What is a grammar?

Language view:
A grammar is a set of structural rules that govern the composition of sentences, phrases and words.

Computational view:
A grammar (often called a “formal grammar”) is a set of rules that describe what strings are valid in a formal language.

Lots of different kinds of grammars:
- regular
- context-free
- context-sensitive
- recursively enumerable
- transformation grammars
Context Free Grammars (CFG)

- How many people have heard of them?
- What do you know about them?
- Where are they used?

CFG production rules

- \( S \rightarrow NP \ VP \)

  - left hand side
  - right hand side
  - (single symbol)
  - (one or more symbols)

CFG example

- \( S \rightarrow A \ B \ C \)
- \( A \rightarrow I \)
- \( B \rightarrow \text{really} \)
- \( B \rightarrow \text{really, B} \)
- \( C \rightarrow \text{like cs} \)

Grammars “generate” or “derive” strings:

- \( S \rightarrow \_ \)

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right.
Grammars “generate” or “derive” strings:

- **S** → **A** **B** **C**
- **A** → **I**
- **B** → *really*
- **B** → *really, B*
- **C** → *like cs*

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right.
Grammars “generate” or “derive” strings:

- $S \rightarrow A \ B \ C$
- $A \rightarrow I$
- $B \rightarrow$ really
- $B \rightarrow$ really, $B$
- $C \rightarrow$ like $cs$

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right.

No more rules apply, so we’re done!
Grammars "generate" or "derive" strings:

S → A B C
A → I
B → really
B → really, B
C → like cs

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right.

Is this the only string that can be derived?

Grammars describe a language, i.e. the strings (aka sentences) that are part of that language:

S → A B C
A → I
B → really
B → really, B
C → like cs

We can apply a rule by substituting the symbol on the left hand side with the symbols on the right.

I really, really, ... like cs
CFGs formally

\[ G = (\mathit{NT}, \mathit{T}, \mathit{P}, \mathit{S}) \]

\textbf{NT}: finite set of nonterminal symbols

\textbf{T}: finite set of terminal symbols, \textbf{NT} and \textbf{T} are disjoint

\textbf{P}: finite set of productions of the form

\[ A \rightarrow \alpha, \ A \in \text{NT} \text{ and } \alpha \in (\mathit{T} \cup \text{NT})^* \]

\textbf{S} \in \text{NT}: start symbol