Thinking about grammars
• Consider an expression language involving integers 1, 2 and 3 and the + operator
• These rules make the + operator left associative
  \(<e> ::= <int> | <e> + <int>\)
  \(<int> ::= 1 | 2 | 3\)
• Note that using the "|" notation obscures the fact that there are really five rules
  \(<@> ::= <int>\)
  \(<@> ::= <@> + <int>\)
  \(<int> ::= 1\)
  \(<int> ::= 2\)
  \(<int> ::= 3\)

A graphical view
• Each rule is a little tree with a non-terminal as its root and children which are non-terminals or terminals
• Here's how we we might visualize the grammar using ovals for non-terminals and strings as terminals

Generating a string & parse tree
• Create a parse tree P consisting of the node
  \(<e>\)
• Repeat until P has no non-terminals leaf nodes
  – Select a leaf node L that is a non-terminal
  – Select a grammar tree T that has the same non-terminal as its root and make a copy of it
  – Replace the leaf L in P with the copy of T

1 + 2 + 3
Here's an example showing the parse tree for 1+2+3

the grammar rules

the parse tree

1 + 2 + 3
Here's an example showing the derivation of 1+2+3