Literate Programming Proposal

Kevin Carmona-Murphy

March 4, 2016

Abstract

roadnet is a proposed Ruby based compiler for converting an XML description of a road network into an HTML/CSS visual representation. This document outlines the methodology, languages and frameworks used, as well as implenetation and testing steps.

1 Summary

roadnet is the name given to a new tool which will allow for the generation of simple graphical road networks using a specially developed road topology language. This tool will convert an XML file containing the hierarchy of the road network into an HTML page displaying the network from a bird's-eye viewpoint. Different road types will be supported, and things like number of lanes, lane width, road length, and number of intersections will be specifiable.

1.1 Problem Inspiration

The inspiration for the tool stems from a pass-time I once cherished as a child. I would often spend hours sketching small neighbourhoods onto a large piece of drafting paper. Attention to detail was especially evident in how I drew the roads; from a bird's eye perspective, lane widths were meticulously calculated and enforced with a ruler through the entirety of my drawings.

Attempting to bring rekindle that hobby and bring it into the digital world is the motivation for such a tool. While of no practical purpose for transportation engineers, it offers an academic and practical insight into the fundamentals of compiler design using modern, high-level languages.

2 Languges & Tools

The compiler will transform an XML file into an HTML file with embedded CSS. XML was chosen as the source because it allows one the freedom to write nested hierarchies with tags that are specified by attributes. This is important given that a road topology is a network of elements that are often repeated and are always connected to one another. An XML file assures that each element has a parent, just like each segment of road generally intersects with another.

HTML will be used as the output representation because it is a very widely available markup language which is compatible across platforms. With embedded CSS, **roadnet** will be able to create a road network with roads of different lengths, lane widths, etc.

The compiler itself will be written in Ruby in order to take advantage of the excellent Nokogiri gem which makes the task of parsing elements in XML and HTML files a breeze. In the implementation, Nokogiri will parse the source XML file and create a token tree, which will then be manipulated by Ruby.

Finally, noweb is the Literate Programming tool being used in the development of **roadnet**. noweb is language-agnostic, and outputs to either LaTeX or HTML.

3 Resources

Various resources will be consulted over the course of the development of the tool. Primarly, the Internet will be of greatest help while I figure out the workings of Nokogiri and Ruby. Once I'm able to retrieve a syntax tree is from a file, I'll need to shift my focus and consult the notes on context-free grammars and EBNF chains from class in order to create rules for the semantics validator. Finally, my trusty brain, let's hope, will pave the way for an efficient implementation of the embedded CSS styles that will be used to give structure to the outputted HTML.

Resources already consulted:

- https://github.com/sparklemotion/nokogiri
- http://programmers.stackexchange.com/questions/165543/how-to-write-a-very-basic-compiler
- https://en.wikibooks.org/wiki/LaTeX
- http://www.literateprogramming.com/noweb_hacker.pdf

Resources to be consulted:

- The Dragon Book
- http://ruby-doc.org/
- http://www.nokogiri.org/tutorials/parsing_an_html_xml_document.html

4 Implementation & Testing

4.1 XML example

The below XML code shows a very rudimentary code structure that describes a road network of an intersection that is flanked by a road, an avenue, and a street. The street travels for a bit (stretch) followed by an intersection crossed by another street. The same street then continues for a little bit longer.

```
<network>
        <intersection>
                <road>
                </road>
                <avenue>
                </avenue>
                <street>
                        <stretch>
                        </stretch>
                        <intersection>
                                <street>
                                </street>
                        </intersection>
                        <stretch>
                        </stretch>
                </street>
        </intersection>
</network>
```

Of course, attributes will be used to specify the length of the stretches, as well as other features of the street such as number of lanes and lane width. Attributes for an intersection will specify road angle offsets, and turning radii, among other things. The code expects that there is at least one root intersection. One major limitation of the XML representation is that all roads are diverging - that is no cycles can be formed in the network given the current design.

4.2 Verification

Testing can be done using a tool such as ANTLR. Otherwise, simple test tokens will have to be produced by the parser and validated before moving onto building the HTML/CSS output engine.