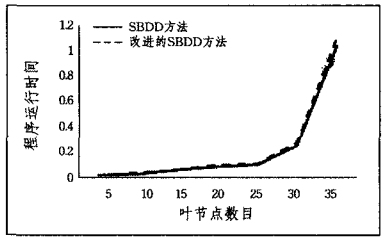

图 7 运行时间对比图

由图 5—图 7 可得出结论：基于动态故障树的 SBDD 方法和 SBDD 方法在运算时间上非常接近。但是在总节点数目和不交化割集数目上，基于动态故障树的 SBDD 方法明显小于 SBDD 方法，并且不存在无效割集。

结束语 本文提出的基于动态故障树的 SBDD 方法能够有效地将基于 Pandora 的动态故障树转化为 SBDD。该方法通过动态地删除无效节点，消除了无效分支。该方法生成的 SBDD 中的节点数目更少，且更加准确，不包含无效的不交化割集。其存在的问题是：针对 Pandora 动态故障树的排序方法主要考虑静态变量的位置关系和静态变量与动态变量的依赖关系，未考虑相同门之间的动态变量的排序关系。因此，若要完善改进的 SBDD 方法需要对故障树模式与 BDD 的内在关系有更深入、透彻的理解，提出考虑更加全面的 Pandora 动态故障树的排序方法。

参 考 文 献

[1] WALKER M D. Pandora: a logic for the qualitative analysis of temporal fault trees[D]. The University of Hull, 2009.

[2] WALKER M, PAPADOPOULOS Y. Synthesis and analysis of temporal fault trees with PANDORA: The time of Priority AND gates[J]. Nonlinear Analysis: Hybrid Systems, 2008, 2(2): 368-382.

[3] GE D, LIN M, YANG Y, et al. Quantitative analysis of dynamic fault trees using improved Sequential Binary Decision Diagrams [J]. Reliability Engineering & System Safety, 2015, 142: 289-299.

[4] RUIJTERS E, STOELINGA M. Fault tree analysis: A survey of

the state-of-the-art in modeling, analysis and tools[J]. Computer Science Review, 2015, 15-16(3): 29-62.

[5] FU G Z, HUANG Z H, LI H Q, et al. Fault tree analysis on kinematic accuracy of wafer stage using BDD and DFTA technique [C]// 2013 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE). IEEE, 2013: 260-262.

[6] GE D, LI D, CHOU Q, et al. Quantification of highly coupled dynamic fault tree using IRVPM and SBDD[J]. Quality and Reliability Engineering International, 2016, 32(1): 139-151.

[7] ZHU P, HAN J, LIU L, et al. A stochastic approach for the analysis of fault trees with priority AND gates[J]. IEEE Transactions on Reliability, 2014, 63(2): 480-494.

[8] LIU H. On variable ordering heuristics of BDD-based fault tree analysis[D]. Hangzhou: Zhejiang Normal University, 2012. (in Chinese)
刘华. 基于 BDD 故障树分析的启发式变量排序研究[D]. 杭州：浙江师范大学，2012.

[9] SUN Y, DU S G. A Novel Ordering Method of Binary Decision Diagram [J]. Journal of Systems and Management, 2008(2): 210-216, 220. (in Chinese)
孙艳，杜素果. 一种二元决策图底事件排序的新方法[J]. 系统管理报，2008(2)：210-216，220.

[10] DUAN S, ZHANG X R, LIU S K, et al. Transformation method of fault tree based on BDD[J]. Computer Engineering and Application, 2009, 45(21): 51-54. (in Chinese)
段珊，张修如，刘树锟，等. 一种故障树向 BDD 的转化方法[J]. 计算机工程与应用，2009，45(21)：51-54.

[11] GAO S C. Methods and Implementation of Dynamic Fault Tree Analysis[D]. Changsha: National University of Defense Technolog, 2005. (in Chinese)
高顺川. 动态故障树分析方法及其实现[D]. 长沙：国防科学技术大学，2005.

[12] MO Y C, YANG Q S. Random generation and variable ordering of fault tree[J]. Journal of Zhejiang University(Engineering Science), 2011, 45(9): 1539-1543. (in Chinese)
莫毓昌，杨全胜. 故障树随机生成及变量排序[J]. 浙江大学学报(工学版)，2011，45(9)：1539-1543.

(上接第 194 页)

分割后，减少了对模态窗口打开事件和模态窗口关闭事件的关注程度所导致。

Q3：在进行局部测试时，模型分割前的多级形态模型只能整体进行形态扩展，因此生成的测试用例与完整测试系统所有 GUI 事件是完全一样的；而模型分割后，可以针对性地对每一个受测模态窗口的模型单独进行形态扩展，并生成测试用例。从表 3 可以看出，模型分割后，测试用例数量和测试用例长度都得到了极大的缩减。图 7 给出了实验组二经模型分割后测试用例的各项数值相对分割前的比例，随着模型级数的增大，模型分割对测试用例数量、长度的缩减程度增大，当 $k=4$ 时，经模型分割后，测试占 GUI 事件总量 30%的一半，模态窗口所需的测试用例总数仅为原模型测试用例总数的 21%，测试用例总长度缩减了 85%，测试用例平均长度缩减了 29%。

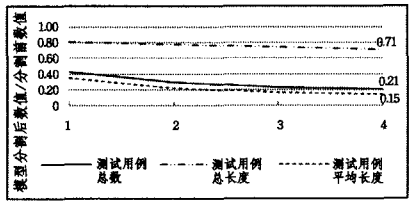

图 7　实验组二模型分割后测试用例的各项数值的占比

Q4：由表 5 可以看出，由于植入的错误均在受测的模态窗口内，模型分割并未对错误检测效力造成影响。对于受测部分而言，经模型分割后的多级形态模型可以实现与未分割模型等同的测试覆盖。

结束语　多级形态模型可以实现测试资源与错误检测效力的合理均衡，并让测试人员可以从不同的形态角度考察 GUI。经过模型分割后的多级形态模型可以实现对 GUI 重点部分的侧重测试，在缩短测试用例长度、减少测试用例数量的同时，进一步提高多级形态模型的可控性与灵活性，显著提升了测试效率。

参 考 文 献

[1] MEMON A M, BAO N N. Advances in Automated Model-Based System Testing of Software Applications with a GUI Front-End [J]. Advances in Computers, 2010, 80(10): 121-162.

[2] MYERS B, HOLLAN J, CRUZ I, et al. Strategic directions in human-computer interaction[J]. ACM Computing Surveys, 1996, 28(4): 794-809.

[3] ARLT S, PAHL S, SCHÖF M, et al. Trends in Model-based GUI Testing[J]. Advances in Computers, 2012, 86: 183-222.

[4] MEMON A M, POLLACK M E, SOFFA M L. Hierarchical GUI test case generation using automated planning[J]. IEEE Transactions on Software Engineering, 2001, 27(2): 144-155.

[5] BELLI F. Finite state testing and analysis of graphical user interfaces[C]//12th International Symposium on Software Reliability Engineering, 2001(ISSRE 2001). IEEE, 2001: 34-43.

[6] BELLI F, BEYAZIT M. Exploiting Model Morphology for Event-Based Testing[J]. IEEE Transactions on Software Engineering, 2015, 41(2): 113-134.

[7] MEMON A M. GUI Testing: Pitfalls and Process[J]. Computer, 2002, 35(8): 87-88.

[8] MEMON A M. An event-flow model of GUI-based applications for testing [J]. Software Testing Verification & Reliability, 2007, 17(3): 137-157.

[9] AHO P, SUAREZ M, MEMON A, et al. Making GUI Testing Practical: Bridging the Gaps[C]// International Conference on Information Technology-New Generations. IEEE, 2015: 439-444.

[10] THIMBLEBY H. The directed Chinese Postman Problem[J]. Software Practice & Experience, 2003, 33(11): 1081-1096.

[11] YUAN X, MEMON A M. Generating Event Sequence-Based Test Cases Using GUI Runtime State Feedback [J]. IEEE Transactions on Software Engineering, 2010, 36(1): 81-95.