Basic Parsing with Context Free Grammars

Lecture #5

SNU 4th Industrial Revolution Academy: Artificial Intelligence Agent

Analyzing Linguistic Units

- Morphological parsing:
  - analyze words into morphemes and affixes
  - rule-based, FSAs, FSTs
- Phonological parsing:
  - analyze sounds into words and phrases
- POS Tagging
- Syntactic parsing:
  - identify component parts and how related
  - to see if a sentence is grammatical
  - to assign an abstract representation of meaning

Syntactic Parsing

- Declarative formalisms like CFGs define the legal strings of a language but don't specify how to recognize or assign structure to them
- Parsing algorithms specify how to recognize the strings of a language and assign each string one or more syntactic structures
- Parse trees useful for grammar checking, semantic analysis, MT, QA, information extraction, speech recognition…and almost every task in NLP

Parsing is a Form of Search

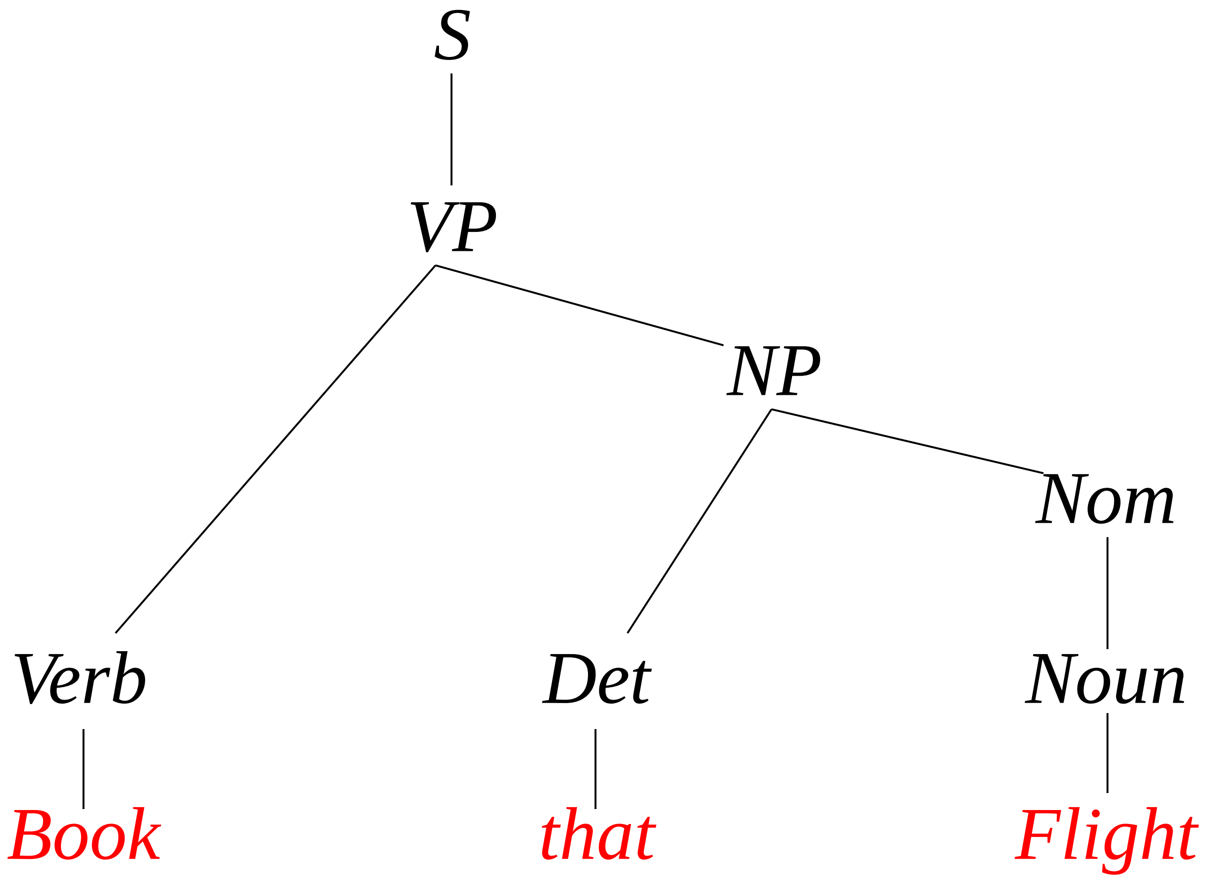
- Searching FSAs
  - Finding the right path through the automaton
  - Search space defined by structure of FSA
- Searching CFGs
  - Finding the right parse tree among all possible parse trees
  - Search space defined by the grammar
- Constraints provided by the input sentence and the automaton or grammar

CFG for Fragment of English

S → NP VP	VP → V
S → Aux NP VP	Det → that | this | a
S → VP	N → book | flight | meal | money
NP → Det Nom	V → book | include | prefer
NP → PropN	Aux → does
Nom → N Nom	Prep → from | to | on
Nom → N	PropN → Houston | TWA
Nom → Nom PP	
VP → V NP	

TopD BotUp E.g. LC's

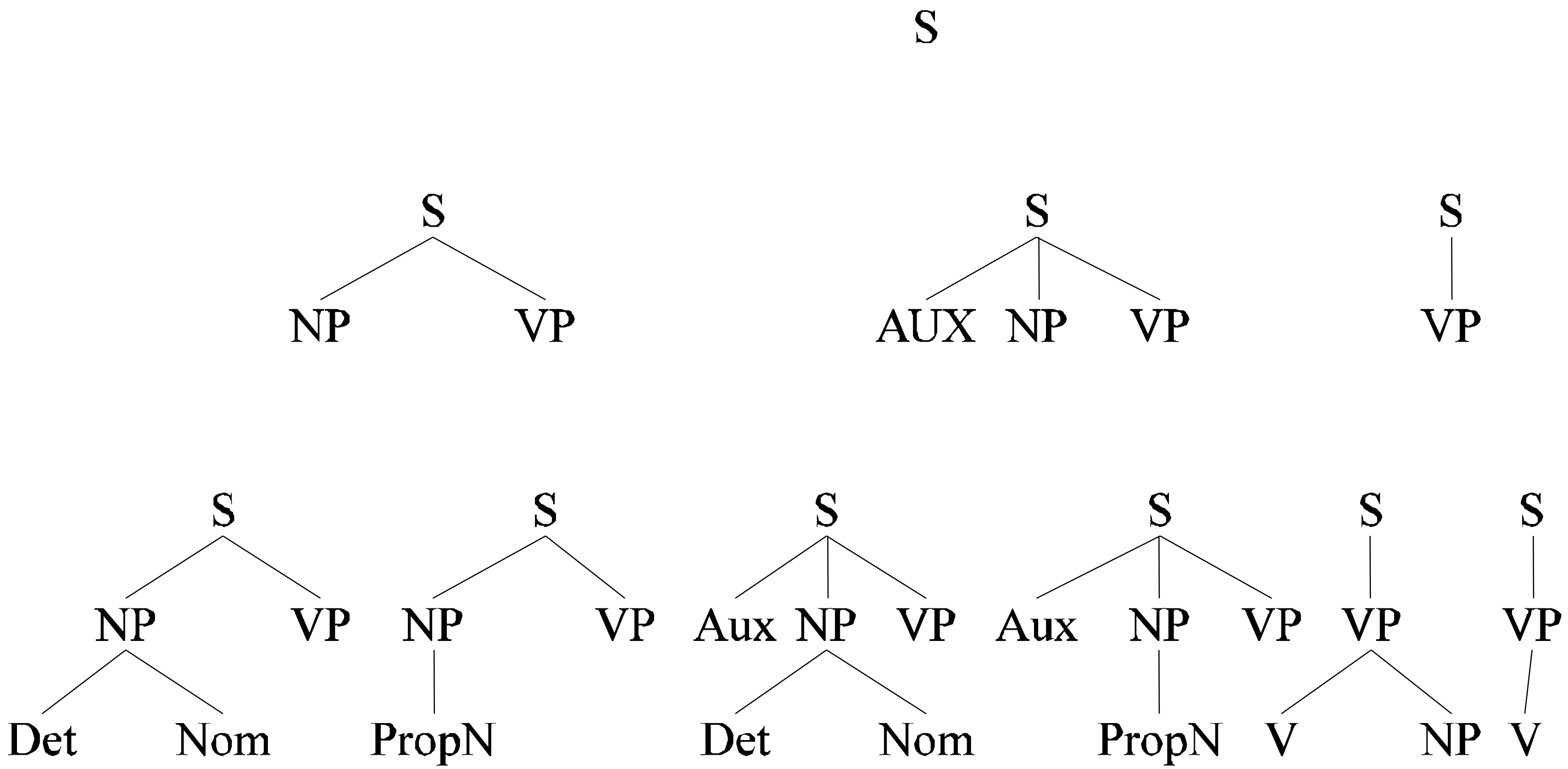
Parse Tree for "Book that flight" for Prior CFG



Top-Down Parser

- Builds from the root S node to the leaves
- Find a rule to apply by matching the left hand side of a rule
- Build a tree by replacing LHS with the right hand side
- Assuming we build all trees in parallel:
  - Find all trees with root S (or all rules w/lhs S)
  - Next expand all constituents in these trees/rules
  - Continue until leaves are pos
  - Candidate trees failing to match pos of input string are rejected (e.g. Book that flight can only match subtree 5)

Top Down Space

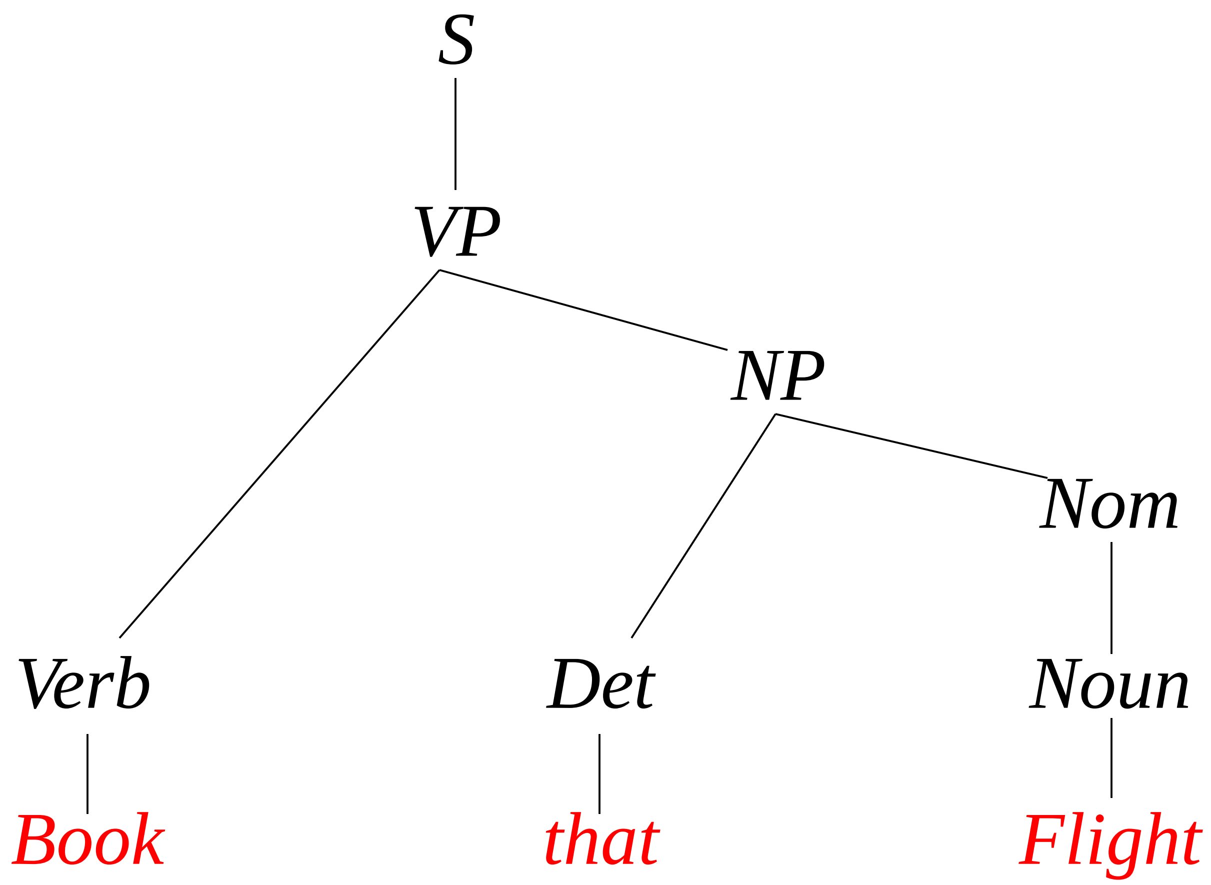


CFG for Fragment of English

S → NP VP	VP → V
S → Aux NP VP	Det → that (5) | this | a
S → VP (1)	N → book | flight (7) | meal | money
NP → Det Nom (4)	V → book (3) | include | prefer
NP →PropN	Aux → does
Nom → N Nom	Prep →from | to | on
Nom → N (6)	PropN → Houston | TWA
Nom → Nom PP	
VP → V NP (2)	

TopD BotUp E.g. LC's

Parse Tree for "Book that flight" for Prior CFG



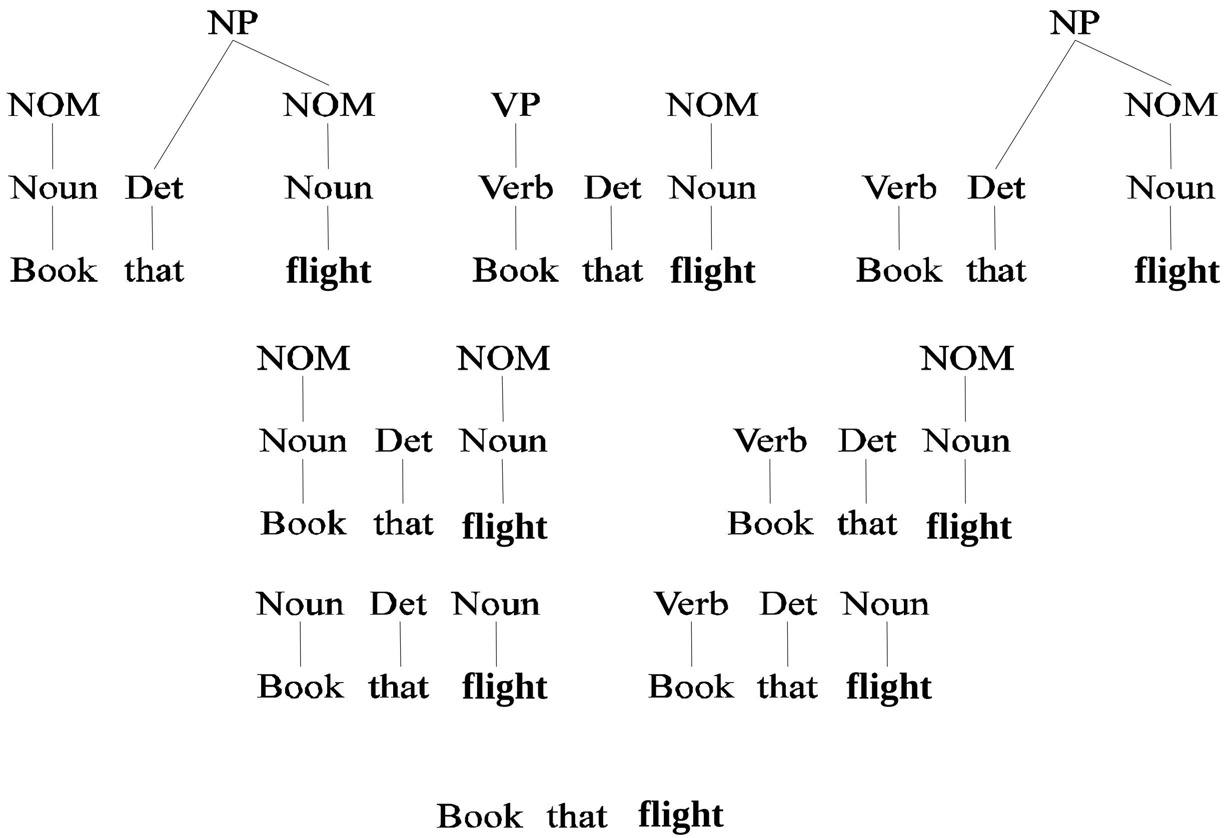
Bottom-Up Parsing

- Parser begins with words of input and builds up trees, applying grammar rules whose right hand side match
  - Book that flight

N	Det	N	V	Det	N
Book	that	flight	Book	that	flight

  - 'Book' ambiguous
  - Parse continues until an S root node reached or no further node expansion possible

Bottom-Up Space

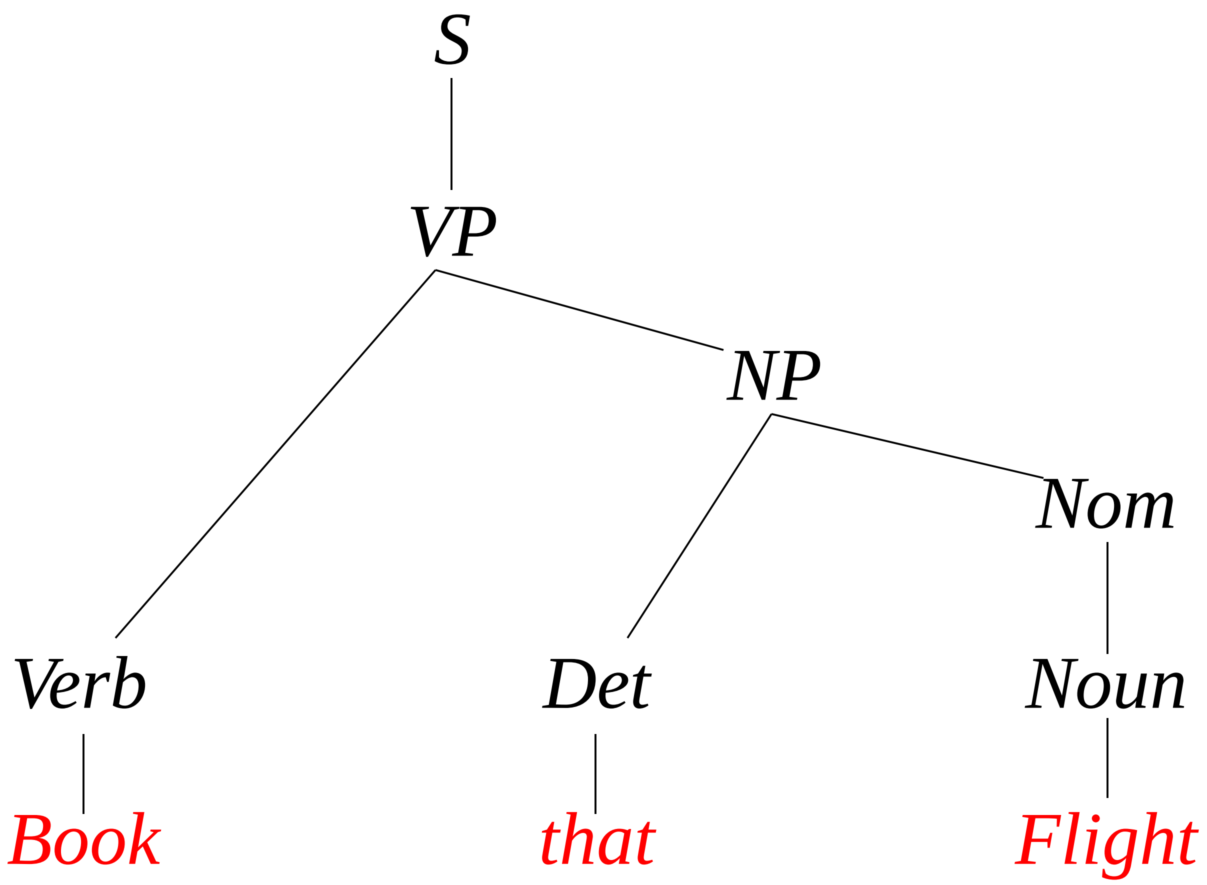


CFG for Fragment of English

S → NP VP	VP → V
S → Aux NP VP	Det → that (2) | this | a
S → VP (7)	N → book | flight (3) | meal | money
NP → Det Nom (5)	V → book (1) | include | prefer
NP →PropN	Aux → does
Nom → N Nom	Prep →from | to | on
Nom → N (4)	PropN → Houston | TWA
Nom → Nom PP	
VP → V NP (6)	

TopD   BotUp   E.g.   LC's

Parse Tree for “Book that flight” for Prior CFG



Control

- Of course, we left out how to keep track of the spaces and how to make choices
  - Which node to try to expand next
  - Which grammar rule to use to expand a node

A Top-Down Parsing Strategy

- **Depth-first search**:
  - Agenda of search states: expand search space incrementally, exploring most recently generated state (tree) each time
  - When you reach a state (tree) inconsistent with input, backtrack to most recent unexplored state (tree)
- Which node to expand?
  - **Leftmost** or rightmost
- Which grammar rule to use?
  - Order in the grammar??

Top-Down, Depth-First, Left-Right Strategy

- Initialize agenda with ‘S’ tree and ptr to first word and make this current search state (cur)
- Loop until successful parse or empty agenda
  - Apply all applicable grammar rules to leftmost unexpanded node of cur
    - If this node is a POS category and matches that of the current input, push this onto agenda
    - O.w. push new trees onto agenda
  - Pop new cur from agenda
- Does this flight include a meal?

Top-Down, Depth-First, Left-to-Right Search

Curr: S

[Does]

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP → PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

Top-Down, Depth-First, Left-to-Right Search

Curr:

S

[Does]

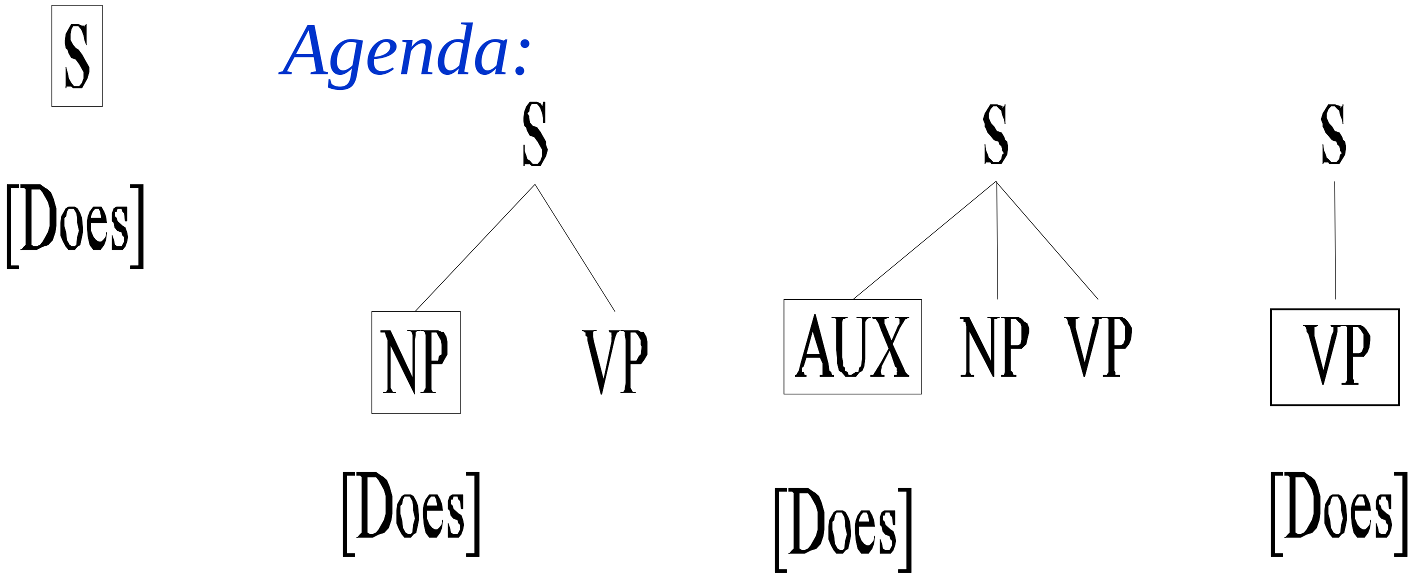
Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP → PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

Top-Down, Depth-First, Left-to-Right Search

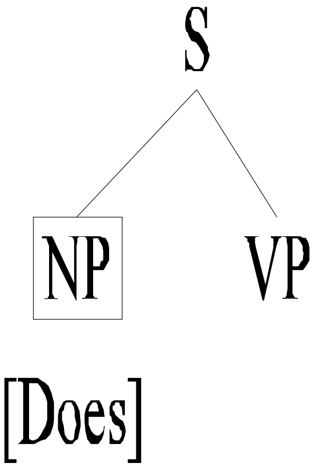
Curr:

Agenda:

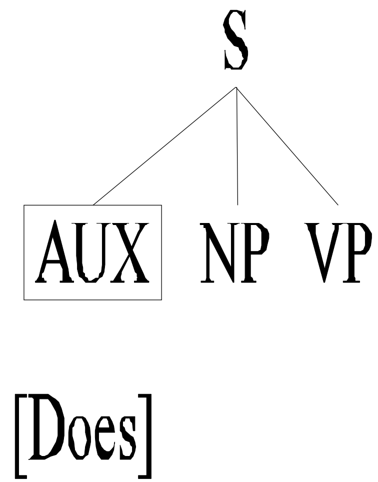


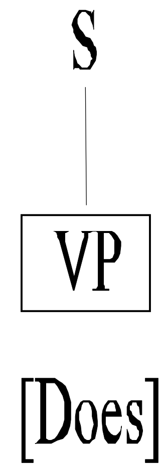
Top-Down, Depth-First, Left-to-Right Search

Curr:



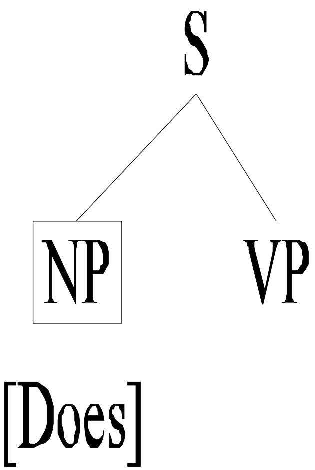
Agenda:





Top-Down, Depth-First, Left-to-Right Search

Curr:

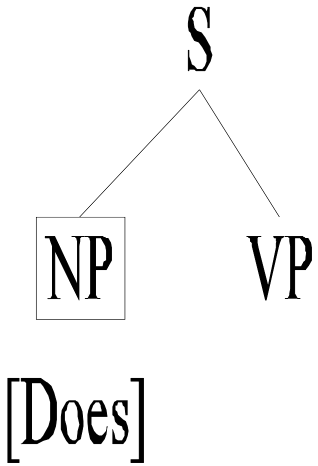


Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP →PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

Top-Down, Depth-First, Left-to-Right Search

Curr:

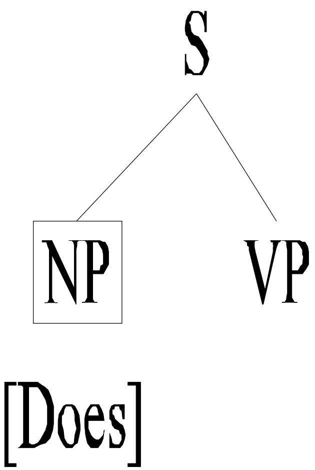


Grammar:

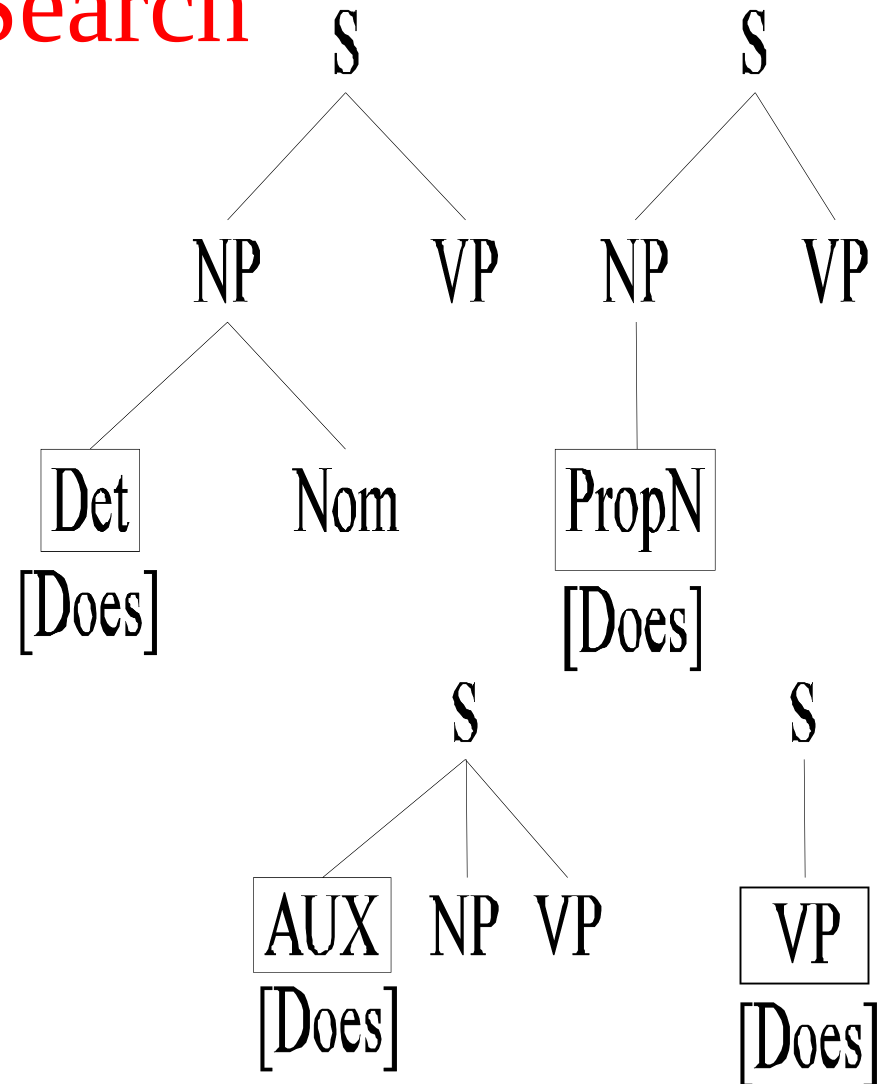
S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP → PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

Top-Down, Depth-First, Left-to-Right Search

Curr:

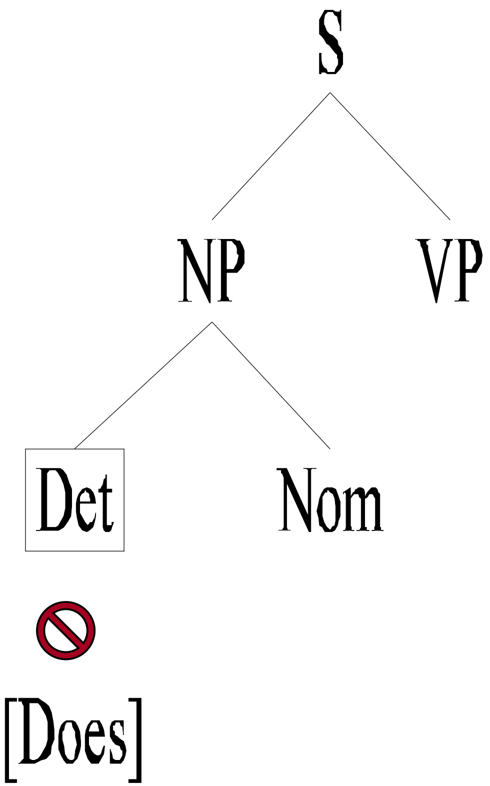


Agenda:

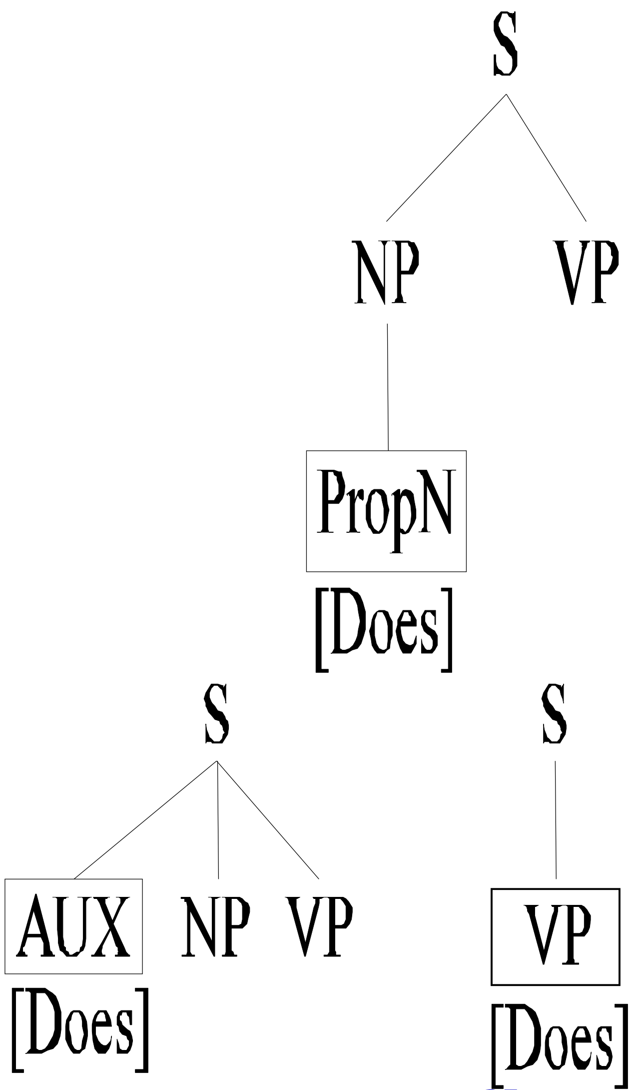


Top-Down, Depth-First, Left-to-Right Search

Curr:

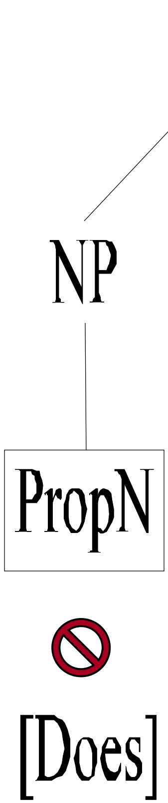


Agenda:

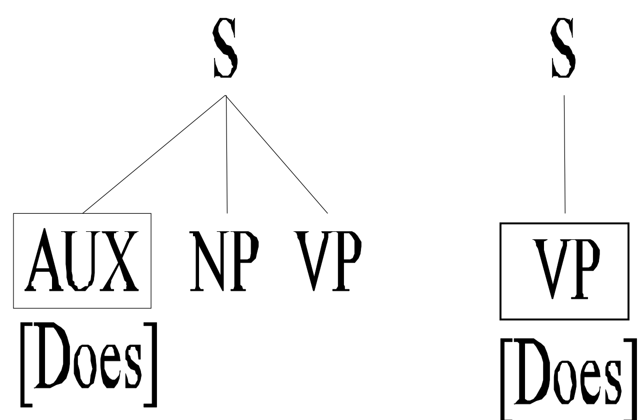


Top-Down, Depth-First, Left-to-Right Search

Curr:



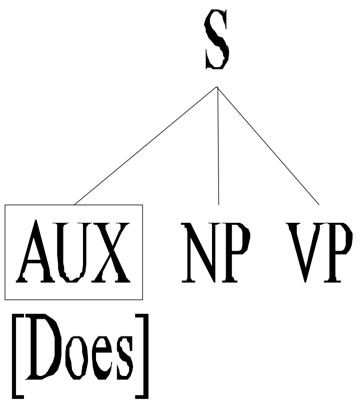
Agenda:

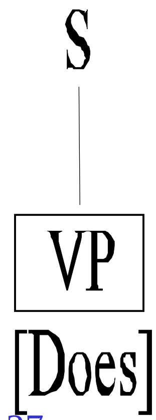


Top-Down, Depth-First, Left-to-Right Search

Curr:

Agenda:

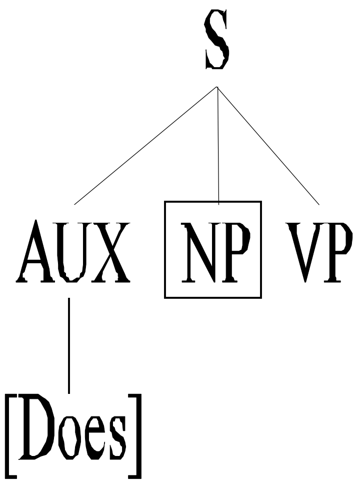




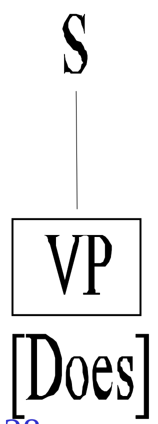
Top-Down, Depth-First, Left-to-Right Search

Curr:

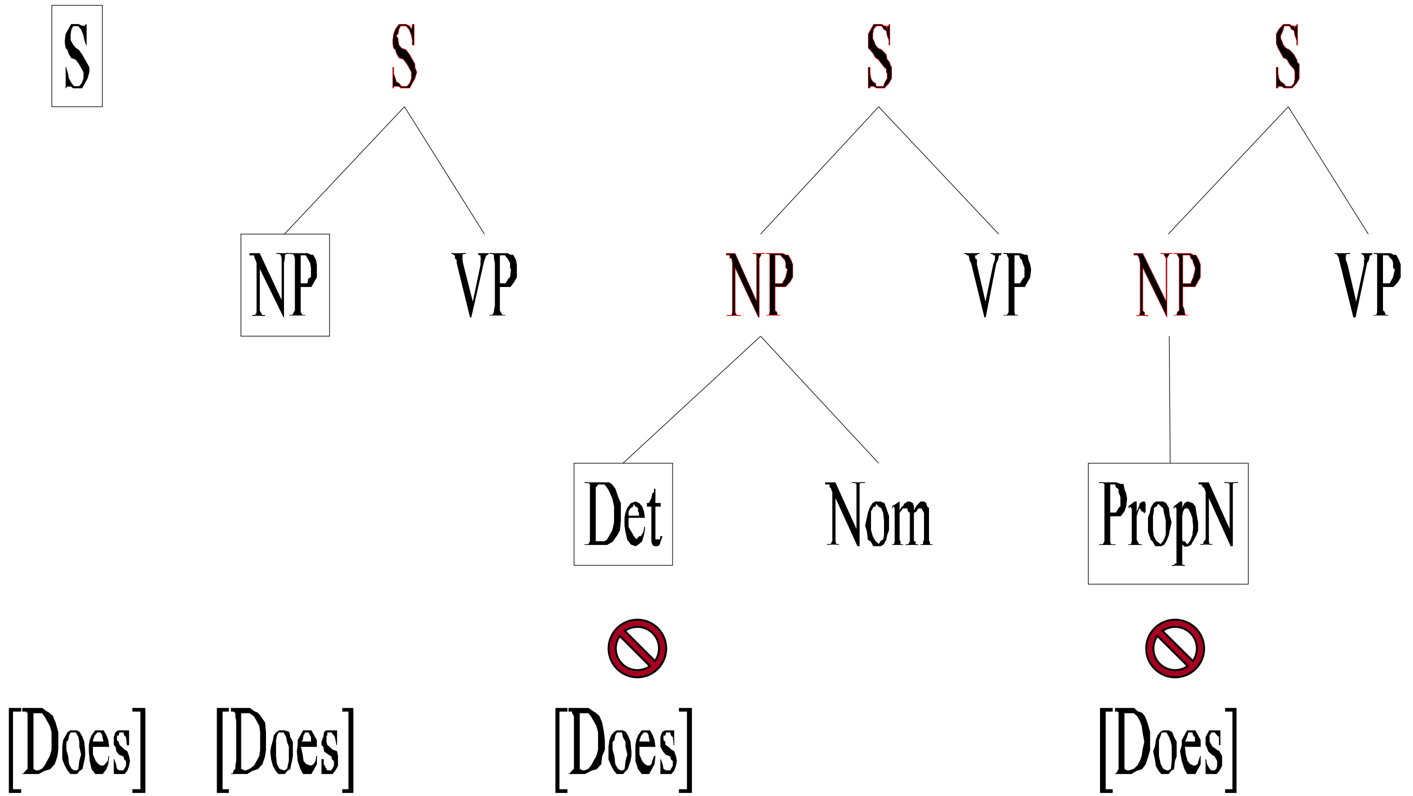
Agenda:



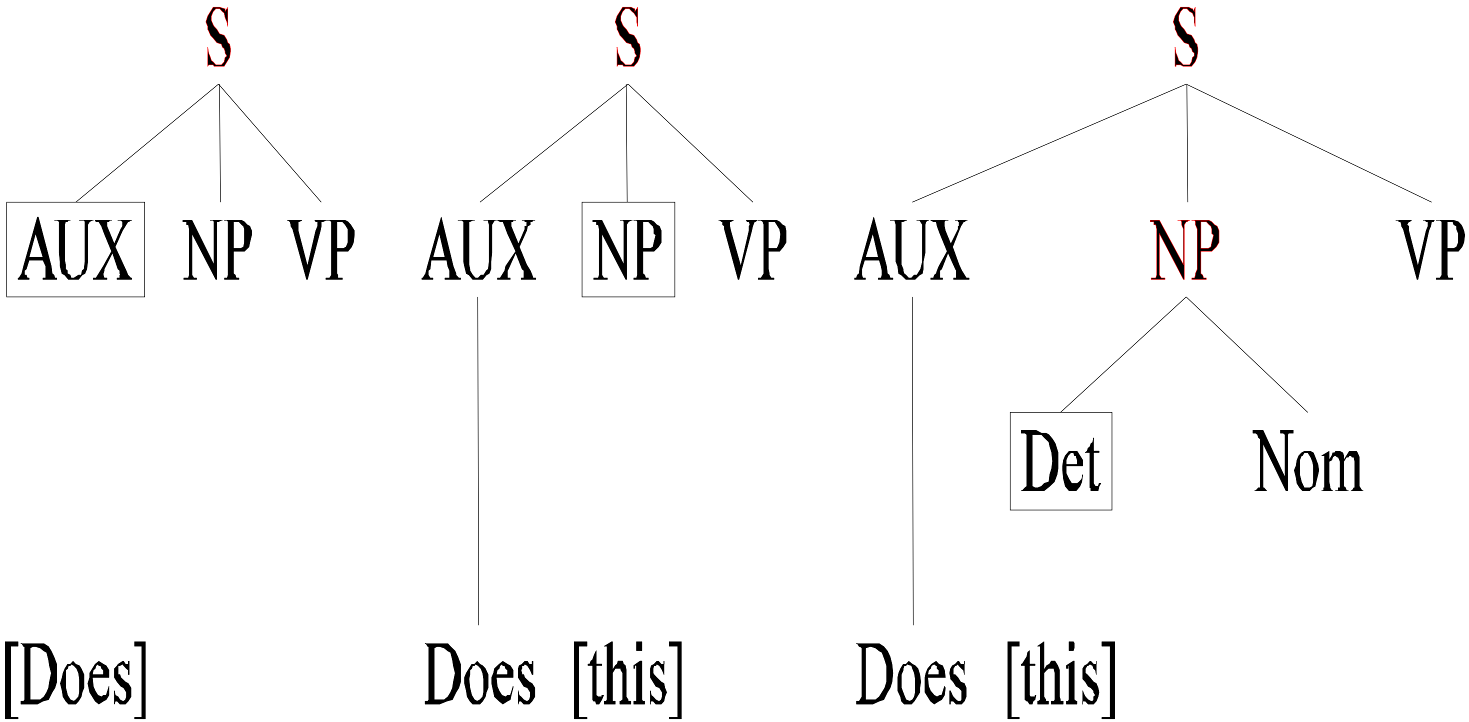
Continue putting NP "rules" on agenda



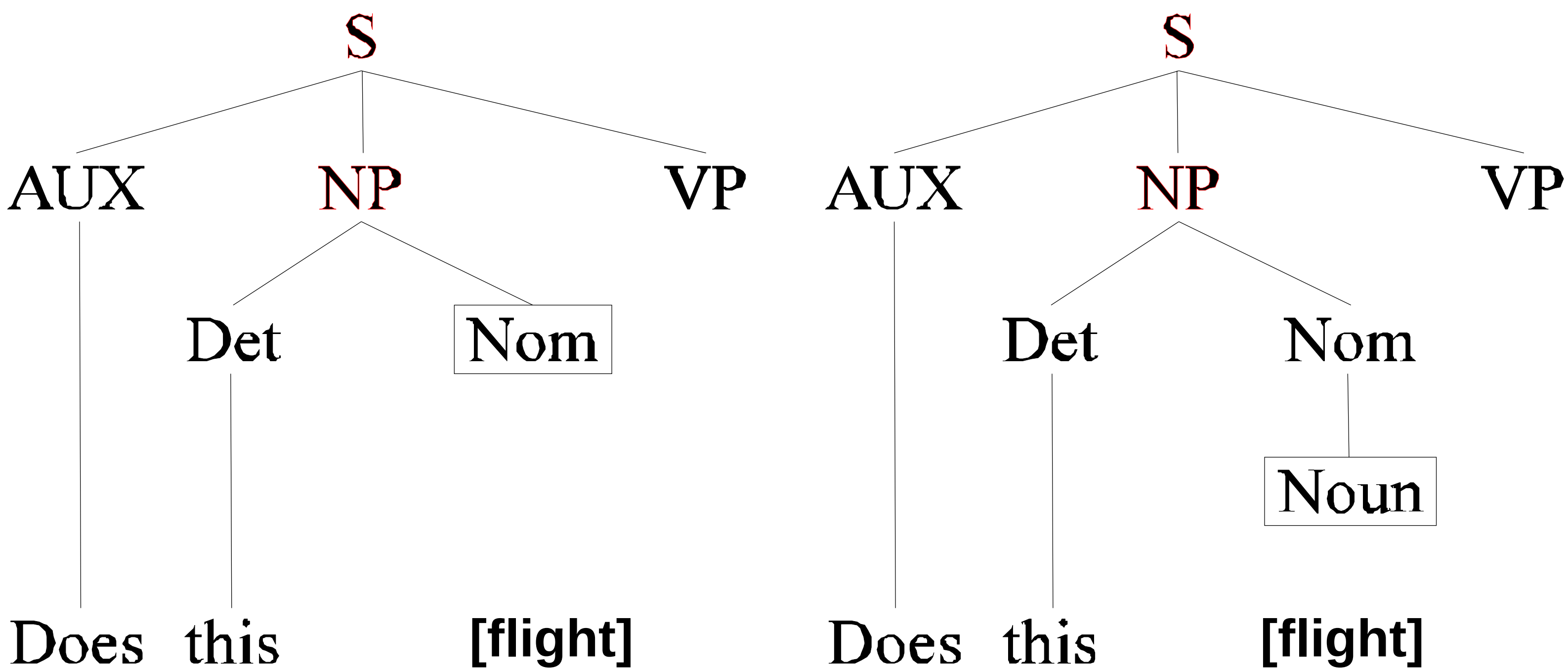
"Does this flight include a meal?" Parsing Overview



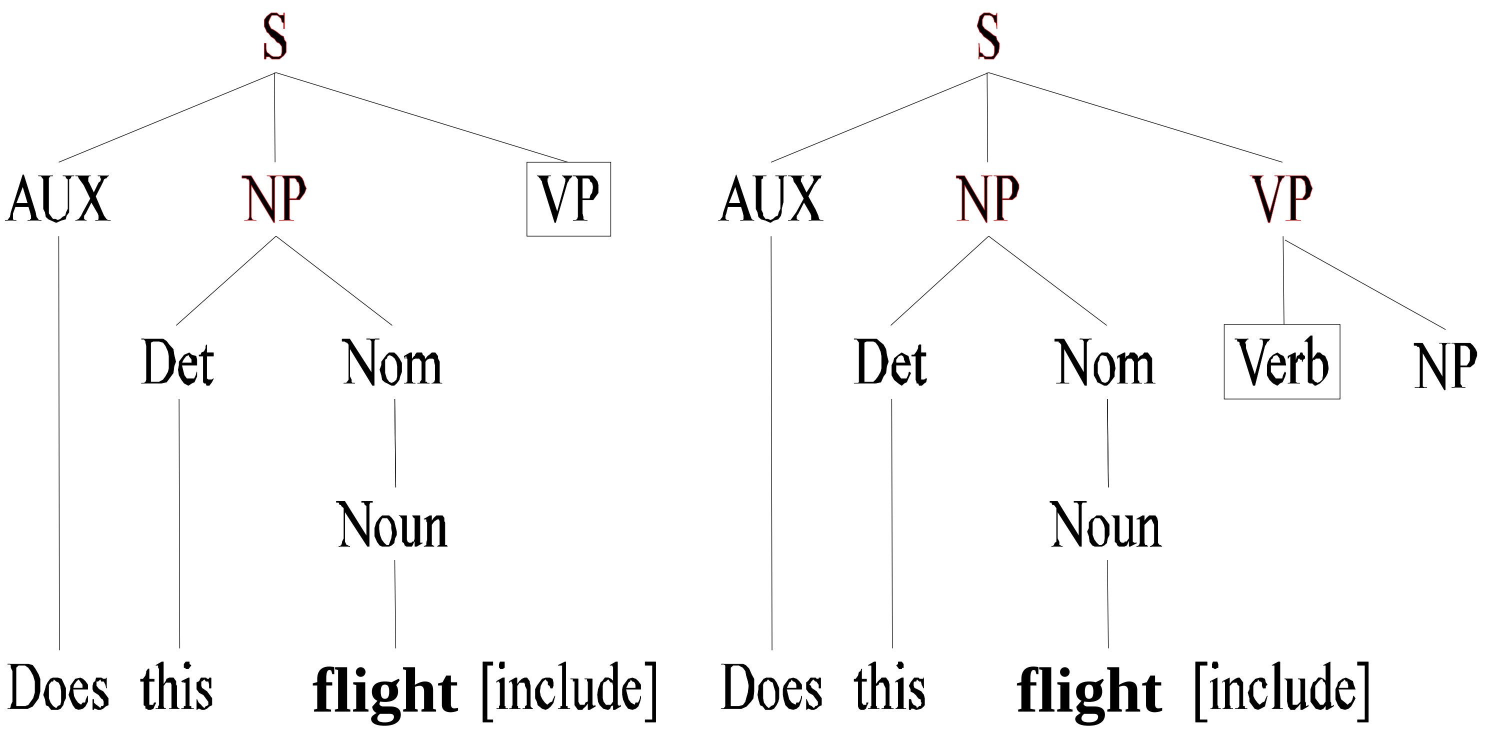
"Does this flight include a meal?" Parsing Overview (cont.)



"Does this flight include a meal?" Parsing Overview (cont.)



"Does this flight include a meal?" Parsing Overview (cont.)



A Bottom-Up Parsing Strategy

- **Depth-first search**:
  - State of parse is going to be initialized to the input words
  - At each step, look for Right Hand Side of a rule in the state, replace the matched right hand side with the Left Hand Side of the rule and continue
  - Agenda of search states: expand search space incrementally, exploring most recently generated state each time
  - When you reach a state that contains only the start symbol, you have successfully parsed

Bottom Up: “Book that flight”

- Curr: N det N
  Agenda: V det N

- Curr: Nom det N
  Agenda: N det Nom, V det N

- Curr: Nom det Nom
  Agenda: N det Nom, V det N

- Curr: Nom NP
  Agenda: N det Nom, V det N

- Curr: N det Nom
  Agenda: V det N

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP → PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

Bottom Up: “Book that flight”

- Curr: V det N
  Agenda:

- Curr: VP det N
  Agenda: V det Nom

- Curr: VP NP
  Agenda: V det Nom

- Curr: S NP
  Agenda: V det Nom

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP → PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

Bottom Up: "Book that flight"

- Curr: V det Nom
  Agenda:
- Curr: V NP
  Agenda:
- Curr: VP
  Agenda:
- Curr: S
  Agenda:
- SUCCESS!!!!

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP →PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

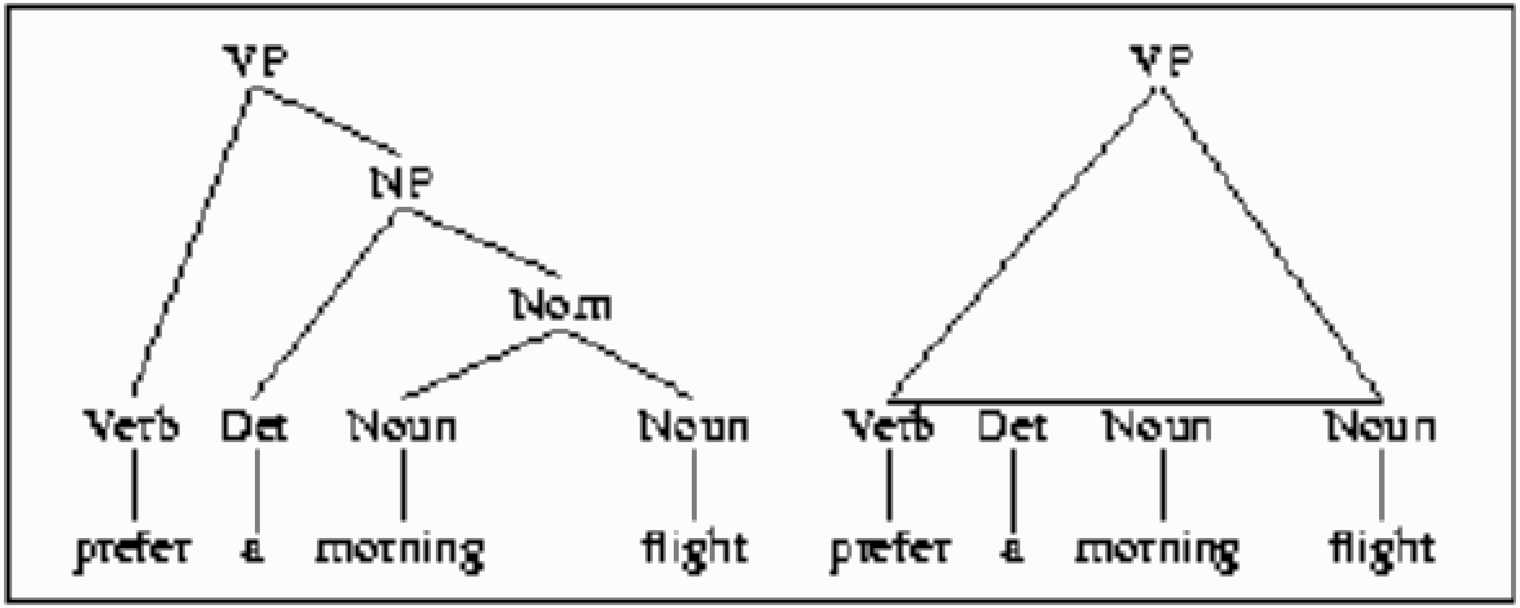
What’s wrong with….

- Top-Down parsers never explore illegal parses (e.g. can’t form an S) -- but waste time on trees that can never match the input
- Bottom-Up parsers never explore trees inconsistent with input -- but waste time exploring illegal parses (no S root)
- For both: control strategy -- how explore search space?
  - Pursuing all parses in parallel or backtrack or …?
  - Which rule to apply next?
  - Which node to expand next?

Left Corners: Top-Down Parsing with Bottom-Up Filtering

- We saw: Top-Down, depth-first, L2R parsing
  - Expands non-terminals along the tree's left edge down to leftmost leaf of tree
  - Moves on to expand down to next leftmost leaf…
  - Note: In successful parse, current input word will be first word in derivation of node the parser currently processing
  - So….look ahead to left-corner of the tree
    - B is a left-corner of A if A =*=> Bα
    - Build table with left-corners of all non-terminals in grammar and consult before applying rule

Left Corners



Calculating Left Corners

For each constituent on the LHS of a rule, follow through LHS until you find a preterminal (lexical category). That's the left corner.

Consider S – one rule at a time

Det

PropN

Aux

V

Same procedure for other constituents

Grammar:

S → NP VP
S → Aux NP VP
S → VP
NP → Det Nom
NP → PropN
Nom → N Nom
Nom → N
Nom → Nom PP
VP → V NP
VP → V

Left-Corner Table for CFG

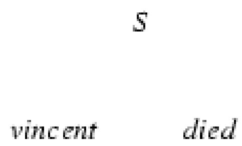
Category	Left Corners
S	Det, PropN, Aux, V
NP	Det, PropN
Nom	N
VP	V

Left-Corner Example

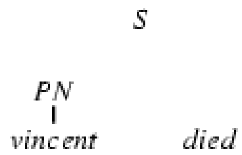
- *Assume that we again have the following grammar:*
- *Now, let's look at how a left-corner recognizer would proceed to recognize vincent died.*

$S \longrightarrow NP\ VP$
$NP \longrightarrow Det\ N$
$NP \longrightarrow PN$
$VP \longrightarrow IV$
$Det \longrightarrow the$
$N \longrightarrow robber$
$PN \longrightarrow Vincent$
$IV \longrightarrow died$

1.) Input: *vincent died*. Recognize an *s*. (Top-down prediction.)

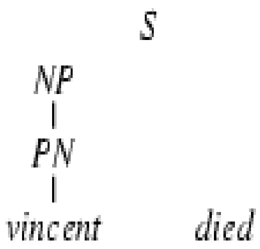


2.) The category of the first word of the input is *PN*. (Bottom-up step using a lexical rule.)



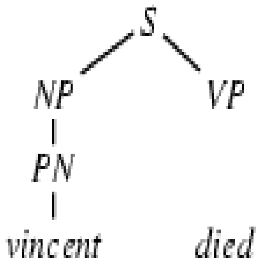
Left-Corner Example

3.) Select a rule that has *PN* at its left corner: $NP \longrightarrow PN$. (Bottom-up step using a phrase structure rule.)



4.) Select a rule that has *NP* at its left corner: $S \longrightarrow NP\ VP$. (Bottom-up step.)

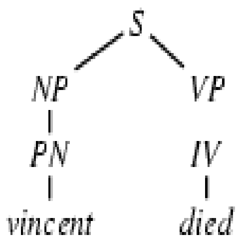
5.) Match! The left hand side of the rule matches with *S*, the category we are trying to recognize.



Left-Corner Example

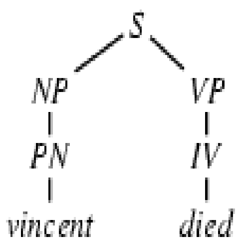
6.) Input: *died*. Recognize a *VP*. (Top-down prediction.)

7.) The category of the first word of the input is *IV*. (Bottom-up step.)



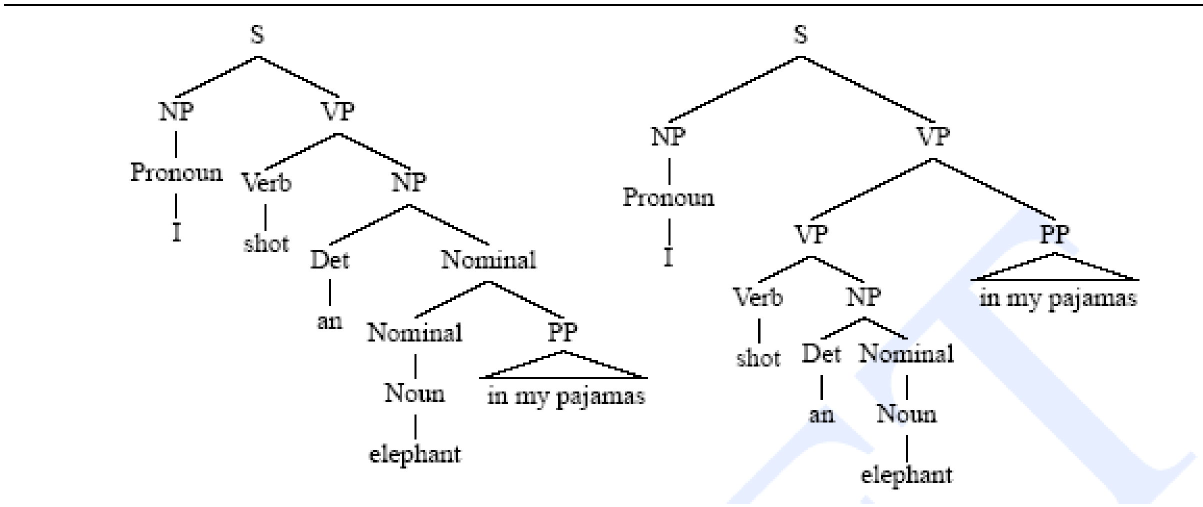
8.) Select a rule that has *IV* at its left corner: $VP \longrightarrow IV$. (Bottom-up step.)

9.) Match! The left hand side of the rule matches with *VP*, the category we are trying to recognize.



Ambiguity

- Structural ambiguity – occurs when the grammar assigns more than one possible parse to a sentence
  - Attachment ambiguity – attached to the parse tree more than one place (We saw the Eiffel Tower flying to Paris)
  - Coordination ambiguity – “old men and women”



Dynamic Programming Parsing Methods – CKY Parsing

- Bottom-up
- Chomsky Normal Form(CNF)
  - A->B C or A -> *w*
- Conversion to CNF
  - Mix terminals and non-terminals -> introduce a new dummy non-terminal : INF-VP -> to VP : INF-VP ->TO VP, TO->to
  - Unit productions (single nonterminal on the right) -> rewriting the right-hand side of the original rules with the right-hand side of all the non-unit production rules that they ultimately lead to. A=>B and B->γ (non-unit production), then A-> γ
  - Right-hand side longer than 2 → introduce new non-terminals. S->Aux NP VP : S->X1 VP, X1-> Aux NP

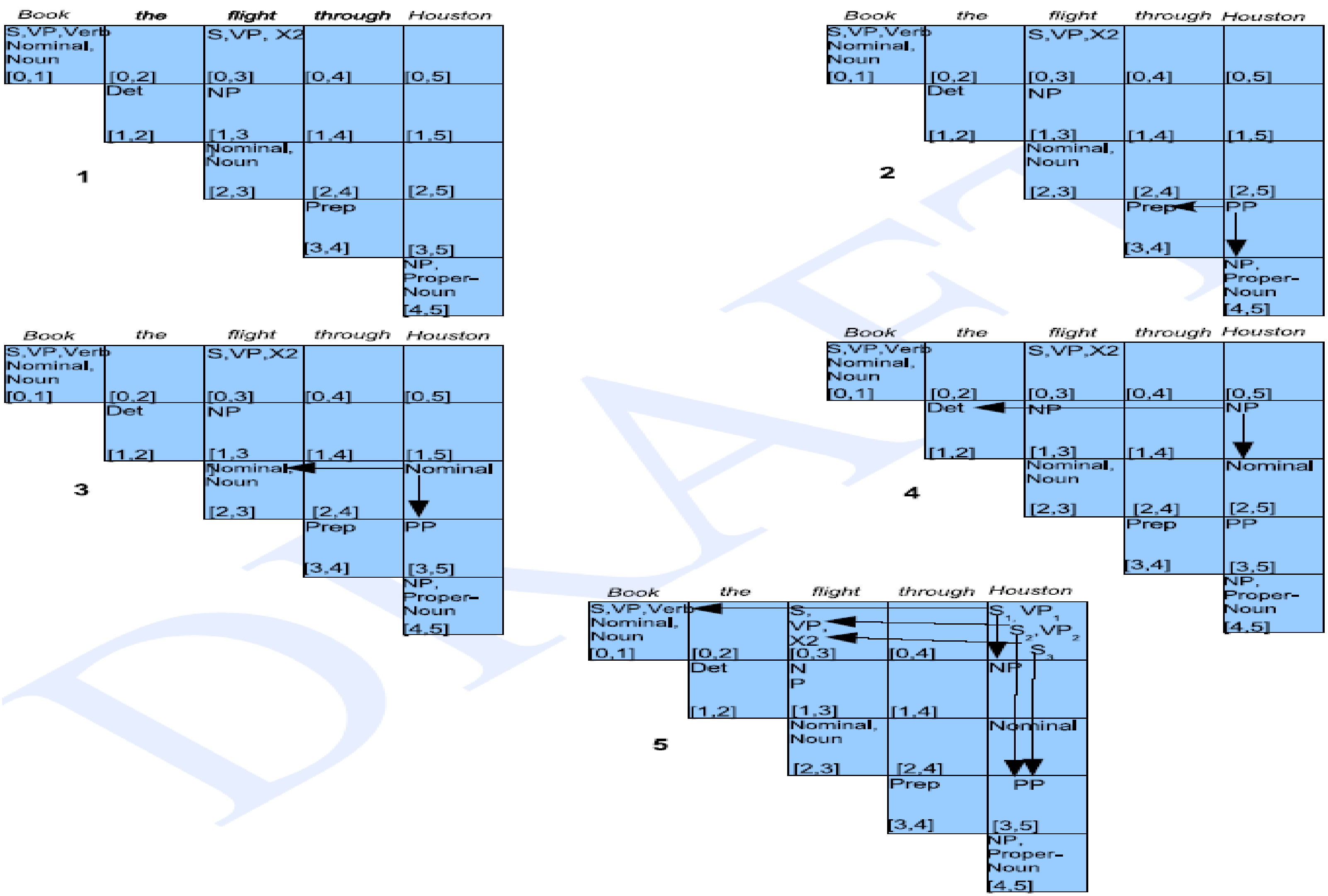
L1 for CKY example

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid TWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Figure 13.1 The $\mathscr{L}_1$ miniature English grammar and lexicon.

CNF of L1

$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$


Book
the
flight
through
Houston
S,VP,Verb Nominal, Noun [0,1]
[0,2]
S,VP, X2 [0,3]
[0,4]
[0,5]
Det [1,2]
NP [1,3]
[1,4]
[1,5]
Nominal, Noun [2,3]
[2,4]
[2,5]
Prep [3,4]
[3,5]
PP
Nominal
NP, Proper-Noun [4,5]
S1, VP1
S2, VP2
S3
1
2
3
4
5

Dynamic Programming Parsing Methods – The Earley Algorithm

- Top-down search
- Single left-to-right pass that fills an array (chart) that has N+1 entries
- Chart contains three kinds of information
  - A subtree corresponding to a single grammar rule
  - Information about the progress made in completing this subtree
  - The position of the subtree with respect to the input
- Dotted rule(.)
  - S ->•VP, [0,0], two numbers- where state begins and where its dot lies.

Dynamic Programming Parsing Methods – The Earley Algorithm

- Three Operators
  - Predictor – to create new states representing top-down expectations generated during the parsing process. Predictor is applied to any state that has a non-terminal immediately to the right of its dot that is not a part-of-speech category.
  - Scanner – When a state has a part-of-speech category to the right of the dot, SCANNER is called to examine the input and incorporate a state corresponding to the prediction of a word with a particular part-of-speech into the chart.
  - Completer- applied to a state when its dot has reached the right end of the rule.

Dynamic Programming Parsing Methods – The Earley Algorithm

Chart[0]	S0	$\gamma \rightarrow \bullet\ S$	[0,0]	Dummy start state
	S1	$S \rightarrow \bullet\ NP\ VP$	[0,0]	Predictor
	S2	$S \rightarrow \bullet\ Aux\ NP\ VP$	[0,0]	Predictor
	S3	$S \rightarrow \bullet\ VP$	[0,0]	Predictor
	S4	$NP \rightarrow \bullet\ Pronoun$	[0,0]	Predictor
	S5	$NP \rightarrow \bullet\ Proper\text{-}Noun$	[0,0]	Predictor
	S6	$NP \rightarrow \bullet\ Det\ Nominal$	[0,0]	Predictor
	S7	$VP \rightarrow \bullet\ Verb$	[0,0]	Predictor
	S8	$VP \rightarrow \bullet\ Verb\ NP$	[0,0]	Predictor
	S9	$VP \rightarrow \bullet\ Verb\ NP\ PP$	[0,0]	Predictor
	S10	$VP \rightarrow \bullet\ Verb\ PP$	[0,0]	Predictor
	S11	$VP \rightarrow \bullet\ VP\ PP$	[0,0]	Predictor
Chart[1]	S12	$Verb \rightarrow book\ \bullet$	[0,1]	Scanner
	S13	$VP \rightarrow Verb\ \bullet$	[0,1]	Completer
	S14	$VP \rightarrow Verb\ \bullet\ NP$	[0,1]	Completer
	S15	$VP \rightarrow Verb\ \bullet\ NP\ PP$	[0,1]	Completer
	S16	$VP \rightarrow Verb\ \bullet\ PP$	[0,1]	Completer
	S17	$S \rightarrow VP\ \bullet$	[0,1]	Completer
	S18	$VP \rightarrow VP\ \bullet\ PP$	[0,1]	Completer
	S19	$NP \rightarrow \bullet\ Pronoun$	[1,1]	Predictor
	S20	$NP \rightarrow \bullet\ Proper\text{-}Noun$	[1,1]	Predictor
	S21	$NP \rightarrow \bullet\ Det\ Nominal$	[1,1]	Predictor
	S22	$PP \rightarrow \bullet\ Prep\ NP$	[1,1]	Predictor
Chart[2]	S23	$Det \rightarrow that\ \bullet$	[1,2]	Scanner
	S24	$NP \rightarrow Det\ \bullet\ Nominal$	[1,2]	Completer
	S25	$Nominal \rightarrow \bullet\ Noun$	[2,2]	Predictor
	S26	$Nominal \rightarrow \bullet\ Nominal\ Noun$	[2,2]	Predictor
	S27	$Nominal \rightarrow \bullet\ Nominal\ PP$	[2,2]	Predictor
Chart[3]	S28	$Noun \rightarrow flight\ \bullet$	[2,3]	Scanner
	S29	$Nominal \rightarrow Noun\ \bullet$	[2,3]	Completer
	S30	$NP \rightarrow Det\ Nominal\ \bullet$	[1,3]	Completer
	S31	$Nominal \rightarrow Nominal\ \bullet\ Noun$	[2,3]	Completer
	S32	$Nominal \rightarrow Nominal\ \bullet\ PP$	[2,3]	Completer
	S33	$VP \rightarrow Verb\ NP\ \bullet$	[0,3]	Completer
	S34	$VP \rightarrow Verb\ NP\ \bullet\ PP$	[0,3]	Completer
	S35	$PP \rightarrow \bullet\ Prep\ NP$	[3,3]	Predictor
	S36	$S \rightarrow VP\ \bullet$	[0,3]	Completer
	S37	$VP \rightarrow VP\ \bullet\ PP$	[0,3]	Completer