Assignment1 Report

Saswat Kumar Pujari
Entry No.- 2019MCS2571

HTML TO LaTeX CONVERTER

September 1, 2019

Introduction

This is a detailed report on Assignment 1 of course-COP701. The Assignment is to make a **"HTML TO LaTeX CONVERTER"**. The converter takes a .html file as input and outputs a .tex file. All the coding has been done in C++ language and the grammar for the parser has been produced by using Flex and Bison tools.

The assignment is divided into the following parts:

1. Generation of HTML Tokens.
2. Production of grammar to parse the tokens.
3. Parsing of tokens using grammar to construct Abstract Syntax Tree for HTML.
4. Conversion of HTML Ast to LaTeX Ast.
5. Conversion of LaTeX Ast to LaTeX code.

1 Steps

1.1 Lexer and Parser

A lexer has been written using flex and bison tools. The lex file contains different regular expressions that are needed to convert the HTML code into a string of HTML tokens. These tokens are then parsed using the grammar defined in a yacc file which contains definitions about various grammar corresponding to different terminals and non-terminals.

The terminals and non-terminals have been divided into four different types:

- **tag**: It is of char* type which basically consists of the terminals derived from the html tags.

- **v**: It is also char* type which consists of the various terminals that hold the various contents of the html tags.
- **s**: It is of struct astnode* type which consists of all the non-terminals of the parser that generate individual ast nodes based on their productions.
- **t**: It is a pointer type to vector of ast nodes of all the the children nodes of any particular ast node corresponding to a particular non-terminal.

The html code is parsed and the various types of tokens and non-terminals are generated which are analyzed meaningfully using the grammar productions to generate an abstract syntax tree.

1.2 HTML Abstract Syntax Tree

The parser analyzes the code and produces an abstract syntax tree which consists of nodes having the following structure:

struct astNode{
string nodeType;
vector<astNode*>attribute;
string value;
vector<astNode*>child;
}

The AST.h file contains the given structure of an HTML Ast node. The nodes in the AST correspond to the various HTML tags. The attributes are defined as follows:

- **nodeType**: It specifies the type of a node which is basically the HMTL tag it corresponds to.
- **attribute**: It is a vector containing the various nodes corresponding to the attributes of a particular HTML tag. For ex- The astnode corresponding to the <a>tag consists of astnode of href attribute in its attribute vector.
- **value**: It consists of the value associated with any node.
- **child**: It is a vector of astnodes which holds the astnodes of html tags present inside a particular html tag.

A function newAst() is called for every HTML tag which makes an astnode for the tag with nodeType that is passed as argument to the function. The astnodes of various tags present inside a particular html tag are built and are kept as vector of child nodes of that tag inside child vector.

1.3 LaTeX Abstract Syntax Tree

The LateX AST also consists of the similar node structure as that of the HTML astnode:
struct latexNode{
string nodeType;
vector<latexNode*>attribute;
string value;
vector<latexNode*>child;
}

After the HTML tree is built, the convert() function is called HTML root node is passed as parameter which basically is used to convert every HTML astnode to corresponding latex node. It makes a latex astnode corresponding to every html astnode and then calls the addchildren() function with the html node and new latex nodes as parameters to build new latex children nodes corresponding to every child node of that html node. As a result, the whole LaTeX AST is created by recursively creating new latex nodes for all the children present inside the vector of a html node.

1.4 Conversion into LaTeX code

The Root node of the LaTeX ast is provided as input to the function produceLatexCode() along with a latex file pointer which checks the nodeType of every latex node and based on the same produces corresponding latex code along with code for all its children by calling function produceChildren(). The produceChildren() takes a latex node as parameter and produces code for every children node of a latex astnode. Finally the whole LaTeX code is written to the output tex file.