Report for HTML-LaTeX convertor by Pratik Karia (2019MCS2568)

September 1, 2019

1 INTRODUCTION (Files Description)

Total 5 files are used to convert HTML(.html) to corresponding Latex(.tex) file. All these 5 files are executed using run.sh file which contains appropriate flex and bison commands to run the below mentioned files.

- **lexical_analyzer.l** - The lexical analyzer file of the parser that detects all the tokens and returns them.
- **parser.y** - The yacc/ parser file where the grammar to parse the html file is written. This file also helps in creation of the AST for the html language
- **convert_AST.cpp** - The converter file that maps node types of AST of parser.l to corresponding in latex language.
- **common_header_file.h** - The common header file that contains the declaration of node structure, enum structure and other functions.
- **main.cpp** - The main file containing the main function and the traversal of the Abstract Syntax Tree

2 LEX and YACC

2.1 LEX

- The Lex file contains the regular expressions for the tags mentioned.
- Four of the tags have been also checked for attributes namely Anchor Tag, Font Tag, Table Tag and Image Tag
- Initially many regular expressions are defined that are used in all the other regular expressions

- Some of these are {punc} which defines some punctuations, {spac} which defines spaces and tabs and {word} which uses {spac}, {punc} and other special characters to form contents of a word token
- The lex file is also handled to consider all the other tags as a data apart from the tags mentioned so as to prevent parse error in case of unknown tags. The unknown tags are handled in grammar by considering it as a DATA, hence they are considered in the production for DATA and node of DATA is created and displayed in the output tex file.

2.2 YACC

- The Yacc file contains the grammar that is used to parse HTML tokens and generate the AST for the HTML language containing the mentioned tag
- The grammar contains mainly 2 parts. The first part contains grammar for the HEAD tag of HTML and the second part contains grammar for the BODY tag of HTML which consists of majority of tags.

  1. The HEAD grammar of HTML contains grammar for TITLE tag
  2. The BODY grammar of HTML contains a common production "body" which expands other subproductions each of which is for a particular tag

- Few greek symbols are also handled by creating a map from HTML to corresponding latex greek symbols. These are handled by handling the node of the GREEK symbols in AST.
- The grammar written to parse the HTML file is a Context Free Grammar
- The main.cpp file takes the input html file and calls yacc file. The yacc file takes the html code and tokenizes it using the lexer and then creates the nodes. The main.cpp has a function for traversal which traverses the AST created.

3 Abstract Syntax Tree

3.1 Contents of Abstract Syntax Tree Node

- **node_type [Type: integer]** - Contains one of the contents of an enum present in the header file. There is one node_type for each production in the grammar
- **data [Type: String]** - The data that can be stored with a node. This is mainly done to store simple text, greek symbols and html comments.

- **attributes [Type: String]** - This consists of attributes associated with HTML tag. This is mainly used in Anchor Tag, Image Tag, Font Tag and Table Tag.

- **children [Type: vector of nodes]** - This contains the children of each node.

3.2 Abstract Syntax Tree Working

- There are 41 enum types each of which is used to create node of a particular production

- Each node is created with a node_type and children which is a vector containing all the nodes created by the production called by it.

- There is a root node that is created at the start of HTML tag which represents the root of the tree.

- The traversal of the Abstract Syntax Tree created is done using Depth First Search(DFS) done in traversal_main function.

- This traversal creates a mapping of each node to corresponding map in LaTex and generates the output tex file

4 Programming Language Used

- C++ programming language is used to create the parser.

- Many of standard library structures like vector, map, stack are used to create the tree nodes and traverse the AST.