Constituency Parsing

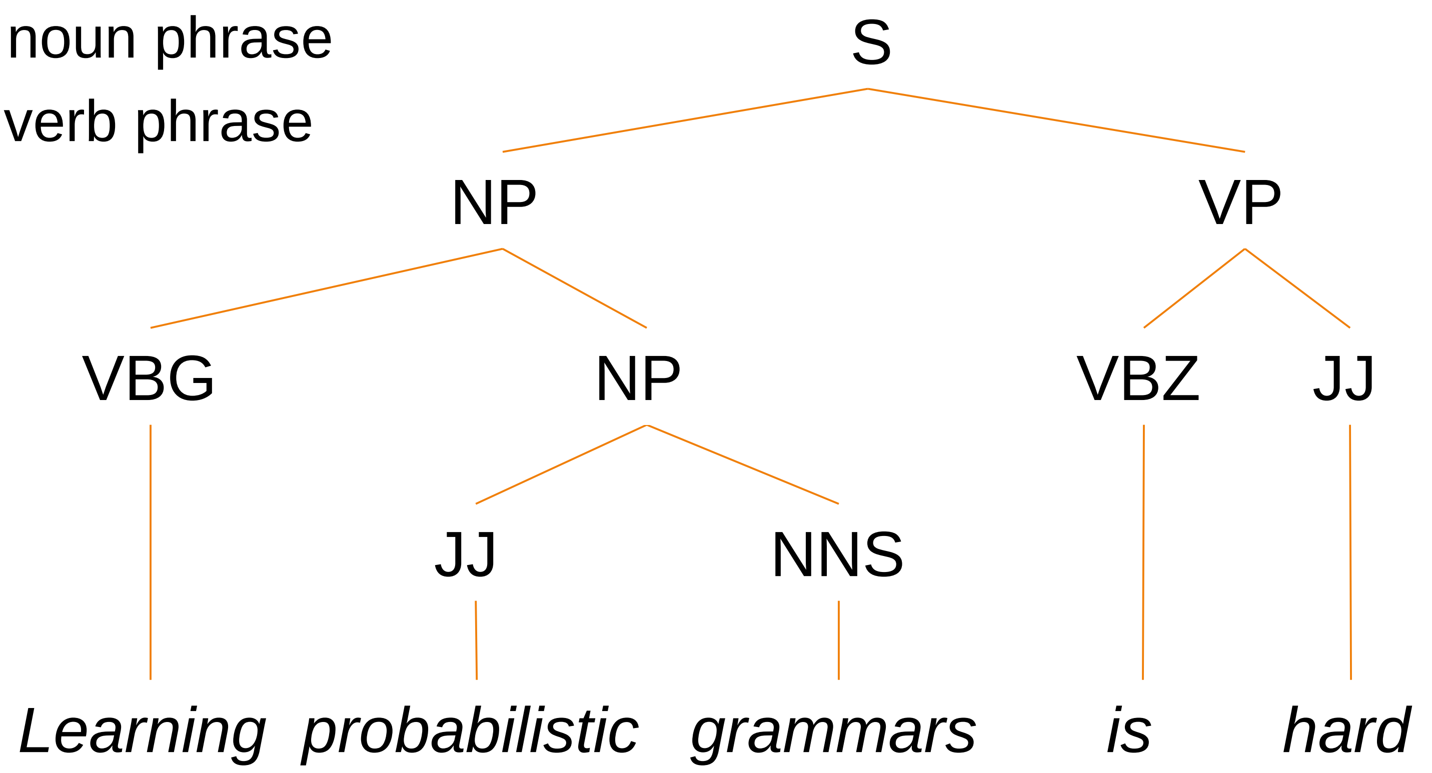
SLP3 Ch 12, 13; INLP Ch 9, 10

Syntax

- Syntax studies rules and processes that govern the structure of sentences
- Syntax is only about structure, not about meaning
  - A sentence can be syntactically well-formed but semantically ill-formed
    - Colorless green ideas sleep furiously.
  - Two semantically identical sentences can have different syntactic structures
    - A dog is chasing a cat.
    - A cat is being chased by a dog.

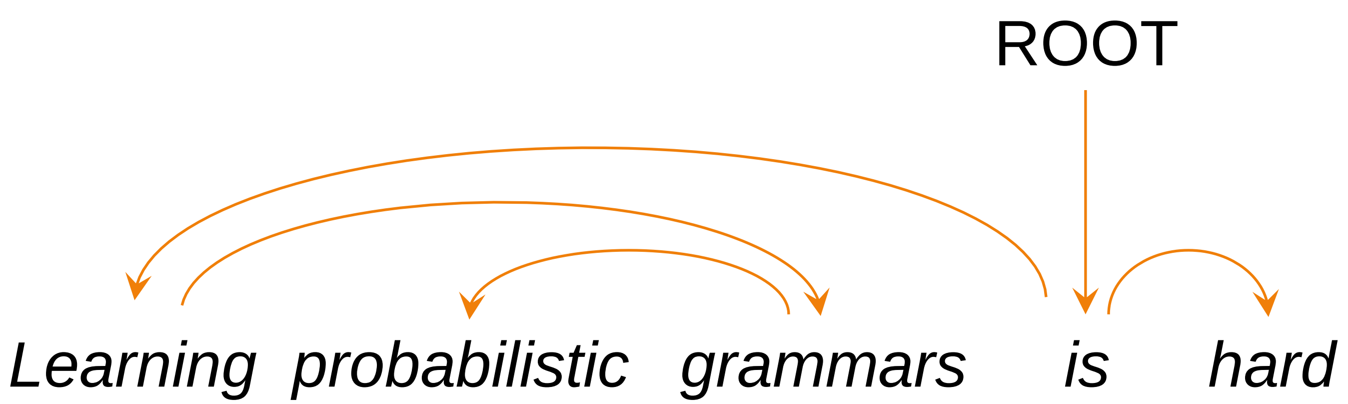
Constituent parse tree

- Also called a phrase structure parse
- Each non-leaf node represents a phrase
  - S: sentence
  - NP: noun phrase
  - VP: verb phrase
  - …



Dependency parse tree

- Each arc represents a binary dependency relation between two words
  - Relations may be typed (labeled)
    - Ex. Subject relation between a verb and a noun

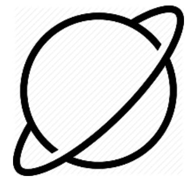


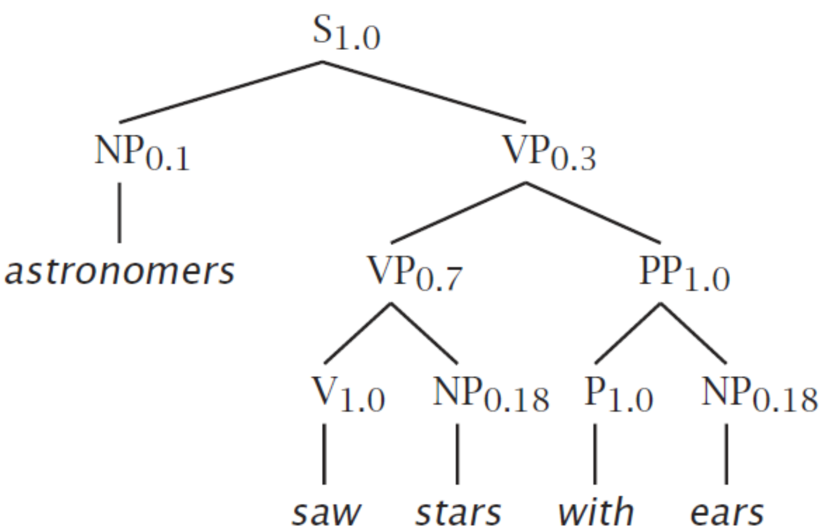
Parse tree scoring

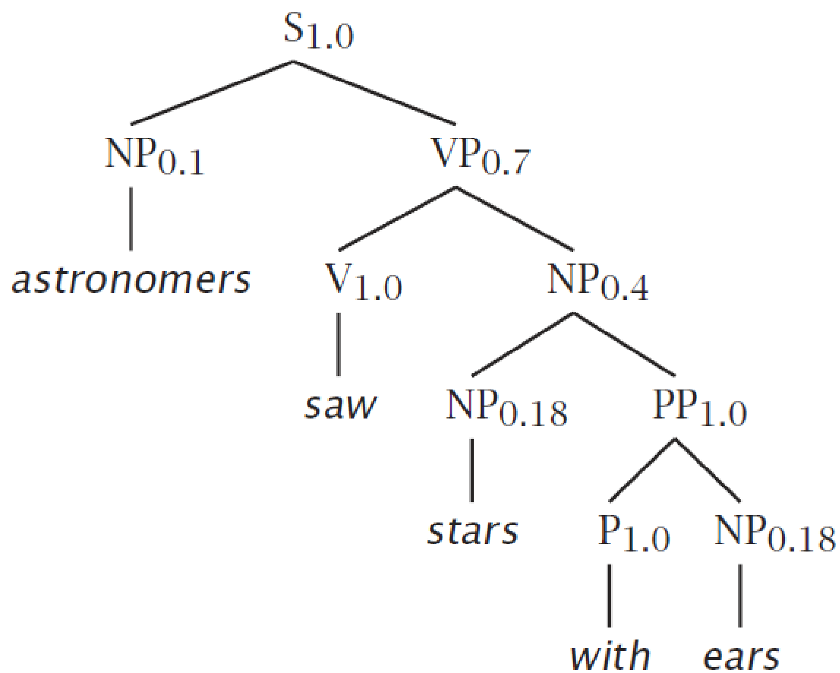
- Goal: assign a probability or score to each parse tree of a sentence
- Why?
  - Disambiguation!
  - A natural language sentence may have many possible parses

Ambiguity

- Astronomers saw stars with ears.







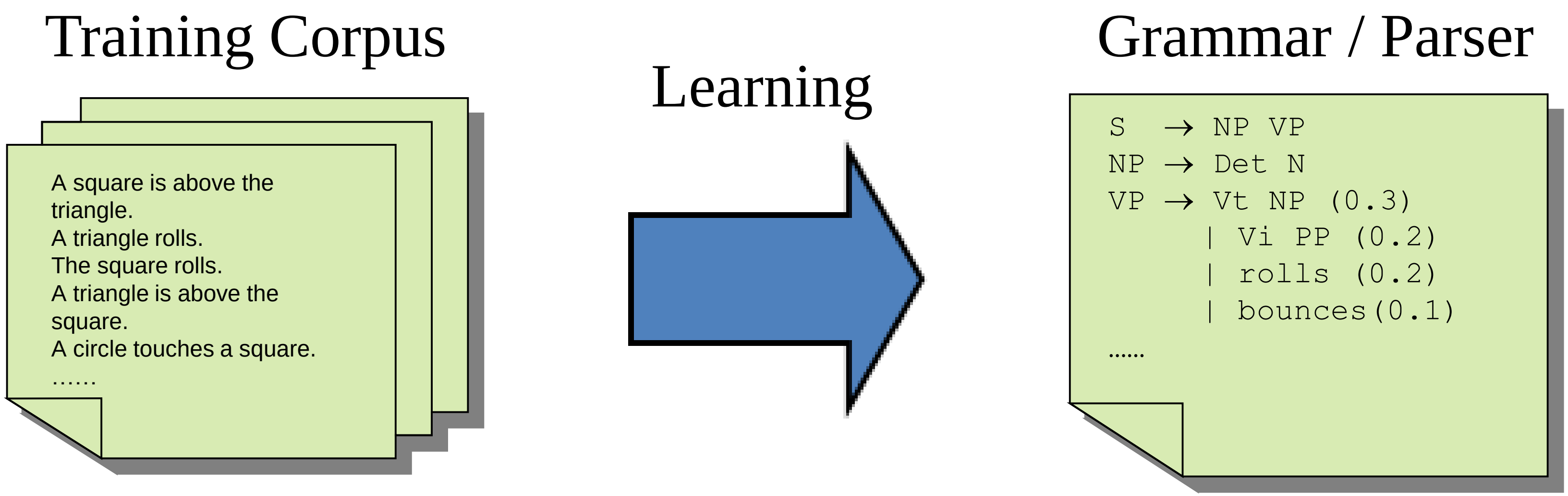
Parse tree scoring

- Goal: assign a probability or score to each parse tree of a sentence
- Common approach:
  - Decompose a parse tree into many parts
  - Score each part
  - Take the sum or product

Parsing

- Goal: given a sentence, find its highest scored parse tree
- Common approaches:
  - Dynamic programming
  - Greedy search / beam search

Learning



- Supervised Methods
  - Rely on a training corpus of sentences annotated with parses (treebank)
- Unsupervised Methods (Grammar Induction)
  - Do not require annotated data

Context-Free Grammars

Constituency

- Constituents
  - Groups of words within sentences can be shown to act as single units.
  - Ex: (The fox)(jumps(over(the dog)))
- These units form coherent classes
  - Units in the same class behave in similar ways
    - …with respect to their *internal* structure
    - …and with respect to other (*external*) units in the language
  - E.g., noun phrases

Constituency

- For example, it makes sense to say that the following are all ***noun phrases*** in English...

Harry the Horse	a high-class spot such as Mindy's
the Broadway coppers	the reason he comes into the Hot Box
they	three parties from Brooklyn

- Why?
  - Similar internal structures
    - e.g., determiner + modifier + noun + modifier
  - They can all precede verbs (external evidence)

Grammars and Constituency

- Grammar
  - the set of constituents and the rules that govern how they combine
- Lots of different theories of grammar
- Context-free grammars (CFGs)
  - Also known as: Phrase structure grammars
  - One of the simplest and most basic grammar formalisms

Context-Free Grammars

- A context-free grammar has four components
  - A set Σ of terminals (words)
  - A set N of nonterminals (phrases)
  - A start symbol $S \in N$
  - A set R of production rules
    - Specifies how a nonterminal can produce a string of terminals and/or nonterminals

Example Grammar

Grammar rule	Example
S → NP VP	I + want a morning flight
NP → Pronoun	I
| Proper-Noun	Los Angeles
| Det Nominal	a + flight
Nominal → Nominal Noun	morning + flight
| Noun	flights
VP → Verb	do
| Verb NP	want + a flight
| Verb NP PP	leave + Boston + in the morning
| Verb PP	leave + on Thursday
PP → Preposition NP	from + Los Angeles

Example Grammar

$$
\begin{aligned}
\text{Noun} &\rightarrow \text{flights} \mid \text{breeze} \mid \text{trip} \mid \text{morning} \\
\text{Verb} &\rightarrow \text{is} \mid \text{prefer} \mid \text{like} \mid \text{need} \mid \text{want} \mid \text{fly} \\
\text{Adjective} &\rightarrow \text{cheapest} \mid \text{non-stop} \mid \text{first} \mid \text{latest} \\
&\ \mid \text{other} \mid \text{direct} \\
\text{Pronoun} &\rightarrow \text{me} \mid \text{I} \mid \text{you} \mid \text{it} \\
\text{Proper-Noun} &\rightarrow \text{Alaska} \mid \text{Baltimore} \mid \text{Los Angeles} \\
&\ \mid \text{Chicago} \mid \text{United} \mid \text{American} \\
\text{Determiner} &\rightarrow \text{the} \mid \text{a} \mid \text{an} \mid \text{this} \mid \text{these} \mid \text{that} \\
\text{Preposition} &\rightarrow \text{from} \mid \text{to} \mid \text{on} \mid \text{near} \\
\text{Conjunction} &\rightarrow \text{and} \mid \text{or} \mid \text{but}
\end{aligned}
$$

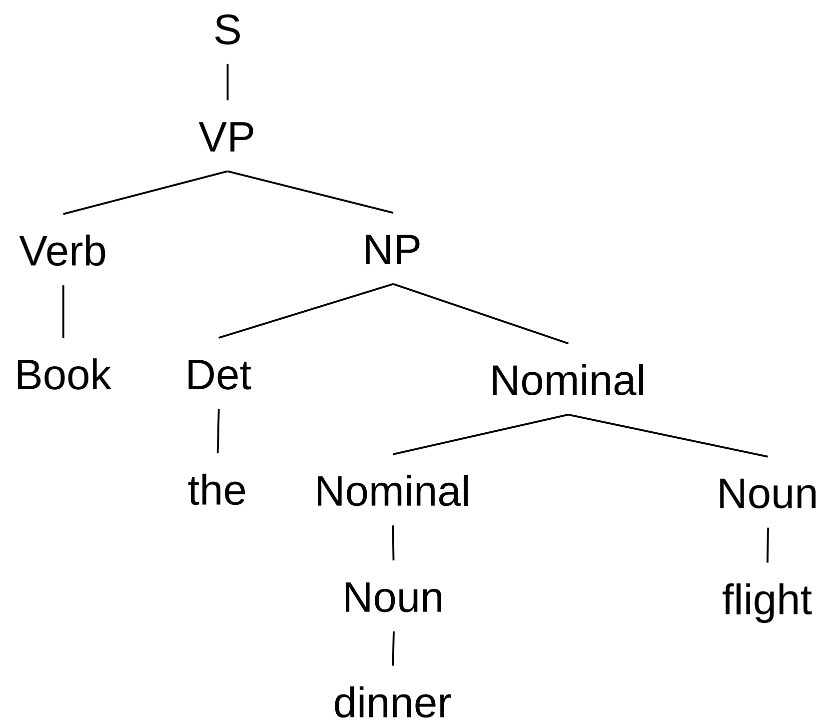
Sentence Generation

- A grammar can be used to generate a string
  - starting from a string containing only the start symbol S
  - recursively applying the rules to rewrite the string
  - until the string contains only terminals
- The generative process specifies the grammatical structure (parse tree) of the string

Example

$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$NP \rightarrow Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow Verb\ NP\ NP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

……



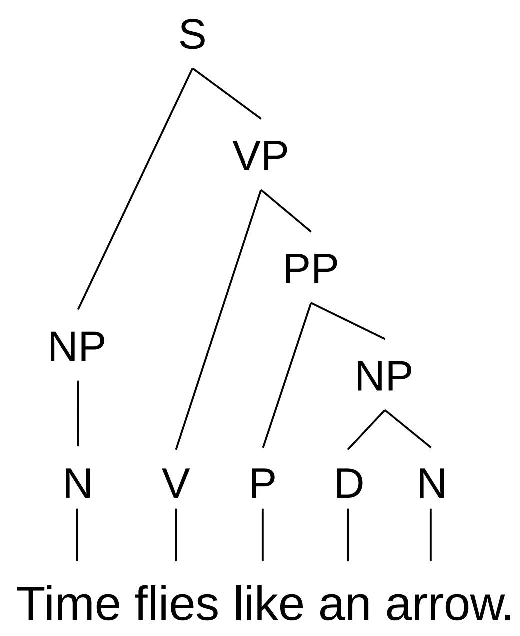
Book the dinner flight

Sentence Parsing

- Parsing is the process of taking a string and a grammar and returning one or more parse tree(s) for that string
  - If no parse tree can be found, then the string does not belong to the language
  - Parsing algorithms: CYK, Earley, etc.
    - To be introduced later

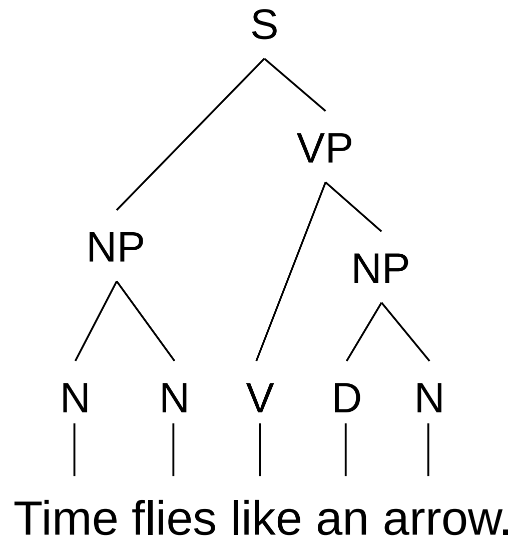
Ambiguity

- A sentence is ambiguous if it has more than one possible parse tree
  - …and hence more than one interpretation
- Examples
  - Time flies like an arrow.



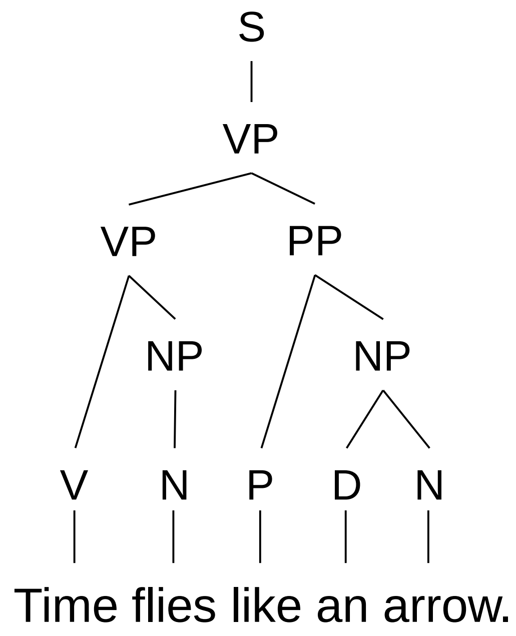
Ambiguity

- A sentence is ambiguous if it has more than one possible parse tree
  - …and hence more than one interpretation
- Examples
  - Time flies like an arrow.



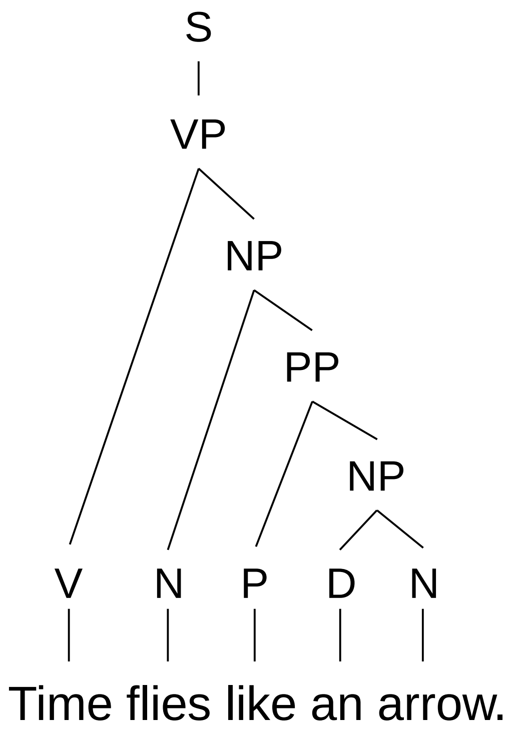
Ambiguity

- A sentence is ambiguous if it has more than one possible parse tree
  - …and hence more than one interpretation
- Examples
  - Time flies like an arrow.



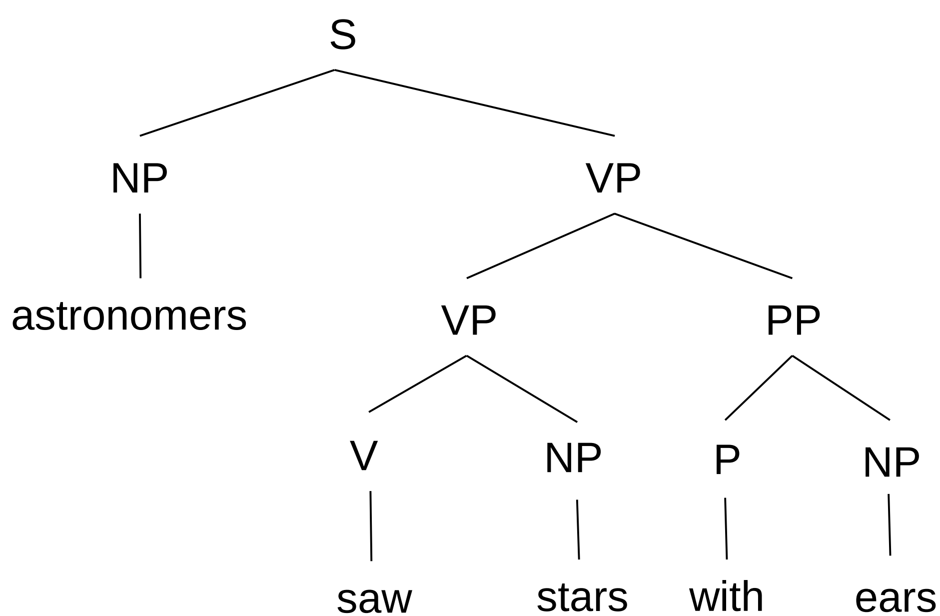
Ambiguity

- A sentence is ambiguous if it has more than one possible parse tree
  - …and hence more than one interpretation
- Examples
  - Time flies like an arrow.



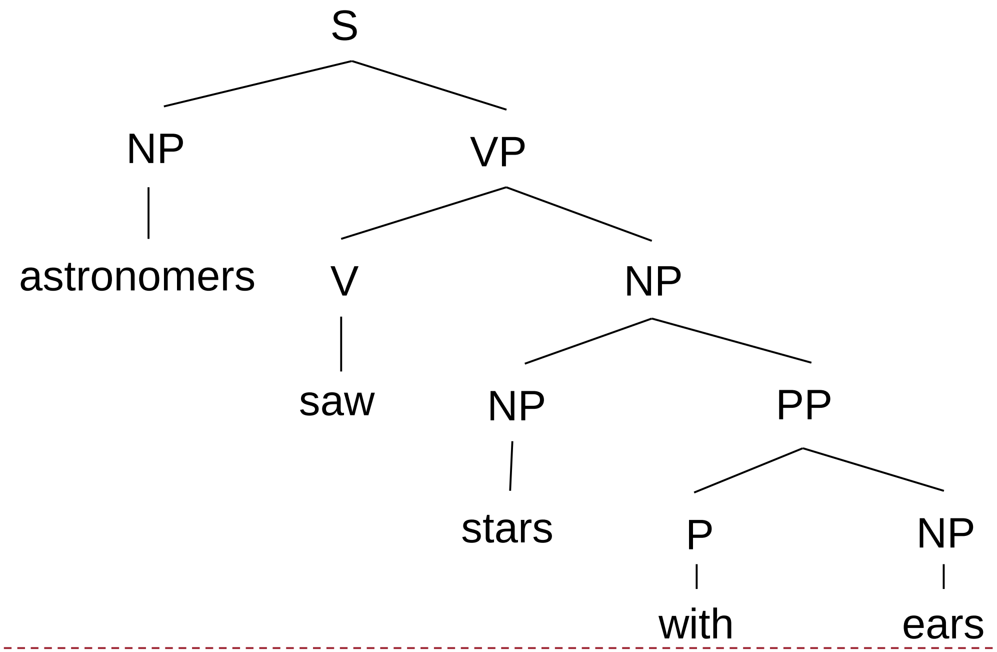
Ambiguity

- A sentence is ambiguous if it has more than one possible parse tree
  - …and hence more than one interpretation
- Examples
  - Astronomers saw stars with ears.



Ambiguity

- A sentence is ambiguous if it has more than one possible parse tree
  - …and hence more than one interpretation
- Examples
  - Astronomers saw stars with ears.



Probabilistic Grammars

- Also called stochastic grammars
- Each rule is associated with a probability

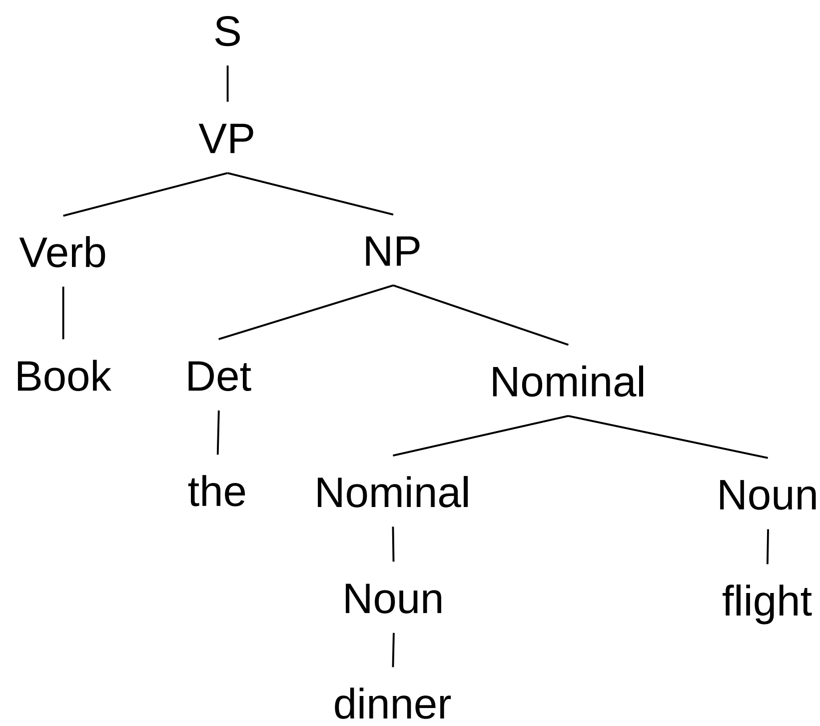
$$\alpha \to \beta : P(\alpha \to \beta | \alpha)$$

- Probability of a parse tree = product of the probabilities of all the rules used in generating the parse tree
- Weighted grammars
  - Replace probabilities with positive weights
  - Same expressiveness

Example

S	$\rightarrow$	$NP\ VP$	[.80]
S	$\rightarrow$	$Aux\ NP\ VP$	[.15]
S	$\rightarrow$	VP	[.05]
NP	$\rightarrow$	$Pronoun$	[.35]
NP	$\rightarrow$	$Proper\text{-}Noun$	[.30]
NP	$\rightarrow$	$Det\ Nominal$	[.20]
NP	$\rightarrow$	$Nominal$	[.15]
$Nominal$	$\rightarrow$	$Noun$	[.75]
$Nominal$	$\rightarrow$	$Nominal\ Noun$	[.20]
$Nominal$	$\rightarrow$	$Nominal\ PP$	[.05]
VP	$\rightarrow$	$Verb$	[.35]
VP	$\rightarrow$	$Verb\ NP$	[.20]
VP	$\rightarrow$	$Verb\ NP\ PP$	[.10]
VP	$\rightarrow$	$Verb\ PP$	[.15]
VP	$\rightarrow$	$Verb\ NP\ NP$	[.05]
VP	$\rightarrow$	$VP\ PP$	[.15]
PP	$\rightarrow$	$Preposition\ NP$	[1.0]

....



Book the dinner flight

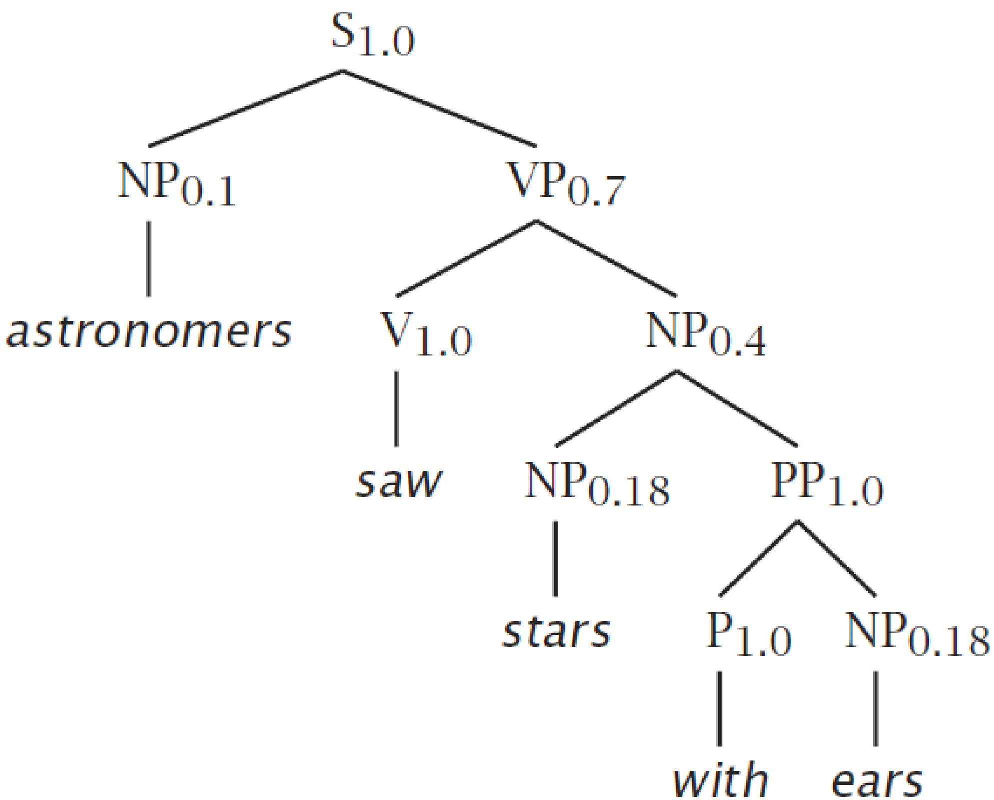
$P(T) = .05 \times .20 \times .20 \times .20 \times .75 \times .30 \times .60 \times .10 \times .40 = 2.2 \times 10^{-6}$

Example

$S \to NP\ VP$	1.0		$NP \to NP\ PP$	0.4
$PP \to P\ NP$	1.0		$NP \to$ *astronomers*	0.1
$VP \to V\ NP$	0.7		$NP \to$ *ears*	0.18
$VP \to VP\ PP$	0.3		$NP \to$ *saw*	0.04
$P \to$ *with*	1.0		$NP \to$ *stars*	0.18
$V \to$ *saw*	1.0		$NP \to$ *telescopes*	0.1

Example

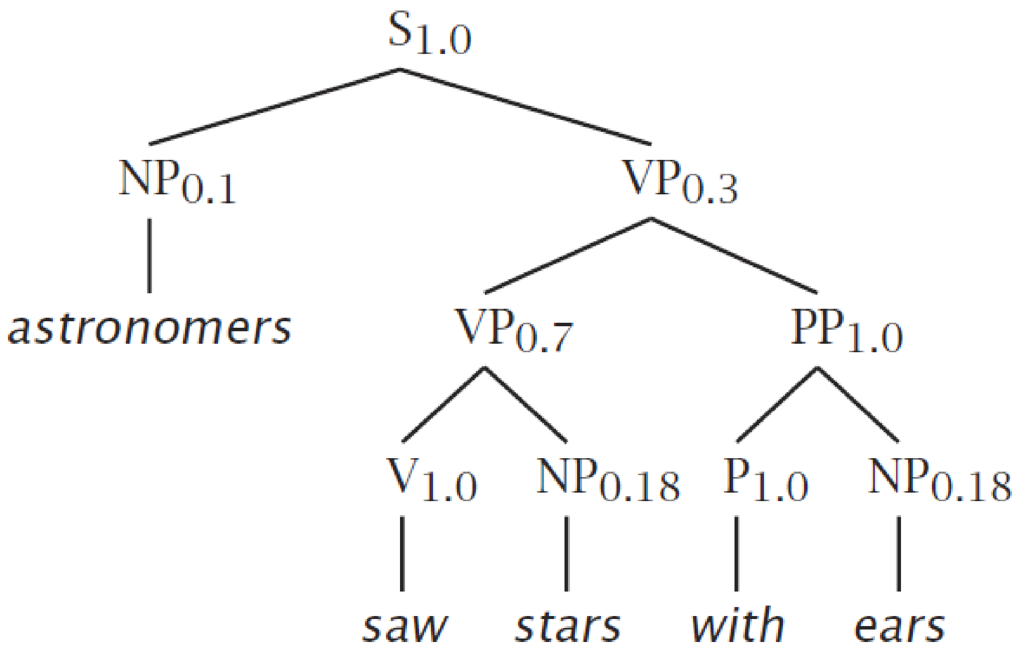
t_1:



$$
\begin{aligned}
P(t_1) &= 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \\
&\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0009072
\end{aligned}
$$

Example

t_2:



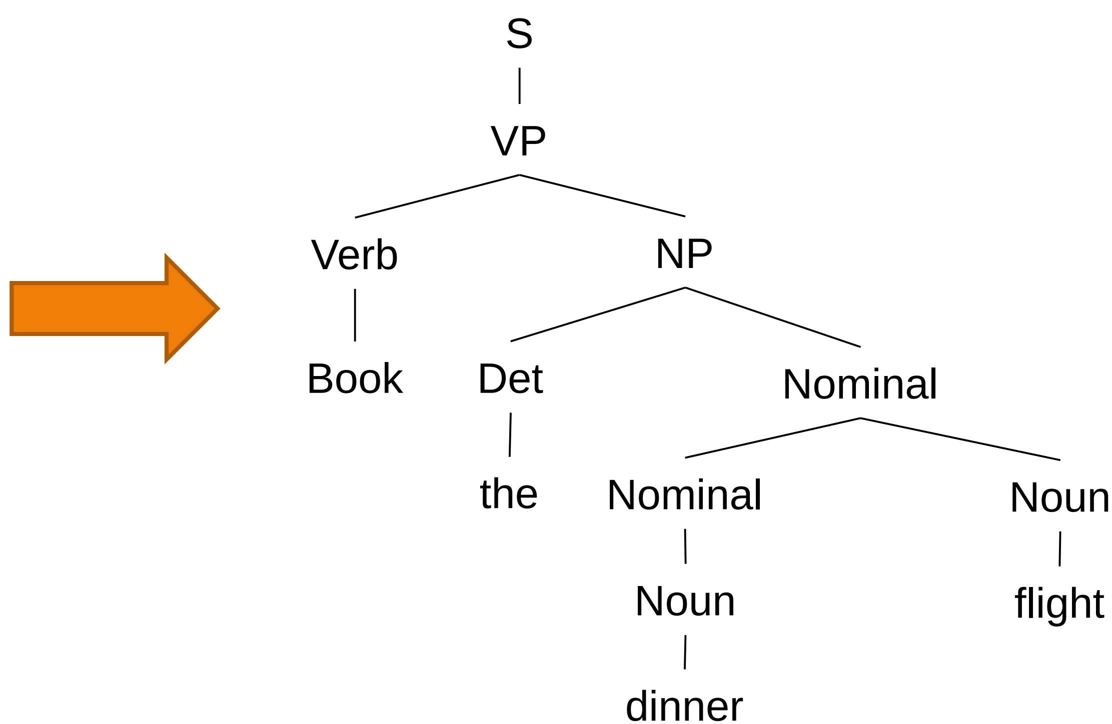
$$
\begin{aligned}
P(t_2) &= 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \\
&\quad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0006804
\end{aligned}
$$

Parsing

Parsing

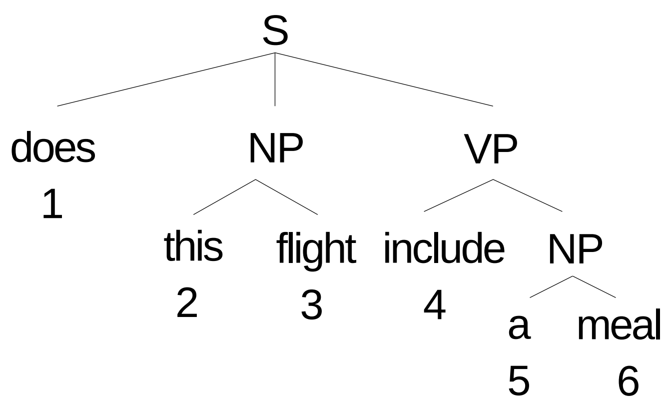
- Parsing with CFGs is the task of assigning proper parse trees to input strings

Book the dinner flight



Parser evaluation

- Represent a parse tree as a collection of tuples: $\langle (l_1, i_1, j_1), (l_2, i_2, j_2), \cdots, (l_n, i_n, j_n) \rangle$
  - l_k is the nonterminal labeling the k-th phrase
  - i_k is the index of the first word in the k-th phrase
  - j_k is the index of the last word in the k-th phrase

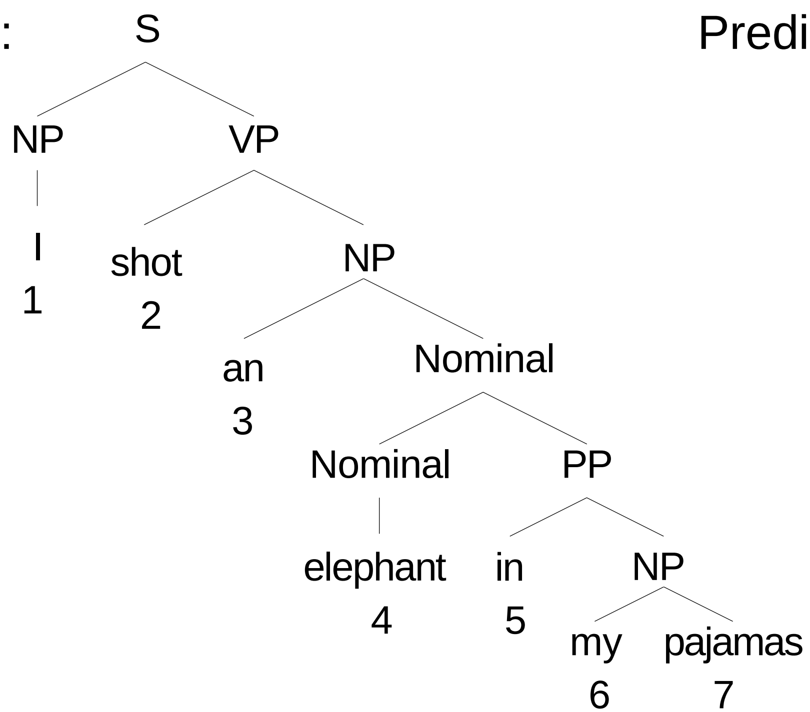


$$\rightarrow \left\{ \begin{array}{c} (S, 1, 6), (NP, 2, 3), \\ (VP, 4, 6), (NP, 5, 6) \end{array} \right\}$$

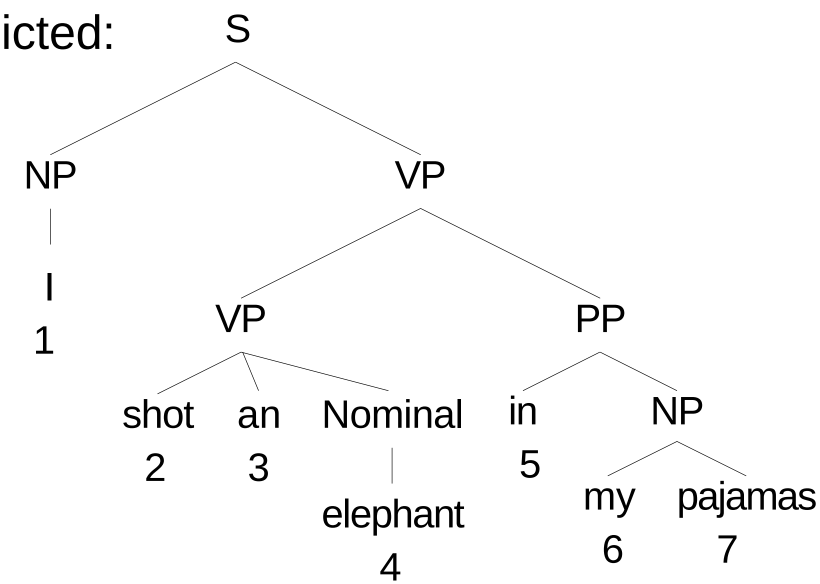
- Convert gold-standard and predicted trees into this representation, then estimate precision, recall, and F1

Tree Comparison Example

Gold:



Predicted:



$$\left\{ \begin{matrix} (NP,3,7), \\ (Norminal,4,7) \end{matrix} \right\} \left\{ \begin{matrix} (NP,1,1),(S,1,7) \\ (VP,2,7),(PP,5,7) \\ (NP,6,7),(Norminal,4,4) \end{matrix} \right\} \left\{ (VP,2,4) \right\}$$

only in gold tree

in both tree

only in predicted tree

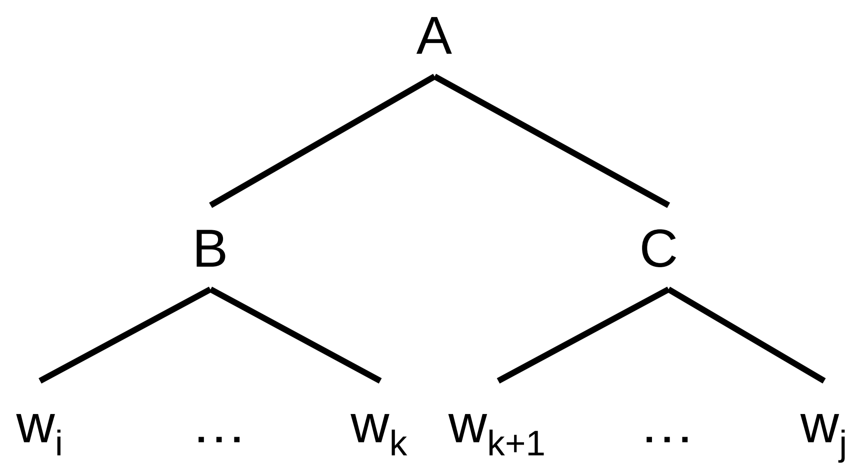
$$P = 6/7$$
$$R = 6/8$$
$$F = 4/5$$

Parsing

- A brute-force approach
  - Enumerate all parse trees consistent with the input string
- Problem
  - Number of binary trees with *n* leaves is the Catalan number C_{n-1}
  - (Exponential growth)

Parsing

- Dynamic programming
  - Divide the problem into many sub-problems
    - Sub-problem: parsing the substring between positions *i* and *j*
  - Solutions to smaller sub-problems are reused in solving larger sub-problems



Parsing algorithms

- Bottom-up DP: CYK
  - Also known as CKY
  - Applies to CFG in Chomsky Normal Form (CNF)
- Top-down DP: Earley parser
  - Applies to any CFG

Chomsky Normal Form

- Only two types of production rules

  $A \rightarrow B\ C$

  $A \rightarrow w$

- What if your grammar isn't in CNF?
- Any arbitrary CFG can be rewritten into CNF
  - The resulting grammar accepts (and rejects) the same set of strings as the original grammar
  - But the resulting parse trees are different (i.e., binarized)

Conversion to CNF

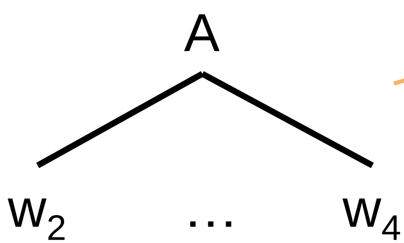
- Eliminate chains of unary productions.
  - So… A → B, B → C turns into A → C
- Introduce new intermediate non-terminals into the grammar that distribute rules with length > 2 over several rules.
  - So… S → A B C turns into S → X C and X → A B
  - X is a symbol that doesn't occur anywhere else in the grammar

CNF conversion

$\mathscr{L}_1$ Grammar	$\mathscr{L}_1$ 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

- Build a table so that a non-terminal A spanning from i to j in the input is placed in cell [i-1, j] in the table.



	1	2	3	4	5
0					
1				A	
2					
3					
4					

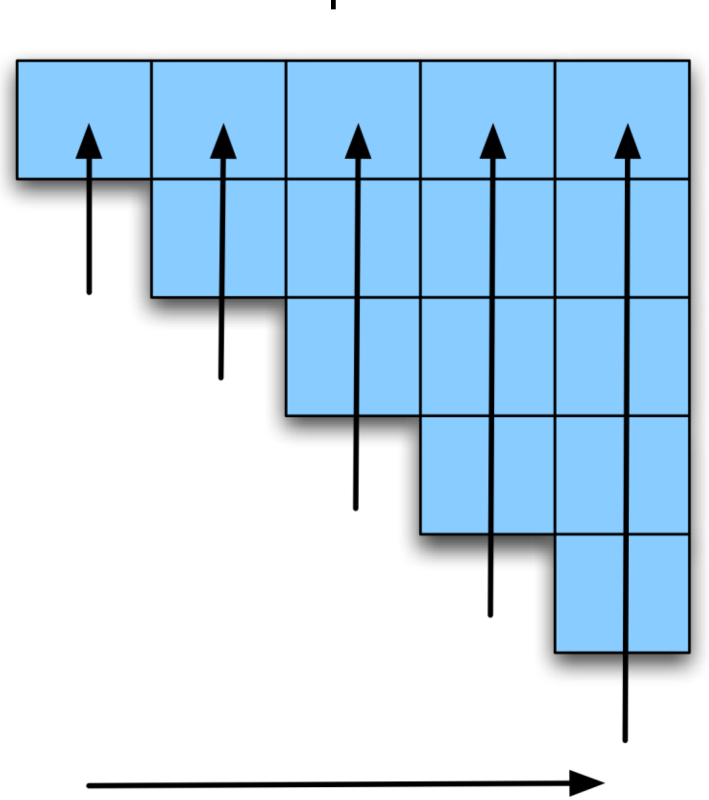
- So a non-terminal spanning an entire string will sit in cell [0, n]
  - Hopefully an *S*

Example

A completed table for input "Book the flight through Houston"

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

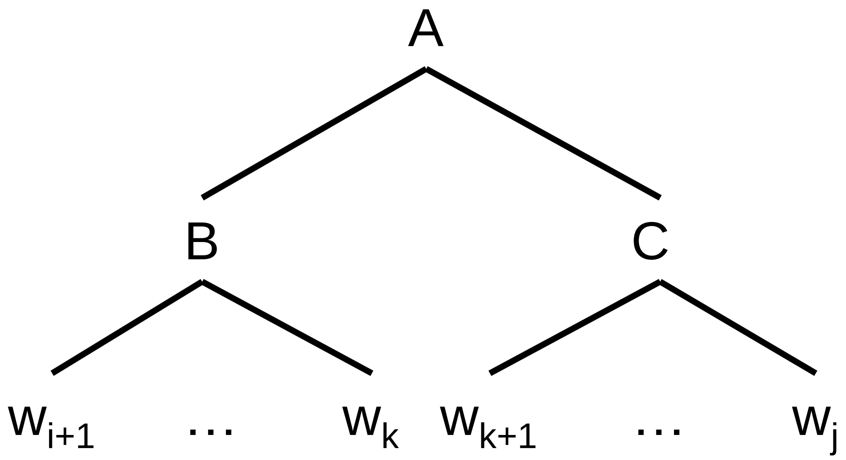
We fill the table from bottom up



CYK

- Base case:
  - A is in cell [i-1,i] iff. there exists a rule $A \rightarrow w_i$

- Recursion:
  - A is in cell [i,j] iff. for some rule $A \rightarrow B\ C$ there is a B in cell [i,k] and a C in cell [k,j] for some k.



CYK Algorithm

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

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      $table[i,j] \leftarrow table[i,j] \cup$
        $\{A \mid A \rightarrow BC \in grammar,$
          $B \in table[i,k],$
          $C \in table[k,j]\}$

- Time complexity: $O(n^3|G|)$
  - n is Sentence length and $|G|$ is number of grammar rules

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0					
1					
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det				
1					
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det				
1		N			
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3				Det	
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3				Det	
4					N

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		
3				Det	NP
4					N

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			
1		N			
2			V		VP
3				Det	NP
4					N

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			S
1		N			
2			V		VP
3				Det	NP
4					N

CYK Parsing

- Is that really a parser?
  - We want a parse tree, not a yes/no answer
- Simple changes
  - Add back-pointers so that each state knows where it came from.
  - After filling the table, recursively retrieve the constituents from the top (i.e., the start symbol) down

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

	1	2	3	4	5
0	Det	NP			S
1		N			
2			V		VP
3				Det	NP
4					N

Ambiguity

- NP → Det N
- NP → N N

	1	2	3	4	5
0	Det N	??			
1		N			
2					
3					
4					

Ambiguity

- NP → Det N
- NP → N N

	1	2	3	4	5
0	Det N	NP			
1		N			
2					
3					
4					

Ambiguity

- NP → Det N
- NP → N N

	1	2	3	4	5
0	Det N	NP			
1		N			
2					
3					
4					

Ambiguity

- NP → Det NP
- NP → NP PP

	1	2	3	4	5
0	Det	NP	??		
1		N	NP		
2			PP		
3					
4					

Ambiguity

- NP → Det NP
- NP → NP PP

	1	2	3	4	5
0	Det	NP	NP		
1		N	NP		
2			PP		
3					
4					

Ambiguity

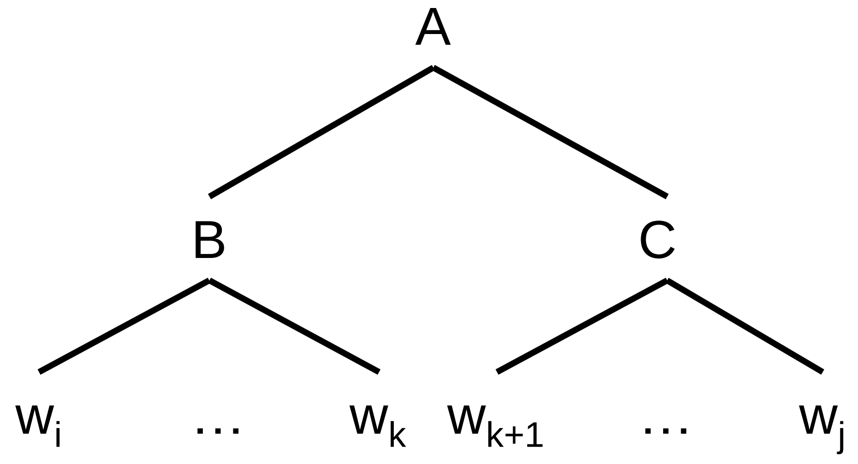
- NP → Det NP
- NP → NP PP

	1	2	3	4	5
0	Det	NP	NP		
1		N	NP		
2			PP		
3					
4					

Probabilistic Parsing

- We have a probabilistic grammar, e.g., PCFG
- We want to find the parse tree of an input string with the highest probability
- In cell [i-1,j] of the table, associate each nonterminal A with the probability of the best parse tree rooted at A covering substring from i to j
- Recursive computation

$$P_{A,i,j} = \max_{B,C,k} P(A \rightarrow BC) \times P_{B,i,k} \times P_{C,k,j}$$



CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0					
1					
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4				
1					
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4				
1		N 0.02			
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2					
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3					
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3				Det 0.4	
4					

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3				Det 0.4	
4					N 0.01

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		
3				Det 0.4	NP 0.001
4					N 0.01

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			
1		N 0.02			
2			V .05		VP .00001
3				Det 0.4	NP 0.001
4					N 0.01

CYK

- *The flight includes a meal.*

Grammar

- S → NP VP [.80]
- NP → Det N [.30]
- VP → V NP [.20]
- V → includes [.05]
- Det → the [.4]
- Det → a [.4]
- N → meal [.01]
- N → flight [.02]

	1	2	3	4	5
0	Det 0.4	NP .0024			S .0000000192
1		N 0.02			
2			V .05		VP .00001
3				Det 0.4	NP 0.001
4					N 0.01

Ambiguity

- NP → Det N [0.7]
- NP → N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8				
1		N 0.02			
2					
3					
4					

Ambiguity

- NP → Det N [0.7]
- NP → N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8	NP .0056			
1		N 0.02			
2					
3					
4					

Ambiguity

- NP → Det N [0.7]
- NP → N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8	NP .0048			
1		N 0.02			
2					
3					
4					

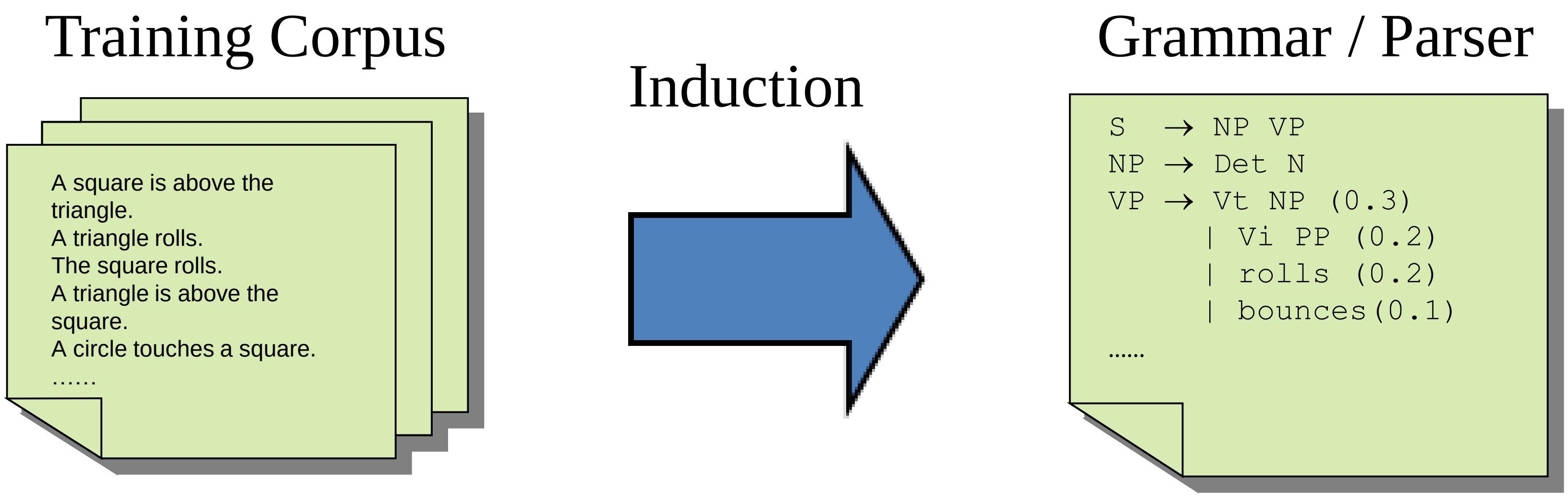
Ambiguity

- NP → Det N [0.7]
- NP → N N [0.3]

	1	2	3	4	5
0	Det 0.4 N 0.8	NP .0056			
1		N 0.02			
2					
3					
4					

Learning

Learning a grammar from a corpus



- Supervised Methods
  - Rely on a training corpus of sentences annotated with parses (treebank)
- Unsupervised Methods (Grammar Induction)
  - Do not require annotated data

Supervised Methods

- Treebank
  - A corpus in which each sentence has been (manually) paired with a parse tree
  - Manual annotation is labor intensive and generally requires linguistic knowledge and detailed guidelines.
  - Most well known is the Wall Street Journal section of the Penn TreeBank.
    - 1 M words from the 1987-1989 Wall Street Journal

Generative Methods

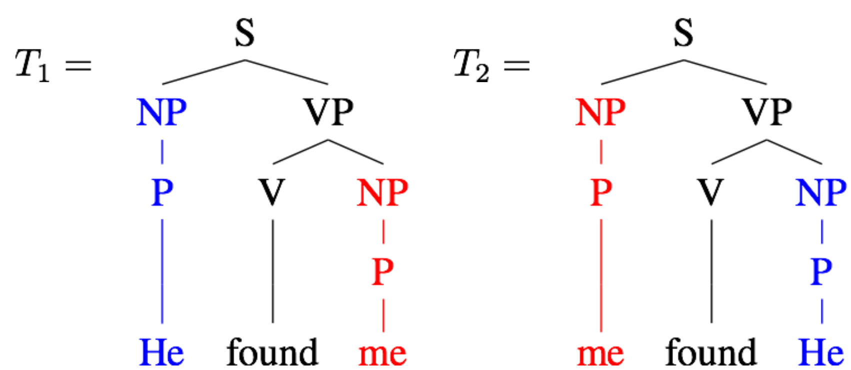
- Learning a PCFG from treebanks
  - Maximum likelihood estimation (treebank grammar)
    - We maximize P(sentence, parse)
    - Closed-form solution: count and normalize

	Counts in treebank	MLE of rule probabilities
VP→Verb	20	0.2
VP→Verb NP	40	0.4
VP→Verb NP NP	25	0.25
VP→Verb PP	15	0.15

Generative Methods

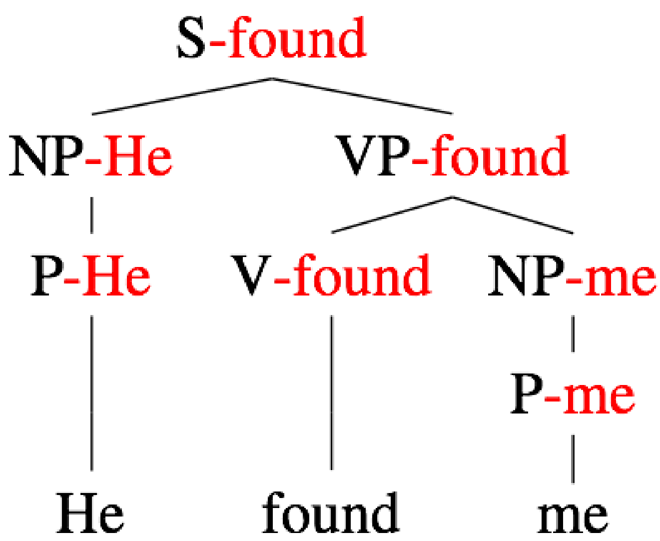
- Learning a PCFG from treebanks
  - Maximum likelihood estimation (treebank grammar)
  - MLE has bad performance (F1 score <80)
    - Main reason: standard treebank nonterminals are not sufficiently informative

$S \rightarrow NP\ VP$ 1
$VP \rightarrow V\ NP$ 0.2
$NP \rightarrow DT\ NP$ 0.5
$NP \rightarrow NP\ NP$ 0.3
… …
$P \rightarrow He$ 0.2
$P \rightarrow me$ 0.2
$V \rightarrow found$ 0.1
… …



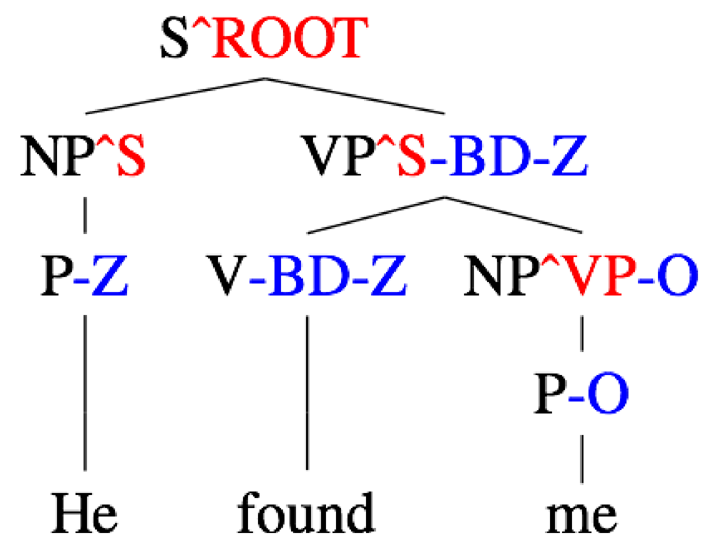
$$P(T_1) = P(T_2)$$

Generative Methods beyond MLE



Lexicalization

[Collins. 1997; Charniak. 2000]



Tree Annotation

[Johnson. 1998; Klein et al. 2003]

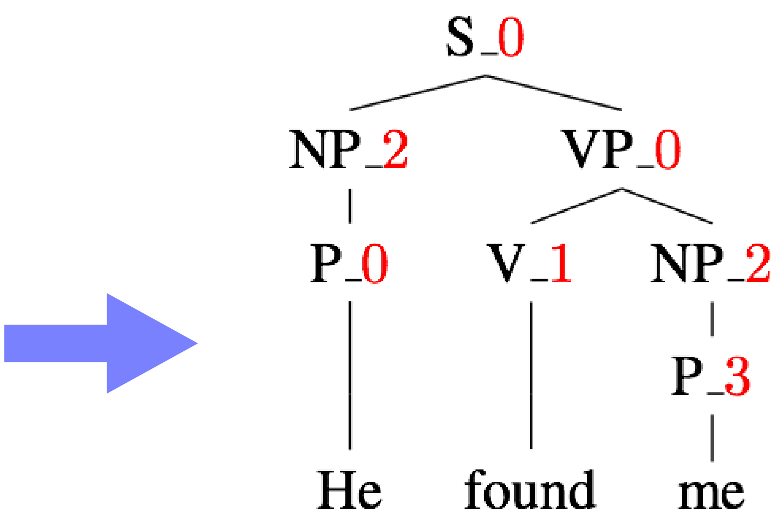
- Problems
  - Each rule has less training data. Smoothing is very important!
  - Need to do additional annotations or design rules

Generative Methods beyond MLE

- Latent Variable Grammars [Matsuzaki et al. 2005; Petrov et al. 2007]
  - Each nonterminal is split into a finite number of subtypes
  - Each subtype rule is associated with a probability
  - Learning by EM
    - the nonterminal subtypes in the training parse trees are the hidden variables

Subtype rule	*Prob.*
S[0] → NP[0] VP[0]	0.1
S[1] → NP[0] VP[0]	0.15
S[0] → NP[1] VP[0]	0.05
S[1] → NP[1] VP[0]	0.08

… …



A subtype parse tree

Discriminative Methods

- We assume a weighted context-free grammar and maximize conditional likelihood P(gold parse | sentence)

$$P(t|x) = \frac{\prod_{r \in (t,x)} \exp\big(W(r)\big)}{Z(x)}$$

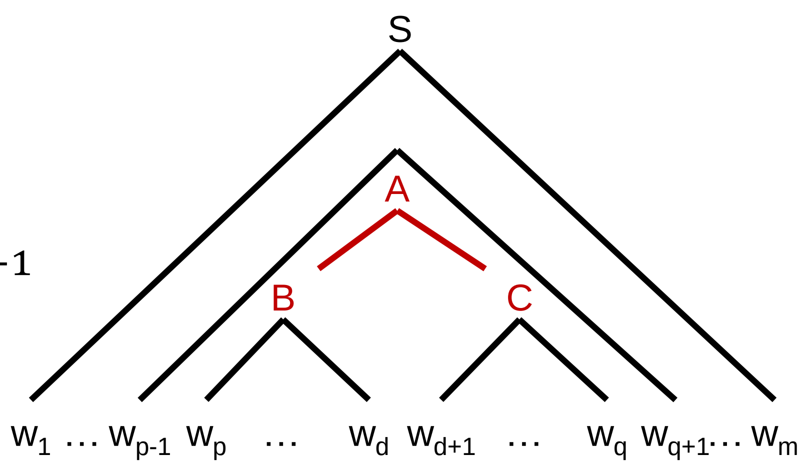
- We formulate the weight of a rule based on the input sentence
  - This can be viewed as having a different set of subtypes for each sentence
  - More expressive than earlier generative methods
- This is a CRF!

Discriminative Methods

- We formulate the weight of a rule based on the input sentence
  - …from features of the (anchored) rule in the sentence

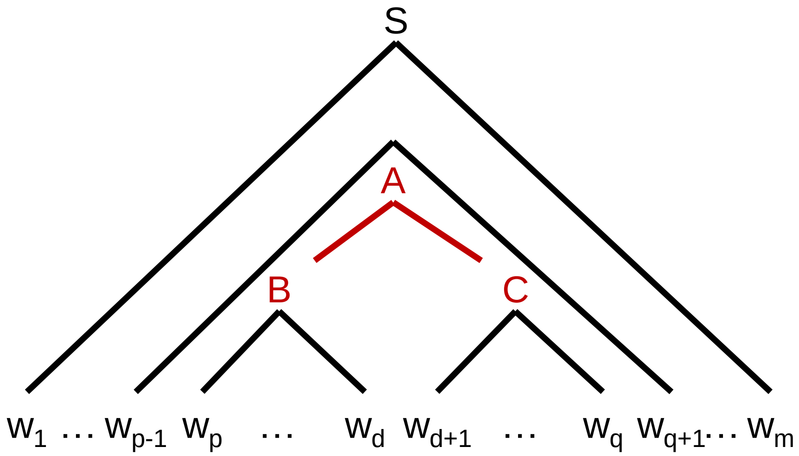
$$W(r: A \to BC, \langle p, q, d \rangle) = \sum_i \alpha_i f_i(r, \langle p, q, d \rangle)$$

- Possible features:
  - Rule identity $A \to BC$
  - Words at the span boundary $w_{p-1}, w_p, w_q, w_{q+1}$
  - Words at the split point w_d, w_{d+1}



Discriminative Methods

- We formulate the weight of a rule based on the input sentence
  - …from features of the (anchored) rule in the sentence
  - …using a neural network with word embeddings at the span boundary and split point as input



Discriminative Methods

- Maximizing conditional likelihood with gradient descent

$$P(t|x) = \frac{\prod_{r \in (t,x)} \exp\big(W(r)\big)}{Z(x)}$$

  - Partition function $Z(x)$ is computed with the inside algorithm (to be introduced next)

- Alternative objective: margin-based loss

Unsupervised Methods

- Learning from sentences with no parse tree annotation
  - …sometimes with POS annotation

- Two tasks
  - Structure search
    - Try to find an optimal set of grammar rules
  - Parameter learning
    - Given a set of grammar rules, try to learn their probabilities

Structure search

- Two classes of approaches
  - Heuristic approaches
    - Create nonterminals and production rules using heuristic criteria and rules
      - Ex: Frequency of a substring, substitutability heuristic, distributional clustering, …
  - Optimization-based approaches
    - Optimizing an explicit objective function (e.g., posterior) of the grammar structure by local search
      - Start with a trivial grammar
      - Search with a set of structure-change operations
- Empirical results
  - Poor on real data, often below simple baselines 😟

Parameter learning

- Maximum marginal likelihood estimation
  - We maximize P(sentence) with the parse tree marginalized
- Learning algorithm?
  - Expectation-maximization!
    - E-step: Compute the distribution of the parse tree for each training sentence
      - Infeasible to enumerate. But can compute expected counts of rule usage using the inside-outside algorithm.
    - M-step: Update the parameters to maximize expected log likelihood based on the distribution over the parse tree
      - Closed-form solution: normalize the expected counts

Inside-outside algorithm

- Assume the grammar is in the Chomsky normal form (CNF)
  - Only two types of rules
    - $N_1 \rightarrow N_2 N_3$
    - $N_1 \rightarrow w$

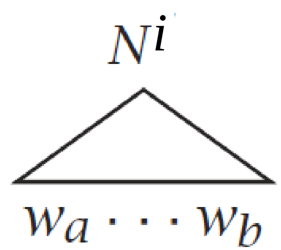
Inside-outside algorithm

- Notations

Sentence: sequence of words $w_1 \cdots w_m$

w_{ab}: the subsequence $w_a \cdots w_b$

N^i_{ab}: nonterminal N^i dominates $w_a \cdots w_b$



Span index:

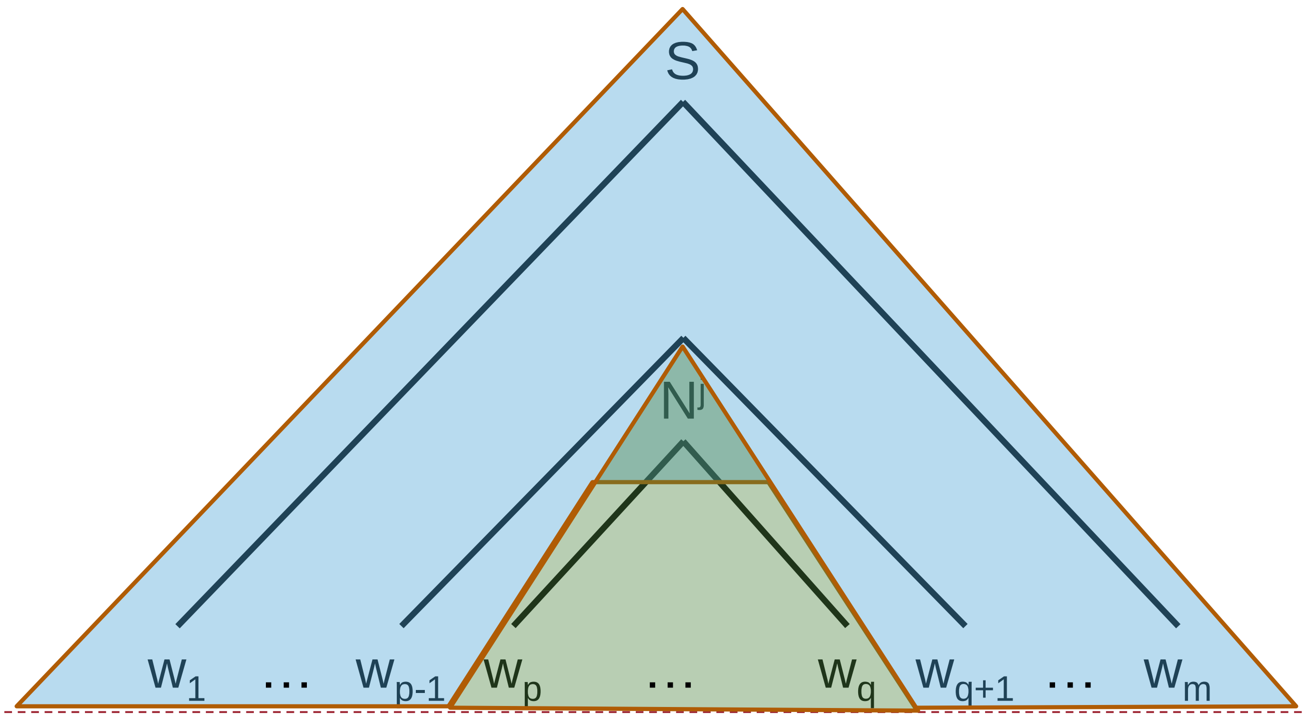
- Here we use [inclusive, inclusive]
- When introducing CYK, we use [exclusive, inclusive], i.e., $(a-1, b)$ for the same span

Inside-outside algorithm

- Given an input string, compute two types of probabilities

$$\text{Outside} = \alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} | G)$$

$$\text{Inside} = \beta_j(p,q) = P(w_{pq} | N^j_{pq}, G)$$



Computing inside probabilities

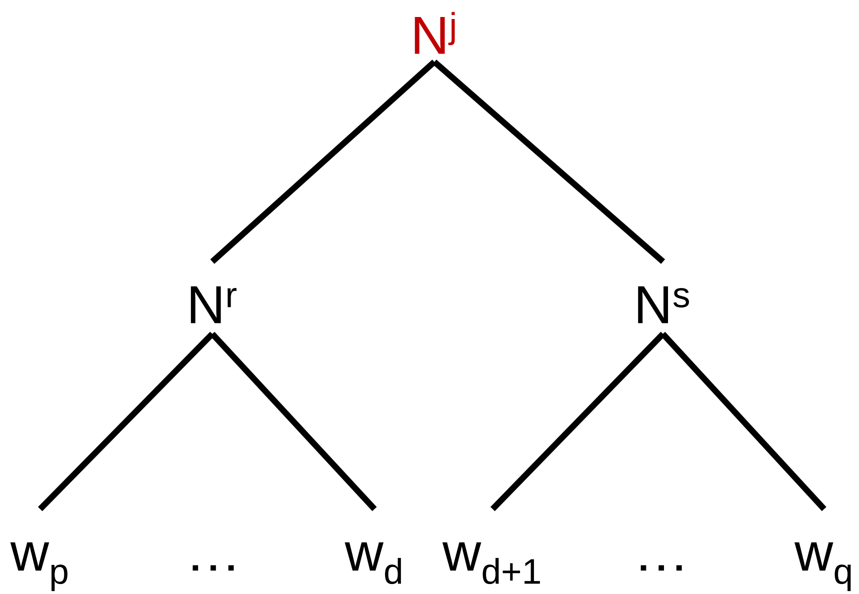
- Base case

$$\begin{aligned}\beta_j(k,k) &= P(w_k \mid N^j_{kk}, G)\\ &= P(N^j \to w_k \mid G)\end{aligned}$$

Computing inside probabilities

- Bottom-up recursion

$$\begin{aligned}\beta_j(p,q) &= P(w_{pq} \mid N^j_{pq}, G) \\ &= \sum_{r,s} \sum_{d=p}^{q-1} P(N^j \to N^r N^s)\beta_r(p,d)\beta_s(d+1,q)\end{aligned}$$



Computing inside probabilities

- Almost the same as CYK parsing, but uses sum instead of max
- Looks familiar?
  - HMM
    - Viterbi algorithm uses max
    - Forward algorithm uses sum
  - HMM is a special case of PCFG
    - Viterbi algorithm is a special case of CYK algorithm
    - Forward algorithm is a special case of inside algorithm

Computing outside probabilities

$$\text{Outside} = \alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} | G)$$

- Base case

(Nonterminal #1 is S)

$$\alpha_1(1,m) = 1$$
$$\alpha_j(1,m) = 0, \text{ for } j \neq 1$$

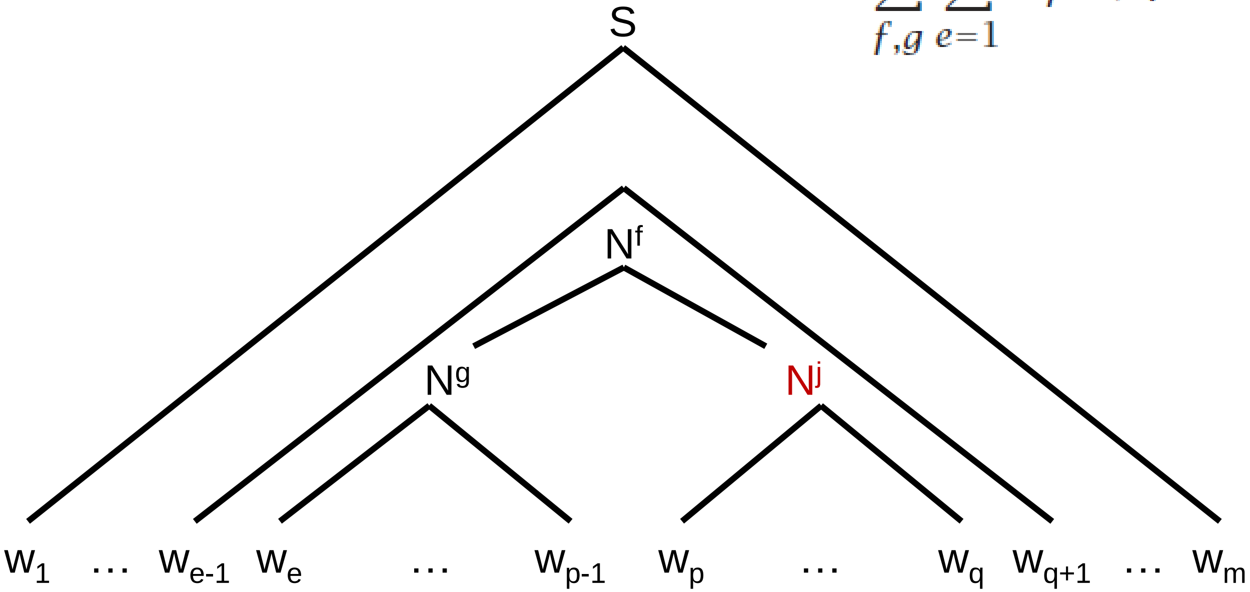
Computing outside probabilities

- Top-down recursion

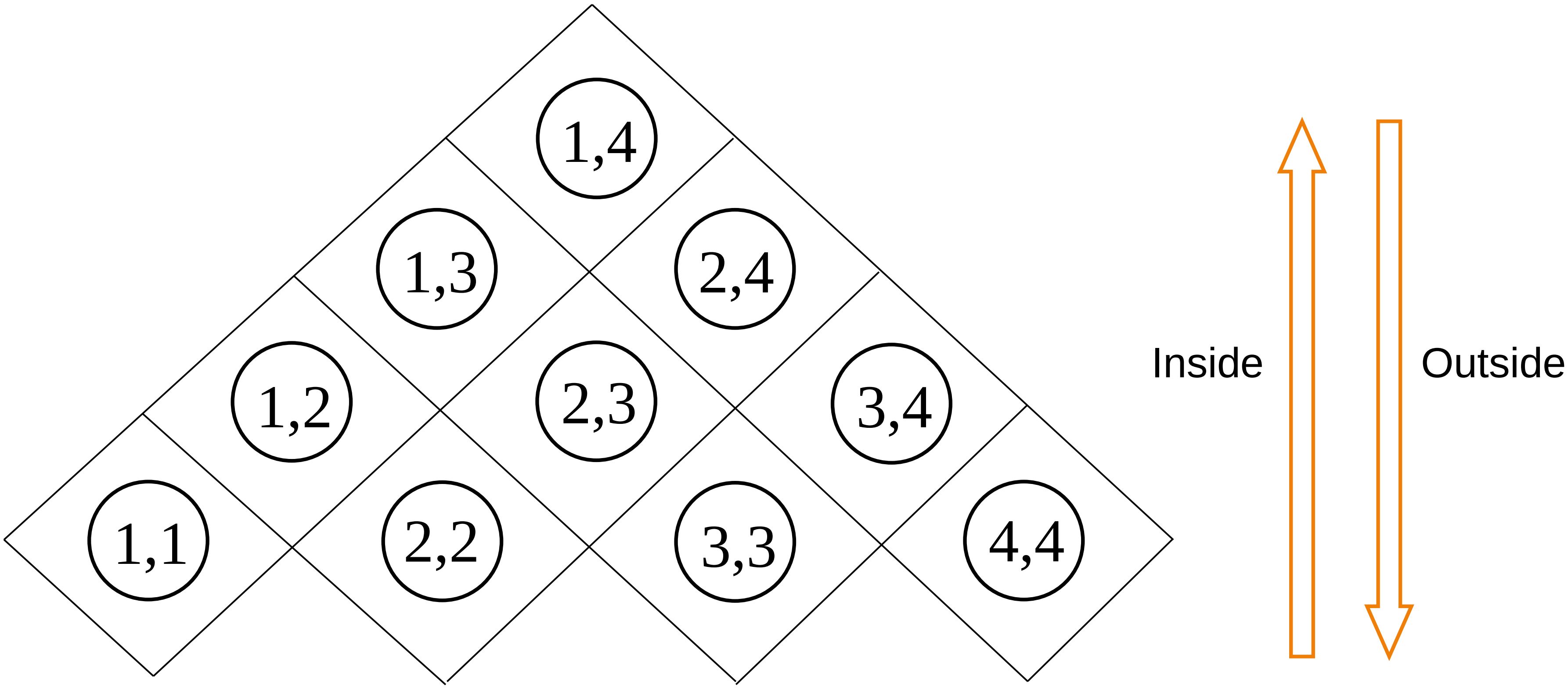
$$\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} | G)$$

$$= [\sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p,e) P(N^f \rightarrow N^j\ N^g) \beta_g(q+1,e)]$$

$$+ [\sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e,q) P(N^f \rightarrow N^g\ N^j) \beta_g(e,p-1)]$$



Inside-outside algorithm



- Time complexity: $O(n^3|G|)$
  - n is Sentence length and $|G|$ is number of grammar rules

Inside-outside algorithm

- Expectation-maximization (EM)
  - Initialize the probabilities (e.g., randomly)
  - Repeat until convergence
    - E-step: compute the expected counts

$$C\left(N^j \rightarrow N^r N^s \text{ used} \middle| X, \Theta^t\right)$$

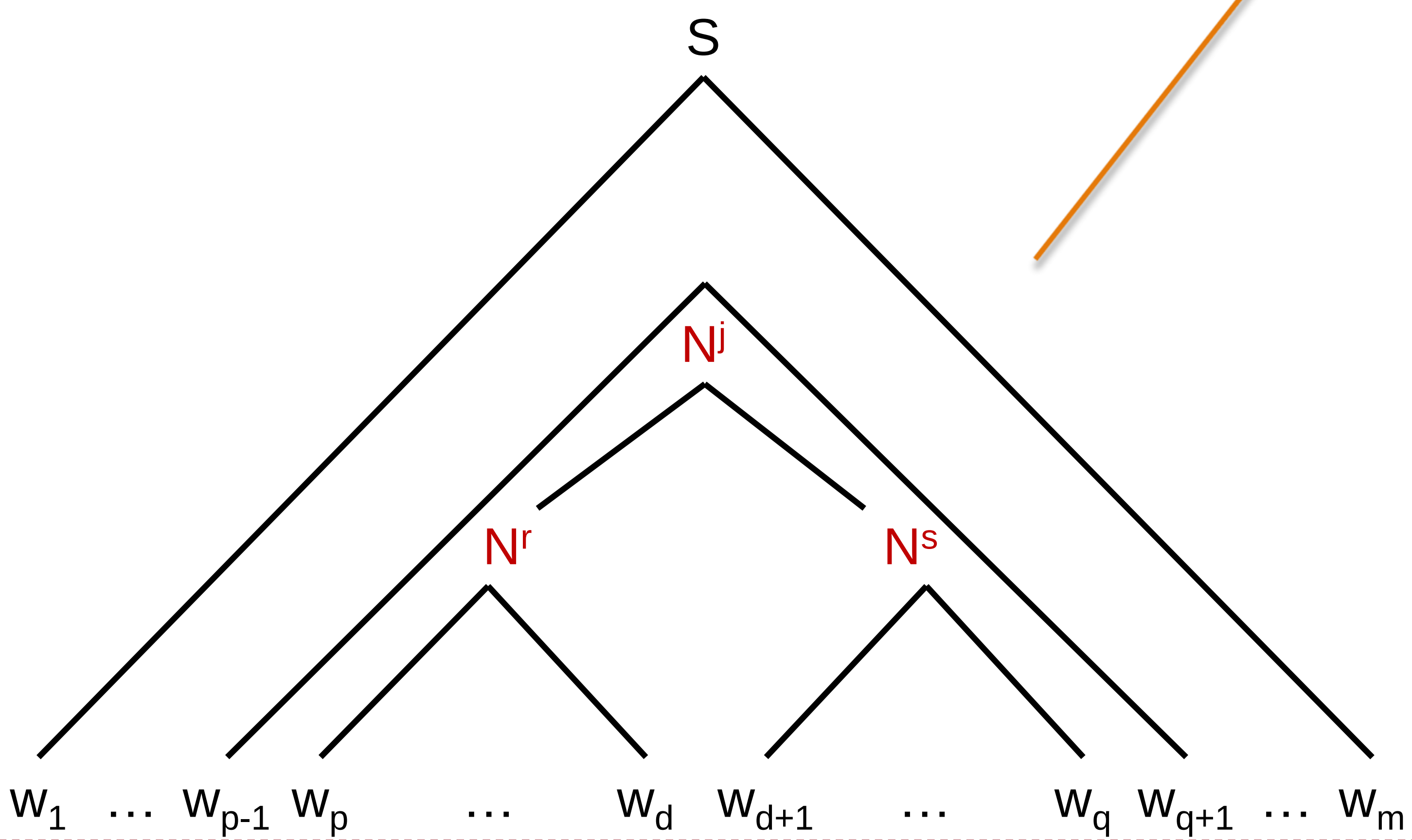
$$= \sum_{w_{1:m} \in X} E_{p(t|w_{1:m})}\left[C\left(N^j \rightarrow N^r N^s \text{ used in } t\right)\right]$$

    - M-step: update the probabilities by normalizing expected counts

$$\theta_{jrs}^{t+1} = P(N^j \rightarrow N^r N^s) = \frac{C(N^j \rightarrow N^r N^s \text{ used}|X, \Theta^t)}{C(N^j \text{ used}|X, \Theta^t)}$$

Expected counts

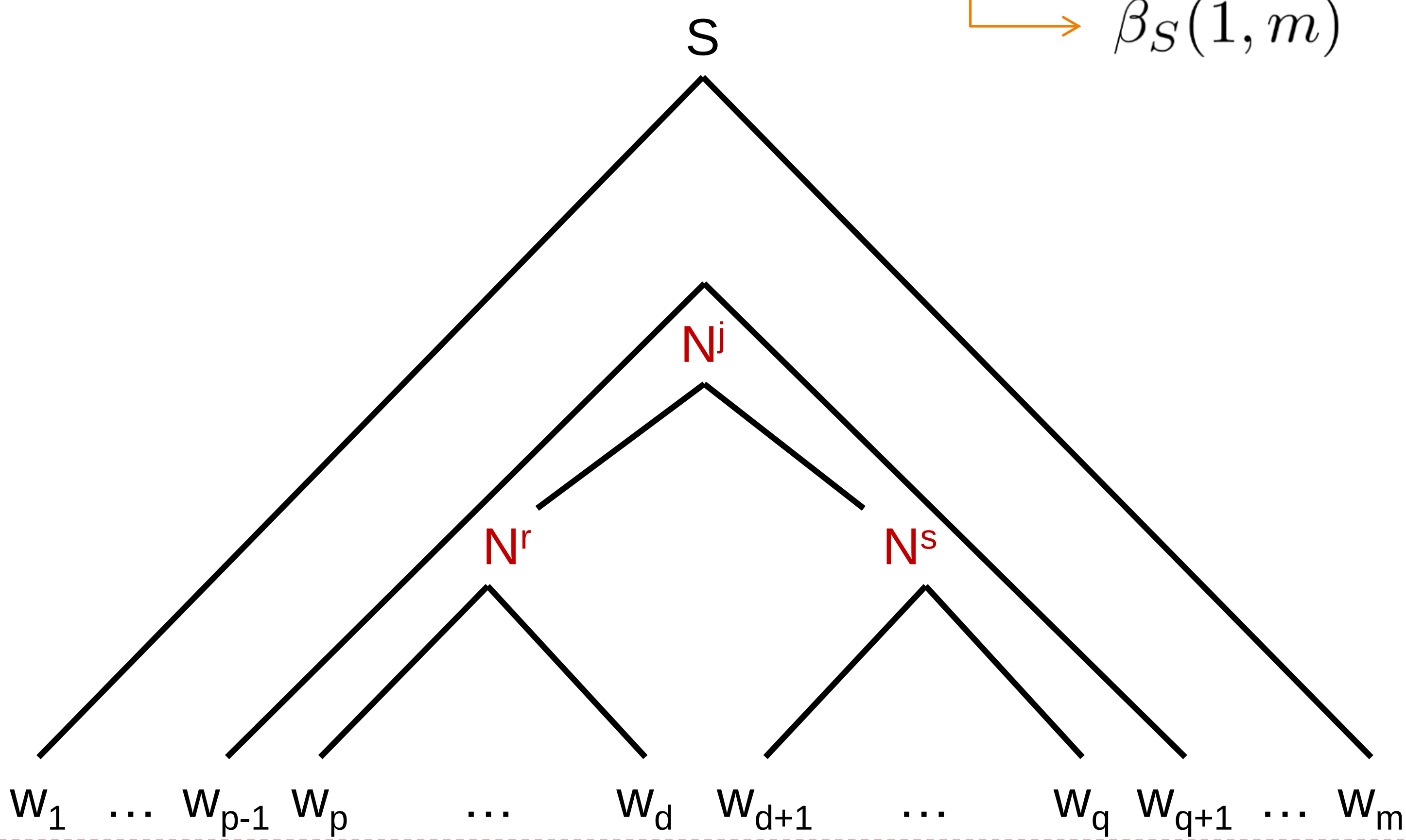
$$C(N^j \rightarrow N^r N^s \text{ used}|X, \Theta^t) = \sum_{w_{1,m} \in X} \sum_{p=1}^{m-1} \sum_{q=p+1}^{m} \sum_{d=p}^{q-1} P(\ldots|w_{1,m}, \Theta^t)$$



Expected counts

$$P(\ldots|w_{1,m},\Theta^t) = \frac{\alpha_j(p,q)P(N^j \to N^r N^s)\beta_r(p,d)\beta_s(d+1,q)}{\boxed{P(w_{1,m})}}$$

$$\beta_S(1,m)$$



Inside-outside algorithm

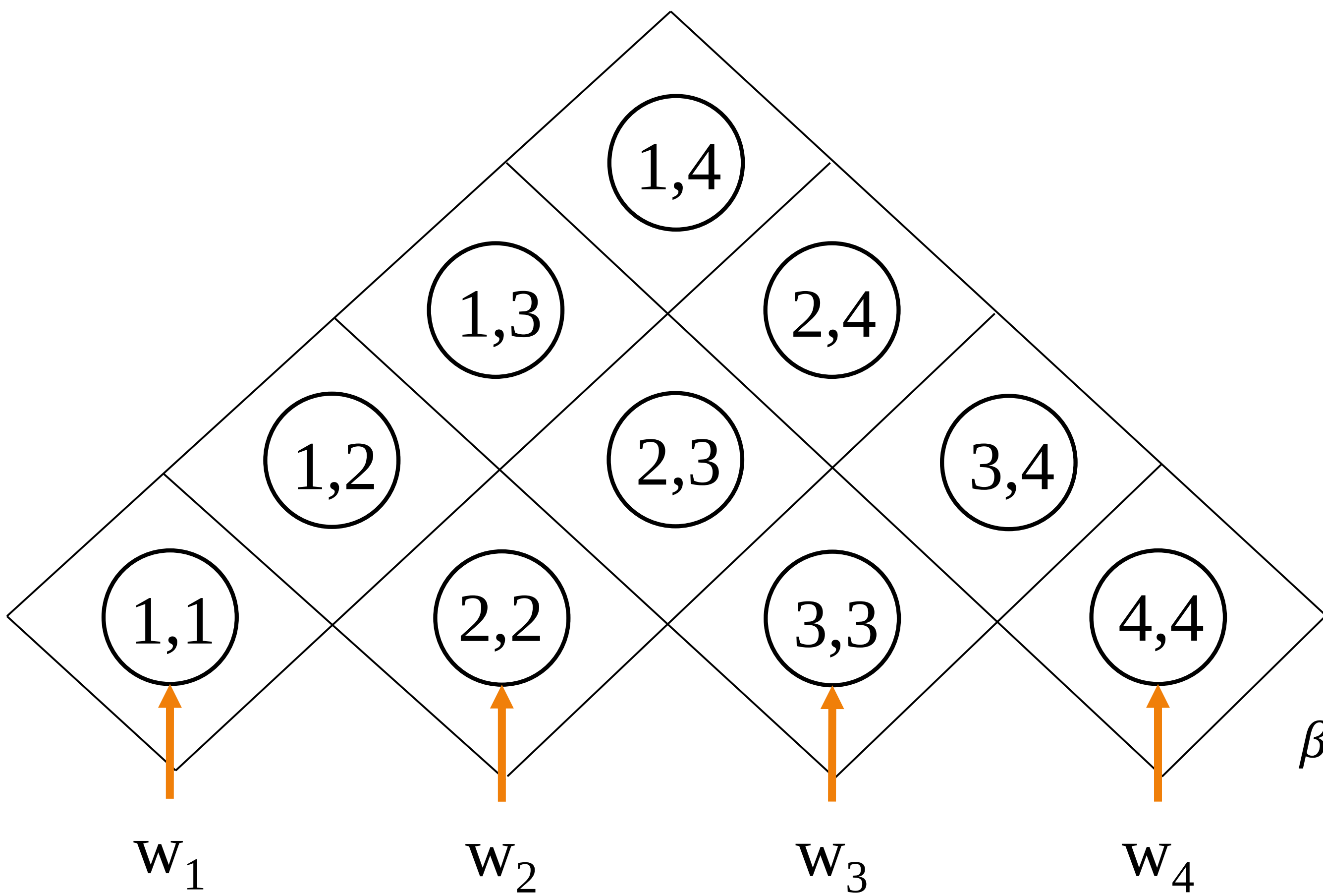
- Expectation-maximization (EM)
  - Initialize the probabilities (e.g., randomly)
  - Repeat until convergence
    - E-step
      - Compute the inside probabilities
      - Compute the outside probabilities
      - Compute the expected counts
    - M-step
      - Update the probabilities

Inside-outside are just backprop!

- Expected counts can be computed by backprop
  - $C\left(N^j \rightarrow N^r N^s \text{ used} \middle| X, \Theta^t\right) = \frac{\partial \log P(w_{1:m})}{\partial \Theta^t_{j,r,s}}$
  - The inside and then backprop procedure is almost the same as Inside-Outside

- See https://aclanthology.org/W16-5901.pdf

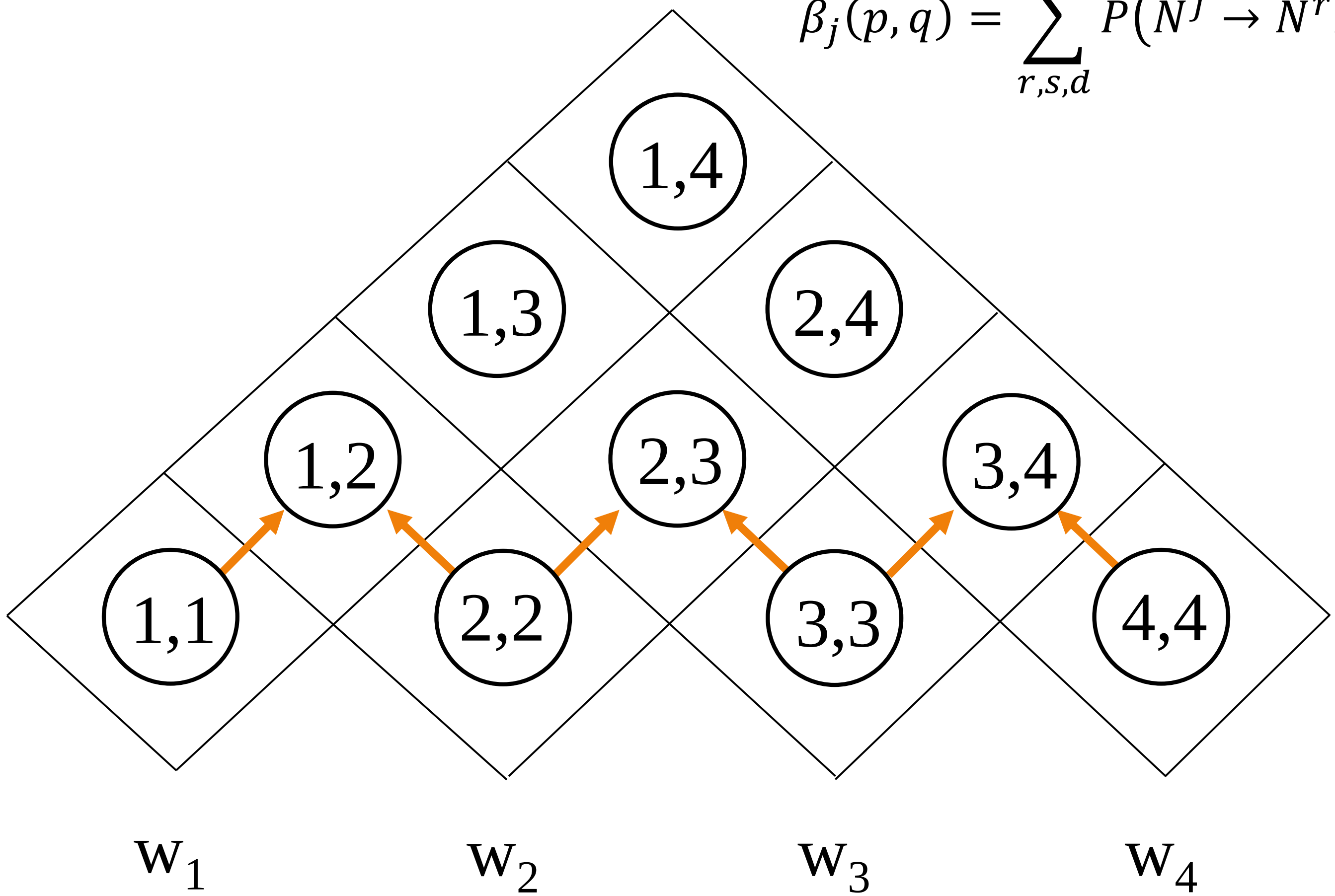
Computation graph of the inside algorithm



$$\beta_j(i,i) = P(N^j \rightarrow w_i)$$

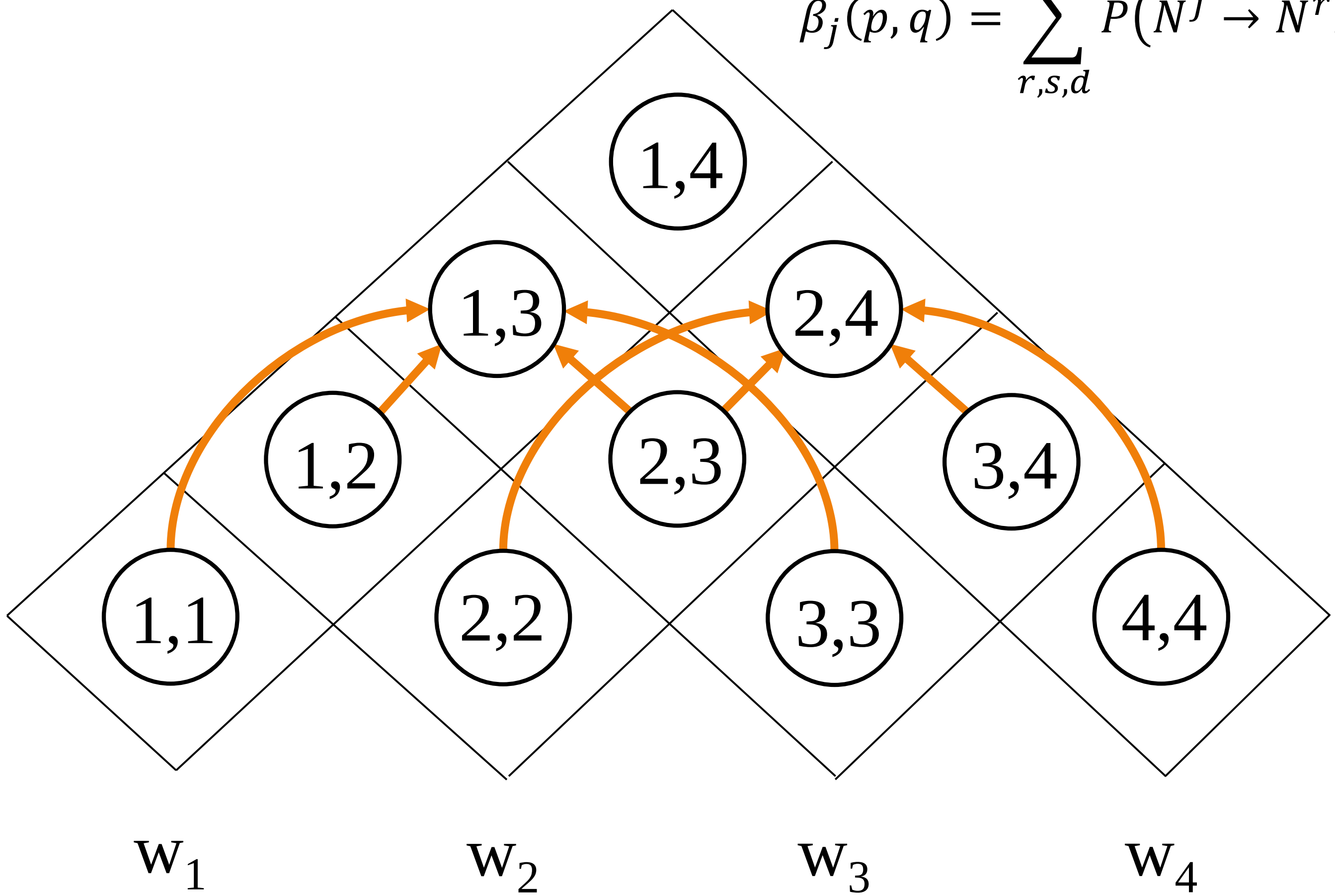
Computation graph of the inside algorithm

$$\beta_j(p,q) = \sum_{r,s,d} P(N^j \rightarrow N^r N^s)\beta_r(p,d)\beta_s(d+1,q)$$



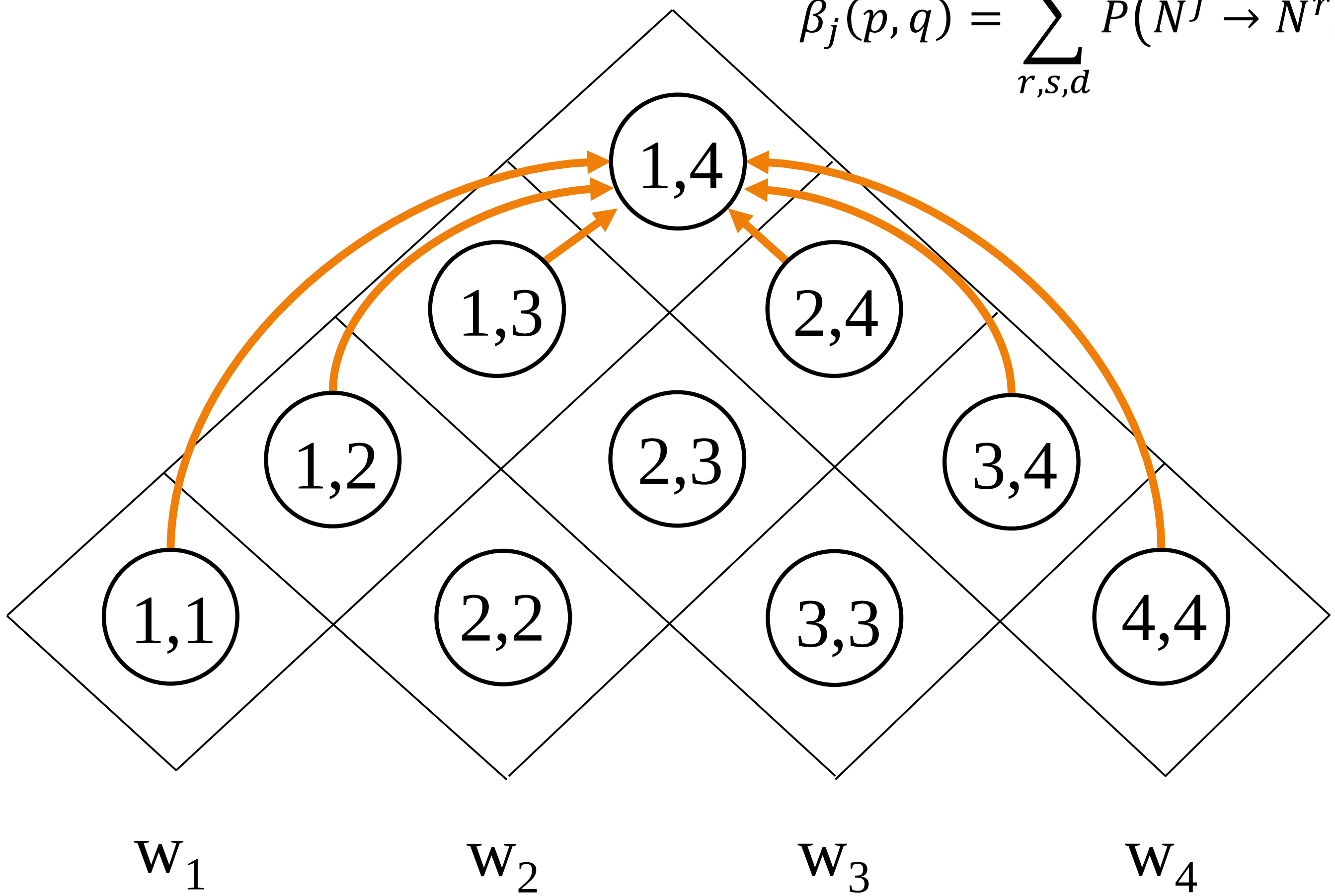
Computation graph of the inside algorithm

$$\beta_j(p,q) = \sum_{r,s,d} P\left(N^j \rightarrow N^r N^s\right)\beta_r(p,d)\beta_s(d+1,q)$$

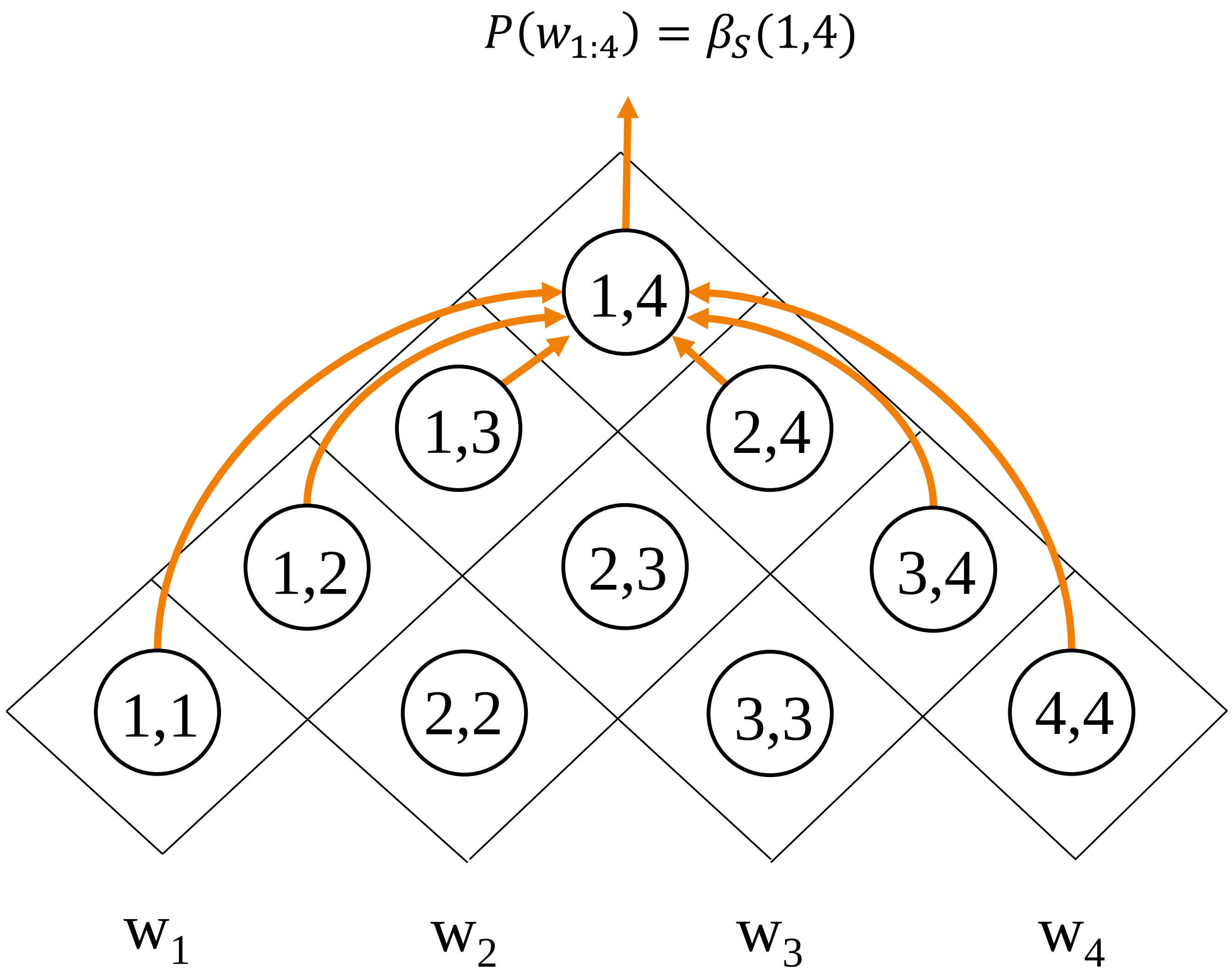


Computation graph of the inside algorithm

$$\beta_j(p,q) = \sum_{r,s,d} P(N^j \to N^r N^s)\beta_r(p,d)\beta_s(d+1,q)$$



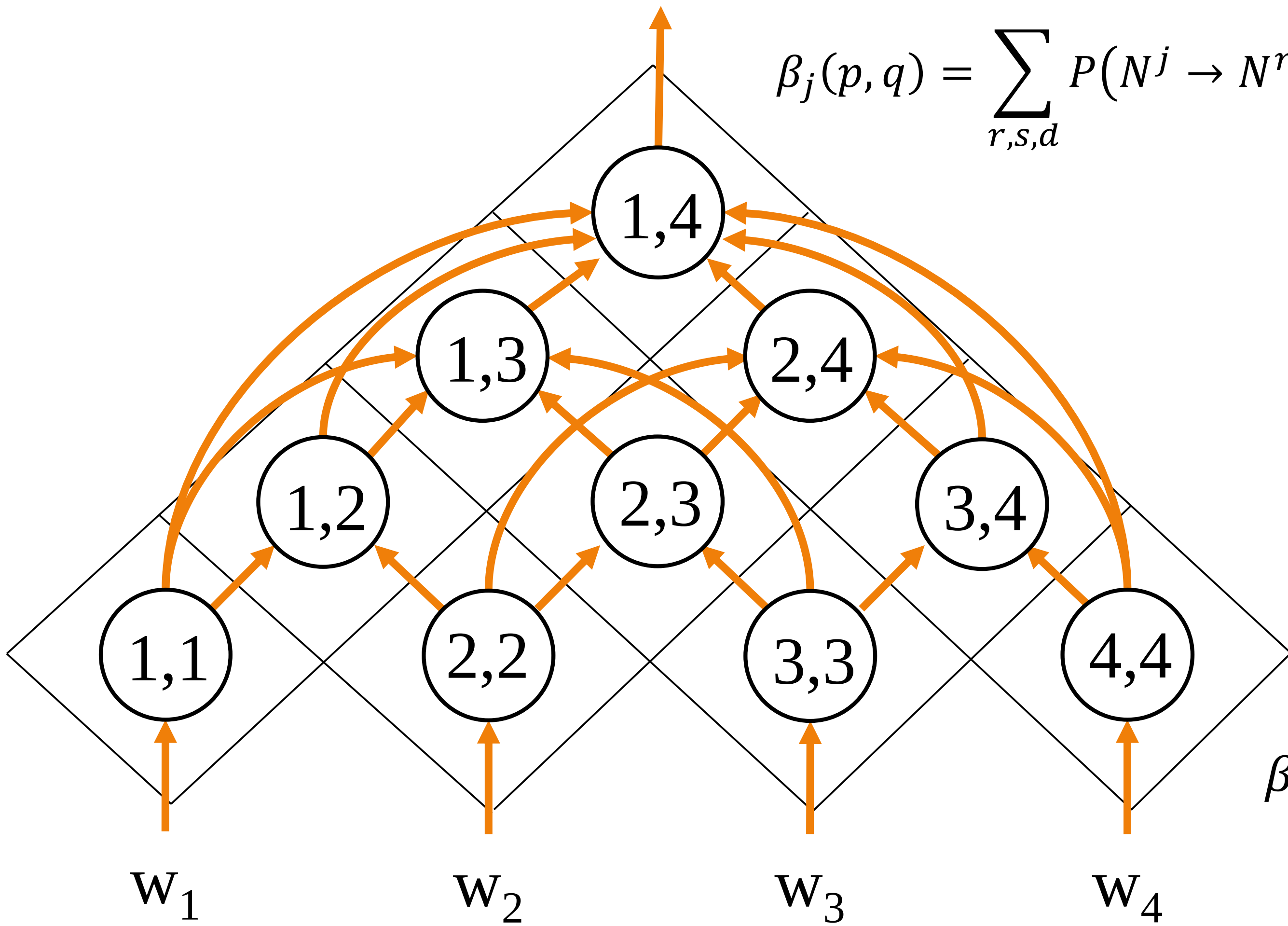
Computation graph of the inside algorithm



Computation graph of the inside algorithm

$$P(w_{1:4}) = \beta_S(1,4)$$

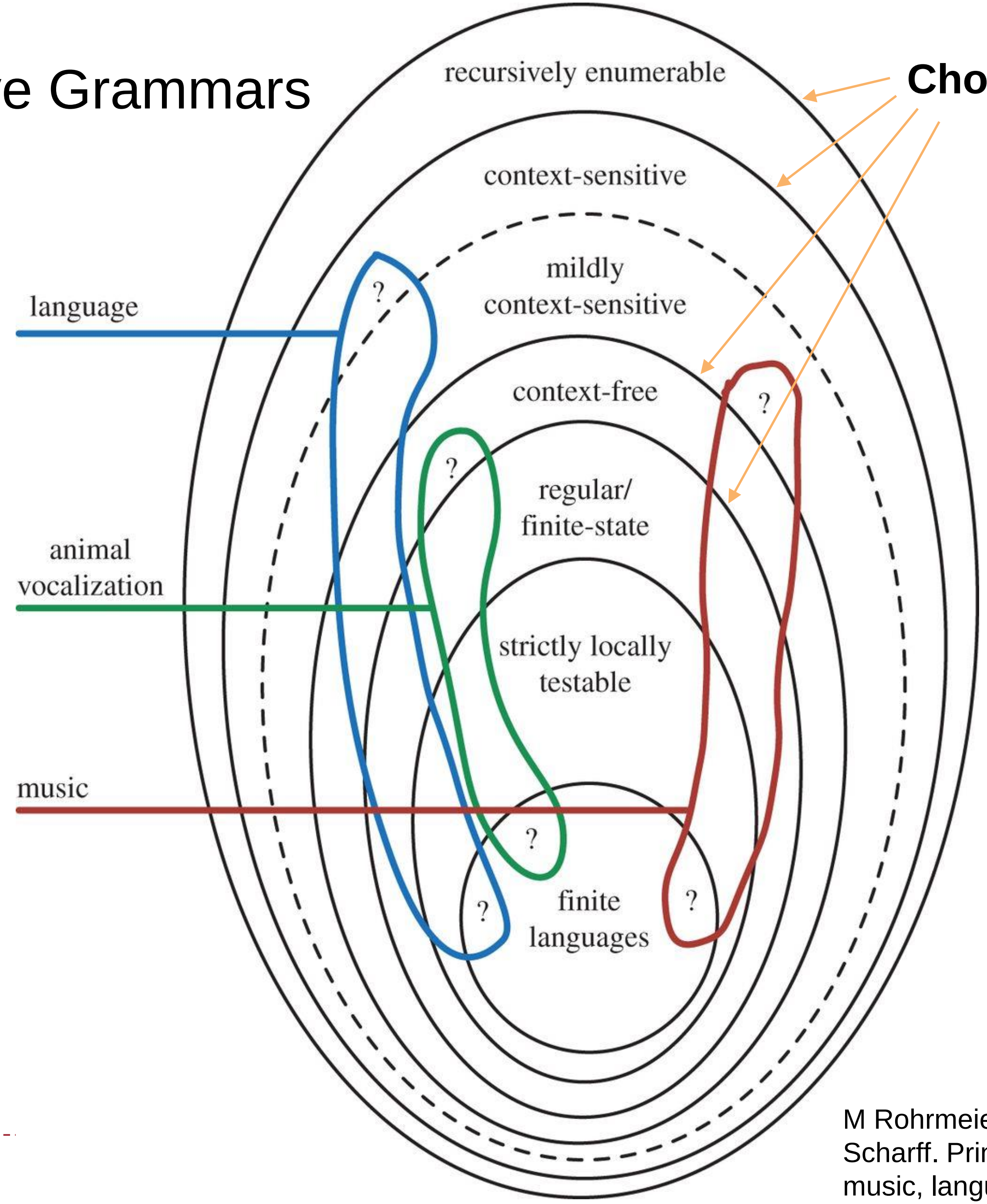
$$\beta_j(p,q) = \sum_{r,s,d} P\left(N^j \rightarrow N^r N^s\right)\beta_r(p,d)\beta_s(d+1,q)$$

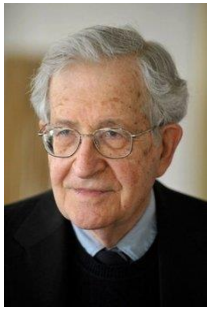


$$\beta_j(i,i) = P(N^j \rightarrow w_i)$$

Beyond CFG

Generative Grammars

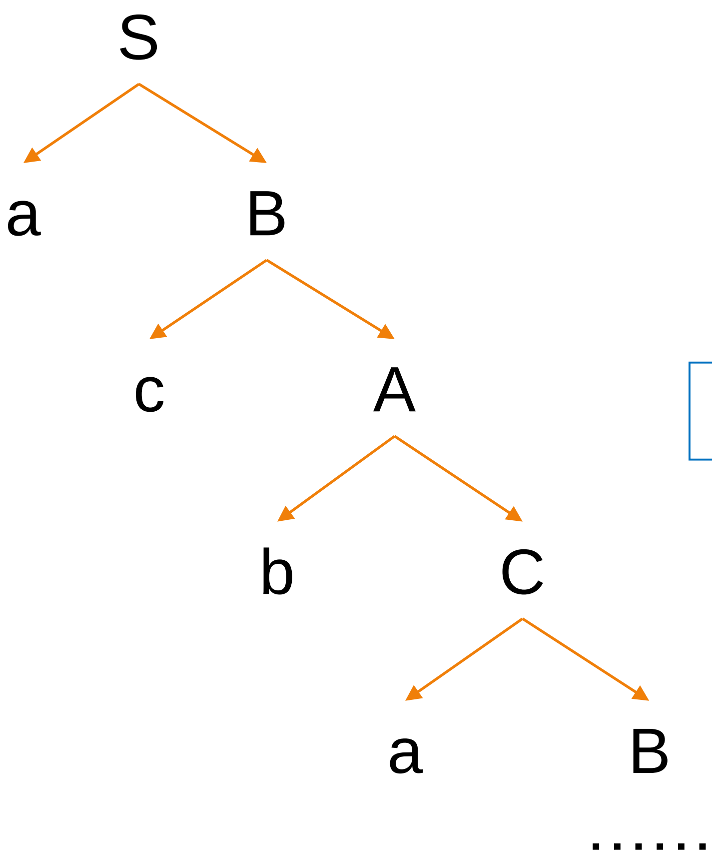


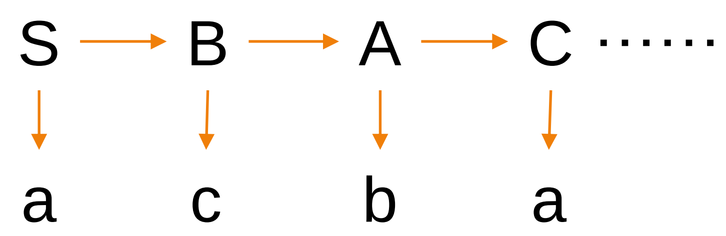


M Rohrmeier, W Zuidema, G Wiggins, C Scharff. Principles of structure building in music, language and animal song.

Regular Grammars

- Production rules are of the form $A \rightarrow aB$ *or* $A \rightarrow a$





HMM = Probabilistic RG ⊂ CFG

Viterbi : Forward-Backward

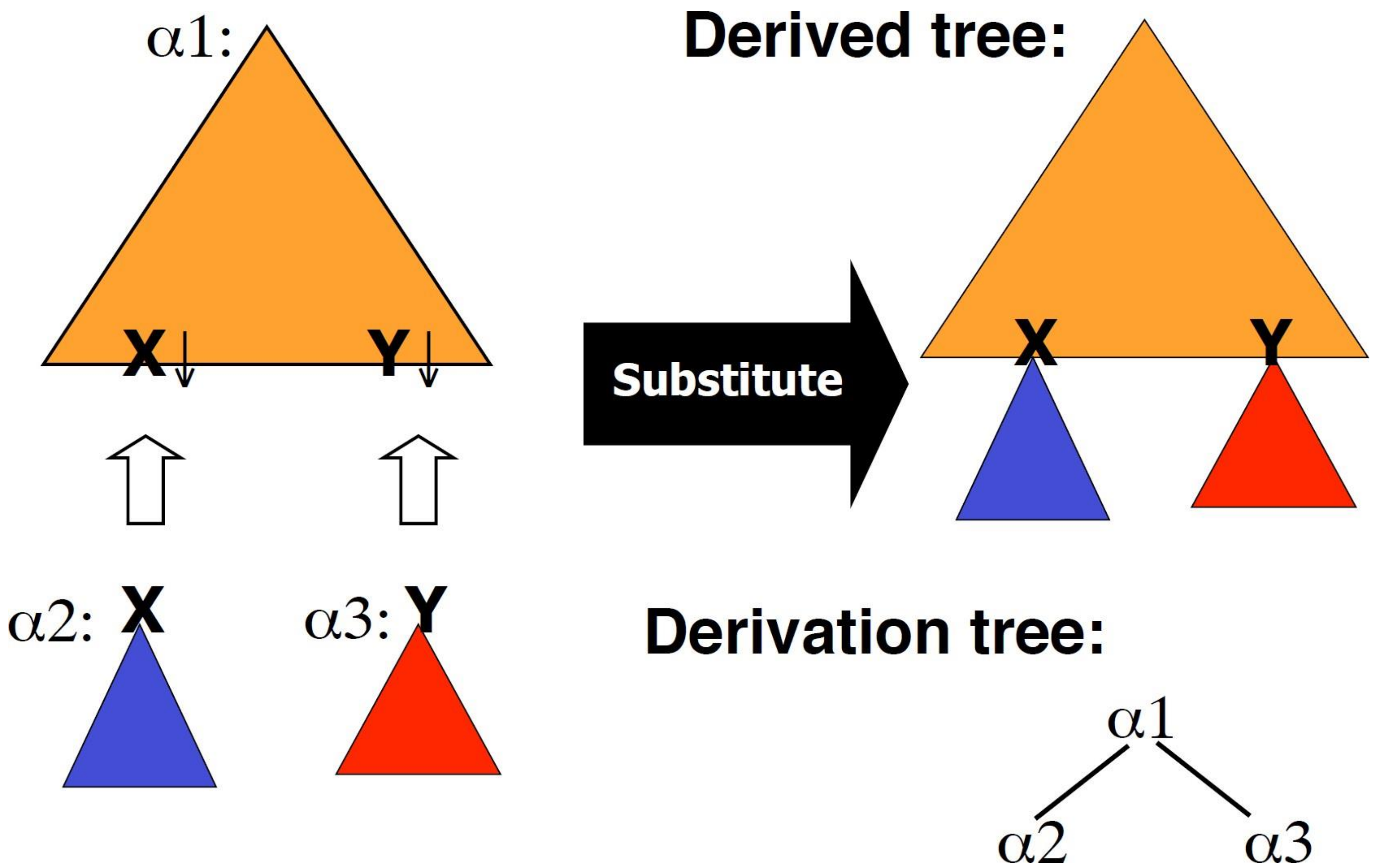
≈

CYK : Inside-Outside

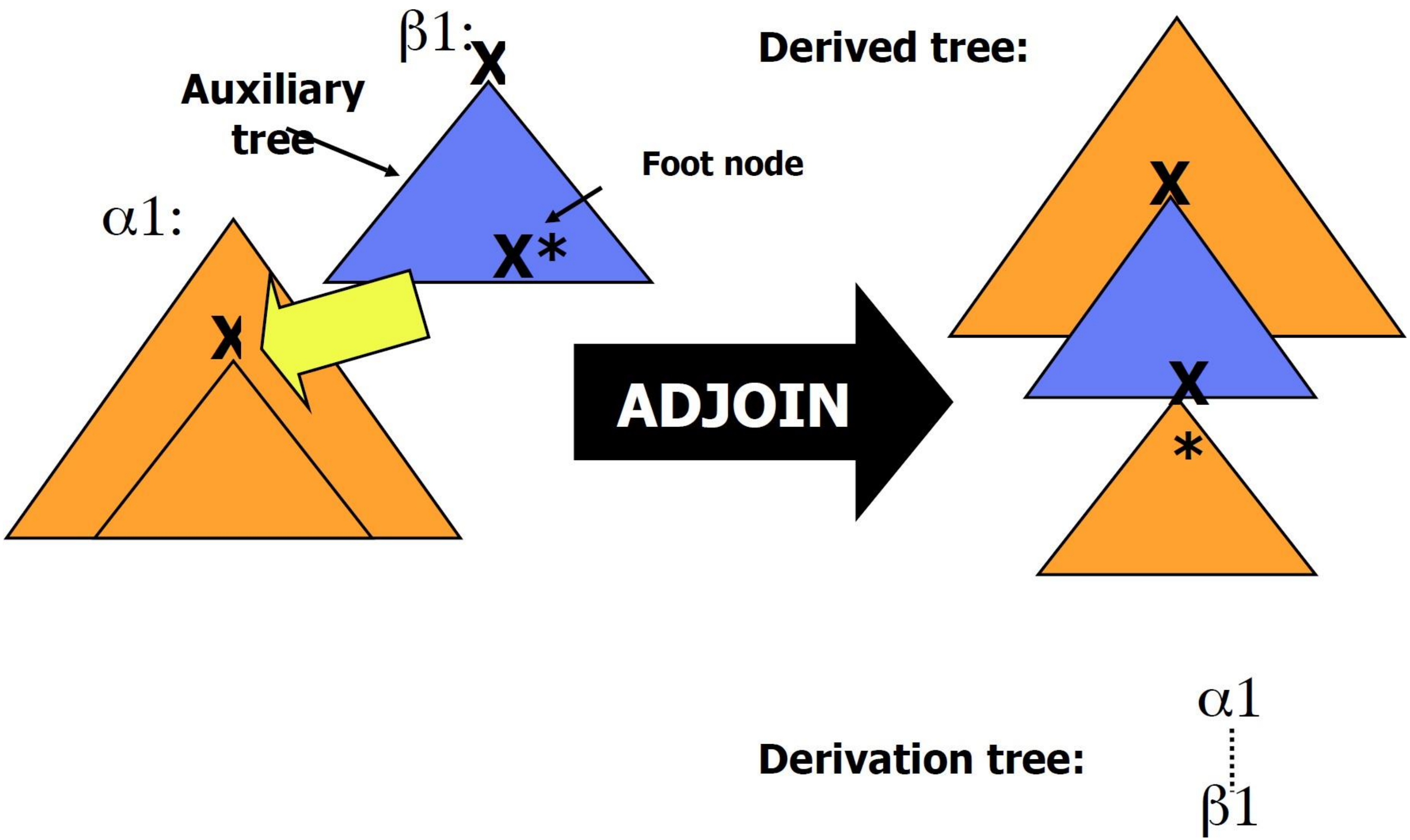
Mildly Context-Sensitive Grammars

- More expressive than CFG
  - Production is no longer independent of context
- Polynomial-time parsing
- Several formalisms
  - Tree-adjoining grammar (TAG)
  - Combinatory categorial grammar (CCG)
  - Linear context-free rewriting systems (LCFRS)
  - Multiple context-free grammars (MCFG)
  - …

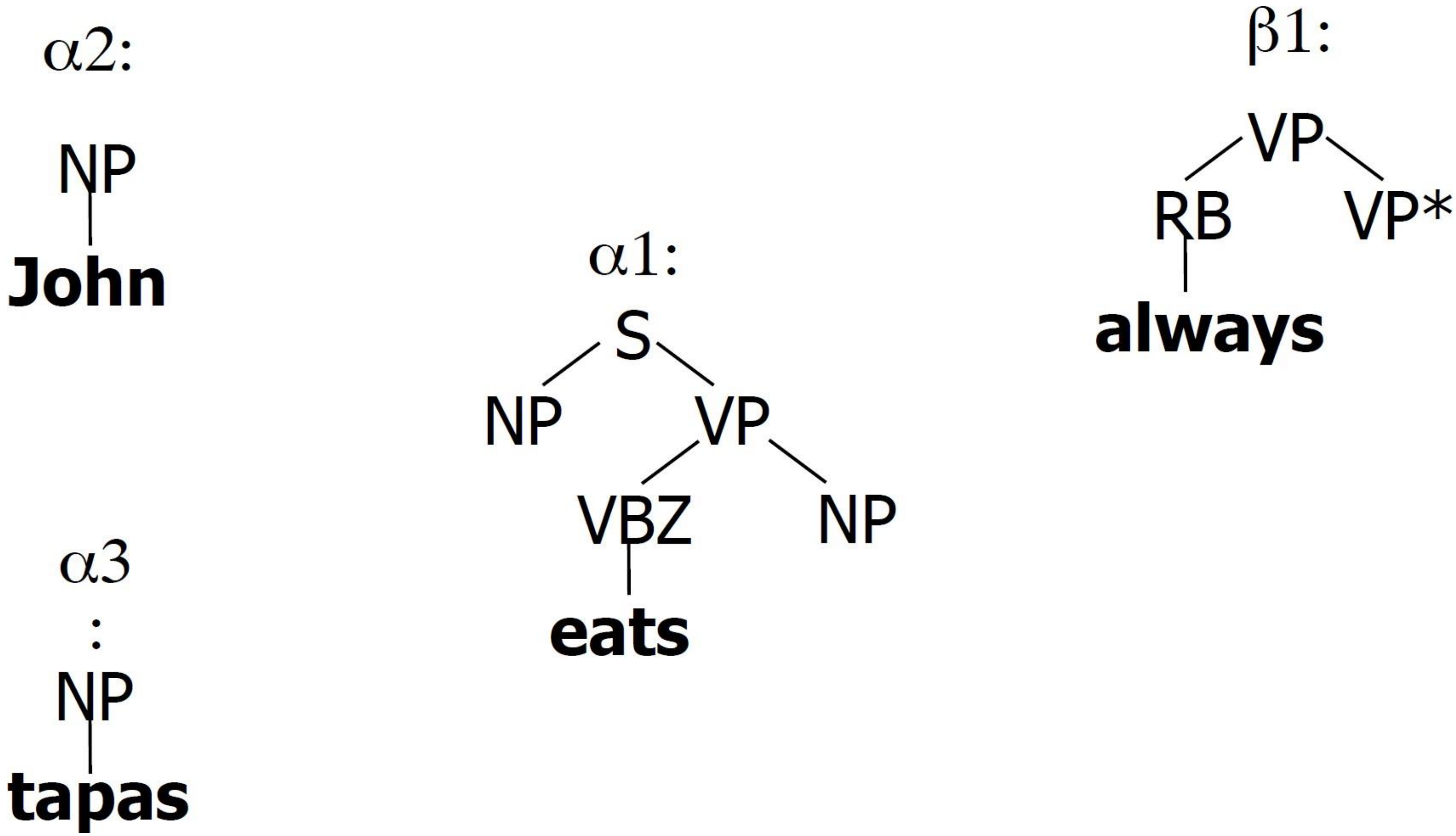
TAG Rule 1: Substitution



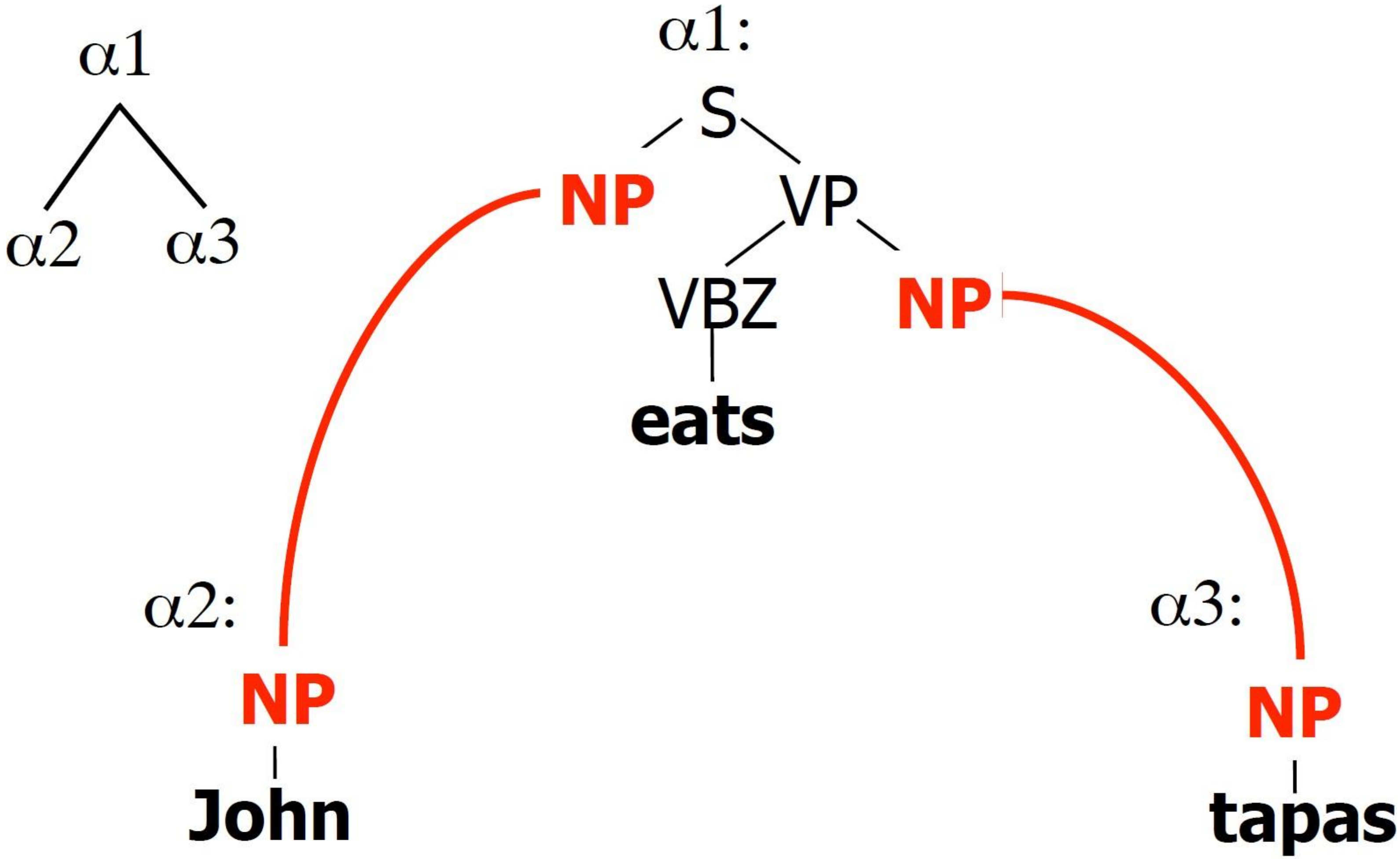
TAG Rule 2: Adjoin



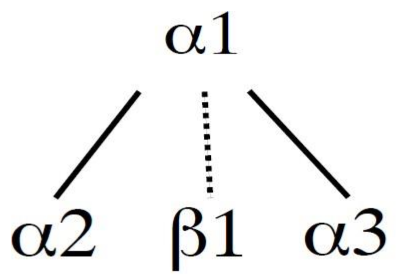
Example: TAG Lexicon

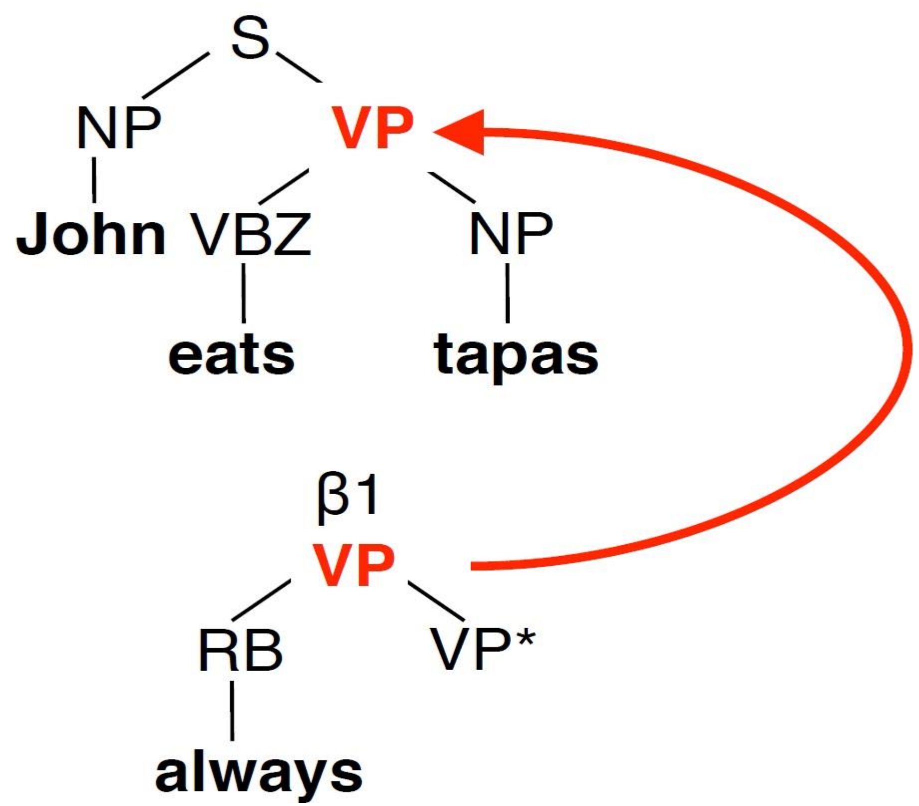


Example: TAG derivation

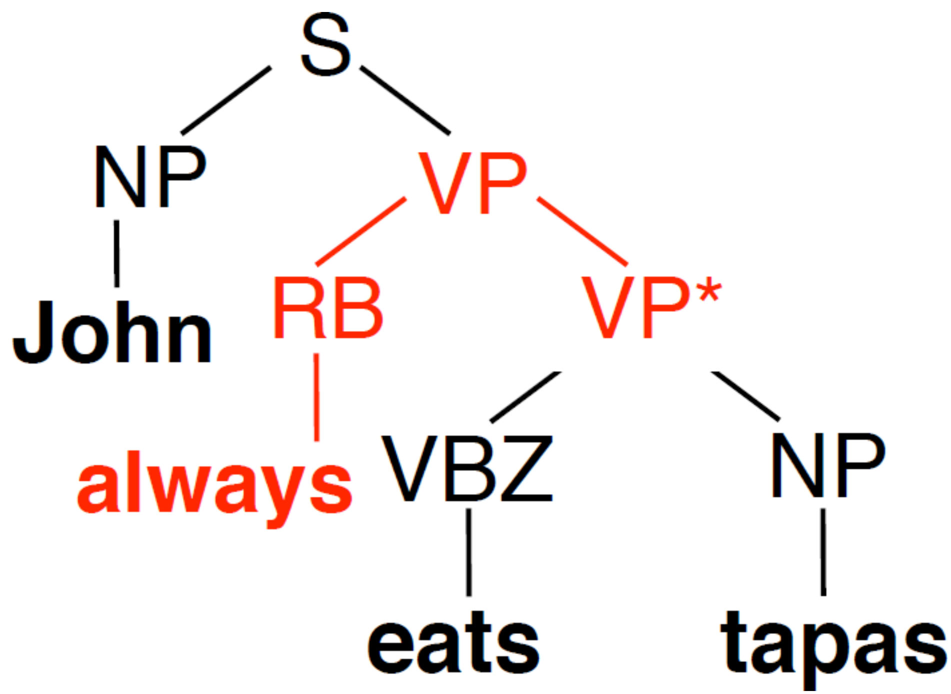


Example: TAG derivation



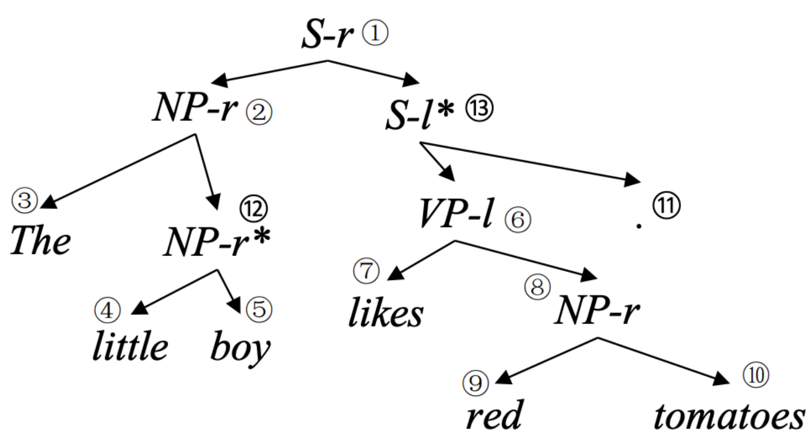


Example: TAG derivation



Transition-based parsing

- Read words from left to right and take a sequence of actions to construct a tree
- Actions are picked greedily or using beam search
- More on this later…



stack	buffer	action	node
[]	[The little ...]	SHIFT	③
[The]	[little boy ...]	SHIFT	④
[The little]	[boy likes ...]	SHIFT	⑤
[... little boy]	[likes red ...]	REDUCE-R-NP	②
...	...	...	...

(a) bottom-up system

stack	buffer	action	node
[]	[The little ...]	NT-S	①
[(S]	[The little ...]	NT-NP	②
[(S (NP]	[The little ...]	SHIFT	③
[... (NP The]	[little boy ...]	SHIFT	④
[... The little]	[boy likes ...]	SHIFT	⑤
[... little boy]	[likes red ...]	REDUCE	/
...	...	...	...

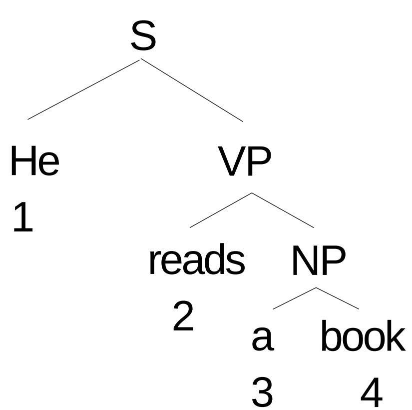
(b) top-down system

stack	buffer	action	node
[]	[The little ...]	SHIFT	③
[The]	[little boy ...]	PJ-NP	②
[The NP]	[little boy ...]	SHIFT	④
[... NP little]	[boy likes...]	SHIFT	⑤
[... little boy]	[likes red ...]	REDUCE	/
...	...	...	...

(c) in-order system

Span-based parsing

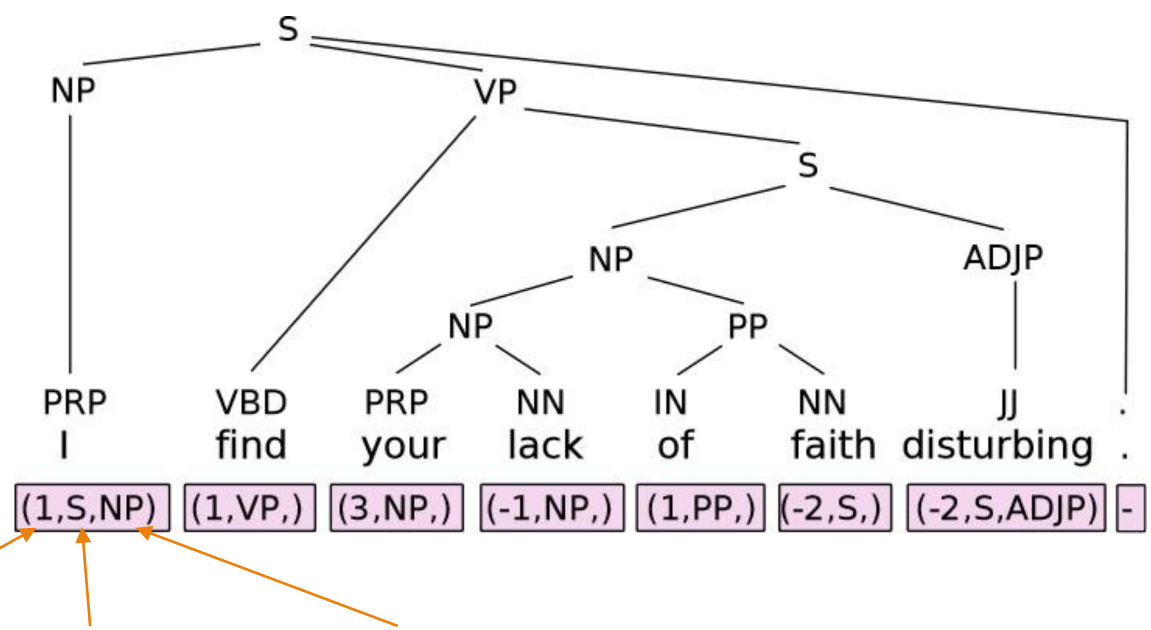
- Discriminative parsing
  - Maximize conditional likelihood P(gold parse | sentence)
- Tree score = product of span scores
  - Span score computed from features or using a neural network
- We can still use CYK!



$$s(t) = s(1,1) \times s(2,2) \times s(3,3) \times s(4,4) \times s(3,4) \times s(2,4) \times s(1,4)$$

Parsing as sequence labeling

- Cast constituency parsing as a sequence labeling task
  - Advantage: faster parsing speed



# of common ancestor shared with the next word

lowest common ancestor (LCA) to the next word

unary chain connecting the LCA and the current word

Summary

Constituency Parsing

- Context-Free Grammars
  - Terminals, nonterminals, start symbol, production rules
  - Probabilistic Grammars: each rule has a probability
- Parsing
  - CYK algorithm on CNF
- Learning
  - Supervised: generative & discriminative methods
  - Unsupervised: inside-outside algorithm
- Beyond CFG
  - RG, MCSG
  - Transition/span-based parsing, parsing as sequence labeling