HTML To LaTeX Converter

Indian Institute of Technology, Delhi

Naman Agarwal
2019MCS2566

Programming Language Used: C++

List of Files

1. lextest.l
2. yacctest.y
3. main.cpp
4. ast.h
5. ast.cpp
6. treeconverter.h
7. treeconverter.cpp
8. latextreetostring.h
9. latextreetostring.cpp
10. makefile
11. run.sh

HTML tags considered:

- html
- head
- title
- body
- h1, h2, h3, h4
- p
- div
- a (only with href attribute)
- ul, ol, li, dl, dt, dd
- table (only with border attribute), tr, th, td, caption
- figure, figcaption, img (only with src, width and height attributes)
- br
- font (only with size attribute)
- center
- b, u, i, tt, small, strong, em

Lex File Function

This file contains all the rule for tokenization of .html file. It contains seven states which are *INITIAL, line, table, figure, l, font* and *comment.*

The tokenization starts in *INITIAL* state where it ignores all characters, white spaces, tabs, and newline characters and only return token to the yacc file and change states for certain specified HTML tags.
It ignores the <!DOCTYPE html> tag.
It goes to line state when it comes across any of the following tags.

- title
- h1, h2, h3, h4
- p
- li, dt, dd

If any state encounters "<!–" sequence it goes to the comment state where until "−>" sequence is encountered, all every thing is ignored.

In the *line* state, it tokenize the closing tags of all the tags that can have text between them. the list of these tags are:

- title
- p
- h1, h2, h3, h4
- a
- th, td, caption
- figcaption
- li, dd, dt

Text can only be placed in between these tags.
The text placed in attributes of *table, img, a* and *font* are tokenized in there respective states.

The things that can be parsed as a line are:

- Text (of course!)
- Some special characters, math operators and greek symbols
- Text formatters i.e. *b, i, u, em, tt, strong, smal* and *font* tags
- Position formatter i.e. *center* tag
- images i.e. *img* tag
- links i.e. *a* tag

You cannot give caption to an image in which is not placed in *figure* tag and *figure* tag cannot come in between any tag parsed as a line.

In *table* state, the lexer only tokenizes the tag which can come in between the *table* tag, which are *tr, td* and *th* tags, until it encounters the table closing tag after which it returns to the *INITIAL* state.

In *figure* state, the lexer only tokenizes what can come in between a *figure* tag, which is figure caption and image data. When it encounters the figure closing tag, it returns to *INITIAL* state.

In *l* state, the lexer only parses the *href* attribute of the *a* tag. And when the tag ends, it changes state to line.

In *font* state, the lexer only parses the *size* attribute of the *font* tag. And when the tag ends, it changes state to line.

Yacc File Function

This file contains the BNF grammer which specifies the order in which the tags can be placed in the .html file. If the order specified does not match with the token generated by the lexer, this file gives a parser error.

Here, the Abstract Syntax Tree (AST) is generated for the structure of the html file which is passed as input.

No node is generated by the parser for *div* tag. But it verifies that the tag is opened and closed properly.

Structure of AST

Each node of AST contains three fields namely *nodeType, data* and children.

The first field stores the type of node. It can take values like HTML, HEAD, TITLE, BODY, TABLE, etc. The second field stores data of the node if any. This is used to store the attribute values of *font, a, table* and *img* tags and the text of LINE node. The third field (as the name suggests) stores the pointer to the children of the nodes.

The BODY node can have H1, H2, H3, H4, P, BR, FIGURE, LIST, DLIST and TABLE type of nodes as its children nodes.
Similarly, TABLE can have TABLEROW, CAPTION nodes as its children and TABLEROW can have TABLEHEADER AND TABLEDATA nodes as its children.

Nodes that can have lines in between them like heading, paragraph, list items, etc. have a sequence of nodes as their children. This sequence is stores a node each for the opening and closing tag of tags like *b, i, u, em, font, a, image, center*, etc., and one node for the plain text in between any two nodes.

Translation of HTML AST to LaTeX AST

The LaTeX AST also contains node of the same type as the HTML AST.

Translation was done by traversing the AST and for each node of the HTML AST, a new node was created in the LaTeX AST which had a node type named similar to the syntax which could map to the html tags and so would create a pdf file that looks similar to the webpage generated by the html file.

Examples:

- The HEAD node and TITLE node were combined to create a Latex-Title node in Latex AST which included the children of the title node.
- All different types of closing nodes of textformatter types from html AST were changed to a CLOSE type as they would map to the same string which is "}"
- TABLE node in HTML AST had caption and TABLE as children, while Table node in latex has caption and Tabular node as its children and all rows are stored inside this TABULAR node.

EXTRA THINGS ADDED

1. You can choose to add border to the table. The border attribute in table can have value either '0' denoting no border or '1' denoting some border.
2. All special characters that can be directly typed from the keyboard can be printed.
3. Following math symbols:
   $\forall, \exists, \leq, \geq, \neq, \subset, \supset, \subseteq, \supseteq, \in, \vee, \wedge, \cup, \cap, \sum, \prod$
4. Following greek symbols:
   $\alpha, \beta, \gamma, \delta, \epsilon, \eta, \theta, \lambda, \mu, \phi, \chi, \omega, \sigma, \tau$,A,B, $\Gamma, \Delta, \Theta, \Lambda, \Xi, \Pi, \Sigma, \Omega, \iota$