COP 701 HTML to LaTeX Converter
2019MCS2574

Vivek Singh

September 1, 2019

1 Details of lex and yacc operations :

1.1 LEX file

- Have used **flex** as a tool for scanning the HTML file and generating tokens.
- The **lexer.l** file contains the code for lex operations.
- The lex file uses **caseless option** to tackle the case insensitivity of HTML. Have defined **spac, special words, text & greek** type of regex to store information in tags. Some tags have attributes in it hence separate tokens are generated for each attribute e.g for **a: name, src, title** etc. All HTML tags are tokenized passes to yacc file.

1.2 YACC file

- Have used **Bison** for parsing the file and creating equivalent LaTeX document.
- The **parser.y** file contains the code for yacc operations.
- Yacc file returns an AST to **ast.cpp** file. Yacc file uses union of char* s and struct node as types of each non terminals. Tokens are used as terminals.
- Yacc file contains **Context free grammar CFG** for rule definition.
- ast.h file is included which contains information about the structure of AST node. Function **makenode** is used to create a new node. Function **addchildren** takes two node pointer as argument and makes second node as a child of the first node. Root is assigned to **doc_ start** which is starting production in grammar.

2 AST structure

- AST structure is defined in **ast.h file.**
- AST structure contains following in each node
  1. **nodetype** to store the type of node which will help in recognition of each node.
  2. **string data** to store the data
  3. **vector children** , stores node which is used as children in Abstract Syntax tree.
  4. **vector attribute** , is used to store attributes of HTML tags, in pair wise mapping.
  5. **vector tdata** , to store list of information of a node .

3 Translating the AST

- After successful creation of AST from Grammar rules of yacc file we traverse the AST .
- Each Html tag data is stored in node with a nodetype for recognition while traversing AST.
- Tree traversal is DFS type traveral i.e from left to right.
- Mapping from each html node is done to equivalent LaTeX tags using a start map and end map for each AST node.

4 Programming Language used

- **C++ 11** is used for compiling lex and yacc file
- **Flex** is used for lex operations.
- **Bison** is used for yacc operations.
- **lexer.l** is LEX file
- **parser.y** is YACC file.
- **ast.h** is header file
- **ast.cpp** contains main function for tree traversal and conversion.
- **run.sh** is shell file that takes two arguments. First input Second output.