Hierarchical GUI Test Case Generation Using Automated Planning

Christian Halaschek-Wiener

Motivation

- GUI testing inherently hard
  - Many items to test
    - Test underlying software
    - Must conform with GUI specs
  - Huge space of interactions with GUI
    - Large coverage area for test cases
Motivation

- Test Case Generation
  - Manual creation time consuming
- Present technique to automatically generate test cases
Approach Overview

- GUI consists of components that generate events
  - Lead to other GUIs or actions
- Given a GUI, its actions, and a goal
  - AI Planner generates paths through the system that achieve the goal
    - Each path is a test case
AI Planning Overview

- **Input**
  - Set of objects
  - Set of operators
  - Initial state
  - Goal state
- **Objects** are GUI components/events
- **Operators** are the composition/actions of using them
AI Planning Overview

- **Output**
  - Set of plan steps (operators)
  - Ordering constraints on steps
  - Causal links
    - Structure of plan
  - Set of binding constraints on variables of operators

- **Ordering constraints**
  - State $S_i$ occurs before $S_k$

- **Causal links**
  - Two states with a post-condition of $S_i$ that is a pre-condition of $S_k$
AI Planning Overview

- Results in partially-ordered plans
- Can use/add ordering constraints to create total-order plans
Hierarchical Task Network (HTN) Planning

- Actions are modeled at different levels of abstraction
  - Operators at level $n$
    - Use one or more methods at level $n-1$
- GUI planning can be modeled as HTN planning problem
  - Provides greater planning efficiency
  - Modifications to GUI only require modifications to sub-plans
## Steps

<table>
<thead>
<tr>
<th>Phase</th>
<th>Step</th>
<th>Test Designer</th>
<th>PATHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>1</td>
<td></td>
<td>Derive Hierarchical GUI Operators</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Define Preconditions and Effects of Operators</td>
<td></td>
</tr>
<tr>
<td>Plan Generation</td>
<td>3</td>
<td>Identify a Task $\mathcal{T}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td>Generate Test Cases for $\mathcal{T}$</td>
</tr>
</tbody>
</table>
GUI Operators

- First construct operator set
  - Made up of GUI events
  - Note, one operator for each GUI event is inefficient
    - Exploit structural properties of GUI to create operators at different levels
  - Partition events into classes based on structural properties
GUI Events

- Different classes of events
  - Menu-open
    - Expand GUI events
      - e.g. File, Edit…
  - Unrestricted-focus
    - Open GUI window, not restricting focus
      - e.g. PPT basic shapes
  - System-interaction
    - Underlying software function
  - Restricted-focus
    - Preferences dialog
Operator Classes

- Two classes of planning operators
  - System-interaction
    - All sequences of zero or more menu-open and unrestricted events followed by a system interaction event
  - Example
    - Edit (menu-open) select, Paste (sys-int)
      - Results in operator EDIT_PASTE = <Edit, Paste>
  - Use an operator event mapping
    - Reduces number of operators made available to the planner
Operator Classes

- Abstract Operator
  - Restricted-focus events
  - Encapsulate events by creating new planning problem
    - Layer of abstraction
  - Essentially is sub-planning problem
    - Prefix - sequence of menu-open and unrestricted-focus that lead to this event
    - Suffix - the restricted-focus user interaction
      - Planner determines the suffix
  - Both are substituted back into the plan
Abstract Operator
Operator-Event Mappings

\textbf{GUI Events} = \{ File, Edit, New, Open, Save, SaveAs, Cut, Copy, Paste, Open.Up, Open.Select, Open.Cancel, Open.Open, SaveAs.Up, SaveAs.Select, SaveAs.Cancel, SaveAs.Save \}.

(a)


(b)

<table>
<thead>
<tr>
<th>Operator Name</th>
<th>Operator Type</th>
<th>GUI Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE.NEW</td>
<td>Sys. Interaction Abstract</td>
<td>&lt;File, New&gt;</td>
</tr>
<tr>
<td>FILE_OPEN</td>
<td>Sys. Interaction Abstract</td>
<td>&lt;File, Open&gt;</td>
</tr>
<tr>
<td>FILE_SAVE</td>
<td>Sys. Interaction Abstract</td>
<td>&lt;File, Save&gt;</td>
</tr>
<tr>
<td>FILE_SAVEAS</td>
<td>Sys. Interaction Abstract</td>
<td>&lt;File, SaveAs&gt;</td>
</tr>
<tr>
<td>EDIT_CUT</td>
<td>Sys. Interaction Abstract</td>
<td>&lt;Edit, Cut&gt;</td>
</tr>
<tr>
<td>EDIT_COPY</td>
<td>Sys. Interaction Abstract</td>
<td>&lt;Edit, Copy&gt;</td>
</tr>
<tr>
<td>EDIT_PASTE</td>
<td>Sys. Interaction Abstract</td>
<td>&lt;Edit, Paste&gt;</td>
</tr>
</tbody>
</table>
Abstraction Advantages

- Wordpad
  - 10:1 reduction ration in operator number
  - 325 GUI events
  - Reduced to 32 system interaction and abstract operators
Preconditions and Effects

- Test designer must next specify the preconditions and effects for each operator
- Definition language provided by the planner

\[
\text{Operator} :: \text{EDIT\_CUT}
\]

**Preconditions:**

\[
\text{EXISTS} \text{ Obj in Objects}
\]

\[
\text{Selected(Obj)}.
\]

**Effects:**

\[
\text{FORALL} \text{ Obj in Objects}
\]

\[
\text{Selected(Obj) } \Rightarrow
\]

\[
\text{ADD inClipboard(Obj)}
\]

\[
\text{DEL onScreen(Obj)}
\]

\[
\text{DEL Selected(Obj)}.
\]
Identify Planner Task

- Consists of initial and goal states

Initial State:
- isCurrent(root)
- contains(root private)
- contains(private Figures)
- contains(private Latex)
- contains(private Samples)
- contains(private Courses)
- contains(private Thesis)
- contains(root public)
- contains(public html)
- contains(gif)
- containsfile(gif doc2.doc)
- containsfile(private Document.doc)
- containsfile(Samples report.doc)
- currentFont(Times Normal 12pt)
- in(doc2.doc This)
- in(doc2.doc is)
- in(doc2.doc the)
- in(doc2.doc text.)
- isText(This)
- isText(is)
- isText(the)
- isText(text)
- after(This is)
- after(is the)
- after(final text.)

Goal State:
- containsfile(public new.doc)
- in(new.doc This)
- in(new.doc is)
- in(new.doc the)
- in(new.doc final)
- in(new.doc text.)
- after(This is)
- after(is the)
- after(final text.)

----------
Similar descriptions for Document.doc and report.doc

----------
Big Picture
## Resulting Plans

<table>
<thead>
<tr>
<th>Plan No.</th>
<th>Plan Step</th>
<th>Plan Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>FILE-OPEN(&quot;private&quot;, &quot;Document.doc&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;that&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;must&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;be&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;modified&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-IN-TEXT(&quot;final&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILE-SAVEAS(&quot;public&quot;, &quot;new.doc&quot;)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>FILE-OPEN(&quot;public&quot;, &quot;doc2.doc&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-IN-TEXT(&quot;is&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-IN-TEXT(&quot;the&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;needs&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;to&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;be&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;modified&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-IN-TEXT(&quot;final&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-IN-TEXT(&quot;text&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILE-SAVEAS(&quot;public&quot;, &quot;new.doc&quot;)</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>FILE-OPEN(&quot;public&quot;, &quot;doc2.doc&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-IN-TEXT(&quot;is&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-IN-TEXT(&quot;the&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;to&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;be&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE-TEXT(&quot;modified&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-IN-TEXT(&quot;final&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-IN-TEXT(&quot;text&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SELECT-TEXT(&quot;needs&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EDIT-CUT(&quot;needs&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILE-SAVEAS(&quot;public&quot;, &quot;new.doc&quot;)</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>FILE-NEW(&quot;public&quot;, &quot;new.doc&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-TEXT(&quot;This&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-TEXT(&quot;is&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-TEXT(&quot;the&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-TEXT(&quot;final&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE-TEXT(&quot;text&quot;, Times, Italics, 12pt)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FILE-SAVEAS(&quot;public&quot;, &quot;new.doc&quot;)</td>
</tr>
</tbody>
</table>
Results

- Time to generate plans

<table>
<thead>
<tr>
<th>Task No.</th>
<th>Plan Time (sec)</th>
<th>Sub Plan Time</th>
<th>Total Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.40</td>
<td>0.04</td>
<td>0.44</td>
</tr>
<tr>
<td>2</td>
<td>3.16</td>
<td>0.00</td>
<td>3.16</td>
</tr>
<tr>
<td>3</td>
<td>3.17</td>
<td>0.00</td>
<td>3.17</td>
</tr>
<tr>
<td>4</td>
<td>3.20</td>
<td>0.01</td>
<td>3.21</td>
</tr>
<tr>
<td>5</td>
<td>3.38</td>
<td>0.01</td>
<td>3.39</td>
</tr>
<tr>
<td>6</td>
<td>3.44</td>
<td>0.02</td>
<td>3.46</td>
</tr>
<tr>
<td>7</td>
<td>4.09</td>
<td>0.04</td>
<td>4.13</td>
</tr>
<tr>
<td>8</td>
<td>8.88</td>
<td>0.02</td>
<td>8.90</td>
</tr>
<tr>
<td>9</td>
<td>40.47</td>
<td>0.04</td>
<td>40.51</td>
</tr>
</tbody>
</table>
Performance Results

- Single level vs. Hierarchical

<table>
<thead>
<tr>
<th>Task No.</th>
<th>Plan Length</th>
<th>Time (sec.)</th>
<th>Plan Length</th>
<th>Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18</td>
<td>8.93</td>
<td>3</td>
<td>0.11</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>47.62</td>
<td>4</td>
<td>0.18</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>189.87</td>
<td>5</td>
<td>0.14</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
<td>3312.72</td>
<td>6</td>
<td>7.18</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>0.1</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>13.01</td>
</tr>
</tbody>
</table>

- Due to reduction in computation space because of abstraction - 10:1 reduction
Results

- Also note that they implemented an automated test execution system
  - Generate mouse/keyboard events
Limitations

- Test cases generated depend on tasks defined by user
  - Poor tasks...poor coverage
    - Developing coverage measures

- Depend heavily on hierarchical structure of GUI
  - If poorly structured, no abstract operators

- Should be used in conjunction with other generators
Critique

- Frequency that operator abstraction can actually be used
- User involvement
  - State commonly used operators could be maintained in libraries
Related Work

- **Record/Playback tools**
  - Sessions are played back to recreate GUI steps

- **FSM**
  - Used for test case generation
  - Once built, generation process automatic
  - Scaling problems

- **Genetic Algorithms**
  - Expert generates initial GUI events
  - GA extends and modifies them
  - Assumption - Expert takes short path, where novice will take longer ones
Questions & Comments