EE669: Natural Language Processing
Lecture on Basic Parsing

Read Chapter 3 of Allen and Chapters 9 and 10 ofJurafsky and Martin
Topics

- Top-down Parsing with CFGs
- The Search Problem in Parsing
- Bottom-up Parsers
- Mixed Mode Parsers
- Top-down Chart Parsers

The Ambiguity of the Day:
I saw her duck.
Example: I saw her duck.

- Contains a combination of lexical and structural ambiguity:

  **duck**: Noun or Verb

  **her**: possessive pronoun or objective case third person pronoun. In the first case, *her* is acting like a Determiner, in the second, it is acting as a subject of an embedded predicate.

  **see**: takes either a direct object or an object plus infinitive Verb phrase.
Top-down Parsing with CFGs

Example Grammar:

1. S → NP VP
2. NP → Art Noun
3. NP → Name
4. PP → Prep NP
5. VP → Verb
6. VP → Verb NP
7. VP → Verb NP PP
8. VP → Verb PP
Top-down Parsing with CFGs: State-of-Parse Information

For Top-down CFG parsing, three pieces of information determine the state of the parse.

1. **Current Sentence Position:** where the parser is in the sentence.

2. **Current Actions:** a list of actions being worked on (corresponds to the right-hand-side (RHS) of the rules).

   **Example:**

   (S) ; initial action list
   (NP VP) ; action list after using rule 1

3. **Backup Points:** for backtracking if there is more than one path to take from a given node, take one and save the others.
Top-down Parsing with CFGs: The CFG algorithm

For the current state:

Case 1: If the top of Current Actions is a word category and the next word in
the sentence is of that category, then:

1. Update Current Sentence Position to the next word.
2. Update Current Actions by popping off the category action.

Case 2: If the top of Current Actions is a word category and the next word
in the sentence is not of that category, then:

Replace the current state with a Backup Point if there is one, otherwise
fail.
Top-down Parsing with CFGs: The CFG algorithm *continued*

Case 3: If the top of Current Actions names the Left-Hand-Side (LHS) of a rule:

1. Replace the LHS non-terminal at the top of Current Actions with the RHS of the rule
2. For any remaining applicable rules, create Backup States consisting of the previous state (with the LHS non-terminal at the top of Current Actions replaced with the RHS of that rule) and Current Position.

Case 4: If the Current Actions stack is empty, and there are no more words in the sentence left to process then:

The parse is complete→ Success!

Case 5: If the Current Actions stack is empty, and there are more words in the sentence to process then:

Replace the current state with a Backup Point if there is one, otherwise fail.
Top-down Parsing Example: 1 Fred 2 ran 3 in 4 the 5 snow 6

1. Current Position = 1, Current Actions = (S), Backup Points = ()
   
   • Apply rules in order, matching Top of Current Actions to the head of the rule.
   
   • Rule 1’s head matches S, so the S is replaced by the RHS of S → NP VP in Current Actions.

2. Current Position = 1, Current Actions = (NP VP), Backup Points = ()
   
   • Work on the leftmost non-terminal in Current Actions, NP.
   
   • There are two NP rules:
     1. NP → Art Noun: Current Actions → (Art Noun VP)
     2. NP → Name; Use for Backup state, giving Backup point → ((Name VP), 1)
Top-down Parsing Example: 1 Fred 2 ran 3 in 4 the 5 snow 6 continued

3. Current Position = 1, Current Actions = (Art Noun VP), Backup Points = (((Name VP), 1))
   • Since Fred is not an Art, set Current Actions to the popped Backup Point.

4. Current Position = 1, Current Actions = (Name VP), Backup Points = ()
   • When there is a match between a category non-terminal and the word in the Current Position of the sentence, update Current Position to the next word and remove the category from the Current Actions List.
      1. Current Actions → (VP)
      2. Current Position → 2
Top-down Parsing Example: 1 Fred 2 ran 3 in 4 the 5 snow 6 continued

5. Current Position = 2, Current Actions = (VP), Backup Points = ()

   • There are 4 rules for VP:
     1. VP → Verb: Current Actions → (Verb)
     2. VP → Verb NP: insert (Verb NP) in Backup Points.
     3. VP → Verb NP PP: insert (Verb NP PP) in Backup Points.
     4. VP → Verb PP: insert (Verb PP) in Backup Points.


   • ran is a verb so Current Position becomes 3 and Current State becomes ().
Top-down Parsing Example: 1 Fred 2 ran 3 in 4 the 5 snow 6 continued

   • Since there are more words in the sentence and no actions left, this parse has failed, but we can still use Backup Points to try another alternative parse.

   • ran is a verb so Current Position becomes 3 and Current Actions becomes (NP).

   • There are two NP rules:
     1. NP → Art Noun: Current Actions → (Art Noun)
     2. NP → Name: Use for Backup state, giving Backup point → ((Name), 3)
Top-down Parsing Example: 1 Fred 2 ran 3 in 4 the 5 snow 6 continued

    • *in* is not an art, so use top Backup State.

    • *in* is not a Name, so use top Backup State.

    • *ran* is a verb so Current Position becomes 3 and Current Actions becomes (NP PP).
Top-down Parsing Example: 1 Fred 2 ran 3 in 4 the 5 snow 6 continued

   • There are two NP rules:
     1. NP → Art Noun: Current Actions → (Art Noun PP)
     2. NP → Name: Use for Backup state, giving Backup point → (Name PP)

   • in is not an Art, so use top Backup State.

15. Current Position = 3, Current Actions = (Name PP), Backup Points = (((Verb PP), 2))
   • in is not a Name, so use top Backup State.
**Top-down Parsing Example:** 1 Fred 2 ran 3 in 4 the 5 snow 6 continued

16. Current Position = 2, Current Actions = (Verb PP), Backup Points = ()
   - *ran* is a Verb so Current Position becomes 3 and Current Actions becomes (PP).

17. Current Position = 3, Current Actions = (PP), Backup Points = ()
   - There is one PP rule:
     PP $\rightarrow$ Prep NP
     Current Actions $\rightarrow$ (Prep NP)

18. Current Position = 3, Current Actions = (Prep NP), Backup Points = ()
   - *in* is a Prep so Current Position becomes 4 and Current Actions becomes (NP).
Top-down Parsing Example: 1 Fred 2 ran 3 in 4 the 5 snow 6 continued

19. Current Position = 4, Current Actions = (NP), Backup Points = ()

   • There are two NP rules:
     1. NP → Art Noun: Current Actions → (Art Noun)
     2. NP → Name Use for Backup state, giving Backup point → (Name)

20. Current Position = 4, Current Actions = (Art Noun), Backup Points = (((Name), 4))

   • the is an Art so Current Position becomes 5 and Current Actions becomes
     (Noun).

21. Current Position = 5, Current Actions = (Noun), Backup Points = (((Name), 4))

   • snow is a Noun so Current Position becomes 6 and Current Actions
     becomes ( ).
Top-down Parsing Example: 1 Fred 2 ran 3 in 4 the 5 snow 6 continued

22. Current Position = 6, Current Actions = (), Backup Points = (((Name), 4))

• Since there are no more Current Actions and the Current Position is at end-of-sentence, the parse has succeeded.
Search Problem in Parsing

- Depth-first versus Breadth-first search

- If we combine the current state and backup states into a possibility list, then:
  - In breadth-first search, the list is maintained as a FIFO queue.
  - In depth-first search, the list is maintained as an LIFO queue or stack.

- Left recursion in rules is a problem for the depth-first search strategy. For example:

  ADJP → ADJP Adj
  ADJP → Adj

  Why?
Search Problem in Parsing: Example Search Tree for *The dogs cried.*

1. $S \rightarrow NP\ VP$
2. $NP \rightarrow Art\ Noun$
3. $NP \rightarrow Name$
4. $PP \rightarrow Prep\ NP$
5. $VP \rightarrow Verb$
6. $VP \rightarrow Verb\ NP$
7. $VP \rightarrow Verb\ NP\ PP$
8. $VP \rightarrow Verb\ PP$

Note: Numbers indicate depth-first order, letters indicate breadth-first order.
Search Problem in Parsing: Example Search Tree for *The dogs cried.*
Bottom-up Parsers

- Match RHS of rules and reduce to the LHS.
  
  $the \ boy \rightarrow Det \ Noun$
  
  $Det \ Noun \rightarrow NP$

  because $the$ is a Det, $boy$ is a Noun, and $NP \rightarrow Det \ Noun$.

- In a bottom-up parser of English, the parse succeeds if we can reduce a sentence to $S$ (the start symbol) while using all of the words.

- For our example, we will use a simple grammar and lexicon depicted on the next slide.
Bottom-up Chart Parser

Grammar:

1. S → NP VP
2. NP → Det Adj Noun
3. NP → Det Noun
4. NP → Adj Noun
5. VP → Aux Verb NP
6. VP → Verb NP
7. VP → Verb

Lexicon:
the : Det
large : Adj
can : Noun, Aux, Verb
fell : Verb
**Bottom-up Chart Parser: Important Data Structures**

1. **Chart**: A record of word positions and the ways they combine into larger constituents, given the grammar rules.

2. **Agenda**: A stack of completed constituents that must be added to the Chart.

3. **Word list**: The list of words in the sentence being parsed. Whenever the Agenda is empty, we can work on the next word.
Bottom-up Chart Parser: Updating the Chart

- We build a Chart by entering the constituent with a unique name and arcs indicating how the parse is being carried out.

- Entering a constituent \( c \) (popped from the Agenda and given a unique name, say \( c_{22} \)) between \( p_1 \) and \( p_2 \) on the Chart requires the four steps:

1. Add \( c \) between \( p_1 \) and \( p_2 \):

\[
\begin{array}{c}
\text{c22} \\
\hline
p_1 & p_2
\end{array}
\]

2. For each rule where \( c \) begins the RHS of a rule \( r \) in the grammar, add an active arc for rule \( r \) between \( p_1 \) and \( p_2 \).

\[
\begin{array}{c}
\text{c22} \\
\hline
p_1 & r & p_2
\end{array}
\]
Bottom-up Chart Parser: Updating the Chart

3. For any active arc \( a \) beginning at \( p_0 \) and ending at \( p_1 \), if \( c \) is a subconstituent of \( a \), then add an active arc between \( p_0 \) and \( p_2 \) with the update of rule \( a \) (move the dot).

4. If any of the active arcs added in the previous two steps are completed rules (i.e., account for the RHS of a rule), then add new constituents named by the LHS of the rule to the Agenda.
Bottom-up Chart Parser: Arc Extension Algorithm

To add a constituent $C$ from $p_1$ to $p_2$:

1. Insert $C$ in Chart from $p_1$ to $p_2$.

2. For any active arc $X \rightarrow X_1 \ldots \bullet C \ldots X_n$ from $p_0$ to $p_1$, add a new arc $X \rightarrow X_1 \ldots C \bullet \ldots X_n$ from $p_0$ to $p_2$.

3. For any active arc $X \rightarrow X_1 \ldots X_n \bullet C$ from $p_0$ to $p_1$, add a new constituent of type $X$ from $p_0$ to $p_2$ to the Agenda.
**Bottom-up Chart Parser Algorithm**

Do until there is no input left:

1. If the Agenda is empty, look up all categories for the next word and add them to the Agenda.

2. Select a constituent from the Agenda; call it \( C \) from \( p_1 \) to \( p_2 \).

3. For each grammar rule of the form \( X \rightarrow CX_1 \ldots X_n \), add an active arc of the form \( X \rightarrow C \bullet X_1 \ldots X_n \) from to the Chart from \( p_1 \) to \( p_2 \).

4. Add \( C \) to the Chart using the arc extension algorithm.
Example: 1 The 2 large 3 can 4 fell 5

0. Sentence = (1 The 2 large 3 can 4 fell 5), Agenda = ()

- Agenda is empty, so get next word, the.

- Because the is a Det, the rules that will be applied begin with Det, i.e., rules 2 (NP \rightarrow Det Adj Noun) and 3 (NP \rightarrow Det Noun).

- 1 Det 2 is the key. Put it on Agenda.

- Use a \bullet to indicate how far into the rule we have parsed. For example: NP \rightarrow Det \bullet Adj Noun
Example: 1 The 2 large 3 can 4 fell 5 continued

1. Sentence = (2 large 3 can 4 fell 5), Agenda = (1 Det1 2)
   
   - Add Det1 to the Chart from 1 to 2.

   - Add arcs: NP → Det • Adj Noun and NP → Det • Noun.

   ![Diagram]

   a. NP -> Det . Adj Noun
   b. NP -> Det . Noun

   - The Agenda is now empty, so get next word, large. It is an Adj, so add
     2 Adj1 3 to Agenda.
Example: 1 The 2 large 3 can 4 fell 5 continued

2. Sentence = (3 can 4 fell 5), Agenda = (2 Adj1 3)

- Add Adj1 to the Chart from 2 to 3.
- Rule 4 starts with Adj, NP → Adj Noun, so add arc c. NP → Adj • Noun.
- Adj continues arc a. NP → Det • Adj Noun, so add arc d. NP → Det Adj • Noun.
- The Agenda is now empty, so get next word, can. It is a Noun, an Aux, and a Verb so add 3 Noun1 4, 3 Aux1 4, and 3 Verb1 4 to Agenda.

```
1. Det1 (the)
2. Adj1 (large)

a. NP → Det . Adj Noun
b. NP → Det . Noun
c. NP → Adj . Noun
d. NP → Det Adj . Noun
```
Example: 1 The 2 large 3 can 4 fell 5 continued

3. Sentence = (4 fell 5), Agenda = (3 Noun1 4, 3 Aux1 4, 3 Verb1 4)

- Add Noun1 to Chart from 3 to 4.

![Chart Diagram]

- No active arcs to add (i.e., no RHS of rules are headed by Noun or are partially continued by Noun).
- Can finish up arc c. NP → Adj • Noun, so add 2 NP2 4 to Agenda.
- Can finish up arc d. NP → Det Adj • Noun, so add 1 NP1 4 to Agenda.
Example: 1 The 2 large 3 can 4 fell 5 continued

4 and 5. Sentence = (4 fell 5), Agenda = (1 NP1 4, 2 NP2 4, 3 Aux1 4, 3 Verb1 4)

- Put NP1 in the Chart with a new arc e from 1 to 4 with S → NP • VP.
- Put NP2 in Chart with a new arc f from 2 to 4 with S → NP • VP.

\[ \begin{align*}
\text{a. } & \text{NP} \rightarrow \text{Det} \cdot \text{Adj Noun} \\
\text{b. } & \text{NP} \rightarrow \text{Det} \cdot \text{Noun} \\
\text{c. } & \text{NP} \rightarrow \text{Adj} \cdot \text{Noun} \\
\text{d. } & \text{NP} \rightarrow \text{Det Adj} \cdot \text{Noun} \\
\text{e. } & \text{S} \rightarrow \text{NP} \cdot \text{VP (NP1)} \\
\text{f. } & \text{S} \rightarrow \text{NP} \cdot \text{VP (NP2)}
\end{align*} \]
Example: 1 The 2 large 3 can 4 fell 5 continued

6 and 7. Sentence = (4 fell 5), Agenda = (3 Aux1 4, 3 Verb1 4)

- Add Aux1 to the Chart with a new arc g from 3 to 4 with VP → Aux
- Verb NP.
- Add Verb1 to Chart with a new arc h from 3 to 4 with VP → Verb • NP.
- Can finish up rule 7. VP → Verb, so add 3 VP1 4 to Agenda.
Example: The large can fell continued

8. Sentence = (fell), Agenda = (VP1)
   - Add VP1 to Chart, but cannot complete any S → NP • VP rules.

- The Agenda is now empty, so the algorithm gets the next word, fell. It is a Verb so add Verb2 to Agenda.
Example: 1 The 2 large 3 can 4 fell 5 continued

9. Sentence = (), Agenda = (4 Verb2 5)

- Create arc i by applying rule 6 with VP → Verb ⋅ NP.
- Finish rule 7. VP → Verb, so add 4 VP2 5 to Agenda.
- Continue arc g to produce arc j with VP → Aux Verb ⋅ NP.

```
NP1 (Rule 2)
  |   |   |
  |   |   |   |
NP2 (Rule 4)  Verb1 (can)  Aux1 (can)
               |   |   |
               |   |   |   |
VP1 (Rule 7)  |   |   |   |
               |   |   |   |   |
|   |   |   |   |
```

**Arcs:**

- a. NP → Det . Adj Noun
- b. NP → Det . Noun
- c. NP → Adj . Noun
- d. NP → Det Adj . Noun
- e. S → NP . VP (NP1)
- f. S → NP . VP (NP2)
- g. VP → Aux . Verb NP
- h. VP → Verb . NP
- i. VP → Verb . NP
- j. VP → Aux Verb . NP
Example: The large can fell continued

10. Sentence = (), Agenda = (4 VP2 5)
   - Add VP2 to Chart.
   - Finish up arc e for position 1 to 5 and arc f for position 2 to 5, and then add 1 S 5 and 2 S 5, respectively, to the Agenda.

11. Sentence = (), Agenda = (1 S1 5, 2 S2 5)
   - Add 1 S1 5 and 2 S2 5 to the Chart.
   - S1 allows the parser to succeed because it accounts for all of the words!
   - The Agenda is empty so no more possible parses.
Example: The large can fell continued

\[
\begin{align*}
S & \rightarrow NP \cdot VP (NP1) \\
S & \rightarrow NP \cdot VP (NP2) \\
VP & \rightarrow Aux \cdot Verb NP \\
VP & \rightarrow Verb . NP \\
VP & \rightarrow Aux Verb . NP \\
S1 & \rightarrow (rule 1) \\
S2 & \rightarrow (rule 2) \\
VP1 & \rightarrow (Rule 7) \\
Verb1 & \rightarrow (can) \\
Aux1 & \rightarrow (can) \\
NP2 & \rightarrow (Rule 4) \\
NP1 & \rightarrow (Rule 2) \\
\end{align*}
\]
Efficiency of Chart Parsing

- The Chart improves the efficiency of the parsing algorithm from $O(C^n)$ for a pure search parser (e.g., RTN) to $O(G \times n^3)$.

- The Chart parser is more efficient because it only needs to parse (or construct) a certain NP (over a particular string of words) once, even though the same constituent could be a member of more than one parse alternative (successful or not).

- Given a sentence with 12 words with $C = 10$ and $G = 1000$, the Chart parser would require on the order of $1000 \times 12^3 = 1,728,000$ compared to an ATN without a Chart with $10^{12}$ operations.
Problems with Bottom-up Parsers

- All word senses for a word must be considered, even if they are impossible given the surrounding words.

**Example:** He can make you sick.

A bottom-up parser must consider *can* as a Modal, Noun, or Verb, even if no rule predicts the lexical class.

- Many impossible structures are constructed in searching for a bottom-up parse. For example, consider the Chart in figure 3.19 from the book for the sentence *The large can can hold water.*
Problems with Bottom-up Parsers \textit{continued}

\begin{itemize}
\item \textbf{S1} (rule 1)
\item \textbf{S2} (rule 1) $\leftarrow$impossible
\item \textbf{NP2} (rule 4)
\item \textbf{VP1} (rule 5)
\item \textbf{NP1} (rule 2)
\item \textbf{VP2} (rule 6)
\item \textbf{Noun1} *\textbf{Noun2}
\item \textbf{*Verb1} *\textbf{Verb2} \textbf{Verb3} *\textbf{Verb4}
\item \textbf{Art1} \textbf{Adj1} *\textbf{Aux1} \textbf{Aux2} *\textbf{Noun3} \textbf{Art2} \textbf{Noun4}
\end{itemize}

\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
1 & 2 & large & 3 & can & 4 & can & 5 & hold & 6 & the & 7 & water & 8 \\
\hline
\end{tabular}
Problems with Top-down Parsers

• An advantage of top-down parsers is that they never consider word categories in positions where they could not occur syntactically.

• On the other hand, a top-down parser can operate for a long time rewriting rules before words are even considered.

• They can also do considerable amount of work before getting to the correct sentence rule.

For example, if we parse *Did you hit Bill?* given the following rule ordering, a simple top-down parser will do lots of work (especially given depth-first search) before getting the correct sentence structure.

\[
S \rightarrow \text{NP VP} \\
S \rightarrow \text{Aux NP VP}
\]
Problems with Top-down Parsers \textit{continued}

- In fact, the same work can be done over and over again.

  For example, given the following rules, if we parse: \textit{Fred saw the dog, saw} will be processed four times and \textit{the dog, twice}.

\begin{align*}
  \text{VP} & \rightarrow \text{Verb} \\
  \text{VP} & \rightarrow \text{Verb NP PP} \\
  \text{VP} & \rightarrow \text{Verb PP} \\
  \text{VP} & \rightarrow \text{Verb NP}
\end{align*}
Mixed Mode Parsers

- To address these problems, we could combine top-down and bottom-up methods, e.g., build a top-down CFG parser with a Chart.

- A top-down CFG parser with a Chart is, like the bottom-up Chart parser, driven by the Agenda. The difference between the two methods is in how new arcs are generated from the grammar.
Top-down Chart Parser

Initialization: For every rule in the grammar $S \rightarrow X_1 \ldots X_k$, add an arc labeled $S \rightarrow \bullet X_1 \ldots X_k$, and perform the arc introduction step.

Parsing: Do until there is no more input:

1. If the Agenda is empty, look for interpretations of the next word and add them to the Agenda.

2. Select a constituent $C$ from the Agenda.

3. Using the arc extension algorithm, combine $C$ with every active arc on the Chart. Add new constituents to the Agenda.

4. For any active arcs from step 3, add them to the Chart using the arc introduction algorithm.
Top-down Chart Parser \textit{continued}

\textbf{Top-Down Arc Introduction:} To add an arc $S \rightarrow C_1 \ldots \bullet C_i \ldots C_n$ ending at position $j$, do the following:

For each rule in the grammar $C_i \rightarrow X_1 \ldots X_k$, recursively add the new arc $C_i \rightarrow \bullet X_1 \ldots X_k$, from position $j$ to $j$. 
Bottom-up Chart Parser

Grammar:

1. $S \rightarrow NP \ VP$
2. $NP \rightarrow \text{Det} \ Noun$
3. $NP \rightarrow \text{Det} \ Adj \ Noun$
4. $NP \rightarrow \text{Adj} \ Noun$
5. $VP \rightarrow \text{Aux} \ Verb \ NP$
6. $VP \rightarrow \text{Verb} \ NP$

Lexicon:

green: Adj, Noun
water: Noun, Verb
hit: Noun, Verb
the: Det
window: Noun
Example:  1 Green  2 water  3 hit  4 the  5 window  6

1. S is the starting symbol, so initially, all productions must begin with S, giving an arc from position 1 to 1 of $S \rightarrow \bullet NP\ VP$. The arc introduction algorithm then adds an arc with the following information going from position 1 to 1.

NP $\rightarrow$ $\bullet$ Det Noun

NP $\rightarrow$ $\bullet$ Det Adj Noun

NP $\rightarrow$ $\bullet$ Adj Noun

Thus the initial Chart is below:

1

S $\rightarrow$ . NP VP

NP $\rightarrow$ . Det Noun

NP $\rightarrow$ . Det Adj Noun

NP $\rightarrow$ . Adj Noun
Example: 1 Green 2 water 3 hit 4 the 5 window 6 continued

2. Since the Agenda is empty, we look at the next word green and add an Adj from position 1 to 2 and a Noun from position 1 to 2 to the Agenda. If we consider the Noun first, there is no active arc to combine it with.

3. Next, we consider the word green as an Adj from position 1 to 2. It can be combined with the active arc, NP $\rightarrow$ $\bullet$ Adj Noun. Hence, we can enter the constituent ADJ1 from 1 to 2 and extend that active arc from position 1 to 2 as follows: NP $\rightarrow$ Adj $\bullet$ Noun.
Example: 1 Green 2 water 3 hit 4 the 5 window 6 continued

4. Since the Agenda is empty, we look at the next word water, which can be a Noun or a Verb, so we add a Noun from position 2 to 3 and a Verb from position 2 to 3 to the Agenda. If we consider the Verb first, there is no active arc to combine it with.
**Example:** 1 Green 2 water 3 hit 4 the 5 window 6 *continued*

5. Next, we consider the word *water* as a Noun from position 2 to 3. It can be combined with the active arc, NP → Adj • Noun.
Hence, we can enter the constituent NOUN1 from 2 to 3 and extend that active arc as follows: NP → Adj Noun •.
This completes an NP, which will be added to the Agenda from position 1 to 3.

6. Next, we pop the NP from position 1 to 3 from the Agenda. This NP can be used to extend the S rule, S → • NP VP.
Hence, we can enter the constituent NP1 in the Chart from position 1 to 3 and extend the active arc from position 1 to 3 as: S → NP • VP.
The arc introduction algorithm adds from position 3 to 3:

   VP → • Aux Verb NP
   VP → • Verb NP
Example: 1 Green 2 water 3 hit 4 the 5 window 6 *continued*

\[
\begin{array}{c|c|c|c|c|c}
1 & 2 & 3 & 4 & 5 & 6 \\
\hline
\text{green} & \text{water} & \text{hit} & \text{the} & \text{window} & \\
\hline
\end{array}
\]

NP → Adj . Noun

S → NP . VP

S → . NP VP

VP → . Aux Verb NP

NP → . Det Noun

VP → . Verb NP

NP → . Det Adj Noun

NP → . Adj Noun
**Example: 1 Green 2 water 3 hit 4 the 5 window 6 continued**

---

7. Since the Agenda is empty, we look at the next word *hit*, which can be a Noun or a Verb, so we add a Noun from position 3 to 4 and a Verb from position 3 to 4 to the Agenda. If we consider the Noun first, there is no active arc to combine it with.

---

8. Next, we consider the word *hit* as a Verb from position 3 to 4. It can be combined with `VP → • Verb NP`, hence we can enter the constituent `VERB1` from 3 to 4 and extend the active arc from position 3 to 4 as follows: `VP → Verb • NP`.

The arc introduction algorithm adds from position 4 to 4:

- `NP → • Det Noun`
- `NP → • Det Adj Noun`
- `NP → • Adj Noun`
Example: 1 Green 2 water 3 hit 4 the 5 window 6 continued
Example: 1 Green 2 water 3 hit 4 the 5 window 6 continued

9. Since the Agenda is empty, we look at the next word the, which can only be a Det from position 4 to 5, so it is added to the Agenda. When it is popped from the Agenda, it can continue two of the active arcs: NP → • Det Noun and NP → • Det Adj Noun.

Hence, we can enter the constituent DET1 from 4 to 5 and extend the active arcs from 4 to 5:

\[
\begin{align*}
\text{NP } & \rightarrow \text{ Det • Noun} \\
\text{NP } & \rightarrow \text{ Det • Adj Noun}
\end{align*}
\]

10. The Agenda is empty, so we look at the next word window, which can be a Noun from position 5 to 6. When it is popped from the Agenda, it can continue the active arc: NP → Det • Noun.

Hence, we can enter the constituent NOUN2 from 5 to 6 and extend the active arc: NP → Det Noun •. This completes the NP, which is added to the Agenda from position 4 to 6.
Example: 1 Green 2 water 3 hit 4 the 5 window 6 continued

11. Next, we remove the NP from position 4 to 6 from the Agenda. This NP can be used to extend the VP rule: $\text{VP} \rightarrow \text{Verb} \bullet \text{NP}$.

Hence, we can add NP2 to the Chart from position 4 to 6 and extend the VP rule as follows: $\text{VP} \rightarrow \text{Verb} \ \text{NP} \bullet$.

This completes the VP, which is added to the Agenda from position 3 to 6.
Example: 1 Green 2 water 3 hit 4 the 5 window 6 continued
Example: 1 Green 2 water 3 hit 4 the 5 window 6 continued

12. Next, we remove the VP from position 3 to 6 from the Agenda. This VP can be used to extend the S rule: $S \rightarrow NP \bullet VP$.
Hence, the VP1 constituent is added to the Chart from 3 to 6 and we extend the active arc as follows: $S \rightarrow NP VP \bullet$. This finishes the S rule, so we add an S to the Agenda from position 1 to 6.

13. Pop the S from the Agenda and place it on the Chart from position 1 to 6. This finishes the parse.

- This parser has a worst-case complexity of $O(G*n^3)$ just like the bottom-up version. In practice though, this tends to be more efficient.
- The top-down Chart parser is similar to the Earley Algorithm.
Example: 1 Green 2 water 3 hit 4 the 5 window 6 continued