FUNCTIONAL PROGRAMMING CS-IS-2010-1 MIDTERM EVALUATION PROJECT REPORT

Haskell Web Scraper

Saptarishi Dhanuka

Ashoka University

Contents

1 Problem Statement and Requirements

Assigned Project Statement

Develop a scraper using Haskell to extract text and code snippets separately.

1. **Input**: Scrape the text and code snippets from the given text source
2. **Output**: A Word document containing the text and `.txt` file containing the code.
3. **Method**: Write the algorithm to scrape (you can use the `tagsoup` library) and all the input-output facilities using Haskell. Do not use any other language.

1.1 Problem Description

The given web page is made of text and code snippets, which we need to scrape and extract separately into a `.docx` file containing the text portions and a `.txt` file which has the code snippets.
For this, we need to fetch the given web page, parse and analyze its HTML structure to identify the HTML tags of the code snippets and the tags of the rest of the text, so that we can effectively separate them into different documents.

1.2 Requirements

1. The scraper will be written entirely in Haskell
2. The scraper will get all text of the given page and write it into a `.docx` file.
3. The scraper will get all the code snippets of the given page and write it into a `.txt` file.

2 Specifications

1. The scraper will use Haskell HTTP libraries for fetching the contents of the given web page.
2. The scraper will separate the code snippets from the rest of the textual content using the `tagsoup` library
3. The scraper will write the text into a Word document and the code snippets into a `.txt` file using the `tagsoup` and `pandoc` libraries.

1. The scraper will mainly utilise the `tagsoup` and `scalpel` libraries for the parsing and extraction of the HTML content and separation of the text and code snippets
2. The scraper will use the HTTP libraries for fetching the given web page
3. The scraper will use the standard module `Prelude` for writing the code snippets data into a `.txt` file
4. The scraper will use `Pandoc` library for writing the textual data into a `.docx` file
5. **Limitations**
   a) Since the HTML structure of different web pages can vary, it is not necessary that this particular scraper will work for all web pages. It is designed specifically for the given page and may work for some other pages. But no generalisation can be made about the correctness of its text and code extraction for other pages.
   b) Moreover, it will not necessarily work for web pages with malformed HTML or different structure.
   c) It is not designed to be robust to design changes, which is in line with the rule stated on the `tagsoup` library's documentation example. If the site's HTML structure changes, for instance if the code snippets change from being enclosed in `<pre>` tags to `<code>` tags, then the scraper will not be able to separate out the code from the text and extract them accurately.

3 Design and Architecture

4 Choice of Tools, Platforms, and Languages

5 Test Plan

6 Prototype Implementation Details

7 Plan for Completion