Syntax

Constituency Parsing
(using Context Free Grammars)

A sample Context Free Grammar (CFG):

Grammar

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

Non-terminals

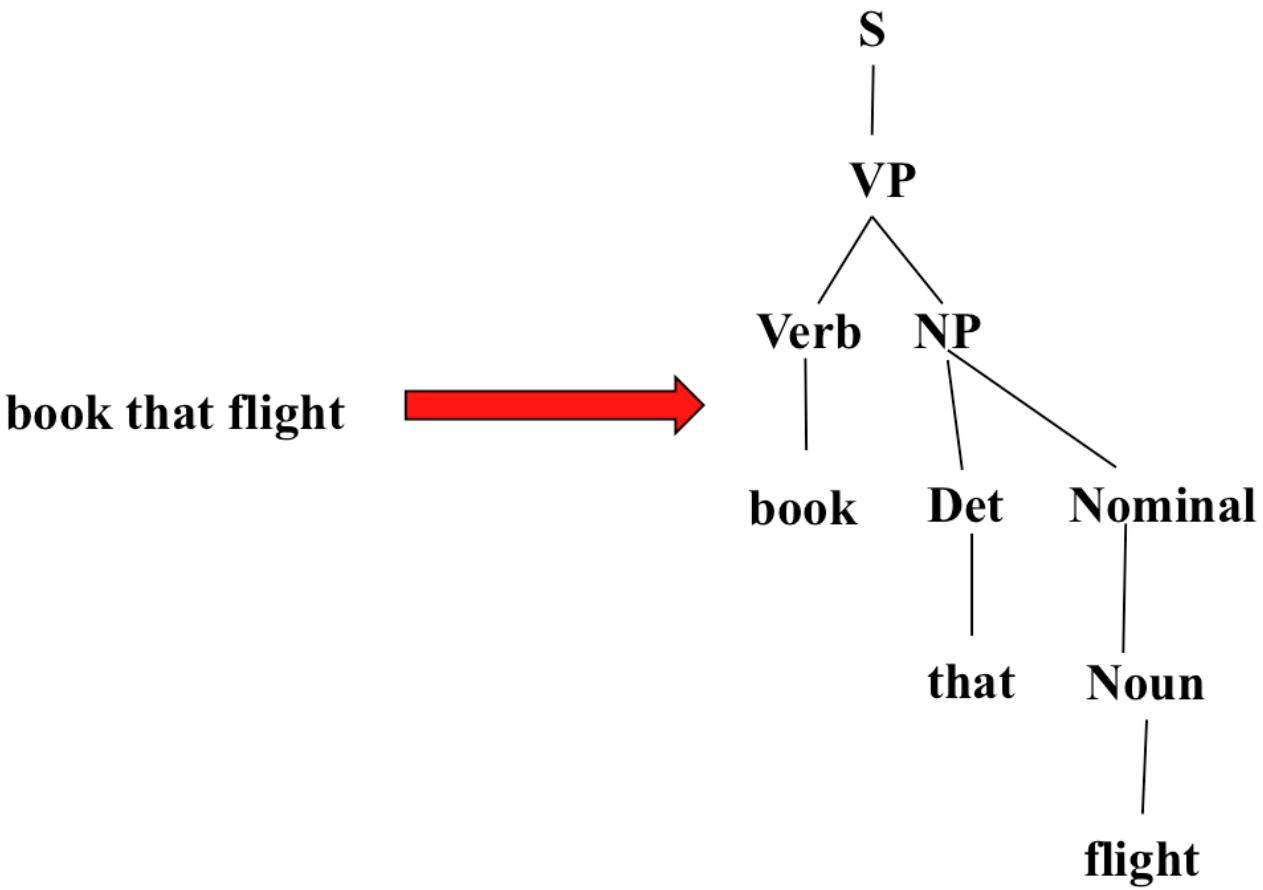
Lexicon

Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | he | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

Pre-terminals

Terminals

Parse tree of a string according to this CFG:



What is Parsing?

- The process of taking a string and a grammar and returning all possible parse trees for that string

A sentence can have multiple parse trees

What is Parsing?

- The process of taking a string and a grammar and returning all possible parse trees for that string
- That is, find all trees, whose root is the start symbol S, which cover exactly the words in the input

What is Parsing?

- The process of taking a string and a grammar and returning all possible parse trees for that string
- That is, find all trees, whose root is the start symbol S, which cover exactly the words in the input

What are the constraints? "book that flight"

- There must be three leaves, *book*, *that* and *flight*
- The tree must have one root, the start symbol S

What is Parsing?

- The process of taking a string and a grammar and returning all possible parse trees for that string
- That is, find all trees, whose root is the start symbol S, which cover exactly the words in the input

What are the constraints? "book that flight"

- There must be three leaves, *book*, *that* and *flight*
- The tree must have one root, the start symbol S
- Give rise to two search strategies: *top-down* (goal-oriented) and *bottom-up* (data-directed)

Top-down: start with S, expand using the grammar rules, try to find a tree that will have exactly these leaves
Bottom-up: start from each given word and build a tree upward, try to combine the trees using grammar rules, see if we can build up a full tree till S

Top-Down Parsing

- Searches for a parse tree by trying to build upon the root node S down to the leaves
- Start by assuming that the input can be derived by the designated start symbol S
- Find all trees that can start with S, by looking at the grammar rules with S on the left-hand side
- Trees are grown downward until they eventually reach the POS categories at the bottom

Top-Down Parsing

- Searches for a parse tree by trying to build upon the root node S down to the leaves
- Start by assuming that the input can be derived by the designated start symbol S
- Find all trees that can start with S, by looking at the grammar rules with S on the left-hand side
- Trees are grown downward until they eventually reach the POS categories at the bottom
- Trees whose leaves fail to match the words in the input can be rejected

Grammar

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → VP PP
PP → Prep NP

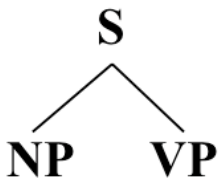
Lexicon

Det → the | a | that | this
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | he | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

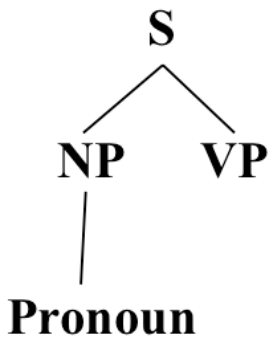
Top-Down Parsing

S

Top-Down Parsing

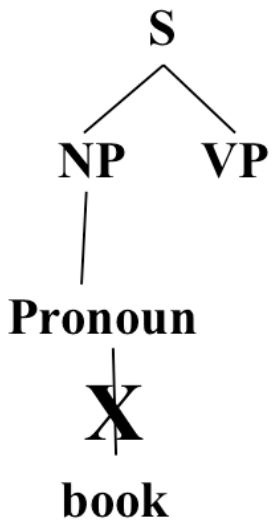


Top-Down Parsing



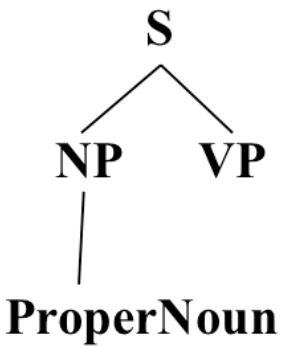
Pronoun is a pre-terminal

Top-Down Parsing

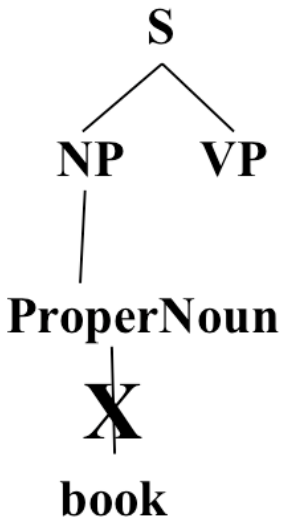


But a Pronoun cannot match with "book". So discard this path and try some other path.

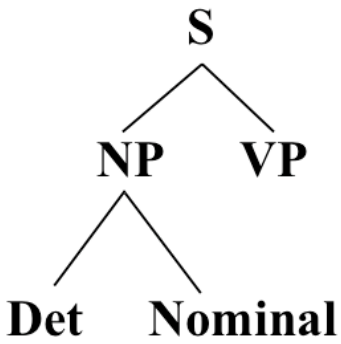
Top-Down Parsing



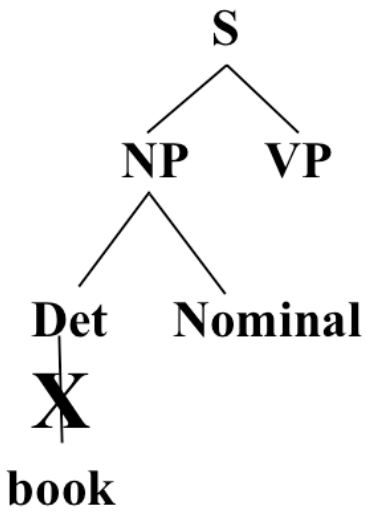
Top-Down Parsing



Top-Down Parsing

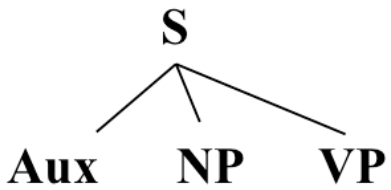


Top-Down Parsing

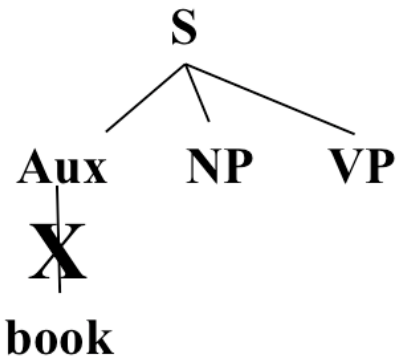


We have now tried all possible expansions for NP, but none matched with "book". Hence we have to discard the rule S -> NP VP and try some other expansion of S.

Top-Down Parsing

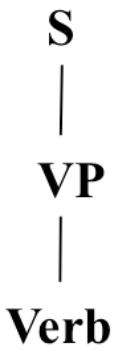


Top-Down Parsing



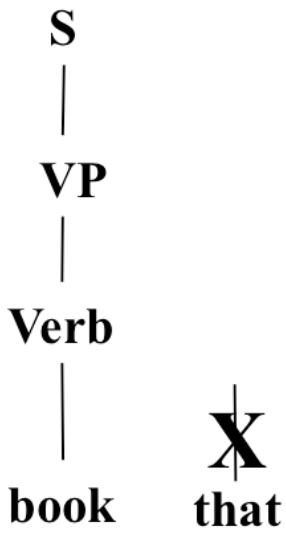
Top-Down Parsing

Top-Down Parsing



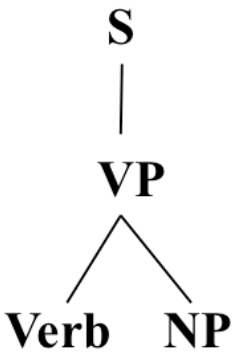
Top-Down Parsing

Top-Down Parsing

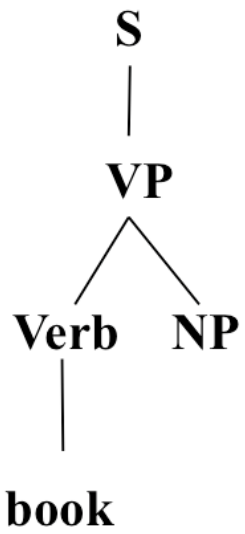


This expansion matches the first word "book" but cannot match the rest of the text. So we need to discard S -> VP and try some other rule.

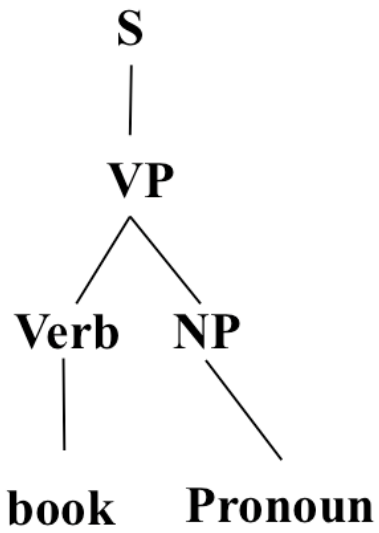
Top-Down Parsing



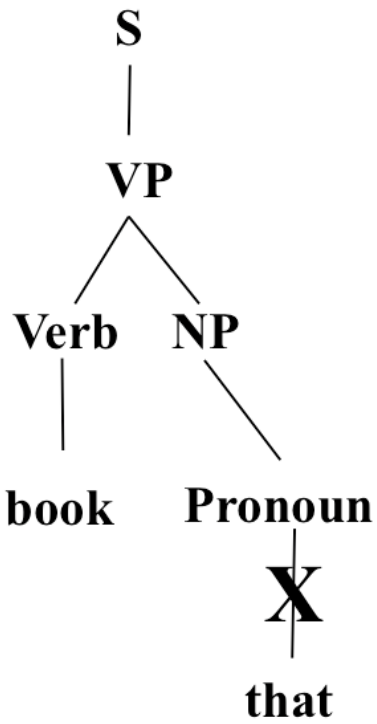
Top-Down Parsing



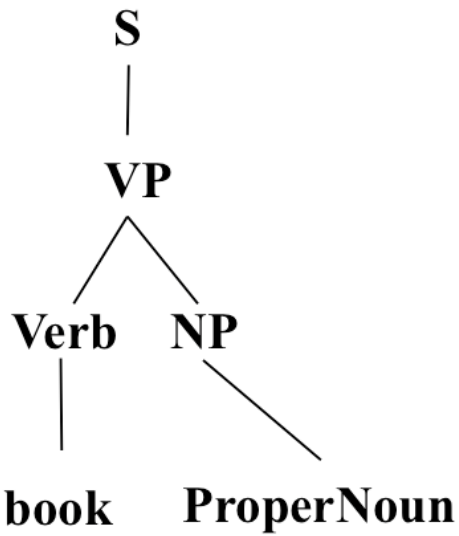
Top-Down Parsing



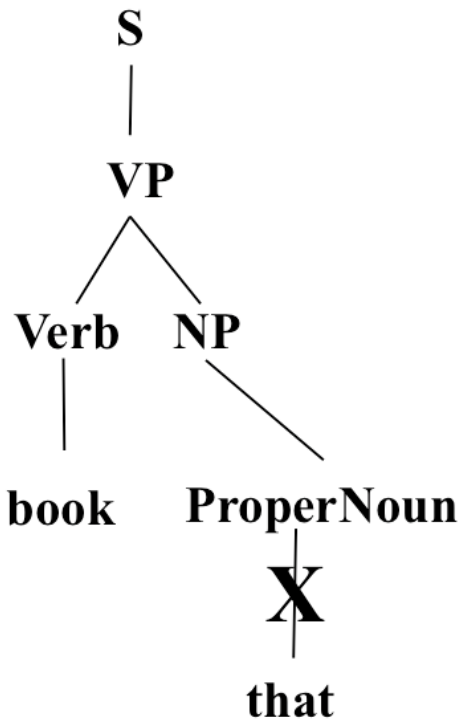
Top-Down Parsing



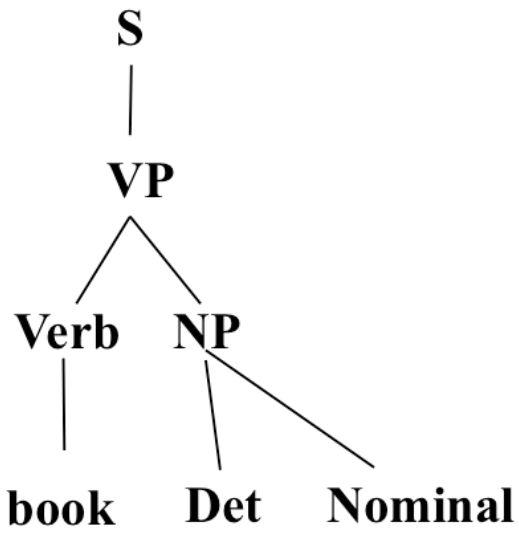
Top-Down Parsing



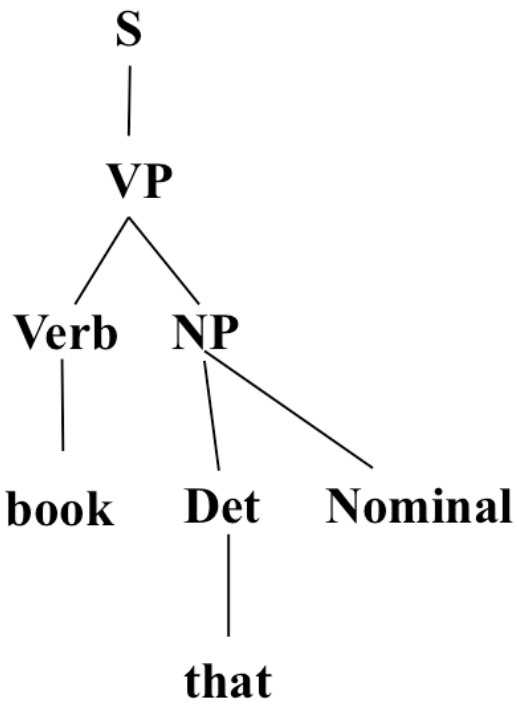
Top-Down Parsing



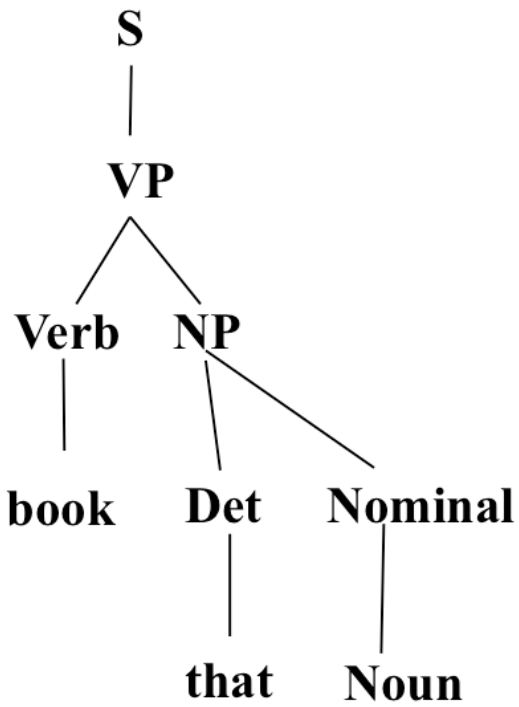
Top-Down Parsing



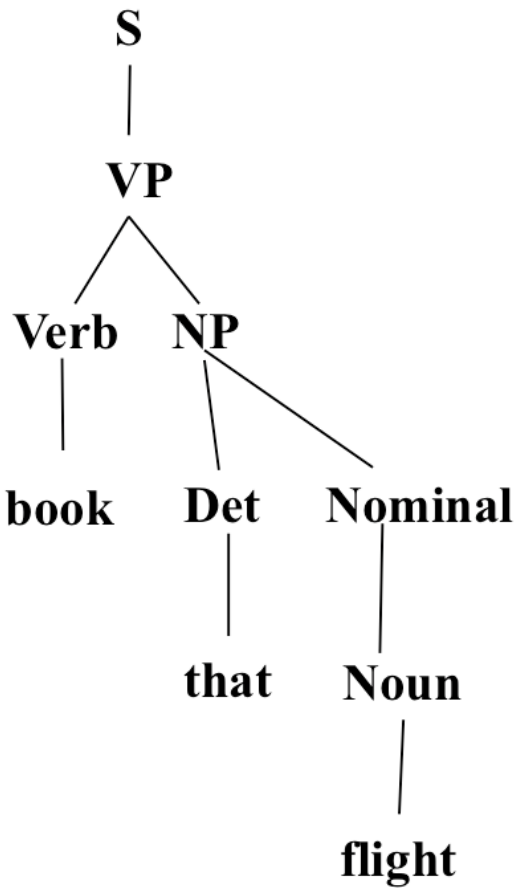
Top-Down Parsing



Top-Down Parsing



Top-Down Parsing



So, by systematically following the grammar rules, we can generate a parse tree that covers exactly the given text.

We can try ordering the grammar rules such that more frequently used rules are tried earlier.

Still, top-down parsing can involve trying many steps / derivations that will not lead to the given text.

Bottom-Up Parsing

- The parser starts with the words of the input, and tries to build trees from the words up, by applying rules from the grammar one at a time
- Parser looks for the places in the parse-in-progress where the right-hand-side of some rule might fit.

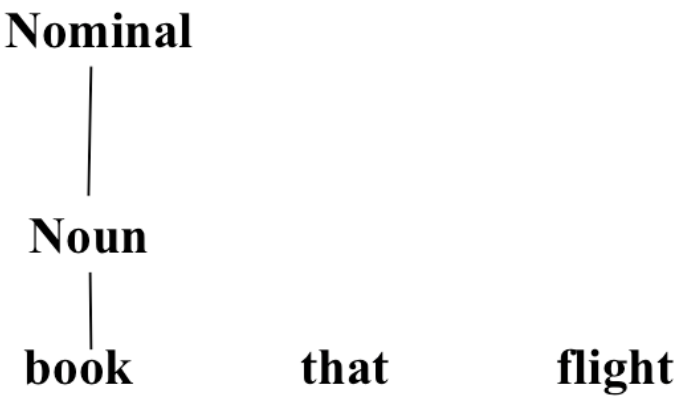
Bottom-Up Parsing

book **that** **flight**

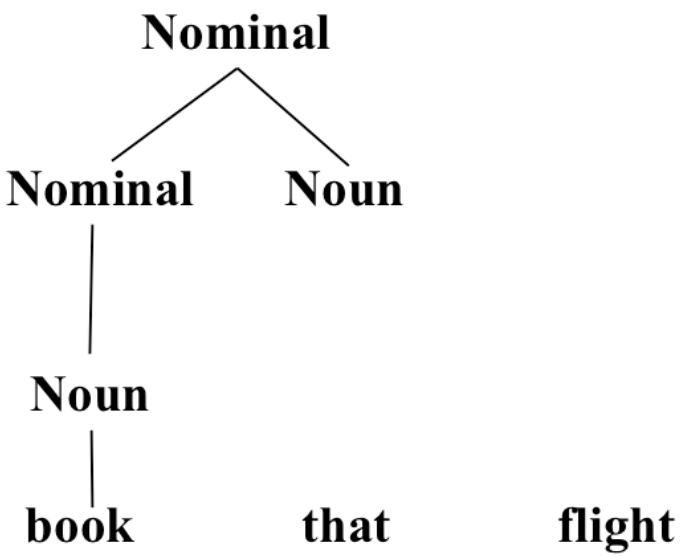
Bottom-Up Parsing

We will start building parse trees upward starting from each given word, and later see if we can join the different trees as per the grammar rules.

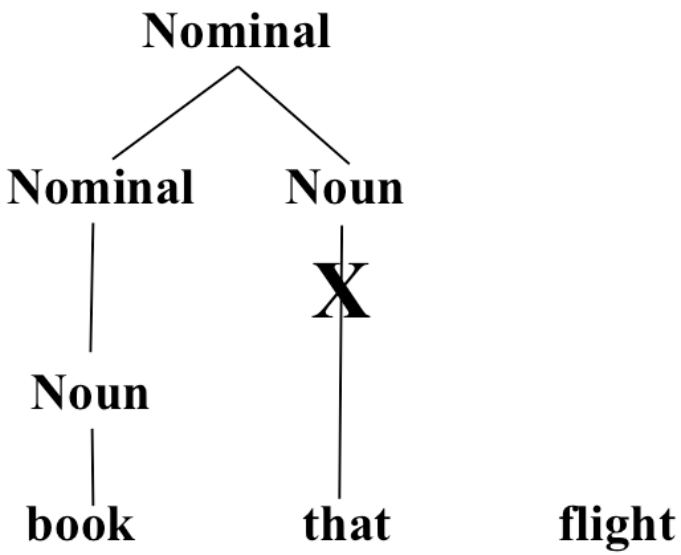
Bottom-Up Parsing



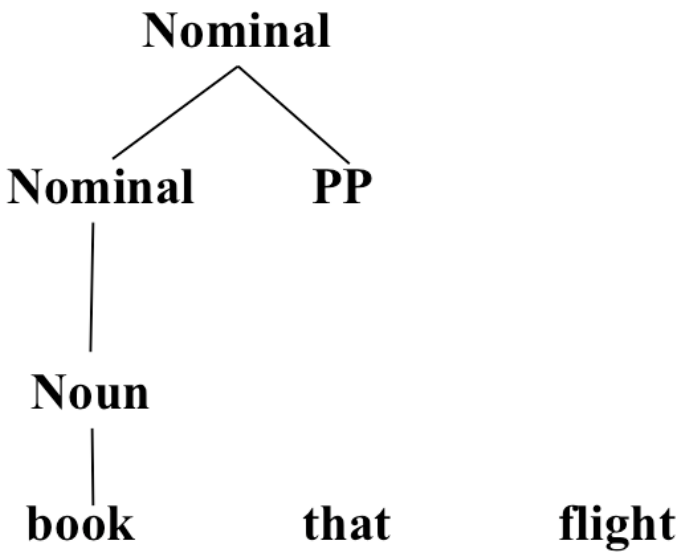
Bottom-Up Parsing



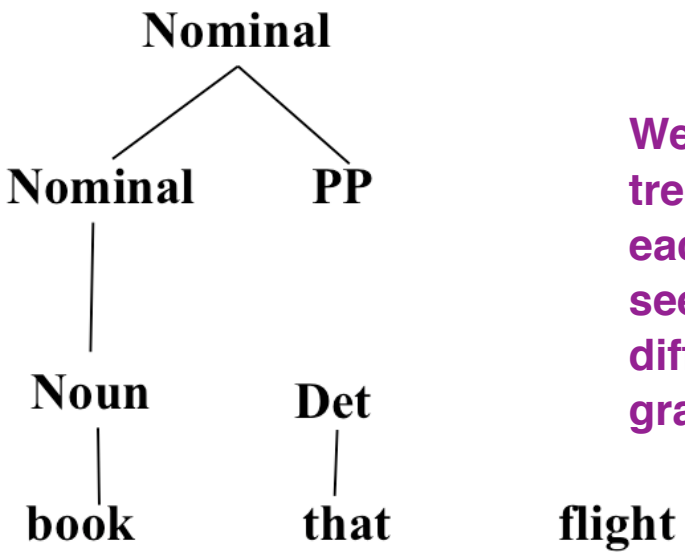
Bottom-Up Parsing



Bottom-Up Parsing

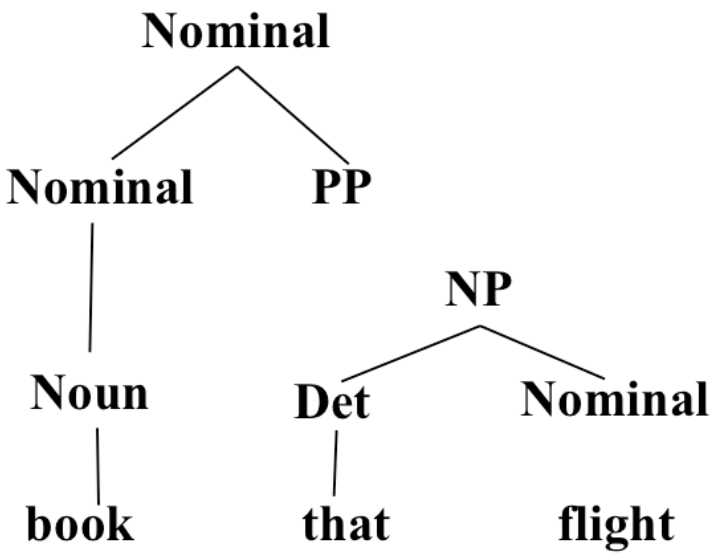


Bottom-Up Parsing

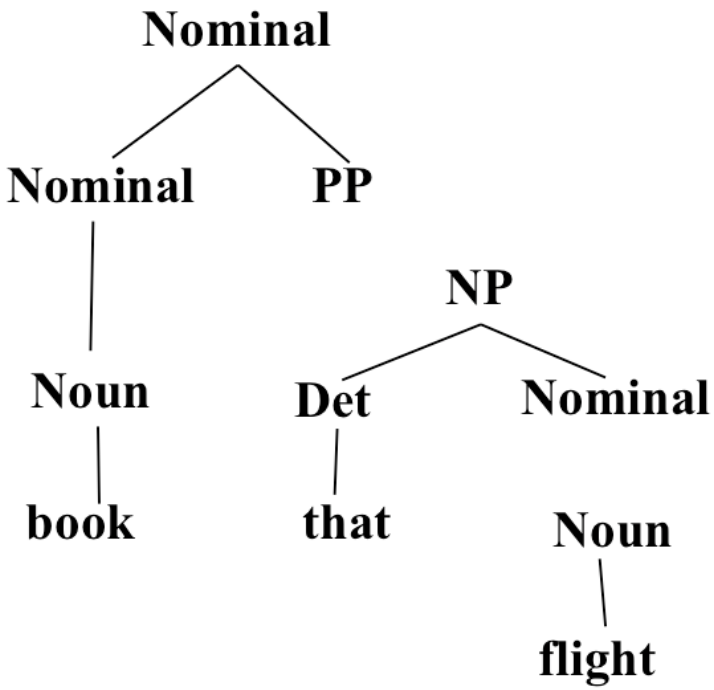


We will start building parse trees upward starting from each given word, and later see if we can join the different trees as per the grammar rules.

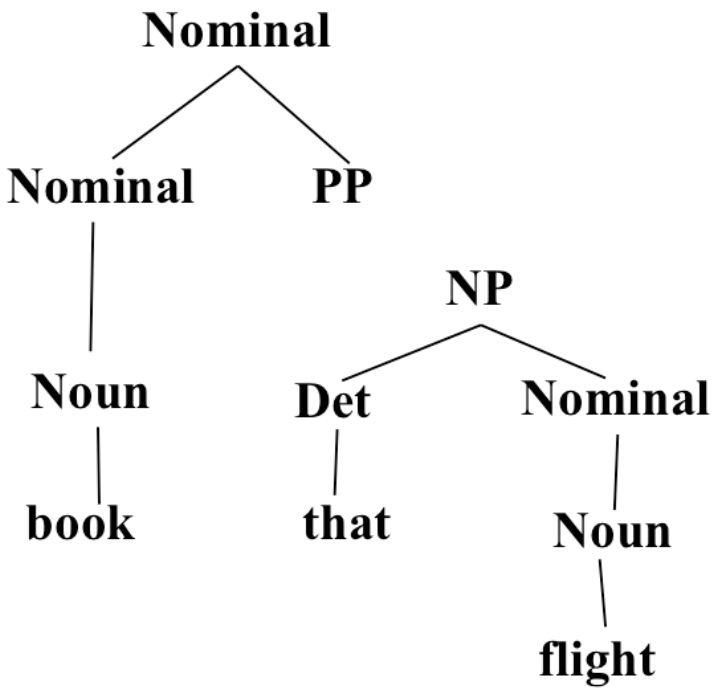
Bottom-Up Parsing



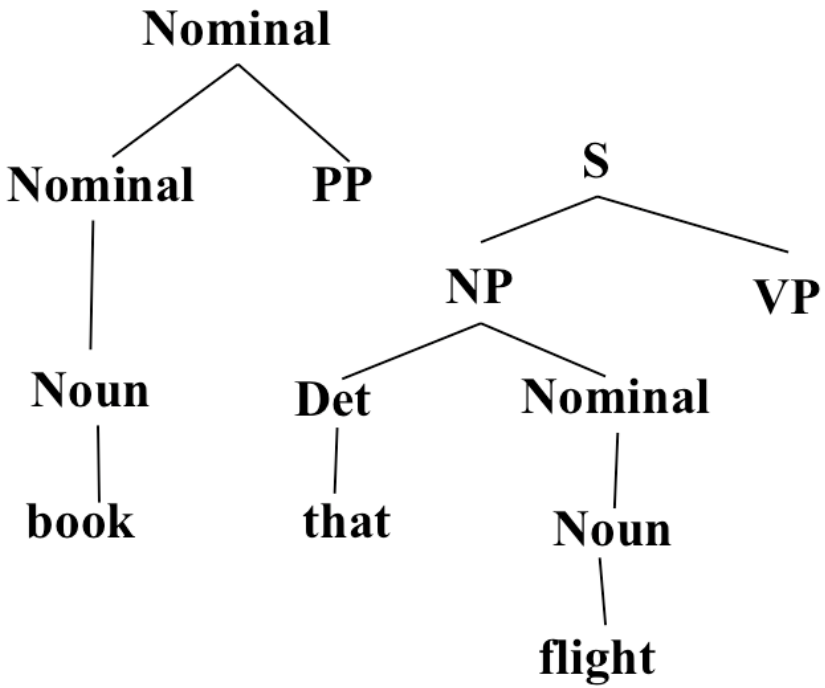
Bottom-Up Parsing



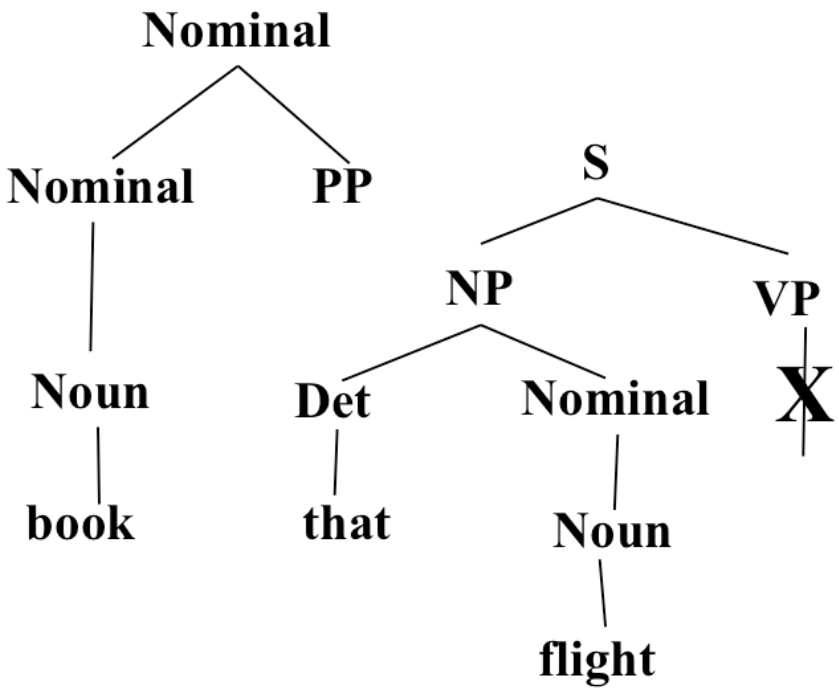
Bottom-Up Parsing



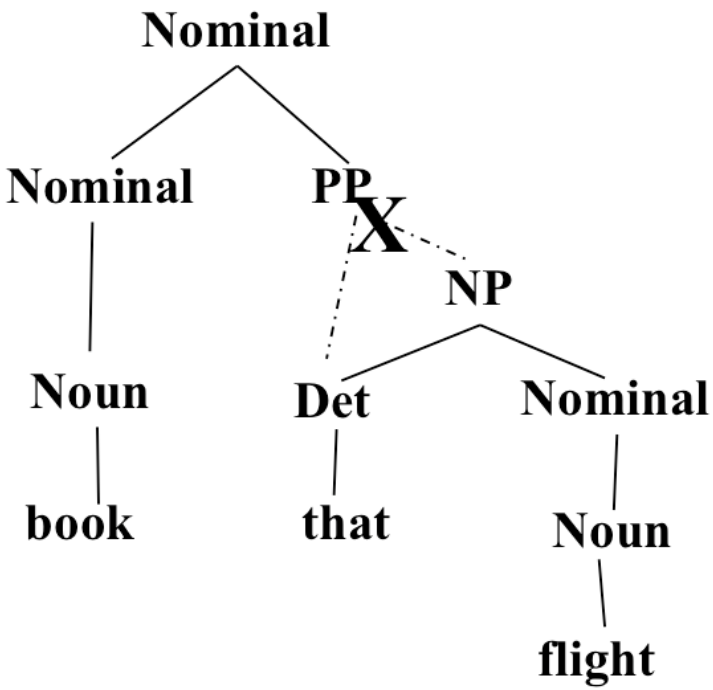
Bottom-Up Parsing



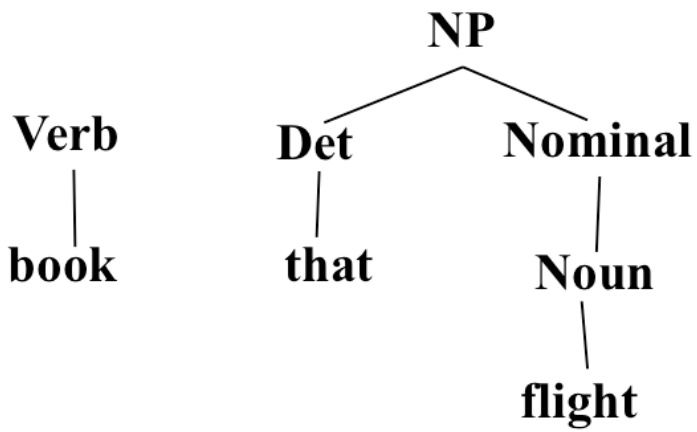
Bottom-Up Parsing



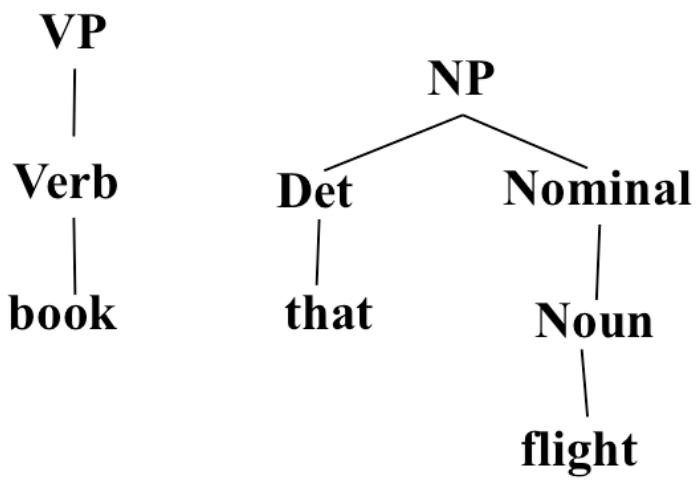
Bottom-Up Parsing



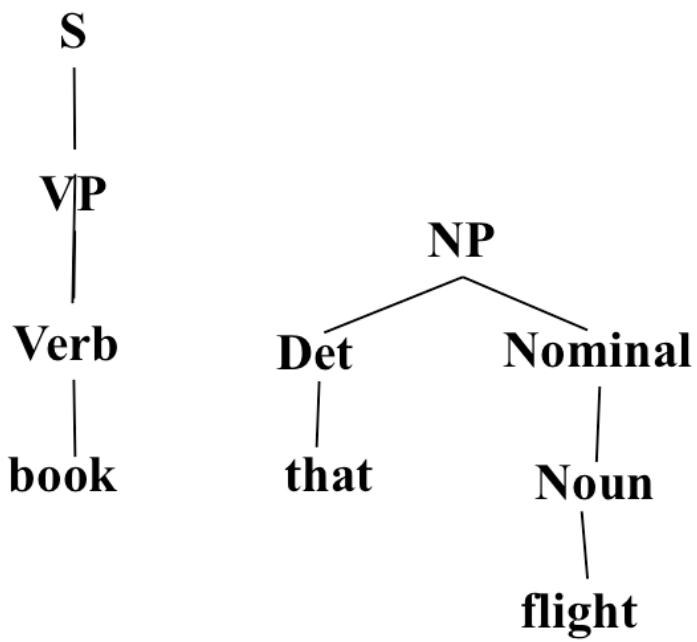
Bottom-Up Parsing



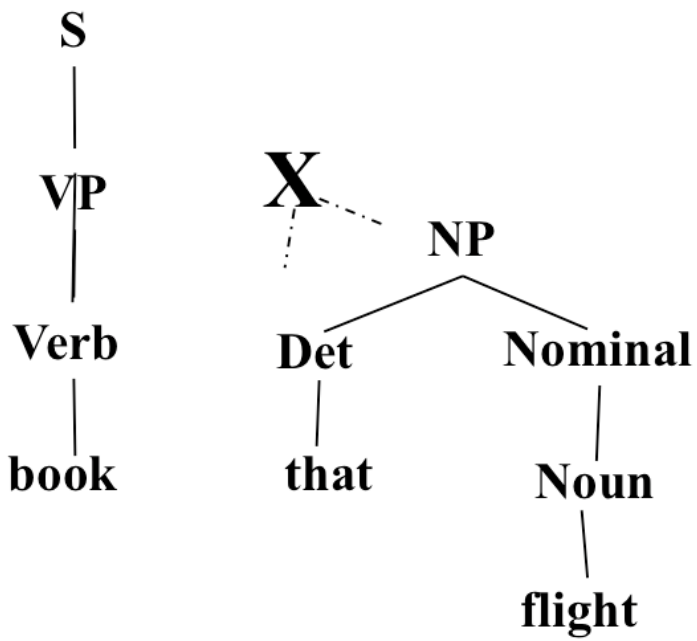
Bottom-Up Parsing



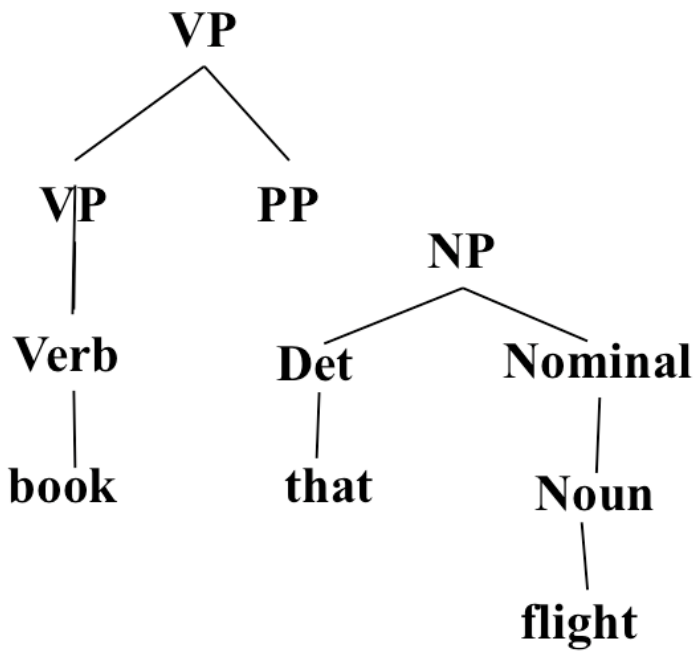
Bottom-Up Parsing



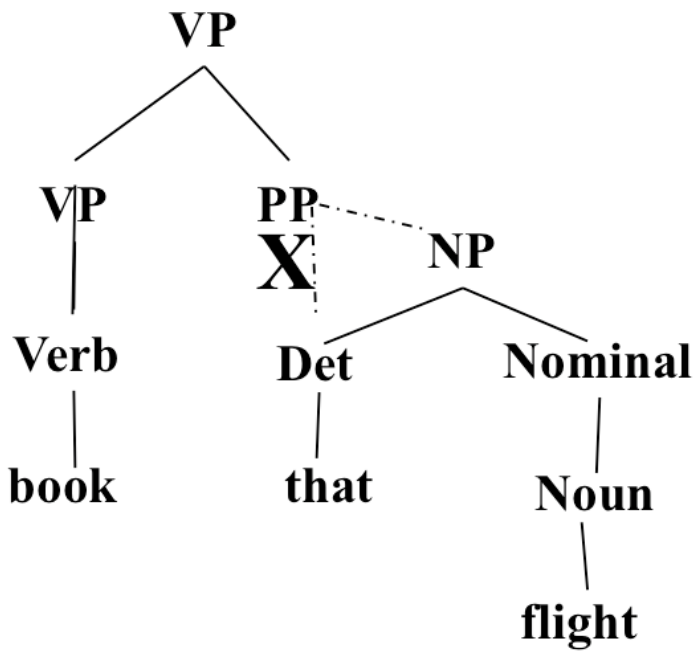
Bottom-Up Parsing



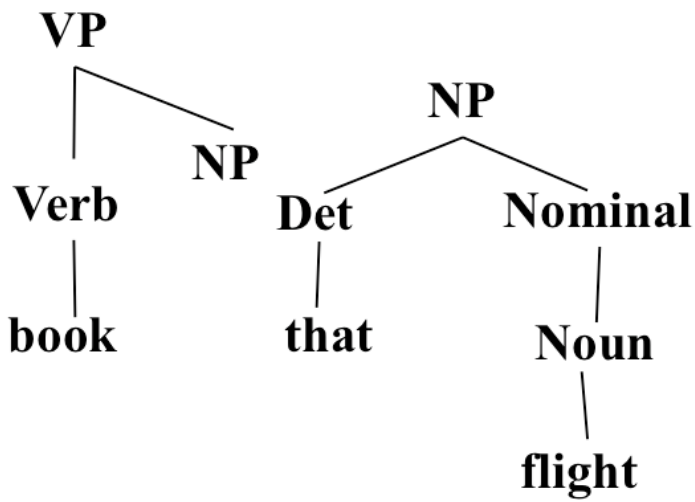
Bottom-Up Parsing



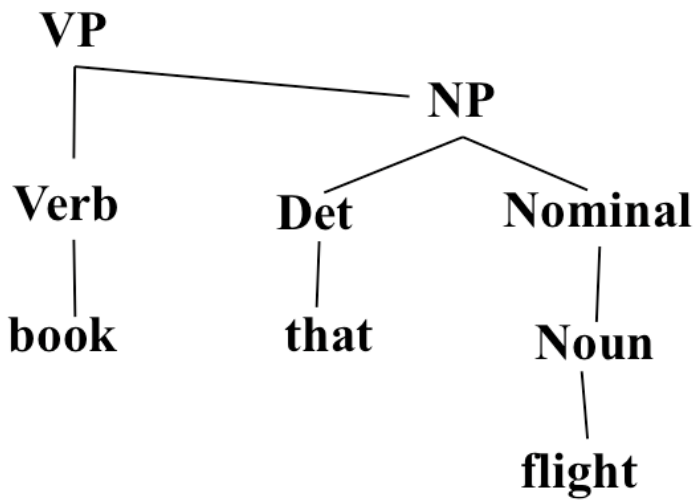
Bottom-Up Parsing



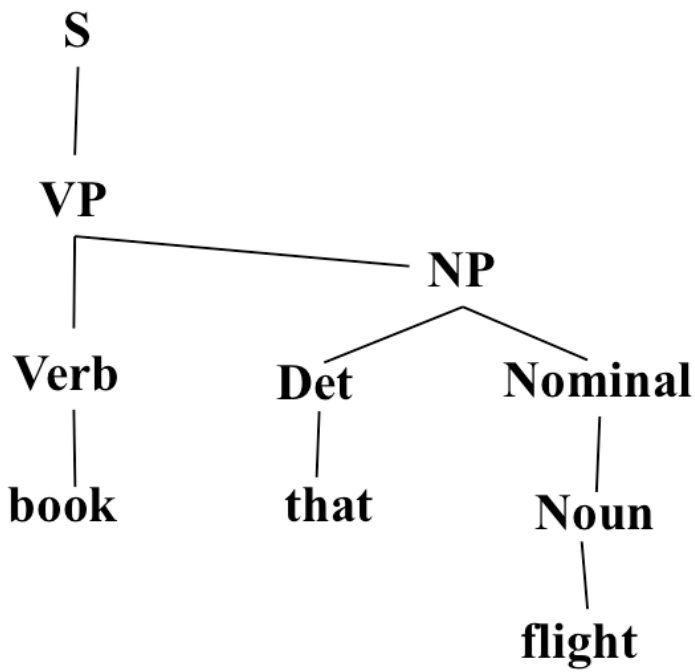
Bottom-Up Parsing



Bottom-Up Parsing



Bottom-Up Parsing



Top-Down vs. Bottom-Up

Top-Down vs. Bottom-Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.

Top-Down vs. Bottom-Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.
- Bottom up never explores options that do not connect to the actual sentence but can explore options that can never lead to a full parse.

Top-Down vs. Bottom-Up

- Top down never explores options that will not lead to a full parse, but can explore many options that never connect to the actual sentence.
- Bottom up never explores options that do not connect to the actual sentence but can explore options that can never lead to a full parse.
- Relative amounts of wasted search depend on how much the grammar branches in each direction.

Dynamic Programming Parsing

Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.

Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.
- Caching (memoizing) critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs.

Dynamic Programming Parsing

- To avoid extensive repeated work, must cache intermediate results, i.e. completed phrases.
- Caching (memoizing) critical to obtaining a polynomial time parsing (recognition) algorithm for CFGs.
- Dynamic programming algorithms based on both top-down and bottom-up search can achieve $O(n^3)$ recognition time where n is the length of the input string.

Dynamic Programming Parsing Methods

- CKY (Cocke-Kasami-Younger) algorithm: bottom-up, requires normalizing the grammar

Requires the grammar to be in Chomsky Normal Form

Dynamic Programming Parsing Methods

- CKY (Cocke-Kasami-Younger) algorithm: bottom-up, requires normalizing the grammar
- Earley Parser - top-down, does not require normalizing grammar, more complex