State of Research

Denis Aleshin

August 12, 2009

1 CircuitSimulatorUI

This is the main front-end that we have worked with in order to interface with the rest of the project. This interface can be used to work with, train, and test most of the recognizers and groupers we have, as well as contains the refiner project.

1.1 Bugs and Issues

For the sake of productivity, we concentrated on the functionality of the circuit panel, meaning that the other panels do not necessarily work. One more (not completely important, but aesthetic) bug is that the Sketch and InkSketch objects in the panel do not get synchronized by their displacement... this causes some jumping around when you draw on a loaded file. The old code for doing this was breaking the recognition proccess, so we scrapped it.

1.2 To Do

The only recognizer not completely working with the CircuitSimulator is the ImageAligner. This is because when the unknown template has 10 strokes and the known template has 6 strokes, there are 10P6 possible ways to assign the stroke correlations. Since each correlation takes a few seconds to check, this is simply way too inefficient. The code is all in place, and Eric is working on a way to prune the search space to make it useable.

The ImageAligner is the recognizer that is supposed to have comparable scores for partial shapes, and shapes with extra strokes. It also returns all of the errors it finds. This can be taken advantage of in the refiner.

1.3 How to Use

The main CircuitSimulatorUI has a circuitPanel object, which represents the drawing surface. We have two modules- RecognitionManager and DisplayManager that subscribe to this panel. Within the RecognitionManager are the recognizers, groupers and the refiner object, as well as the ContextDomain. Each of these is plugged in through an abstract class, so to edit a specific recognizer, for example, start with the ShapeRecognizer project which contains the abstract class for recognizers.

All of the functionality (including testing) is accessible through the tools menu.

2 ContextDomain

This class is responsible for abstracting away the domain information from our recognition proccess. As can be seen from the abstract class, any Domain object must provide methods for viewing all of the labels and classes within a system, and provide code for determening adjacency and connectedness within a sketch.

2.1 Bugs and Issues

There are no known bugs in this file.

2.2 To Do

Determening connections is a very important part of our code, and needs to be improved. The simplest way of doing that right now would be to remove the assumption that a wire is connected on both sides.

2.3 How to Use

This is plugged in to the CircuitSimulatorUI project.

3 PrimitiveRecognizer

This is a fully functional front-end to the new grouping techniques. Rather than messing around with CircuitSimulator and getting it more cluttered, I made a simple frontend on the side that can be used to play around and test these items.

3.1 Bugs and Issues

There is no output formatting for the testing. I put a breakpoint at the end of the testing function and copy the testResults out of the 'watch' window. It's hacky, but it's quick and functional. A nice thing to do would be to implement an actual output function.

3.2 To Do

The primitive class in Sketch only supports lines, arcs and circles. Most of the arcs we see in our sketches are actually parabolas. Implementing a parabola object would greately improve recognition!

The grouper that uses primitive objects doesn't support all of the rules we see in shapes. Merging lines that form continuous primitive objects would be great for incorporating touchup strokes. Also a method for adding Not-bubbles onto NOT gates.

3.3 How to Use

The main form contains a sketchPanel which contains the sketch. The DisplayManager and RecognitionManager subscribe to this panel. The recognition Manager contains the primitive classifier as well as the two groupers implemented so far. All of the functionality is accessible through the toolbar on the main form.