Andrew Plum

CS 210

1/24/2023

Week 3 Notes

* Coding grammar
  + The set of rules of how to build a tree – a parse tree
  + Context free grammar
  + Finding the parse tree
    - Is finding the string
* BNF grammar consists of
  + Tokens
    - Smallest unit - atomic
  + Non-terminal symbols
  + Productions
  + Start symbol
* To build a parse tree
  + Put a start symbol at top
  + Done when all leaves are tokens
* BNF vairiations
  + Some -> or = or ::=
  + Some leave out angle brackets
  + Some use single quotes to differentiate it from meta-symbols e.g. ‘|’ to distinguish |
* EBNF
  + Additional syntax to simplify some grammar chores
  + {x}to mean zero or more repetitions of x
  + [x] to mean x is optional (i.e. x | <empty>)
  + () for grouping
  + | anywhere to mean a choice among alternatives
  + Quotes around tokens, if necessary, to distinguish from all these meta-symbols
  + Anything EBNF can do BNF can do except its just more work
* Syntax Diagrams
  + “railroad diagrams”
  + A simple production is just a chain of boxes
  + Elses are bypasses
  + Branching
    - Give it multiple paths
  + Loops
    - Use loops for EBNF curly brackets
      * <exp> ::= <addend> {+ <addend>}
  + Practice:
    - Ex1: Make syntax diagram BNF: <SL>::={<s>;}EBNF: <SL> ::= {<s>;}
      * T
  + Syntax diagrams
    - Easier for people to read casually
    - Harder to read prescisely
    - Harder to make machine readable
* Formal context-free grammars
  + - In study of formal languages and automata, grammars are expressed in yet another notation; these are context free grammars
      * S -> aSb | X
      * X -> cX | e
        + In this example, any number of a’s followed by any number of c’s followed by any number of b’s
      * Other kinds of grammars are regular grammars and context sensitive grammers and unrestricted grammars
    - BNF and EBNF the notation is different
    - Use grammars to define lexical structure
  + Multiple audiences for grammar
    - Novices – want to find out what a legal program looks like
    - Experts – advanced users and language system implementers – want an exact, detailed definition
    - Tools – parser and scanner generators
* Yet another compiler compiler (YACC)
  + We need to create two files
    - myFile.i – specify all pattern matching rules for lex() and
    - myFile.y – grammars rules for yacc ().
* The first YACC assignment is just follow the videos and do exactly as they do
  + % signs in .l file are really important
  + Look at slides – it has some code and how to compile and do the assignment
  + dog.l is the stuff we are doing in this classs
  + Lex is tokenizing the file I believe