基于模型的图形用户界面事件交互图生成方法

丰 凯 高建华
(上海师范大学计算机科学与技术系 上海 200234)

摘 要 用户界面(GUI)测试是一项很困难的工作,一个重要的原因是背景事件会影响测试结果,基于模型的图形用户界面测试技术可以解决这个问题。目前基于模型的图形用户界面测试技术有两种常用的GUI模型:事件流图(EFG)和事件交互图(EIG)。这两种模型可以表示GUI事件之间的交互关系,其中EIG是从EFG转换而来的。通过一个简单的GUI实例对GUI中的事件进行了明确的划分,并且为适应文中的划分事件改进了原有的MX算法。最后根据GUI的事件驱动性和对GUI事件的划分提出了一种由EFG转换成EIG的新方法:驱动算法。实例表明,此方法使EFG转换成EIG的过程更简单。

关键词 图形用户界面,基于模型测试,事件流图,事件交互图
中图法分类号 TP311 **文献标识码** A

GUI Event Interaction Graph Generation Method Based on Model

FENG Kai GAO Jian-hua
(Department of Computer Science and Technology, Shanghai Normal University, Shanghai 200234, China)

Abstract The test of GUI is a veritable challenge. One of the concernful reasons is that there are many context events which would cause determinate influences over the results of testing. This problem can be solved by the technologies of model-based GUI testing. Two of the frequently used graphical user interface(GUI) model in model-based GUI testing technologies are the event flow graph(EFG) and the event interaction graph(EIG). Both of the two kinds of models give a certain presentation of the events interacting relationships, and the EIG makes significant improvements form the EFG. Our paper provides the GUI events with a specific division through a simple GUI instance and also improves the MX algorithm for the sake of adaptation to the new division of the events. Finally, according to the property of event-driven in GUI and the divided GUI events in this paper, the paper proposes a new method: driven algorithm. Some examples show that the method makes the EFG converted into EIG more easier.

Keywords Graphical user interface(GUI), Model-based testing, Event flow graph(EFG), Event interaction graph(EIG)

1 引言

图形用户界面(Graphical User Interfaces, GUIs)是当今软件最重要的组成部分之一,软件开发人员几乎花费60%的代码用于执行GUIs[1]。而GUIs是对部件集合的设计,其中的部件用于回应单个事件。所以GUI测试意味着一个基于GUI软件应用程序的测试,完全是由执行在GUI部件上的序列事件决定的,而且软件的正确性是通过检测GUI部件决定的。

GUI的一个重要特征的是,它的行为和行为的使用背景是紧密相关的。当用户在GUI部件(如:文本框,"确定"按钮)上调用事件序列(如:对文本框写入值,点击"确定"按钮)时有可能会改变软件的状态,影响后续事件的执行。正如Mathur所说[2],在一个软件系统中,事件和状态之间是紧密联系的。对背景敏感和对状态敏感的GUI事件给测试带来很多问题,每个事件需要在多个背景下进行测试。在实践中,有些GUI测试设计者运用捕获/回放工具(Capture/Replay Tools)为图形用户界面创建测试用例[3],由于这种工具提供很少的自动化测试,测试用例的产生是劳动密集型的,因此经常会导致测试不充分。

基于模型的GUI测试技术可以解决这个问题,主要的测试技术包括:随机方法[4]、限制事件序列长度的方法[2,5]和综合考虑事件序列长度及位置的方法[6]。这些方法更多地考虑到了测试事件的背景,也逐渐开发出了事件流图(Event Flow Graph, EFG)和事件交互图(Event Interaction Graph, EIG)。对已经开发出的GUI模型可以使用图的遍历技术用于自动执行,而且EFG可以从正在执行的GUI中使用GUI抓取工具(GUI Ripper Tools)和逆向工程技术自动获取,EFG图中的事件序列代表了所有可能在GUI上执行的事件序列[5]。由于EFG包含所有可能的GUI事件,比较复杂,因此可通过改进EFG得到EIG模型[7],而由EIG生成的测试用例的有效性已经得到验证[8]。但是在基于模型的GUI测试技术中存在着一些缺陷,如GUI事件划分不够详细[7,8],没有明确的EFG到EIG的转换规则,而且EFG生成EIG算法复杂且包

本文受国家自然科学基金项目(61073163),上海市科委项目(09220503000),上海市引进技术的吸收与创新计划(2010CH-014)资助。
丰 凯(1986-),男,硕士生,主要研究方向为软件可靠性设计理论与方法、软件测试技术,E-mail:fjjk88@163.com;高建华(1963-),男,博士,教授,主要研究方向为软件可靠性理论与设计、软件开发环境与开发技术、数据安全与计算机安全、网络测试、LSI/VLSI测试等。

含事件类型不够全面。针对这些不足，本文做出了一些改进并提出了一个新的由 EFG 转换成 EIG 的算法：

(1)更全面详细地划分了 GUI 中的事件；

(2)明确了 EFG 转换成 EIG 的准则并改进了 MX 算法[7]；

(3)提出了驱动算法。

本文第 1 节介绍了 GUI 中基于模型测试技术的应用，并指出当今技术中存在的缺陷，列出本文所做的工作；第 2 节对 GUI 中的事件进行详细分类，并介绍了 GUI 模型；第 3 节改进了当前从 EFG 到 EIG 的转换准则及算法，并提出一种新转换的准则和算法；最后是对本文的总结及未来工作的描述。

2 GUI 事件分类和 GUI 模型

在 Microsoft Office Word 2003 中，“编辑”下拉菜单中，有“剪切”、“复制”、“粘贴”、“定位”、“查找”等菜单项；“工具”下拉菜单中的“语言”菜单项中有“中文简繁转换”子菜单。为方便处理 GUI 事件和有效地分析 EFG 和 EIG，在此把“中文简繁转换”加入“编辑”菜单中，并简化“编辑”菜单项、“定位”弹出框和“中文简繁转换”弹出框，如图 1 所示。

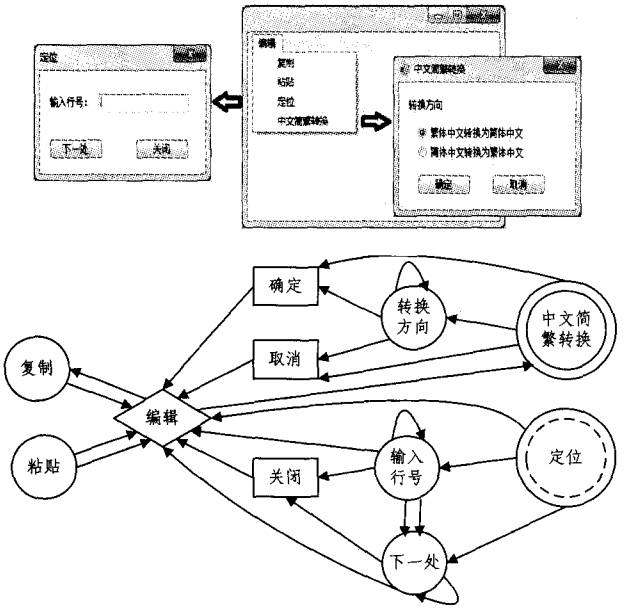

图 1 一个简单的图形用户界面及其事件流图

2.1 GUI 事件分类

GUI 中存在大量的事件，下面以图 1 为例对 GUI 中的事件进行系统的分类，以便更好地从 EFG 转换到 EIG。

(1)可达事件(reachability events)用于打开窗口或菜单，用 R 表示。其中打开菜单事件(open-menus events)用 ROM 表示，在模型图中用菱形表示；打开窗口事件(open-windows events)可分为打开模态窗口(modal windows)的限制性聚焦事件(restricted-focus events)和打开非模态窗口(modeless windows)的非限制性聚焦事件(unrestricted-focus events)，分别用 RF 和 RUF 表示，在模型图中分别用双圈和内圈为虚线的双圈表示，ROM={编辑}；RF={中文简繁转换}；RUF={定位}。

(2)终止事件(termination events)是除可达事件外由部件操作 GUI 结构的事件，即关闭窗口，用 T 表示，在模型图中用矩形框表示，T={关闭；确定；取消}。

(3)系统交互事件(system-interaction events)是除可达事件外不操作 GUI 结构的事件，用 S 表示，在模型中用圆表示，S={复制；粘贴；输入行号；下一处；转换方向}。

为方便记录事件，本文中把模态窗口中的系统交互事件记为 SM，终止事件记为 TM；非模态窗口中的系统交互事件记为 SUM，终止事件记为 TUM。

需要注意的是：打开模态窗口后必须退出才能对其它事件进行操作，而非模态窗口打开后不关闭也可以对其它事件进行操作。

2.2 GUI 模型

2.2.1 事件流图(EFG)

事件流图(EFG)是模拟 GUI 的方法之一，它模拟窗口(或部件)内的事件及事件间的相互作用关系。在 EFG 中模拟所有可能发生在 GUI 中的事件序列，所以在 EFG 中包含节点(nodes)和边(edges)，其中，节点代表 GUI 中的事件，边代表 GUI 中各事件间的相互作用关系，从节点 n_x 到节点 n_y 的边表示执行 n_x 代表的事件后可以立即执行 n_y 代表的事件，这个关系叫做跟踪关系。EFG 可用两个集合表示：(1)节点集合 N，表示 GUI 中的事件；(2)边的集合 E，即有序对 (n_x,n_y) 的集合，其中 $n_x \subseteq N, n_y \subseteq N$，表示 EFG 中的有向边，如果 n_y 跟踪 n_x，则 $(n_x,n_y) \in E$。所以，EFG 可表示为一个二元组 (N,E)。

2.2.2 事件交互图(EIG)

Brooks 等人[9]通过 GUI 测试指出上千种错误，其中出现在应用程序的底层业务逻辑所占比重很大，而 GUI 代码本身的错误很少。因为打开菜单和窗口的代码不可能与其他事件代码进行交互，所以与这些事件进行交互测试发现的错误会很少。在 GUI 中由于大量的事件不和 GUI 应用程序的业务逻辑交互，例如打开菜单和窗口事件，而 EFG 中的事件代表在 GUI 中所有可能执行的事件，所以在 EFG 中去掉那些不和 GUI 业务逻辑交互的事件，得到细化的 EFG 称为事件交互图(EIG)[8]。最后得出，EIG 中的事件是 GUI 事件中的可达事件(T)和系统交互事件(S)(去掉了可达事件(R))，然后根据事件的驱动性对这些事件进行交互得到 EIG。

2.2.3 模型图相关定义

由以上对 EFG 和 EIG 的介绍可知，在可执行的事件序列和模型图的路径之间存在相关对应关系。从直观上可看出，一个事件流路径代表一个可以在 GUI 上执行序列的事件序列。在形式上，EFG 和 EIG 中的事件流路径的相关定义如下。

定义 1 在 EFG 和 EIG 中没有孤立的节点：如果 $(n_x,n_j) \in E, (n_{j+k},n_y) \in E, \{(n_{j+i},n_{j+i+1}),0 \leqslant i \leqslant (k-1)\} \in E$，且存在一个节点序列(可能为空)$n_j;n_{j+1};n_{j+2};\cdots;n_{j+k}$，其中 x,y,i,j,k 都是正整数，则从节点 n_x 到节点 n_y 存在一条事件流路径。

由定义 1 可知，各节点间可以直接或间接地连接起来。EFG 和 EIG 的边和节点〈$n_1;n_2;\cdots;n_k$〉用于事件流路径，根据二者的关系可得到一个有序对的集合。在 EFG 和 EIG 中每一对节点代表一个边。如图 1 中存在的两个事件流路径〈复制；编辑；中文简繁转换；取消〉和〈简体中文转换为繁体中文；确定；编辑；复制；编辑；粘贴〉，则从“复制”到“粘贴”存在事件流路径〈复制；编辑；复制；编辑；粘贴〉。

定义 2 如果在 EFG 和 EIG 中存在节点 $n_1;n_2;\cdots;n_k$ 且节点 $n_2;\cdots;n_k$ 中没有模态窗口中的终止事件 T，则事件流路径$\langle n_1;n_2;\cdots;n_k\rangle$可以自由交互，交互时前后顺序不变。

例如图 1 中路径〈复制；编辑；中文简繁转换；取消〉是可以自由交互的，因为“编辑”和“中文简繁转换”不是终止事件(T)。而路径〈简体中文转换为繁体中文；确定；编辑；复制；编辑；粘贴〉并不能自由交互，如〈简体中文转换为繁体中文；编辑〉不成立，因为“确定”是模态窗口中的一个终止事件。

需要注意的是：模型图中的事件可以与自身交互，而且事件交互是有方向性的，即“e_x 与 e_y 交互”并不等于“e_y 与 e_x 交互”，如图 4 中的事件交互{下一处；转换方向}，反过来则不成立。EIG 中的节点代表系统交互事件(S)或终止事件(T)。从节点 n_x(代表 e_x)到节点 n_y(表示 e_y)意味着 n_x 与 e_y 交互。图 1 中的 EFG 生成的 EIG 如图 4 所示。从图 4 可知，EIG 中的节点比 EFG 中的节点少，但是边可能比 EFG 中的多。

3 EFG 转换成 EIG 的算法

文献[7]中 Memon 和 Xie Qing 提出一种 EFG 转换成 EIG 的算法，本文称为“MX 算法”。下面为适应本文的 GUI 事件划分对 MX 算法进行了改进，并根据 GUI 中事件的驱动性和本文对 GUI 事件的划分提出了一种新的转换算法：驱动算法。

3.1 MX 算法及其改进

文献[7]把 GUI 中的事件分为可达事件和系统交互事件，所有 EIG 中的事件类型定义为系统交互事件，所以它所对应的 EFG 转换成 EIG 的算法如下所示，其中第 8′行是为适应本文中 GUI 事件类型的划分对原算法的第 8 行做出的改进。

MX 算法：

```
/ * N Nodes set of EIG * /
/ * E Edges set of EIG * /
1. PROCEDURE::GenerateEIG(Event Flow Graph(N,E)) {
2.   N=N
3.   E=E
4.   FORALL n∈N DO
5.     start(n)={n_i|(n,n_i)∈E,and n≠n_i}
6.     end(n)={n_i|(n_i,n)∈E,and n≠n_i}
7.   FORALL n∈N DO
8.     IF EventType(n)≠system-interaction
8'.    IF EventType(n)=reachability
9.       FORALL n_x∈end(n) DO
10.        FORALL n_y∈start(n) DO
11.          E=E∪(n_x,n_y)
12.          IF n_x≠n_y
13.            start(n_x)=start(n_x)∪{n_y}
14.            end(n_y)=end(n_y)∪{n_x}
15.      FORALL n_x∈end(n) DO
16.        remove n form start(n_x)
17.      FORALL n_y∈start(n) DO
18.        remove n form end(n_x)
19.      remove n from N
20.      remove all edges(n,n_i) from E
21.      remove all edges(n_i,n) from E
}
```

改进后的 EIG 转换算法是：第 1 行生成 EIG 事件；第 2 行和第 3 行初始化 EIG，把 EFG 中的节点集 N 和边集 E 都赋予 EIG；第 4 行到第 6 行根据边中节点的顺序把所有节点分成开始节点集 $start(n)$和结束节点集 $end(n)$两类；第 7 行到第 14 行是遍历所有的节点，如果节点 n 不是系统交互事件，就把边(n_x,n)和边(n,n_y)转换成边(n_x,n_y)，其中节点 n 代表可达事件 R；若 $n_x\neq n_y$，把 n_x 加到结束节点集 $end(n)$，把 n_y 加到开始节点集 $start(n)$，因为在 EIG 中这些事件可以与自身交互。第 15 行到 18 行是从开始节点集 $start(n)$和结束节点集 $end(n)$中去掉节点 n。第 19 行到每 21 行是去掉 EFG 生成 EIG 时不要的节点和边，即节点 n、边(n_x,n)和边(n,n_y)。从图 1 到图 4 可看到具体的转换过程。

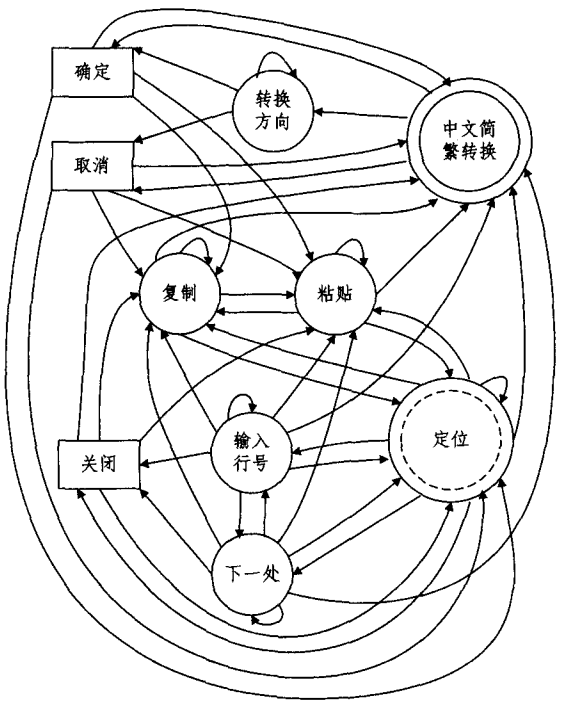

图 2 事件流图去掉“编辑”

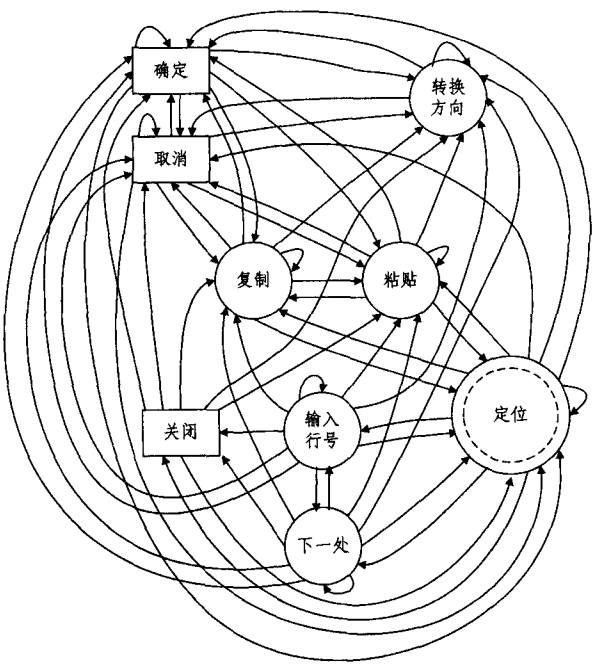

图 3 事件流图去掉“编辑”和“中文简繁转换”

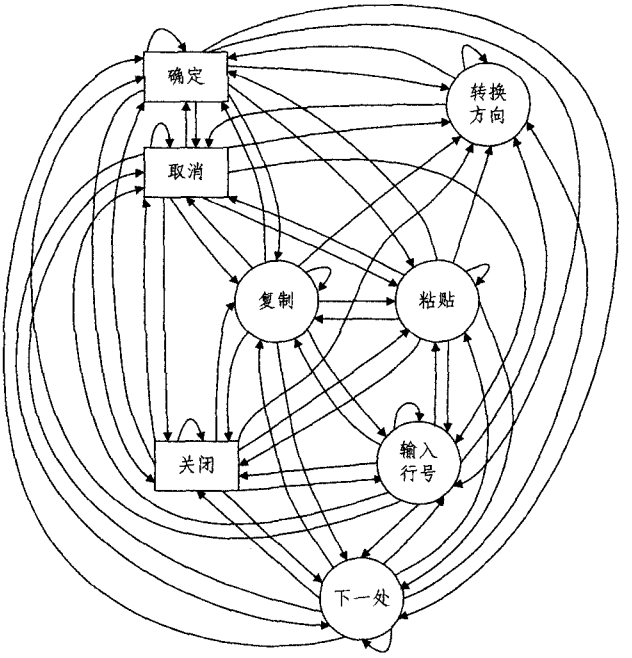

图 4　事件交互图

本文中划分 GUI 的事件比文献[7,8]更详细，分成可达性事件(R_{OM}；R_F 和 R_{UF})、终止性事件(T)和系统交互事件(S)，所以在 EIG 中不仅包含系统交互事件(S)，还包含终止性事件(T)。为适应本文的 GUI 事件划分，把上述算法的第 8 行代码修改成第 8′行，便可得到对应的 EFG 转换成 EIG 算法。

MX 算法对应的 EIG 转换准则如下：

(1)依次去掉 EFG 中的可达事件(R)，即去掉打开按钮事件(R_{OM})，如图 2 所示；去掉限制聚焦事件(R_F)，如图 3 所示；去掉非限制聚焦事件(R_{UF})，如图 4 所示；

(2)在去掉各个可达事件时，对于去掉的边(n_x，R)和边(R，n_y)，重新生成边(n_x，n_y)(x，y 可以重复)；

(3)对(2)中的事件 n_y 存储映射(R，n_y)，用于在测试阶段执行[4,6]；这是因为使用模型图代表 GUI 的基本动机是运用图遍历算法进行“走”图，即沿着 EIG 的节点和边进行枚举事件。在走图时产生的测试用例称为烟雾测试(Smoke Test)[7]，如在图 4 中长度为 2 的两个烟雾测试用例：〈复制；剪切〉和〈剪切；粘贴〉。由于 EIG 的节点不代表可达事件(如“编辑”事件)，在测试时，这些事件(这里是“编辑”事件)要到达 EIG 中的事件，需要在早期存储映射：{剪切→(编辑；剪切)，粘贴→(编辑；粘贴)，复制→(编辑；复制)}来自动生成测试用例，即在使用 EIG 中的事件时自动产生一个包含可达事件的可执行测试用例。因此，这两个烟雾测试用例将“扩展”成〈编辑；复制；编辑；剪切〉和〈编辑；剪切；编辑；复制〉。所以，从图 1 中的 EFG 中移除“编辑”得到图 2 后，有 4 个映射：(编辑；复制)、(编辑；粘贴)、(编辑；中文简繁转换)和(编辑；定位)；从图 2 中移除“中文简繁转换”得到图 3 后，有 3 个映射：(中文简繁转换；转换方向)、(中文简繁转换；确定)、(中文简繁转换；取消)；从图 3 中移除“定位”得到图 4 后，有 9 个映射：(定位；输入)、(定位；下一处)、(定位；关闭)、(定位；语言)、(定位；转换方向)、(定位；确定)、(定位；取消)、(定位；复制)、(定位；粘贴)。

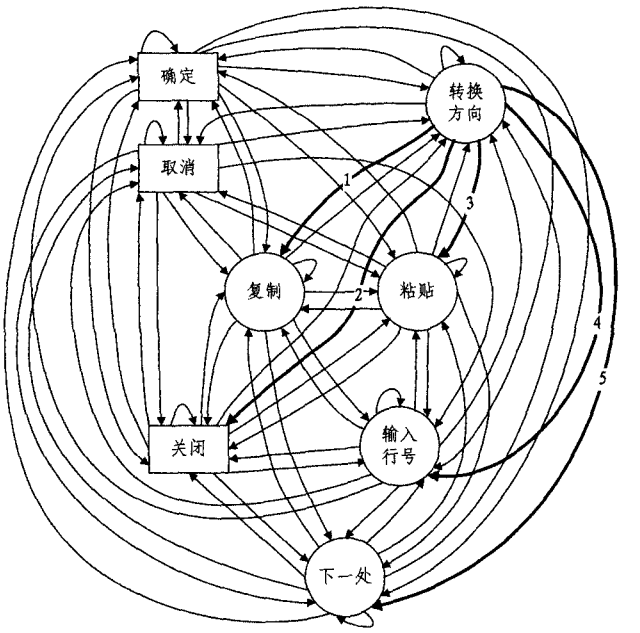

图 5　驱动算法：从事件流图到事件交互图的中间转换步骤

3.2　驱动算法

由 GUI 中的事件驱动性及 EFG 生成 EIG 的过程可知，在图 1 中是菜单、模态窗口和非模态窗口之间的交互并且三者之间可以自由交互，模态窗口和非模态窗口是由菜单中的部件打开的。菜单事件和非模态窗口中的事件是开放性的，即在它们中的事件可以无限制地进行自由交互。而模态窗口中的事件是限制性的，即它由终止事件与菜单和非模态窗口进行自由交互，而在模态窗口本身中的事件可以自由交互。

根据 GUI 中的事件驱动性和 EIG 的转换过程，本文提出一种新的 EFG 转换成 EIG 的算法：

驱动算法：

```
  /* N Nodes set of EIG */
  /* E Edges set of EIG */
1. PROCEDURE::GenerateEIG(Event Flow Graph(N,E)) {
2.   N=N
3.   E=E
4.   FORALL n∈N DO
5.     IF system-interaction events in modal windows
6.       EventType(n)=S_M
7.     IF termination events in modal windows
8.       EventType(n)= T_M
9.     FORALL e∈E DO
10.      remove all edges e from E
11.    FORALL n∈N DO
12.      IF EventType(n)=reachability
13.      remove n from N
14.    FORALL n_x∈N,n_y∈N,DO
15.      E=E∪(n_x,n_y)
16.        IF EventType(n_x)= S_M AND(EventType(n_y)≠S_M OR
             EventType(n_x)≠S_T)
17.          remove all edges(n_x,n_y) from E
  }
```

在驱动算法中：第 1 行生成 EIG 事件；第 2 行和第 3 行初始化 EIG，把 EFG 中的节点集 N 和边集 E 都赋予 EIG；第 4

(下转第 203 页)

Conference on Computer Vision. 2007
[7] 程有龙,李斌,张文聪,等,融合先验知识的自适应行人跟踪算法[J]. 模式识别与人工智能,2009,22(5):704-708
[8] 李娟,邵春福,杨励雅,等. 基于 Kalman 滤波的行人跟踪方法研究[J]. 交通运输系统工程与信息,2009,9(6):148-153
[9] Felzenszwalb P F, Girshick R B, McAllester D. Discriminatively trained deformable part models [C]// IEEE Conf. on Computer Vision and Pattern Recognition. 2009
[10] Viola P, Jones M J, Snow D. Detecting Pedestrians Using Patterns of Motion and Appearance [J]. International Journal of Computer Vision, 2005, 63(2): 153-161
[11] Pavan M, Pelillo M, Dominant sets and pairwise clustering [J]. PAMI, 2007, 29(1): 167-172
[12] Ge J, Luo Y, Tei G. Real-time pedestrian detection and tracking at nighttime for driver-assistance systems [J]. Intelligent Transportation Systems, IEEE Transactions on, 2009, 10(2): 283-298
[13] Geismann P, Schneider G. A two-staged approach to vision-based pedestrian recognition using Haar and HOG features [C]// Intelligent Vehicles Symposium, 2008 IEEE. IEEE, 2008: 554-559

(上接第 187 页)

行到第 8 行中遍历所有节点 N,定义在模态窗口内的系统交互事件(S)和终止事件(T)分别为 S_M 和 S_T;第 9 行和第 10 行是删除 EFG 中所有的边;第 11 行到第 13 行是遍历所有的节点 N,去掉其中所有的可达事件(R)。第 14 行到第 17 行遍历所有的节点并令其自由交互生成边,去掉边(n_x,n_y),如果 n_x 不是 S_M 且 n_y 不是 S_M 或 T_M。由于事件的驱动性,模态窗口中的事件只能和本窗口内的事件交互,因此模态窗口内的非终止事件与另一窗口或菜单内的事件交互是错误的,需要移除这些事件序列。从图 1 到图 5 再到图 4 可以看到具体的转换过程。

驱动算法对应的 EIG 转换准则如下:

(1)依次去掉 EFG 中的可达事件(R)和所有的边(n_x,n_y);

(2)对剩余的节点任取两个进行有序排列,生成边(n_x,n_y)(x,y 可以相等),如图 5 所示;

(3)去掉边(n_x,n_y),如果 $n_x=S_M$,$n_y\neq S_M$ 或 $n_y\neq T_M$,即去掉图 5 中有标号的 5 条边得到图 4;

(4)对(1)中去掉的边(R,n_y)存储成映射(R,n_y),用于在测试阶段执行;共有 16 个映射:(编辑;复制)、(编辑;粘贴)、(编辑;中文简繁转换)、(编辑;定位)、(中文简繁转换;转换方向)、(中文简繁转换;确定)、(中文简繁转换;取消)、(定位;输入)、(定位;下一处)、(定位;关闭)、(定位;语言)、(定位;转换方向)、(定位;确定)、(定位;取消)、(定位;复制)、(定位;粘贴)。

3.3 MX 算法与驱动算法的比较

MX 算法和驱动算法都是由 EFG 转换成 EIG。但是,驱动算法的优越性体现在两个方面:

(1)对于 EFG 转换成 EIG 过程中 GUI 的事件划分,驱动算法比 MX 算法更详细。因为本文对 GUI 中的事件进行了更加全面详细的划分,在驱动算法中使用了本文的 GUI 事件划分方式。

(2)对于 EFG 转换成 EIG 的过程,驱动算法比 MX 算法更简单。从实例的转换过程图可看到,即从图 1 到图 4,在 MX 算法中去掉可达事件和与其相连的边后,重新连接各个节点、生成新边的过程很繁杂,容易出错和遗漏。而在驱动算法中就没有这一繁杂的过程,这个算法中去掉了 EIG 中不具有的可达事件和所有的边,然后对其中的所有事件进行两两排列配对(也可以与自身配对),最后移除其中不正确的边。由实例可知,对所有事件进行简单的排列配对(如图 5)比逐个去掉可达事件及对与可达事件相连的事件根据跟踪关系重新配对(如图 2、图 3)要简单很多。

结束语 在基于模型的 GUI 测试技术中,EIG 是一种简单、高效的 GUI 模型。在 EFG 转换成 EIG 时,本文对 GUI 中的事件进行了更加全面详细的划分,并改进了原有的 MX 算法,提出了驱动算法,使得 EFG 生成 EIG 时更加简单。但是由于实验的局限性和为了有效地分析 GUI 模型,本文只对一个菜单下的事件情况进行了研究,有待于进一步提高。在未来的工作中我们会考虑不同菜单事件下的交互情况,以及在生成 GUI 测试用例时可能不需要表述所有的 GUI 事件关系,而是去掉中间的 EFG,由 GUI 直接生成 EIG,以便直接生成有效的测试用例。

参 考 文 献

[1] Memon A M. GUI Testing: Pitfalls and Process [J]. IEEE Computer, 2002, 8(35): 90-91
[2] Mathur A P. Foundations of Software Testing: Fundamental Algorithms and Techniques[M]. New Delhi: Pearson Education in South Asia, 2008: 33-384
[3] Memon A M. Advances in Computers[M]. Amsterdam, Netherlands: Elsevier Ltd, 2003: 150-203
[4] Memon A M, Xie Qing. Using Transient/ Persistent Errors to Develop Automated Test Oracles for Event-Driven Software[A]// Proceeding of 19th IEEE International Conference on Automated Software Engineering IEEE INFOCOM[C]. 2004: 186-195
[5] Memon A M. A Comprehensive Framework for Testing Graphical User Interfaces[D]. Pittsburgh: Department of Computer Science, University of Pittsburgh, 2001
[6] Yuan Xun, Myra B C, Memon A M. GUI Interaction Testing: Incorporating Event Context [J]. IEEE Transactions on Software Engineering, 2011, 37(4): 559-574
[7] Memon A M, Xie Qing. Studying the Fault- Detection Effectiveness of GUI Test Cases for Rapidly Evolving Software[J]. IEEE Transactions on Software Engineering, 2005, 31(10): 884-896
[8] Xie Qing, Memon A M. Using a Pilot Study to Derive a GUI Model for Automated Testing[J]. ACM Transactions on Software Engineering And Methodology, 2008, 18(2): 1-35
[9] Brooks P A, Robinson B P, Memon A M. An Initial Characterization of Industrial Graphical User Interface Systems[A]// Proceeding of First IEEE International Conference on Software Testing Verification and Validation IEEE INFOCOM [C]. Denver, CO: IEEE, 2009: 11-20