Dynamic and Interactive Program Synthesis

We have developed a new technique for synthesizing code that is:

- Dynamic
- Easy-to-use
- Interactive

Our algorithms generate and evaluate code at runtime and hence can synthesize real-world Java code that utilizes nearly the entire language, including reflection, native calls, and other advanced language features.

Overview

Users can provide a specification of the desired behavior in a dynamic state of an incomplete program, allowing concrete reasoning. This partial dynamic specification, or `pdSpec`, is used to synthesize candidate code that is shown to the user, who may then give further `pdSpecs` in different program states to interactively refine it.

Users may also express domain knowledge by writing a code `skeleton` to shape the search space. We supplement this information by first showing users more likely candidates, which we have computed by analyzing millions of lines of real-world programs.

Motivation

We have developed a new technique for synthesizing code that is:

- Dynamic
- Easy-to-use
- Interactive

Our algorithms generate and evaluate code at runtime and hence can synthesize real-world Java code that utilizes nearly the entire language, including reflection, native calls, and other advanced language features.

Example

Imagine that we are writing GUI code to handle clicks on data displayed in a tree. We have written the code below, but we are not sure how to find the actual element the user clicked. Moreover, we do not even know what type it is.

```java
private static final class ClickHandler extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        if (tree.isEmpty()) return;
        JMenuItem clicked = null; // what should this be?
        handleClick(clicked);
    }
}
```

Even though we do not know the type of the expression we want, we can run the code and click on a specific element in the tree. Knowing that in Java most objects have a `toString` method that gives a String representation of their value, we can click on an element labeled “Tux” and give the `pdSpec`

```java
clicked.toString().contains("Tux")
```

where `clicked` is the value of the clicked element after the synthesis.

Results

We have been able to search many statements quickly and synthesize up to ten lines of code at a time.

Conclusions

We have developed a program synthesis tool that is dynamic, easy-to-use, and interactive and that can synthesize code that uses arbitrary libraries.

Our open-source Eclipse plugin for Java is available at:

http://www.cs.berkeley.edu/~joel/codehint/