SWE1017

Natural Language Processing

Venue:AB2-205 *Topic: Basic Parsing Strategy*

Prof. Tulasi Prasad Sariki,

SCSE, VIT Chennai Campus

www.learnersdesk.weebly.com

Simple CFG for ATIS English

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

A Fragment of English Grammar

S → NP VP

VP → V NP

NP → NNP | ART N

NNP → Ram

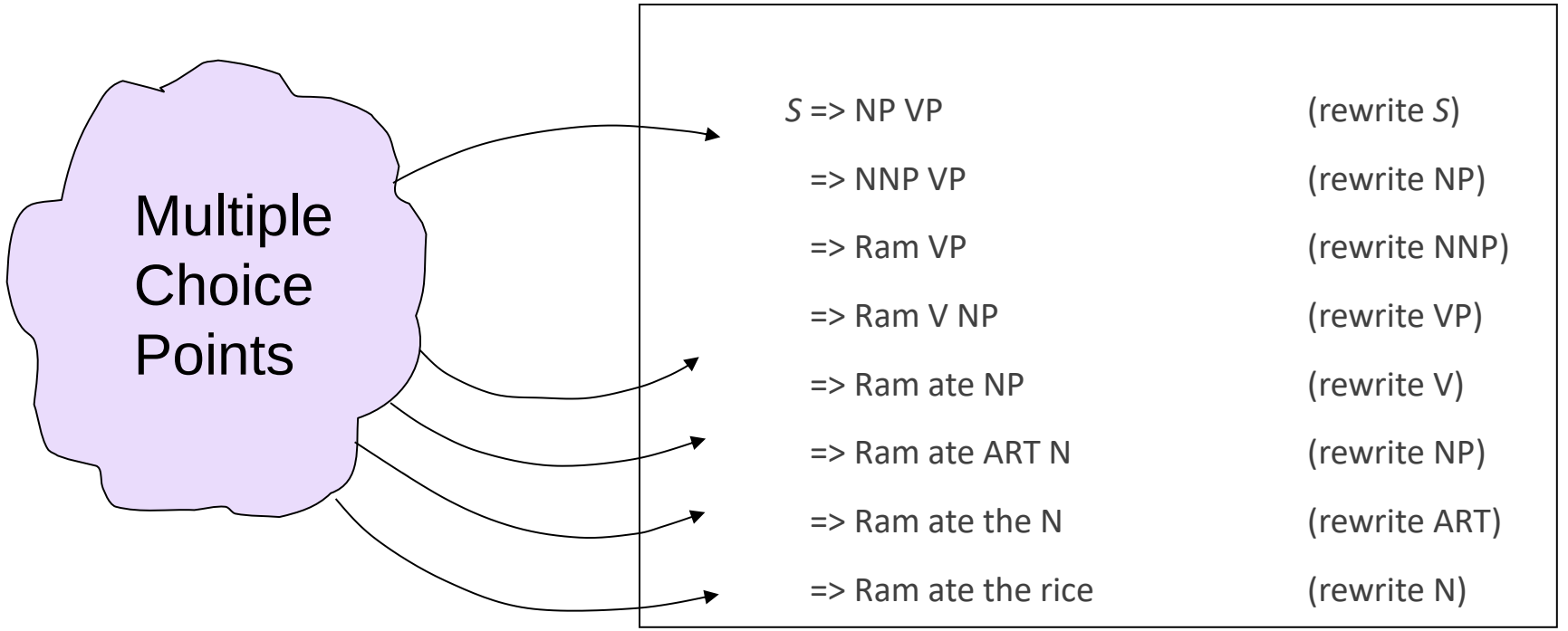
V → ate | saw

ART → a | an | the

N → rice | apple | movie

Derivation

- *S* is a special symbol called start symbol.



Two Strategies : Top-Down & Bottom-Up

Top down : Start with *S* and generate the sentence.

Bottom up : Start with the words in the sentence and use the rewrite rules backwards to reduce the sequence of symbols to produce *S*.

Previous slide showed top-down strategy.

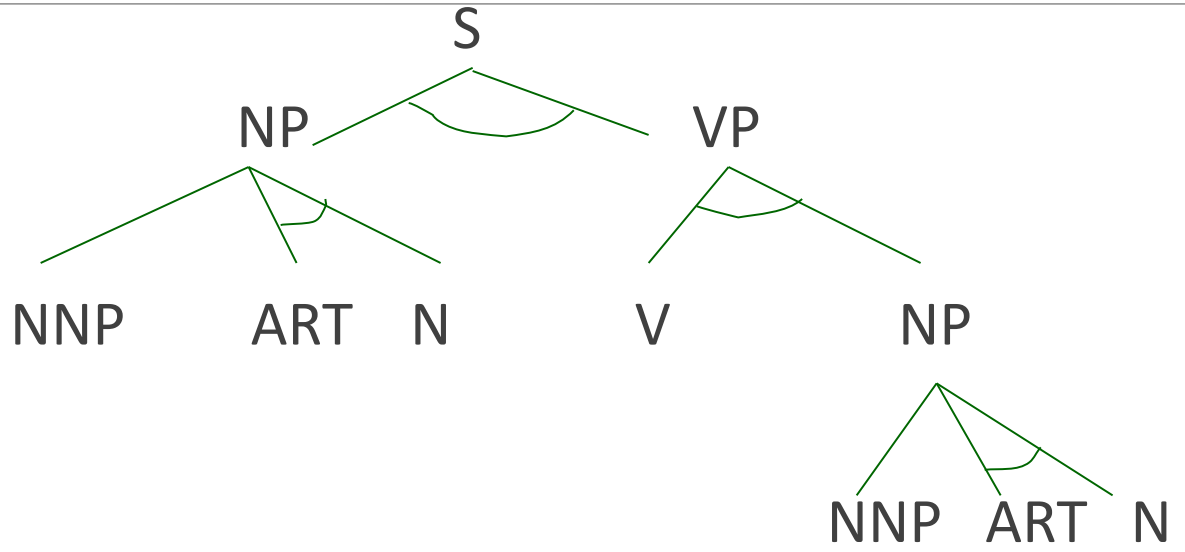
Bottom-Up Derivation

	Ram ate the rice	
=>	NNP ate the rice	(rewrite Ram)
=>	NNP V the rice	(rewrite ate)
=>	NNP V ART rice	(rewrite the)
=>	NNP V ART N	(rewrite rice)
=>	NP V ART N	(rewrite NNP)
=>	NP V NP	(rewrite ART N)
=>	NP VP	(rewrite V NP)
=>	S	

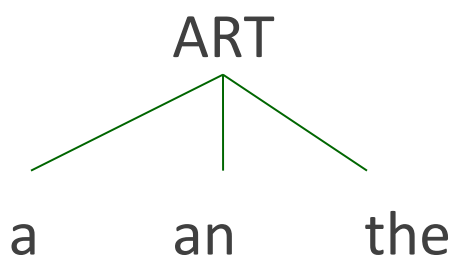
Parsing Algorithm

A procedure that “searches” through the grammatical rules to find a combination that generates a tree which stands for the structure of the sentence.

Parsing as Search (State Space : AND–OR Graph)



The leaves have links to words in the language, *e.g.,*



→ AND node

→ OR node

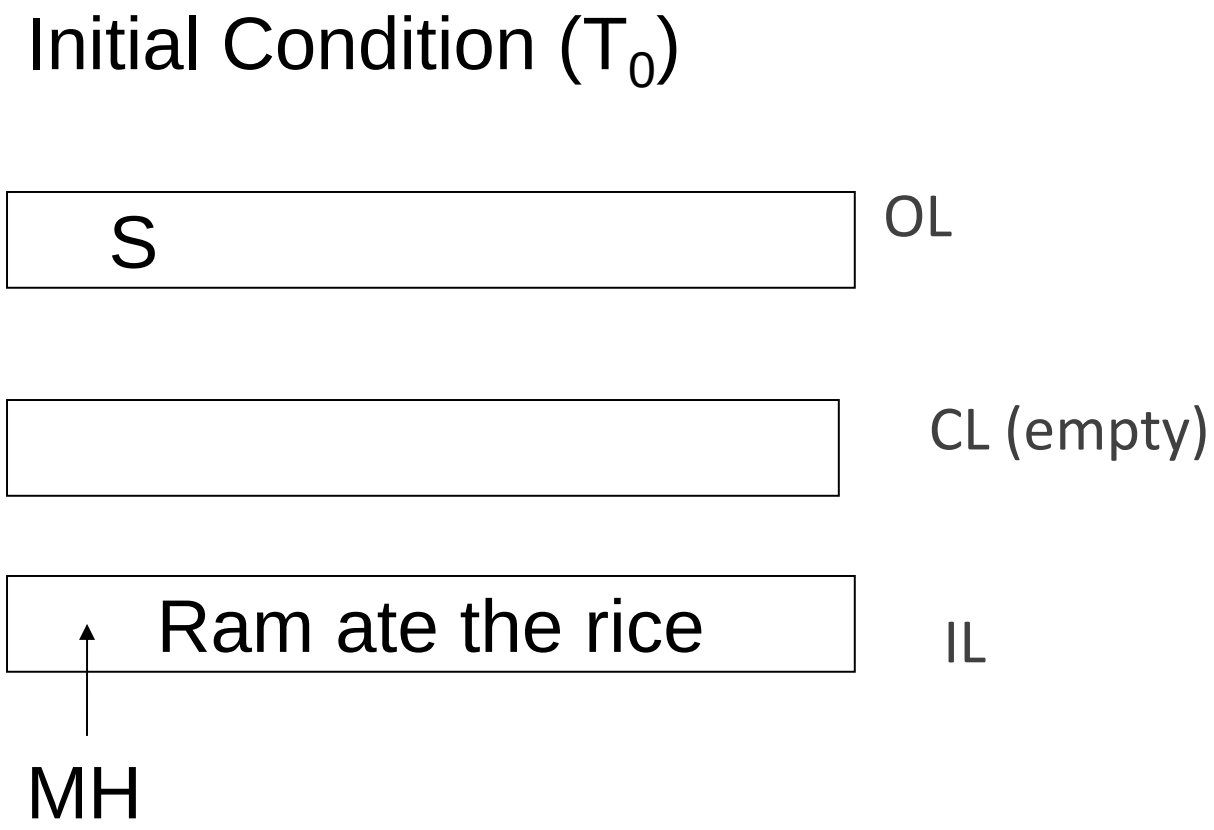
Top-Down Parsing

DFS on the AND-OR graph

Data structures:

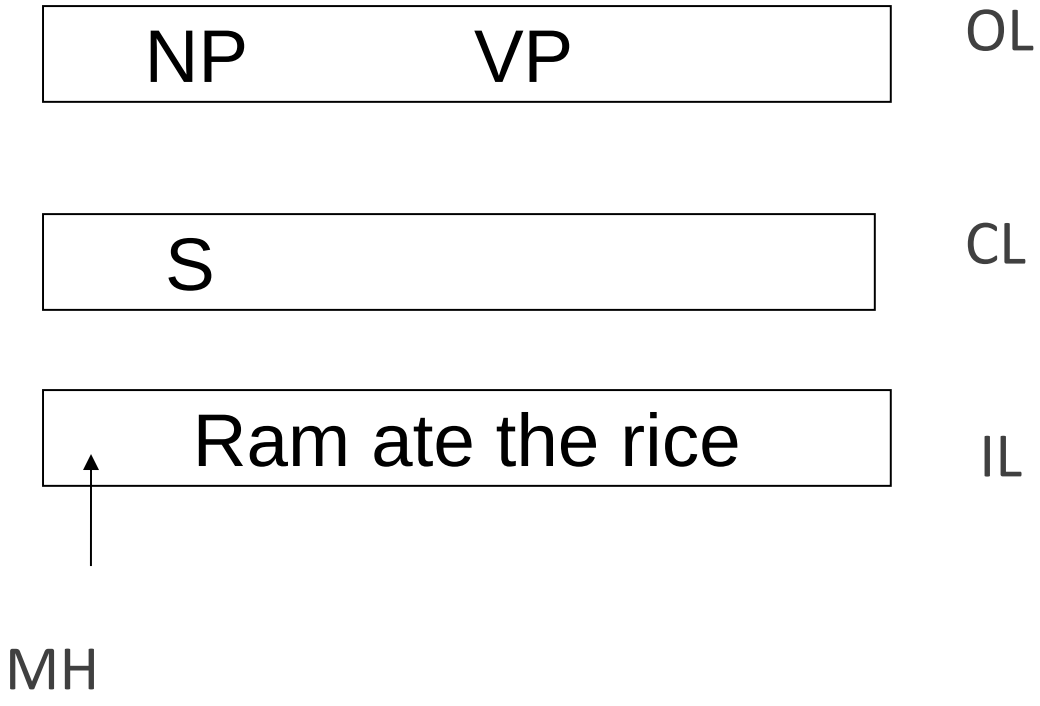
- *Open List (OL)*: Nodes to be expanded
- *Closed List (CL)*: Expanded Nodes
- *Input List (IL)*: Words of sentence to be parsed
- *Moving Head (MH)*: Walks over the IL

Trace of Top-Down Parsing

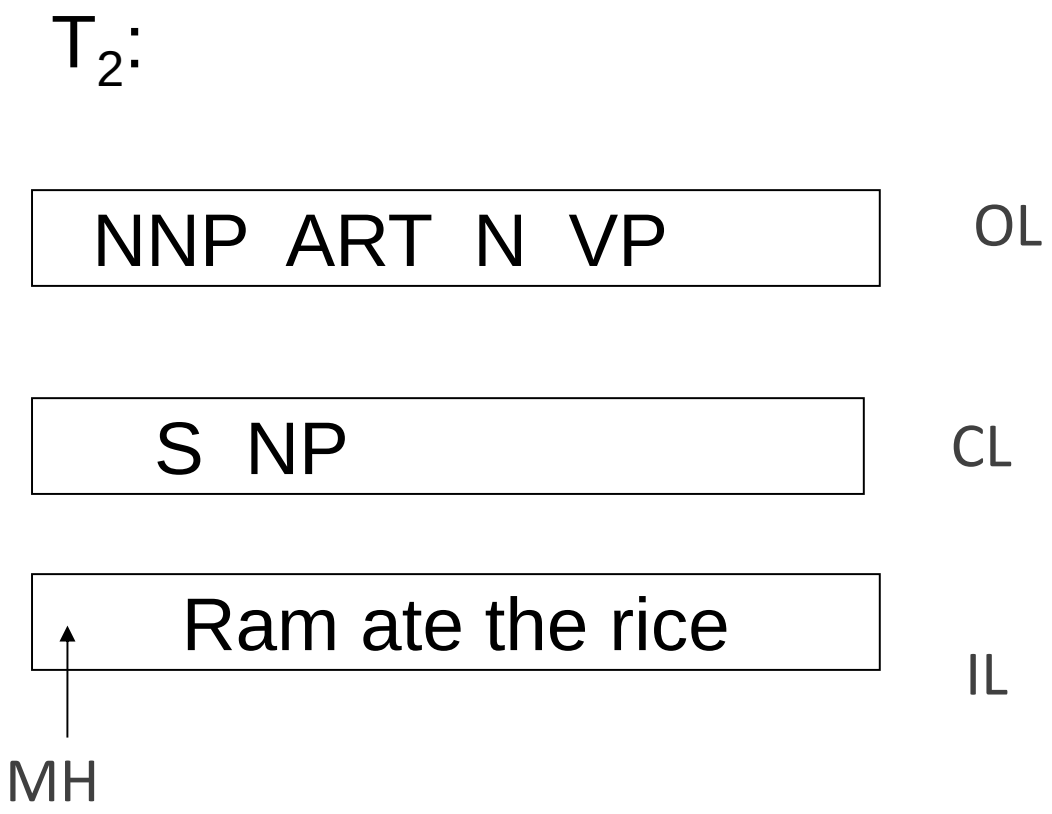


Trace of Top-Down Parsing

T_1:

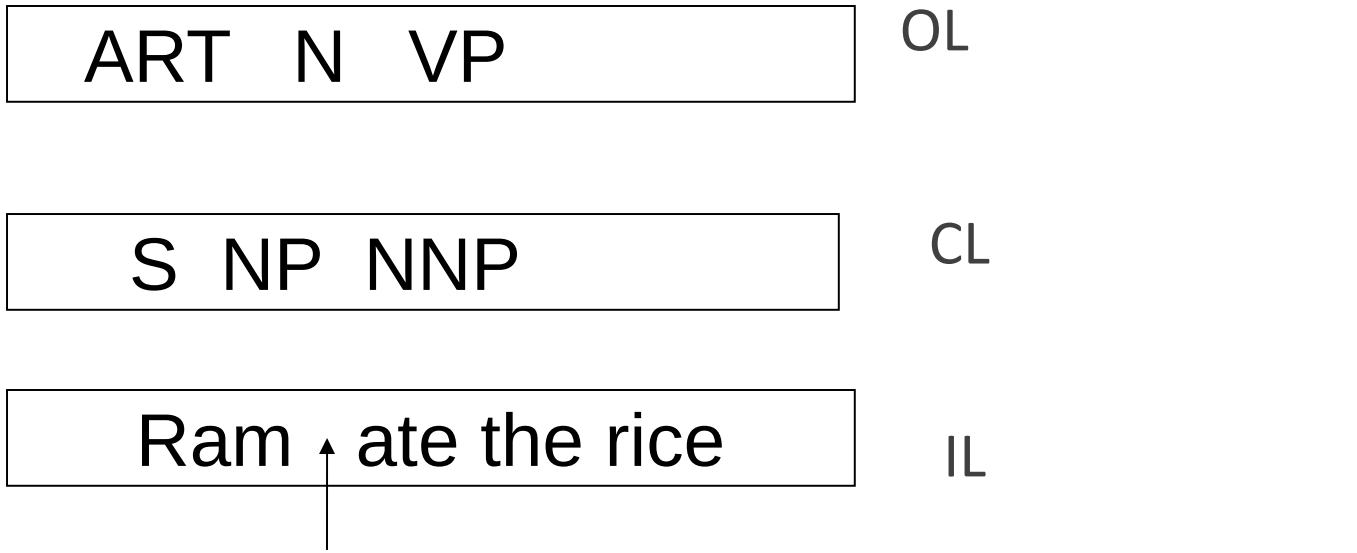


Trace of Top-Down Parsing



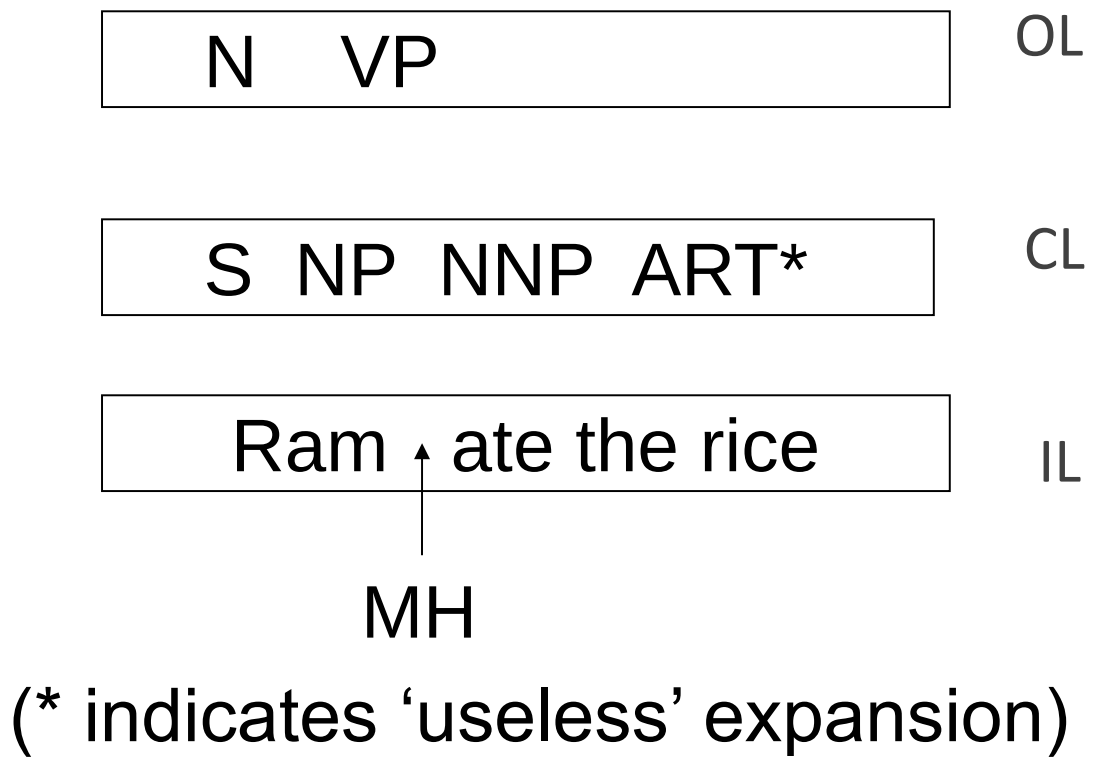
Trace of Top-Down Parsing

T_3:



Trace of Top-Down Parsing

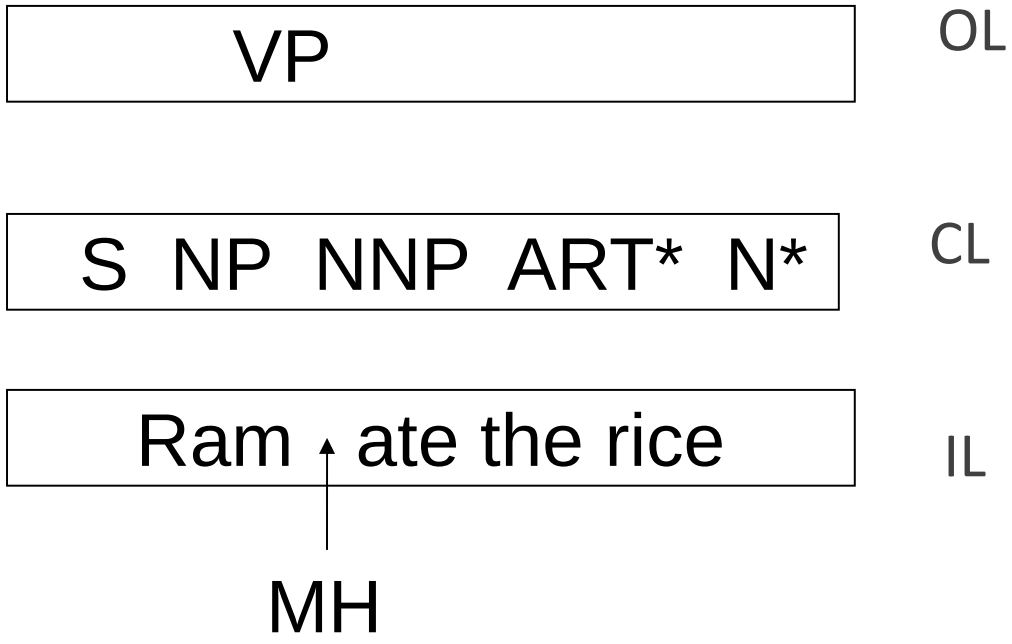
T_4:



(* indicates ‘useless’ expansion)

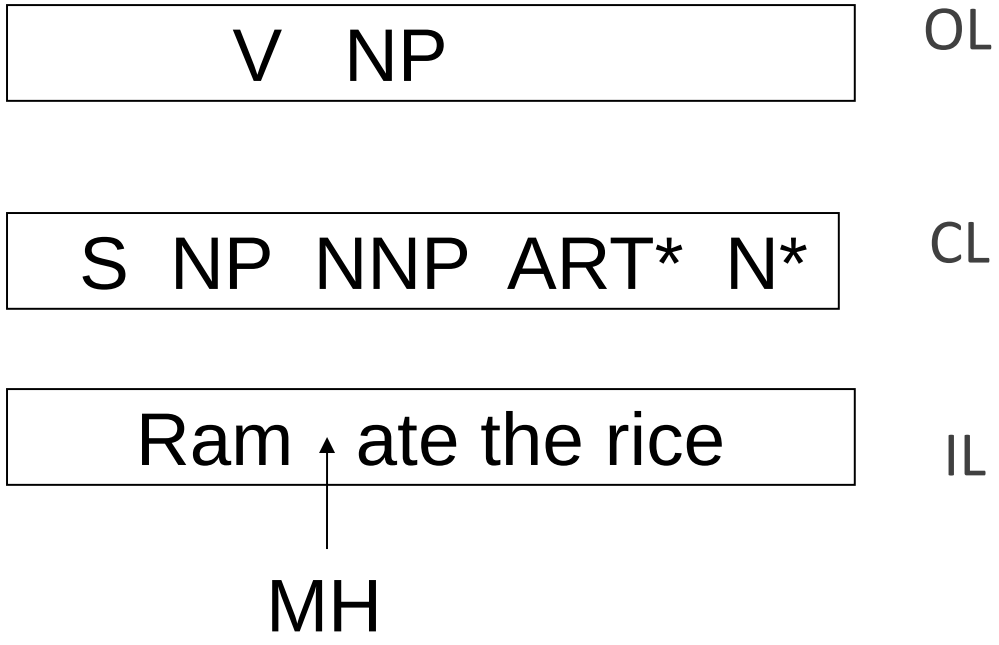
Trace of Top-Down Parsing

T_5:



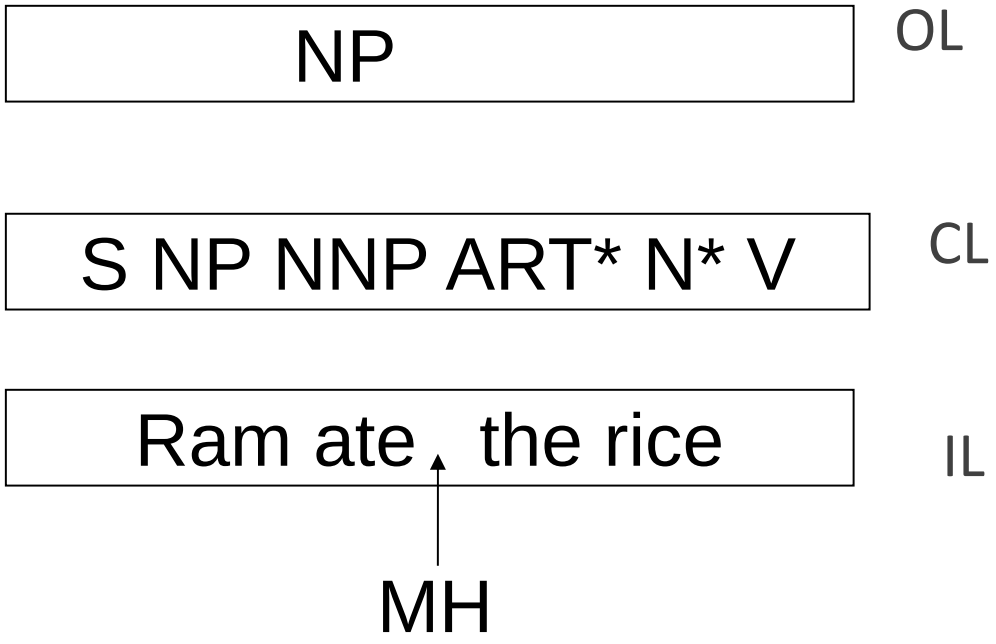
Trace of Top-Down Parsing

T_6:



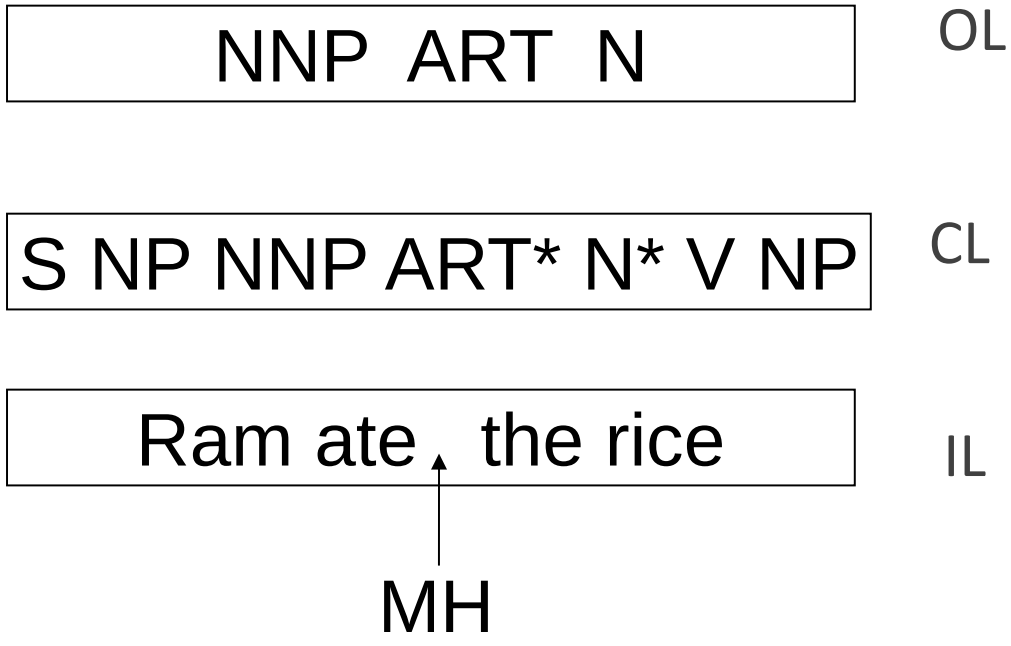
Trace of Top-Down Parsing

T_7:



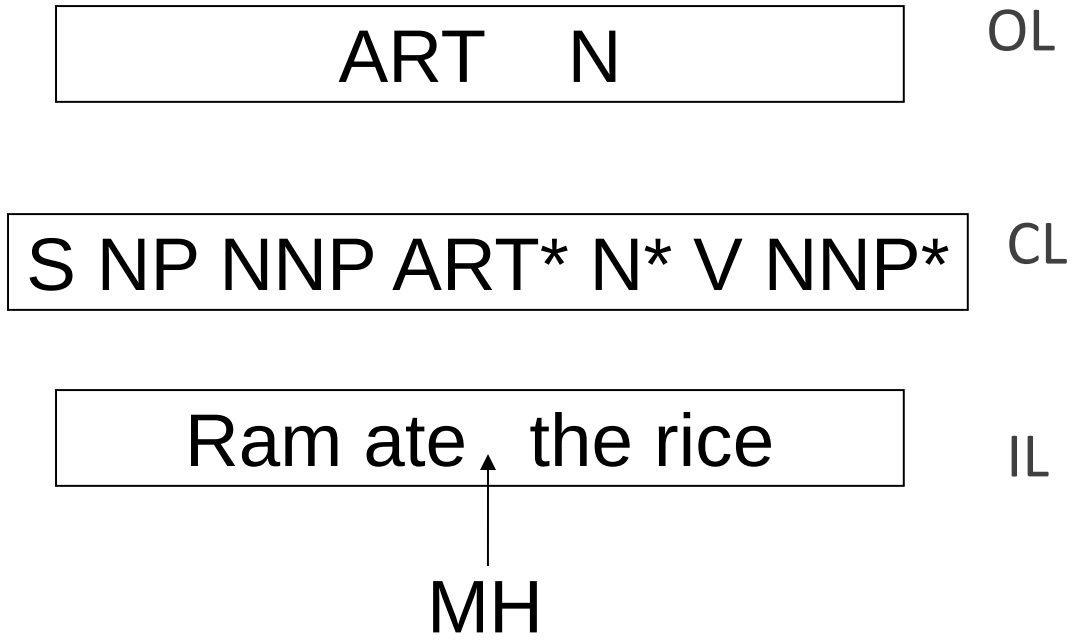
Trace of Top-Down Parsing

T_8:



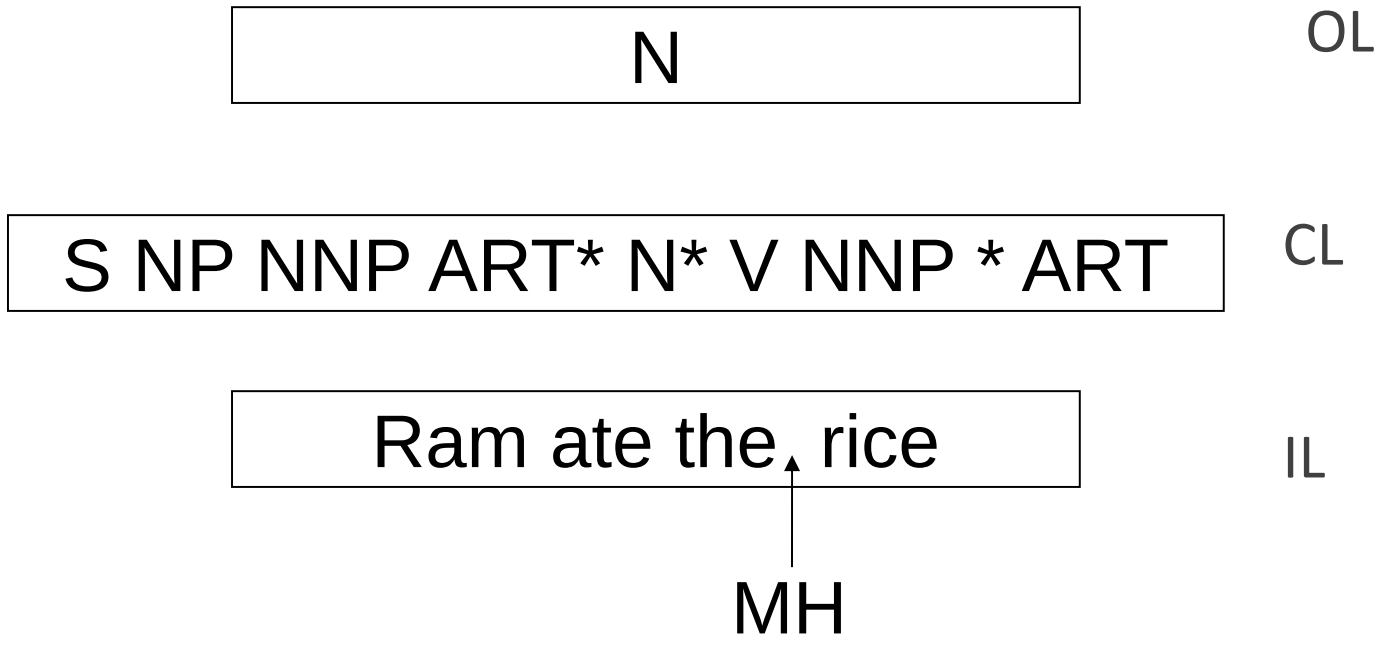
Trace of Top-Down Parsing

T_9:

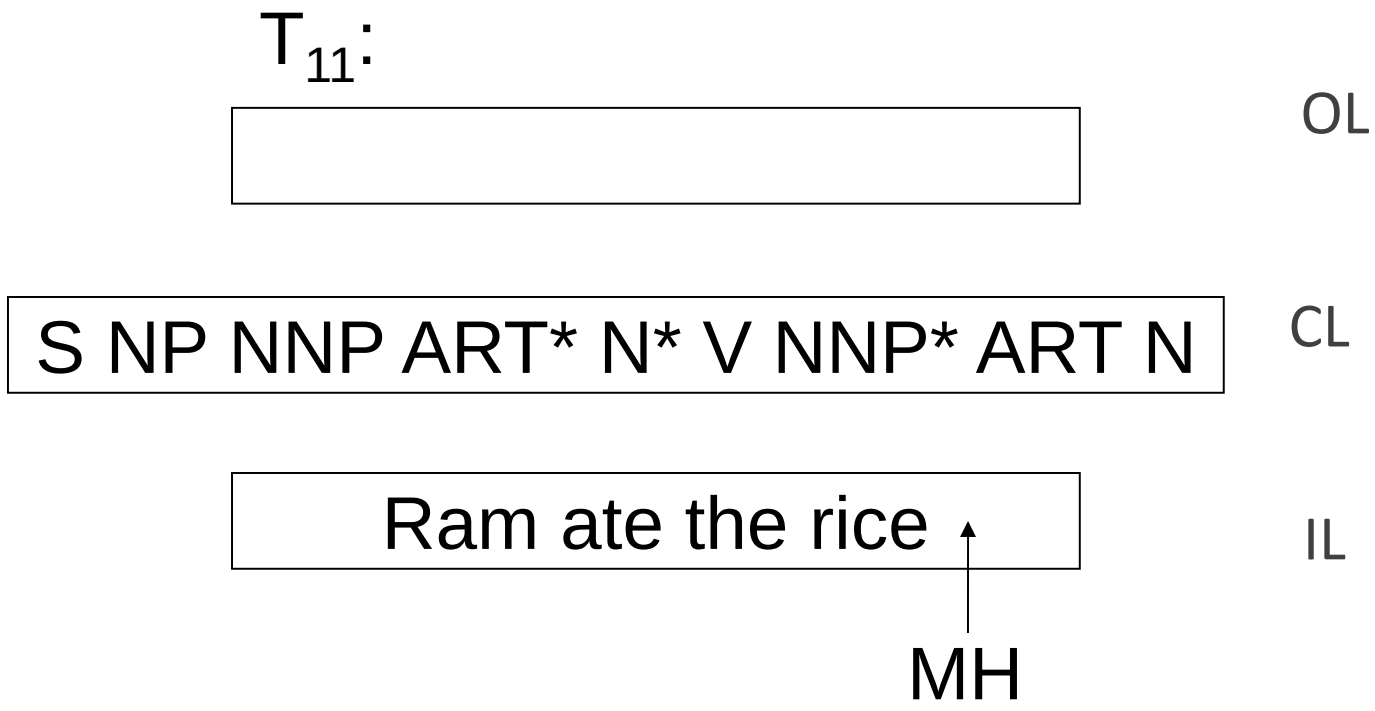


Trace of Top-Down Parsing

T_{10}:



Trace of Top-Down Parsing



Successful Termination: OL empty AND MH at the end of IL.

Bottom-Up Parsing

Basic idea:

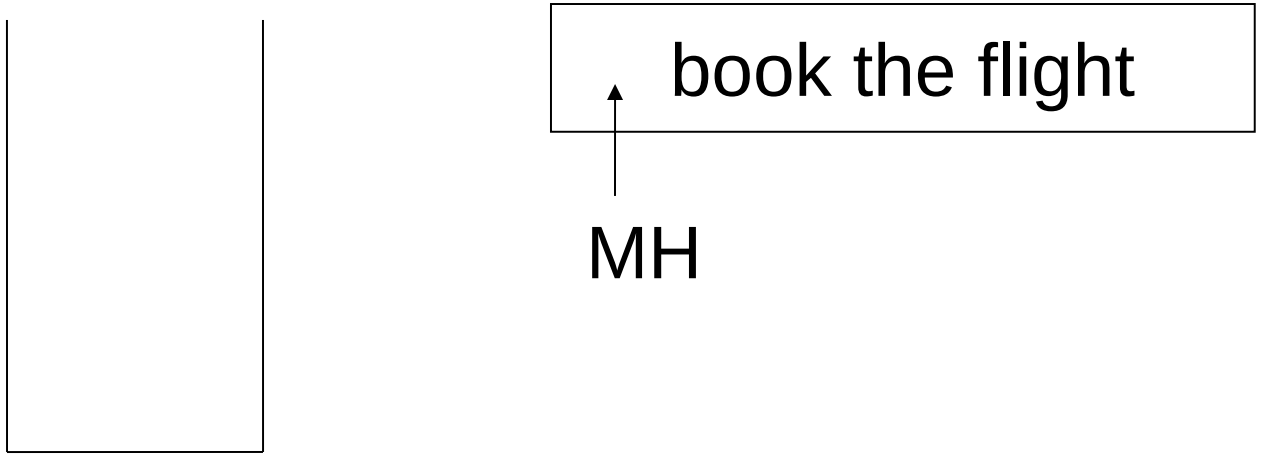
- Refer to words from the lexicon.
- Obtain all POSs for each word.
- Keep combining until *S* is obtained.

Implementation of Bottom-up Parsing

- Through a stack
- Push words into the stack
- Look for a “handle” to reduce to a non-terminal
- Termination by “start symbol on stack” and “end of sentence”.

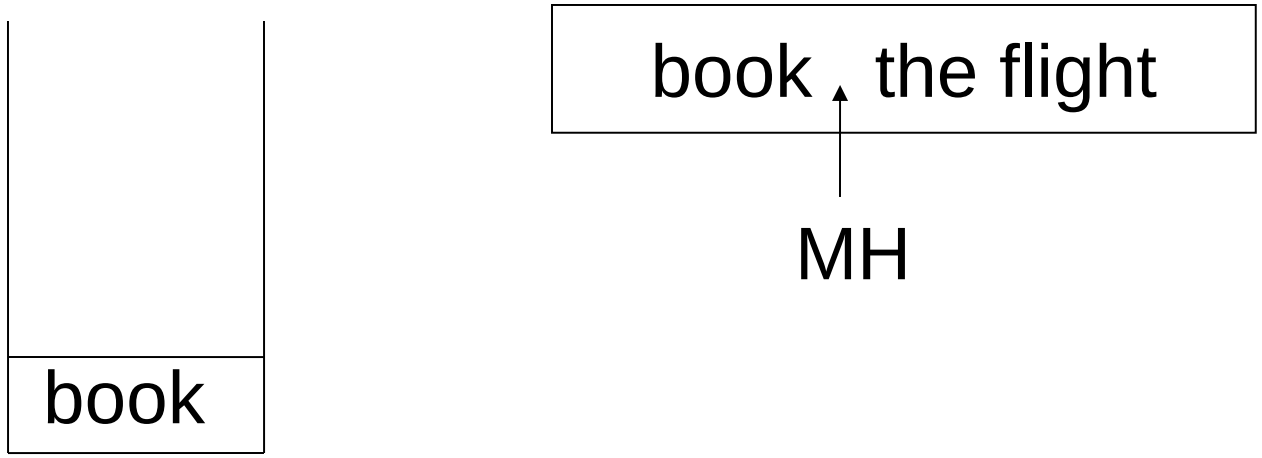
Trace of Bottom-Up Parsing

T_0



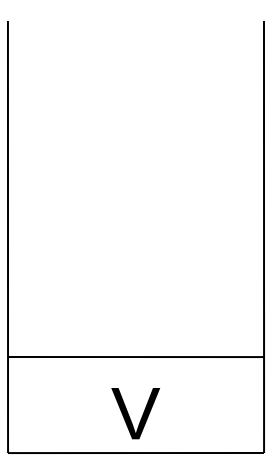
Trace of Bottom-Up Parsing

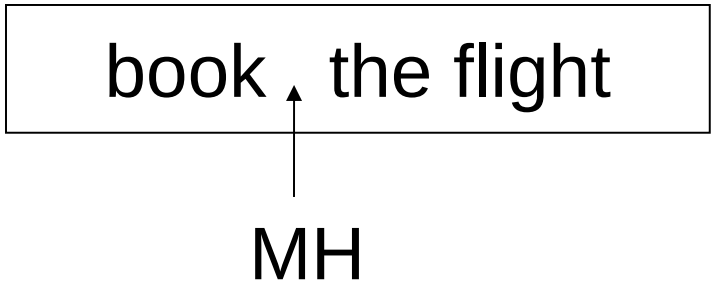
Push ‘book’; advance input pointer



Trace of Bottom-Up Parsing

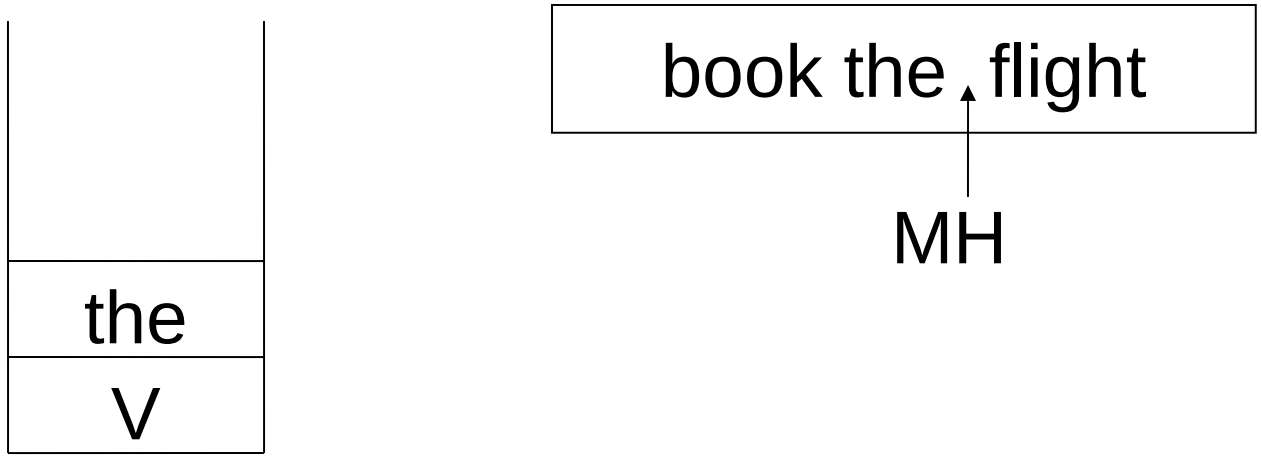
Reduce ‘book’





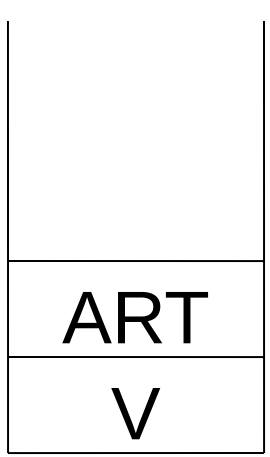
Trace of Bottom-Up Parsing

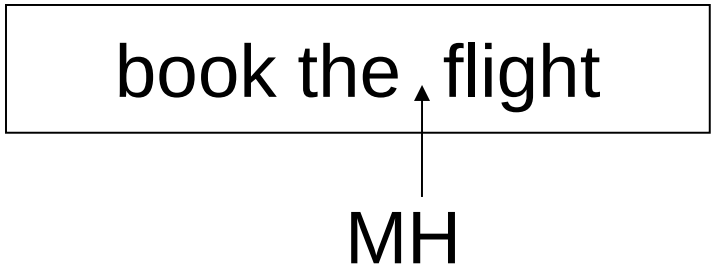
Push ‘the’; advance input pointer



Trace of Bottom-Up Parsing

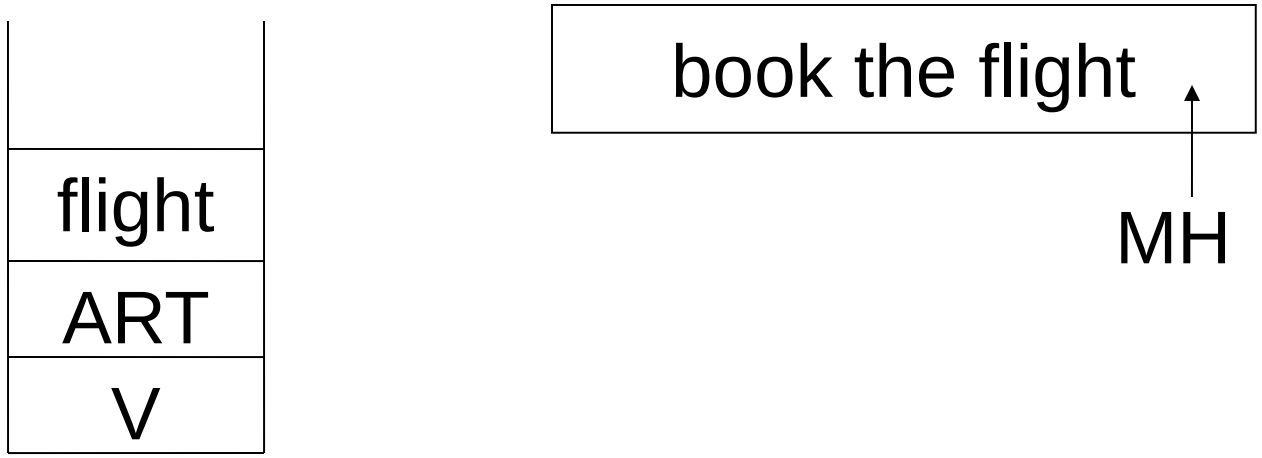
Reduce ‘the’





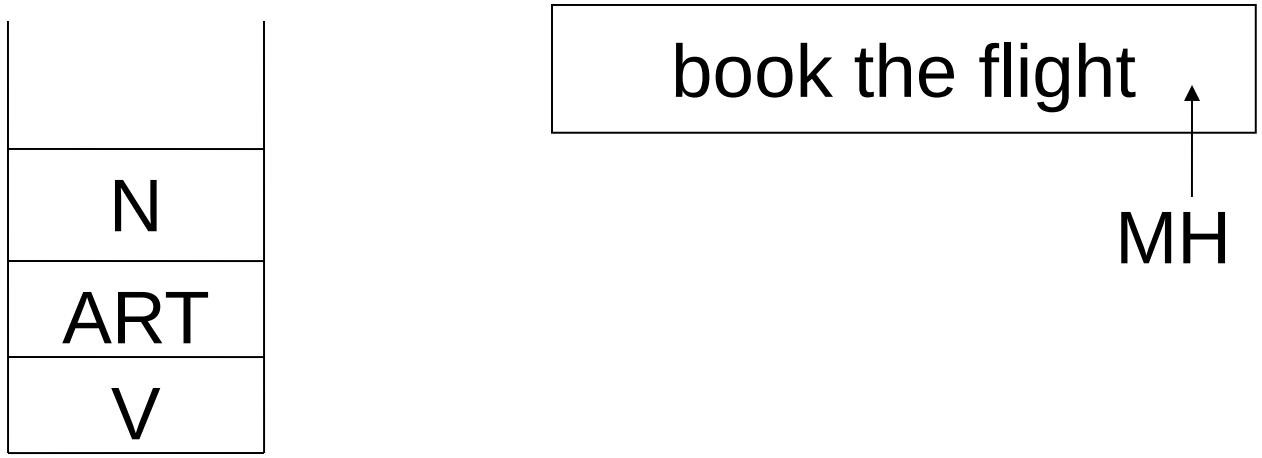
Trace of Bottom-Up Parsing

Push ‘flight’; advance pointer



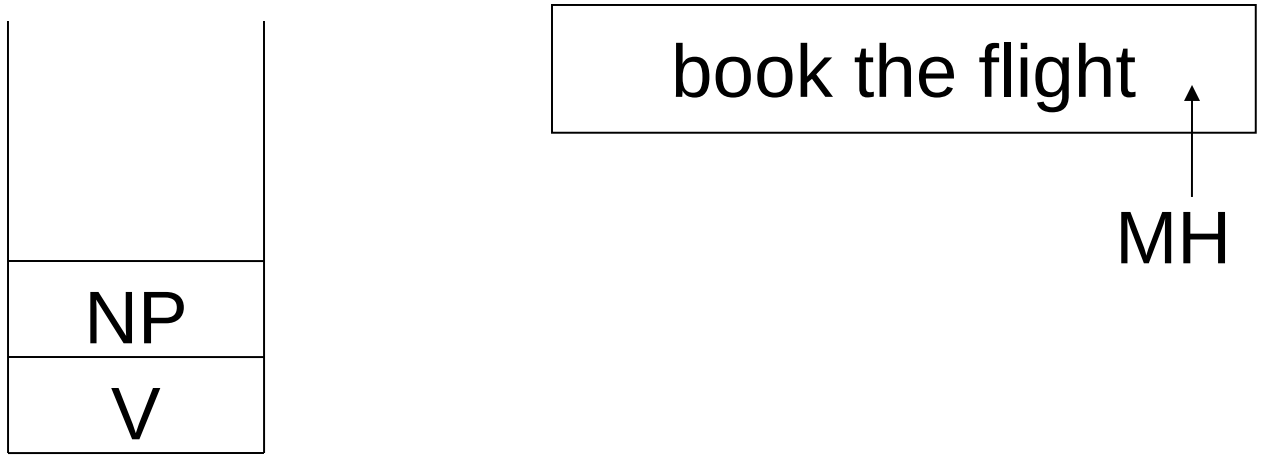
Trace of Bottom-Up Parsing

Reduce ‘flight’



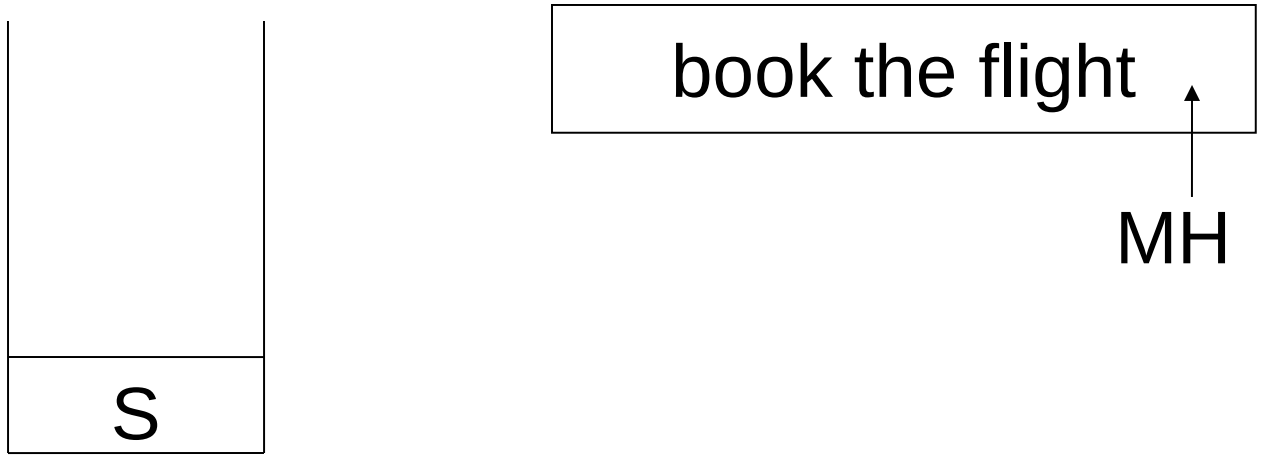
Trace of Bottom-Up Parsing

Reduce ‘ART N’ by ‘NP’

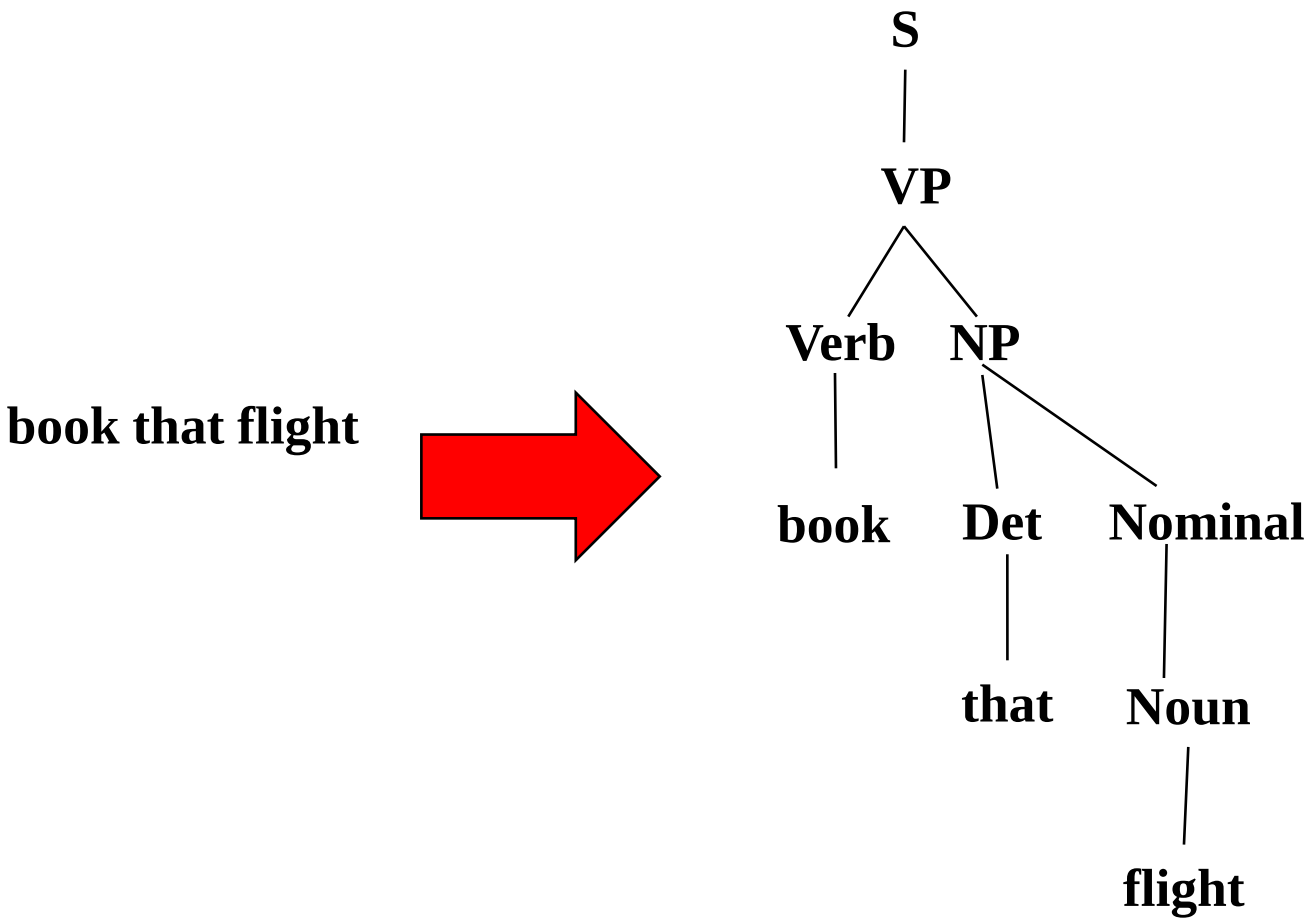


Trace of Bottom-Up Parsing

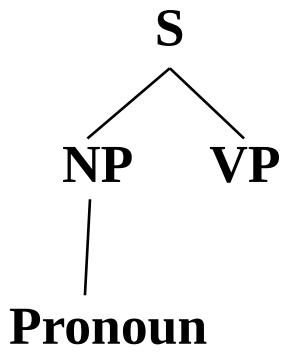
Reduce ‘V NP’ by ‘S’; *termination* by *S* on stack and input exhausted.



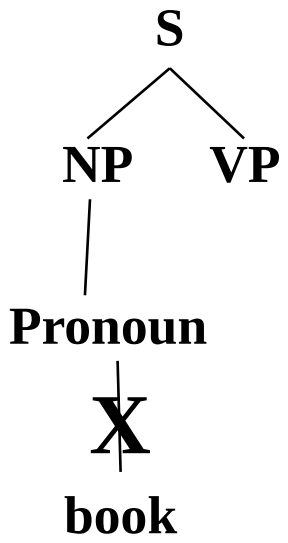
Parsing Example



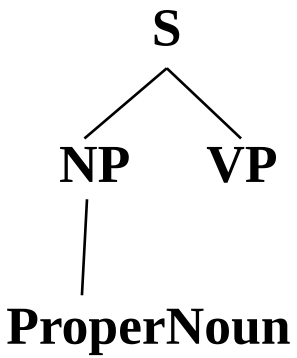
Top Down Parsing



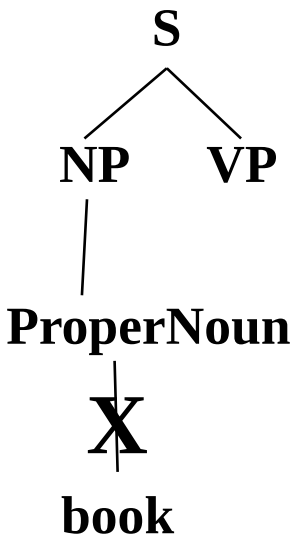
Top Down Parsing



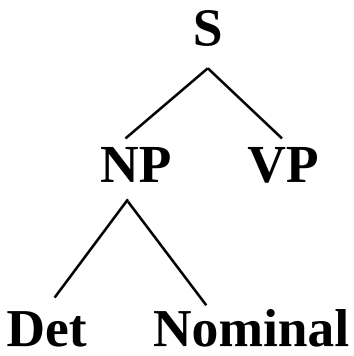
Top Down Parsing



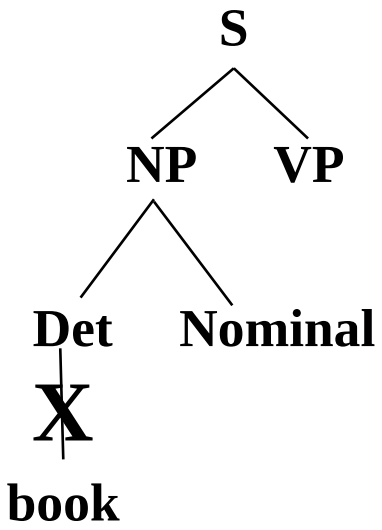
Top Down Parsing



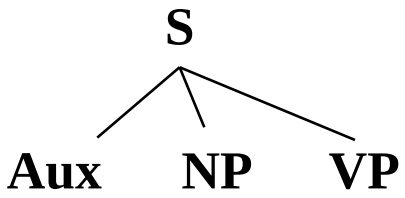
Top Down Parsing



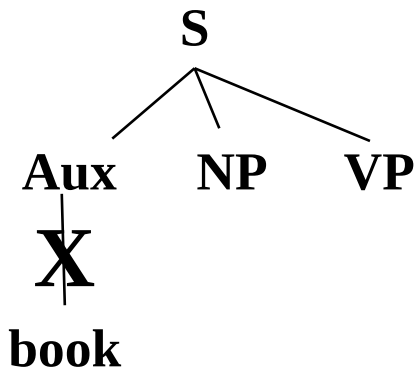
Top Down Parsing



Top Down Parsing

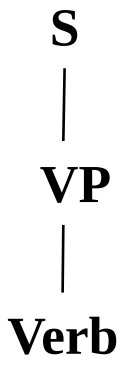


Top Down Parsing



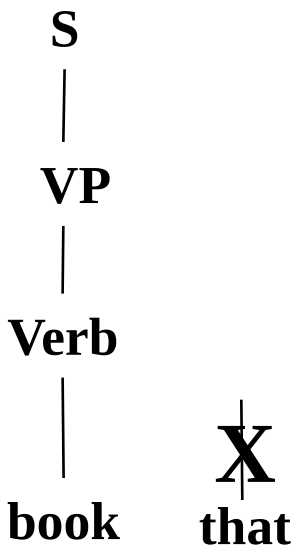
Top Down Parsing

Top Down Parsing

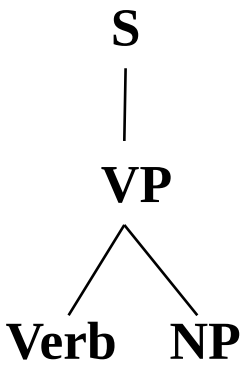


Top Down Parsing

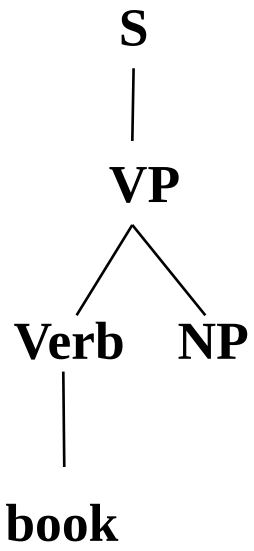
Top Down Parsing



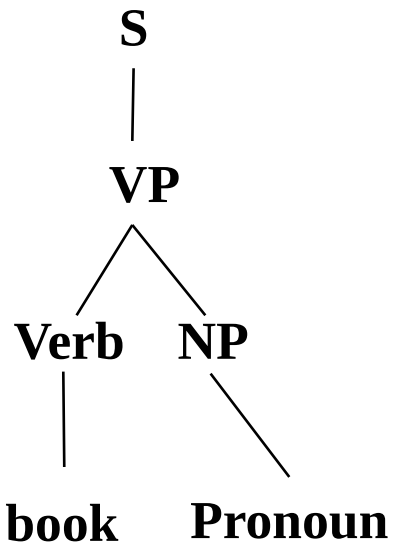
Top Down Parsing



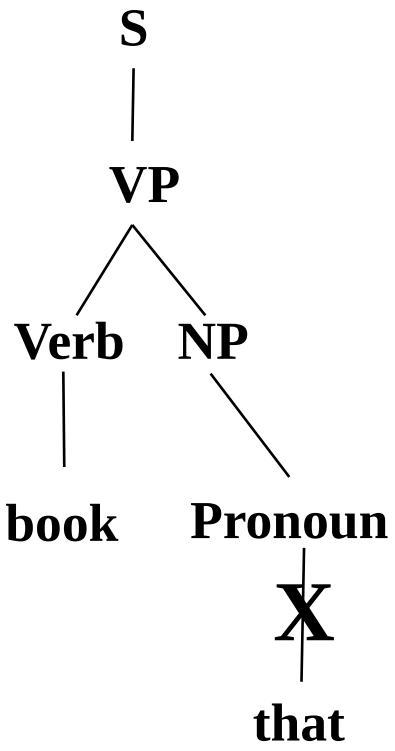
Top Down Parsing



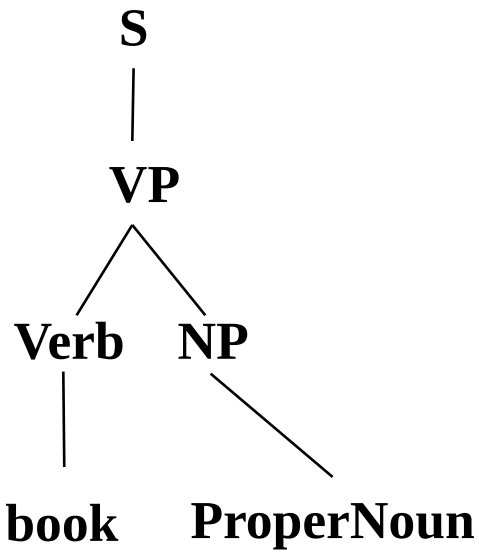
Top Down Parsing



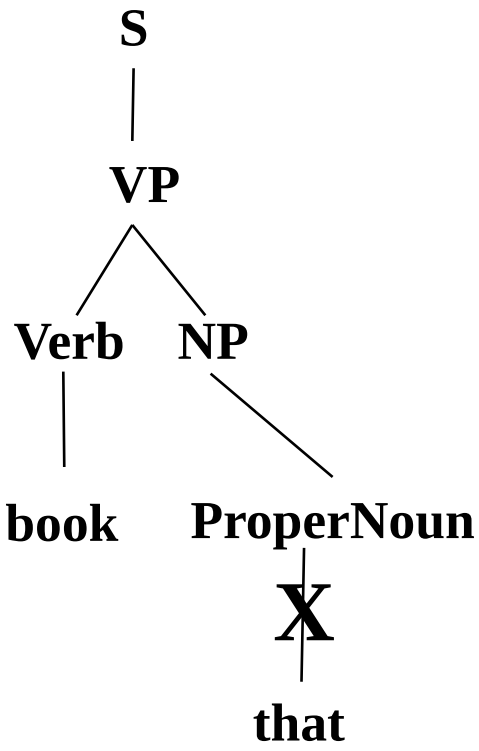
Top Down Parsing



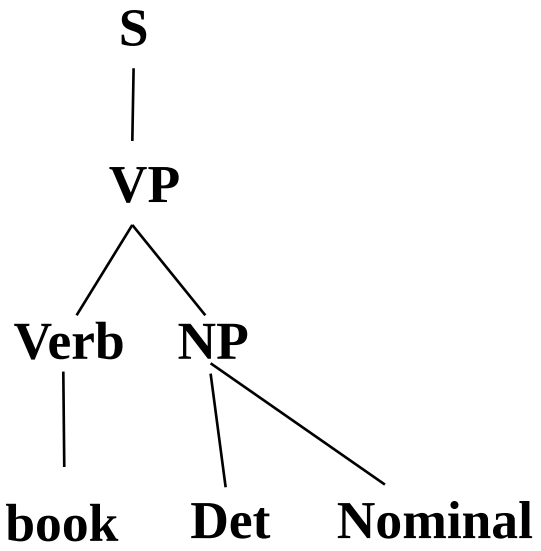
Top Down Parsing



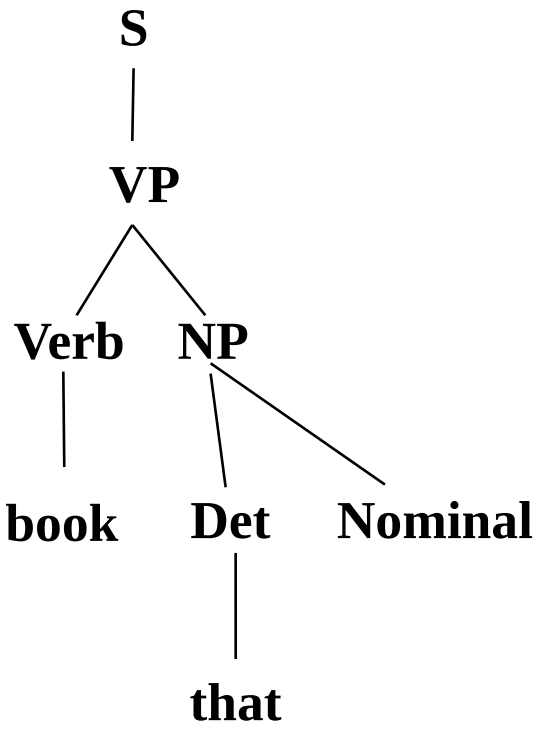
Top Down Parsing



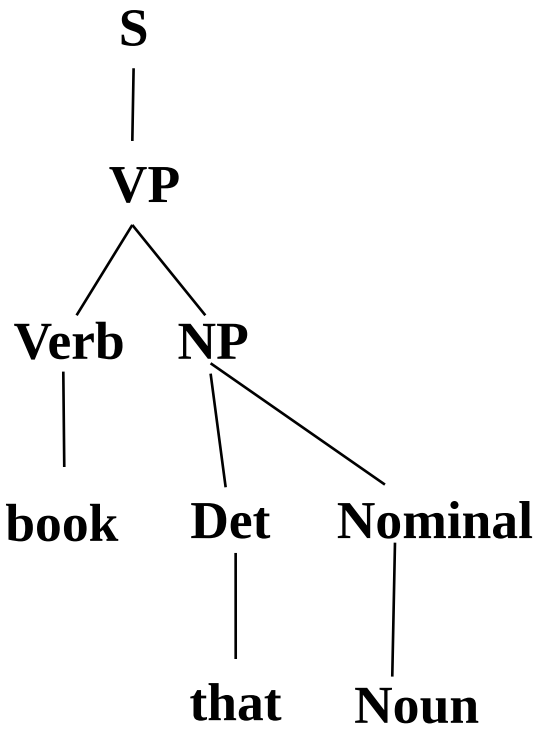
Top Down Parsing



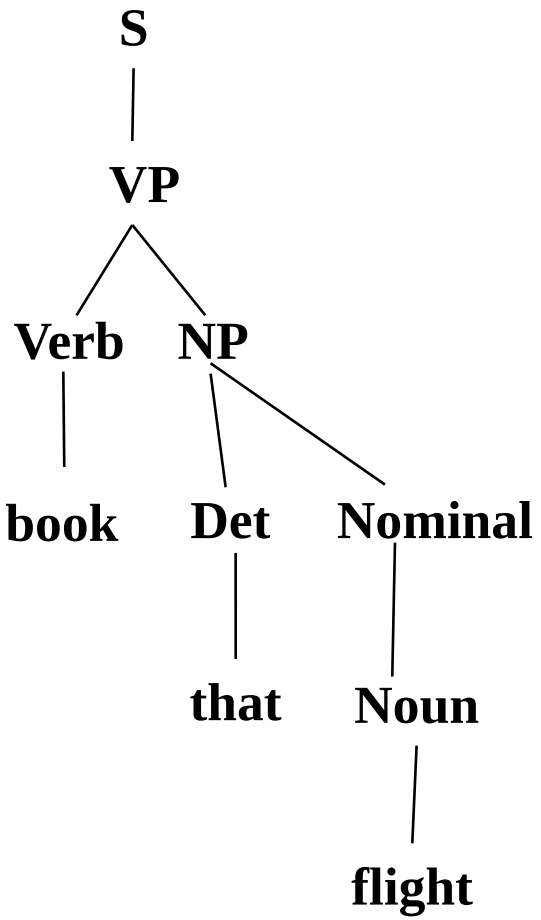
Top Down Parsing



Top Down Parsing



Top Down Parsing

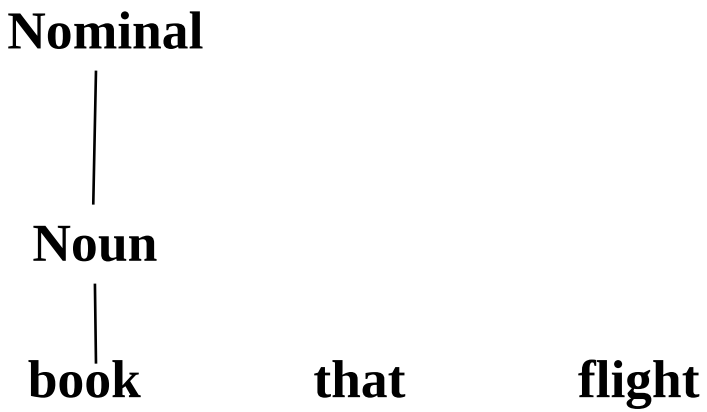


Bottom Up Parsing

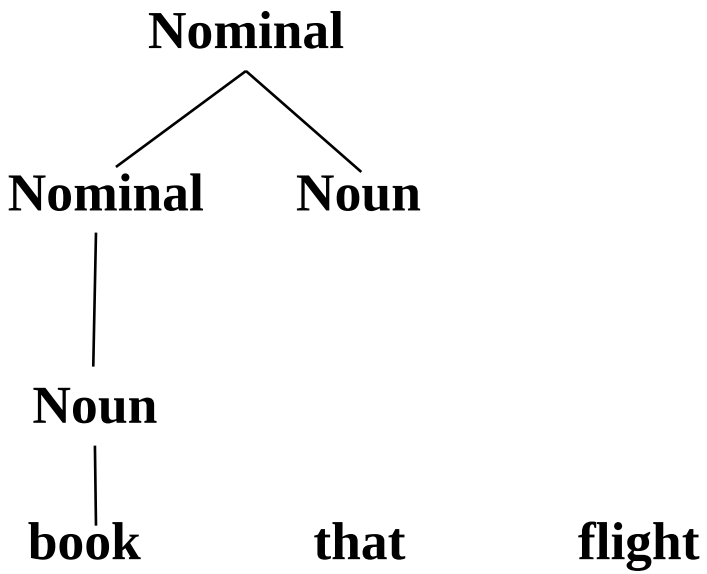
book **that** **flight**

Bottom Up Parsing

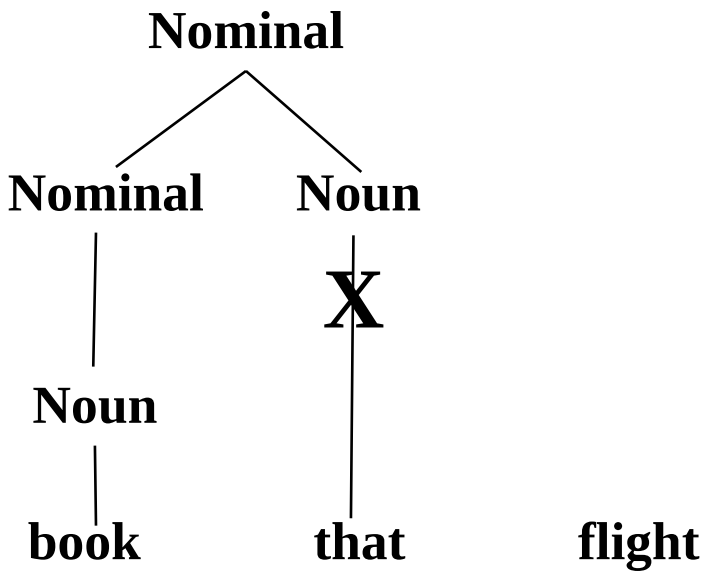
Bottom Up Parsing



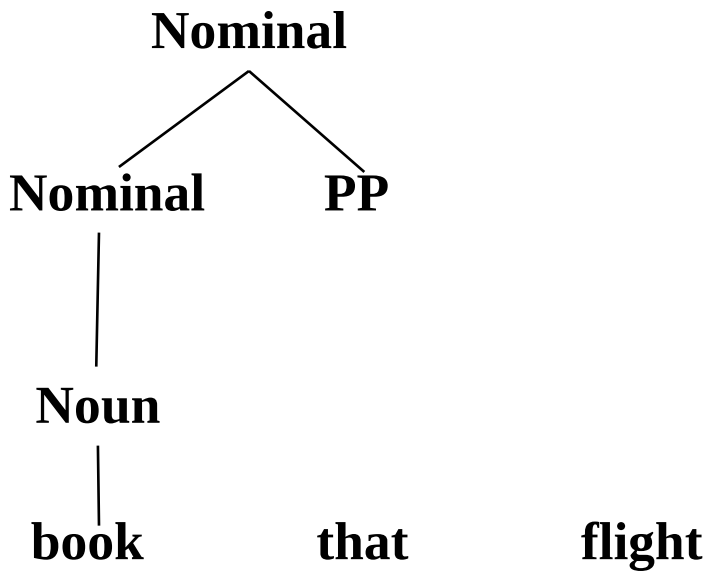
Bottom Up Parsing



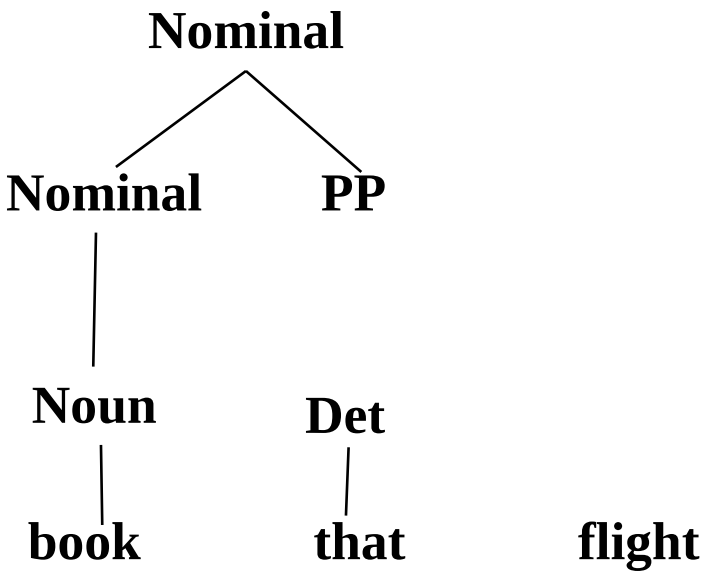
Bottom Up Parsing



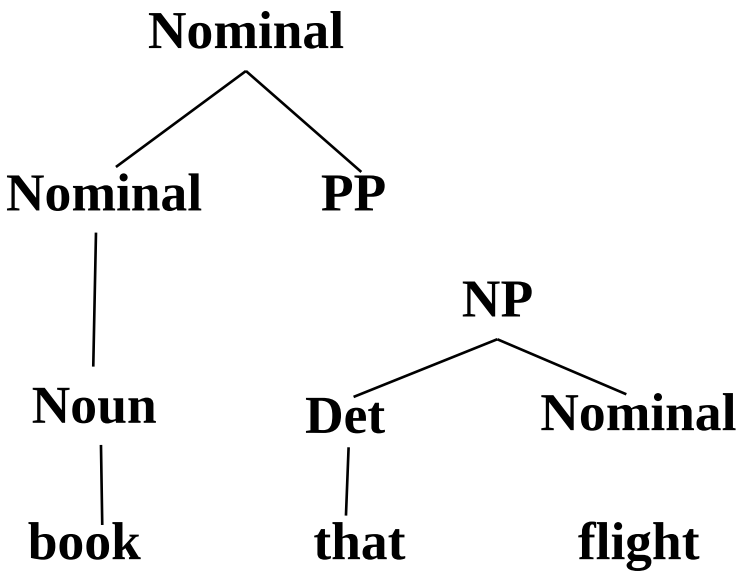
Bottom Up Parsing



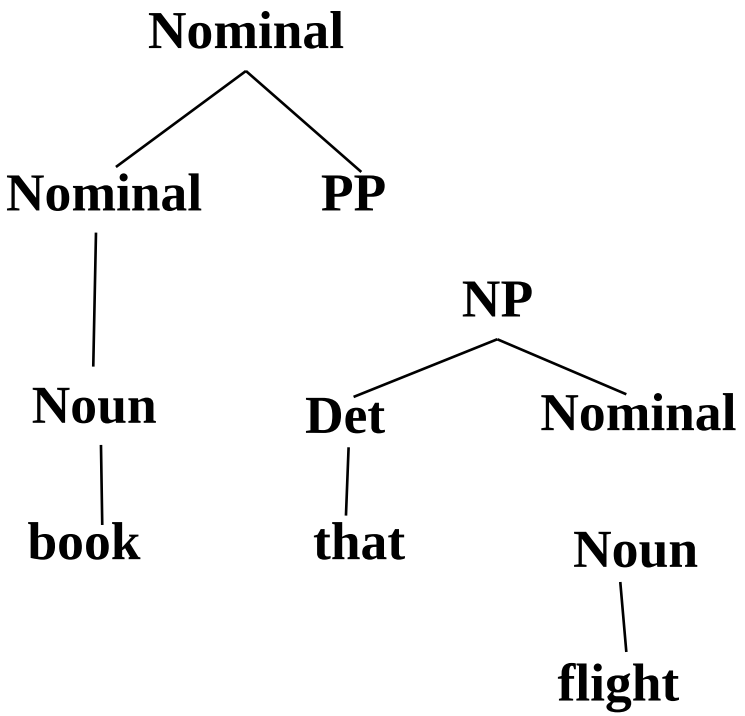
Bottom Up Parsing



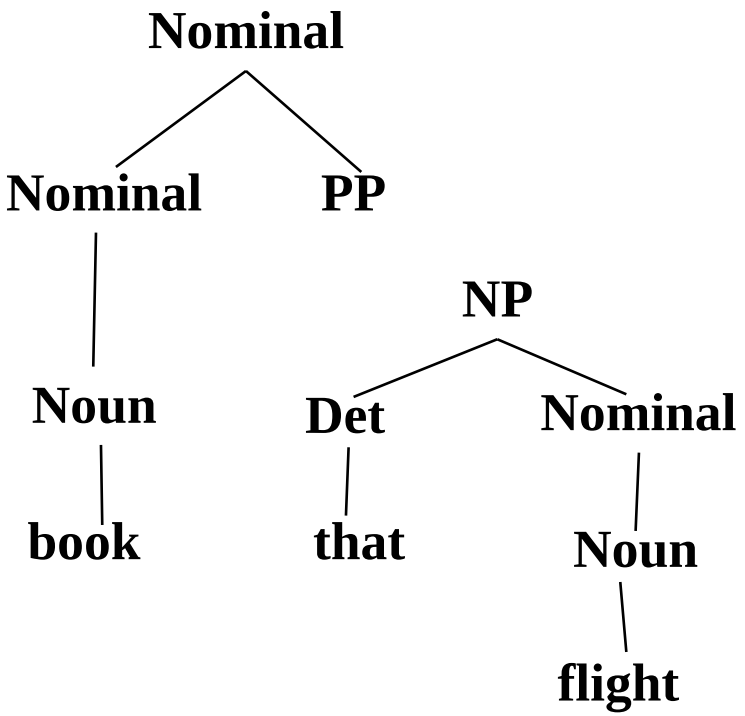
Bottom Up Parsing



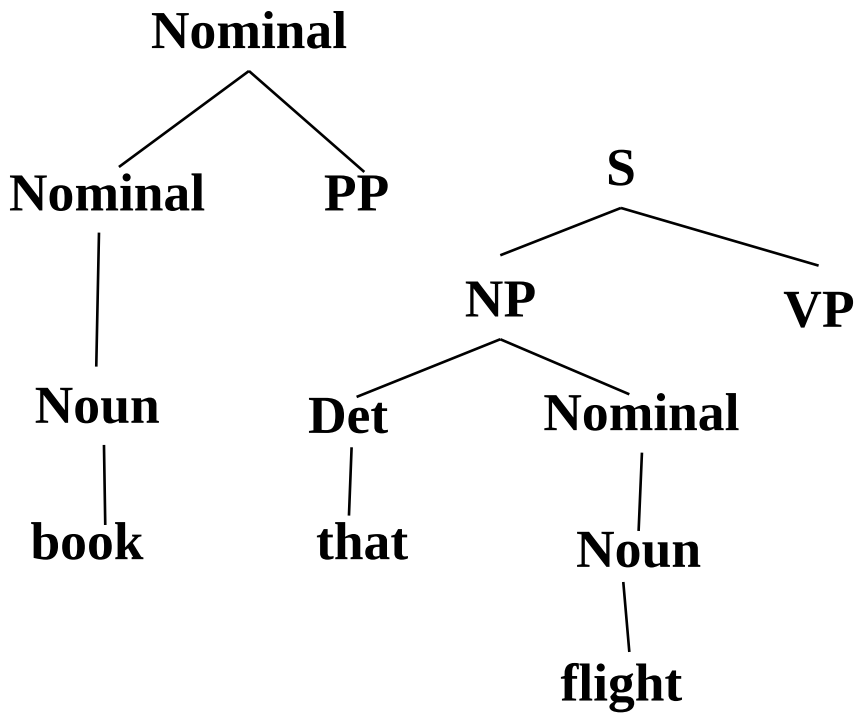
Bottom Up Parsing



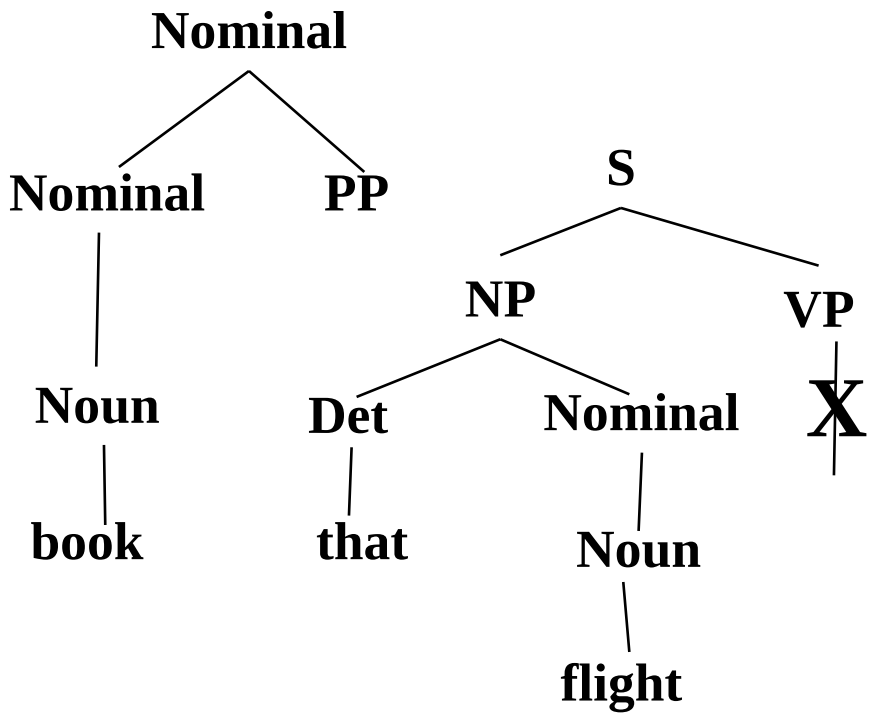
Bottom Up Parsing



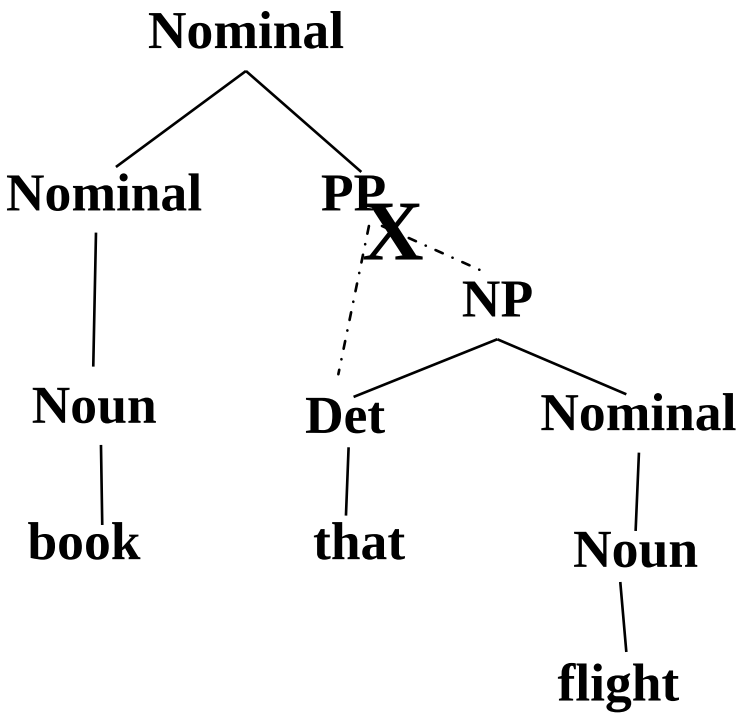
Bottom Up Parsing



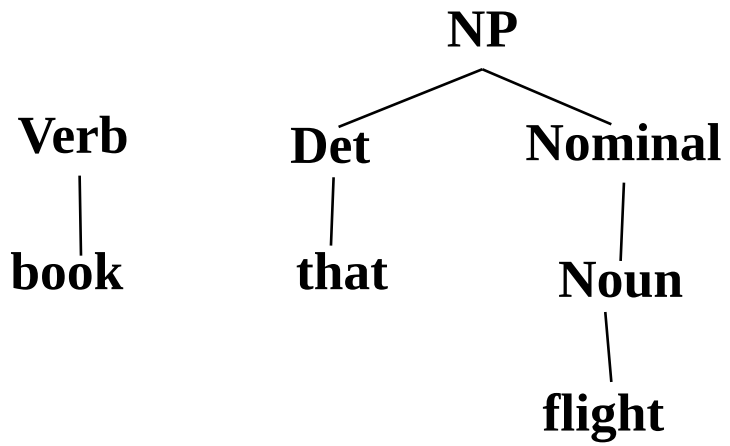
Bottom Up Parsing



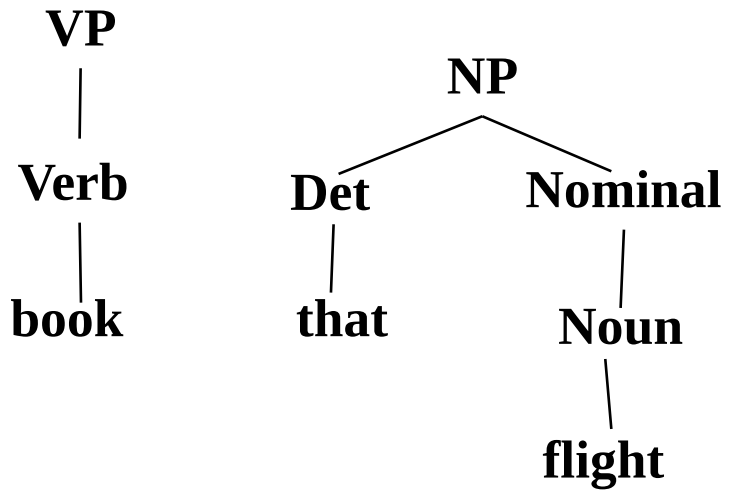
Bottom Up Parsing



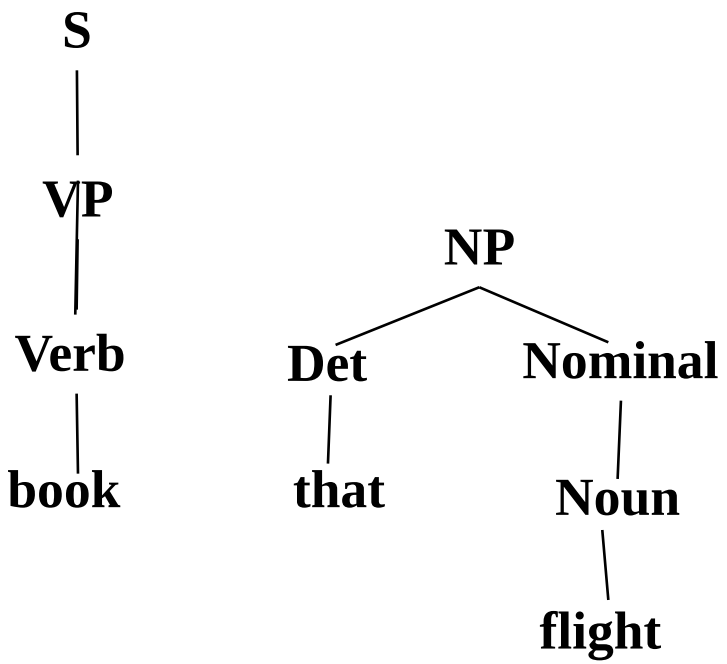
Bottom Up Parsing



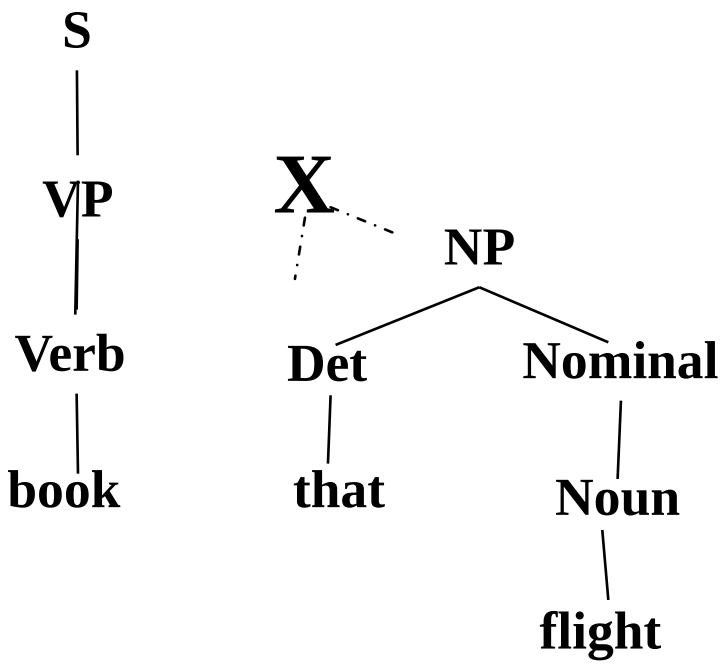
Bottom Up Parsing



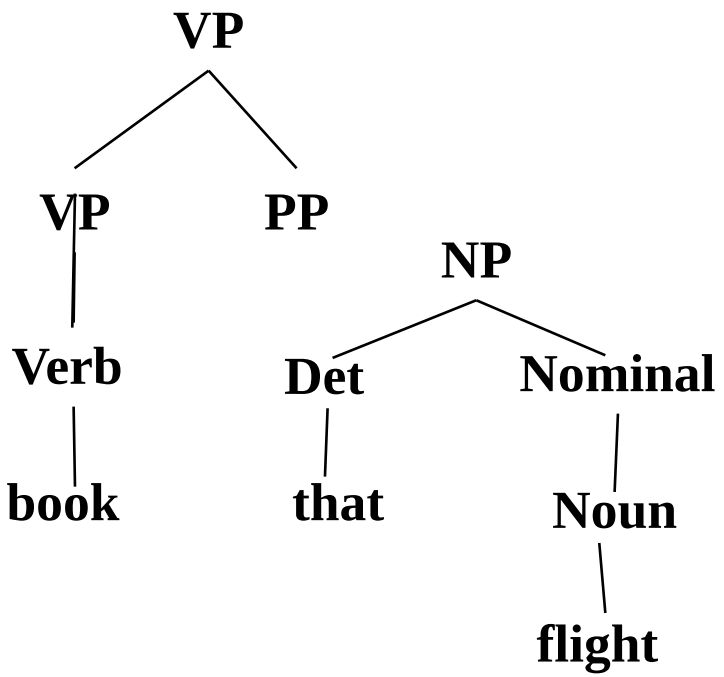
Bottom Up Parsing



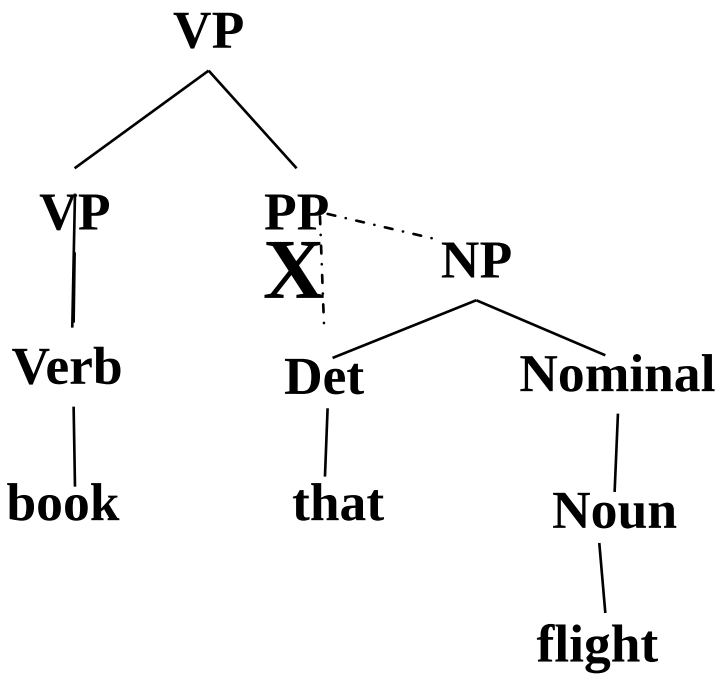
Bottom Up Parsing



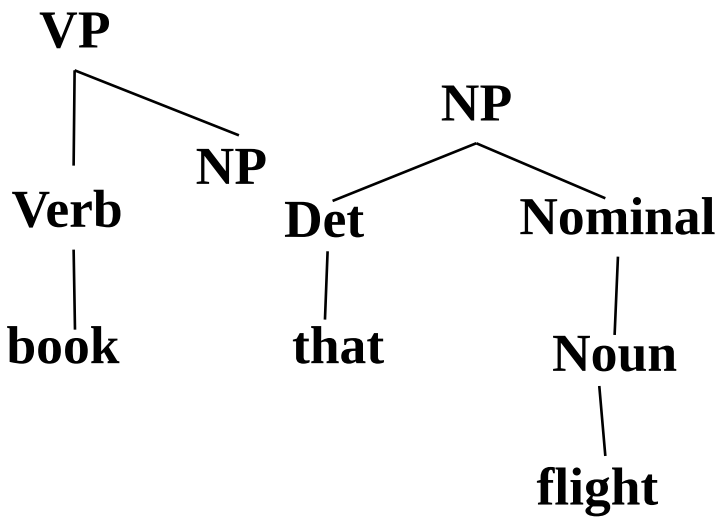
Bottom Up Parsing



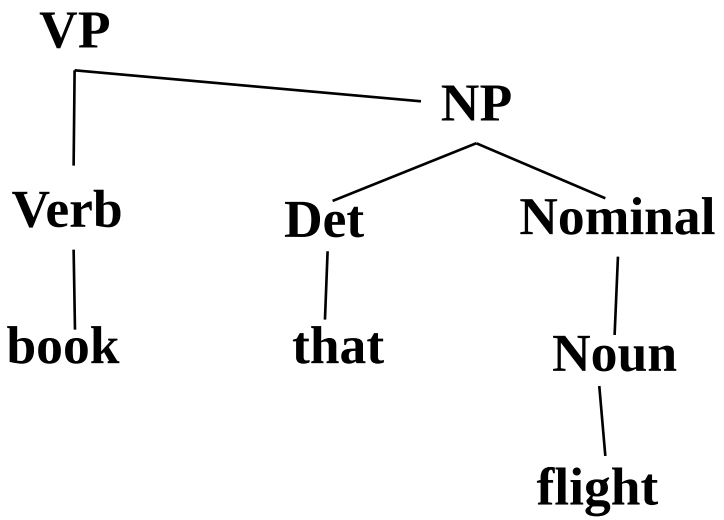
Bottom Up Parsing



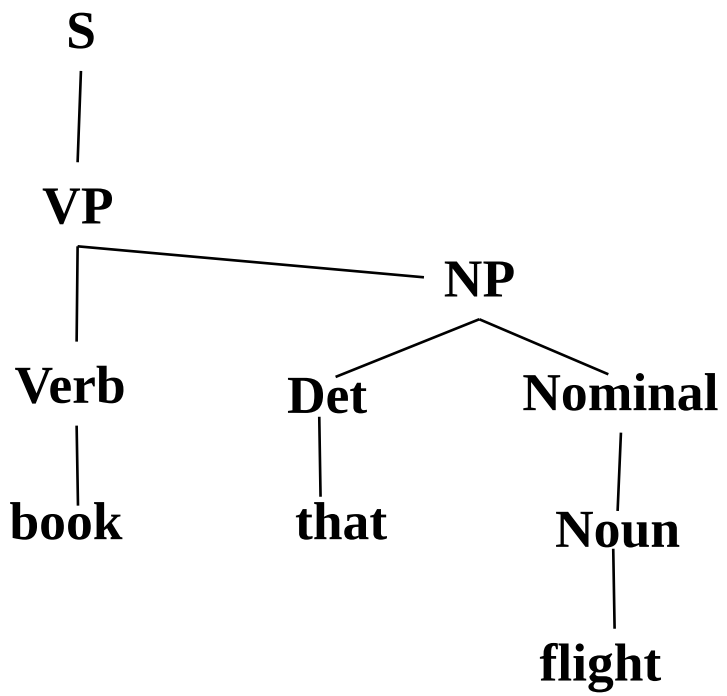
Bottom Up Parsing



Bottom Up Parsing



Bottom Up Parsing



Top Down vs. Bottom Up

- Top down 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.

Efficiency Issues

To reuse the work already done for constituent subtrees.

Ex of inefficiency: Consider the sentence *the train from Chennai to Vizag via Nellore*

Grammar:

S ⟶ NP

NP ⟶ NP PP | ART N| NNP

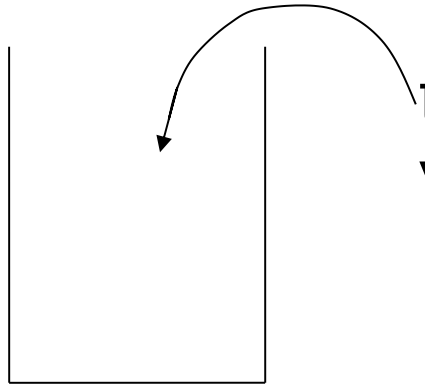
ART ⟶ the

N ⟶ train

NNP ⟶ Chennai | Vizag | Nellore

PP ⟶ P NP

Possible False Steps



the train from Chennai to Vizag via Nellore

push “the”; reduce to “ART”;

push “train”; reduce to “N”;

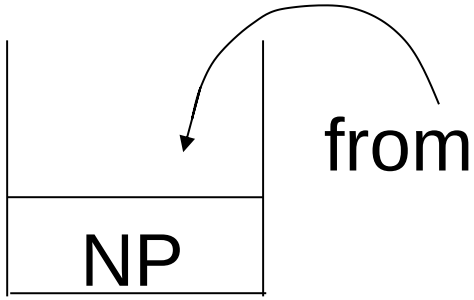
reduce “ART N” to “NP”;

reduce “NP” to “S”.

A

Possible False Steps

Perform A, and then



from Chennai to Vizag via Nellore

push “from”; reduce to “P”;

push “Chennai”; reduce to “NNP”;

reduce to “NP”; reduce “P NP” to “PP”;

Reduce “NP PP” to “NP”;

reduce “NP” to “S”.

B

Possible False Steps

Similarly for

"……. to Vizag via Nellore"

and

"…….. via Nellore"

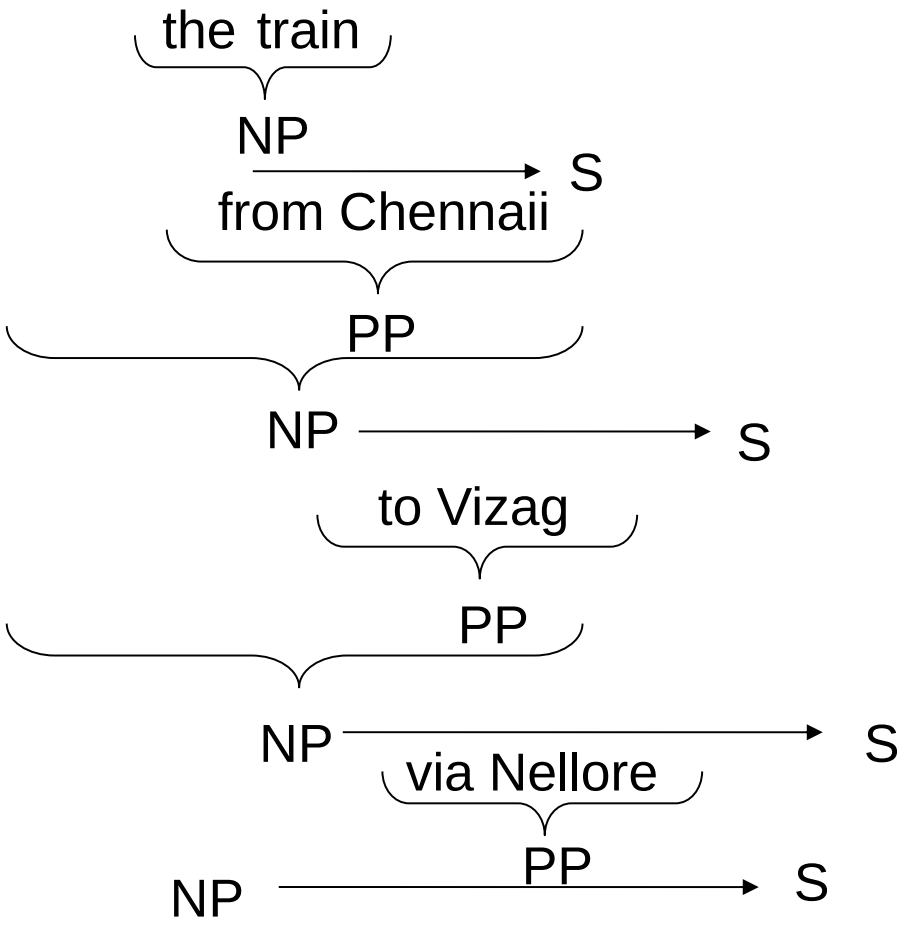
Shift reduce conflicts occur for

S $\longrightarrow$ NP

NP $\longrightarrow$NP PP

Should "NP" be reduced to "S" or should one search for "PP" and then get bigger "NP"?

Reduplication of Work



of Repetitions for Subtree Computation

the train	4 times
from Chennai	3 times
the train from Chennai	3 times
to Vizag	2 times
the train from Chennai to Vizag	2 times
via Nellore	1 time
the train from Chennai to Vizag via Nellore	1 time

Can the “subtrees already computed” be reused?

Chart Parsing : Earley Algorithm (Dynamic Programming based)

Sentence: *book the flight*

Grammar:

S → NP VP | VP

NP → ART N | NNP

VP → V | V NP

ART → a | an | the

N → book | flight

V → book

Definitions

CHART is the data structure that stores the record of *matched constituents* and *expected constituents* through the use of *dotted rules.*

A *dotted rule* is of the form

where *B* is the *matched constituent* and *C* is the *expected constituent.*

Definitions

PREDICTOR is the procedure that records by *transitive closure* the set of dotted rules for a given state of the input processing.

SCANNER is the procedure that consumes the next input token.

COMPLETER is the procedure that

- takes a dotted rule for which the dot is at the rightmost end and
- advances the dots for the rules for which a matched constituent was awaiting completion.

END