CKY Parsing

Ling571
Deep Processing Approaches to NLP
January 11, 2017

Roadmap

- Motivation:
  - Inefficiencies of parsing-as-search
- Strategy: Dynamic Programming
- Chomsky Normal Form
  - Weak and strong equivalence
- CKY parsing algorithm

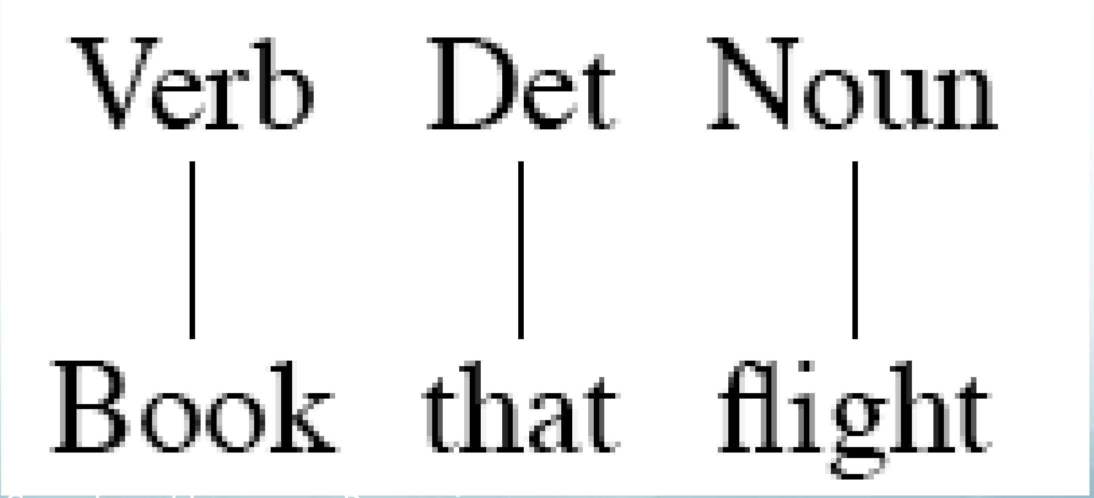
Bottom-Up Parsing

- Try to find all trees that span the input
  - Start with input string
    - Book that flight.
  - Use all productions with current subtree(s) on RHS
    - E.g., N →→ Book; V →→ Book
  - Stop when spanned by S (or no more rules apply)

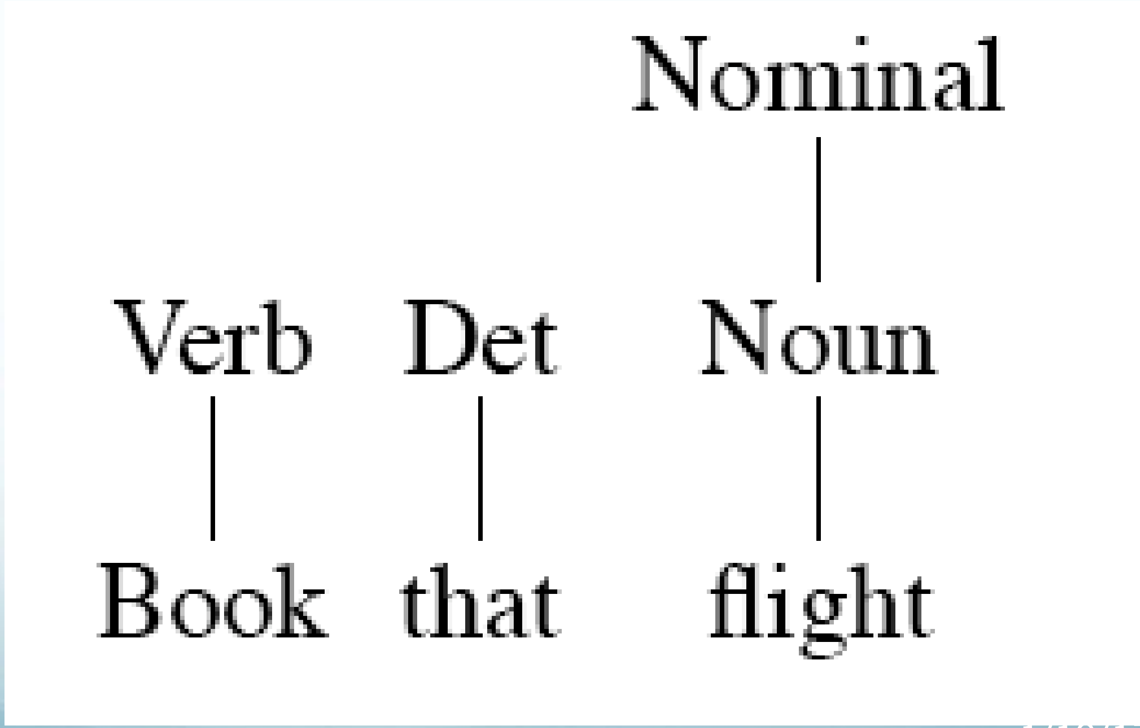
Bottom-Up Search

Book that flight

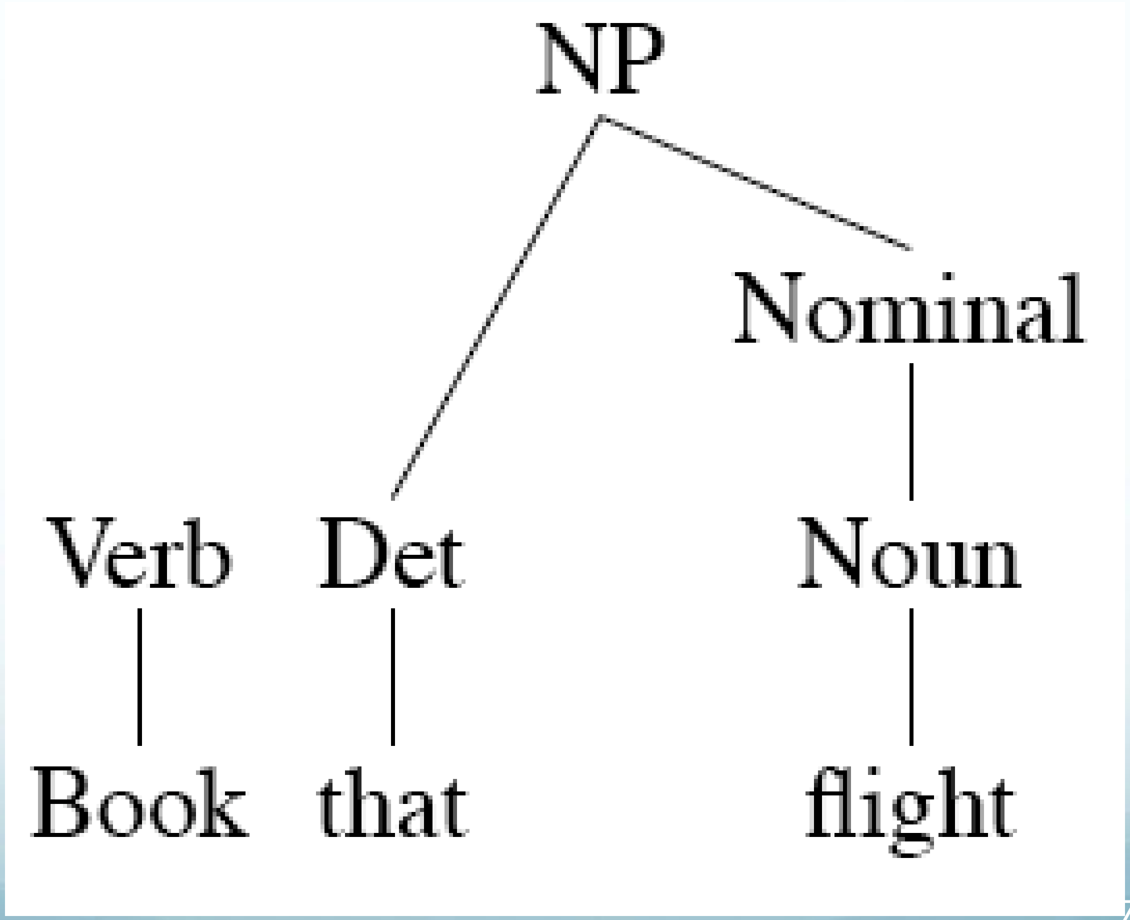
Bottom-Up Search



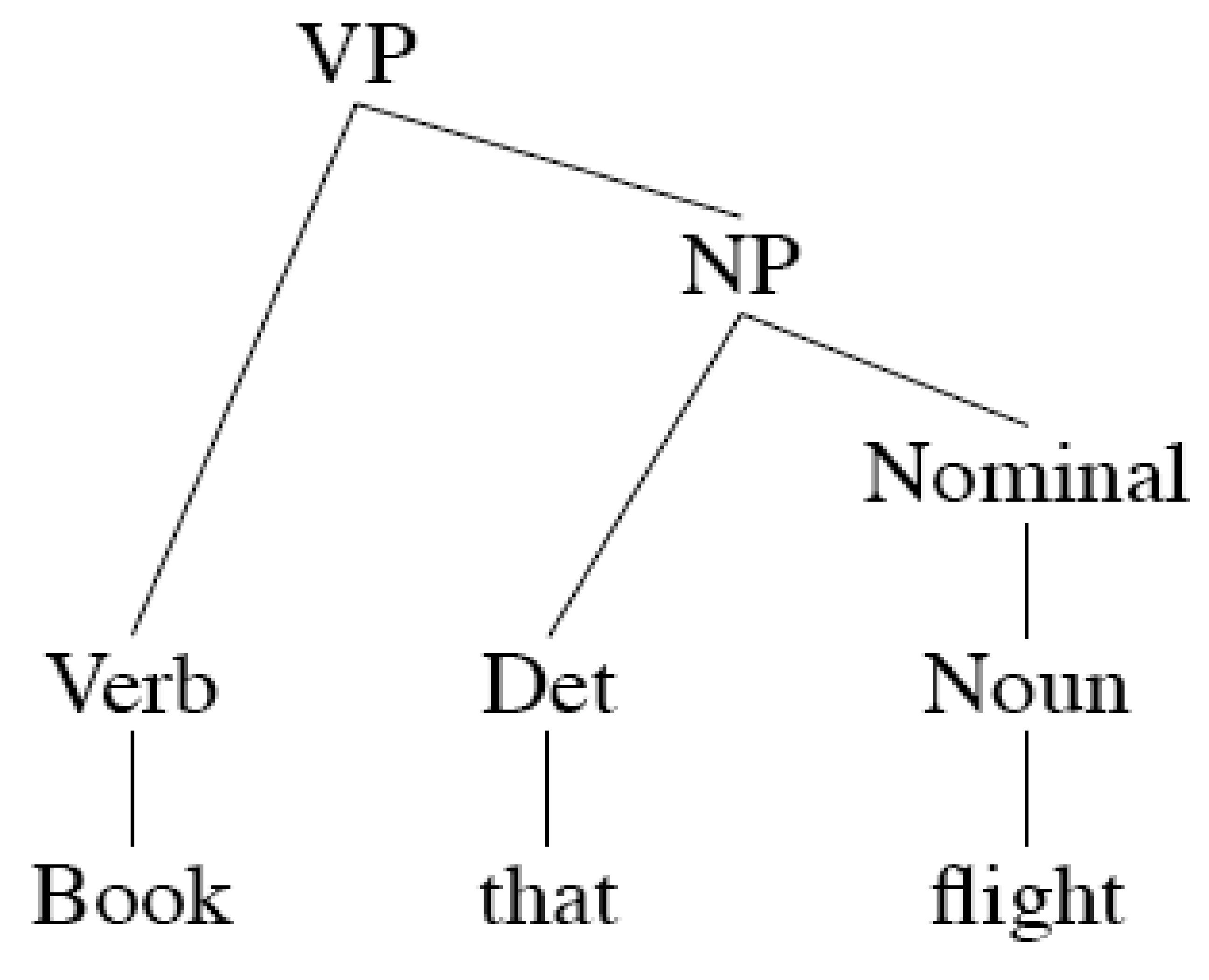
Bottom-Up Search



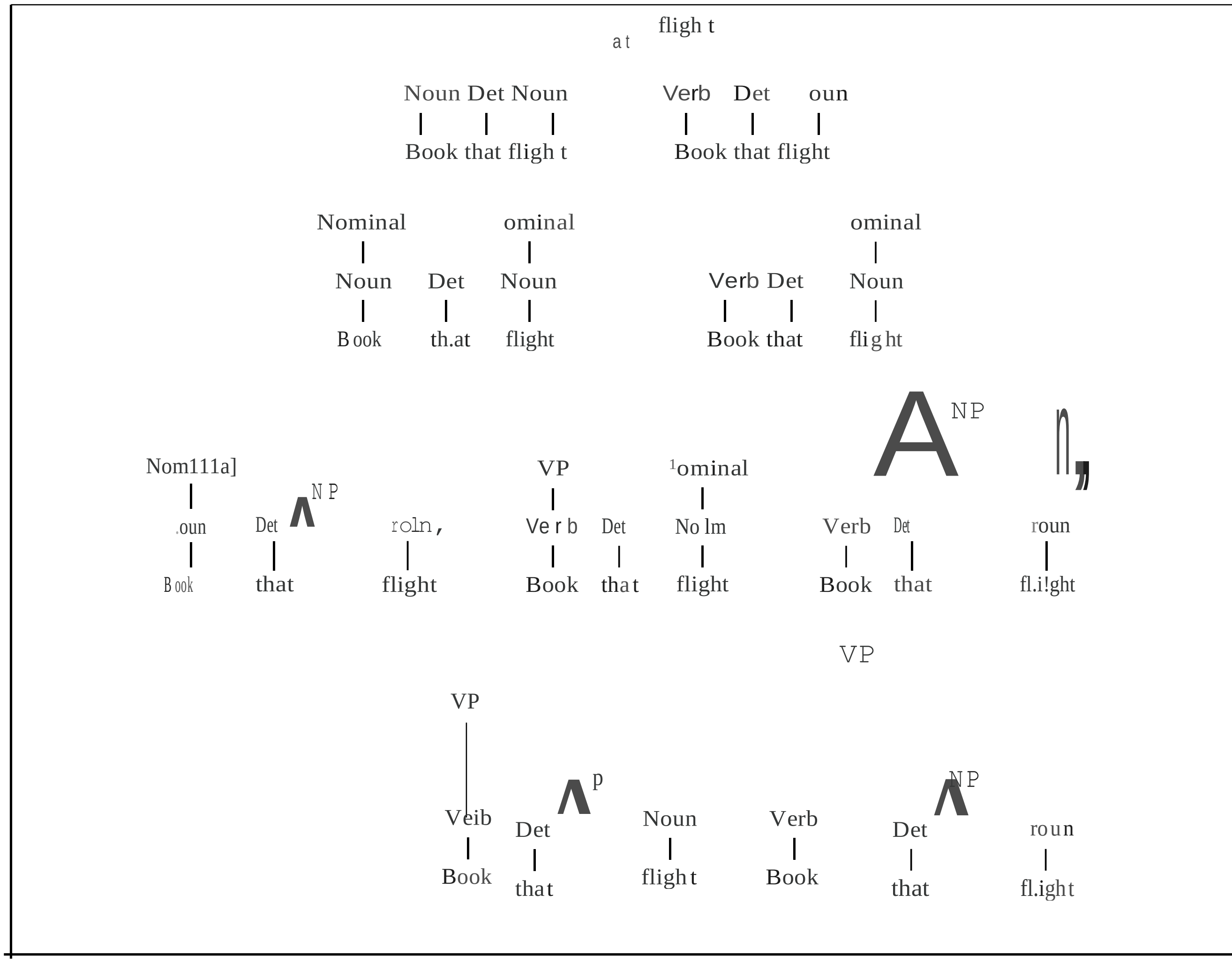
Bottom-Up Search



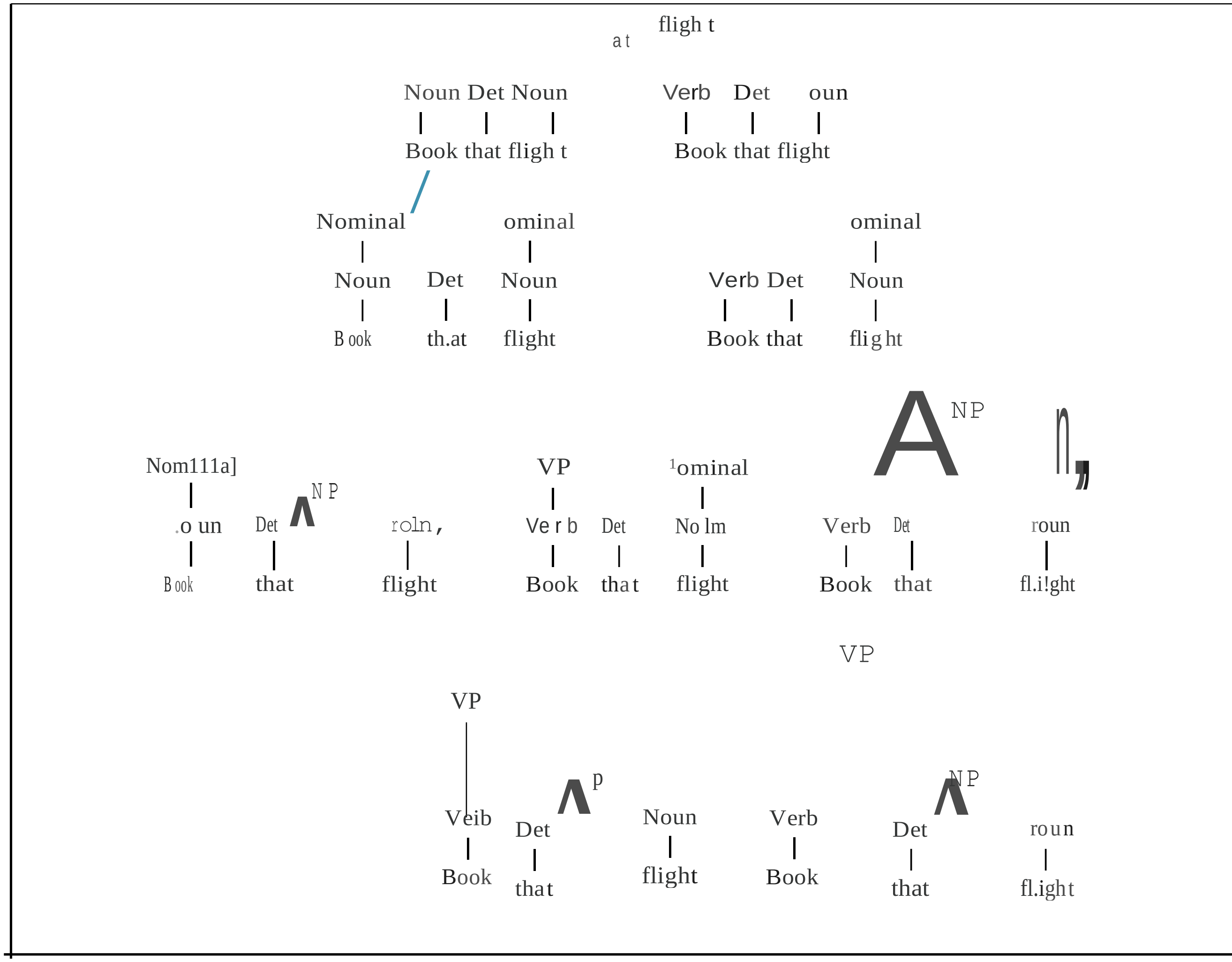
Bottom-Up Search



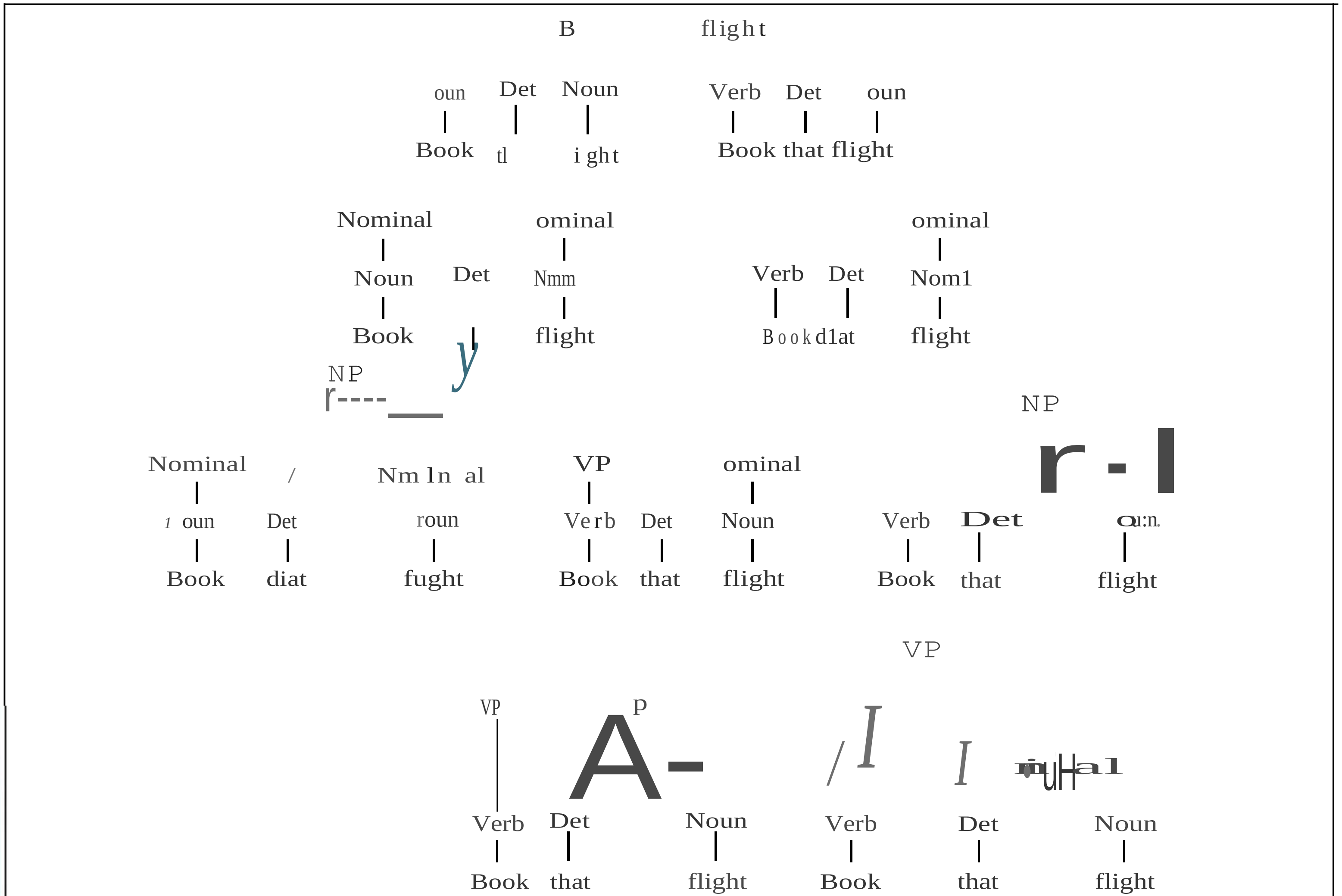
Bottom-Up Search



Bottom-Up Search



Bottom-Up Search



Pros and Cons of Bottom-Up Search

- Pros:
  - Will not explore trees that don’t match input
  - Recursive rules less problematic
  - Useful for incremental/ fragment parsing

- Cons:
  - Explore subtrees that will not fit full sentences

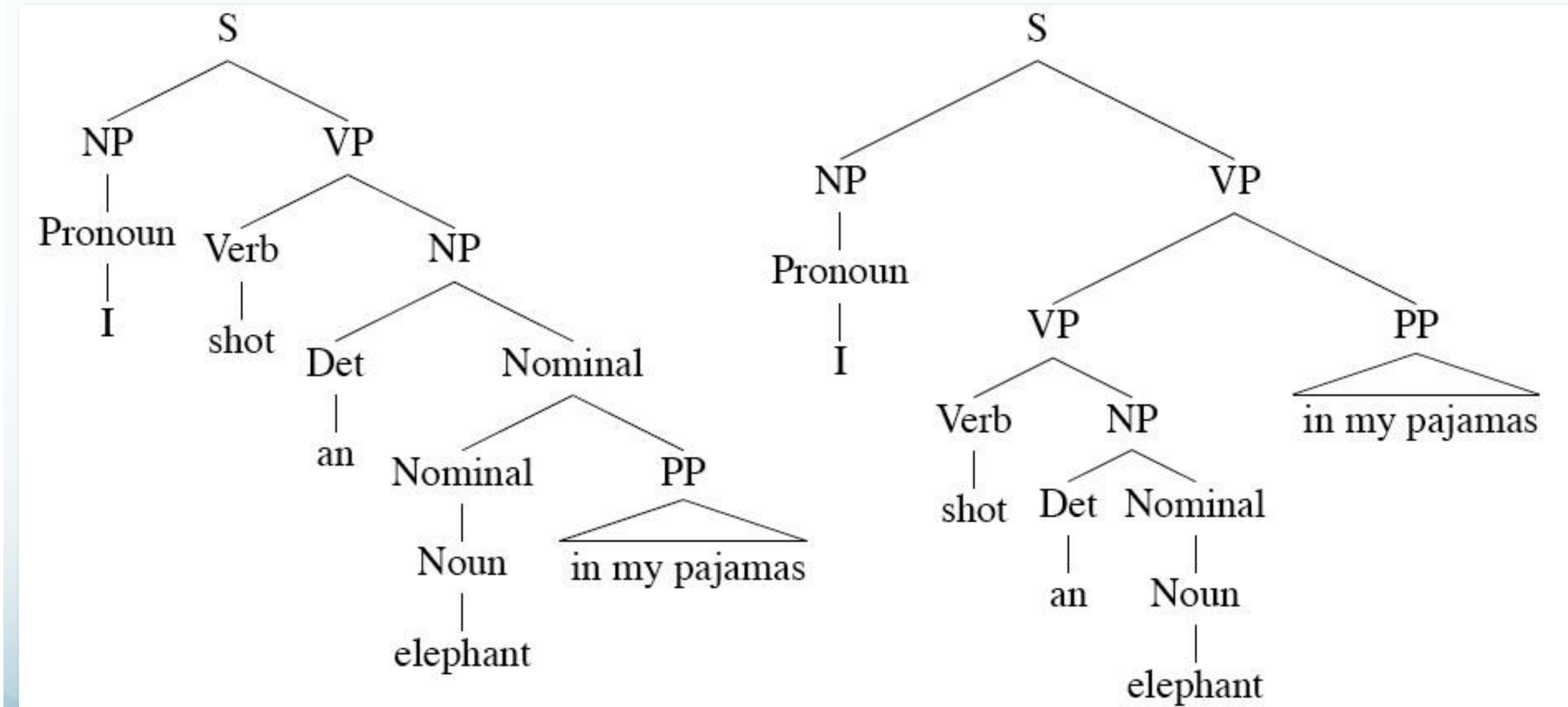
Parsing Challenges

- Ambiguity
- Repeated substructure
- Recursion

Parsing Ambiguity

- Many sources of parse ambiguity
  - Lexical ambiguity
    - Book/N; Book/V
  - Structural ambiguity: Main types:
    - Attachment ambiguity
      - Constituent can attach in multiple places
        - *I shot an elephant in my pyjamas*.
    - Coordination ambiguity
      - Different constituents can be conjoined
        - *Old men and women*

Ambiguity


S
NP
VP
Pronoun
Verb
NP
I
shot
Det
Nominal
an
Nominal
PP
Noun
in my pajamas
elephant
S
NP
VP
Pronoun
VP
PP
I
Verb
NP
in my pajamas
shot
Det
Nominal
an
Noun
elephant

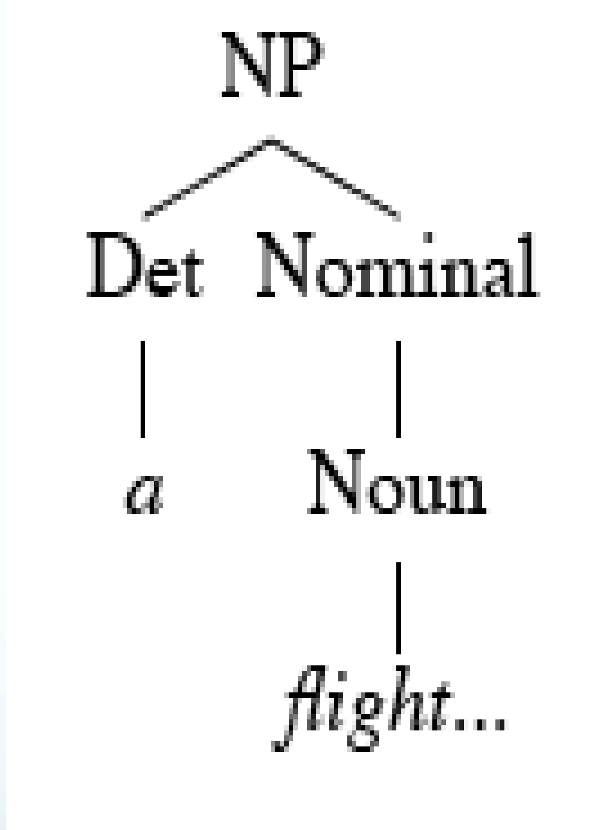
Disambiguation

- Global ambiguity:
  - Multiple complete alternative parses
  - Need strategy to select correct one
    - Approaches exploit other information
      - Statistical
        - Some prepositional structs more likely to attach high/low
        - Some phrases more likely, e.g., (old (men and women))
      - Semantic
      - Pragmatic
        - E.g., elephants and pyjamas
  - Alternatively, keep all
- Local ambiguity:
  - Ambiguity in subtree, resolved globally

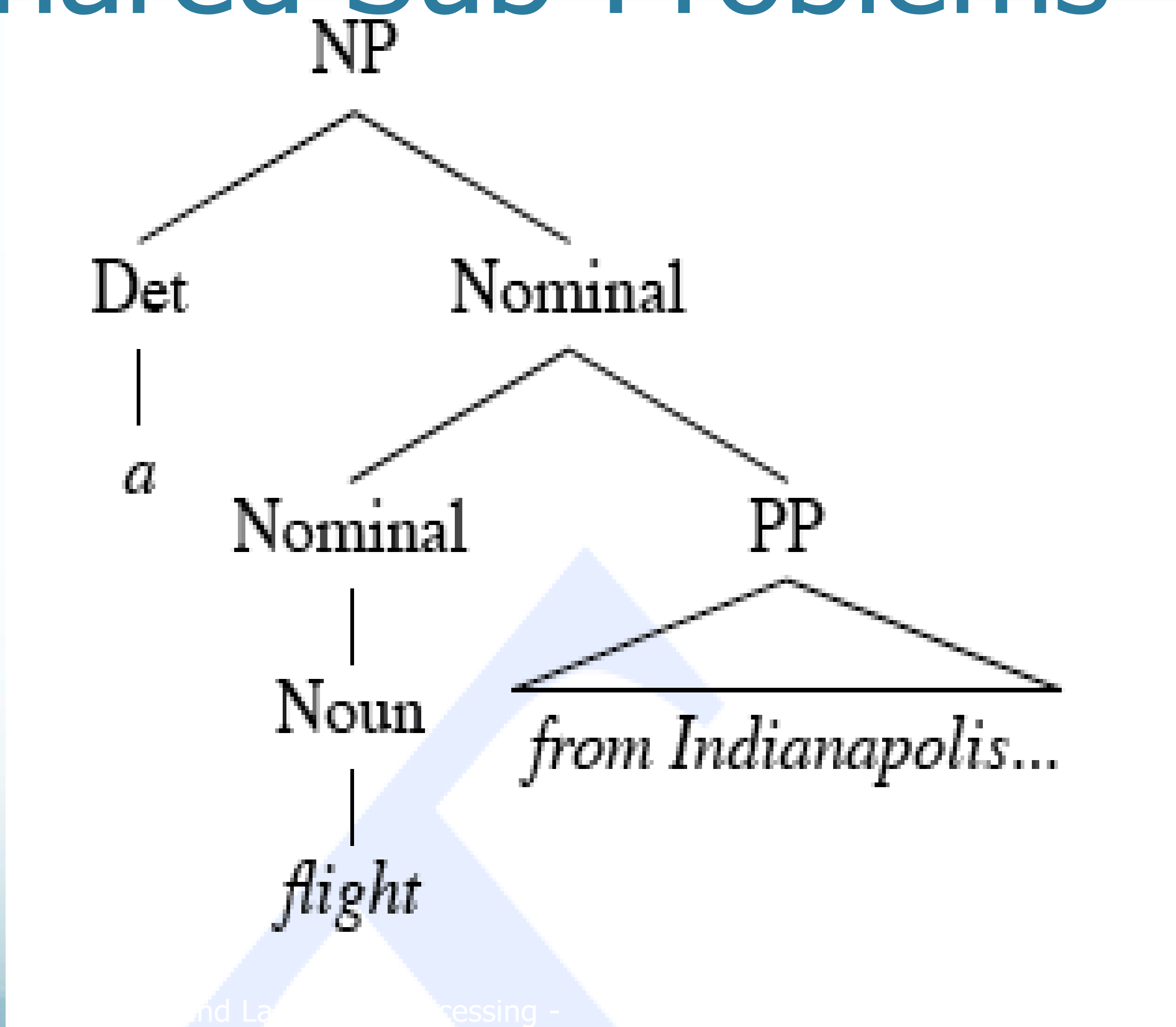
Repeated Work

- Top-down and bottom-up parsing both lead to repeated substructures
  - Globally bad parses can construct good subtrees
    - But overall parse will fail
    - Require reconstruction on other branch
  - No static backtracking strategy can avoid
- Efficient parsing techniques require storage of shared substructure
  - Typically with dynamic programming
- Example: *a flight from Indianapolis to Houston on TWA*

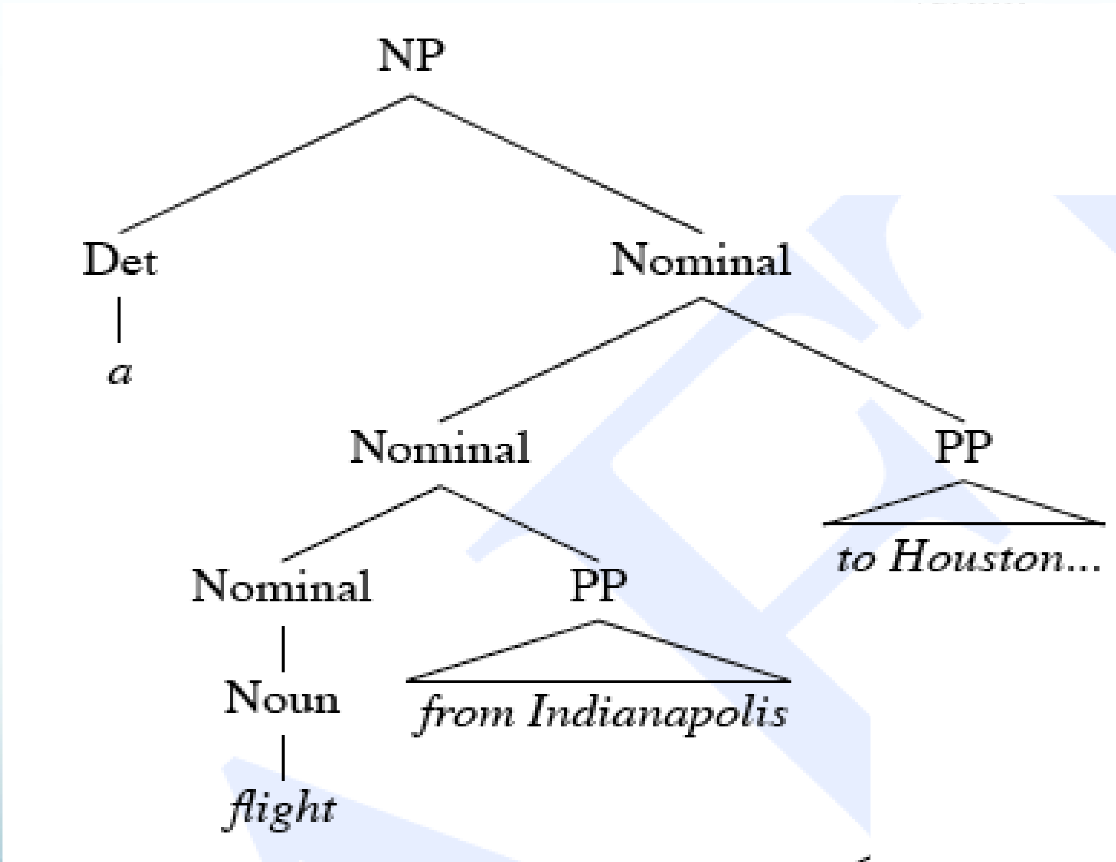
Shared Sub-Problems



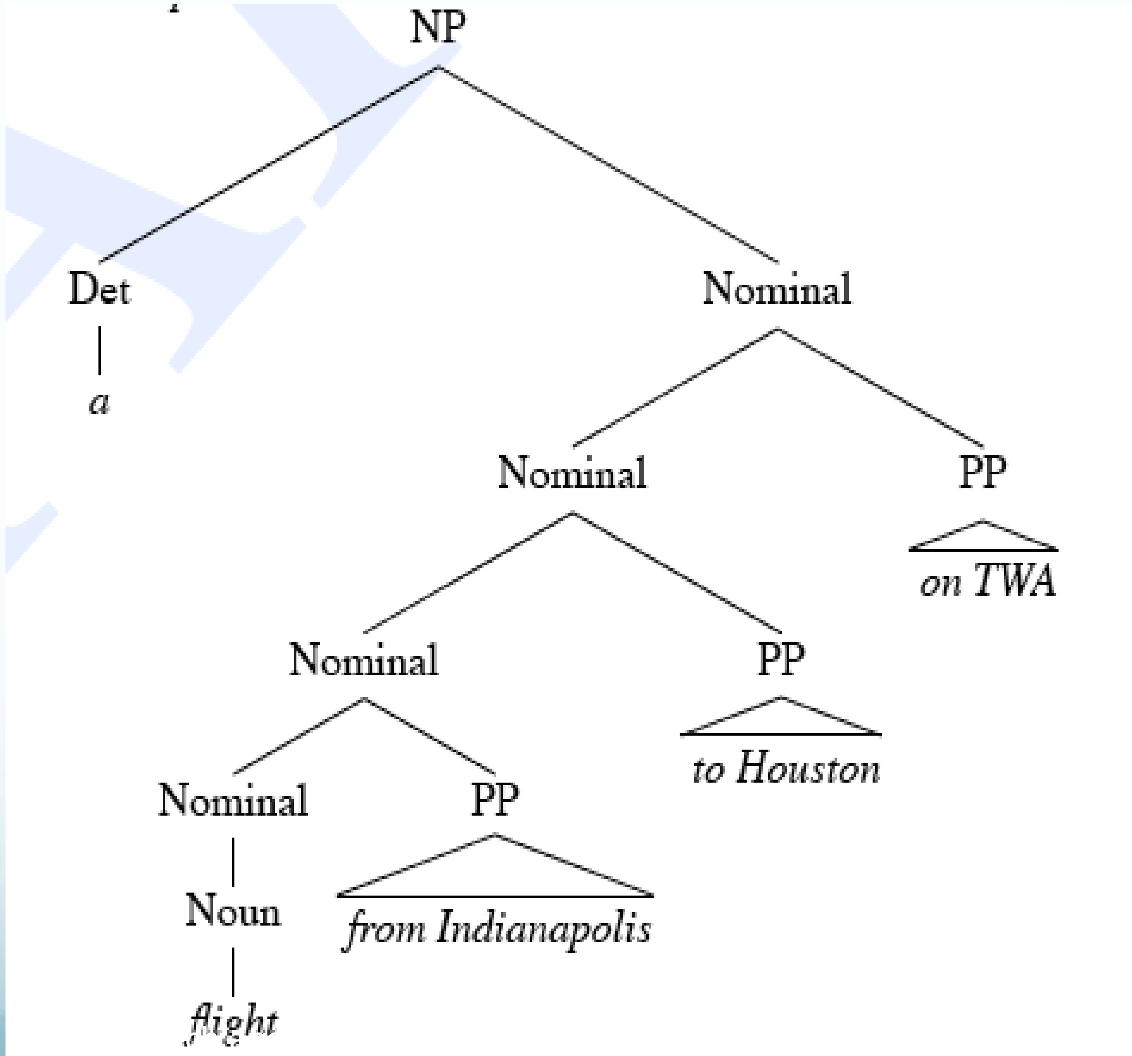
Shared Sub-Problems



Shared Sub-Problems



Shared Sub-Problems



Recursion

- Many grammars have recursive rules
  - E.g., S →→ S Conj S

- In search approaches, recursion is problematic
  - Can yield infinite searches
    - Esp., top-down

Dynamic Programming

- Challenge: Repeated substructure →→ Repeated work
- Insight:
  - Global parse composed of parse substructures
  - Can record parses of substructures
- Dynamic programming avoids repeated work by tabulating solutions to subproblems
  - Here, stores subtrees

Parsing w/Dynamic Programming

- Avoids repeated work
- Allows implementation of (relatively) efficient parsing algorithms
  - Polynomial time in input length
    - Typically cubic (n^3) or less
- Several different implementations
  - Cocke-Kasami-Younger (CKY) algorithm
  - Earley algorithm
  - Chart parsing

Chomsky Normal Form (CNF)

- CKY parsing requires grammars in CNF
- Chomsky Normal Form
  - All productions of the form:
    - A →→ B C, or
    - A →→ a
- However, most of our grammars are not of this form
  - E.g., S →→ Wh-NP Aux NP VP
- Need a general conversion procedure
  - Any arbitrary grammar can be converted to CNF

Grammar Equivalence and Form

- Grammar equivalence
  - Weak: Accept the same language, May produce different analyses
  - Strong: Accept same language, Produce same structure

CNF Conversion

- Three main conditions:
  - Hybrid rules:
    - INF-VP $\rightarrow$ to VP
  - Unit productions:
    - A $\rightarrow$ B
  - Long productions:
    - A $\rightarrow$ B C D

CNF Conversion

- Hybrid rule conversion:
  - Replace all terminals with dummy non-terminals
  - E.g., INF-VP →→ to VP
    - INF-VP →→ TO VP; TO →→ to

- Unit productions:
  - Rewrite RHS with RHS of all derivable non-unit productions
    - If $A \overset{*}{\Rightarrow} B$ and B →→ w, then add A →→ w

CNF Conversion

- Long productions:
  - Introduce new non-terminals and spread over rules
  - S →→ Aux NP VP
    - S →→ X1 VP; X1 →→ Aux NP

- For all non-conforming rules,
  - Convert terminals to dummy non-terminals
  - Convert unit productions
  - Binarize all resulting rules

2 $_1$ **Grammar**	S1i **in CNF**
S --+ lP VP	*S --+ l\TP VP*
S --+ Aux NP VP	*S--+ Xl VP*
	X1 --+ Auxi\TP
S --+ VP	*S --+ book* I *include* I *preje r*
	S--+ VerbNP
	S--+ X2PP
	S --+Verb PP
	S--+ VPPP
NP --+ Pronoun	*NP --+I* I *she* I *rne*
l\TP --+ Proper-Noun	*i\TP --+ TWA* I *Houston*
l\TP --+ D e t Nominal	*i\TP --+ Det l\Tominal*
Norninal --+ Noun	*Nominal --+ book* I *flight* I *meal* I *money*
Norninal --+ Nominal Noun	*Nominal --+ Nominal Noun*
l\Torninal --+ l\To m i n a l PP	*i\Tominal --+ Nominal PP*
VP --+ Verb	*VP --+ book* I *include* I *prefer*
VP --+ Verb NP	*VP --+ Verb NP*
VP --+ Verb NP PP	*VP--+ X2 PP*
	X2 --+ Verb NP
VP --+ Verb PP	*VP --+ Verb PP*
VP--+ VPPP	*VP--+ VPPP*
PP --+ Preposition i\TP	*PP --+ Preposition l\TP*

CKY Parsing

- Cocke-Kasami-Younger parsing algorithm:
  - (Relatively) efficient bottom-up parsing algorithm based on tabulating substring parses to avoid repeated work
  - Approach:
    - Use a CNF grammar
    - Build an (n+1) x (n+1) matrix to store subtrees
      - Upper triangular portion
    - Incrementally build parse spanning whole input string

Dynamic Programming in CKY

- Key idea:
  - For a parse spanning substring [i,j] , there exists some k such there are parses spanning [i,k] and [k,j]
    - We can construct parses for whole sentence by building up from these stored partial parses
- So,
  - To have a rule A →→ B C in [i,j],
    - We must have B in [i,k] and C in [k,j], for some i<k<j
      - CNF grammar forces this for all j>i+1