COMP261 Parsing 1 of 4

Victoria
UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui
CAPITAL CITY UNIVERSITY

Parsing text

Making sense of "structured text":

- Compiling programs (javac, g++, Python etc)
- Rendering web sites (Chrome, Mozilla Firefox, etc)
- Processing database queries (PostgreSQL, MySQL, etc)

- Checking grammar
- Understanding natural language

What is “structured text”

```
<html>
<head><title>My Web Page</title></head>
<body>
<p>Thank you for viewing my silly site!</p>
</body>
</html>
```

```
DELETE FROM DomesticStudentsFor2012
     WHERE mark = ‘E’;
```

Not “structured text”

I KEEP six honest serving-men (they taught me all I knew);
Their names are What and Why and When and How and Where and Who.
I send them over land and sea, I send them east and west;
But after they have worked for me, I give them all a rest.

I let them rest from nine till five, for I am busy then,
As well as breakfast, lunch, and tea, for they are hungry men.
But different folk have different views; I know a person small—
She keeps ten million serving-men, who get no rest at all!

She sends ‘em abroad on her own affairs,
From the second she opens her eyes—
One million Hows, two million Wheres,
And seven million Whys!

The Elephant's child, Rudyard Kipling

What kind of text we really mean?

Text is “structured” if it can be described using a grammar:

- A **grammar** is a set of rules of a specific kind, for forming strings in a formal language.
- The rules describe how to form strings from the language’s alphabet that are valid according to the language’s syntax.
- A grammar does not describe the meaning of the strings or what can be done with them in whatever context – only their form.

A grammar example

A simple html grammar:

HTMLFILE ::= “<html>” [HEAD] BODY “</html>”
HEAD ::= “<head>” TITLE “</head>”
TITLE ::= “<title>” TEXT “</title>”
BODY ::= “<body>” [BODYTAG]* “</body>”
BODYTAG ::= H1TAG | PTAG | OLTAG | ULTAG
H1TAG ::= “<h1>” TEXT “</h1>”
PTAG ::= “ ” TEXT “ ”
OLTAG ::= “
” [LITAG]+ “”
ULTAG ::= “
” [LITAG]+ “”
LITAG ::= “
” TEXT “”
TEXT ::= *sequence of characters other than < and >*

Nonterminals

Non terminals
- elements of the grammar that are not part of th
- defined by rules.

Top level nonterminal usually first

HTMLFILE ::= "<html>" [HEAD] BODY "</html>"
HEAD ::= "<head>" TITLE "</head>"
TITLE ::= "<title>" TEXT "</title>"
BODY ::= "<body>" [BODYTAG]* "</body>"
BODYTAG ::= H1TAG | PTAG | OLTAG | ULTAG
H1TAG ::= "<h1>" TEXT "</h1>"
PTAG ::= " " TEXT " "
OLTAG ::= "
" [LITAG]+ "
"
ULTAG ::= "
" [LITAG]+ "
"
LITAG ::= "
" TEXT "
"
TEXT ::= *sequence of characters other than < and >*

| = "or"
[*NT*] = "optional"
[*NT*]* = "any number of times"
[*NT*]+ = "one or more times"

Terminals

Terminals
- literal strings or patterns of characters

HTMLFILE ::= "<html>" [HEAD] BODY "</html>"
HEAD ::= "<head>" TITLE "</head>"
TITLE ::= "<title>" TEXT "</title>"
BODY ::= "<body>" [BODYTAG]* "</body>"
BODYTAG ::= H1TAG | PTAG | OLTAG | ULTAG
H1TAG ::= "<h1>" TEXT "</h1>"
PTAG ::= " " TEXT " "
OLTAG ::= "
" [LITAG]+ "
"
ULTAG ::= "
" [LITAG]+ "
"
LITAG ::= "
" TEXT "
"
TEXT ::= *sequence of characters other than < and >*

Using the Grammar

Given some text:

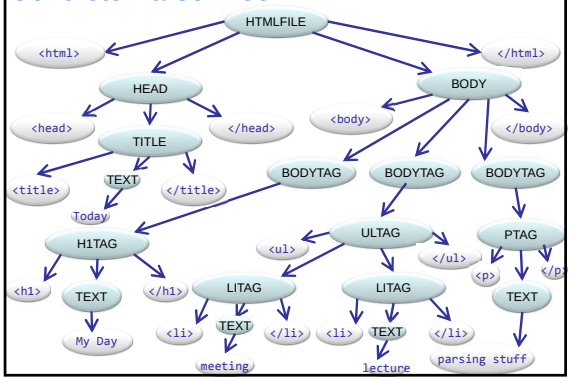
```
<html>
<head><title> Today</title></head>
<body><h1> My Day </h1>
<ul><li>meeting</li><li> lecture </li></ul>
<p> parsing stuff</p>
</body>
</html>
```

- Is it a valid piece of HTML?
  - Does it conform to the grammar rules?
- What is the structure? (Needed in order to process it)
  - what are the components?
  - what types are the components?
  - how are they related?

What kind of structure?

- Text conforming to a grammar has a tree structure
  - Ordered tree – order of the children matters
  - Each node in the tree and its children correspond to a grammar rule
  - Each internal node labeled by the nonterminal on LHS of rule
  - Leaves correspond to terminals.
- A **concrete parse tree** represents the syntactic structure of a string according to some formal grammar, showing all the components of the rules
- An **abstract syntax tree** leaves out elements of the rules that are not essential to the structure.

Concrete Parse Tree:



Is that too much information?

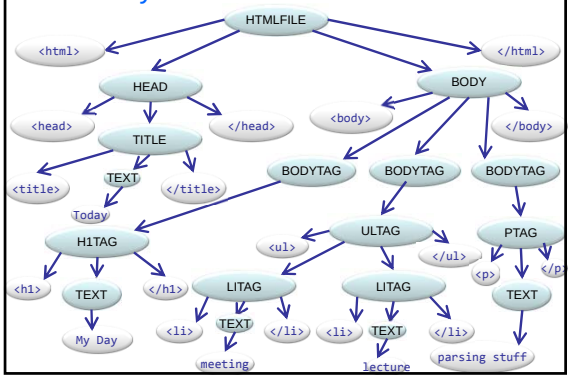
Yes!

- For example, we know that every HEAD will contain "<head>" and "</head>" terminals, we only care about what TITLE there is and only the unknown string part of that title.

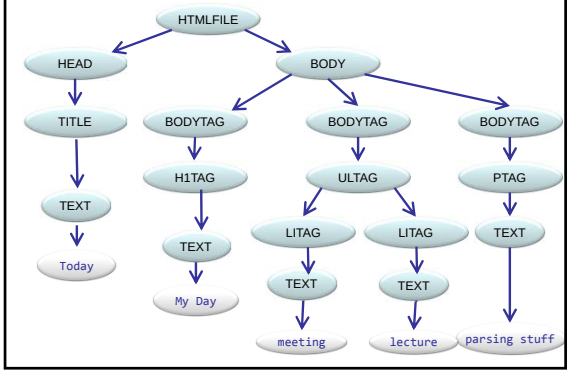
Definition:

1. An **abstract syntax tree (AST)** is a tree representation of the abstract syntactic structure of the text.
2. Each node of the tree denotes a construct occurring in the text.
3. The syntax is 'abstract' in that it does not represent all the elements of the full syntax.

Abstract Syntax Tree :



Abstract Syntax Tree (AST)



How do we write programs to do this?

- The process of getting from the *input string* to the parse tree consists of *two steps:*
  1. *Lexical analysis:* the process of converting a sequence of characters into a sequence of tokens.
     - Note that java.util.Scanner allows us to do lexical analysis with great ease!
  2. *Syntactic analysis or parsing:* the process of analysing text, made of a sequence of tokens to determine its grammatical structure with respect to a given grammar.
     - Assignment will require you to write a recursive descent parser discussed in the next lecture!