COP701(Assignment 1) : HTML to LaTeX Converter

Het Shaileshkumar Patel(2019MCS2562)

September 1, 2019

1 Introduction :

In this submission I have created HTML to Latex converter in python language using PLY library for Lex and YACC implementation (This report was written in html and then converted using my code to LaTeX to generate pdf.)

In this submission I have created five major files.

1. **run.py** : Our main file which will be called from the shell file with file arguments to generate output file
2. **LEX_ply.py** : Which will give us our tokenizer(i.e. it will tokenize our input file)
3. **YACC_ply.py** : Which will contain our parser generated using CFG and this file will generate HTML's ast
4. **node.py** : This file will contain declaration of node representation which is used to create AST
5. **createLatex.py** : This file will contain necessary code to map HTML ast to equivalent LaTeX ast and generate output file using LaTeX ast

2 Overview of LEX and YACC operations:

2.1 LEX :

- Here the lex file will read input file and generate the stream of token which will be consumed by YACC.
- Every token generated will have a type and a value. And every types of tokens that can be generated should be written in a tuple named tokens in this file.

- Here in my LEX file I haven't written matching expression for every tag but instead I have written functions for opening OPENING_TAG, CLOSING_TAG and SINGLE_TAG which will generate token in such a way that their type will be represented by tag's name with some prefix according to the type. e.g. html tag's opening tag's token will have type as HTML_O ,html tag's ending tag type will be HTML_E as it is exit tag of html tag and hence E and br will have type BR_S as it is single tag

- If there is any unknown tag that we don't know in advance and we haven't added its type in tokens tuple then it will give its type as OPENING_TAG, CLOSING_TAG, SINGLE_TAG accordingly. And while conversion I have written that part as it is. I have done this to make sure that there are no lexical ot syntax error if we end up having any new tag.

2.2 YACC :

- In yacc file I have written CFG required to build parser by YACC.

- I have written the grammer rules in the YACC file's beginnig as comment

3 Abstract Syntax Tree Structure :

- To implement AST I have created a new Node class in node.py . Every node will have the following three fields

  1. **type** [datatype:String] : This string will hold type of the node
  2. **attributes** [datatype:dictionary] : This field will hold the attributes reuired for the node in a dictionary. This can be used for several purposes such as I have stores value attribute in STRING type of node and while writing that to file I will get the string from the attributes' dictionary.
  3. **children** [datatype:list] : This field will hold the children of the current node. We are using this list as we can have multiple children.

4 HTML Abstract Syntax Tree generation :

- Here in YACC_ply.py file I have written function for every production in my grammer and p[0] will represent what our left side of the production will return from where it was expanded. Here in every production except the starting procduction I am returning a list of nodes.

- So in parent production when a terminal's production gets called it will return a list of nodes which I will add to that node's children using function defined in Node class.

- Suppose I have a production like **body_data : body_data P_O body_data P_E data** so here I will return **p[0]=sum([p[1],[Node('P',{},p[3])],p[5]],[])** as p[1] and p[3] are list of nodes and node(s) return by p[1] and p[5] will be the children of our p node whose children will be p[3]. And here sum function will take these three lists of nodes and will give us a single list of nodes which will be assigned to its parent

5 HTML AST to LaTeX AST Mapping

- In file createLatex.py I have written a function called mapHTMLastTo-LATEXast(root) which will take a root node as a parameter and will map HTML AST to LaTeX AST
- In that function I have created a mapping for every node type of HTML AST to LaTeX AST.
- Now I will convert the type of root node accordingly and then make recursive calls to its children as new roots and so on.

6 LaTeX output file generation

- I have written a function called createLatexFileFromLatexAst(node,file) in createLatex.py file. To generate output file we will call this function with root node of LaTeX AST and output file as a parameter to this function and we will get the specified output file.
- In that function I have written some other inner function which are reposnsible for conversion of every node type of LaTeX AST.
- First of all I will get the corrosponding function for a given node type using a dictionary in which we will have node types as keys & its corrosponding function as its value.
- After that we will call that function and In that function I have written some mapping in LaTeX file and then I will recursively call this function on every chidren of this node. After that children list is finished I will write some post operations That are required in LaTeX such as {center}, closing a curly brace, etc.