Natural Language Processing
CSE 325/425

Sihong Xie

Lecture 16:

- CYK parsing algorithm

Syntactic parsing

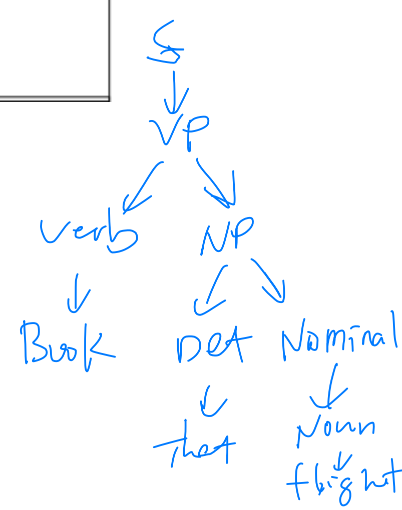
Syntactic parsing: assign valid parsing trees to a sentene, using a CFG.

- derivations of a sentence starting from S can be represented as a tree.
- generation of sentences using a CFG is the reverse process.

S → *NP VP*	*Det* → *that* | *this* | *a*
S → *Aux NP VP*	*Noun* → *book* | *flight* | *meal* | *money*
S → *VP*	*Verb* → *book* | *include* | *prefer*
NP → *Det Nominal*	*Aux* → *does*
Nominal → *Noun*	
Nominal → *Noun Nominal*	*Prep* → *from* | *to* | *on*
NP → *Proper-Noun*	*Proper-Noun* → *Houston* | *TWA*
VP → *Verb*	
VP → *Verb NP*	*Nominal* → *Nominal PP*

⟺ *Book that flight.*

Parsing tree.



S → VP
VP → Verb NP
Verb → Book
NP → Det Nominal
Det → that
Nominal → Noun
Noun → flight

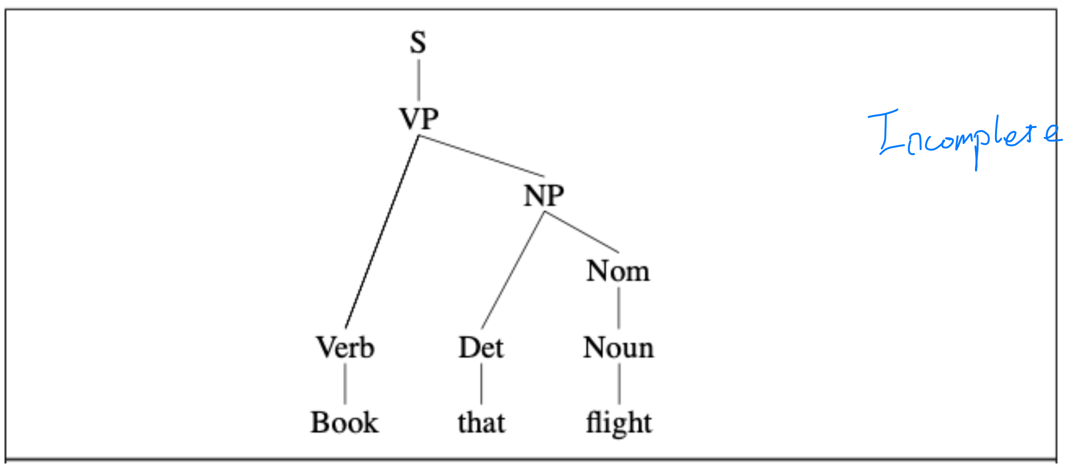
Syntactic parsing as search

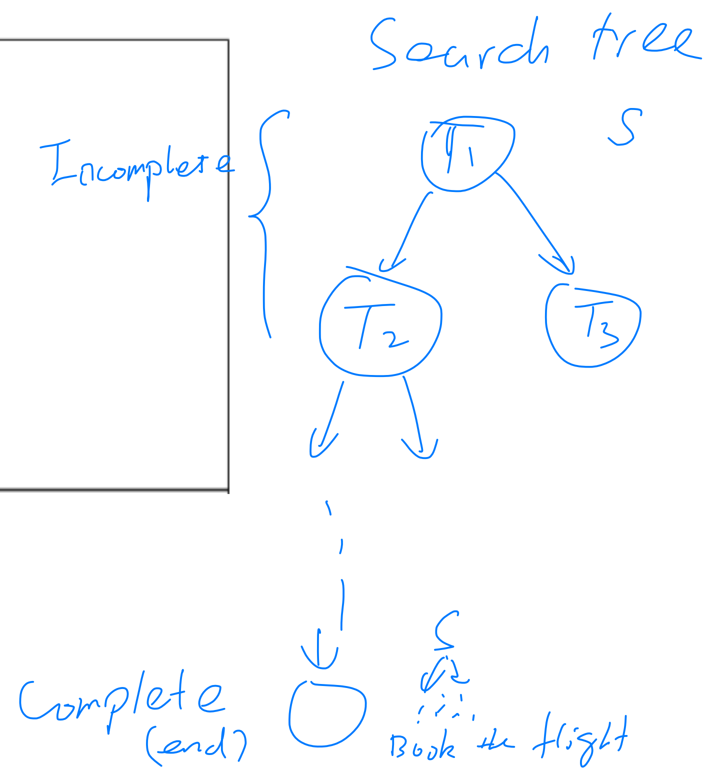
Search algorithms are commonly used in computer science and AI.

Input: sentence *"Book that flight"*

Output: parsing trees with root = S and leaves = the sentence

- Top-down and buttom-up searches.
  - *Note: not to go up and down a fixed parsing tree but in a search tree.*



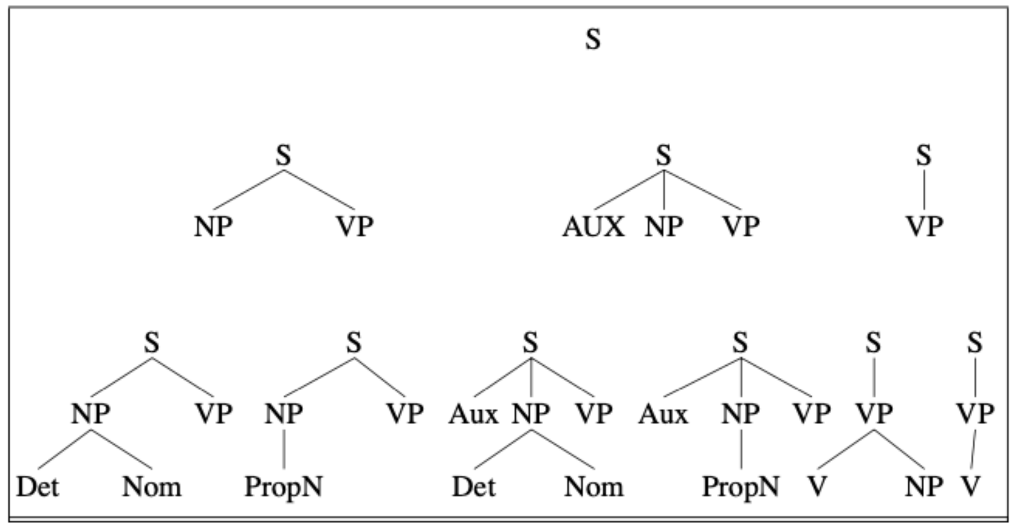


Syntactic parsing as search

Top-down search

- start from the root = S
- expand a non-terminal on an unfinished tree, using a rule in the given CFG.
- Pruning: when there is no hope of matching
- Stop with a match of the input sentence

S → *NP VP*	*Det* → *that* | *this* | *a*
S → *Aux NP VP*	*Noun* → *book* | *flight* | *meal* | *money*
S → *VP*	*Verb* → *book* | *include* | *prefer*
NP → *Det Nominal*	*Aux* → *does*
Nominal → *Noun*	
Nominal → *Noun Nominal*	*Prep* → *from* | *to* | *on*
NP → *Proper-Noun*	*Proper-Noun* → *Houston* | *TWA*
VP → *Verb*	
VP → *Verb NP*	*Nominal* → *Nominal PP*

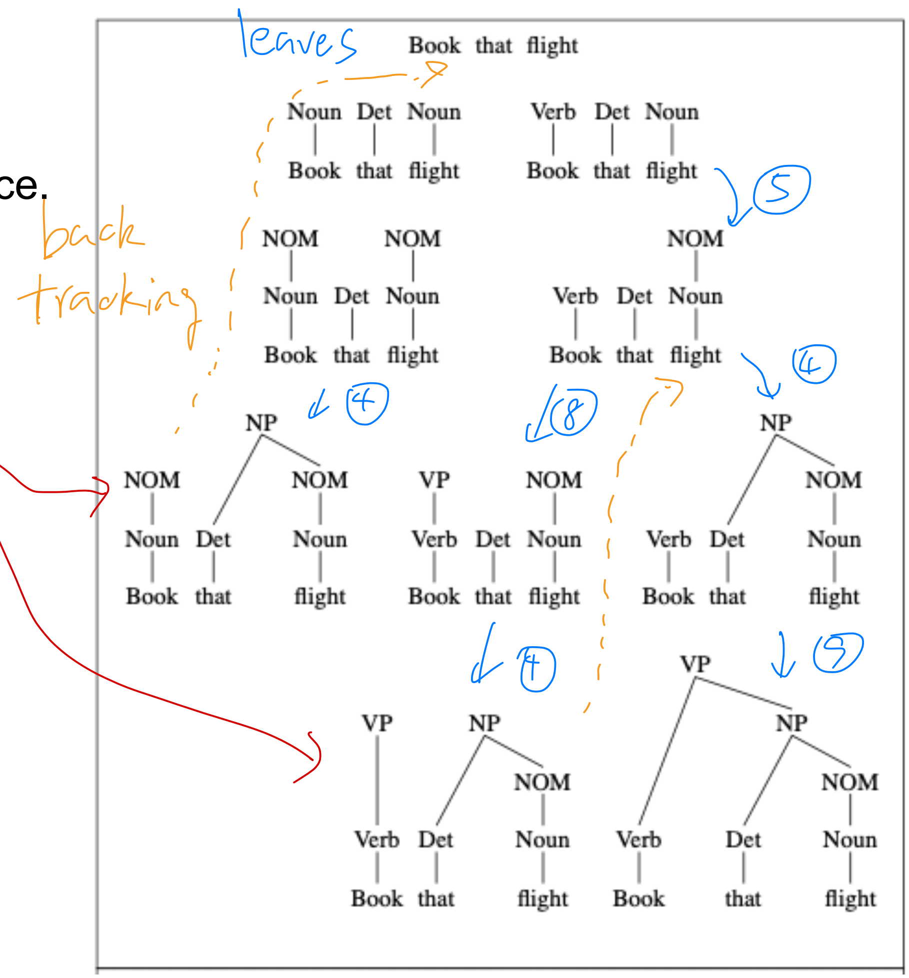


Syntactic parsing as search

Bottom-up search

- start from the leaves (bottom) = input sentence.
- create a non-terminal from existing symbols, using a rule in the given CFG.
- Pruning: when can't find a match of any right-hand-side match
- Stop when arriving at root = S.

$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
4 $NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
5 $Nominal \rightarrow Noun$	
$Nominal \rightarrow Noun\ Nominal$	$Prep \rightarrow from \mid to \mid on$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid TWA$
8 $VP \rightarrow Verb$	
9 $VP \rightarrow Verb\ NP$	$Nominal \rightarrow Nominal\ PP$



Discussion of search-based methods

- Ambiguity and multiple “valid” parses.
  - Attachment ambiguity

VP

NP VP

I saw [the Grand Canyon] [flying to New York.]

?

  - Coordination ambiguity

AP

ADJ NP

old [men and women]

Coordination

AP

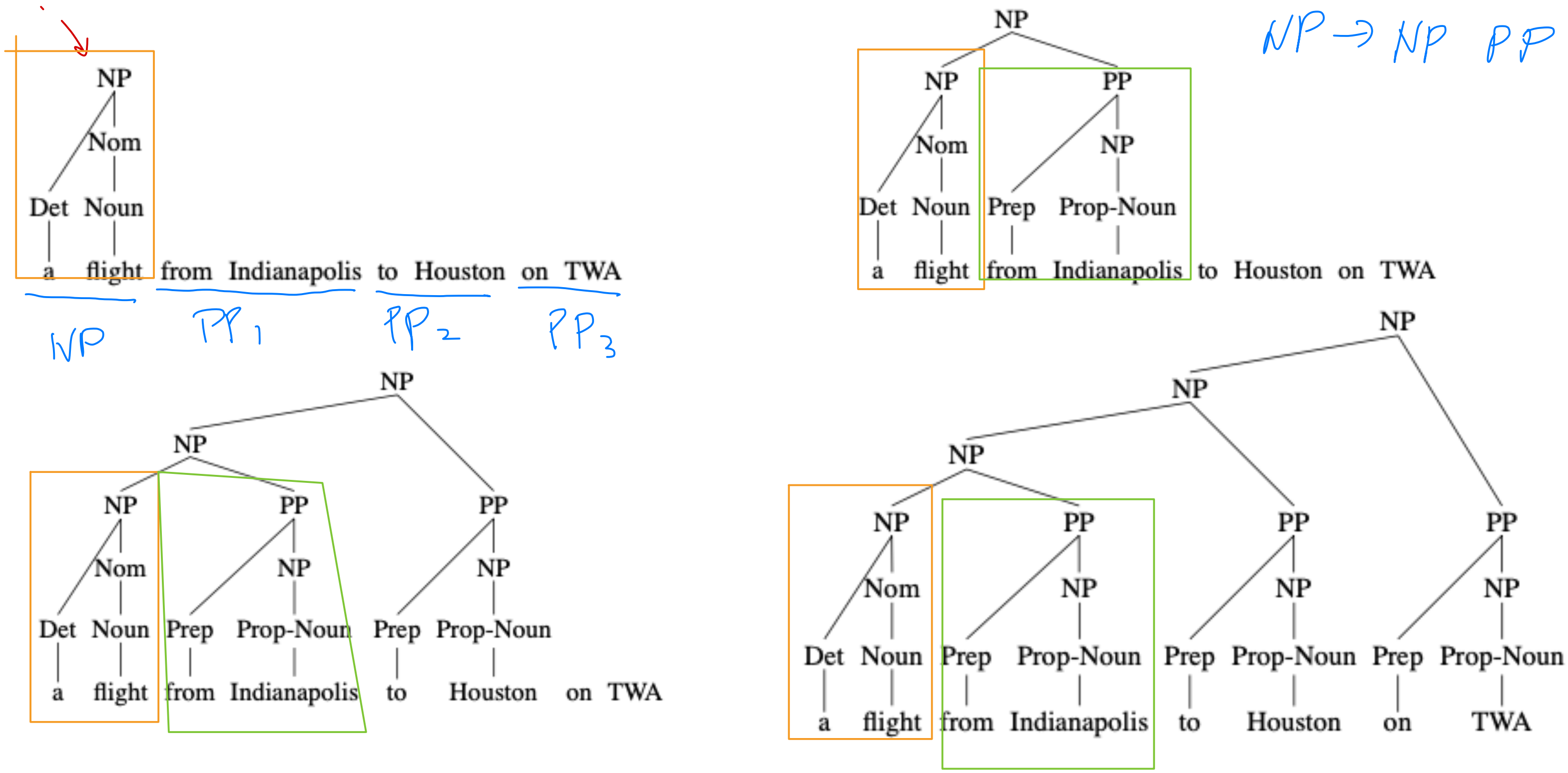
NP

Discussion of search-based methods

- Costly: repeatly parsing the same constituent.
  - the PP and the attachment ambiguity lead to exponential time complexity $C(n) = \frac{1}{n+1}\binom{2n}{n}$

S

NP → NP PP



CYK parsing algorithm

- It is a dynamic programming (DP) algorithm.
- Properties of DP:
    - Overlapping sub-problems: can re-use solutions to sub-problems;
        - Subproblems = parsing a sub-part of a sentence.
    - Optimal structures: optimal solutions => optimal sub-problem solutions.
        - successful parse of a constituent => successful parse of its parts.
- Reduce the cost of search by caching the solution to subproblems.
    - don’t repeatly parsing “*a flight*”.

Chomsky normal form (CNF)

- Allow simple a data structure for the DP algorithm called “CKY”
- Restrict CFG to the following form
  - There can be at most two symbols on the right hand side of any rule;
  - If there are two, both should be non-terminals (e.g. $A \rightarrow BC$);
  - If there is one, it should be a terminal (e.g. $A \rightarrow a$).
- There is no loss of expressiveness by the following conversions:
  - mixed production $A \rightarrow Bc \Rightarrow (A \rightarrow BC$ and $C \rightarrow c)$
  - unit production $A \rightarrow B \Rightarrow$ eliminate B by changing any $B \rightarrow \boldsymbol{\gamma}$ to $A \rightarrow \boldsymbol{\gamma}$.
  - long right hand side $A \rightarrow BCD \Rightarrow (A \rightarrow XD$ and $X \rightarrow BC)$;
    - can handle more than three symbols by recursively doing this conversion.

Chomsky normal form (CNF)

Conversion algorithm

1. handle mixed RHS
2. handle unit productions
3. handle long RHS

$\mathcal{L}_1$ **Grammar**	$\mathcal{L}_1$ **in CNF**
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

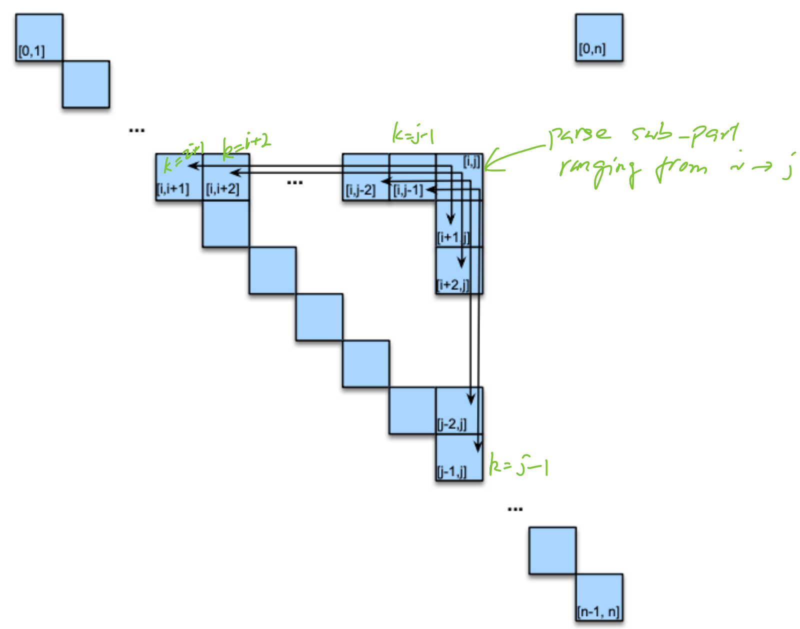
CYK parsing algorithm

- Use a matrix of size (n+1) x (n+1) if the sentence has n words.

  0 Book 1 the 2 flight 3 through 4 Houston 5

  i k1 k2 j

- The cell *(i, j)* represents all possible non-terminals that can generate the constituent ranging *i* to *j*.
- Due to CNF, each constituent can be broken down into two sub-constituents:
  - $(i, j) = (i, k) + (k\cancel{+1}, j)$
  - $j > i + 1$ and $0 \leq i < k < j \leq n$



e.g. $i = 0$

$j > 1$

$j = 2, 3, 4, 5$

CYK parsing algorithm

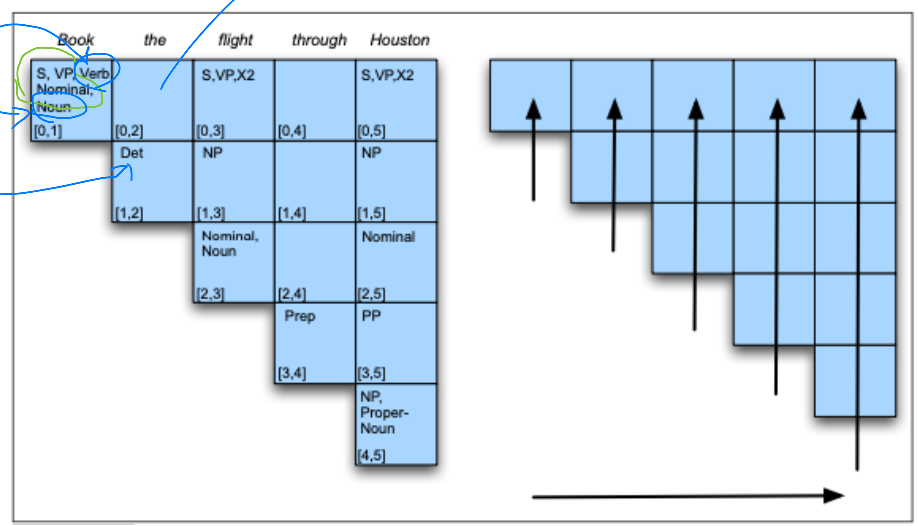
function CKY-PARSE(*words, grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do** ← going through cols.
  for all $\{A \mid A \rightarrow words[j] \in grammar\}$
    $table[j-1, j] \leftarrow table[j-1, j] \cup A$ } POS-tag the j-th word
  for $i \leftarrow$ **from** $j-2$ **down to** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
        $table[i,j] \leftarrow table[i,j] \cup A$

$$\overset{i\ \ j}{[0, 2]} = [0, \overset{k}{1}] + [1, 2]$$

verb → book
noun → book

Det → the



CYK parsing algorithm