CS 173 Lecture 15a: Context-Free Grammars

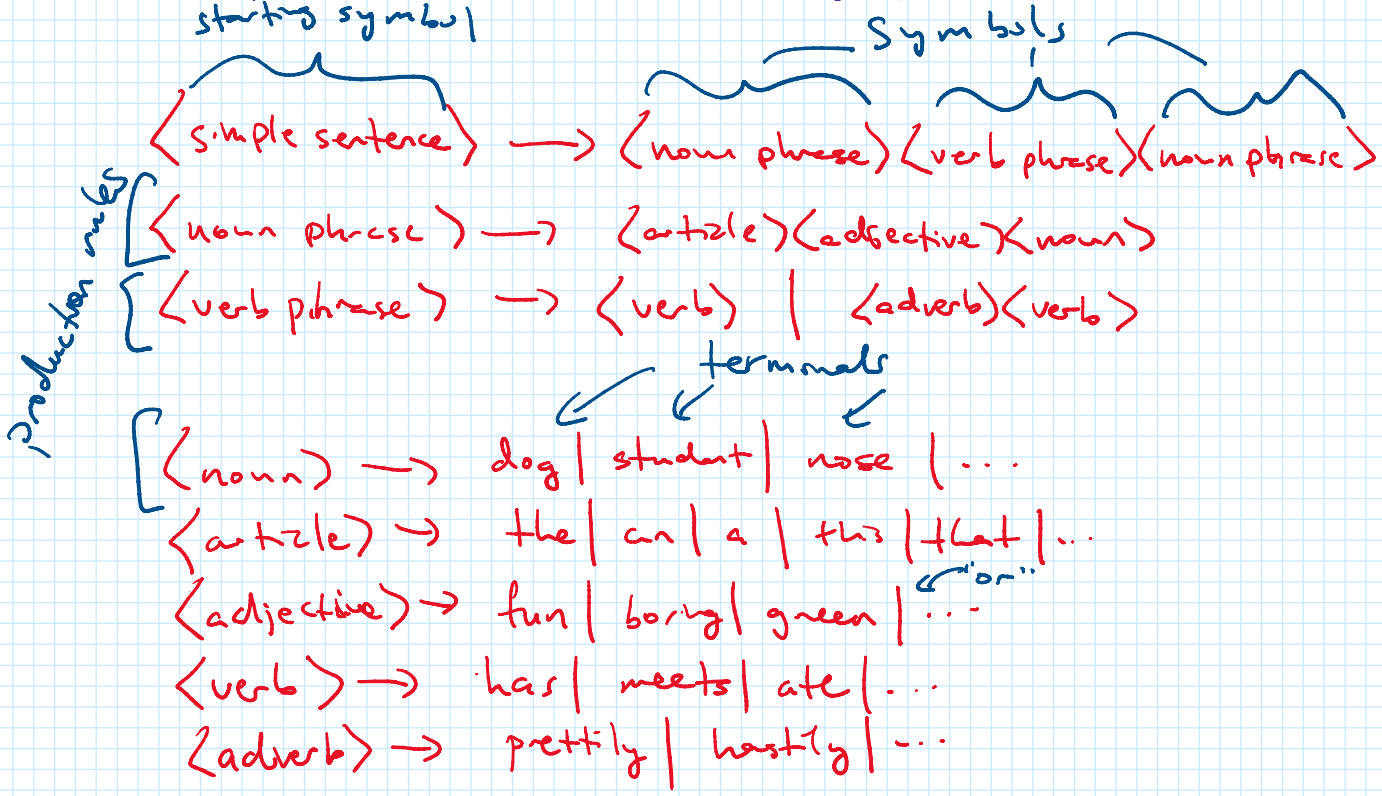
Informally: a recursive way of construct strings ... (see Chapter)

Historically:

- Linguistics
  - attempt to model human language via constructing sentences out of recursive phrases.
- Computer Science
  - a way to construct new programming languages
  - parsing → compiling

CFGs consists of

- symbols
- terminals
- production rules
- starting symbol



CFG: starting w/ a starting symbol:

CFG: starting w/ a starting symbol:

replace symbols w/ one of the options in the production rule

then recurse until only terminals are left

⟨simple sentence⟩

→ ⟨np⟩⟨vp⟩⟨np⟩ ← choice: ⟨v⟩ instead of ⟨adv⟩⟨v⟩

→ ⟨art⟩⟨adj⟩⟨n⟩⟨v⟩⟨art⟩⟨adj⟩⟨n⟩

→ the fun dog has a green nose.

↖ ↑ ↗ ...

choices

⟨ss⟩

→ ⟨np⟩⟨vp⟩⟨np⟩

→ ⟨art⟩⟨adj⟩⟨n⟩⟨adv⟩⟨v⟩⟨art⟩⟨adj⟩⟨n⟩

→ This funny student prettily ate an boring homework.

valid string in the grammar.

bad English "an boring"

makes no sense

X

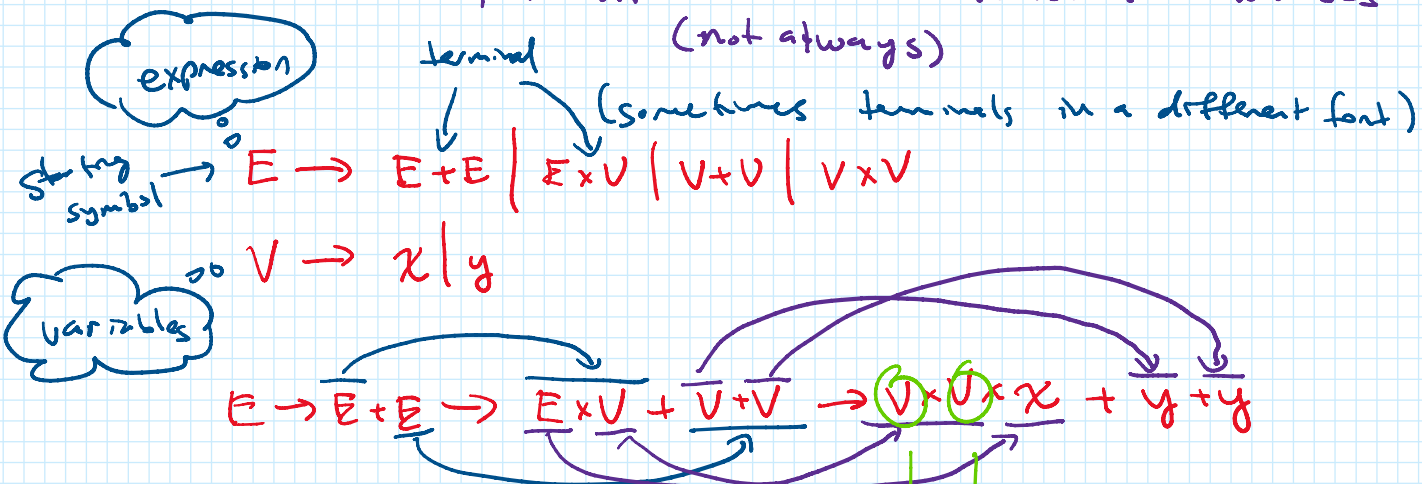
starting symbol's prod rule written first

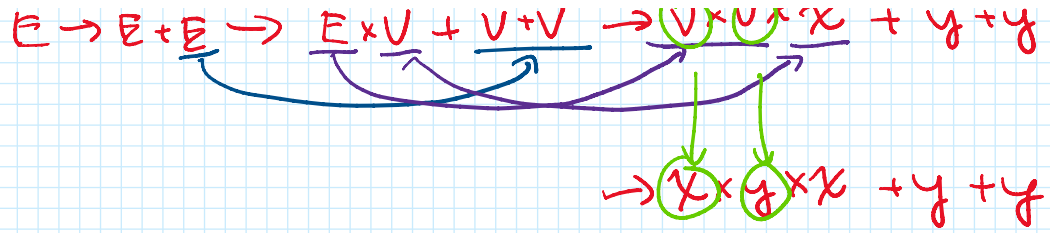
Some conventions: symbols are often upper case letters

terminals are lower case letters or numbers

(not always)

(sometimes terminals in a different font)





"CFG for 'simple' arithmetic expressions w/ variables x & y"