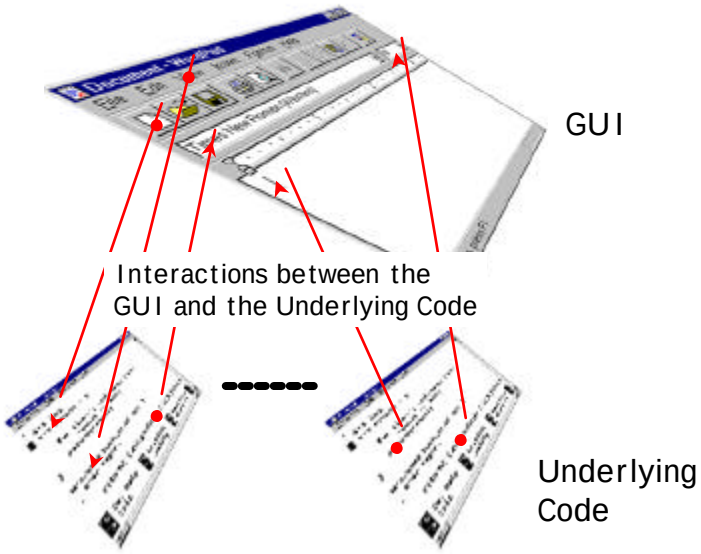
1

Plan Generation for GUI Testing

- *The 21st International Conference on Software Engineering*
- *The Fifth International Conference on Artificial Intelligence Planning and Scheduling*
- *IEEE Transactions on Software Engineering*

2

Research Focus



Why Planning for GUI Testing

- GUIs are Event Driven
- Individual User Events
  - NOT ENOUGH !
  - Sequences of User Events lead to Different States
- Test Case: Sequence of User Events
- How to Generate Test Cases ?
- Use Planning to Select Likely Test Cases

Selecting Test Sequences

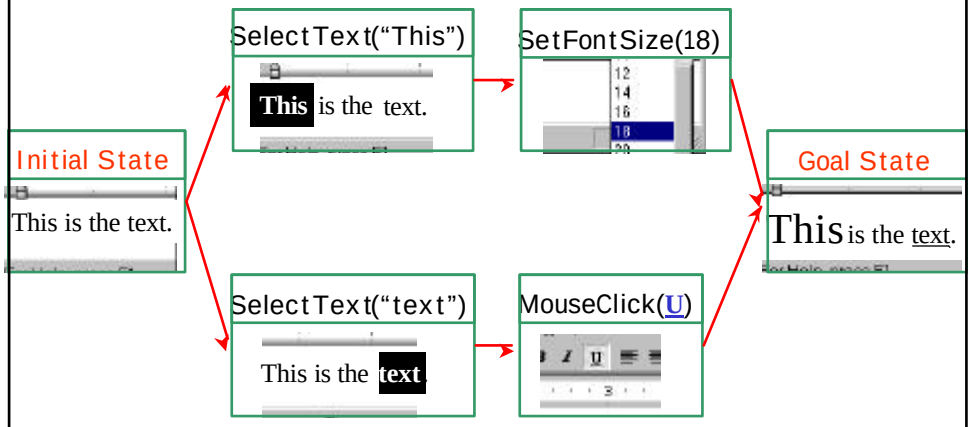
- Infinitely Many
- Randomly Choose Sequences
- Expert Chooses Sequences
- Automatically Generate Events for COMMONLY USED TASKS

Initial State

Goal State

5

A Plan for a GUI Task



6

Outline

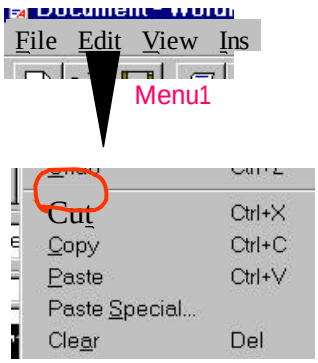
- Using Planning for Test Case Generation
  - Overall Approach
  - Exploiting GUI Structure
  - Generating Alternative Test Cases
- Experimental Results
- Related Research
- Concluding Remarks

Overview of Test Generation

Phase	Step	Test Designer	Automatic Planning-based System
Setup	1		Derive Planning Operators from GUI
	2	Code Preconditions and Effects of Operators	
Test Case Generation	3	Specify a Task (Initial and Goal States)	
	4		Generate Test Cases

Straightforward Approach

- Define One Operator for each User Action



Operator :: *CUT*
Preconditions:
isCurrent(Menu2).

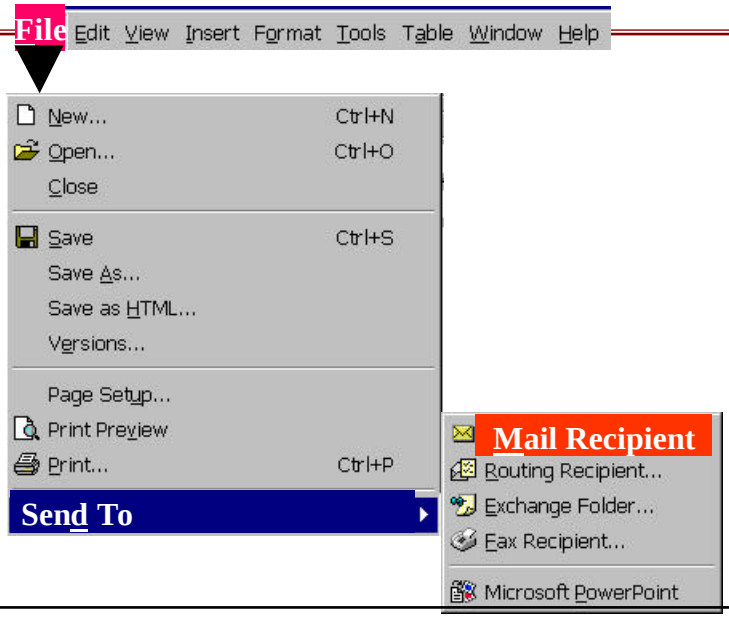
Effects:
FORALL Obj in Objects
Selected(Obj) ?
ADD inClipboard(Obj)
DEL onScreen(Obj)
DEL Selected(Obj)
ADD isCurrent(Menu1)
DEL isCurrent(Menu2).

Exploit the GUI's Structure

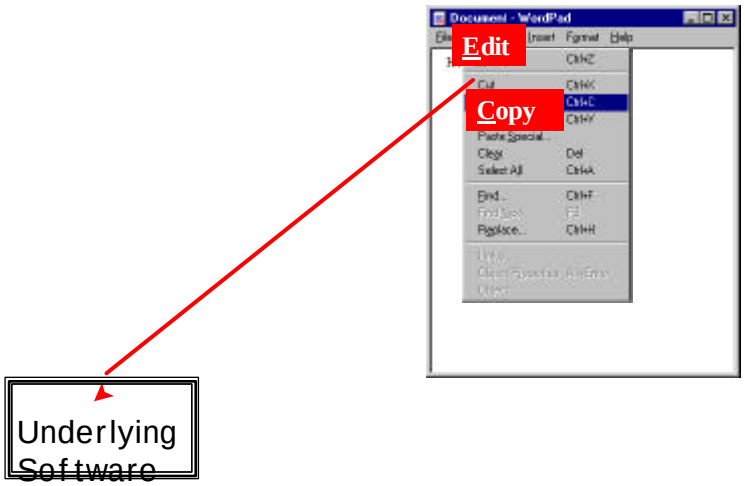
- Reduce the Number of Operators
  - System more Efficient
  - Easier for the Test Designer

Opening Modal Windows

Opening Menus



Interacting with the Underlying Software



13

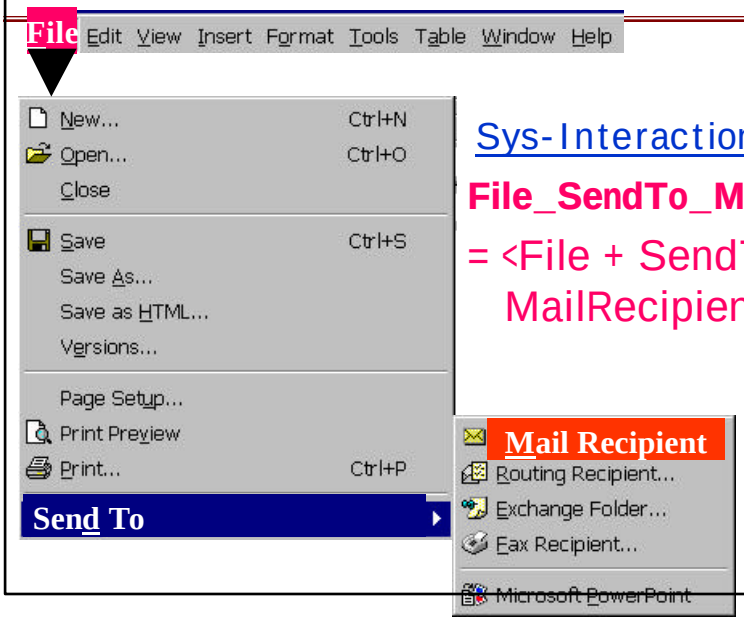
Create Hierarchical Operators

Two Types of Abstractions

- Combine Buttons ? Create System-Interaction Operators
- Decompose GUI Hierarchically ? Create Abstract Operators

14

Create System-Interaction Operators

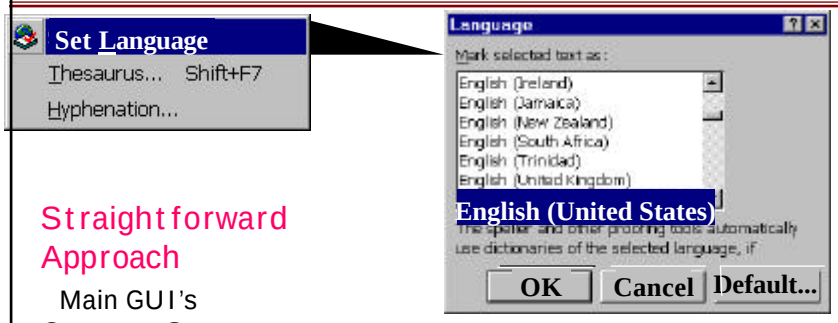


Sys-Interaction Operator:

File_SendTo_MailRecipient

= <File + SendTo + MailRecipient >

Create Abstract Operators



Straightforward Approach

Main GUI's Operator Set

...
Set Language
SelectFromList()
Default
OK
Cancel
...

Using Abstraction

Main GUI's Operator Set

...
Set Language
...

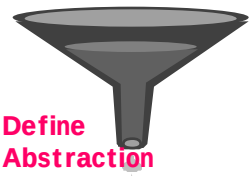
Language Window's Operator Set

SelectFromList()
Default
OK
Cancel

Create Abstract Operators

Language Window's Operator Set

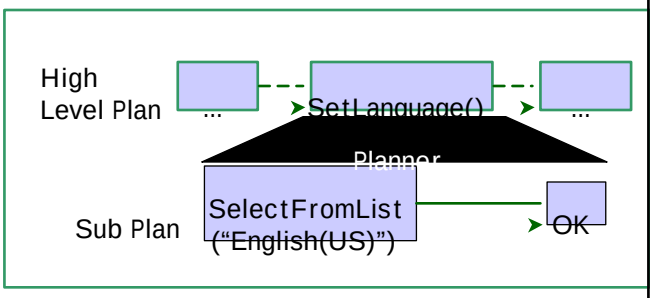
SelectFromList()
Default
OK
Cancel



Define Abstraction

SetLanguage()

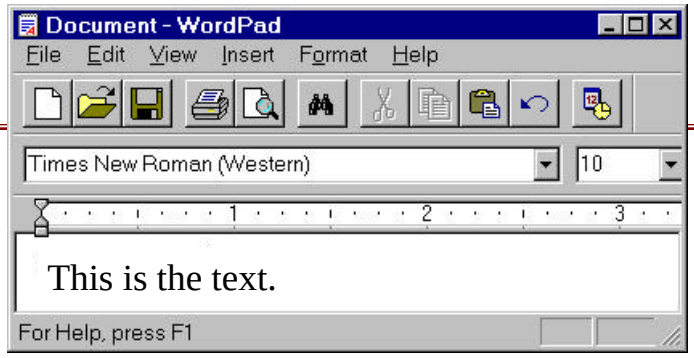
Abstract Operator



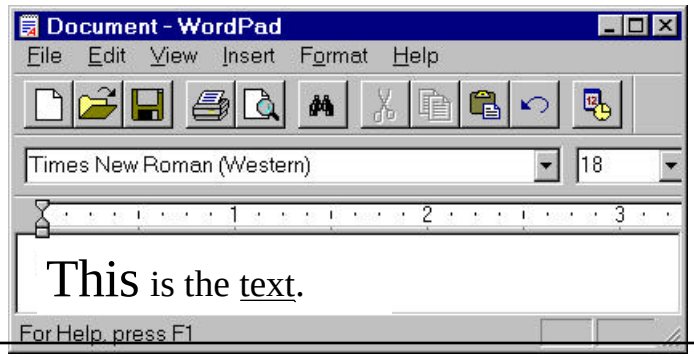
Effects of Exploiting the GUI's Structure

- Reduction in Planning Operators
  - 325 operators ? 32 operators
  - Ratio 10:1 for MS WordPad
  - 20:1 for MS Word
- System Automatically Determines the System-interaction and Abstract Operators

Initial State

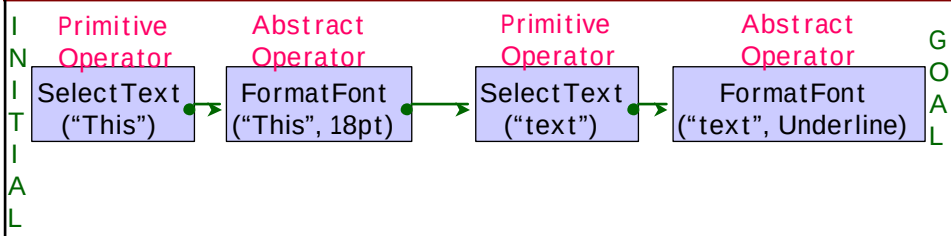


Goal State

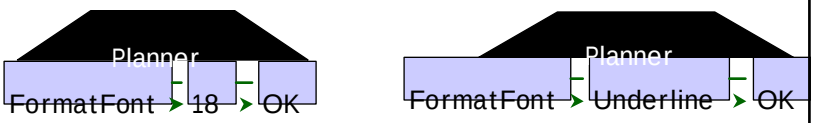


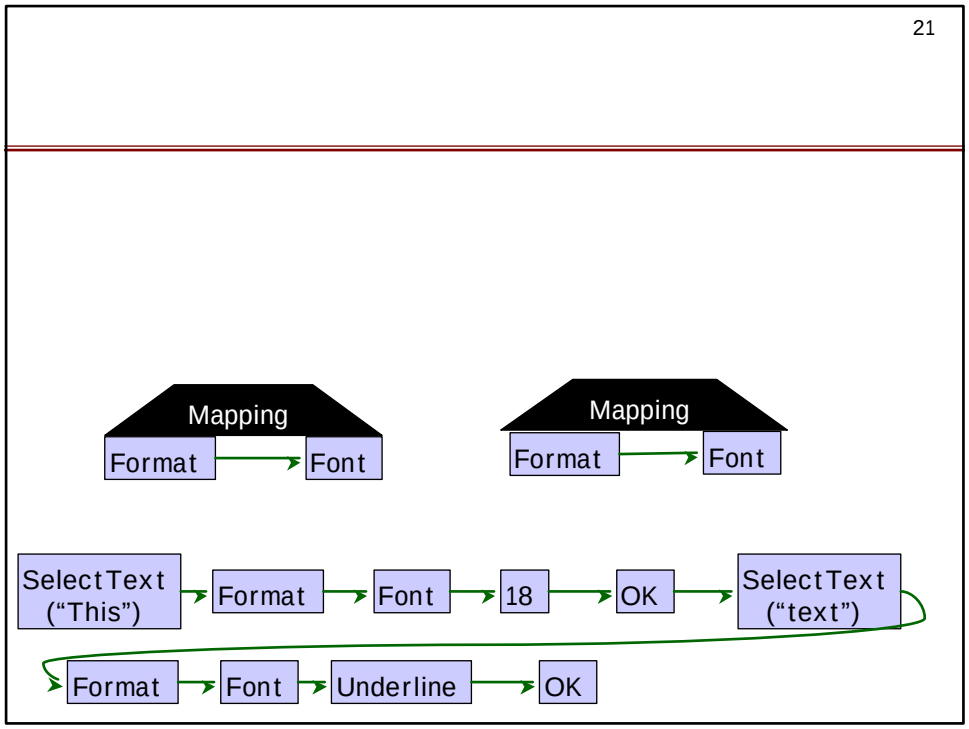
19

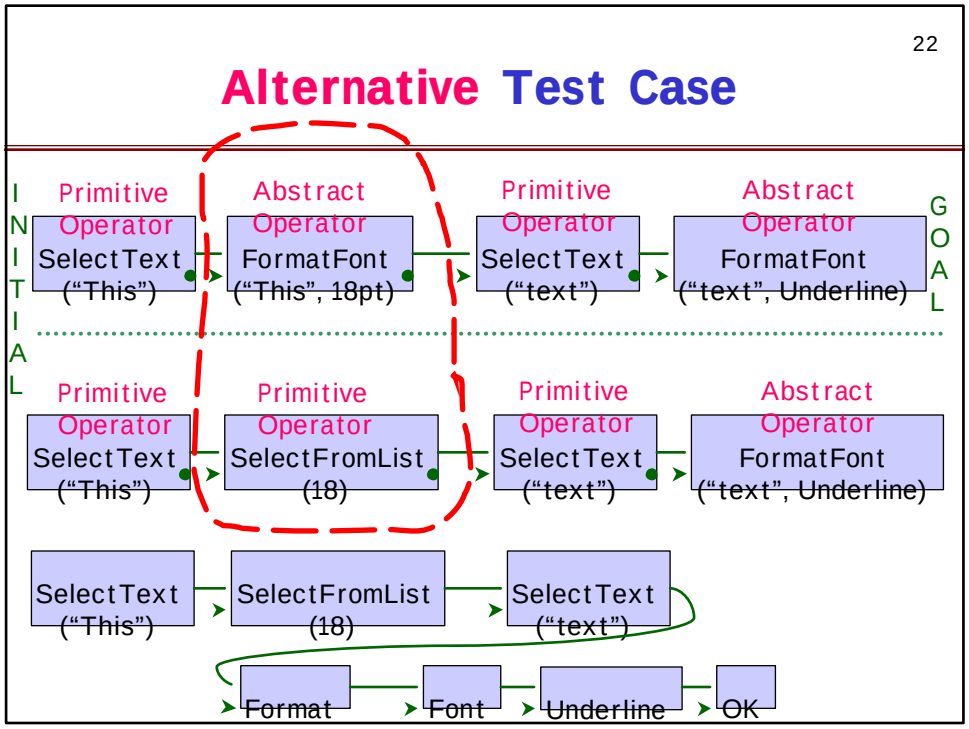
Test Case



20




21
Mapping
Format
Font
Mapping
Format
Font
SelectText ("This")
Format
Font
18
OK
SelectText ("text")
Format
Font
Underline
OK


22
Alternative Test Case
INITIAL
Primitive Operator
SelectText ("This")
Abstract Operator
FormatFont ("This", 18pt)
Primitive Operator
SelectText ("text")
Abstract Operator
FormatFont ("text", Underline)
GOAL
Primitive Operator
SelectText ("This")
Primitive Operator
SelectFromList (18)
Primitive Operator
SelectText ("text")
Abstract Operator
FormatFont ("text", Underline)
SelectText ("This")
SelectFromList (18)
SelectText ("text")
Format
Font
Underline
OK

Methods to Generate Alternative Test Cases

23

- Different Results from Planner
- Abstract Operator Decompositions
- Linearizations of the Partial-order Plan

Feasibility Study

24

- Purpose
  - To Determine whether Planning is a Feasible Approach for GUI Test Case Generation
    - Execution Time
    - Human Effort
- Experimental Design
  - GUI: MS WordPad
  - Planner: IPP [Koehler et al. '97]
  - Hardware Platform: 300 MHz Pentium based Machine, 200 MB RAM, Linux OS
  - 8 Tasks, Multiple Test Cases for each Task

Experimental Results

(Task) Plan No.	Plan Time (sec.)	Sub Plan Time (sec.)	Total Time (sec.)
1	3.16	0	3.16
2	3.17	0	3.17
3	3.2	0.01	3.21
4	3.38	0.01	3.39
5	3.44	0.02	3.46
6	4.09	0.04	4.13
7	8.88	0.02	8.9
8	40.47	0.04	40.51

Concluding Remarks

- Automatic Planning is a Feasible Approach for GUI Test Case Generation
- Automatic Generation of Preconditions and Effects from GUI Specifications
- Generate Expected Output (Automated Verification)