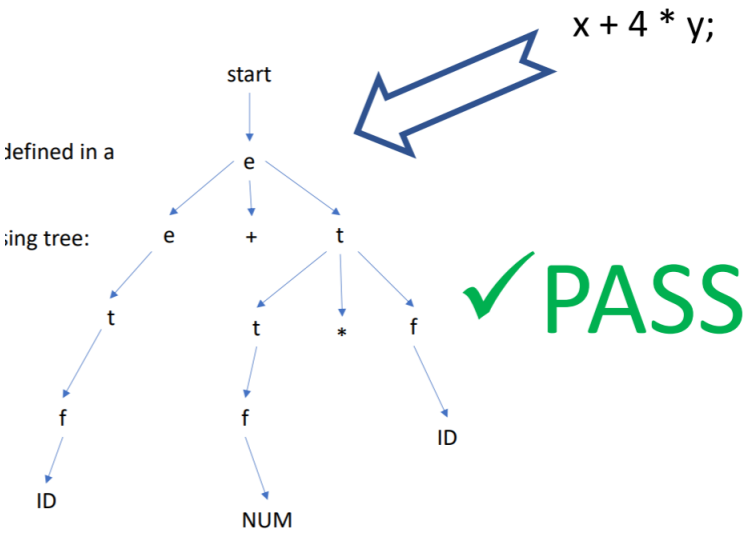
PPL 2/10

Reagan Shirk

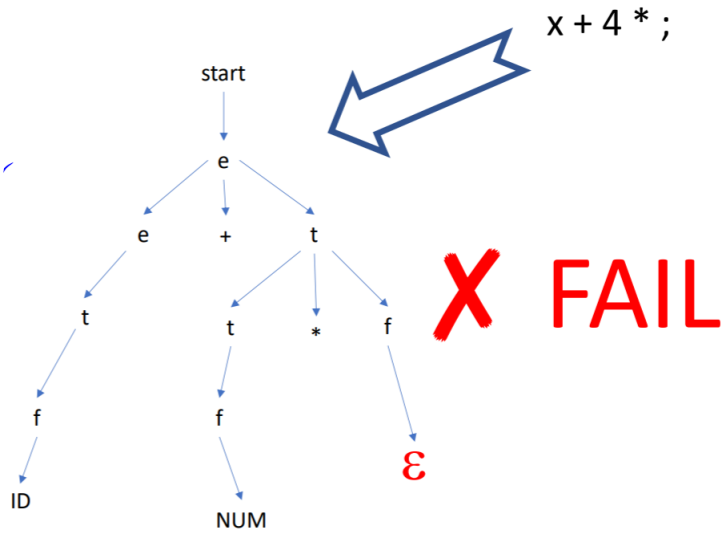
February 10, 2020

Syntactic Analysis

- This comes after Lexical Analysis (second stage in the classical compiling process)
- Syntactic rules of any programming language are defined in a context free grammar (throwback to theory)
  - Constructs a parse tree
  - In the below parse tree: x = e, 4 = t, y = f



- We want the parser to tell us if every token is in the right place, or to tell us if there is a missing/incorrectly placed token
  - An error is reported when this happens - you can abort or you can try to backtrack to a good state and report the errors



Context Free Grammars

- RegEx's can't specify nested constructs like CFG's can
- We needed the ability to represent recursion and to represent something in terms of itself
- Context free means that rules are applied independently of the context or surroundings (finally I've been wondering why it's called context free for a hot second)
- Each rule of a CFG is called a *production*
- left hand side = *variables*, right hand side = *terminals* or *tokens*
- some variable is chosen as the *start symbol*
- CFGs are composed of 4 parts:
  - A set of terminals T
  - A set of non-terminals N
  - A non-terminal S that is the start symbol
  - A set of productions (rules)

Derivation and Parse Trees

- Derivation is taking a left hand part and replacing it by the correct right hand part (or other way around)
  - Begin with start symbol
  - Choose a production with the start symbol on the left hand side
  - Replace the start symbol with the right hand side
  - Choose a non-terminal A in the resulting string
  - Choose a production P with A on the left hand side, replace A with the right hand side of P
  - Repeat
- I'm so zoned out my dudes