Recitation 2
Section 1 in CS314
Sep. 16, 2014
Review 2

• Defining a Language: Semantics and Syntax
  • Semantics: MEANINGS of the sentence
  • Syntax: GRAMMAR we use to create a sentence(formal languages)

• tokens & sentences
  • RE, to check if the token is valid
  • CFG, to check if the sentence is valid(Parse Tree)
Review 2

• Grammar:
  • Terminal Symbols (Tokens)
    • a, b, +, -...
  • Non-terminal Symbols
    • Term, Factor, Stmt, MultiAB, AB...
  • Production Rules (Sentences)
    • if (expression) statement else statement
    • stmt => if ( expr ) stmt else stmt (BNF)

*The left side of the production should always be a non-terminal*
Review 2

• Grammar: Regular Expression & FSA
• Regular Expressions: Formalisms for specifying a language

* Repeat for n times (n=0,1,2,...)
+ Repeat for n times (n=1,2,3,...)
() Combine
| OR
ε NULL
Review 2

• Automata: Formalisms for recognizing patterns in strings
  
  Input(string) → Automata → Output(yes or no)

• FSA: Finite State Automaton
  
  • Description:
    <
    • set of states,
    • labeled transitions (current state, label→transitions, next state(s)),
    • start state,
    • final state(s)
    >
  
  • FAs are NOT unique: two FAs could recognize a same string (eg. invisible by 4)
• \((0|1)^*11(0|1)^*\)
Review 2

- RE
  - NFA with epsilon transition
    - NFA
      - DFA

eg. RE = (1|0)*11

1 more example: NO DFA/NFA/RE for palindromes(eg.11100111)
Review 2

• Grammar: Parse Trees
  • Root: start, should be non-terminal
  • Leaves: terminals
  • Internals: non-terminals
  • For each internal N, N + N’s children=a Production
Review 2

• Grammar: Ambiguity
  • Ambiguity: more than one parse trees could represent a same string.
    John saw the man on the hill *with the telescope*. (dangling else?)

• To overcome the “Ambiguity”, we need to encode:
  • Associativity
  • Precedence
Review 2

• Grammar: Ambiguity

<expr> ::= <expr> + <expr> | <expr> - <expr> |
<expr> * <expr> | <expr> / <expr> |
<var> | <num> <var> ::= a|b|c|...|x|y|z
<num> ::= 0|1|2|3|4|5|6|7|8|9

parse x + 3 * y
• Grammar: Ambiguity

```
<expr> ::= <expr> + <expr> | <expr> - <expr> | 
        <expr> * <expr> | <expr> / <expr> | 
        <var> | <num>
<var> ::= a | b | c | ... | x | y | z
<num> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

```
parse x + 3 * y
```
Review 2

• Grammar: Ambiguity
  \[<expr> ::= <expr> + <term> | <expr> - <term> | <term>\]
  \[<term> ::= <term> * <factor> | <term> / <factor> | <factor>\]
  \[<factor> ::= <var> | <num> | ( <expr> )\]

  Precedence.
  Associativity.
Review 2

• Grammar: Dangling else Problem
  • Dangling else Problem: Which “if” does a “else” belong to.

```
if x == 0 then if y == 0 then z := 1 else w := 2
```
Review 2

Dangling else Problem--- if x == 0 then if y == 0 then z := 1 else w := 2

◆ <stmt> ::= <if-stmt> | <assgn> | ...
◆ <if-stmt> ::= if <expr> then <stmt> | if <expr> then <stmt> else <stmt>

✓ <stmt> ::= <stmt1> | <stmt2>
✓ <stmt1> ::= if <expr> then <stmt1> else <stmt> | <assgn> | ...
✓ <stmt2> ::= if <expr> then <stmt>