

CS626: Speech, NLP and Web

Parsing start

Pushpak Bhattacharyya

Computer Science and Engineering
Department

IIT Bombay

Week 5 of 26th August, 2024

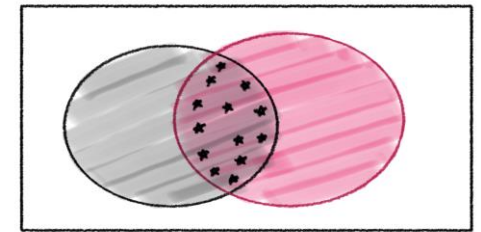
(Thursday, quiz day)

1-slide recap of week of 19th Aug

- Discriminative POS tagging with Beam Search
- CRF and CRF based POS tagging

$$P(Y | X) = \frac{1}{Z(X)} \exp \left(\sum_c \varphi_c(Y_c, X_c) \right)$$

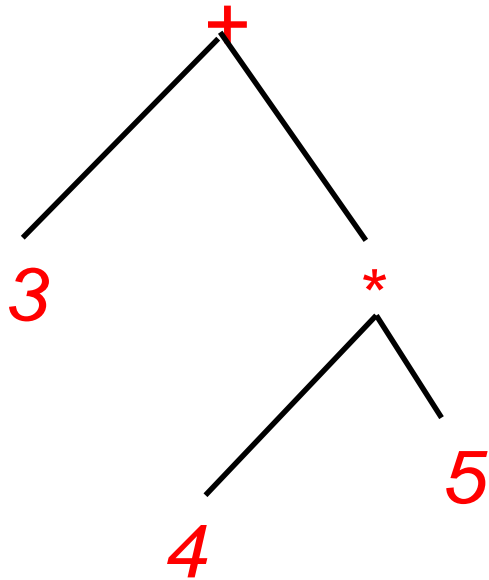
- Penn Tagset
- Evaluation metrics- P , R , F
- Inevitability of probabilistic POS tagging
- Assignment on POS- HMM, CRF, Benchmarking against ChatGPT



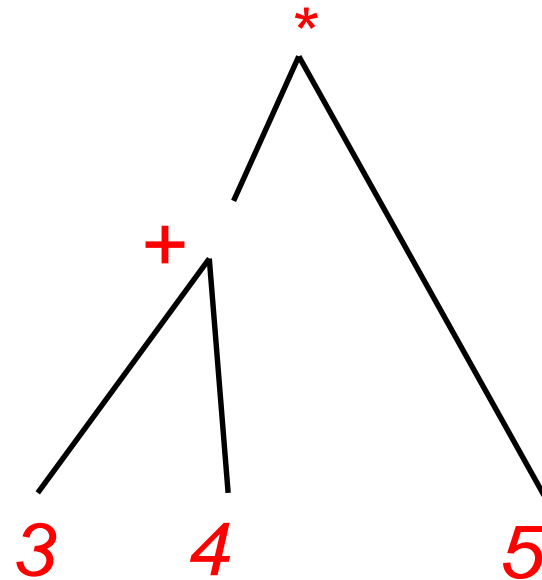
■ SET S_1 ... OBTAINED
■ SET S_2 ... ACTUAL
■ $S_1 \cap S_2$... TRUE POSITIVES
 $S_1 - (S_1 \cap S_2)$... FALSE POSITIVES
 $S_2 - (S_1 \cap S_2)$... FALSE NEGATIVES
 $(S_1 \cup S_2)^c$... TRUE NEGATIVES

Evidence of Deep Structure

Trees frequently used: Arithmetic Expressions



$$3+4*5=23$$



$$3+4*5=35$$

Trees are devices to represent constituents and their interactions to form bigger constituents

Strong evidence of existence of structure is *structural ambiguity*

- Parsing of “Unlockable”

- *Un + lockable*

- *lock + able*

- Meaning: something that cannot be locked
(*this gate is unlockable- open and cannot be locked*)



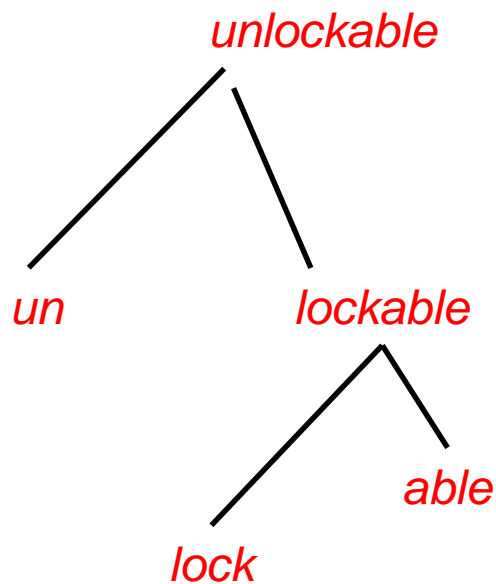
- *Unlock + able*

- *un + lock*

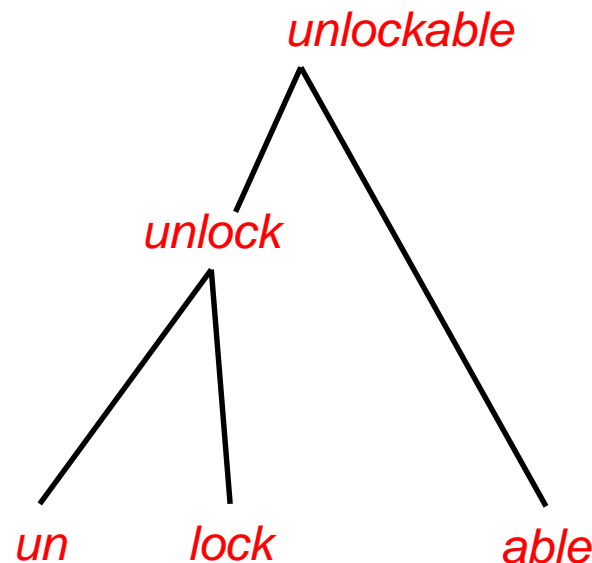
- Meaning: something that can be unlocked (*this gate is unlockable- shut with a lock and can be unlocked*)



Tree Structure: Morphotactics of “Unlockable”: two structures



something that cannot be locked
(“*this gate is unlockable*”- open
and cannot be locked)

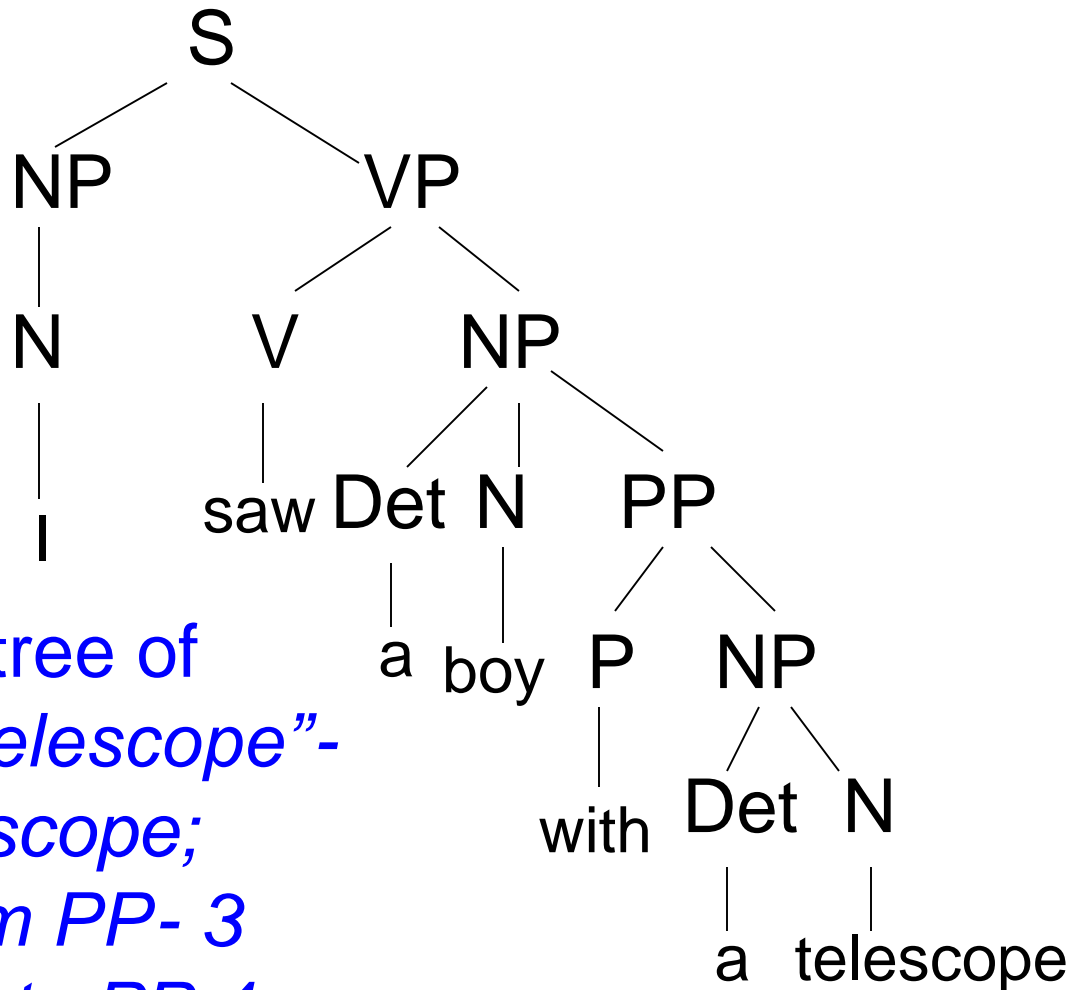


Something that can be unlocked (“*this gate is unlockable*”- shut with a lock,
but can be unlocked)



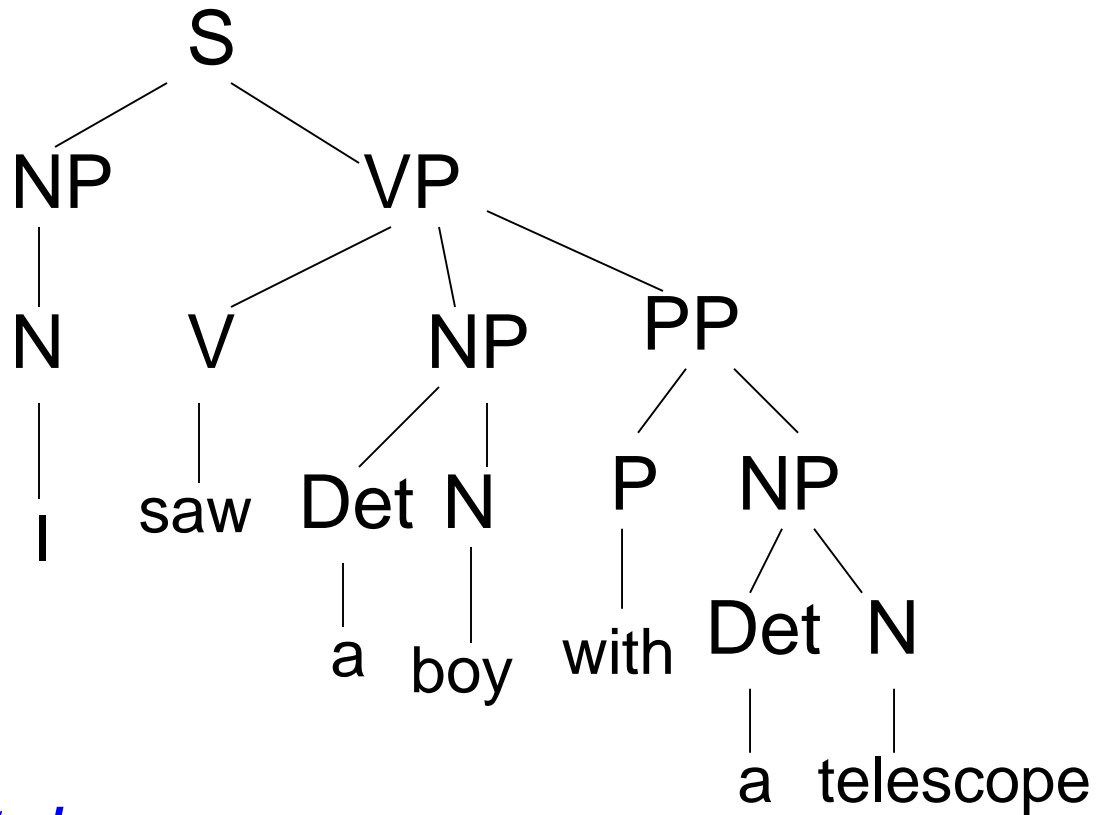
shutterstock.com - 256794006

What is the proof that there is underlying structure? **Structural Ambiguity**



Constituency parse tree of
"I saw a boy with a telescope"-
the boy has the telescope;
distance of 'boy' from PP- 3
arcs, that from 'saw' to PP 4
arcs

Constituency Parse Tree -2



I saw a boy with a telescope-

I have the telescope; distance of 'boy' from PP- 4 arcs, that from 'saw' to PP 3 arcs

SANNIDHI Principle (proximity)

What does POS tagging
Facilitate

Facilitates Chunking: small phrases called **Chunks**

given the sentence

The brown fox sat in front of the fence

POS tagged sequence as

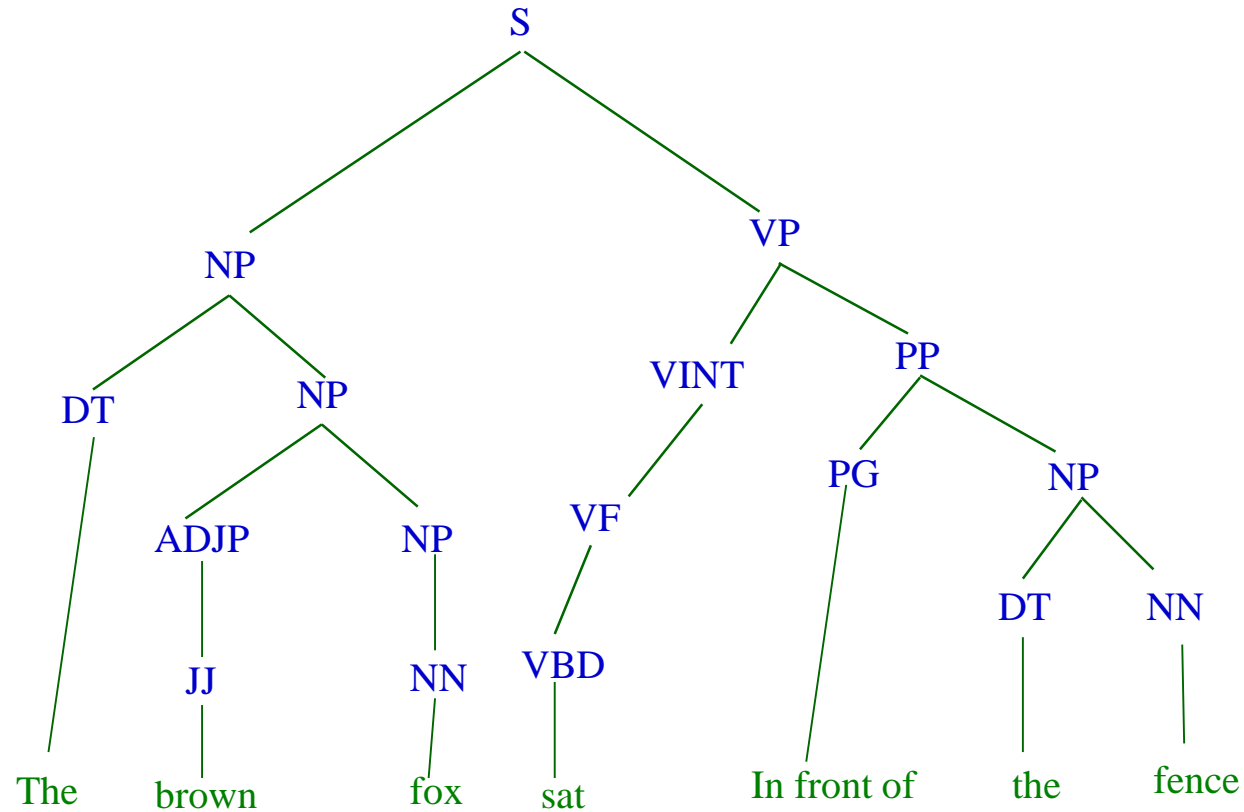
*The_DT brown_JJ fox_NN sat_VBD in_IN
front_NN of_IN the_DT fence_NN*

Chunked sequence as

*The_DT_B_{NC} brown_JJ_I_{NC} fox_NN_I_{NC} sat_VBD_B_{VC}
in_IN_B_{PC} front_NN_I_{PC} of_IN_I_{PC} the_DT_B_{NC}
fence_NN_I_{NC}*

B indicates 'beginning', I indicates 'inside'; NC indicates chunk type- noun chunk

Deep Parse Tree of *the brown fox sat in front of the fence*



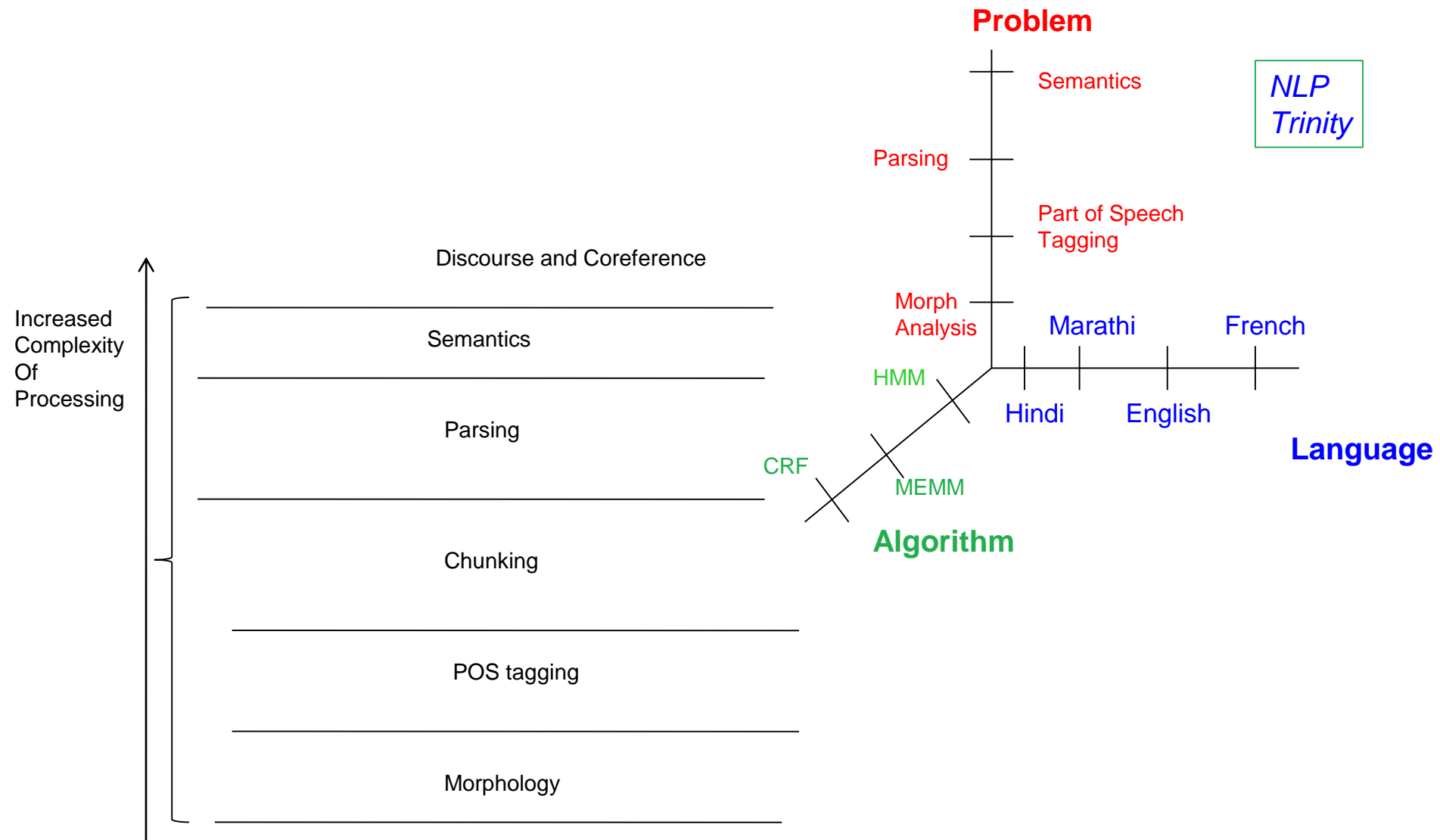
Correct parse can come only if the POS tag sequence is correct; similarly for chunks

Grammar rules

- $S \rightarrow NP VP$
- $NP \rightarrow DT NP \mid ADJP NP \mid PP NP \mid NNS \mid NN$
- $ADJP \rightarrow ADJP JJ \mid JJ$
- $PP \rightarrow PG NP \mid P NP$
- $PG \rightarrow \text{'in front of'} \mid \text{'in lieu of'} \mid \text{'with respect to'} \mid \dots$
- $P \rightarrow \text{'in'} \mid \text{'with'} \mid \text{'by'} \mid \dots$
- $NN \rightarrow \text{'fox'} \mid \text{'fence'} \mid \dots$
- $JJ \rightarrow \text{'brown'} \mid \dots$
- $DT \rightarrow \text{'a'} \mid \text{'an'} \mid \text{'the'} \mid \dots$
- $VP \rightarrow VT NP \mid VINT PP$
- $VT \rightarrow VXG VF \mid VF$
- $VINT \rightarrow VXG VF \mid VF$
- $VXG \rightarrow VXG VX \mid VX$
- $VF \rightarrow VB \mid VBD \mid \dots$
- $VX \rightarrow \text{'am'} \mid \text{'is'} \mid \text{'shall'} \mid \dots$
- $VB \rightarrow \text{'go'} \mid \text{'see'} \mid \dots$
- $VBD \rightarrow \text{'sat'} \mid \text{'went'} \mid \dots$
- $NN \rightarrow \text{'fox'} \mid \text{'fence'} \mid \dots$

A sentence valid according to this grammar is *the fox sat in front of the fence*

' \rightarrow ' means 'is constituted of'

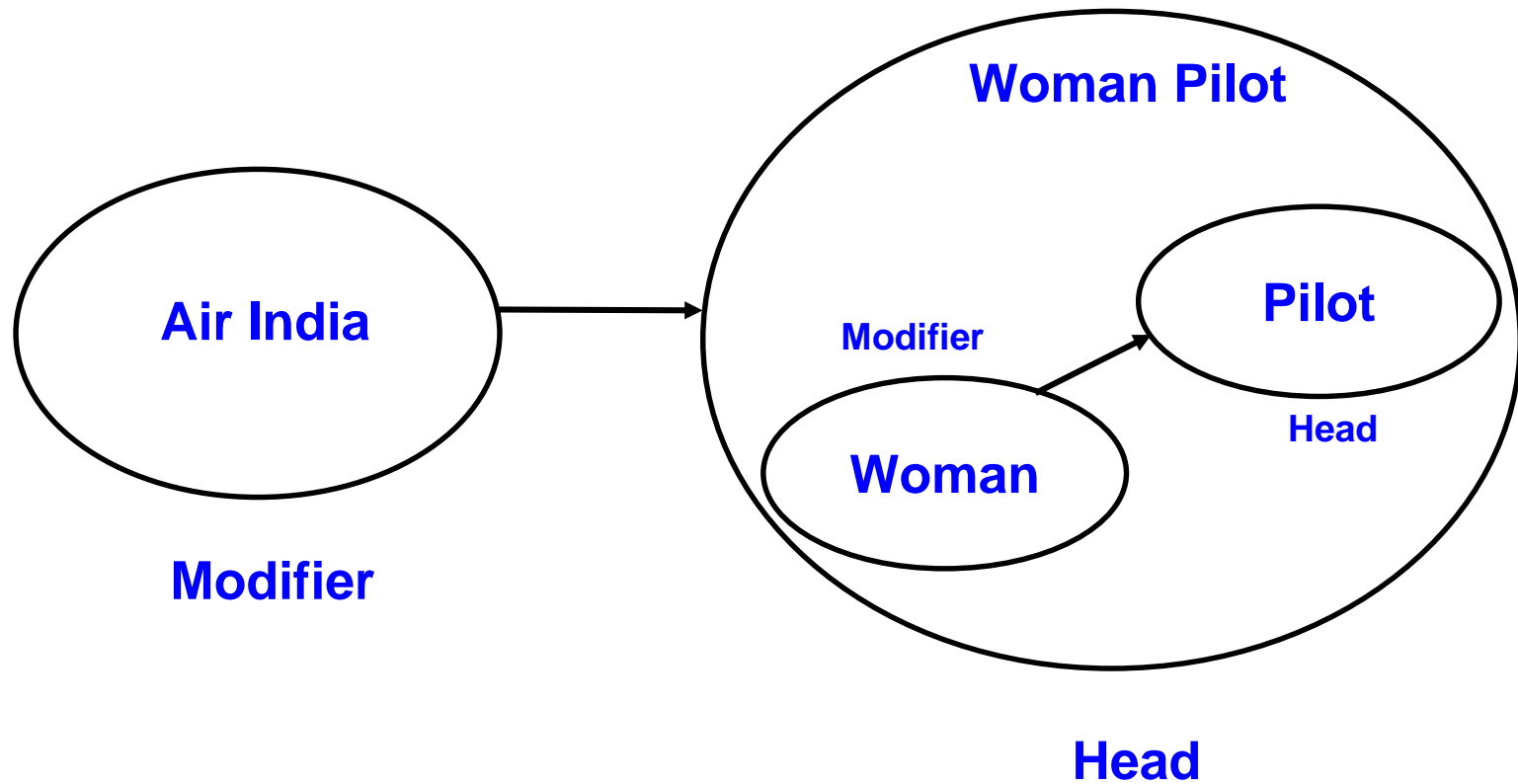


Shallow and Deep Parsing

Air/B-ORG India/I-ORG women/O pilots/O
to/O fly/O over/O North/B-LOC Pole/I-
LOC on/O world/O 's/O longest/O air/O
route/O from/O Bengaluru/B-GPE to/O
San/B-GPE Francisco/I-GPE ./O

An example of named entity marked text according to CONLL format: ORG means organization; GPE means geopolitical entity

Structure of “Air India Women Pilot”



Parsing the sentence, “The detective listened with a wooden face”

```
(ROOT
  (S
    (NP (DT The) (NN detective))
    (VP (VBD listened)
      (PP (IN with)
        (NP (DT a) (JJ wooden) (NN face))))
    (. .)))
```

```
det(detective-2, The-1)
nsubj(listened-3, detective-2)
root(Root-0, listened-3)
prep(listened-3, with-4)
det(face-7, a-5)
amod(face-7, wooden-6)
pobj(with-4, face-7)
```

Bracketed structure is still a
Sequence labelling problem;
The ‘bracket labels come
BETWEEN positions, while
Other labels come **ON**
positions

Need for parsing- sentiment analysis

- An opinion is a quintuple, $(\mathbf{e}_i, \mathbf{a}_{ij}, \mathbf{s}_{ijkl}, \mathbf{h}_k, \mathbf{t}_l)$, where
 - \mathbf{e}_i is the name of an entity,
 - \mathbf{a}_{ij} is an aspect of \mathbf{e}_i ,
 - \mathbf{s}_{ijkl} is the sentiment on aspect \mathbf{a}_{ij} of entity \mathbf{e}_i ,
 - \mathbf{h}_k is the opinion holder,
 - and \mathbf{t}_l is the time when the opinion is expressed by \mathbf{h}_k
- The sentiment \mathbf{s}_{ijkl} is
 - positive, negative, or neutral, or
 - expressed with different strength /intensity levels, e.g., 1–5 stars as used by most review sites on the Web
- When an opinion is on the entity itself as a whole, the special aspect GENERAL is used to denote it. Here, \mathbf{e}_i and \mathbf{a}_{ij} together represent the opinion target.

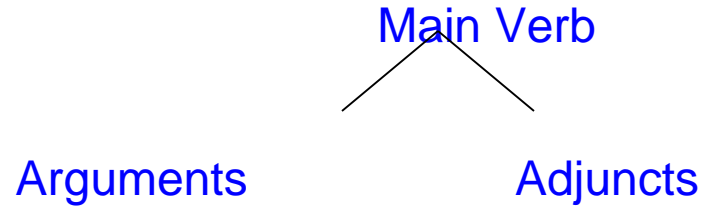
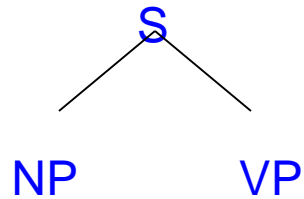
Examples

- “I loved the songs in the movie, though only the cast was liked by my brother”
- Another example: “I loved the color of my Motorola Phone back in 2000”:
Sentiment (S): positive, Entity (e):
Motorola Phone, Aspect (a): color,
Opinion holder (h): I, Time (t): 2000

Example (cntd.)

- Entity: *movie*
- Aspects: *songs, cast*
- Opinion holder: *I, brother*
- Time: *present (I), past (brother)*
- Opinioner-sentiment-aspect: *I-love-song, brother-like-cast*

Two kinds of parse representations: Constituency Vs. Dependency



Dependency Parsing

- Dependency approach is suitable for free word-order language
- Example : Hindi
 - राम ने शाम को देखा (Ram ne Shyam ko dekha)
 - शाम को राम ने देखा (Shyam ko Ram ne dekha)
- One step closer to **Semantics**

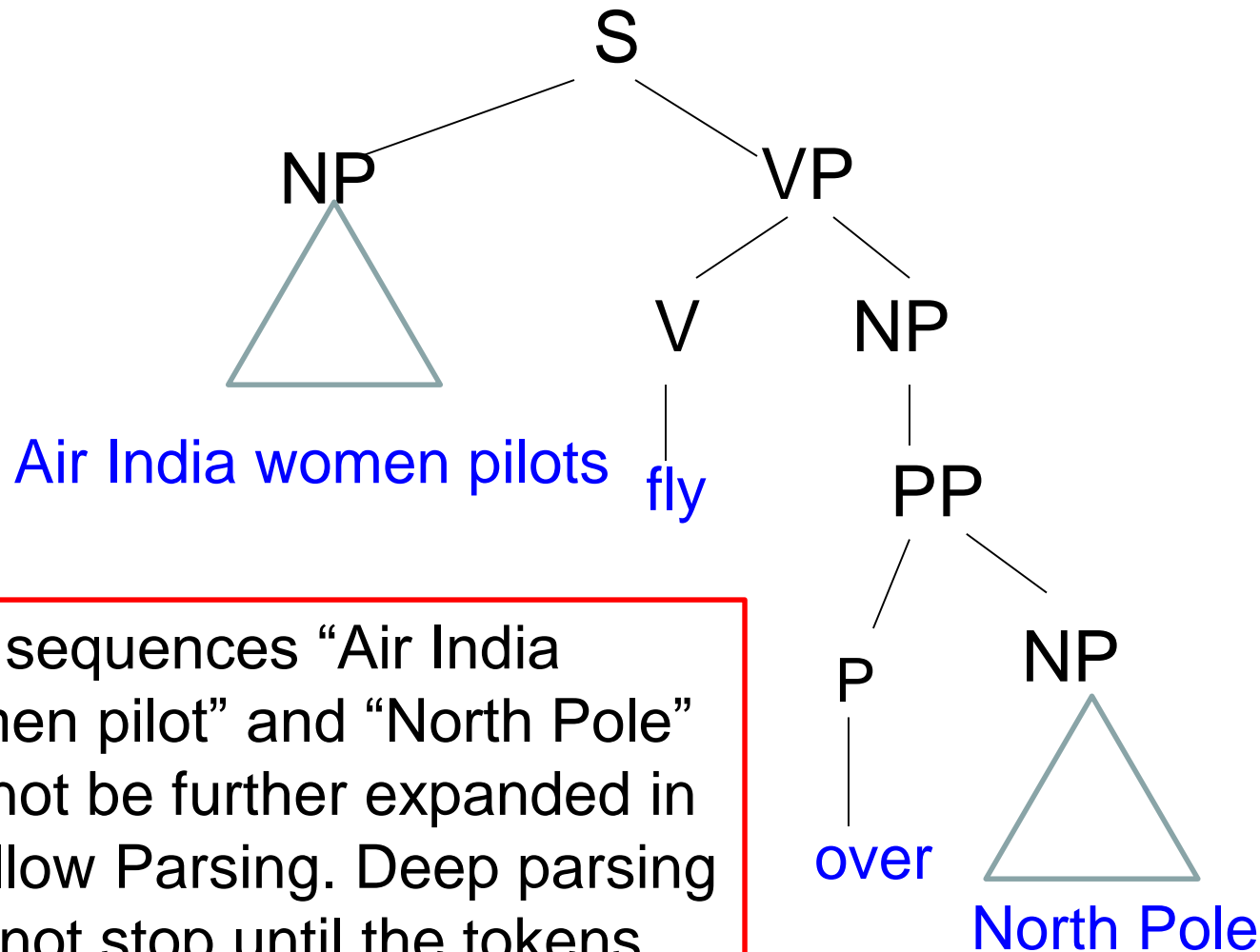
Important observation

- Free word order languages have to encode a special component of meaning called SEMANTIC ROLE (other components being, word sense, nuances, speech acts etc.) in post positions ('raam ne', 'shyaam ko' in Hindi) and/or affixes ("meMne' again in Hindi)
- Non-free word order languages have to encode this in positions (e.g. agent to the left of main verb in English)
- Chinese is an interesting case which is loose with respect to BOTH word position and affixes/pre-positions

[Link](#)

Observations on Shallow and Deep Parsing

“Air India women pilots fly over North Pole”



The sequences “Air India women pilot” and “North Pole” will not be further expanded in Shallow Parsing. Deep parsing Will not stop until the tokens (terminals are reached)

BIO for Chunk (non recursive phrase)

Air/B India/I women/I pilots/I
fly/O over/O North/B Pole/I./O

Target phrases:

Air India women pilots

North Pole

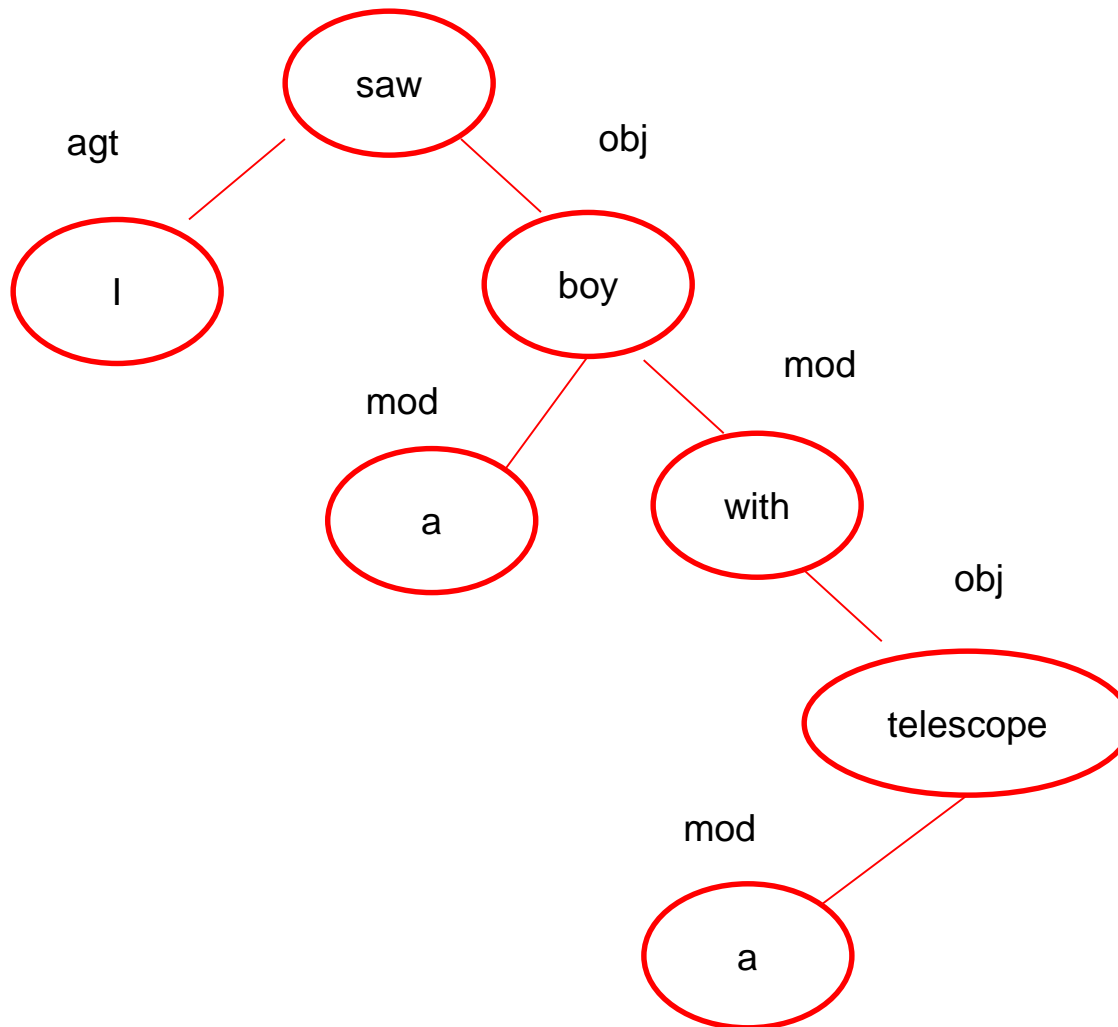
Chunk Extraction: *local* information adequate

- Use *argmax* computation
- Produce labels at positions
- Use **features** ON and AROUND positions

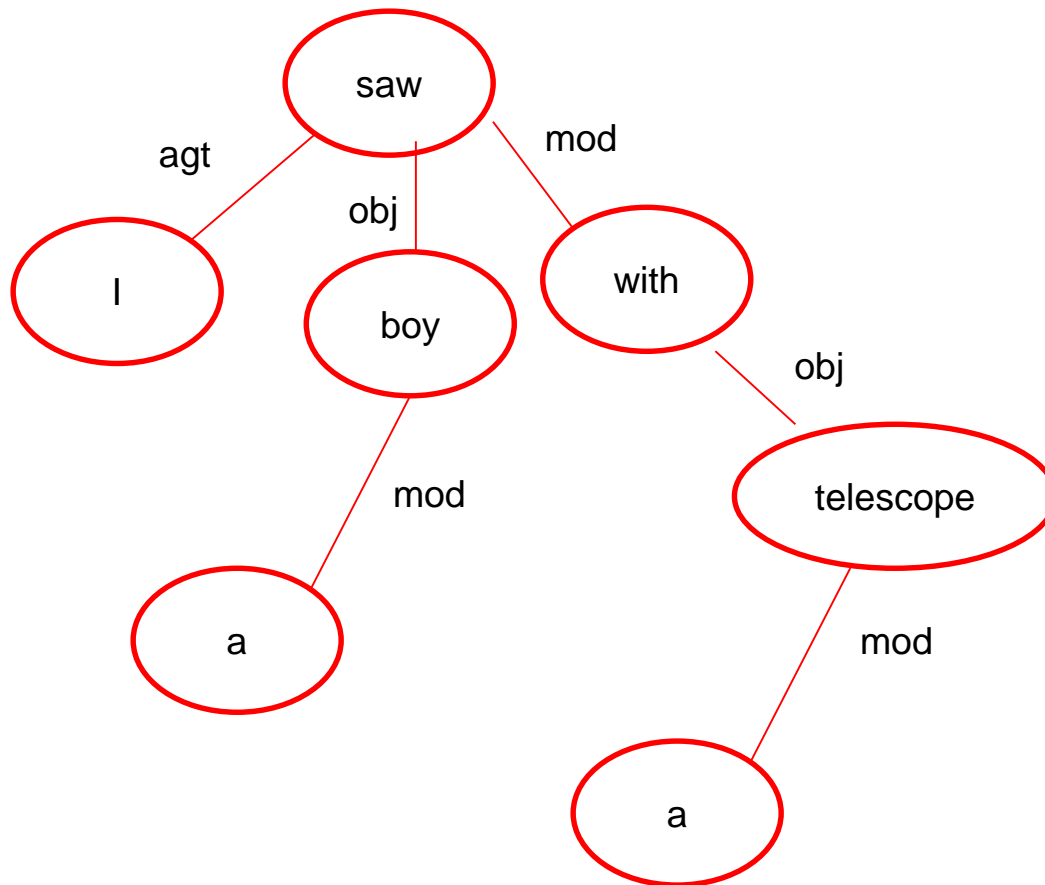
Foundation of Constituency Tree

- Parent child relation means parent is constituted of child(ren)
- If there are multiple children, i.e., multiple constituents, one of them is the head and others are modifiers
- Thus given $VP \rightarrow V NP$, VP is constituted of V and NP
- V is the head and NP the modifier for the VP

Dependency Parse Tree-1: *I saw a boy with a telescope*



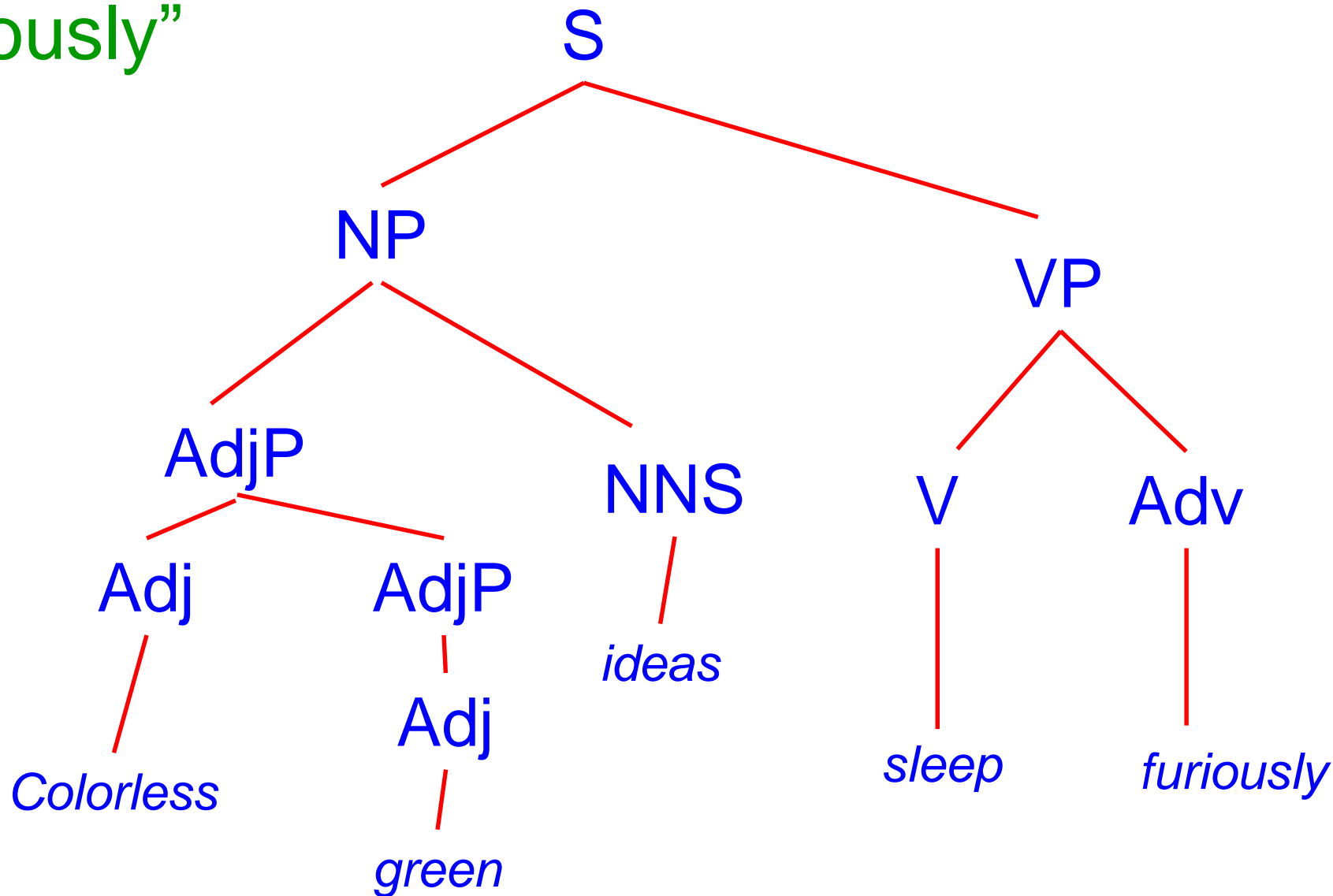
Dependency Parse Tree - 2



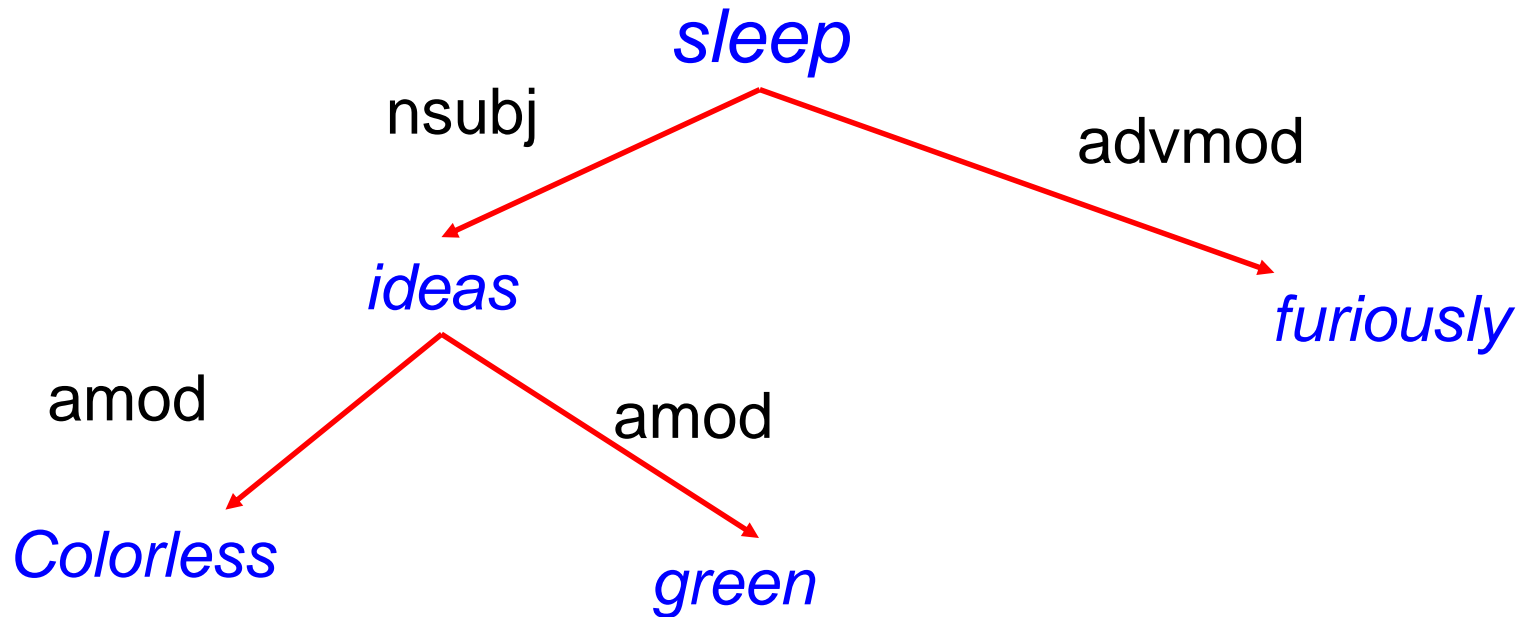
Foundation of Dependency Tree (DT)

- Parent child relation is head-modifier
- Labelled DT: the head-modifier relation is further specified with the type, e.g., *nsubj* meaning nominal subject, *dobj* meaning direct object and *iobj* meaning indirect object of the main verb (*mv*).
- E,g, *Jack*_{nsubj} *gave*_{mv} *a book*_{dobj} *to Jill*_{iobj}.

Constituency parse tree of a famous sentence: “Colorless green ideas sleep furiously”



Dependency parse tree of “Colorless...”: Head Modifier Relations



Syntax-Semantics

- Syntax and semantics influence each other
- However, they can be independent too- as in the “*colourless green ideas...*” sentence
- Consistent with neurolinguistics- Broca’s and Wernicke’s areas

Algorithmics of Parsing

Problem Statement

INPUT: (a) grammar rules, (b) input sentence

OUTPUT: Parse Tree
(Constituency/Dependency)

Top Down

- Start with the *S* symbol and draw its children: say, *NP* and *VP*, assuming the input to be a declarative sentence.
- Now the subtrees under *NP*, followed by that under the *VP* are developed.
- For example, $NP \rightarrow DT\ NN$ could be applied.
- After this, only POS tags will need to be resolved. *DT* will absorb, say, the word '*the*' in the input and *NN*, '*man*'.
- This will complete constructing the *NP* subtree.
- Similarly, *VP* subtree also will be constructed.

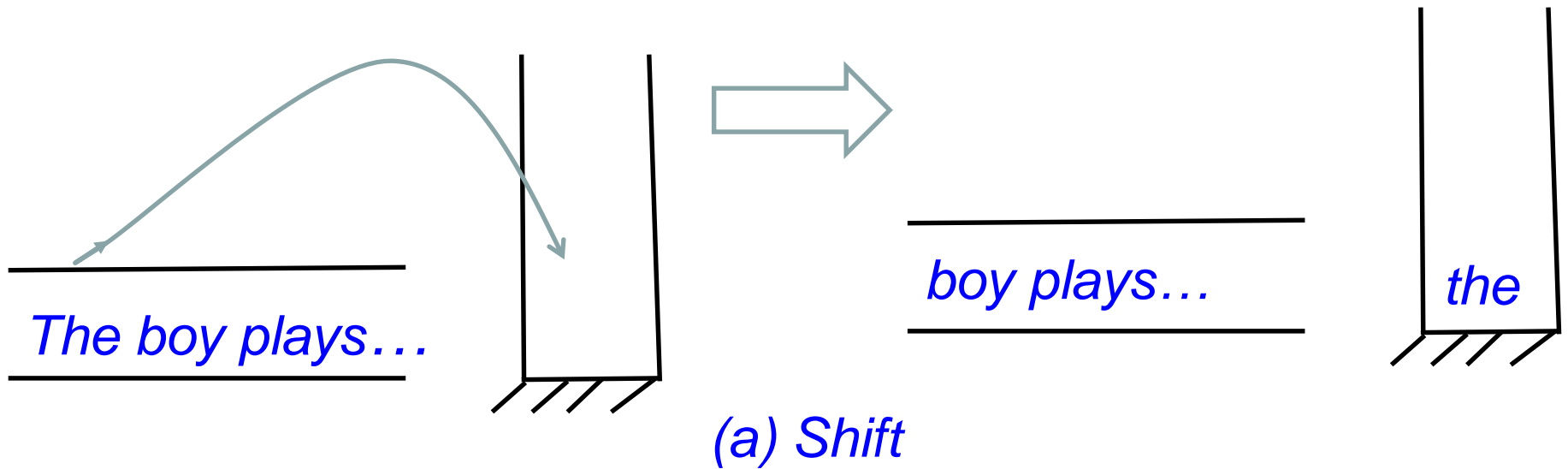
Bottom Up

- The words are resolved to their POS tags.
- Then POS tags are combined by constituency rules, e.g., $NP \rightarrow DT\ NN$. Generated non terminals are then attempted to be combined.
- For example, after generating JJP , NP they are combined to form a bigger NP , by applying $NP \rightarrow JJP\ NP$.

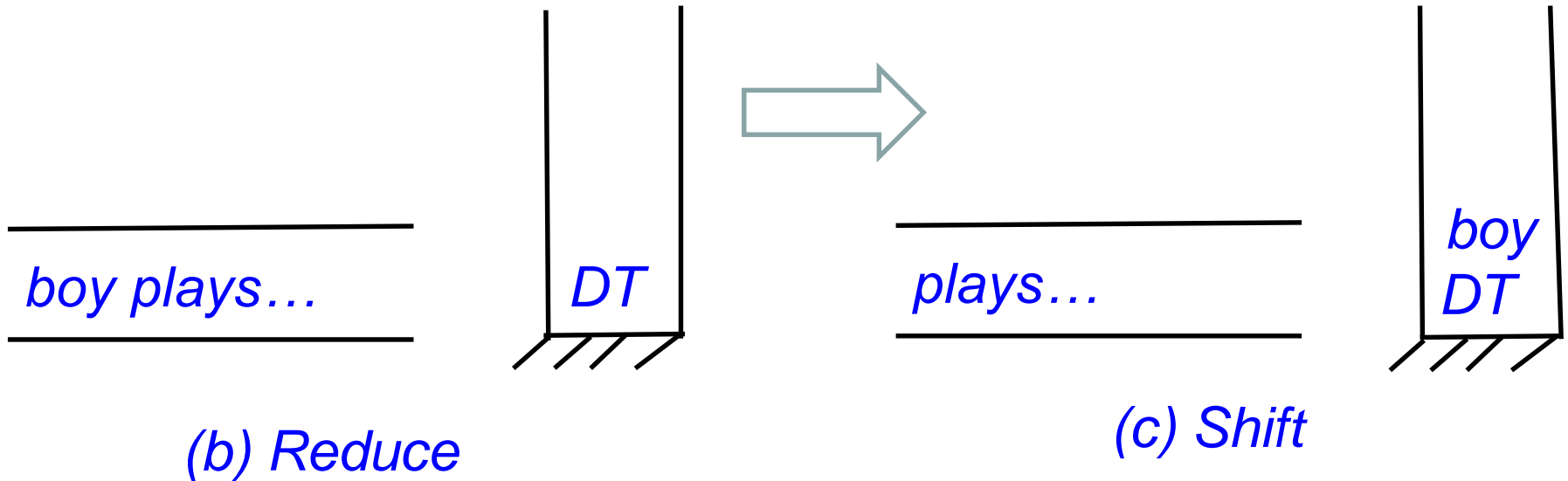
Main Operations

- Doing a left to right scan of the input sentence
- At every word, deciding if the word should (a) create a new constituent or (b) wait until more words get a look-in to create a constituent, and
- On creation of a new constituent, examining if the new constituent can be merged with an adjacent one to form a bigger constituent.

Shift Reduce (1/3)

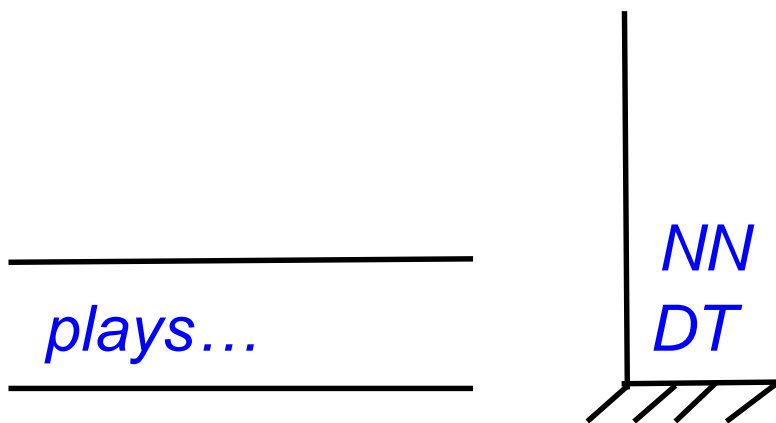


Shift Reduce (2/3)

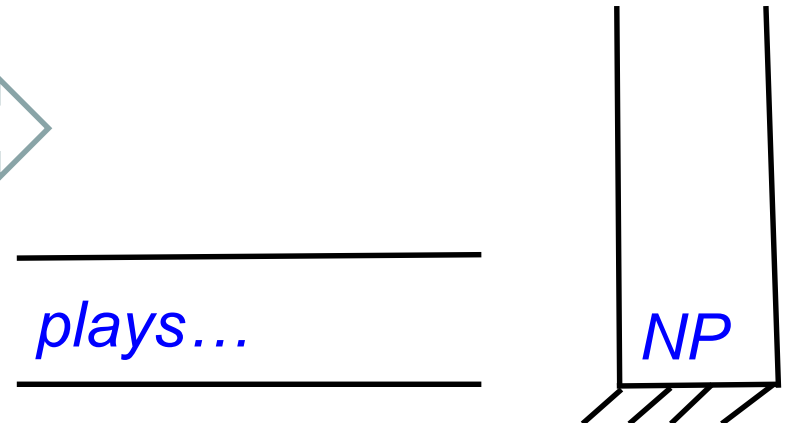


Continuously deciding between 'shift' and 'reduce'; reduce means replacing a word with its POS or replacing two adjacent phrase symbols with another phrase symbol, e.g., replace *DT NN* with *NP*; an ML model can be trained with data to decide between shift and reduce.

Shift Reduce (3/3)



(d) Reduce



(e) Reduce

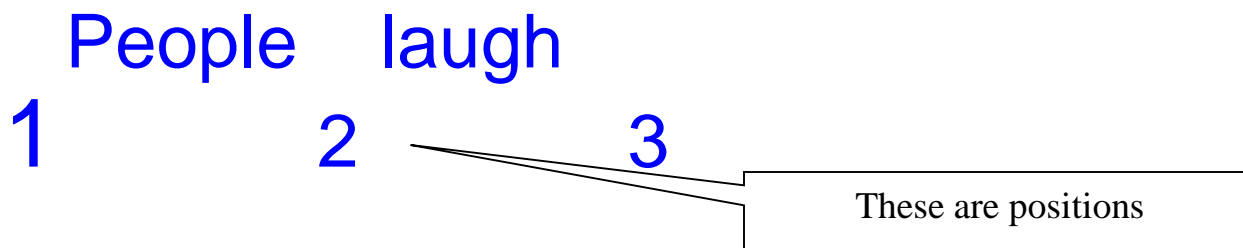
Grammar and Parsing Algorithms

A simplified grammar

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

- The above captures declarative sentences
- 4 kinds of sentences as per traditional grammar
 - Declarative (Sun rises in the east)
 - Interrogative (Does sun rise in the east?)
 - Imperative (Rise in the east please)
 - Exclamatory (Oh, sun rises in the east!)

Example Sentence



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)	-	No DT, retrieve backup state
4. ((VP)2)	-	Consume "People"
5a. ((V ADV)2)	((V)2)	-
6. ((ADV)3)	((V)2)	Consume "laugh"
5b. ((V)2)	-	Backtrack
6. ((.)3)	-	Consume "laugh"

Position of
input pointer

Termination Condition : All inputs over. No symbols remaining.

Note: Input symbols can be pushed back.

Exercise

- Construct examples of Top-Down parsing failure by
 - Input over but stack not empty
 - Stack empty but input not over

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}

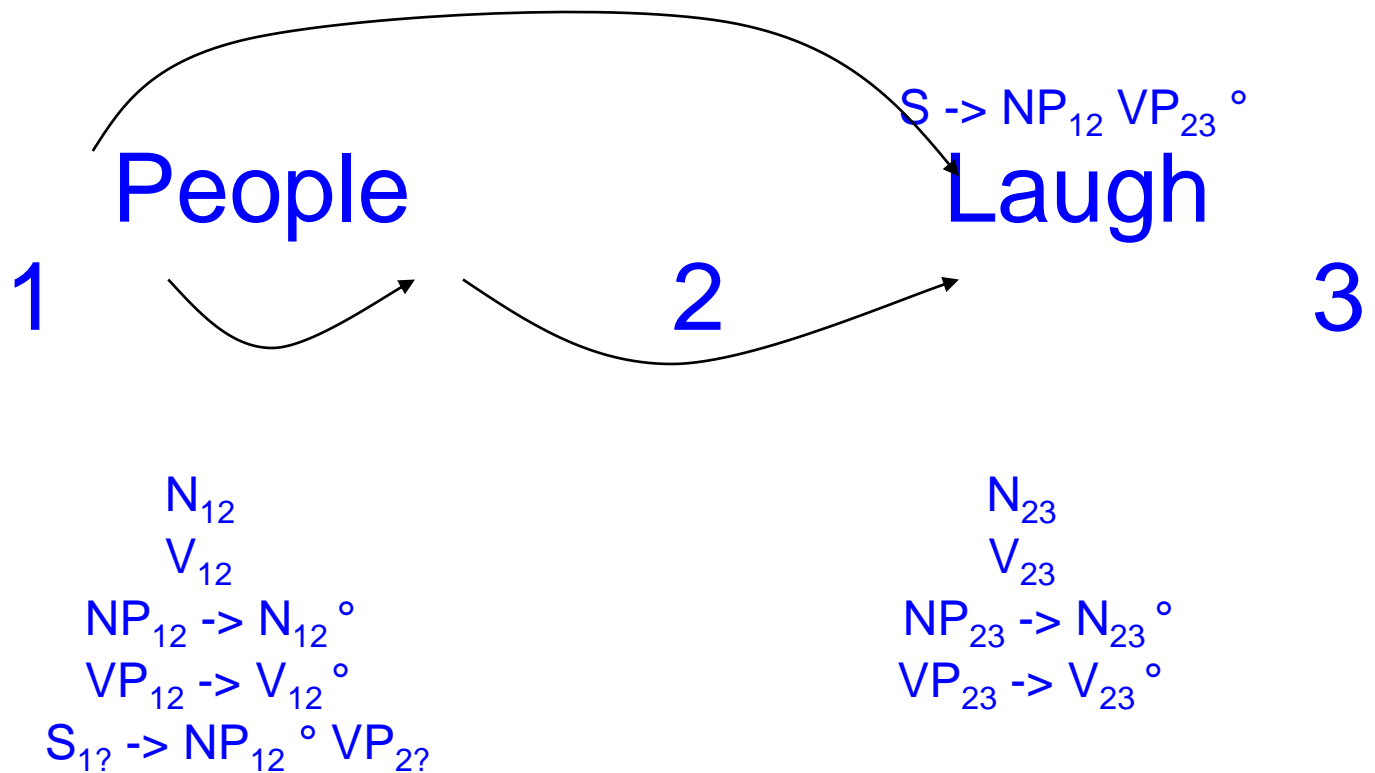
Represents positions; N between
pos 1 and 2

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

End position unknown

DOT: Work to the left done, while the
work on the right remaining

Bottom-Up Parsing (pictorial representation)



Problem with Top-Down Parsing

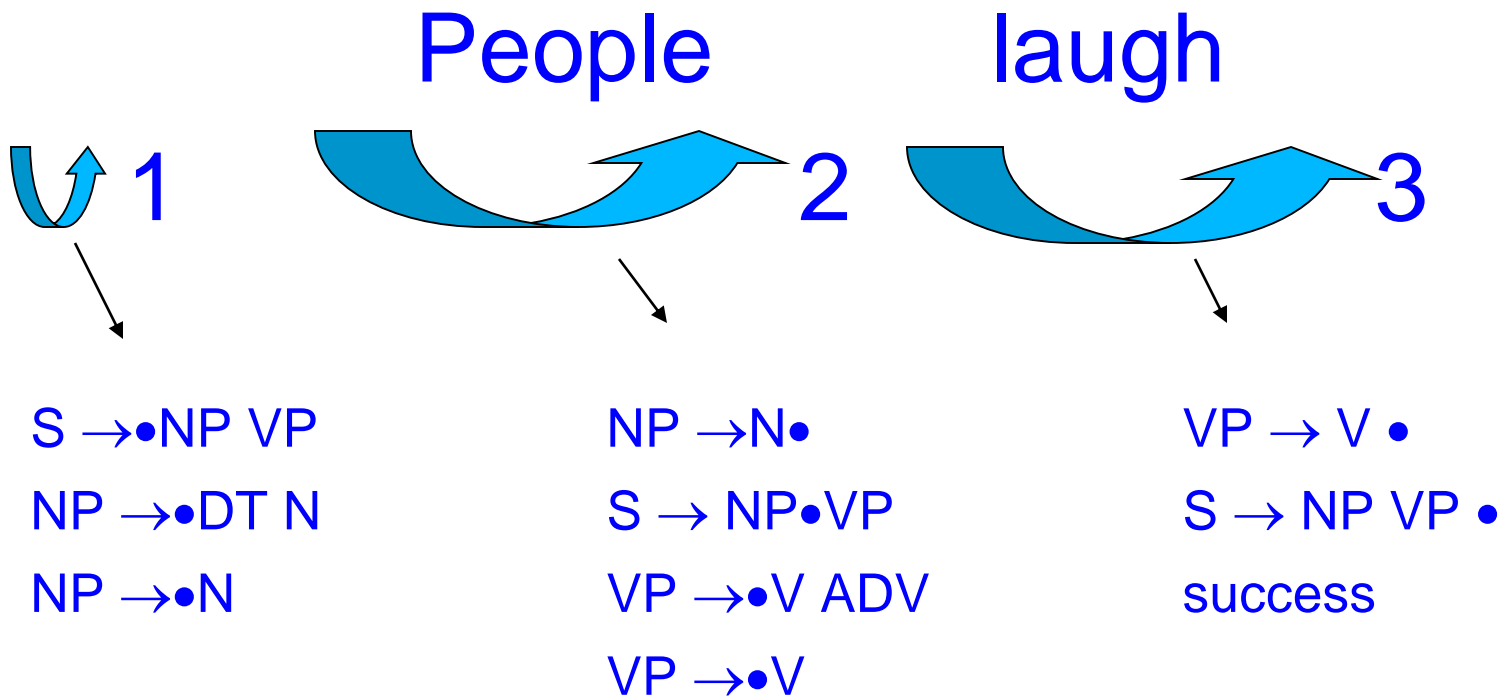
- Left Recursion
 - Suppose you have $A \rightarrow AB$ rule.
Then we will have the expansion as follows:
 - $((A)K) \rightarrow ((AB)K) \rightarrow ((ABB)K) \dots\dots$

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

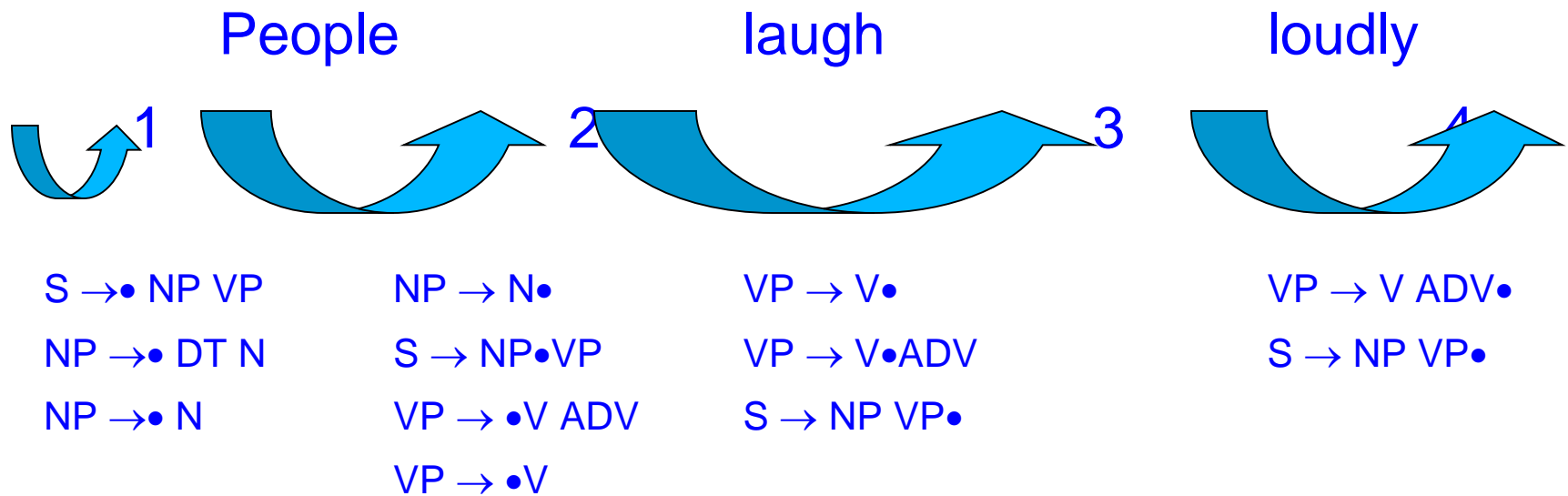
Transitive Closure



Arcs in Parsing

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

Example



An important parsing algorithm

CYK Parsing

A segment of English

- $S \rightarrow NP VP$
- $NP \rightarrow DT NN$
- $NP \rightarrow NNS$
- $NP \rightarrow NP PP$
- $PP \rightarrow P NP$
- $VP \rightarrow VP PP$
- $VP \rightarrow VBD NP$
- $DT \rightarrow \text{the}$
- $NN \rightarrow \text{gunman}$
- $NN \rightarrow \text{building}$
- $VBD \rightarrow \text{sprayed}$
- $NNS \rightarrow \text{bullets}$

GENERATIVE GRAMMAR, due
to Noam Chomsky

Foundational Question

- Grammar rules are context free grammar (CFG) rules
- Is CFG enough powerful to capture language?
- CFG cannot accept/generate $a^n b^n c^n$
- Corresponding language phenomenon: Jack, Mykel and Mohan play tennis, soccer and cricket respectively.

CYK Parsing: Start with (0,1)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT						
1	-----						
2	-----	-----					
3	-----	-----	-----				
4	-----	-----	-----	-----			
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	

CYK: Keep filling diagonals

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT						
1 →	-----	NN					
2 ↓	-----	-----					
3	-----	-----	-----				
4	-----	-----	-----	-----			
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	



CYK: Try getting higher level structures

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP					
1 →	-----	NN					
2 ↓	-----	-----					
3	-----	-----	-----				
4	-----	-----	-----	-----			
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	

CYK: Diagonal continues

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP					
1 	-----	NN					
2 	-----	-----	VBD				
3	-----	-----	-----				
4	-----	-----	-----	-----			
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	

CYK (cont...)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0 →	DT	NP	-----				
1 ↓	-----	NN	-----				
2	-----	-----	VBD				
3	-----	-----	-----				
4	-----	-----	-----	-----			
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	

CYK (cont...)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0 →	DT	NP	-----				
1	-----	NN	-----				
2 ↓	-----	-----	VBD				
3	-----	-----	-----	DT			
4	-----	-----	-----	-----			
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	



CYK (cont...)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0 →	DT	NP	-----	-----			
1 ↓	-----	NN	-----	-----			
2	-----	-----	VBD	-----			
3	-----	-----	-----	DT			
4	-----	-----	-----	-----	NN		
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	

CYK: starts filling the 5th column

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	-----	-----			
1 	-----	NN	-----	-----			
2 	-----	-----	VBD	-----			
3	-----	-----	-----	DT	NP		
4	-----	-----	-----	-----	NN		
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	

CYK (cont...)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	-----	-----			
1	-----	NN	-----	-----			
2	-----	-----	VBD	-----	VP		
3	-----	-----	-----	DT	NP		
4	-----	-----	-----	-----	NN		
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	

CYK (cont...)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	-----	-----			
1	-----	NN	-----	-----	-----		
2	-----	-----	VBD	-----	VP		
3	-----	-----	-----	DT	NP		
4	-----	-----	-----	-----	NN		
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	

CYK: S found, but NO termination!

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	-----	-----	S		
1	-----	NN	-----	-----	-----		
2	-----	-----	VBD	-----	VP		
3	-----	-----	-----	DT	NP		
4	-----	-----	-----	-----	NN		
5	-----	-----	-----	-----	-----		
6	-----	-----	-----	-----	-----	-----	



CYK (cont...)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	-----	-----	S		
1	-----	NN	-----	-----	-----		
2	-----	-----	VBD	-----	VP		
3	-----	-----	-----	DT	NP		
4	-----	-----	-----	-----	NN		
5	-----	-----	-----	-----	-----	P	
6	-----	-----	-----	-----	-----	-----	

CYK (cont...)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	-----	-----	S	-----	
1 	-----	NN	-----	-----	-----	-----	
2 	-----	-----	VBD	-----	VP	-----	
3	-----	-----	-----	DT	NP	-----	
4	-----	-----	-----	-----	NN	-----	
5	-----	-----	-----	-----	-----	P	
6	-----	-----	-----	-----	-----	-----	

CYK: Control moves to last column

0 The 1 gunman 2 sprayed 3 the 4 building 5 with 6 bullets 7.

[illegible]

CYK (cont...)

0 The 1 gunman 2 sprayed 3 the 4 building 5 with 6 bullets 7.

[illegible]

CYK (cont...)

0 The 1 aunman 2 spraved 3 the 4 buildina 5 with 6 bullets 7.

[illegible]

CYK (cont...)

0 The 1 gunman 2 sprayed 3 the 4 building 5 with 6 bullets 7.

[illegible]

CYK: filling the last column

0 The 1 gunman 2 sprayed 3 the 4 building 5 with 6 bullets 7.

[illegible]

CYK: terminates with S in (0,7)

0 The 1 gunman 2 sprayed 3 the 4 building 5 with 6 bullets 7.

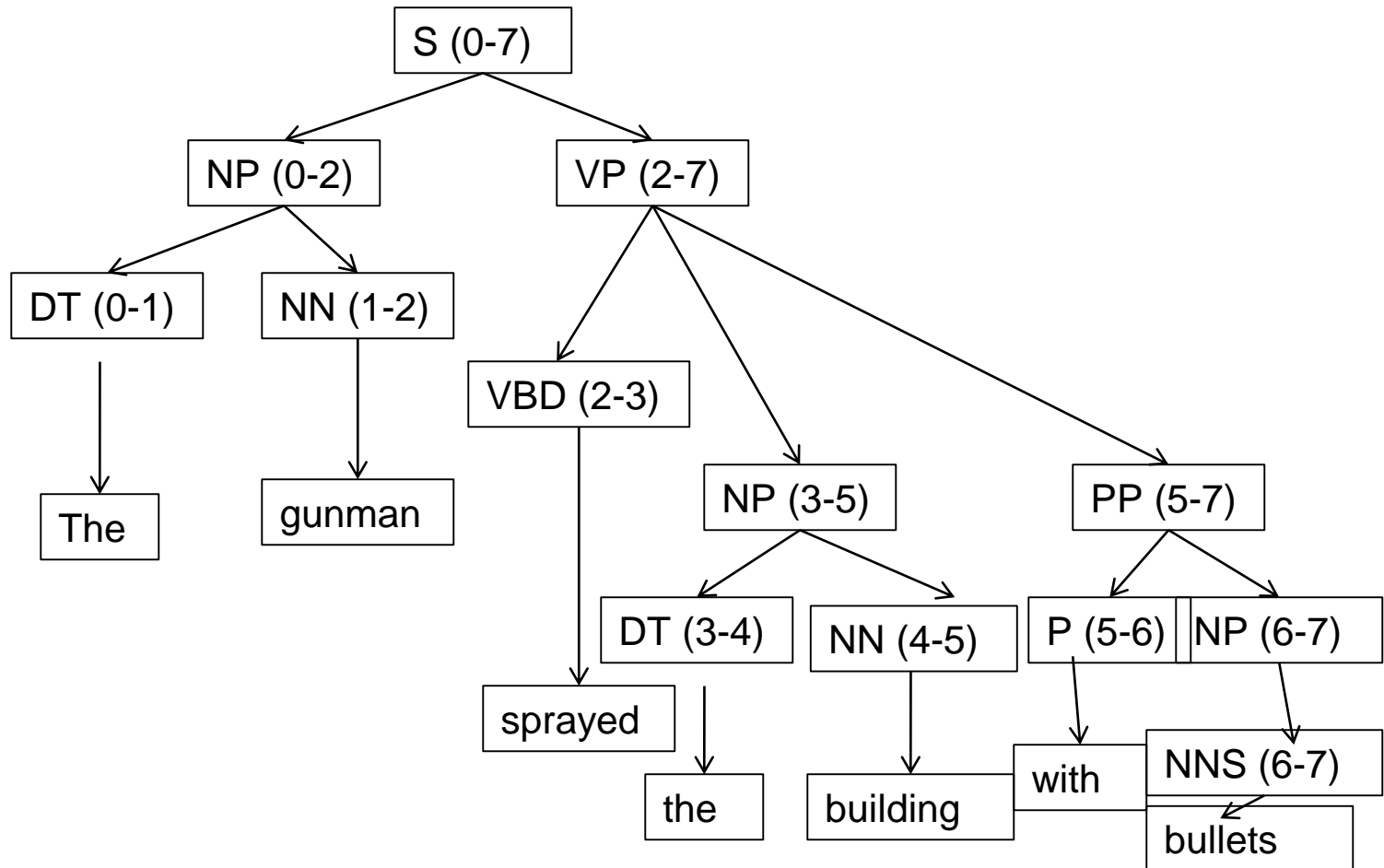
[illegible]

CYK: Extracting the Parse Tree

- The parse tree is obtained by keeping back pointers.

Parse Tree #1

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.



Parse Tree #2

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

