Program Testing and Analysis:

GUI Testing

Dr. Michael Pradel

Software Lab, TU Darmstadt

Warm-up Quiz

What does the following code print?

```
var a = (0.1 + 0.2) + 0.3;
var b = 0.1 + (0.2 + 0.3);
console.log(a === b);
```

true **false** **Something else**

Warm-up Quiz

What does the following code print?

```
var a = (0.1 + 0.2) + 0.3;
var b = 0.1 + (0.2 + 0.3);
console.log(a === b);
```

true **false** **Something else**

Warm-up Quiz

What does the following code print?

```
var a = (0.1 + 0.2) + 0.3;
var b = 0.1 + (0.2 + 0.3);
console.log(a === b);
```

Floating point numbers are represented with finite precision (not only in JavaScript)

true | **false** | **Something else**

Warm-up Quiz

What does the following code print?

```
var a = (0.1 + 0.2) + 0.3;
var b = 0.1 + (0.2 + 0.3);
console.log(a === b);
```

0.30000000000000004
(due to rounding)

true | **false** | **Something else**

GUI Testing

- **Test application via its graphical user interface (GUI)**
- **Possible approaches**
  - Manual testing
  - Semi-automated, e.g., Selenium
- **Here: Automated GUI testing**
  - Purely random testing, e.g., Monkey for Android
  - Today: Sophisticated approaches beyond purely random testing

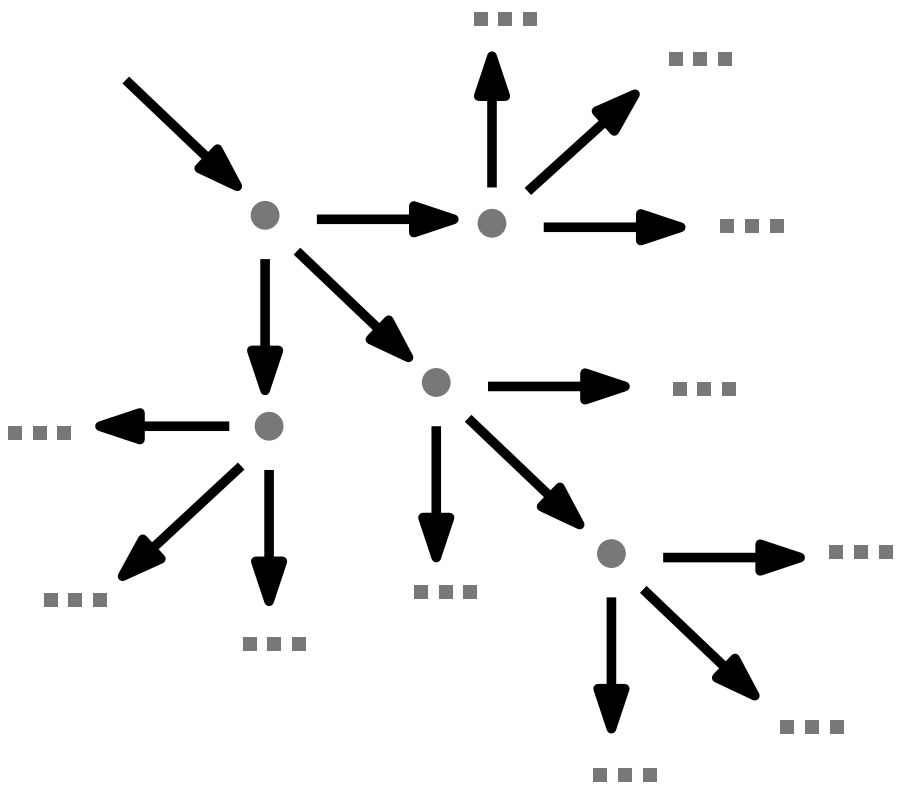
Challenges

- **Two input dimensions**
  - Sequences of events, e.g., clicks, mouse movements (focus of today's lecture)
  - Input values, e.g., strings entered into form
- **Not all events are known ahead of time**
  - Web apps load content dynamically
- **Need a test oracle**
  - When does an execution expose a bug?
- **Huge search space**

Huge Search Space

Challenge: Huge search space

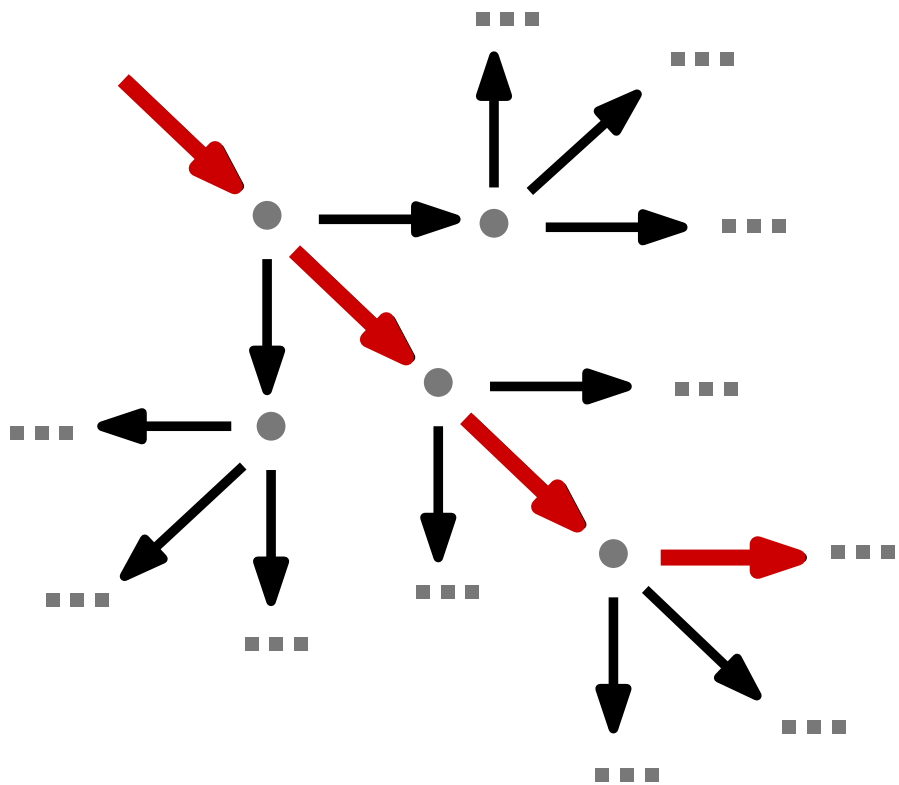
- Too large to explore exhaustively



Huge Search Space

Approach:

Steer search toward potential bugs or not yet explored behavior



Outline

- **Feedback-directed GUI testing**

  Based on *A Framework for Automated Testing of JavaScript Web Applications*, Artzi et al., ICSE 2011

- **Model inference-based testing**

  Based on *Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning* , Choi et al., OOPSLA 2013

- **Responsiveness testing**

  Based on *EventBreak: Analyzing the Responsiveness of User Interfaces through Performance-Guided Test Generation*, Pradel et al., OOPSLA 2014

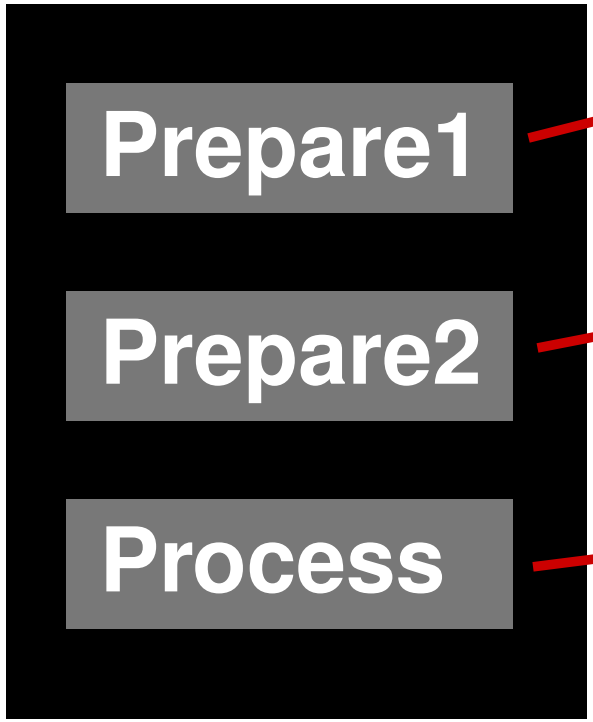
Artemis

Feedback-directed generation of GUI tests

- Start with randomly selected events
- Gather feedback from execution
- Steer toward particularly interesting behavior
- Implemented for web applications
- Test oracle: JavaScript exceptions and invalid HTML

Example

Application with 3 buttons:




```
prepared1 = true;
```

```
prepared2 = true;
```

```
if (prepared1)
  if (prepared2)
    window.alert("hi");
```

Initially, `prepared1=prepared2=false`

Gathering Feedback

Feedback gathered while executing generated sequences of events:

- Available events
- Source code of handlers attached to events
- Memory locations read & written
- Branch coverage

Artemis: Algorithm

- Input: URL u
- Add sequence $[load\ u]$ to worklist
- While worklist not empty
  - Execute next sequence and gather feedback
  - Add new sequences to worklist
    - Modify inputs of existing sequence
    - Extend sequence with additional event
    - Create new sequence with new start URL
  - Prioritize worklist

Prioritization by Coverage

Coverage-guided prioritization

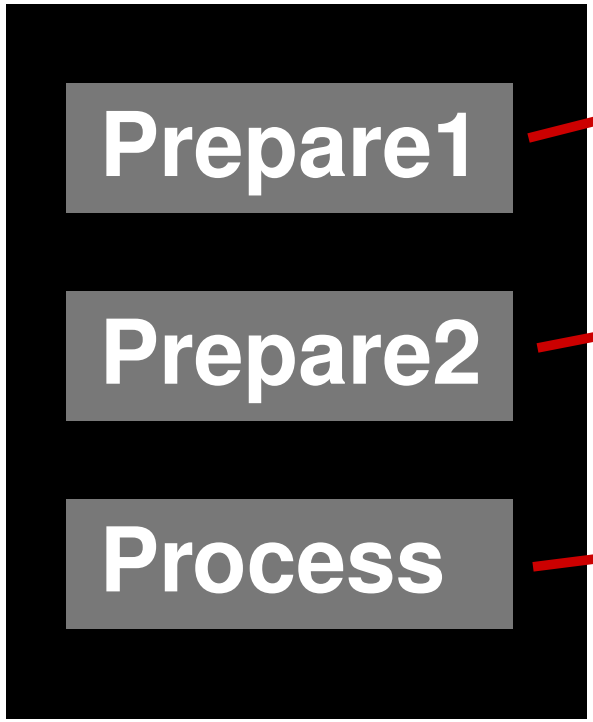
- Keep track of branch points in each handler
  - Branch here means entry of handler or control flow branch
- Prioritize sequences that trigger handlers with low coverage

$$P(e_1, ..., e_k) = 1 - cov(e_1) \cdot ... \cdot cov(e_k)$$

where $cov(e) = \dfrac{\text{covered branches of } e\text{'s handler}}{\text{all discovered branches of } e\text{'s handler}}$

Example

Application with 3 buttons:




```
prepared1 = true;
```

```
prepared2 = true;
```

```
if (prepared1)
  if (prepared2)
    window.alert("hi");
```

Initially, `prepared1=prepared2=false`

Suppose to have executed:

Prepare 2, Process

$$cov(Prep1) = \frac{0}{1} = 0$$

$$cov(Prep2) = \frac{1}{1} = 1$$

$$cov(Process) = \frac{2}{5}$$

Possible next sequences:

- Prepare 2, Process, Prepare 1 $\longrightarrow P = 1 - 1 \cdot \frac{2}{5} \cdot 0 = 1$
- — " — , Prepare 2 $\longrightarrow P = 1 - 1 \cdot \frac{2}{5} \cdot 1 = \frac{3}{5}$
- — " — , Process $\longrightarrow P = 1 - 1 \cdot \frac{2}{5} \cdot \frac{2}{5} = \frac{21}{25}$

highest priority $\rightarrow$ execute next

Priorization by Memory Accesses

Prioritize based on read/write sets

- Keep track of memory locations read/written by each handler
- Prioritize sequences where some handlers write values read by a subsequence handler
- Intuition: Can cover interesting behavior only after some handlers have set the right pre-conditions

Outline

- **Feedback-directed GUI testing**

  Based on *A Framework for Automated Testing of JavaScript Web Applications*, Artzi et al., ICSE 2011

- **Model inference-based testing** ←

  Based on *Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning* , Choi et al., OOPSLA 2013

- **Responsiveness testing**

  Based on *EventBreak: Analyzing the Responsiveness of User Interfaces through Performance-Guided Test Generation*, Pradel et al., OOPSLA 2014

SwiftHand

- **Challenge: Restarting the application is expensive**
- **Learn finite-state model of application while exploring it**
  - Explore states with unknown outgoing transitions
  - Continuously refine model by splitting states
- **Explores application with small number of restarts**

Outline

- **Feedback-directed GUI testing**

  Based on *A Framework for Automated Testing of JavaScript Web Applications*, Artzi et al., ICSE 2011

- **Model inference-based testing**

  Based on *Guided GUI Testing of Android Apps with Minimal Restart and Approximate Learning* , Choi et al., OOPSLA 2013

- **Responsiveness testing**

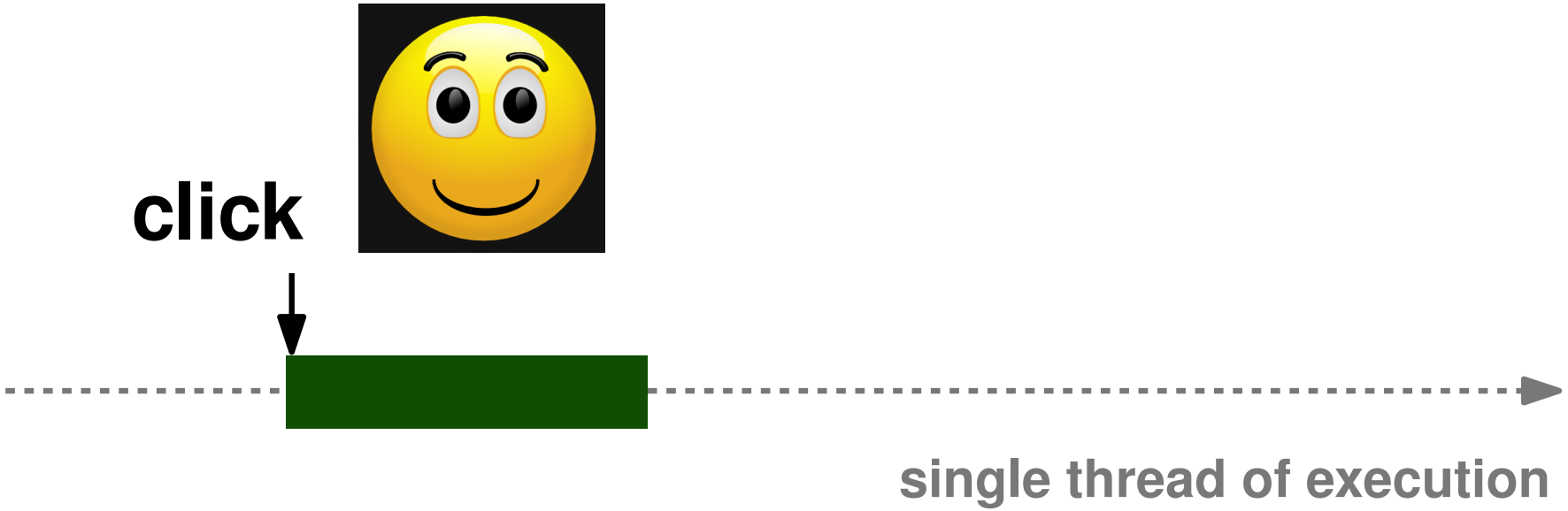
  Based on *EventBreak: Analyzing the Responsiveness of User Interfaces through Performance-Guided Test Generation*, Pradel et al., OOPSLA 2014

Motivation

Event-based UI applications should be responsive

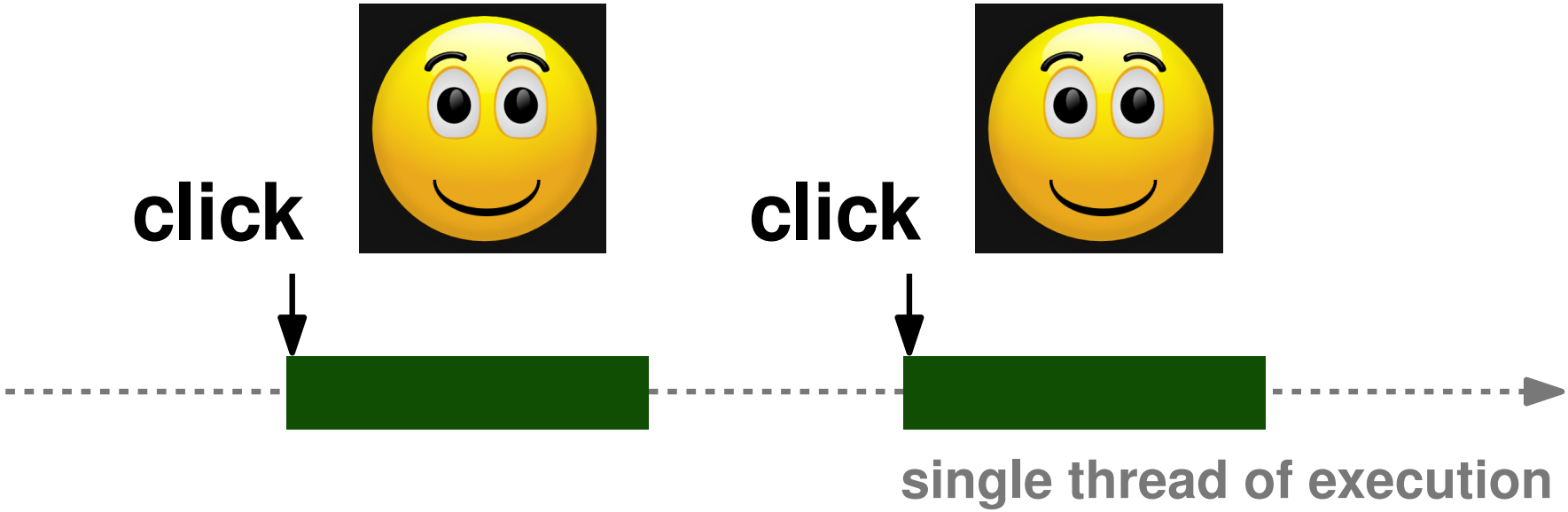
Motivation

Event-based UI applications should be responsive



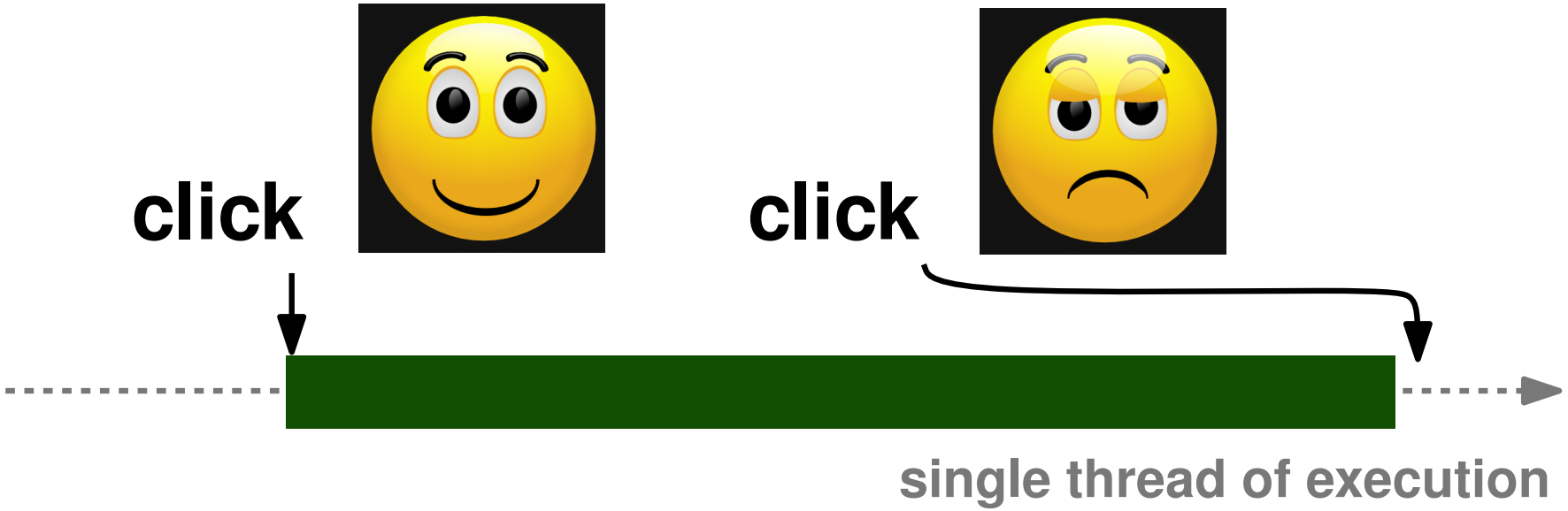
Motivation

Event-based UI applications should be responsive



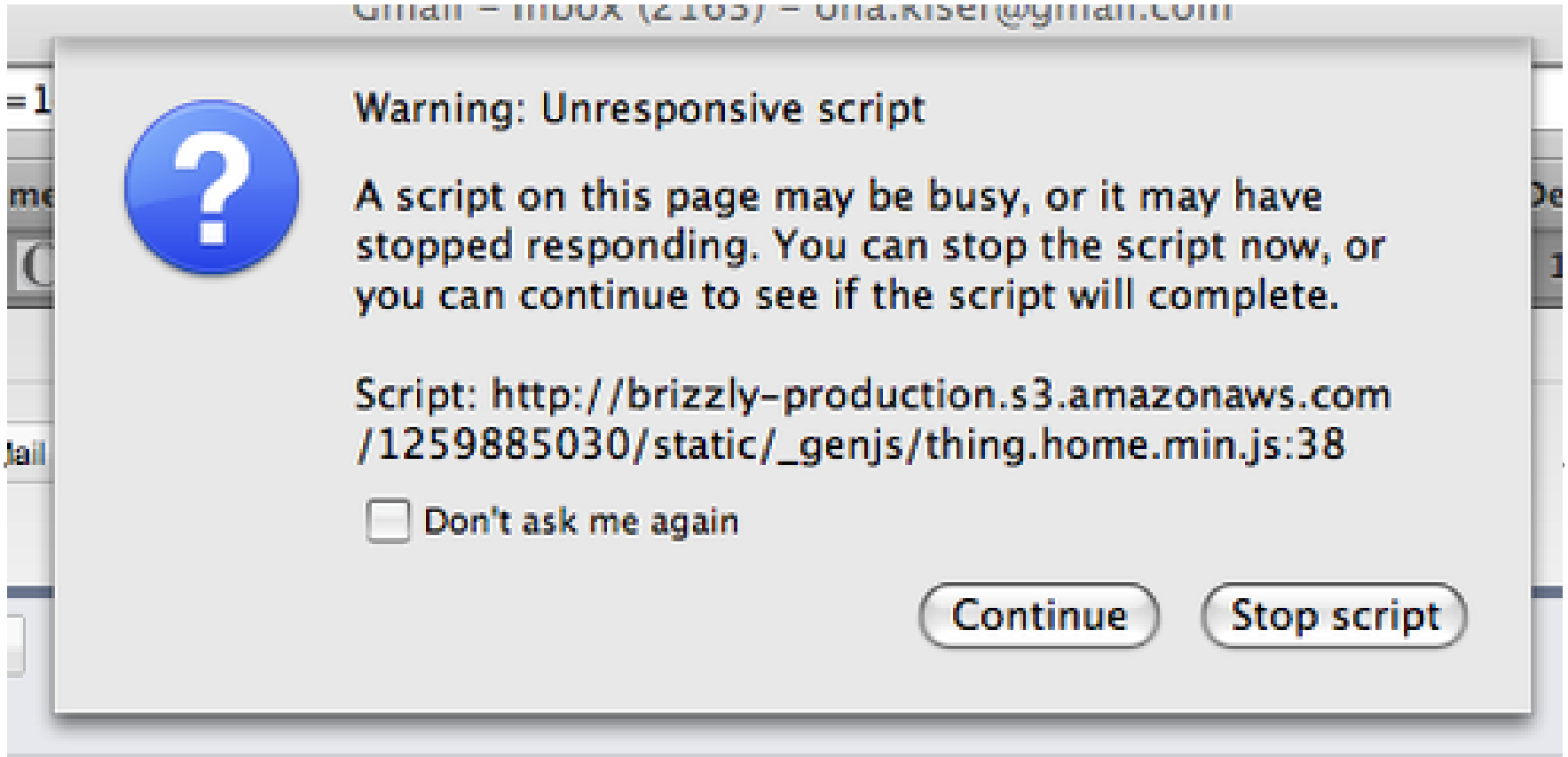
Motivation

Event-based UI applications should be responsive

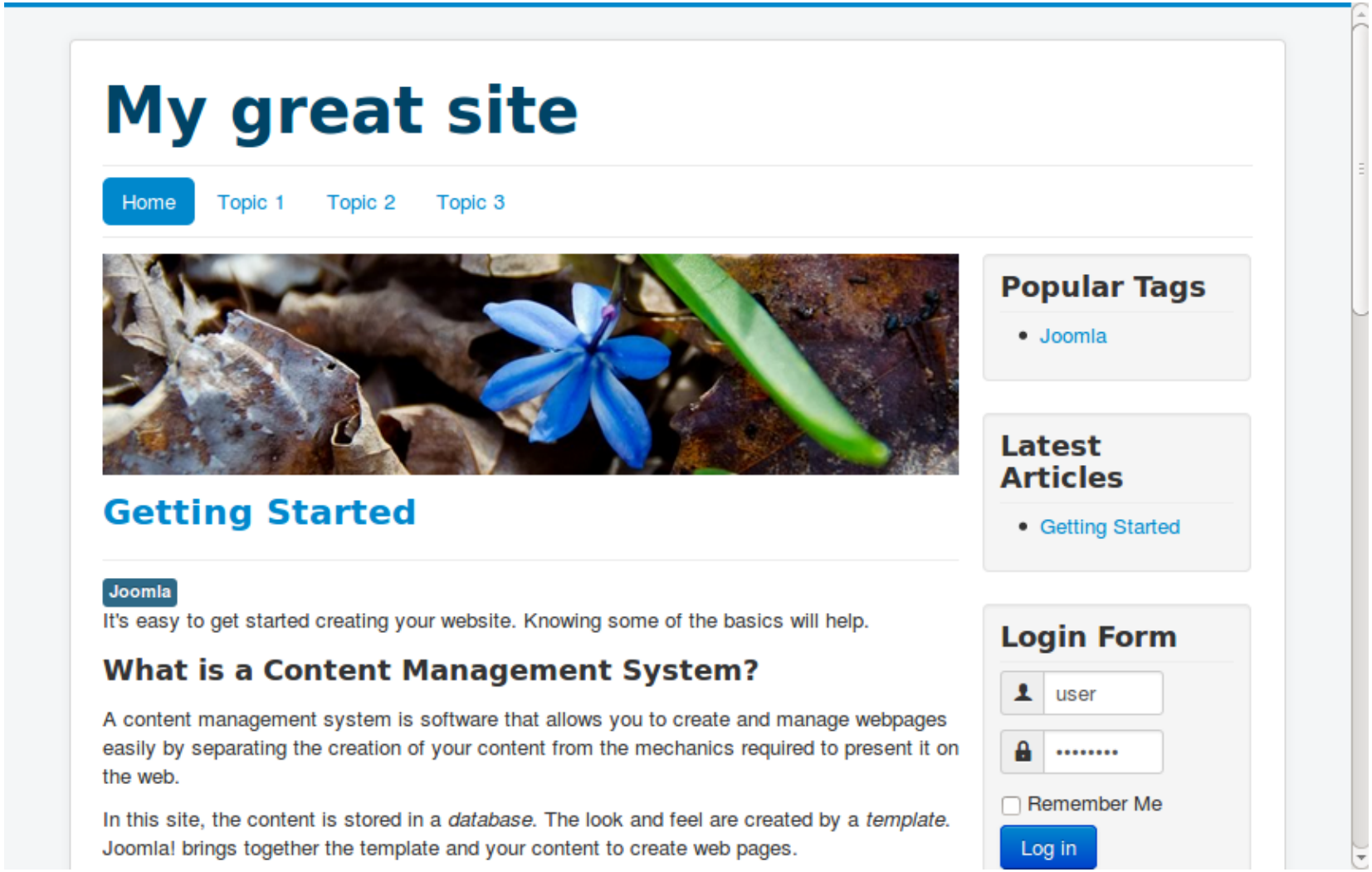


Motivation

Event-based UI applications should be responsive

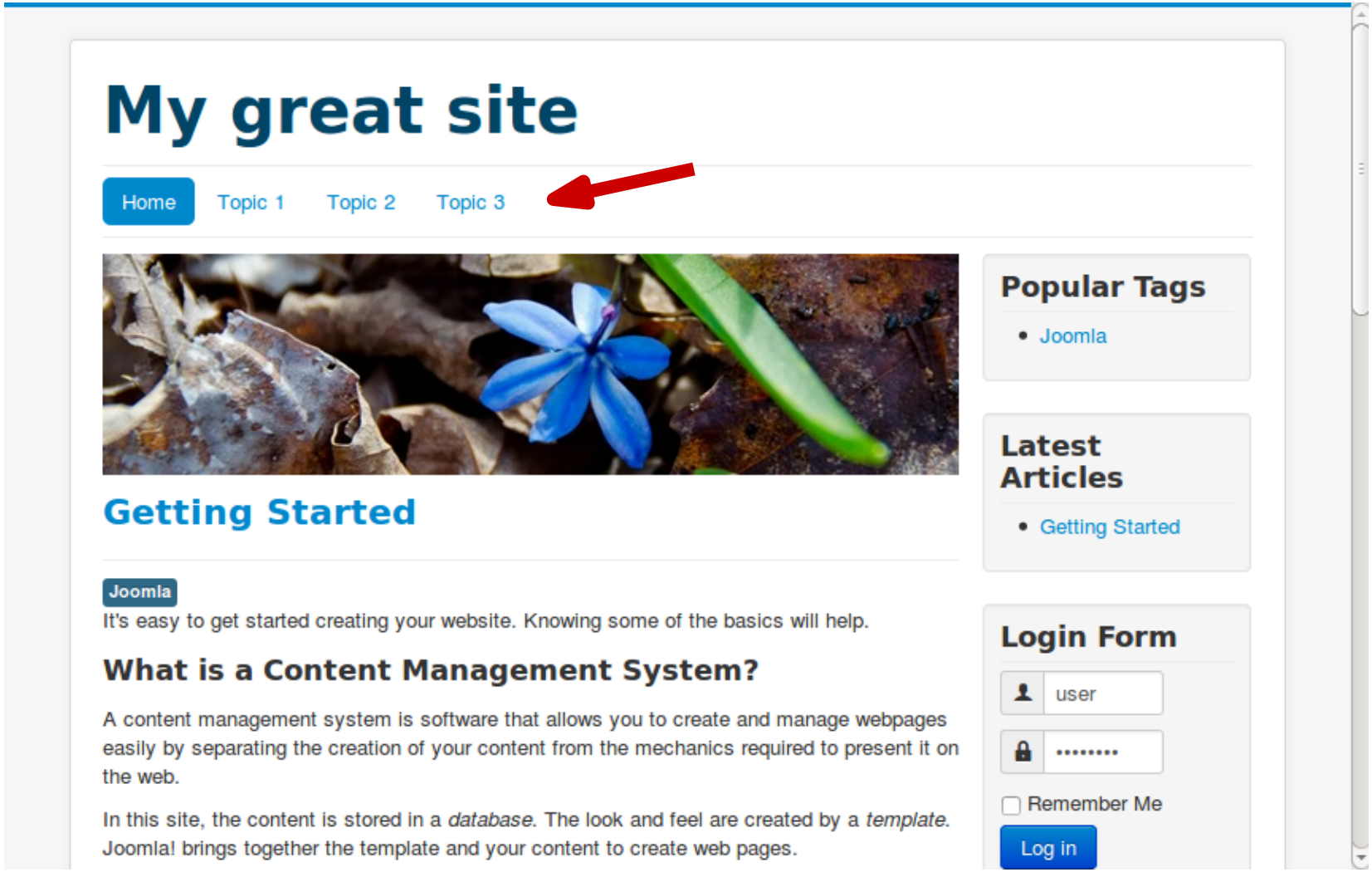


Real-World Example



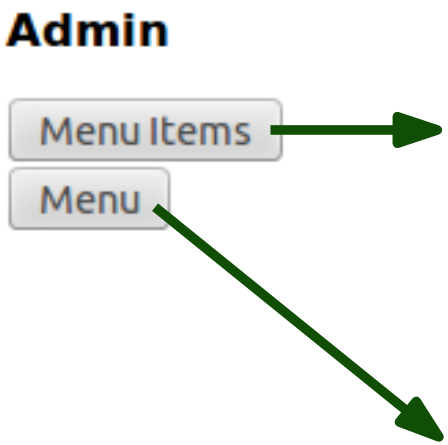
Joomla bug 30274

Real-World Example



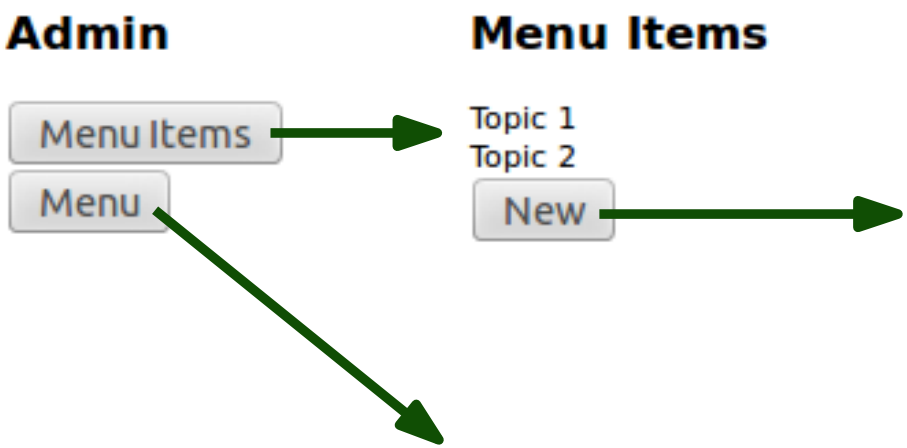
Joomla bug 30274

Real-World Example



Joomla bug 30274

Real-World Example

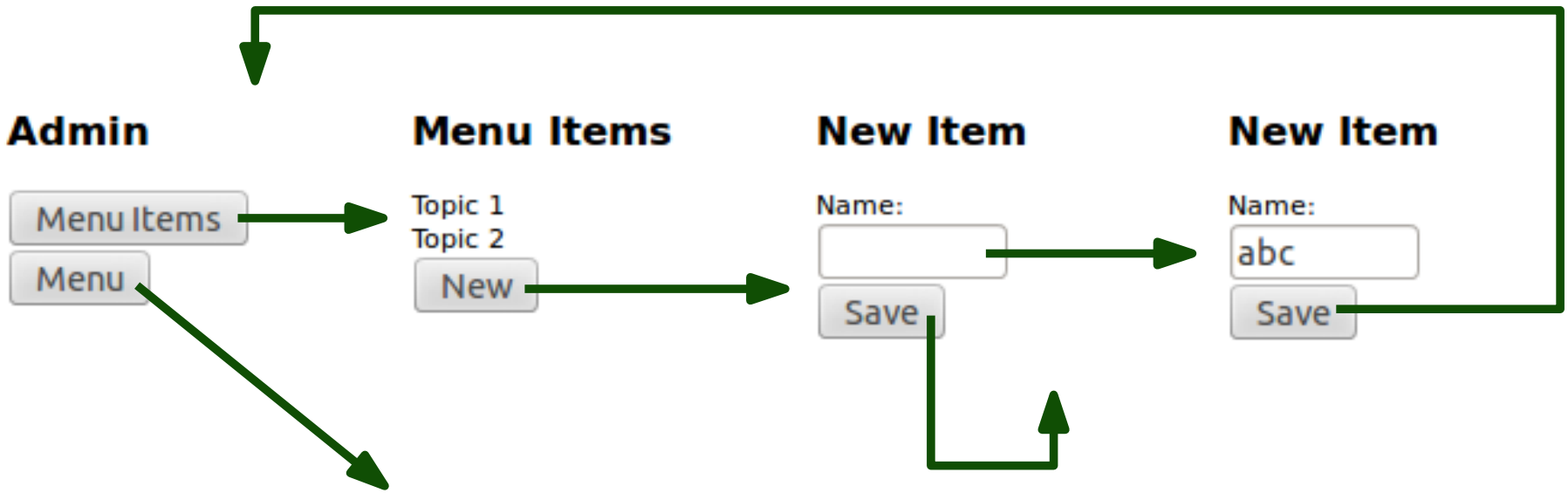


Joomla bug 30274

Real-World Example

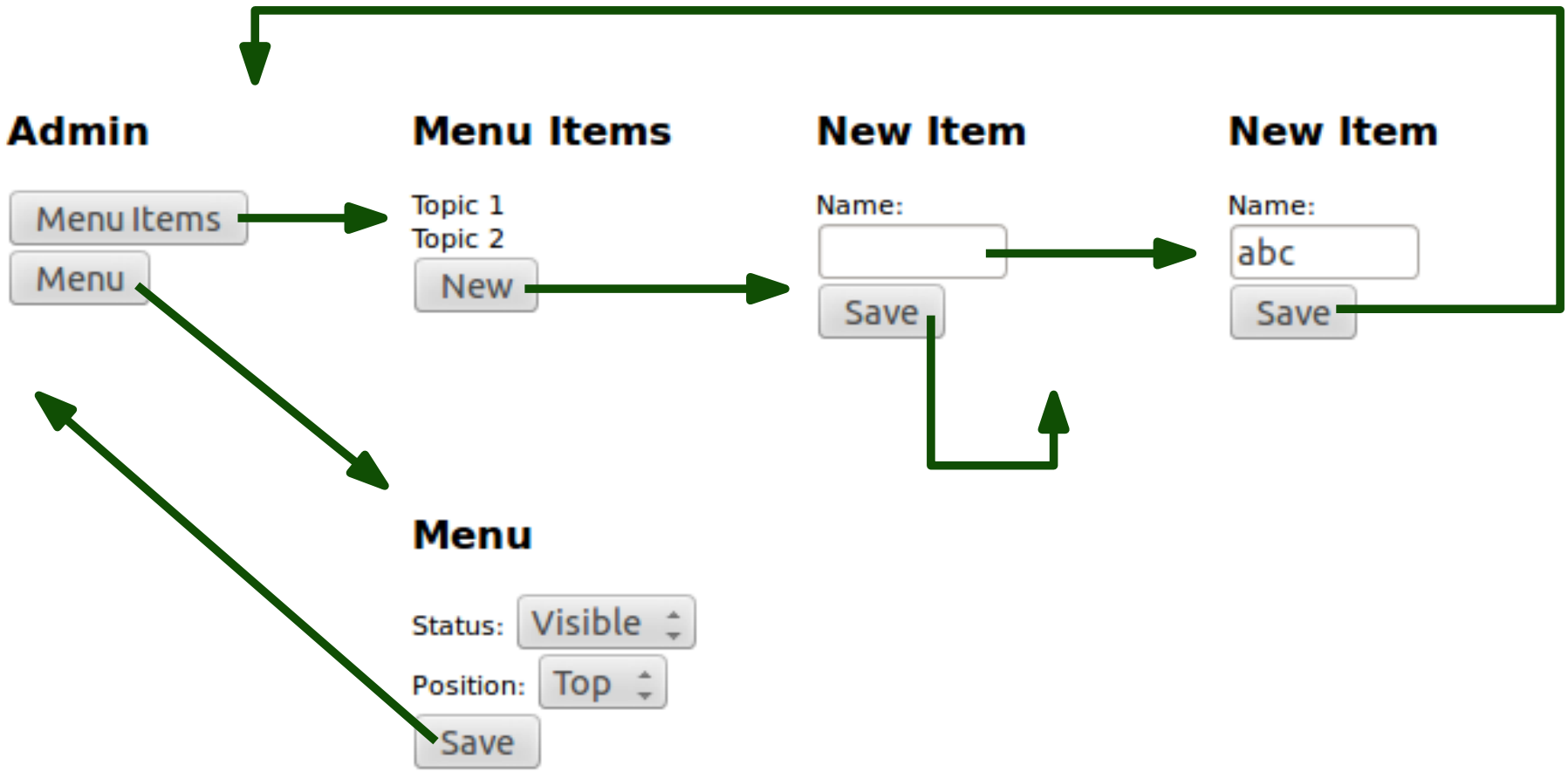
Joomla bug 30274

Real-World Example



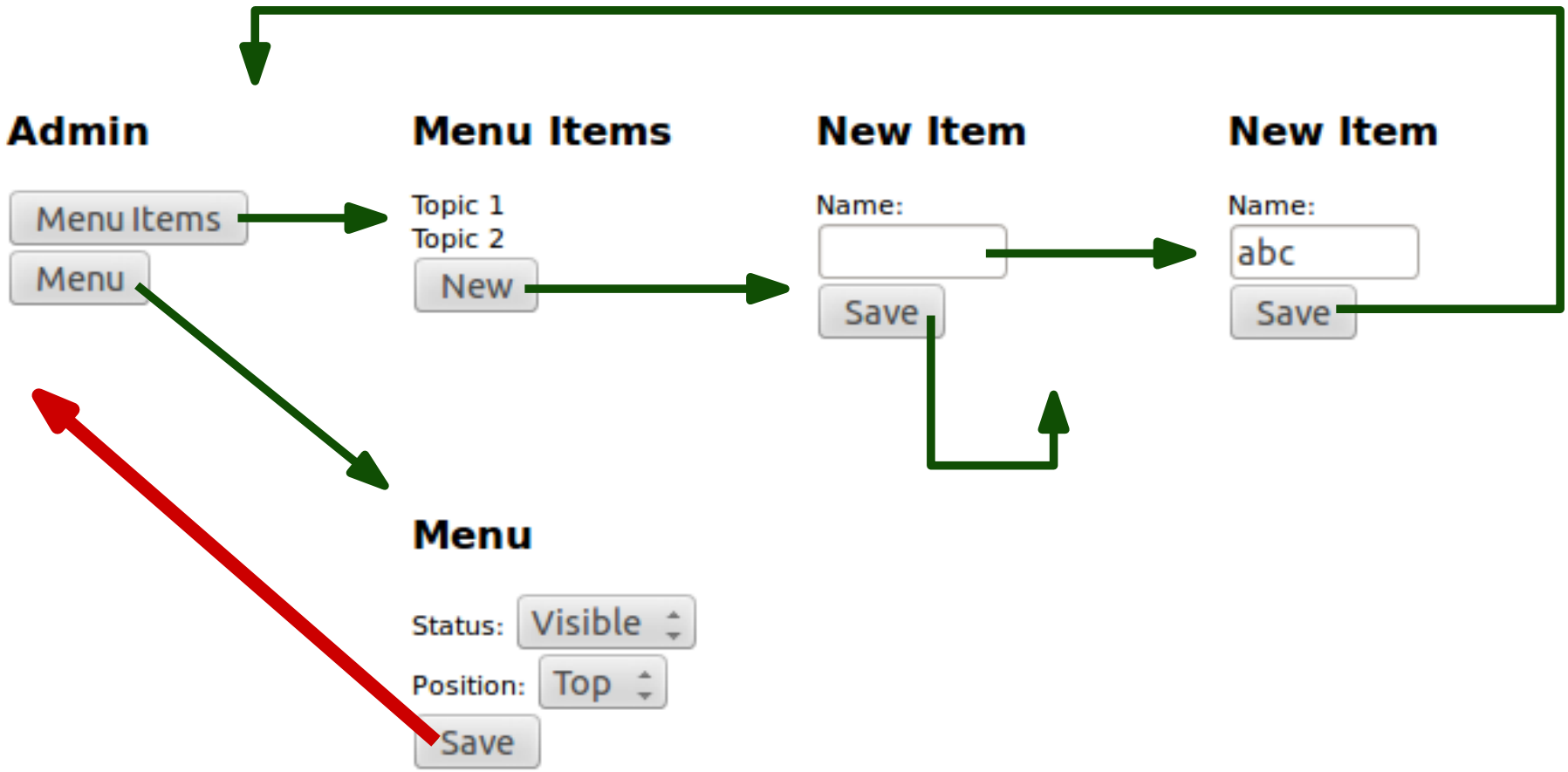
Joomla bug 30274

Real-World Example



Joomla bug 30274

Real-World Example

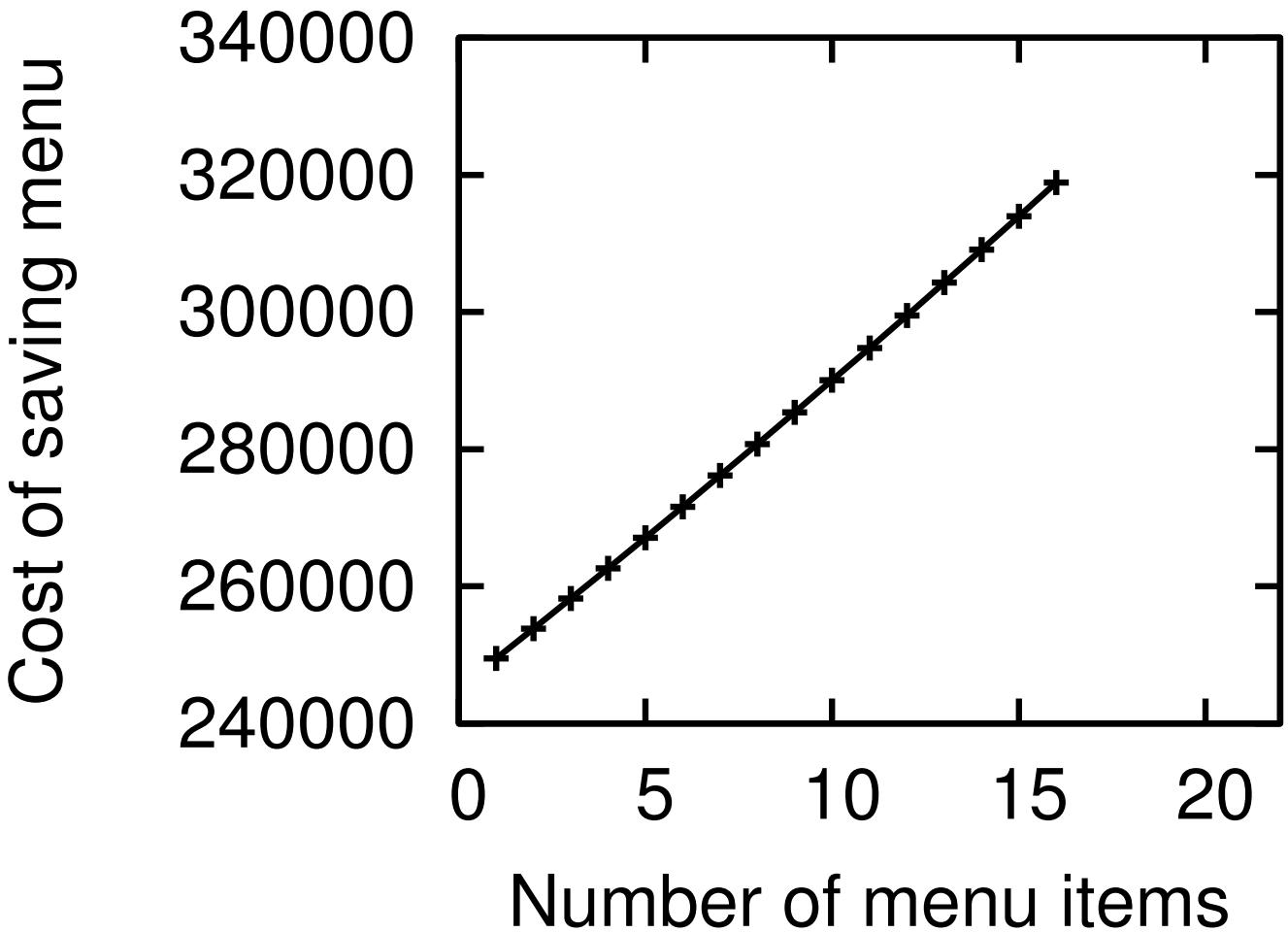


Unresponsive

Joomla bug 30274

Real-World Example

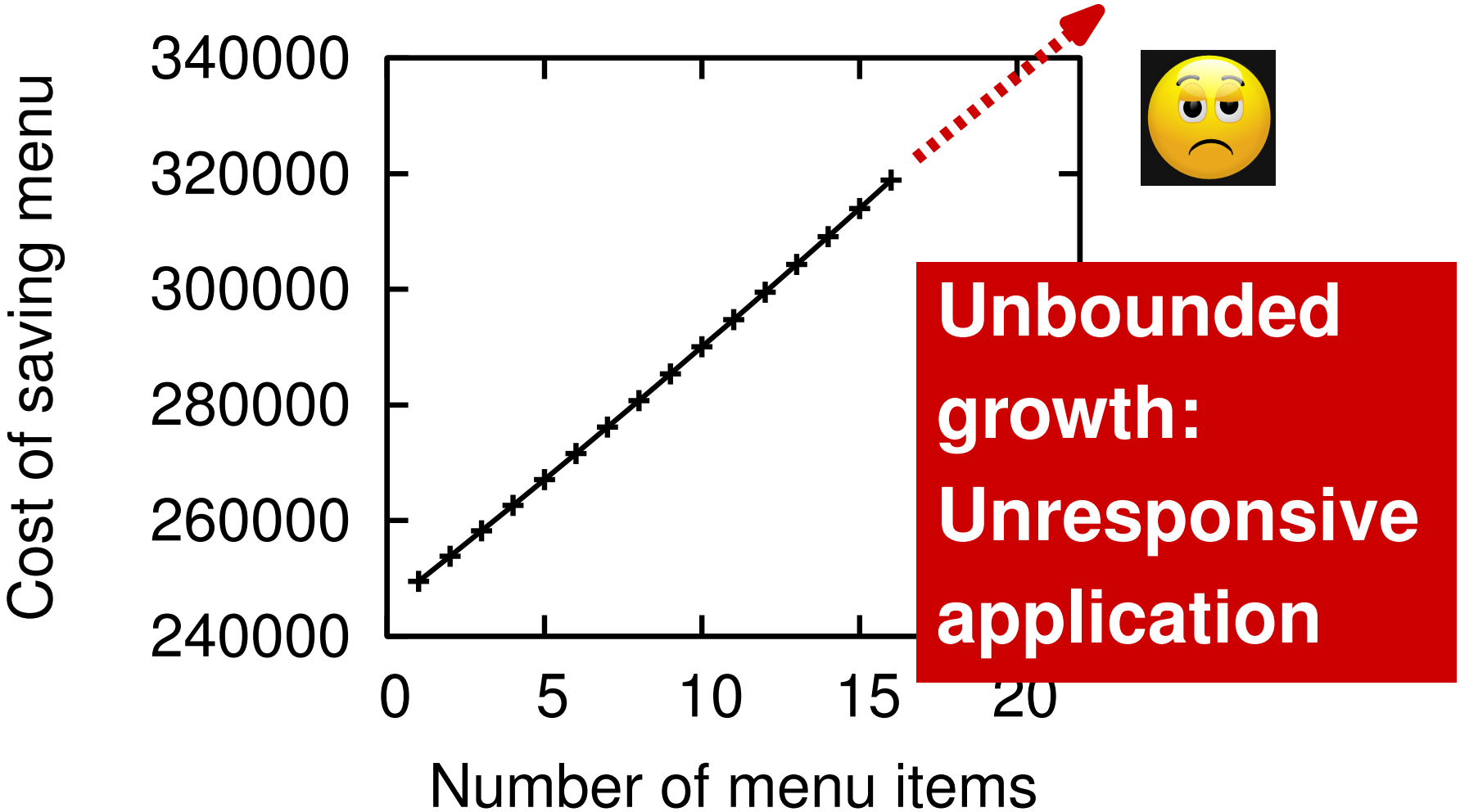
Cost plot for responsiveness problem



Joomla bug 30274

Real-World Example

Cost plot for responsiveness problem



Joomla bug 30274

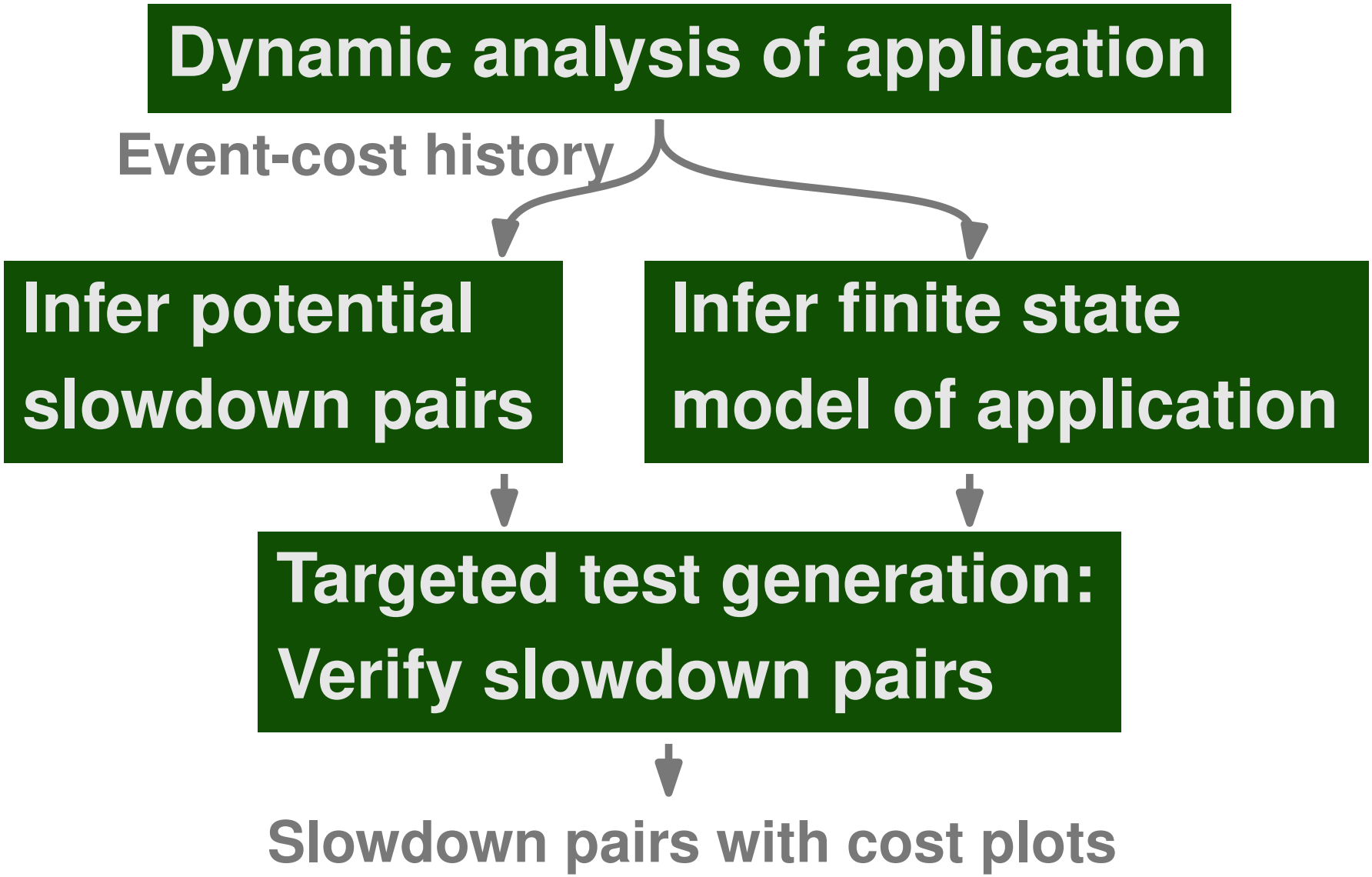
EventBreak: Idea

Analyze responsiveness of web applications through automated testing

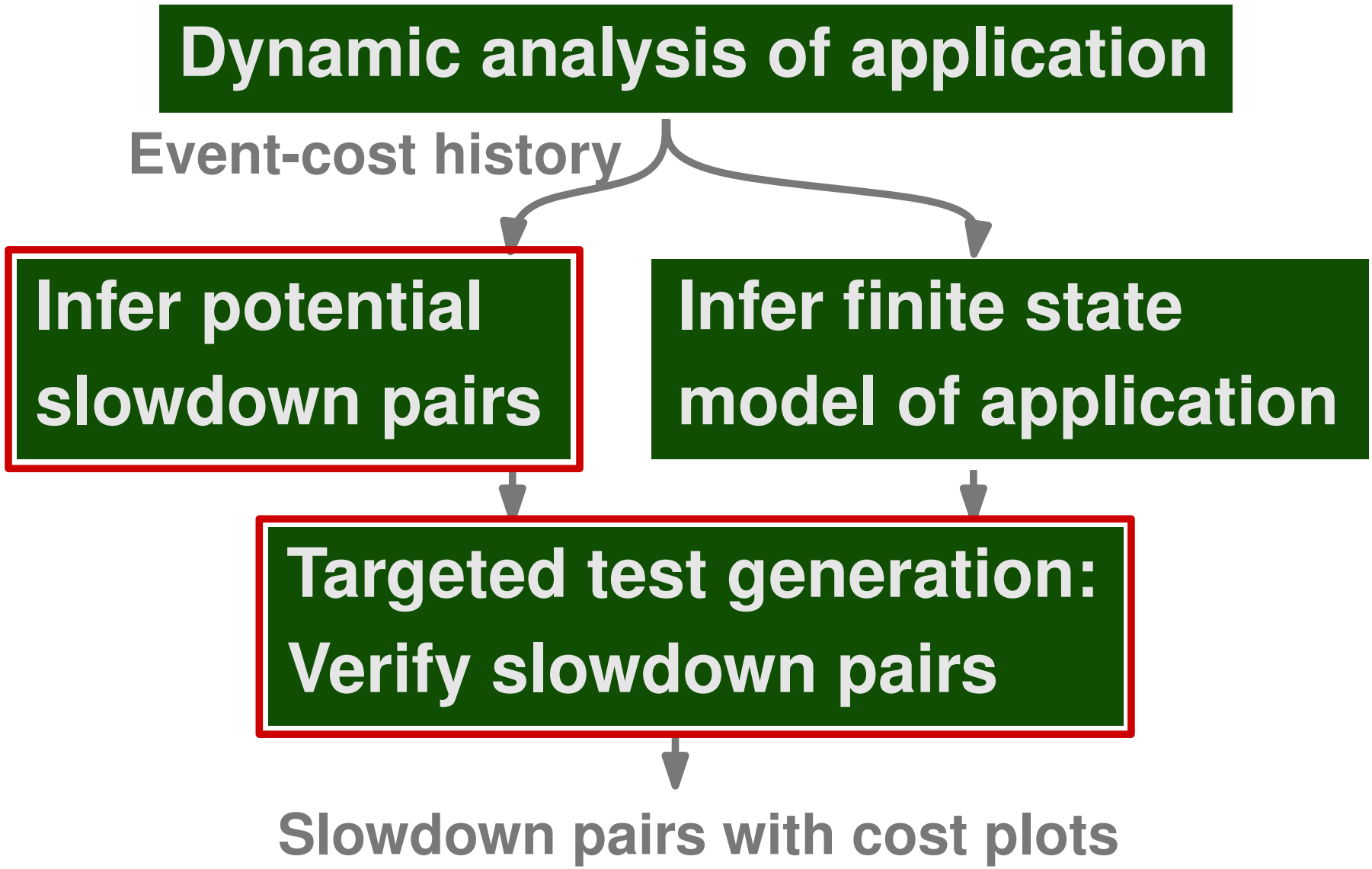
Focus: Slowdown pairs

Event E_{cause} increases cost of event E_{effect}

Overview


Dynamic analysis of application
Event-cost history
Infer potential slowdown pairs
Infer finite state model of application
Targeted test generation: Verify slowdown pairs
Slowdown pairs with cost plots

Overview


Dynamic analysis of application
Event-cost history
Infer potential slowdown pairs
Infer finite state model of application
Targeted test generation: Verify slowdown pairs
Slowdown pairs with cost plots

Event-Cost History

Sequence of event-cost pairs

- **DOM element**
- **Type of event**
- **Pre-state**
- **Post-state**

Number of conditionals evaluated in event handler

Potential Slowdown Pairs

Does A increase cost of B?

Event	Cost
B	5
A	3
B	10
B	12
A	3
B	12

Supporting evidence S

Refuting evidence R

$$Supp = |S| = 1 \qquad Conf = \frac{|S|}{|S| + |R|} = 33\%$$

Algorithm

- Input: History of event-cost pairs (e, c) with unique events E
- For each $(e_1, e_2) \in E \times E$
  - Initialize $S(e_1, e_2) = 0$ and $R(e_1, e_2) = 0$
  - For each minimal subsequence $(e_2, c_2^1), ..., (e_2, c_2^2)$ in history
    - If $c_2^1 < c_2^2$ and e_1 in subsequence, $S(e_1, e_2)$++
    - If $c_2^1 < c_2^2$ xor e_1 in subsequence, $R(e_1, e_2)$++
- Output (e_1, e_2) if $S(e_1, e_2) > min_{supp}$ and $\frac{S(e_1,e_2)}{S(e_1,e_2)+R(e_1,e_2)} > min_{conf}$

Quiz

Event	Cost
B	14
A	4
C	7
B	18
A	4
B	18
A	4
C	9
B	22

What are support and confidence for the potential slowdown pair (A,B)?

Event	Cost
B	14
A	4
C	7
B	18
A	14
B	18
A	4
C	9
B	22

$|S| = 2$

$|R| = 1$

$\rightarrow$ support $= 2$

confidence $= \frac{2}{1+2} = \frac{2}{3}$

Targeted Test Generation

Confirm or reject slowdown pair:

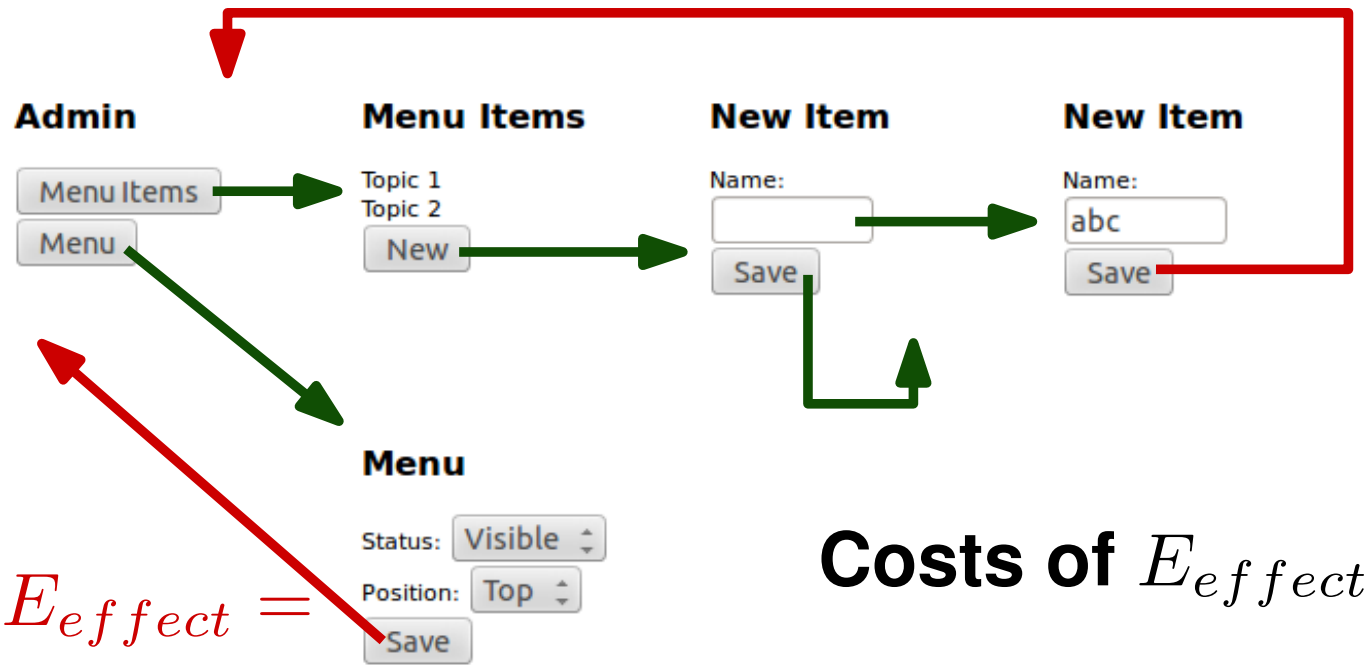
Alternate between E_{effect} **and** E_{cause}

Targeted Test Generation

Confirm or reject slowdown pair:

Alternate between E_{effect} and E_{cause}

E_{cause} = **Save new item**



E_{effect} = **Save menu**

Costs of E_{effect}:

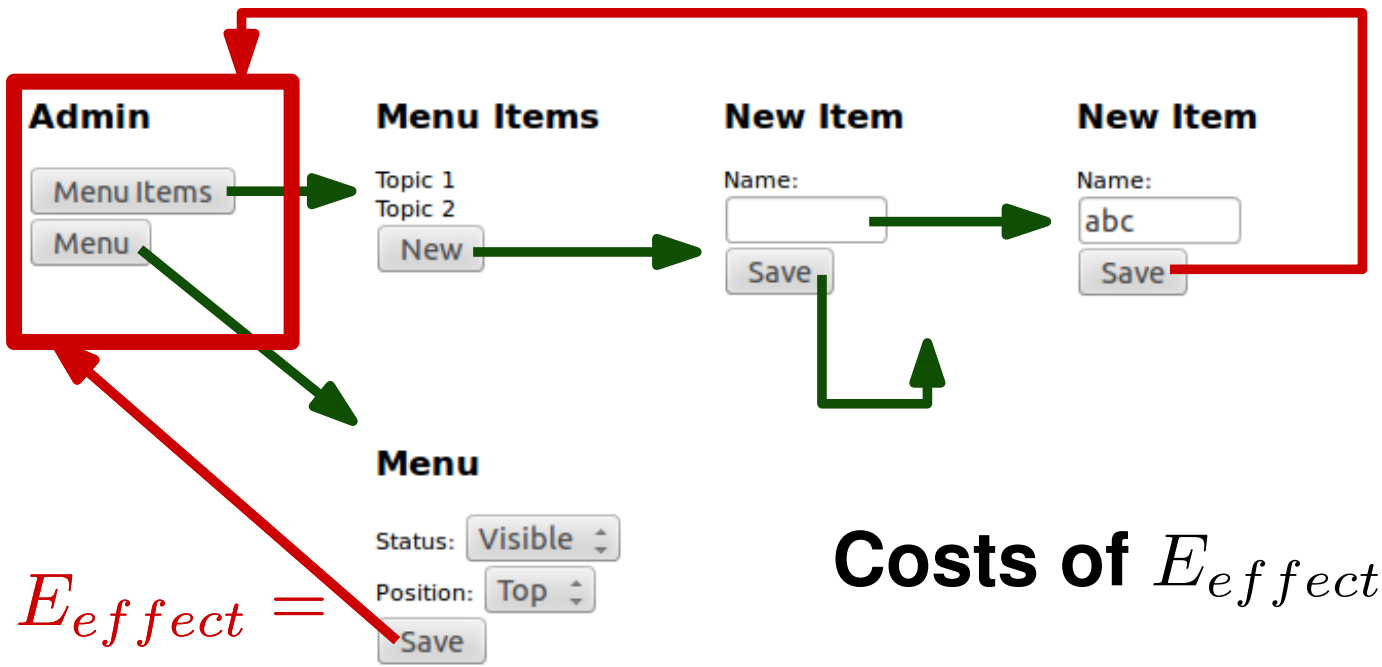
current state → target event

Targeted Test Generation

Confirm or reject slowdown pair:

Alternate between E_{effect} and E_{cause}

E_{cause} = **Save new item**



E_{effect} = **Save menu**

Costs of E_{effect}:

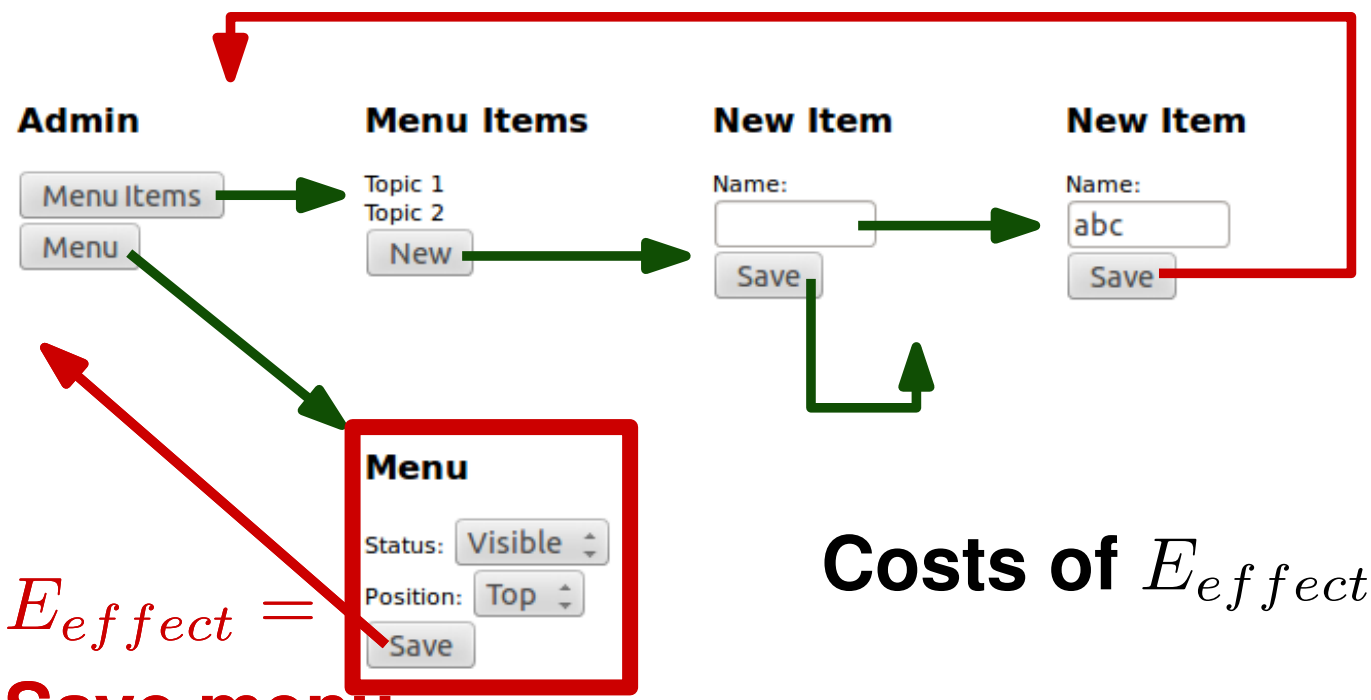
current state → target event

Targeted Test Generation

Confirm or reject slowdown pair:

Alternate between E_{effect} and E_{cause}

E_{cause} = **Save new item**



E_{effect} = **Save menu**

Costs of E_{effect}:

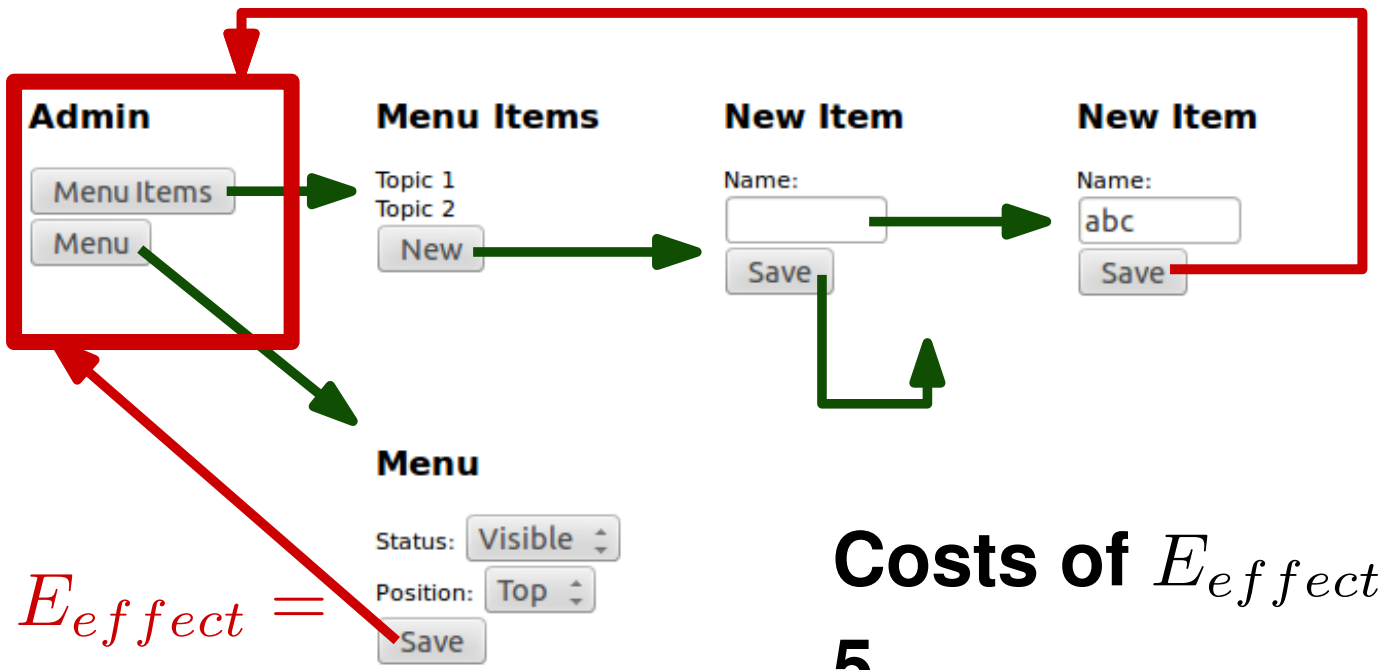
current state → target event

Targeted Test Generation

Confirm or reject slowdown pair:

Alternate between E_{effect} and E_{cause}

E_{cause} = **Save new item**



E_{effect} = **Save menu**

Costs of E_{effect}:

5

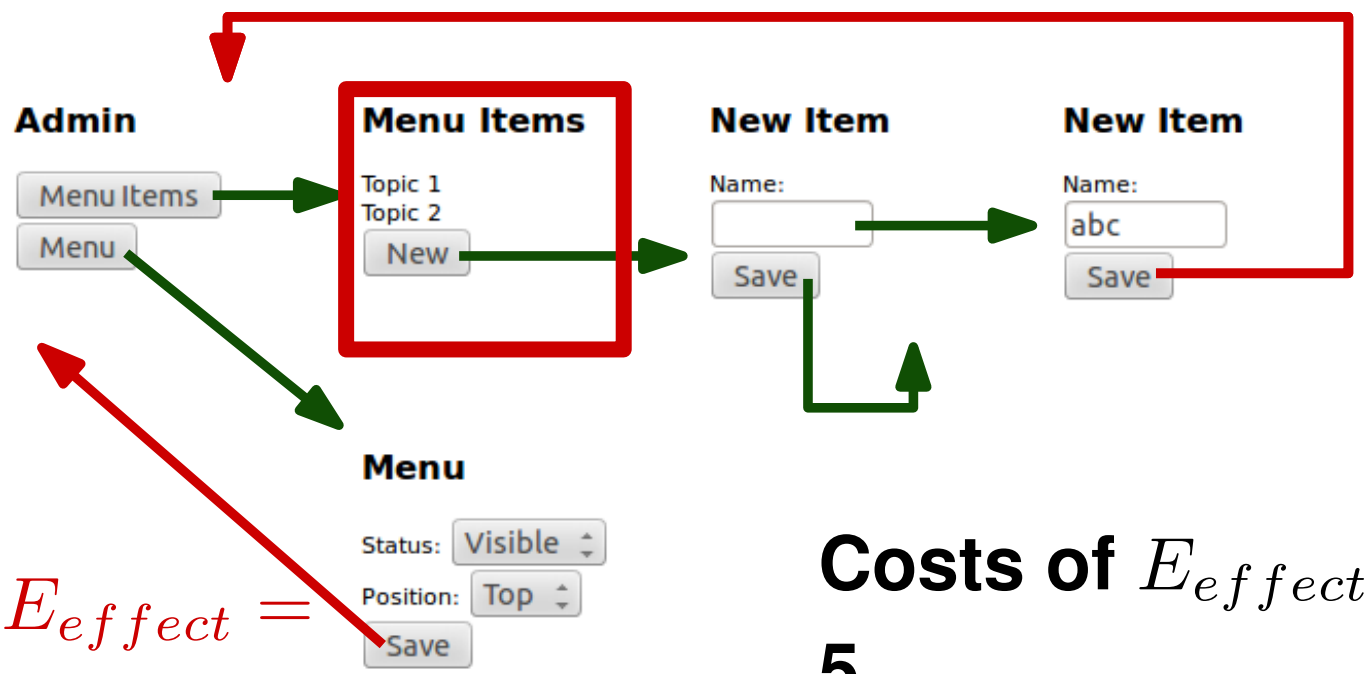
current state → target event

Targeted Test Generation

Confirm or reject slowdown pair:

Alternate between E_{effect} and E_{cause}

E_{cause} = **Save new item**



E_{effect} = **Save menu**

Costs of E_{effect}:

5

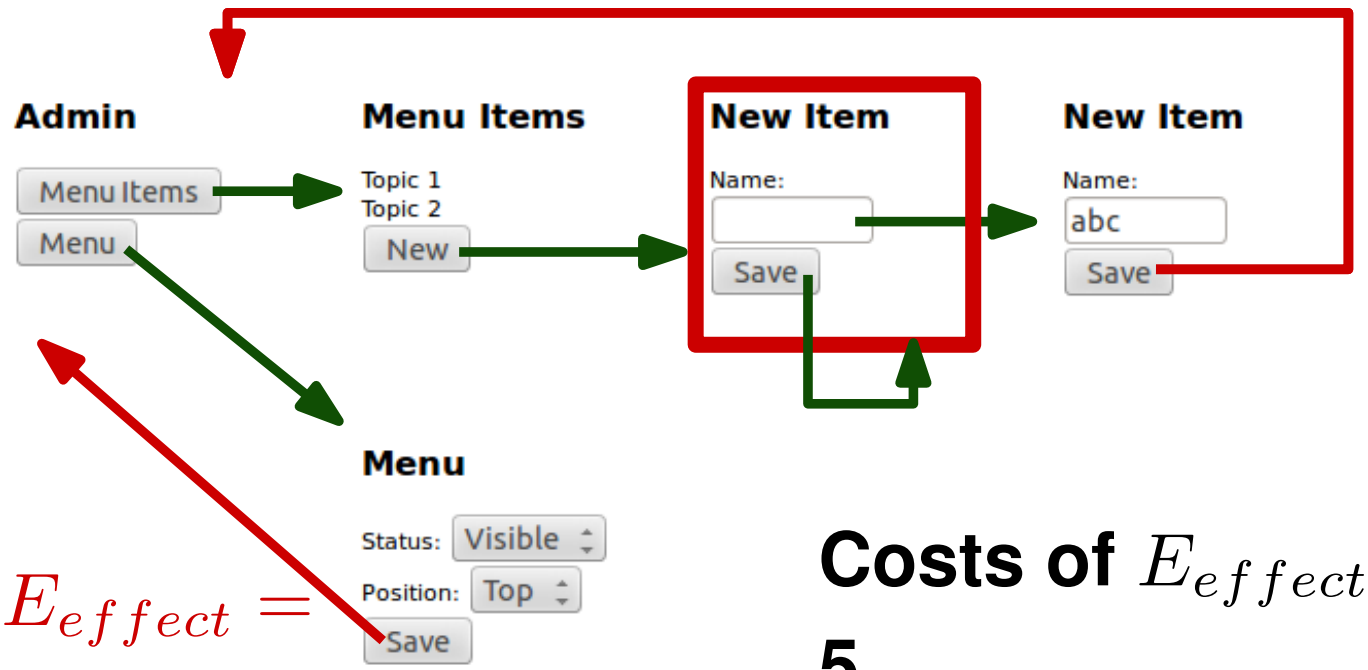
current state → target event

Targeted Test Generation

Confirm or reject slowdown pair:

Alternate between E_{effect} and E_{cause}

E_{cause} = **Save new item**



E_{effect} = **Save menu**

Costs of E_{effect}:

5

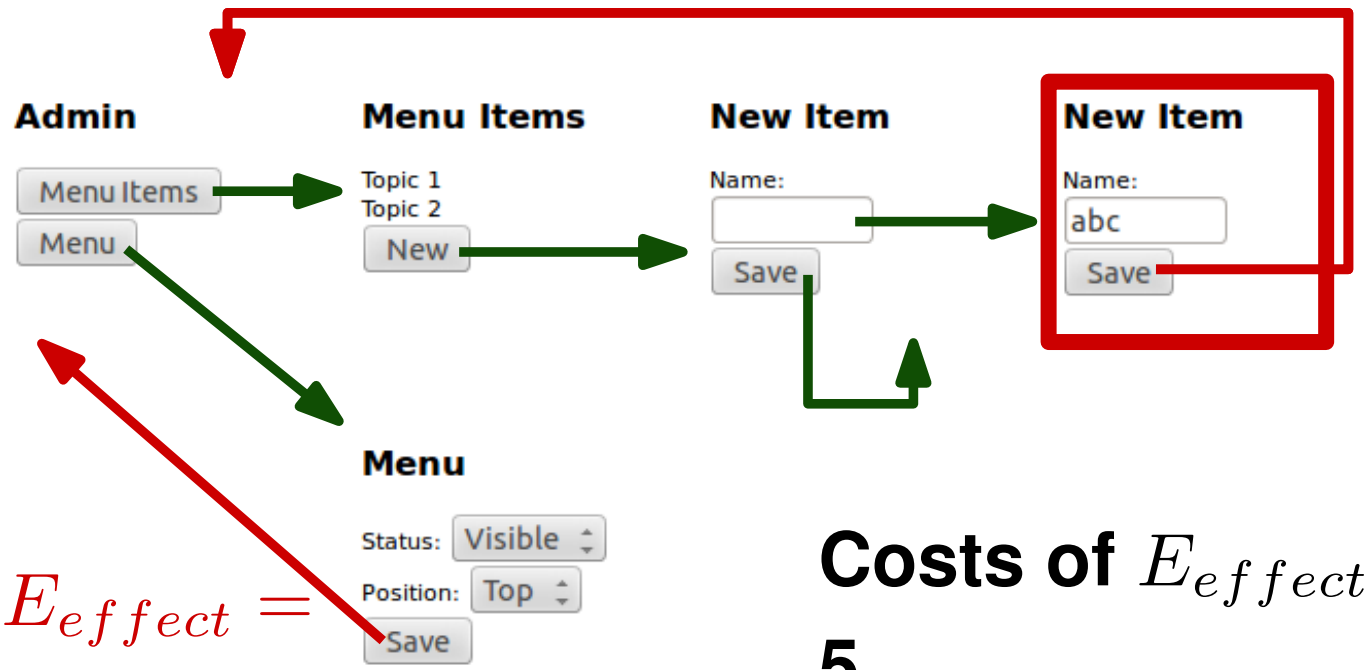
current state → target event

Targeted Test Generation

Confirm or reject slowdown pair:

Alternate between E_{effect} and E_{cause}

E_{cause} = **Save new item**



E_{effect} = **Save menu**

Costs of E_{effect}:

5

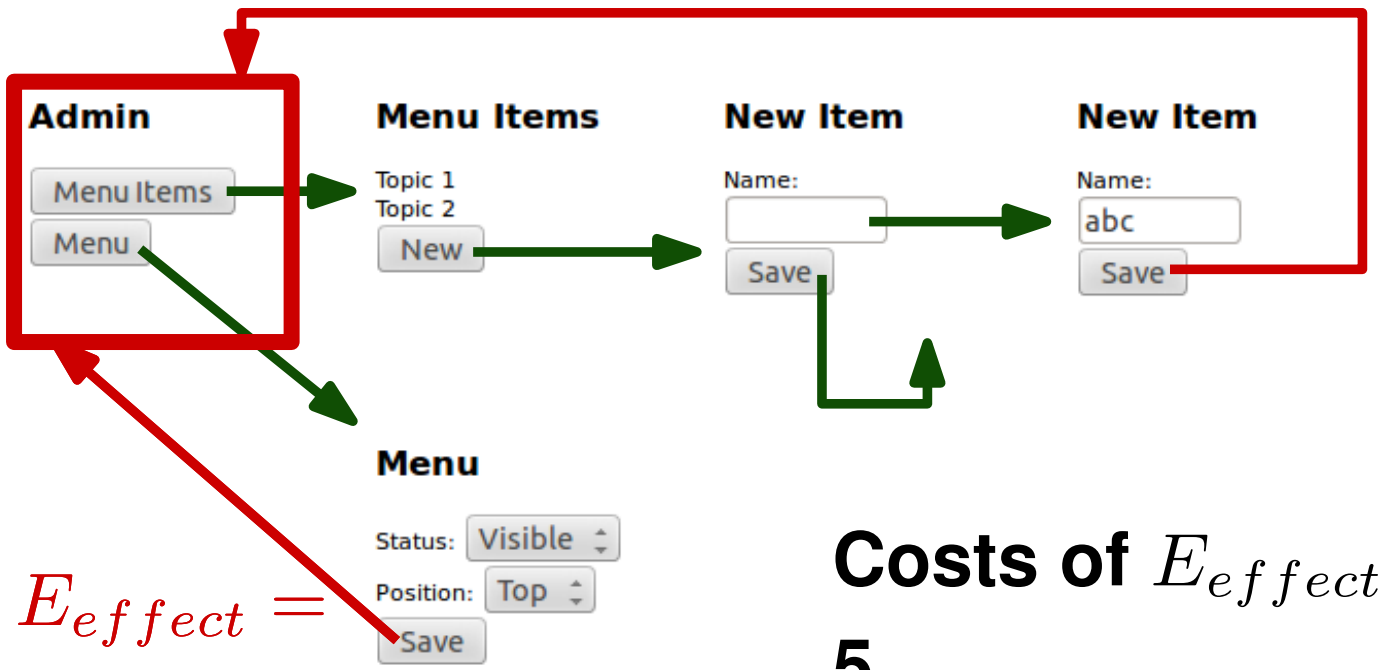
current state → target event

Targeted Test Generation

Confirm or reject slowdown pair:

Alternate between E_{effect} and E_{cause}

E_{cause} = **Save new item**



E_{effect} = **Save menu**

Costs of E_{effect}:

5

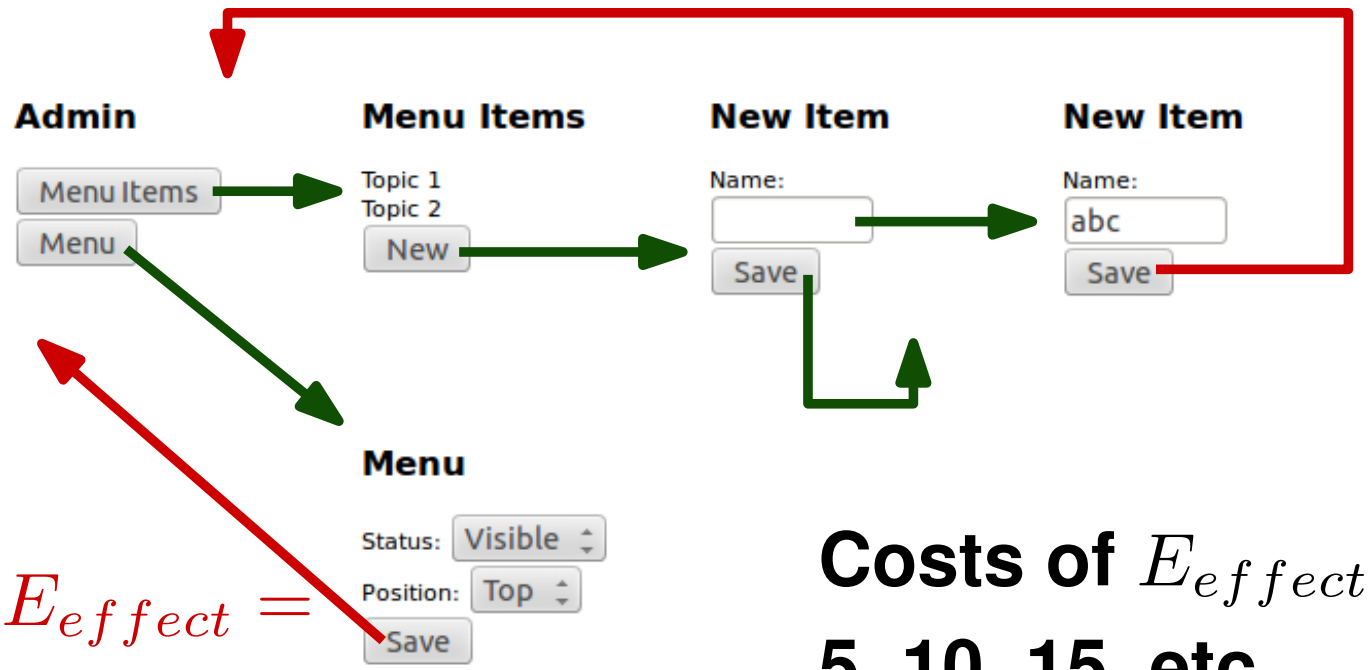
current state → target event

Targeted Test Generation

Confirm or reject slowdown pair:

Alternate between E_{effect} and E_{cause}

E_{cause} = **Save new item**



E_{effect} = **Save menu**

Costs of E_{effect}:

5, 10, 15, etc.

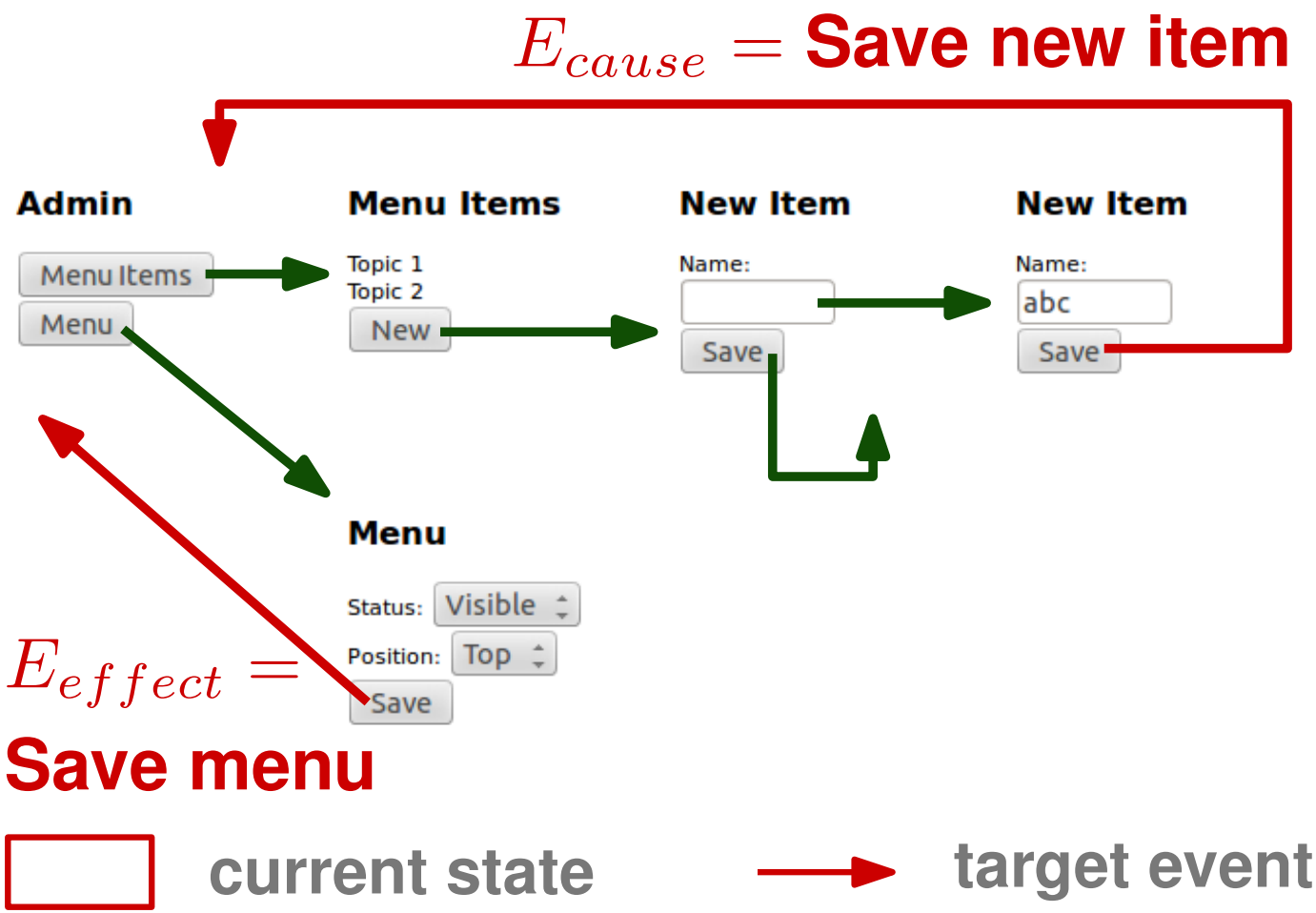
current state → target event

Navigation: Challenge

Shortest event sequence in model may be infeasible in application

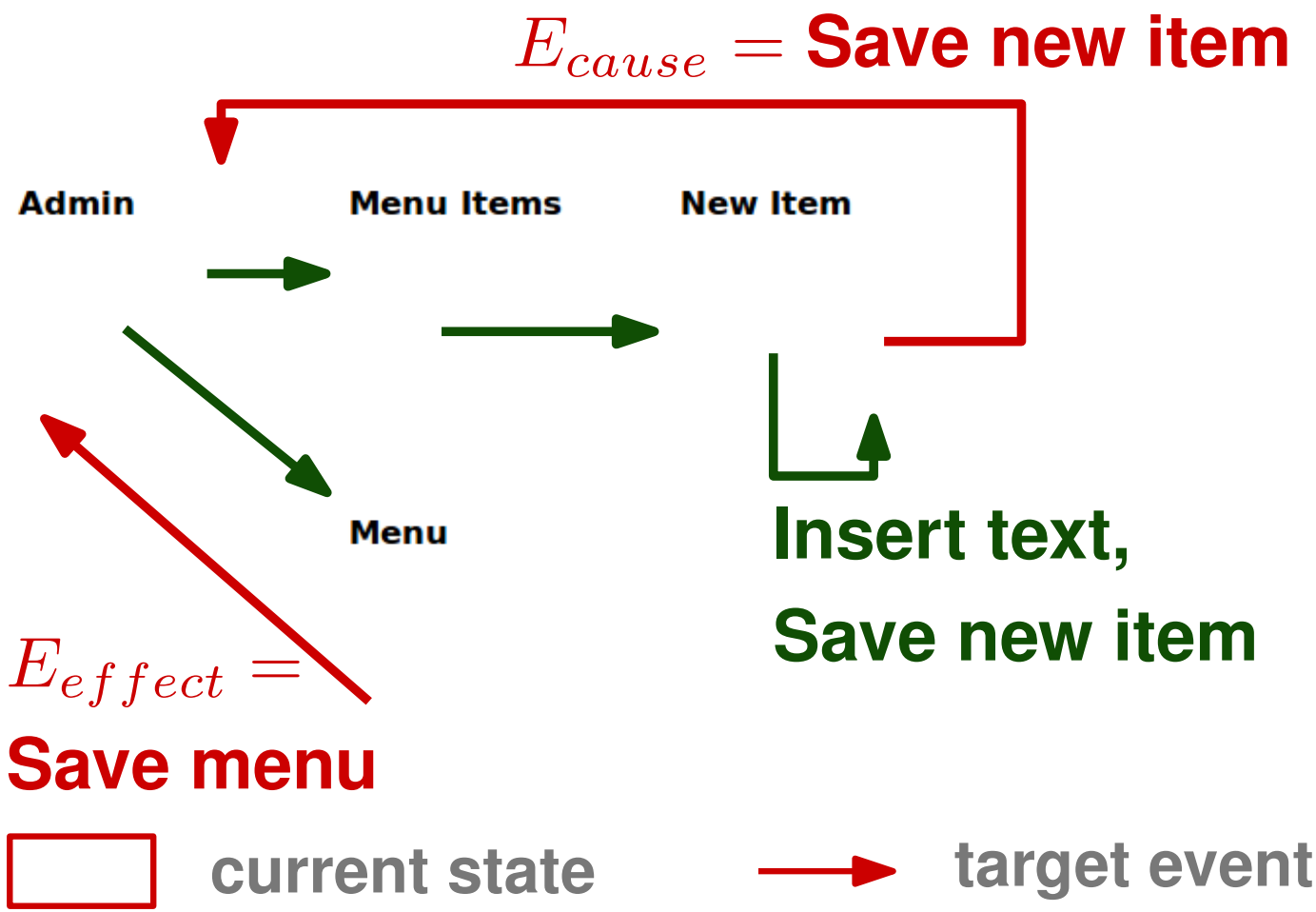
Navigation: Challenge

Shortest event sequence in model may be infeasible in application



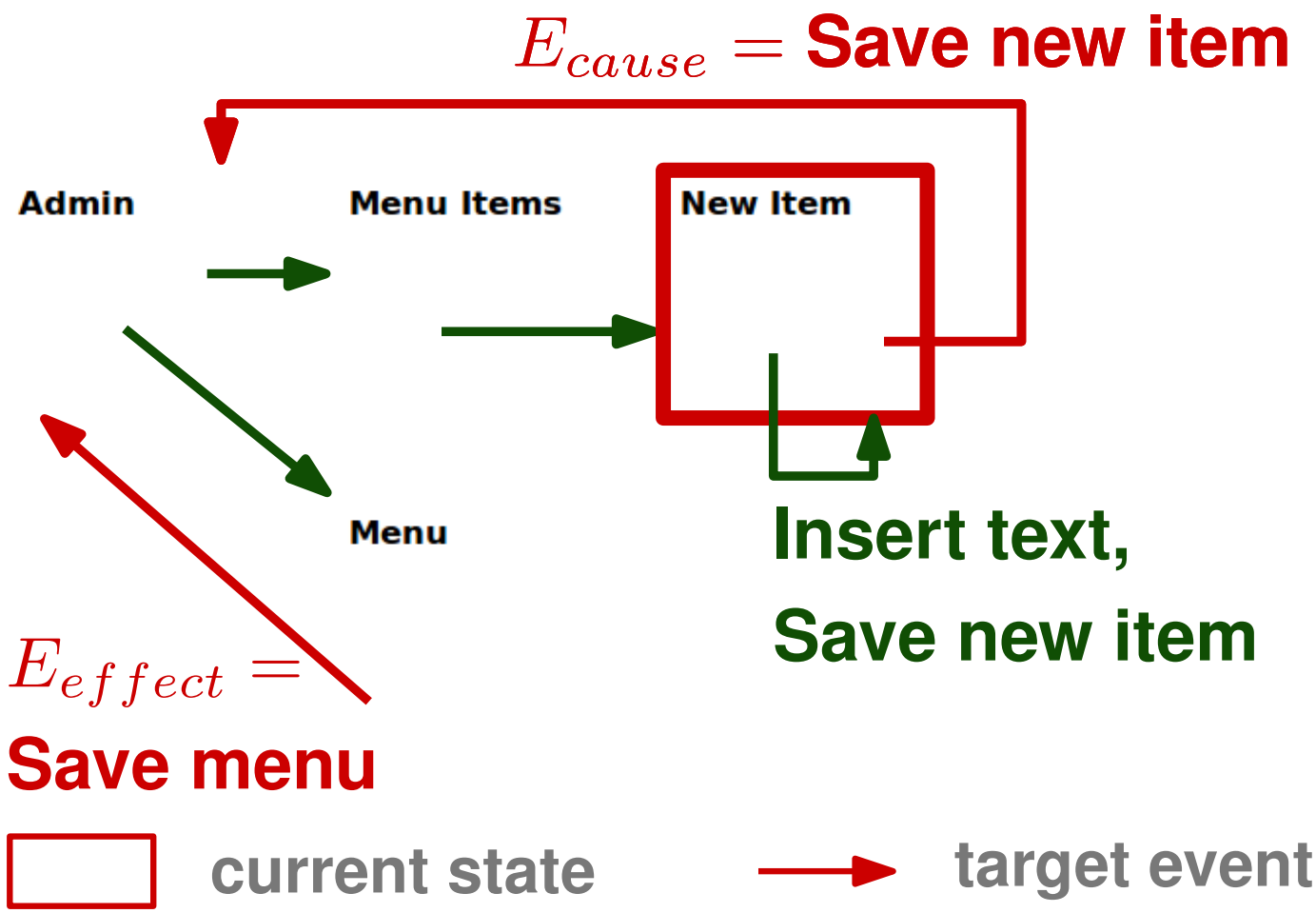
Navigation: Challenge

Shortest event sequence in model may be infeasible in application



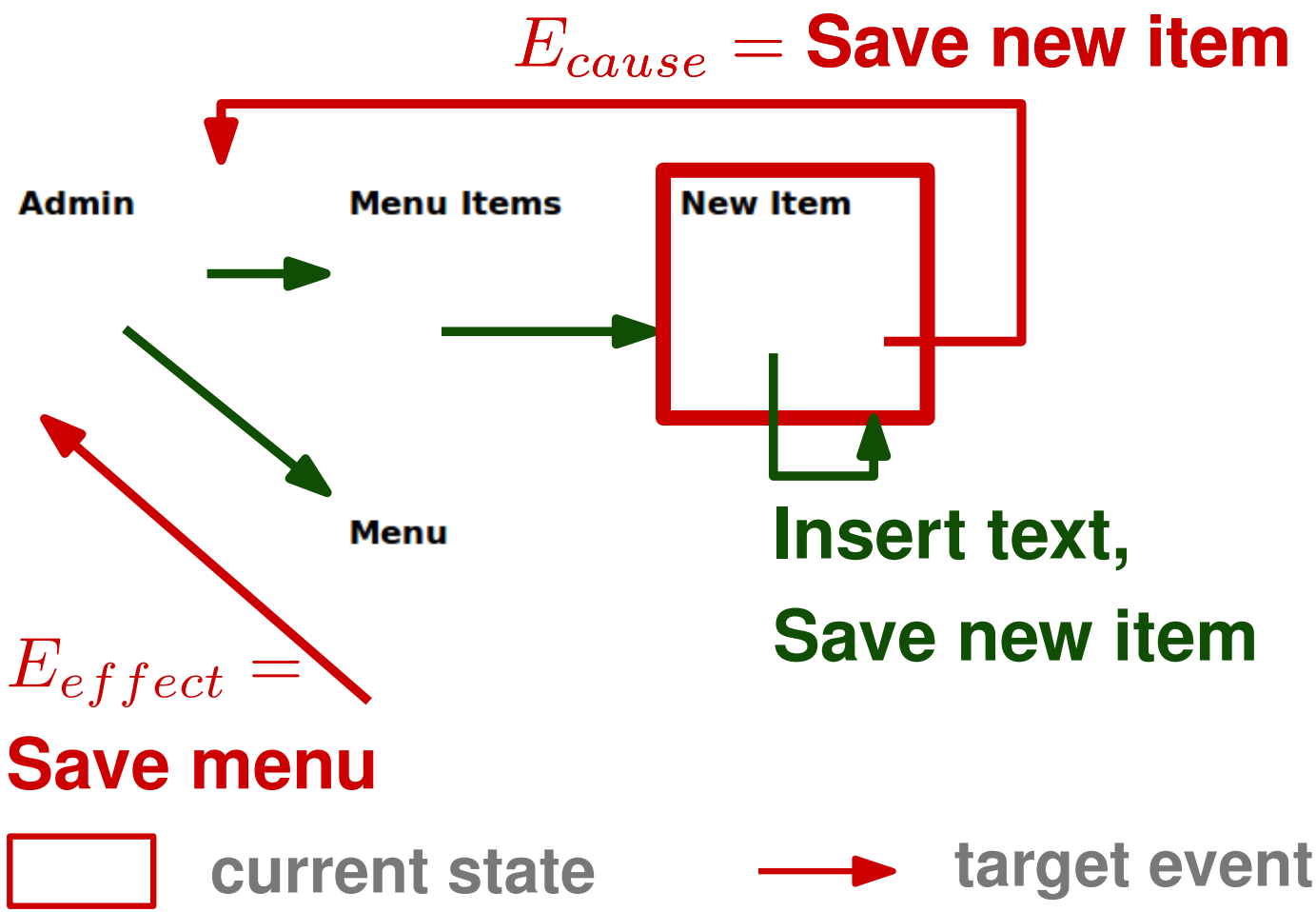
Navigation: Challenge

Shortest event sequence in model may be infeasible in application



Navigation: Approach

- Compute sequences to target event
- Randomly pick from set of first steps



Evaluation

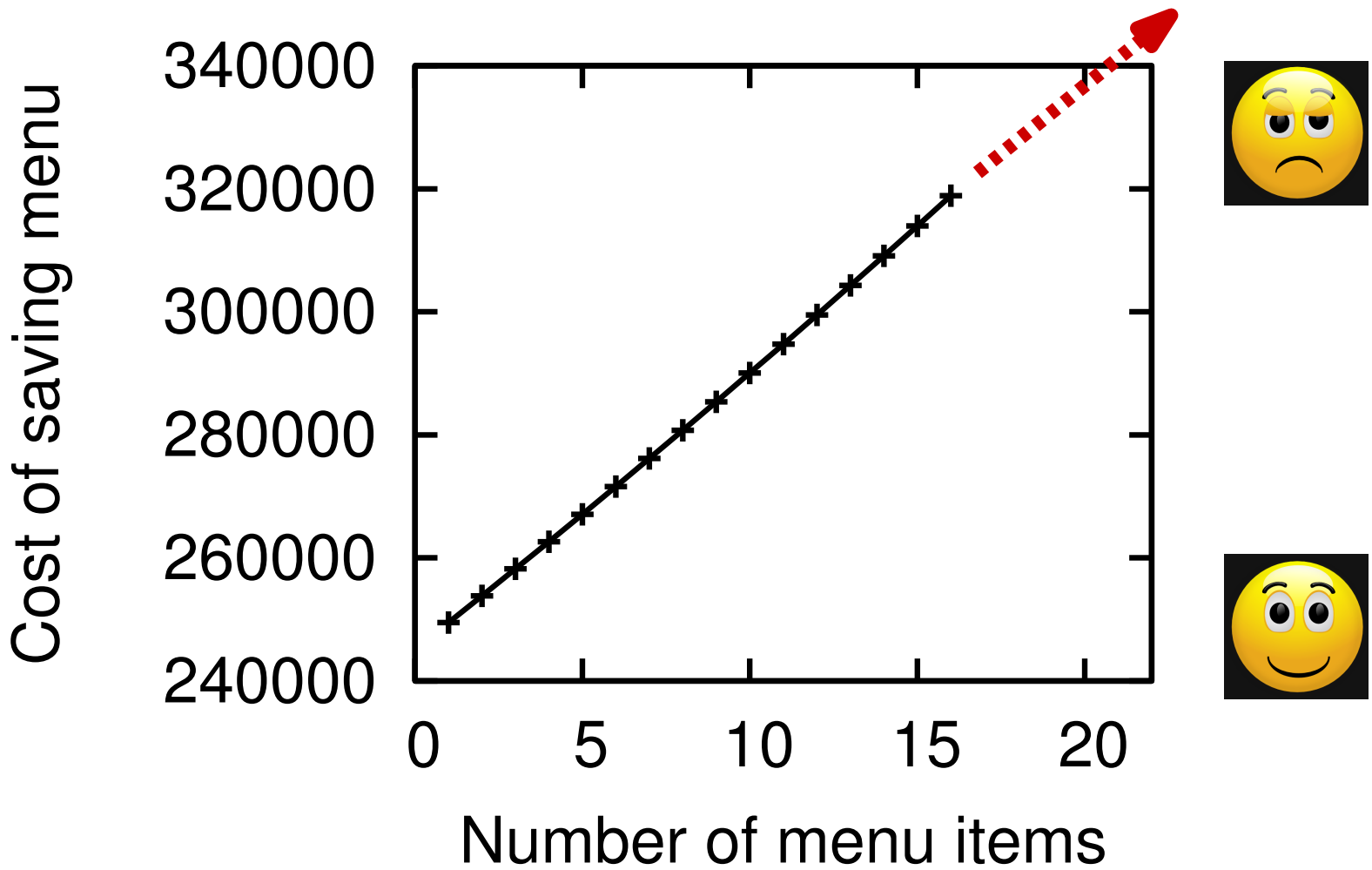
Find responsiveness problems?

- 2 known + 4 previously unknown

Effectiveness of targeted test generation?

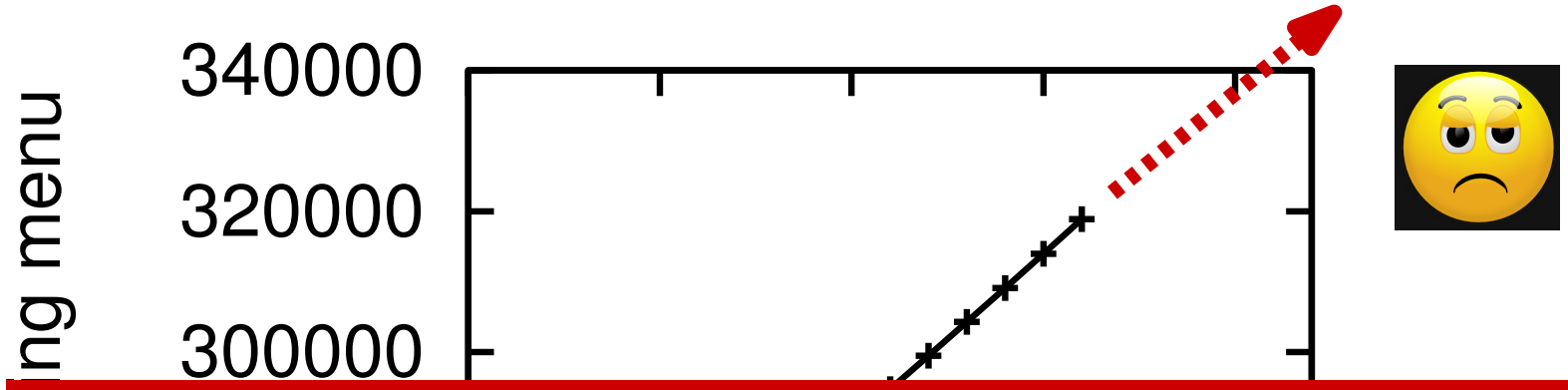
- Reaches 89% of all target events
- Invalidates $>$ 99% of all potential slowdown pairs

Results: Joomla



Joomla bug 30274

Results: Joomla



Several similar examples:

- E_{cause} **accumulates data items**
- E_{effect} **processes all of them and has unbounded cost**

Summary: GUI Testing

- Automated system-level testing
- Black-box and white-box approaches to explore huge search space
  - Artemis: Whitebox
  - SwiftHand and EventBreak: Mostly blackbox
- Different test oracles possible
  - Application crashes (robustness testing)
  - Consistency criterion, e.g., HTML validation
  - Responsiveness (performance testing)