CPSC 498 Proposal: RBTEX

Steven Rosendahl

1 Abstract

Modern LATEX distributions include a tool called `lualatex` that allows users to dynamically produce content via use of Lua code. Unfortunately, the Lua standard libraries do not have as much functionality as other popular scripting languages, such as Ruby. The goal of this project is to incorporate Ruby into LATEX in a manner similar to `lualatex` , but with the power and simplicity of Ruby over Lua.

One of the main benefits of having the ability to dynamically produce LATEX code is that difficult mathematical problems can be solved using a high level language. Ruby's syntax is simplistic enough to make up for the fact that the TEX syntax is verbose; a gem will be provided to perform some mathematical operations and present the result in LATEX.

2 Motivation and Description

2.1 LATEXand Ruby

The current `lualatex` specification allows users to use several environments for writing and running Lua scripts. In addition, `lualatex` provides a built in library called `tex` that allows output to be printed straight to the LATEX document. The library, called RBTEX, will provide similar functionality through a gem called `rbtex`. In addition, the entire Ruby standard library will be available for use; RBTEX documents that need to interact directly with the system will most likely need to be compiled using the `--shell-escape` flag.

To use the library, users will need to have a Ruby version in the path. The code will be pre-processed, and inserted directly into the TEX code before `pdflatex` is called on the document. In addition, users will be provided with several ways in which to interact with Ruby from the TEX environment:

1. `rbtex{}`: This command will provide a way to write multiple lines of Ruby code inside the LATEX document. Any functions defined in this section will be globally defined, so they can be called in the `inrbtex{}` environment and in other `rbtex{}` environments. This code will be pulled verbatim from the LATEX file during the pre-processing stage.

2. `frbtex{}`: It may sometimes be convenient for a user to write an external Ruby file and call it from the LATEX file, rather than writing the code. The `frbtex{}` macro will allow an external script to be loaded into the document. The pre-processor will copy the provided Ruby file into the LATEX document, and will assume that all modules, classes, functions, and variables are globally defined.

The library will come with a program called `rbtex` that complies the provided LATEX document, much like the `luatex` command.

The program will work in four steps. It will first pre-process the TEX file, scanning for the appropriate environments. The ruby code will be ripped out and stored in an `.aux` file. The next step will order the code in the `.aux` file, and attempt to produce a `.rb` file from the provided LATEX document. In the third step, the code in the `.rb` file will be run using the `ruby` command specified in the user's `$PATH` variable. Finally, the post-processor will capture any output specified by the `tex` module in the `rbtex` gem, and place it into the LATEX document. From there, `pdflatex` will take over.

The pre-processor and post-processor will be written in C++ to allow for quick speed when parsing out the Ruby code. Shell script (UNIX) and Batch files (Win) will be provided for calling the program. Standard `pdflatex` flags will be available (they will be simply passed to the `pdflatex` command at the appropriate time). The final version of the program will be accessible through the shell command `rbtex tex_file.tex` and the Windows equivalent.

2.2 Ruby and Math

Ruby includes several mathematical utilities that are sufficient to solve simple math problems. However, Ruby lacks the power to solve higher level problems, such as Ordinary Differential Equations and even Partial Differential Equations. This package will include a gem called `rbtexm` that will have several features geared towards mathematics:

1. ODE Solver: The gem will include a module that can solve a wide variety of first and second order ODE's. If it runs into an ODE that it cannot solve, it will gracefully exit.
2. PDE Solver: This module will allow for simple first and second order PDE's to be solved. It will have functionality to solve only homogeneous PDE's.
3. Integration: This module will allow for integration of functions, when possible.
4. Differentiation: A simple module to differentiate functions.
5. Logic: This module will be able to solve binary logic problems. It will return a truth table that has all possible outcomes.

Time permitting, more mathematical modules will be added. All the models will provide a way to output LaTeX code for use in the document; however, the math module should be able to be used independent of LaTeX. The methods available will take in strings of LaTeX code. The gem will either contain a small LaTeX parser, or will include a third party one.

3 Tentative Schedule

January 19	Proposal and initial git setup
January 26	Building C++ preprocessor
February 2	Building C++ postprocessor
February 9	Building `rbtex` gem
February 16	Begin writing `rbtexm` gem
February 23	Build small LaTeX parser
March 1	Build calculus related utilities
March 8	Build calculus related utiltites
March 15	Build logic related utilities
March 22	Build any extra mathematical utilties
March 29	Testing
April 5	Packaging for Windows, OS X, and Linux
April 12	Extra space to be used as needed
April 19	Extra space to be used as needed