StarCharter:
A Tool for Mining Concept Recipes

Steven She

ECE750-T05: Static Analysis for Software Engineering

November 28st, 2008

Outline

What is StarCharter?

Makes a recipe

describing framework usage

using examples.

What is StarCharter?

Makes a recipe
describing framework usage
using examples.

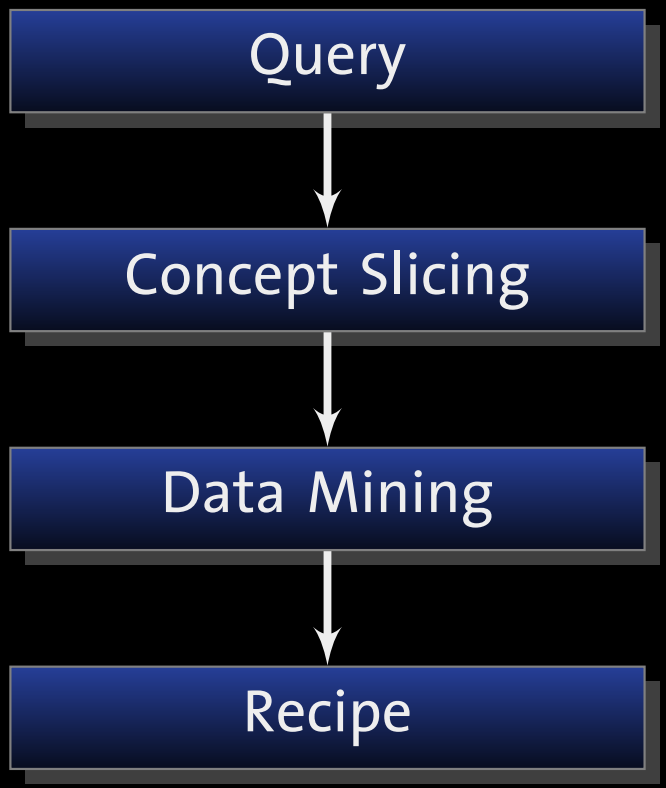
What is StarCharter?

Makes a recipe
describing framework usage
using examples.

What sets StarCharter apart?

- Constructs a single recipe.
- Recipe is a probabilistic model.
- Used for tool support or framework modeling.
- Address usage of white-box frameworks.

Overview



Provide hints for finding concept events.

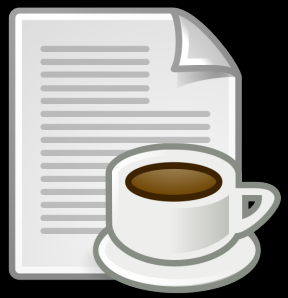
Find related events.

Find usage patterns among multiple examples.

Output a single probabilistic model describing usage.

Concept Slicing

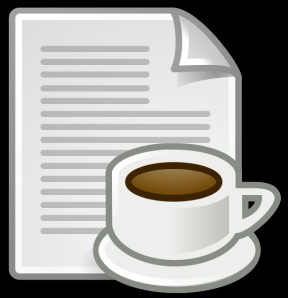
Reduce an example to a minimal set of statements that retains the specified behaviour.



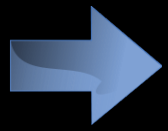
Source code

Concept Slicing

Reduce an example to a minimal set of statements that retains the specified behaviour.



Source code



Concept Slicer with Query

Concept Slicing

Reduce an example to a minimal set of statements that retains the specified behaviour.

Source code

Concept Slicer with Query

Relevant Events

Forward Slice

```
1  public class HelloWorld extends JApplet {
2
3    public static final String HELLO_WORLD = "Hello World!";
4    Image smiley;
5
6    public HelloWorld() {
7      smiley = getImage(getCodeBase(), "images/smile.gif");
8      Container contentPane = getContentPane();
9      JLabel hwLabel = new JLabel(HELLO_WORLD);
10     contentPane.add(hwLabel);
11     play(getCodeBase(), "audio/woohoo.wav");
12   }
13 }
```

Forward Slice

```
1  public class HelloWorld extends JApplet {
2
3    public static final String HELLO_WORLD = "Hello World!";
4    Image smiley;
5
6    public HelloWorld() {
7      smiley = getImage(getCodeBase(), "images/smile.gif");
8      Container contentPane = getContentPane();
9      JLabel hwLabel = new JLabel(HELLO_WORLD);
10     contentPane.add(hwLabel);
11     play(getCodeBase(), "audio/woohoo.wav");
12   }
13 }
```

Backward Slice

```
1  public class HelloWorld extends JApplet {
2
3    public static final String HELLO_WORLD = "Hello World!";
4    Image smiley;
5
6    public HelloWorld() {
7      smiley = getImage(getCodeBase(), "images/smile.gif");
8      Container contentPane = getContentPane();
9      JLabel hwLabel = new JLabel(HELLO_WORLD);
10     contentPane.add(hwLabel);
11     play(getCodeBase(), "audio/woohoo.wav");
12   }
13 }
```

Backward Slice

```
1  public class HelloWorld extends JApplet {
2
3    public static final String HELLO_WORLD = "Hello World!";
4    Image smiley;
5
6    public HelloWorld() {
7      smiley = getImage(getCodeBase(), "images/smile.gif");
8      Container contentPane = getContentPane();
9      JLabel hwLabel = new JLabel(HELLO_WORLD);
10     contentPane.add(hwLabel);
11     play(getCodeBase(), "audio/woohoo.wav");
12   }
13 }
```

Unrelated Statements

```
1  public class HelloWorld extends JApplet {
2
3    public static final String HELLO_WORLD = "Hello␣World!";
4    Image smiley;
5
6    public HelloWorld() {
7      smiley = getImage(getCodeBase(), "images/smile.gif");
8      Container contentPane = getContentPane();
9      JLabel hwLabel = new JLabel(HELLO_WORLD);
10     contentPane.add(hwLabel);
11     play(getCodeBase(), "audio/woohoo.wav");
12   }
13 }
```

Framework Slicing

Framework: javax.swing.*

```
1  public class HelloWorldPresentation extends JApplet {
2
3    public static final String HELLO_WORLD = "Hello World!";
4
5    public HelloWorldPresentation() {
6      Container contentPane = getContentPane();
7      JLabel hwLabel = new JLabel(HELLO_WORLD);
8      StringBuilder builder = new StringBuilder(HELLO_WORLD);
9      Formatter fmt = new Formatter(builder);
10     fmt.format("%s %s", HELLO_WORLD, HELLO_WORLD);
11     contentPane.add(hwLabel);
12   }
13 }
```

Framework Slicing

Framework: javax.swing.*

```
1  public class HelloWorldPresentation extends JApplet {
2
3    public static final String HELLO_WORLD = "Hello World!";
4
5    public HelloWorldPresentation() {
6      Container contentPane = getContentPane();
7      JLabel hwLabel = new JLabel(HELLO_WORLD);
8      StringBuilder builder = new StringBuilder(HELLO_WORLD);
9      Formatter fmt = new Formatter(builder);
10     fmt.format("%s %s", HELLO_WORLD, HELLO_WORLD);
11     contentPane.add(hwLabel);
12   }
13 }
```

Framework Slicing

Framework: javax.swing.*

```
1  public class HelloWorldPresentation extends JApplet {
2
3    public static final String HELLO_WORLD = "Hello World!";
4
5    public HelloWorldPresentation() {
6      Container contentPane = getContentPane();
7      JLabel hwLabel = new JLabel(HELLO_WORLD);
8      StringBuilder builder = new StringBuilder(HELLO_WORLD);
9      Formatter fmt = new Formatter(builder);
10     fmt.format("%s %s", HELLO_WORLD, HELLO_WORLD);
11     contentPane.add(hwLabel);
12   }
13 }
```

Framework Slicing

Framework: javax.swing.*

```
1  public class HelloWorldPresentation extends JApplet {
2
3    public static final String HELLO_WORLD = "Hello World!";
4
5    public HelloWorldPresentation() {
6      Container contentPane = getContentPane();
7      JLabel hwLabel = new JLabel(HELLO_WORLD);
8      StringBuilder builder = new StringBuilder(HELLO_WORLD);
9      Formatter fmt = new Formatter(builder);
10     fmt.format("%s %s", HELLO_WORLD, HELLO_WORLD);
11     contentPane.add(hwLabel);
12   }
13 }
```

Sliced Events

Example	Location	Event
HelloWorld	constructor	add(Component)
HelloWorld	constructor	**new** JLabel(String)
HelloWorld	constructor	getContentPane()

Store both the event and its *location*.

Concept Slicing

Source code

Concept Slicer with Query

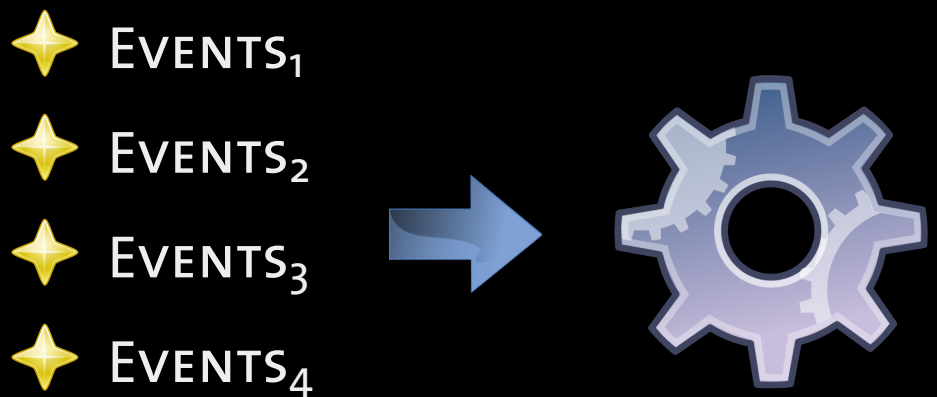
Relevant Events

Creating the Recipe

- Events_1
- Events_2
- Events_3
- Events_4

Sliced Events

Creating the Recipe



Creating the Recipe

Feature Model Mining

Mine for probabilistic expressions called *Association Rules*.

Feature Model Mining

Mine for probabilistic expressions called *Association Rules*.

Association Rule	conf
calls:getContentPane() $\Rightarrow$ *calls*:add(Component)	50%

Feature Model Mining

Mine for probabilistic expressions called *Association Rules*.

Association Rule	conf
calls:getContentPane() ⇒ *calls*:add(Component)	50%
new:Label ⇒ *calls*:add(Component)	100%

Feature Model Mining

Mine for probabilistic expressions called *Association Rules*.

Association Rule	conf
calls:getContentPane() $\Rightarrow$ *calls*:add(Component)	50%
new:Label $\Rightarrow$ *calls*:add(Component)	100%
Hello World $\Rightarrow$ paint $\vee$ constructor	100%

Feature Model Mining

Mine for probabilistic expressions called *Association Rules*.

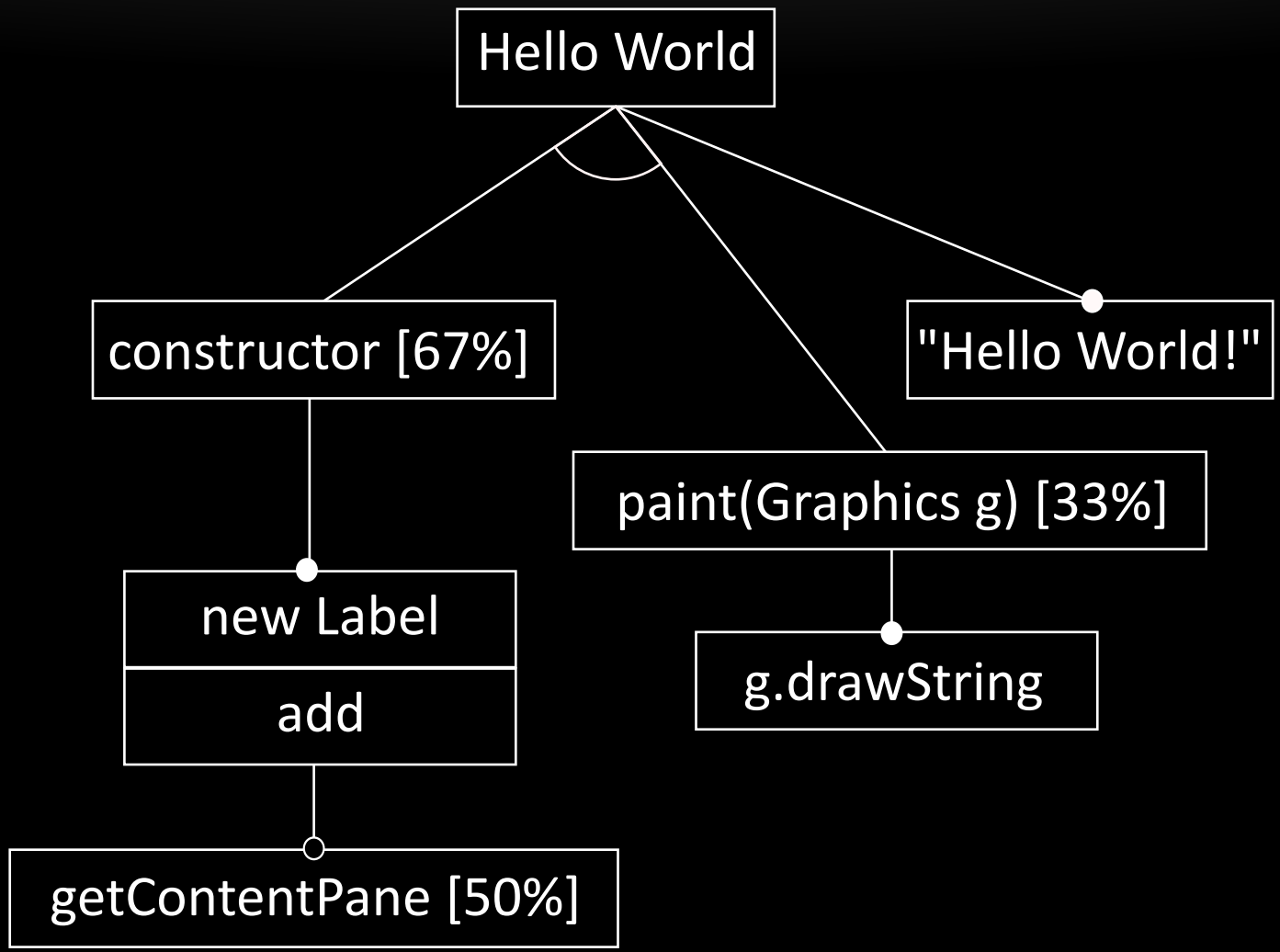
Association Rule	conf
calls:getContentPane() $\Rightarrow$ *calls*:add(Component)	50%
new:Label $\Rightarrow$ *calls*:add(Component)	100%
Hello World $\Rightarrow$ paint $\vee$ constructor	100%
Hello World $\Rightarrow$ constructor	67%

Feature Model Mining

Mine for probabilistic expressions called *Association Rules*.

Association Rule	conf
calls:getContentPane() $\Rightarrow$ *calls*:add(Component)	50%
new:Label $\Rightarrow$ *calls*:add(Component)	100%
Hello World $\Rightarrow$ paint $\vee$ constructor	100%
Hello World $\Rightarrow$ constructor	67%
paint $\Rightarrow \neg$ constructor	100%

Mined Recipe



Step 1: Input Query

```
public class HelloWorldPresentation extends JApplet {
  public static final String HELLO_WORLD = "Hello World!";
  public HelloWorldPresentation() {
    Container contentPane = getContentPane();
    JLabel hwLabel = new JLabel(HELLO_WORLD);
    StringBuilder builder = new StringBuilder(HELLO_WORLD);
    Formatter fmt = new Formatter(builder);
    fmt.format("%s %s", HELLO_WORLD, HELLO_WORLD);
    contentPane.add(hwLabel);
  }
}
```

- User *stars* a line of code in a framework application.
- **Extension:** Possible to star several statements.

Step 1: Input Query

```
public class HelloWorldPresentation extends JApplet {
  public static final String HELLO_WORLD = "Hello World!";
  public HelloWorldPresentation() {
    Container contentPane = getContentPane();
    JLabel hwLabel = new JLabel(HELLO_WORLD);
    StringBuilder builder = new StringBuilder(HELLO_WORLD);
    Formatter fmt = new Formatter(builder);
    fmt.format("%s %s", HELLO_WORLD, HELLO_WORLD);
    contentPane.add(hwLabel);
  }
}
```

- User *stars* a line of code in a framework application.
- **Extension:** Possible to star several statements.

Step 2: Find Related Usages

- Search the *project* for other instances of the starred code.
- Scope currently restricted to a **single class**.
- Use the Eclipse JDT SearchEngine.

Step 2: Find Related Usages

Calls: References to calls of the same name.

```
class A {
  ...
  Formatter x = ...;
  x.format("...", x, y, z);
  ...
}
```

```
class B {
  ...
  Formatter y = ...;
  y.format("Hello␣%s!", x);
  ...
}
```

Step 2: Find Related Usages

Variables: Assigned the same type or literal.

```
class A {
  ...
  JLabel label =
    new JLabel("Hello World!");
  ...
}
```

```
class B {
  ...
  JLabel label =
    new JLabel("Test", LEFT);
  ...
}
```

Step 3: Slice Related Expression

Input to Slicer: $\langle expression, class \rangle$

- Construct a *dependency graph* for each class.
- **Nodes** are expressions.
- **Edges** are *data dependence* and *control dependence*.
- Reachability determines slice.

EXAMPLE.

Step 3: Slice Related Expression

Input to Slicer: $\langle expression, class \rangle$

- Construct a *dependency graph* for each class.
- **Nodes** are expressions.
- **Edges** are *data dependence* and *control dependence*.
- Reachability determines slice.

EXAMPLE.

Step 4: Data Mining

One Slice *is* One Configuration

- Each expression in slice is a feature.
- Expressions in the slice are unordered.
- Also mine on structural properties.

Uses for the Tool

1. An enhanced **search for references** in Eclipse.

   Ranks usages by comparing a related usage with the recipe.
   Similar to PARSEWeb's clustering.

2. The recipe is a model.

   The recipe can be used for modelling FSMLs.

3. Recipe as an expert system.

   Using Herman's tool, it can act like FrUiT.

Uses for the Tool

1. An enhanced **search for references** in Eclipse.

   Ranks usages by comparing a related usage with the recipe.
   Similar to PARSEWeb's clustering.

2. The recipe is a model.

   The recipe can be used for modelling FSMLs.

3. Recipe as an expert system.

   Using Herman's tool, it can act like FrUiT.

Uses for the Tool

1. An enhanced **search for references** in Eclipse.

   Ranks usages by comparing a related usage with the recipe.
   Similar to PARSEWeb's clustering.

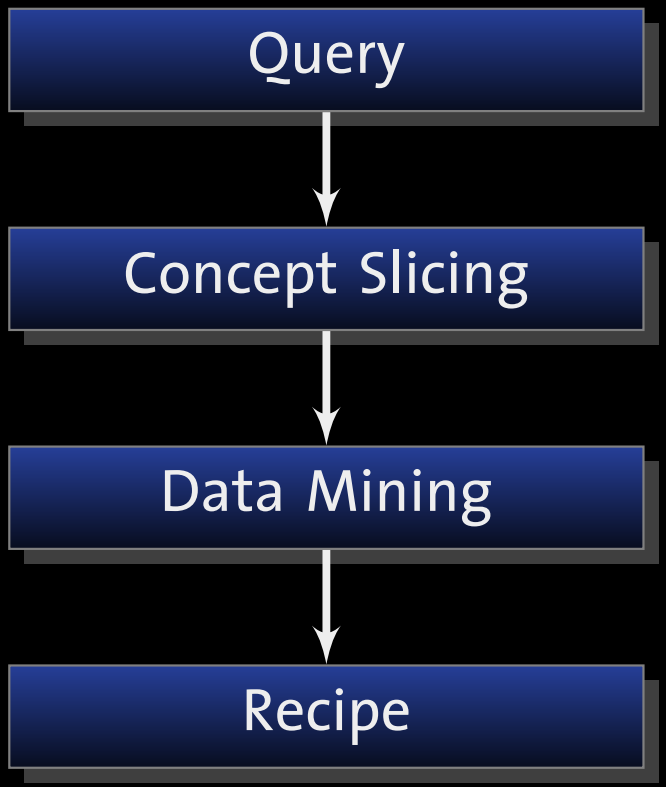
2. The recipe is a model.

   The recipe can be used for modelling FSMLs.

3. Recipe as an expert system.

   Using Herman's tool, it can act like FrUiT.

Conclusions



- Constructs a **recipe** from a set of examples.
- **Slicing** finds related statements given a query.
- **Data mining** merges sliced events from each example into a recipe.

Future Work

- Adding control dependence.
- Fuzzy matching for literals.
- Semantics for queries.
- When is concept slicing useful?