CS344: Introduction to Artificial Intelligence

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

Lecture 20-21– Natural Language Parsing

Parsing of Sentences

Are sentences flat linear structures? Why tree?

- Is there a principle in branching
- When should the constituent give rise to children?
- What is the hierarchy building principle?

Structure Dependency: A Case Study

- **Interrogative Inversion**

(1) John will solve the problem.

Will John solve the problem?

	Declarative	*Interrogative*
(2)	a. Susan must leave.	Must Susan leave?
	b. Harry can swim.	Can Harry swim?
	c. Mary has read the book.	Has Mary read the book?
	d. Bill is sleeping.	Is Bill sleeping?

…………………………………………………………….

*The section, "**Structure dependency a case study**" here is adopted from a talk given by Howard Lasnik (2003) in Delhi university.*

Interrogative inversion
Structure Independent (1st attempt)

(3) **Interrogative inversion process**

Beginning with a declarative, invert the first and second words to construct an interrogative.

	Declarative	Interrogative
(4)	a. The woman must leave.	*Woman the must leave?
	b. A sailor can swim.	*Sailor a can swim?
	c. No boy has read the book.	*Boy no has read the book?
	d. My friend is sleeping.	*Friend my is sleeping?

Interrogative inversion
correct pairings

- Compare the incorrect pairings in (4) with the correct pairings in (5):

	Declarative	Interrogative
(5)	a. The woman must leave.	Must the woman leave?
	b. A sailor can swim.	Can a sailor swim?
	c. No boy has read the book.	Has no boy read the book?
	d. My friend is sleeping.	Is my friend sleeping?

Interrogative inversion
Structure Independent (2nd attempt)

(6) Interrogative inversion process:

- Beginning with a declarative, move the auxiliary verb to the front to construct an interrogative.

Declarative	Interrogative
(7) a. Bill could be sleeping.	*Be Bill could sleeping?
	Could Bill be sleeping?
b. Mary has been reading.	*Been Mary has reading?
	Has Mary been reading?
c. Susan should have left.	*Have Susan should left?
	Should Susan have left?

Structure independent (3rd attempt):

(8) Interrogative inversion process

- Beginning with a declarative, move the first auxiliary verb to the front to construct an interrogative.

	Declarative	Interrogative
(9)	a. The man who is here can swim.	*Is the man who here can swim?
	b. The boy who will play has left.	*Will the boy who play has left?

Structure Dependent Correct Pairings

- For the above examples, fronting the second auxiliary verb gives the correct form:

	Declarative	Interrogative
(10)	a. The man who is here can swim.	Can the man who is here swim?
	b. The boy who will play has left.	Has the boy who will play left?

Natural transformations are structure dependent

(11) Does the child acquiring English learn these properties?

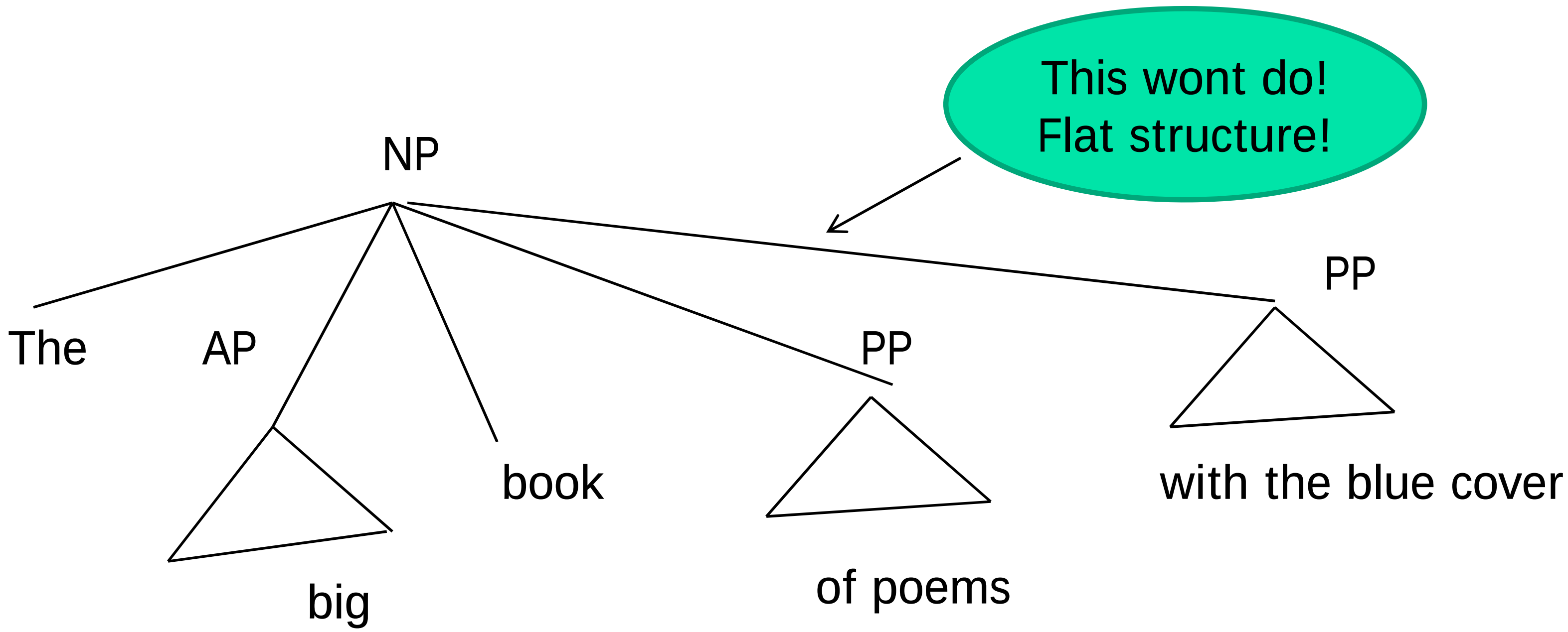
(12) We are not dealing with a peculiarity of English. No known human language has a transformational process that would produce pairings like those in (4), (7) and (9), repeated below:

(4) a. The woman must leave. *Woman the must leave?

(7) a. Bill could be sleeping. *Be Bill could sleeping?

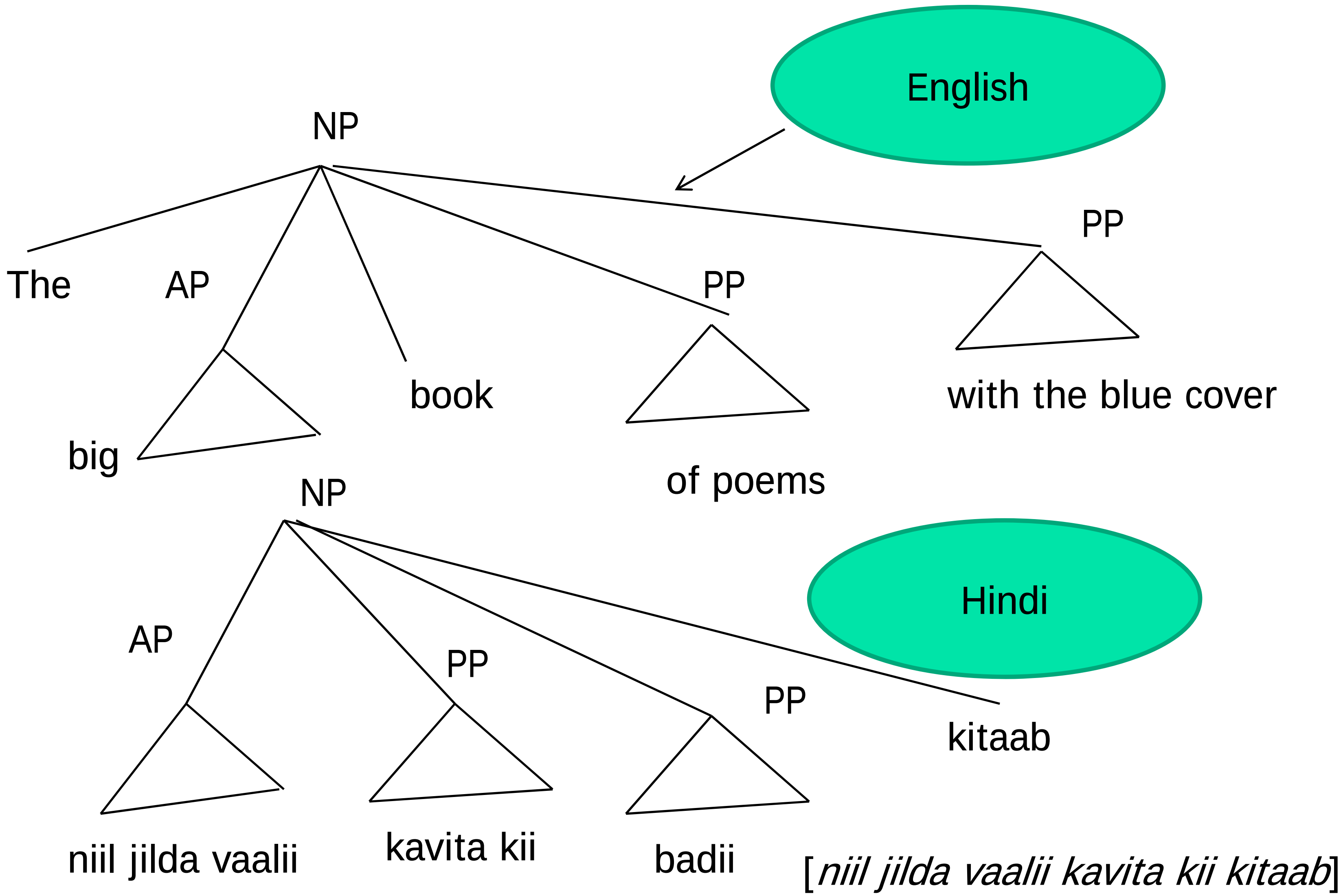
(9) a. The man who is here can swim. *Is the man who here can swim?

Deeper trees needed for capturing sentence structure

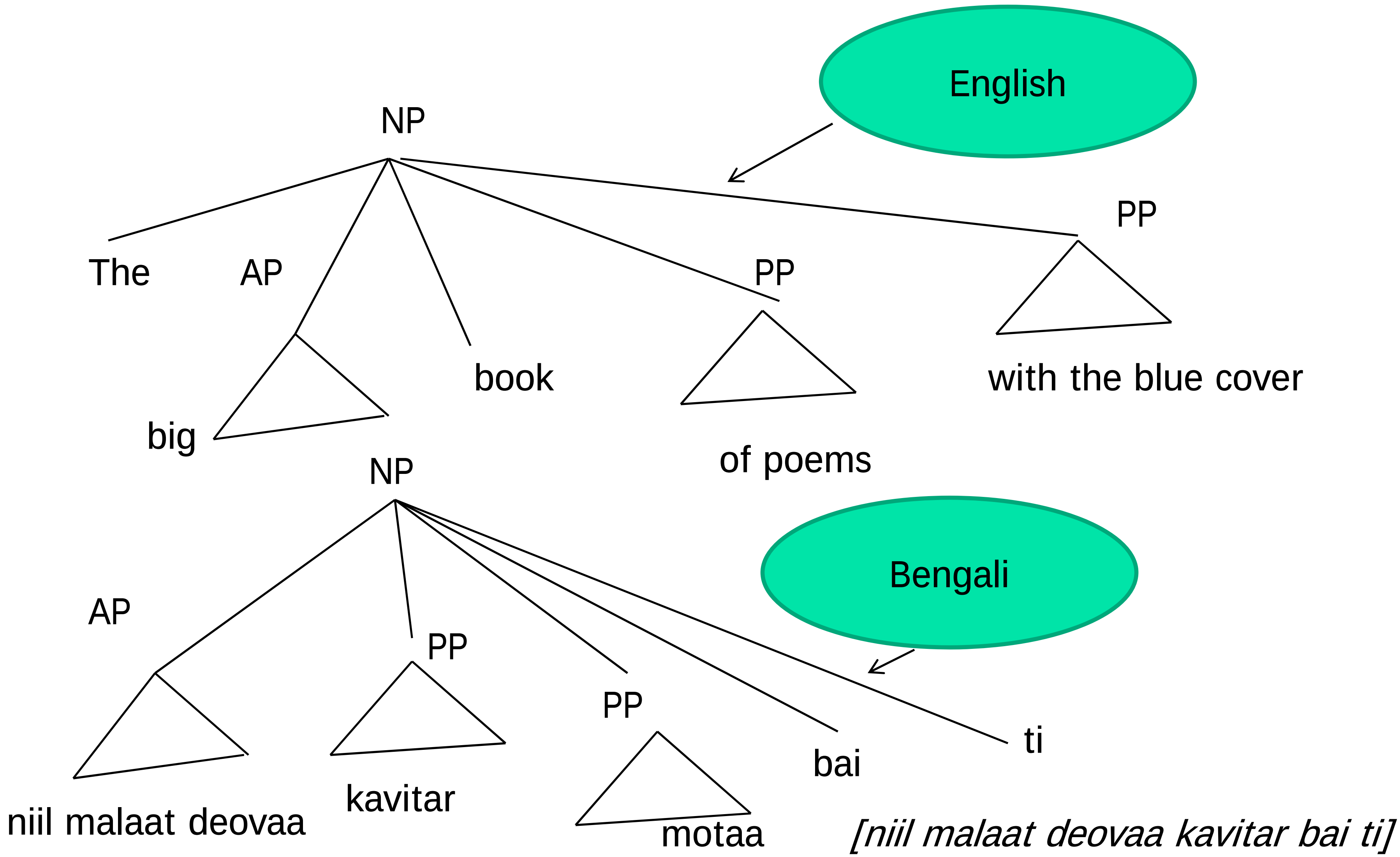


[*The big book of poems with the Blue cover*] is on the table.

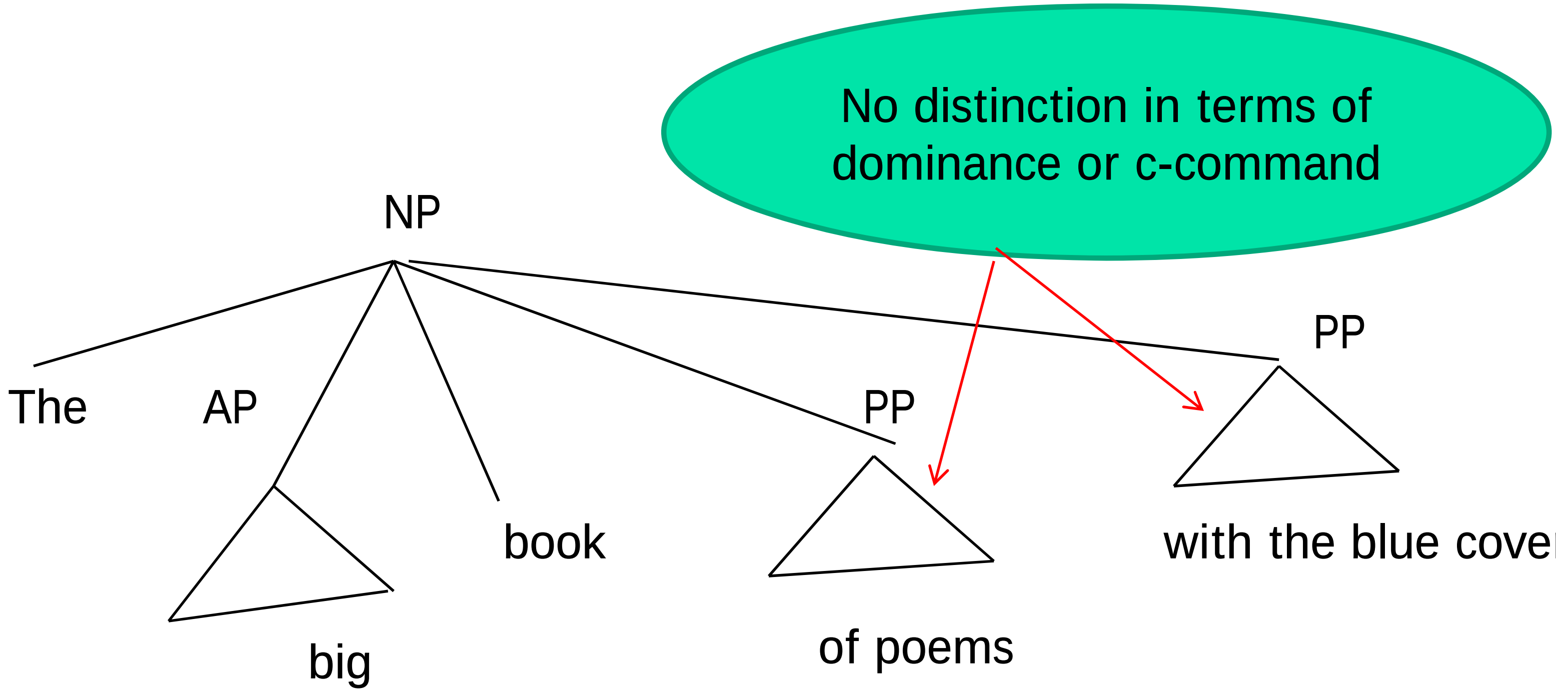
Other languages



Other languages: contd



PPs are at the same level: *flat with respect to the head word "book"*

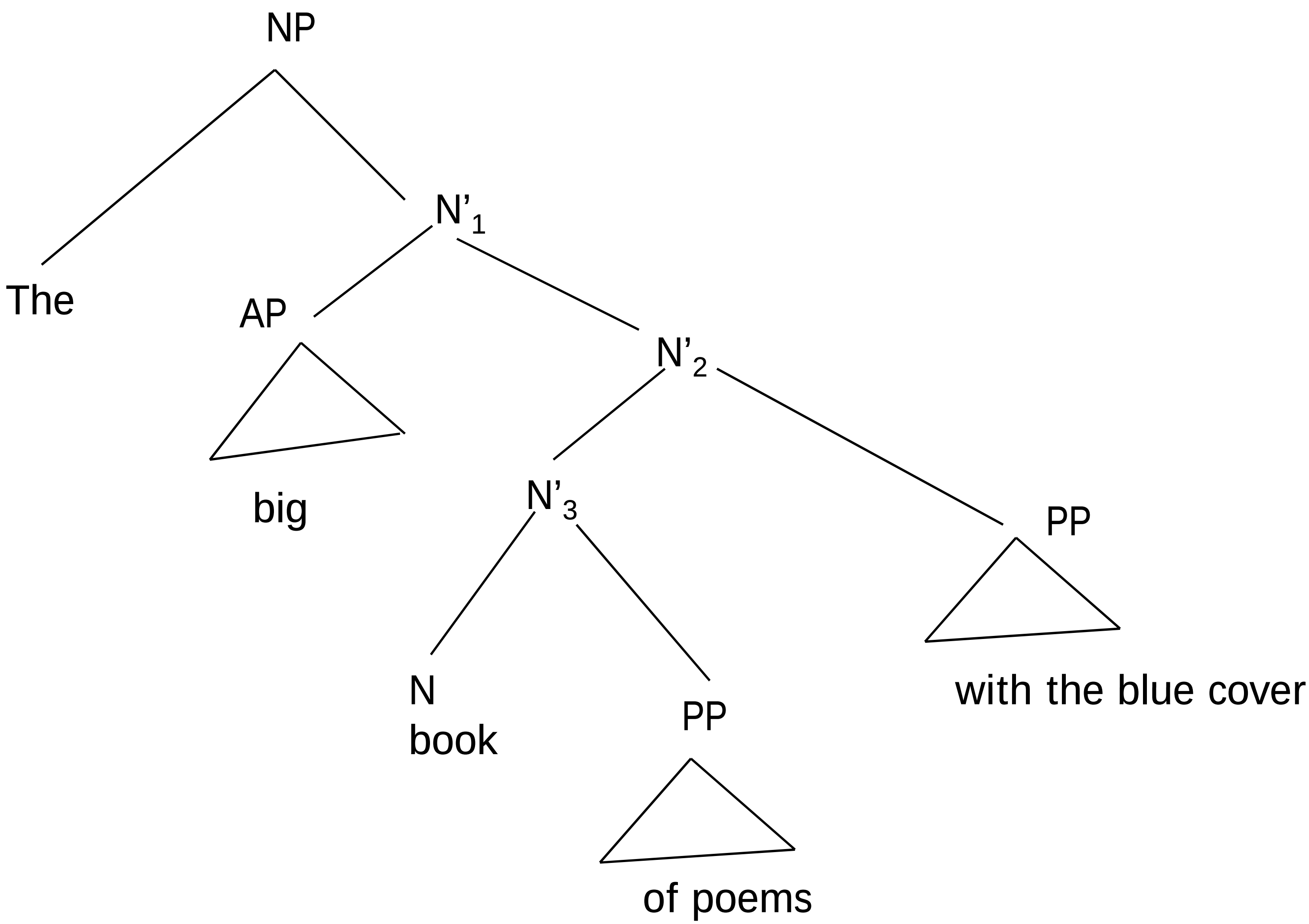


[*The big book of poems with the Blue cover*] is on the table.

“Constituency test of Replacement” runs into problems

- One-replacement:
  - *I bought the big [book of poems with the blue cover] not the small [one]*
  - *One-replacement* targets *book of poems with the blue cover*
- Another one-replacement:
  - *I bought the big [book of poems] with the blue cover not the small [one] with the red cover*
  - *One-replacement* targets *book of poems*

More deeply embedded structure



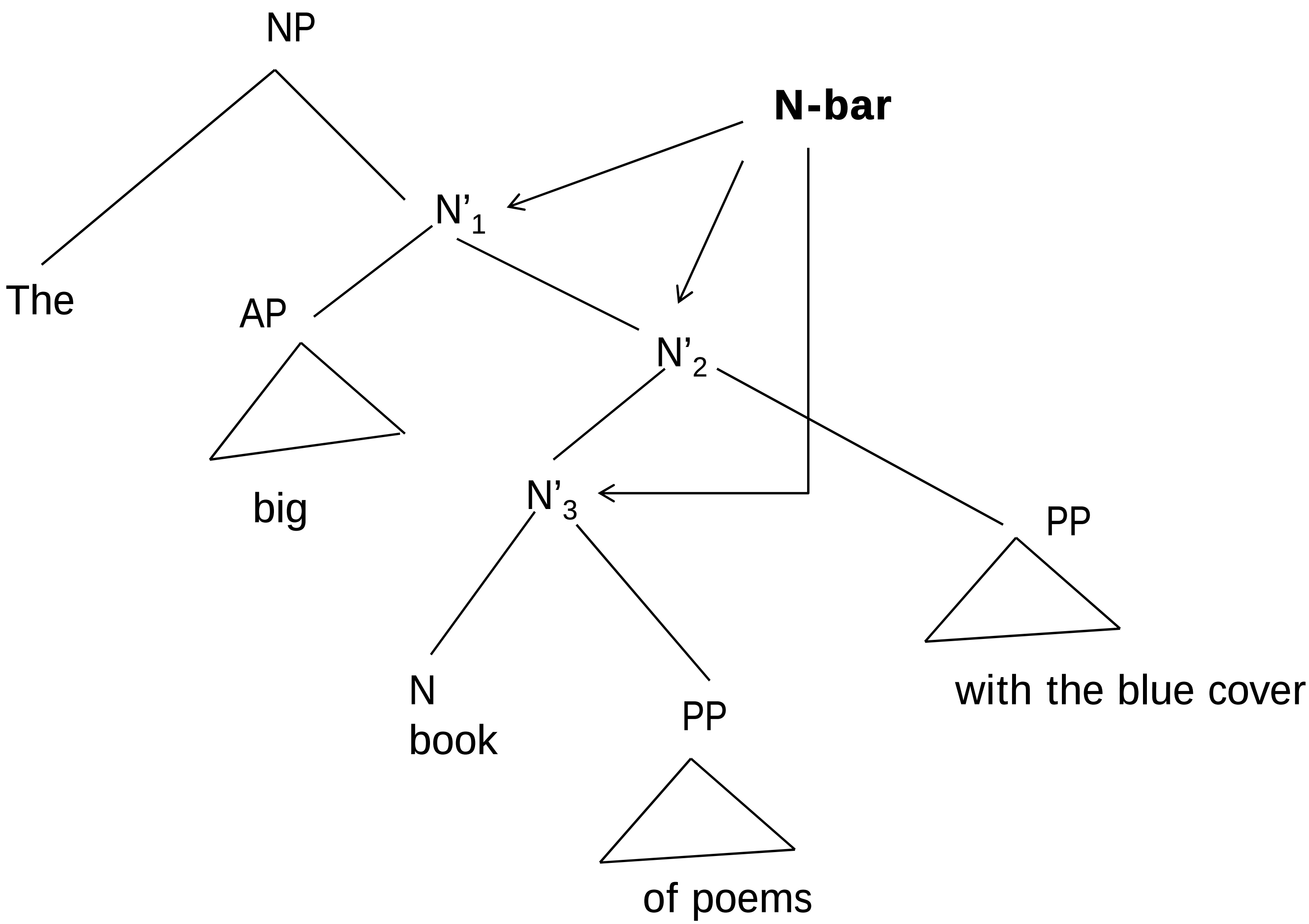
To target N_1'

- I want [$_{NP}$this [$_{N'}$big book of poems with the red cover] and not [$_{N}$that [$_{N}$one]]

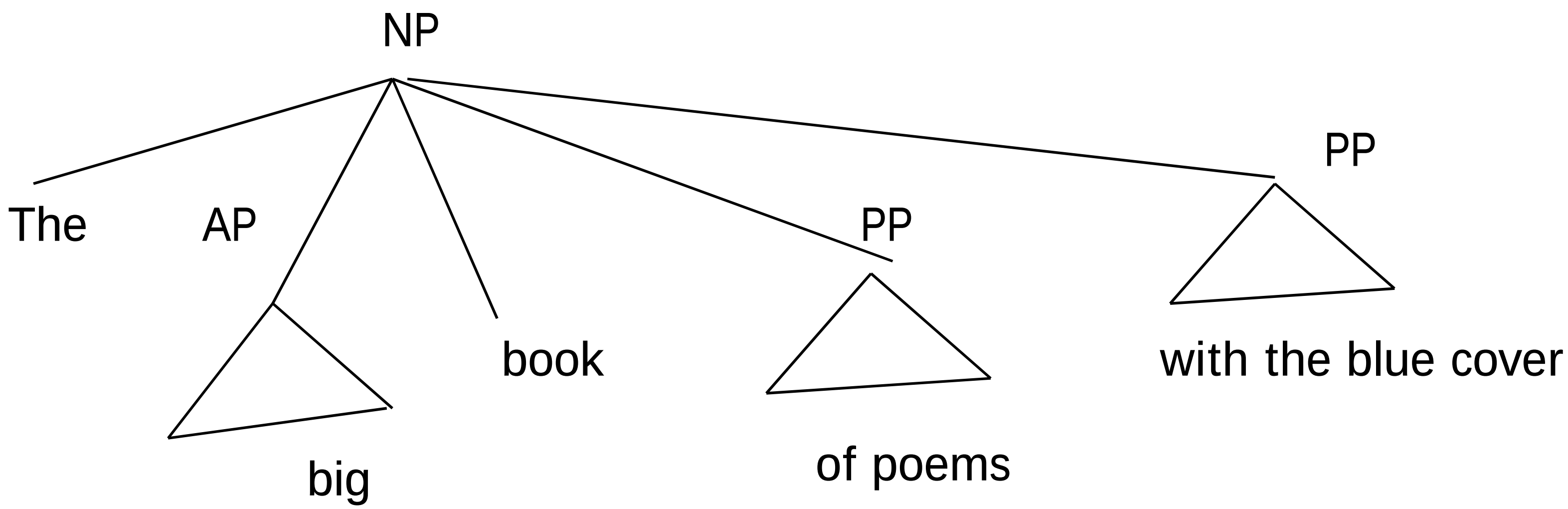
Bar-level projections

- Add intermediate structures
  - NP→ (D) N’
  - N’→ (AP) N’ | N’ (PP) | N (PP)
- () indicates optionality

New rules produce this tree



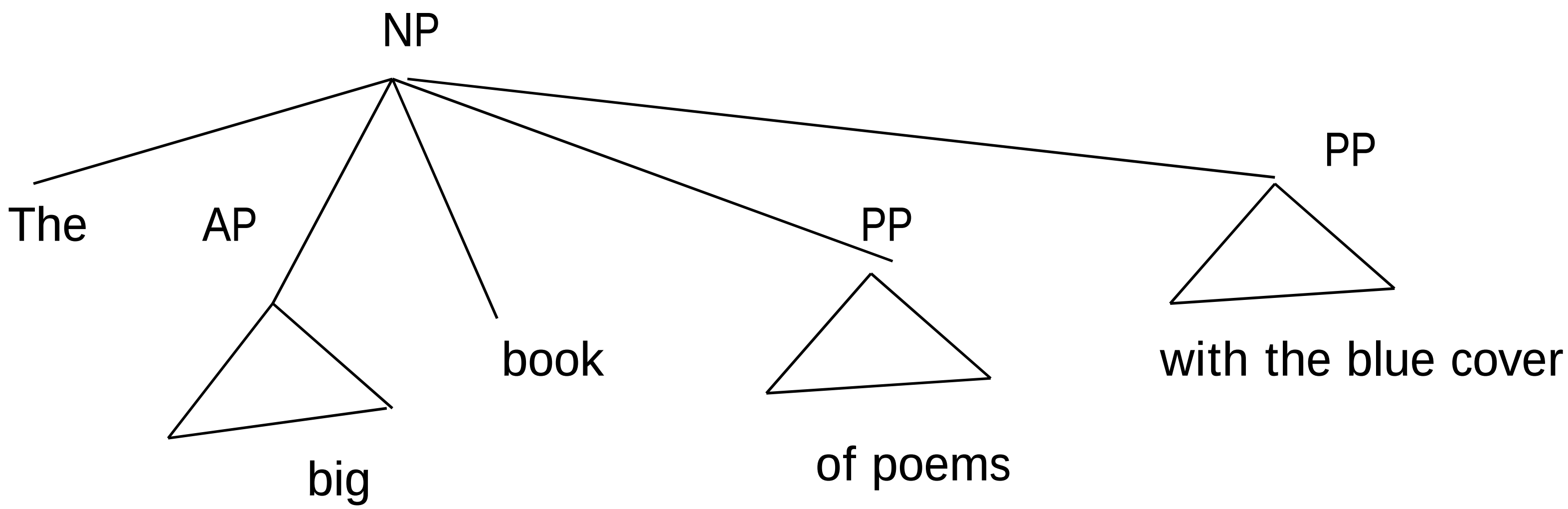
As opposed to this tree



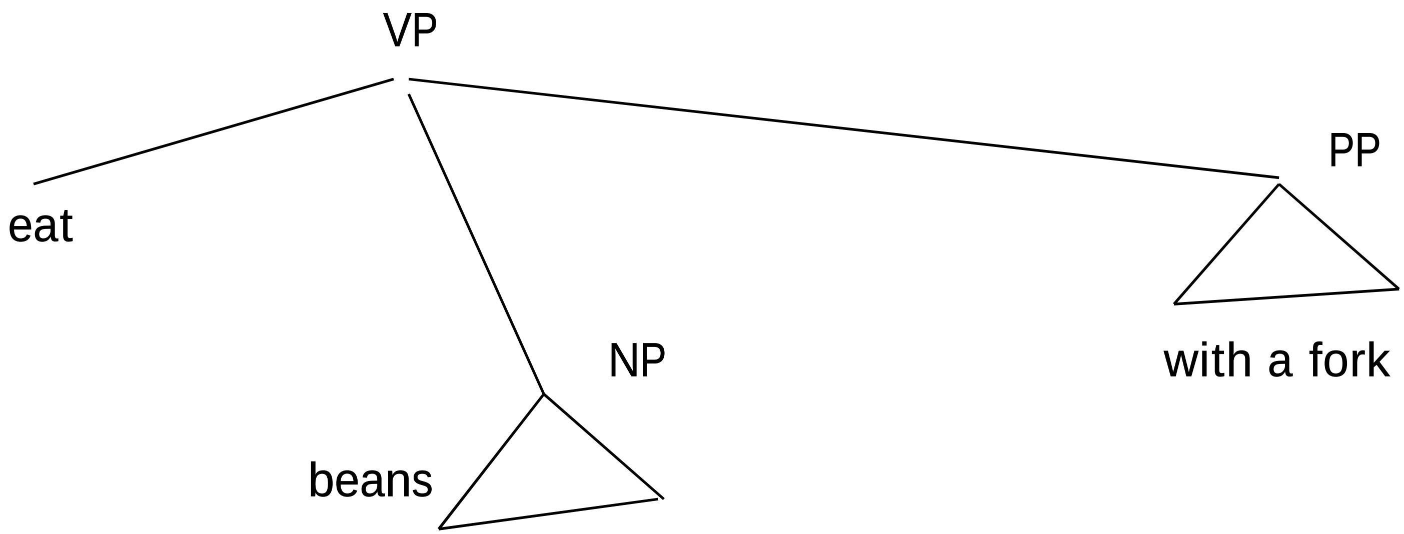
V-bar

- What is the element in verbs corresponding to *one-replacement* for nouns
- *do-so* or *did-so*

As opposed to this tree



I [eat beans with a fork]

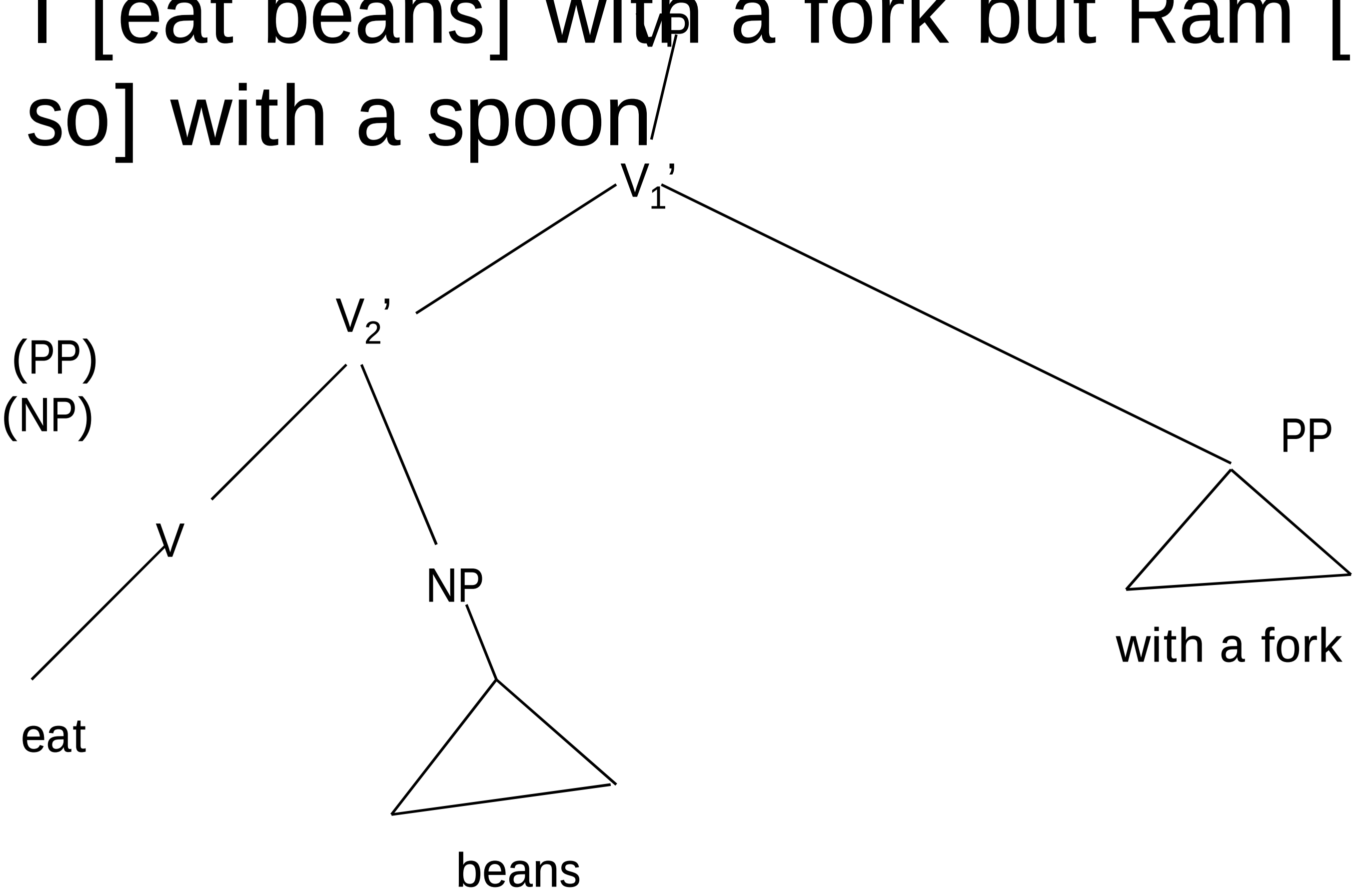


No constituent that groups together V and NP and excludes PP

Need for intermediate constituents

- I [eat beans] with a fork but Ram [does so] with a spoon

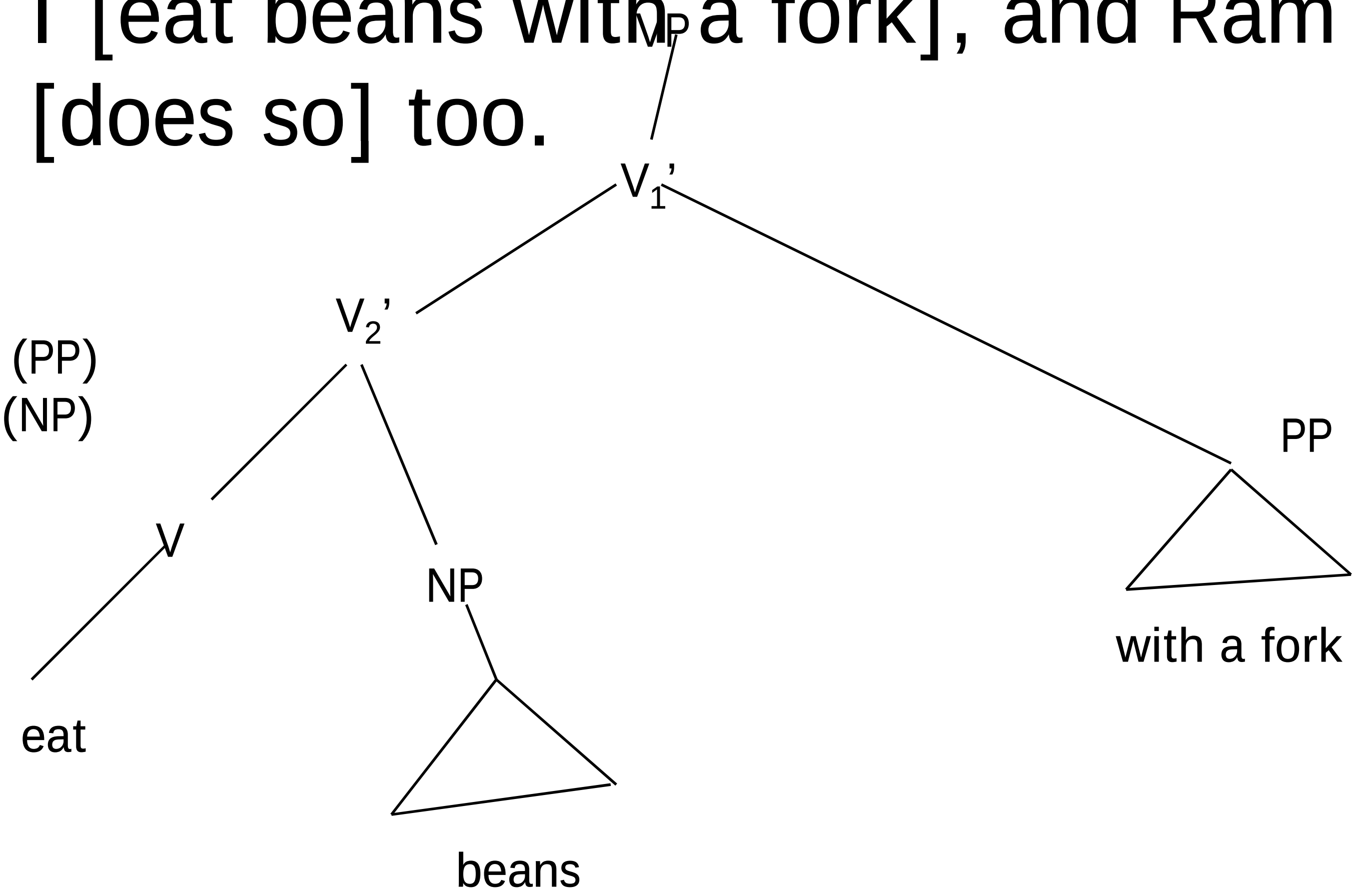
VP→V'
V'→ V' (PP)
V'→ V (NP)



How to target V_1'

- I [eat beans with a fork], and Ram [does so] too.

VP→V'
V'→ V' (PP)
V'→ V (NP)



Parsing Algorithms

A simplified grammar

- $S \rightarrow NP\ VP$
- $NP \rightarrow DT\ N \mid N$
- $VP \rightarrow V\ ADV \mid V$

A segment of English Grammar

- S'→(C) S
- S→{NP/S'} VP
- VP→(AP+) (VAUX) V (AP+) ({NP/S'}) (AP+) (PP+) (AP+)
- NP→(D) (AP+) N (PP+)
- PP→P NP
- AP→(AP) A

Example Sentence

People laugh

1 2 3

These are positions

Lexicon:

People - N, V

Laugh - N, V

This indicate that both Noun and Verb is possible for the word "People"

Top-Down Parsing

State	Backup State	Action
1. ((S) 1)	-	-
2. ((NP VP)1)	-	-
3a. ((DT N VP)1)	((N VP) 1)	-
3b. ((N VP)1)	-	-
4. ((VP)2)	-	Consume "People"
5a. ((V ADV)2)	((V)2)	-
6. ((ADV)3)	((V)2)	Consume "laugh"
5b. ((V)2)	-	-
6. ((.)3)	-	Consume "laugh"

Position of input pointer

Termination Condition : All inputs over. No symbols remaining.

Note: Input symbols can be pushed back.

Discussion for Top-Down Parsing

- This kind of searching is goal driven.
- Gives importance to textual precedence (rule precedence).
- No regard for data, a priori (useless expansions made).

Bottom-Up Parsing

Some conventions:

N_{12}

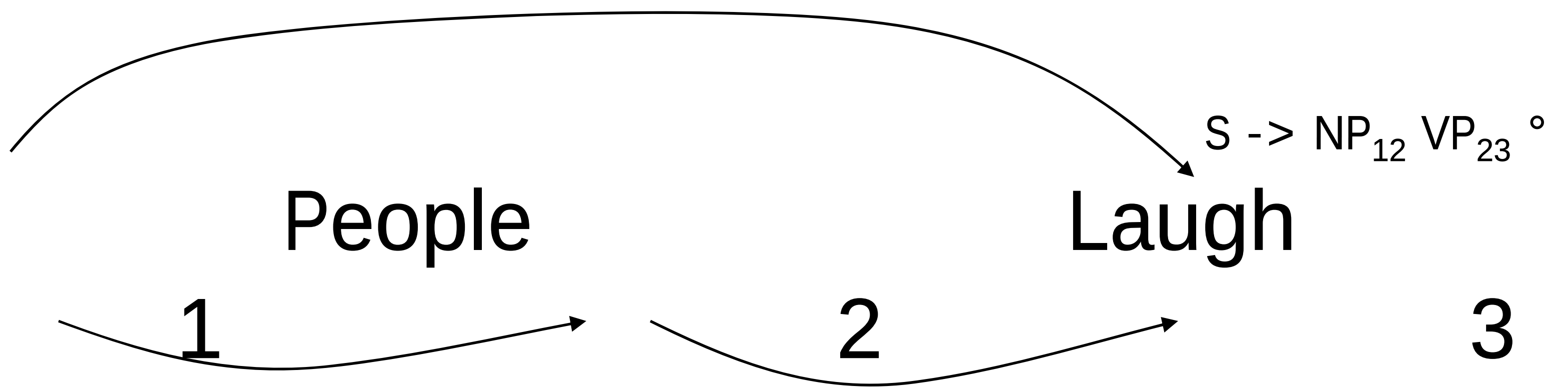
Represents positions

$S_{1?}$ -> $NP_{12} \circ VP_{2?}$

End position unknown

Work on the LHS done, while the work on RHS remaining

Bottom-Up Parsing (pictorial representation)



N_{12}

V_{12}

NP_{12} -> N_{12} °

VP_{12} -> V_{12} °

$S_{1?}$ -> NP_{12} ° $VP_{2?}$

N_{23}

V_{23}

NP_{23} -> N_{23} °

VP_{23} -> V_{23} °

Problem with Top-Down Parsing

- Left Recursion
  - Suppose you have A-> AB rule.

    Then we will have the expansion as follows:
    - ((A)K) -> ((AB)K) -> ((ABB)K) ....

Combining top-down and bottom-up strategies

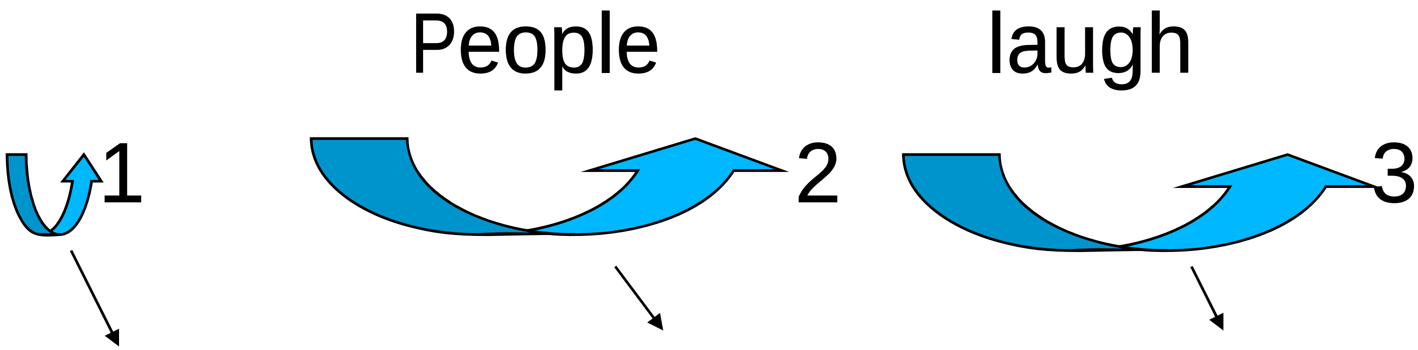
Top-Down Bottom-Up Chart Parsing

- Combines advantages of top-down & bottom-up parsing.
- Does not work in case of left recursion.
  - *e.g.* – "People laugh"
    - People – noun, verb
    - Laugh – noun, verb
  - Grammar – $S \rightarrow NP\ VP$

    $NP \rightarrow DT\ N \mid N$

    $VP \rightarrow V\ ADV \mid V$

Transitive Closure



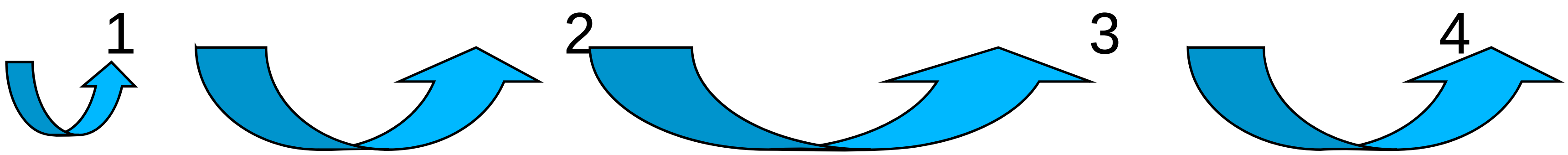
$S \rightarrow \bullet NP\ VP$	$NP \rightarrow N \bullet$	$VP \rightarrow V \bullet$
$NP \rightarrow \bullet DT\ N$	$S \rightarrow NP \bullet VP$	$S \rightarrow NP\ VP \bullet$
$NP \rightarrow \bullet N$	$VP \rightarrow \bullet V\ ADV$	success
	$VP \rightarrow \bullet V$	

Arcs in Parsing

- Each arc represents a chart which records
  - Completed work (left of •)
  - Expected work (right of •)

Example

People laugh loudly



S → • NP VP
NP → • DT N
NP → • N

NP → N•
S → NP•VP
VP → •V ADV
VP → •V

VP → V•
VP → V•ADV
S → NP VP•

VP → V ADV•
S → NP VP•

Dealing With Structural Ambiguity

- Multiple parses for a sentence
  - The man saw the boy with a telescope.
  - The man saw the mountain with a telescope.
  - The man saw the boy with the ponytail.

At the level of syntax, all these sentences are ambiguous. But semantics can disambiguate 2^{nd} & 3^{rd} sentence.

Prepositional Phrase (PP) Attachment Problem

$V – NP_1 – P – NP_2$

(Here P means preposition)

NP_2 attaches to NP_1 ?

or NP_2 attaches to V ?

Parse Trees for a Structurally Ambiguous Sentence

Let the grammar be –

S $\rightarrow$ NP VP

NP $\rightarrow$ DT N | DT N PP

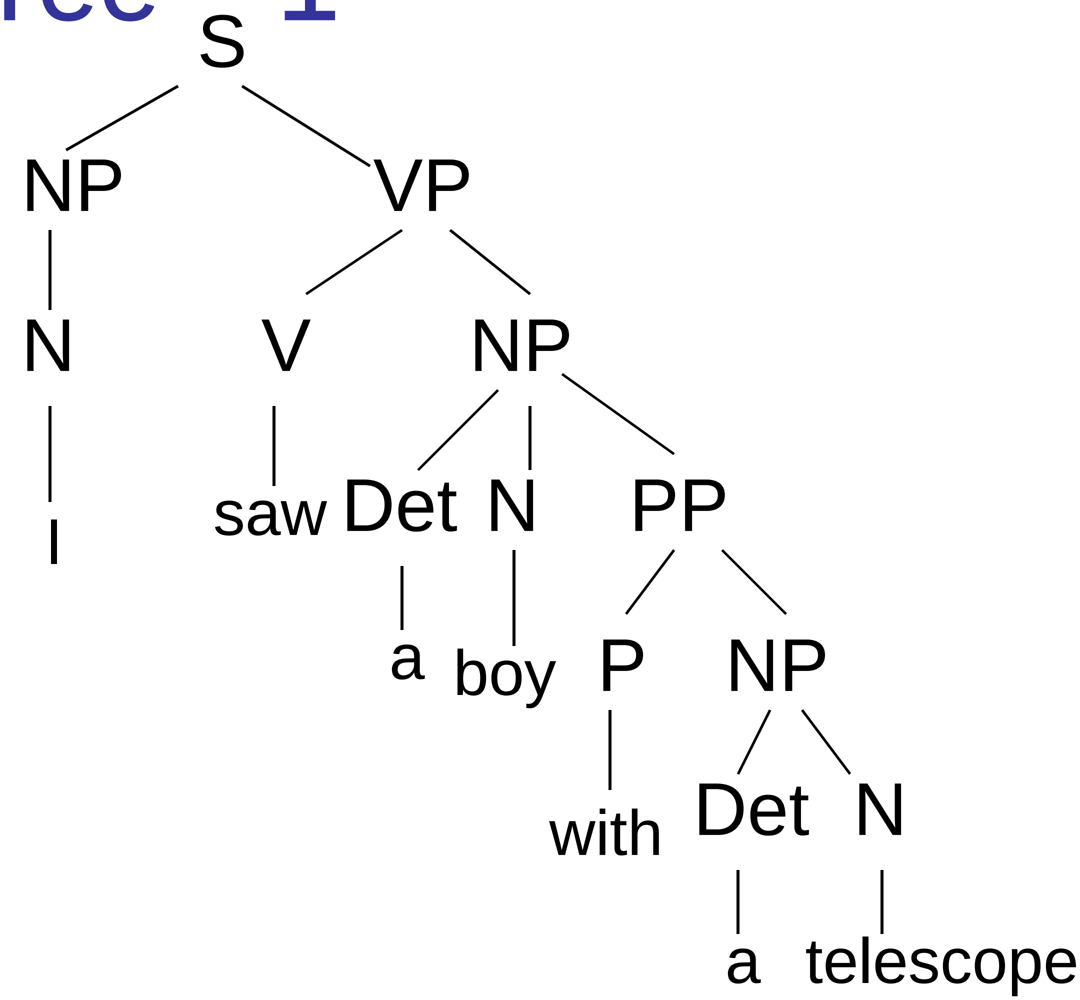
PP $\rightarrow$ P NP

VP $\rightarrow$ V NP PP | V NP

For the sentence,

“I saw a boy with a telescope”

Parse Tree - 1



Parse Tree -2