Problems with the Top-Down Parser

1. Only judges grammaticality.
2. Stops when it finds a single derivation.
3. No semantic knowledge employed.
4. No way to rank the derivations.
5. Problems with left-recursive rules.
6. Problems with ungrammatical sentences.

Efficient Parsing

The top-down parser is terribly inefficient.

*Have the first year Phd students in the computer science department take the Q-exam.*

*Have the first year Phd students in the computer science department taken the Q-exam?*

Chart Parsers

*chart*: data structure that stores partial results of the parsing process in such a way that they can be reused. The chart for an \(n\)-word sentence consists of:

- \(n + 1\) vertices
- a number of edges that connect vertices

\[
\begin{array}{c}
0 & 1 & 2 & 3 & 4 & 5 \\
\text{Judge} & \text{to} & \text{scolded} & \text{the} & \text{defense} & \text{.} \\
S \rightarrow \text{NP} . \text{VP} & & & & & \\
\text{VP} \rightarrow \text{V} \text{NP} . & & & & & \\
S \rightarrow \text{NP} \text{ VP} . & & & & & \\
\end{array}
\]

Chart Parsing: The General Idea

The process of parsing an \(n\)-word sentence consists of forming a chart with \(n + 1\) vertices and adding edges to the chart one at a time.

- **Goal**: To produce a complete edge that spans from vertex 0 to \(n\) and is of category \(S\).
- There is no backtracking.
- Everything that is put in the chart stays there.
- Chart contains all information needed to create parse tree.
Bottom-UP Chart Parsing Algorithm
Do until there is no input left:
1. If the agenda is empty, get next word from the input, look up word categories, add to agenda (as constituent spanning two positions).
2. Select a constituent from the agenda: constituent C from $p_1$ to $p_2$.
3. Insert $C$ into the chart from position $p_1$ to $p_2$.
4. For each rule in the grammar of form $X \rightarrow C X_1 \ldots X_n$, add an active edge of form $X \rightarrow C \circ X_1 \ldots X_n$ from $p_1$ to $p_2$.

5. Extend existing edges that are looking for a $C$.
   (a) For any active edge of form $X \rightarrow X_1 \ldots \circ C X_n$ from $p_0$ to $p_1$,
       add a new active edge $X \rightarrow X_1 \ldots C \circ X_n$ from $p_0$ to $p_2$.
   (b) For any active edge of form $X \rightarrow X_1 \ldots X_n \circ C$ from $p_0$ to $p_1$,
       add a new (completed) constituent of type $X$ from $p_0$ to $p_2$ to the agenda.

Grammar and Lexicon

Grammar:
1. $S \rightarrow NP \ VP$
2. $NP \rightarrow ART \ N$
3. $NP \rightarrow ART \ ADJ \ N$
4. $VP \rightarrow V \ NP$

Lexicon:
the: ART
old: ADJ, N
man: N, V
boat: N

Sentence: 1 The 2 old 3 man 4 the 5 boat 6
The old man the boat.

NP1 (rule 2)  
ART1  
NP2 (rule 3)  
ADJ1  
VP1  
N2  
V1  
NP3 (rule 4)  
ART2  
N3  

NP -> ART . N  
NP -> ART . ADJ N  
NP -> ART ADJ . N  
S -> NP . VP

S (rule 1)

Efficient Parsing

n = sentence length

Time complexity for naive algorithm: exponential in n
Time complexity for bottom-up chart parser: \(\Theta(n^3)\)

Options for improving efficiency:

1. Don’t do twice what you can do once.
2. Don’t represent distinctions that you don’t need.
   Fall leaves fall and spring leaves spring.
3. Don’t do once what you can avoid altogether.
   The can holds the water. (“can” : AUX, V, N)

Bottom-up Chart Parser

Is it any less naive than the top-down parser?

1. Only judges grammaticality. [fixed]
2. Stops when it finds a single derivation. [fixed]
3. No semantic knowledge employed.
4. No way to rank the derivations.
5. Problems with ungrammatical sentences. [better]
6. Terribly inefficient.