**General Context-Free Grammar Parsing:**

A phrase structure grammar

- Also known as a context-free grammar (CFG)

**Example Grammar Rules:**

- $S \rightarrow NP \ VP \ DT$
- $NP \rightarrow DT \ NNS$
- $NP \rightarrow DT \ NN$
- $NP \rightarrow PP$
- $VP \rightarrow VBD \ NP$
- $VP \rightarrow VBD \ PP$
- $PP \rightarrow IN \ NP \ IN$
- $NN \rightarrow {\text{cake}}$
- $NN \rightarrow {\text{mountains}}$
- $NN \rightarrow {\text{children}}$
- $NN \rightarrow {\text{students}}$
- $VBD \rightarrow {\text{saw}}$
- $VBD \rightarrow {\text{ate}}$
- $VBD \rightarrow {\text{slept}}$

**Application of Grammar Rewrite Rules:**

- $S \rightarrow NP \ VP \ DT \ students \ ate \ the \ cake$
- $S \rightarrow NP \ VP \ DT \ children \ saw \ the \ mountains$

**Why we need phrase structure:**

- Syntax gives important clues in information extraction tasks and some cases of named entity recognition.
- We have recently demonstrated that stimulation of [CELLTYPE] human T and natural killer cells with [PROTEIN] induces tyrosine phosphorylation of [PROTEIN] JAK2 and [PROTEIN] Tyk2.
- Things that are the object of phosphorylation are likely tyrosine kinases [PROTEIN] and [PROTEIN] family.

**Phrase Structure is Recursive:**

- The captain of the ship yelled out.
- The velocity of the seismic waves rises to.

**Constituency:**

- How do we know what is a constituent? (Not that linguists don't argue about some cases.)
- Distribution: behaves as a unit that appears in different places.
- Substitution/expansion/pro-forms:
  - $S \rightarrow NP \ VP \ DT \ \text{the} \ \text{children} \ \text{ate} \ \text{the} \ \text{cake}$
  - $S \rightarrow NP \ VP \ DT \ \text{children} \ \text{ate} \ \text{the} \ \text{cake}$
  - $*$ $S \rightarrow \text{children} \ \text{ate} \ \text{the} \ \text{cake} \ \text{to} \ \text{the} \ \text{children}$
Natural language grammars are ambiguous:

- From the parse tree:
  - We can do both depth-first and breadth-first parsing.

- Attachment ambiguities in a real sentence:
  - The board approved [its acquisition] [by Royal Trustco Ltd. [of Toronto] [for $27 a share] [at its monthly meeting].

- What is parsing?
  - We want to examine all structures for a string of words.
  - Parsing is a hidden data problem.

- What is the size of embedded NPs?

- Ambiguity:
  - Natural languages have global ambiguities:
    - Programming language parsers resolve local ambiguities.

- Penn Treebank Sentences: an example
  - (S (NP-SBJ (DT The) (NN move)) (VP (VBD followed) (NP (NP (DT a) (NN round)) (PP (IN of) (NP (NP (JJ similar) (NNS increases)) (PP (IN by) (NP (JJ other) (NNS lenders))) (PP (IN against) (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))) , ,) (S-ADV (NP-SBJ (-NONE- *)) (VP (VBG reflecting) (NP (NP (DT a) (VBG continuing) (NN decline)) (PP-LOC (IN in) (NP (DT that) (NN market))))) , ,))

- What is parsing?
  - We want to run the grammar backwards to find the structure.
  - Parsing can be viewed as a search problem.
  - We search through the legal rewritings of the grammar.

- Examples of local ambiguities:
  - Prepositional phrase attaching to verb:
    - The children ate the cake with a spoon.
  - Prepositional phrase attaching to noun:
    - The cake with a spoon.

- Attachment ambiguities in a real sentence:
  - The board approved [its acquisition] [by Royal Trustco Ltd. [of Toronto] [for $27 a share] [at its monthly meeting].
Humans often do ambiguity maintenance. Have the police come in and look around? Have they eaten their supper? taken out and shot.

But humans also commit early and are "garden pathed": The man who hunts ducks out on weekends. The cotton shirts are made from grows in Mississippi. The horse raced past the barn fell.

### State space search

**States:**
- Start state: 
- Goal test: 
- Algorithm: 

```python
stack = { startState }
solutions = {}
loop
if stack is empty, return solutions
state = remove-front(stack)
if goal(state) push(state, solutions)
stack = pushAll(expand(state, operators), stack)
end
```

### Another phrase structure grammar

A grammar generates a language if a nonterminal is a sequence of terminals and its terminal productions are all rewrite rules which always rewrite as terminals.

For NLP, we usually distinguish out a set \( P \subset N \) of preterminals which always rewrite as terminals.

**Phrase Structure (CF) Grammars**

\[ G = (T,N,S,R) \]

- \( T \) is set of terminals
- \( N \) is set of nonterminals
  - For NLP, we usually distinguish out a set \( P \subset N \) of preterminals which always rewrite as terminals.
- \( S \) is start symbol (one of the nonterminals)
- \( R \) is set of rules/productions of the form \( X \rightarrow \gamma \), where \( X \) is a nonterminal and \( \gamma \) is a sequence of terminals

### Recognizers and parsers

A recognizer is a program for which a given grammar and a given sentence returns yes if the sentence is accepted by the grammar (i.e., the sentence is in the language) and no otherwise.

A parser in addition to doing the work of a recognizer also returns the set of parse trees for the string.
Soundness and completeness

- A parser is sound if every parse it returns is valid/correct.
- A parser terminates if it is guaranteed to not go off into an infinite loop.
- A parser is complete if for any given grammar and sentence it is sound, produces every valid parse for that sentence, and terminates.

(For many purposes, we settle for sound but incomplete parsers: e.g., probabilistic parsers that return a k-best list of possible parses.)

Top-down parsing

- Top-down parsing is goal directed.
- A top-down parser starts with a list of constituents to be built. The top-down parser rewrites the goals in the goal list by matching one against the LHS of the grammar rules, and expanding it with the RHS, attempting to match the sentence to be derived.
- If a goal can be rewritten in several ways, then there is a choice of which rule to apply (search problem).
- Can use depth-first or breadth-first search, and goal ordering.

Bottom-up parsing

- Bottom-up parsing is data directed.
- The initial goal list of a bottom-up parser is the string to be parsed. If a sequence in the goal list matches the RHS of a rule, then it may be replaced by the LHS of the rule.
- Parsing is finished when the goal list contains just the start category.
- If the RHS of several rules match the goal list, then there is a choice of which rule to apply (search problem).
- Can use depth-first or breadth-first search, and goal ordering.
- The standard presentation is shift-reduce parsing.

Principles for success: what one needs to do

- If you are going to do parsing-as-search with a grammar:
  - Left recursive structures must be found, not predicted.
  - Empty categories must be predicted, not found.

- Repeated work: everywhere there is common substructure.
- Repeated work: every rule that is always used by the parser must be rewritten as a left-recursion-free form.
- Repeated work: every rule that is generally useful must be rewritten as a left-recursion-free form.

- Problems with top-down parsing
  - Left recursive rules.
  - A top-down parser will do every possible top-down rewrite. The standard presentation is shift-reduce parsing.
  - A bottom-up parser fails to find the LHS of the grammar.

- Problems with bottom-up parsing
  - Bottom-up parsing is data directed.
  - If a sequence in the goal list matches the RHS of a rule, then it may be replaced by the LHS of the rule. Parsing is finished when the goal list contains just the start category.
  - A bottom-up parser fails to find the LHS of the grammar.

- Soundness and completeness
An alternative way to fix things

Grammar transformations can fix both left-recursion and epsilon productions.

Then you parse the same language but with different trees.

Linguists tend to hate you

But this is a misconception: they shouldn't.

You can fix the trees post hoc.

A second way to fix things

Rather than doing parsing-as-search, we do parsing as dynamic programming.

This is the most standard way to do things.

Doing graph-search or even tree-search.

But there are also other ways of solving the problem of doing repeated work.

Metamorphology (transforming solved subproblems) doing repeated work.

An alternative way to fix things

The main problem in parsing is working out how to 'attach' various kinds of constituents – PPs, adverbial or participial phrases, coordinations, and so on.

Propositional Phrase attachment.

Is the problem 'AI-complete'? Yes, but...

The noun 'man'

What does 'with a telescope' modify?

The verb 'saw'

The noun 'man'

Prepositional Phrase attachment.

The main problem in parsing is working out how to

In the V NP PP context, right attachment gets it right in 55–67% of cases.

But that means it gets it wrong in 33–45% of cases.

Which simple structural factors dominated in early PSY.

[NP → VP] V NP PP

Minimal attachment (Frazier 1978) depends on grammar:

[VP → V NP PP] V NP PP

Right association (Kimball 1973) = 'low' or 'near' attachment = 'late closure' (of NP) [NP → NP PP]

Minimal association = 'high' or 'distant' attachment = 'early closure' (of NP) [VP → V NP PP]

Such simple structural factors dominated in early psychological parsing.

Right association, left closure. (NP)

Minimal association, right closure.

Preferential Phrase attachment.

On participle phrases, coordinations, and so on.

These various kinds of constituents – PPs, adverbials, participles – are attached.

The main problem in parsing is working out how to

You can fix the trees post hoc

But this is a misconception: they shouldn't.

Linguists tend to hate you

Then you parse the same language but with different epsilon productions. Can fix both left-recursion and
A PCFG consists of the usual parts of a CFG

- A set of terminals, \( \{ w_k \} \), \( k = 1, \ldots, V \)
- A set of nonterminals, \( \{ N_i \} \), \( i = 1, \ldots, n \)
- A designated start symbol, \( N_1 \)
- A set of rules, \( \{ N_i \rightarrow \zeta_j \} \)
  (where \( \zeta_j \) is a sequence of terminals and nonterminals)
- A corresponding set of probabilities on rules such that:
  \[ \forall i \sum_j P(N_i \rightarrow \zeta_j) = 1 \]

The PCFG probability of a string \( P(w_1^n) = \sum_t P(w_1^n, t) \) is the probability of a parse of \( w_1^n \):

\[ P(t) \]

A simple PCFG (in CNF)

\[
\begin{align*}
S &\rightarrow \text{NP VP} & 1.0 & \text{NP} \\
&\rightarrow \text{NP PP} & 0.4 & \text{PP} \\
&\rightarrow P \text{ NP} & 1.0 & \text{NP} \\
&\rightarrow \text{astronomers} & 0.1 & \\
&\rightarrow \text{V NP} & 0.7 & \text{NP} \\
&\rightarrow \text{saw} & 0.18 & \\
&\rightarrow \text{VP PP} & 0.3 & \text{NP} \\
&\rightarrow \text{saw} & 0.04 & \\
&\rightarrow \text{P NP} & 1.0 & \text{NP} \\
&\rightarrow \text{with stars} & 0.18 & \\
&\rightarrow \text{V} & 1.0 & \text{NP} \\
&\rightarrow \text{saw} & 0.1 & \\
&\rightarrow \text{P NP} & 1.0 & \text{NP} \\
&\rightarrow \text{with stars} & 0.18 & \\
&\rightarrow \text{with ears} & 0.1 & \\
&\rightarrow \text{V} & 1.0 & \text{NP} \\
&\rightarrow \text{saw} & 0.04 & \\
&\rightarrow \text{P NP} & 1.0 & \text{NP} \\
&\rightarrow \text{with ears} & 0.04 & \\
\end{align*}
\]

The two parse trees' probabilities and the sen-
tence probability

\[
P(t_1) = 0.1 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 0.18 = 0.0009072
\]

\[
P(t_2) = 0.1 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 0.04 \times 0.04 \times 0.18 = 0.0006804
\]

\[
P(w_1^n) = P(t_1) + P(t_2) = 0.0015876
\]