Algorithms for NLP

Parsing II

Yulia Tsvetkov – CMU

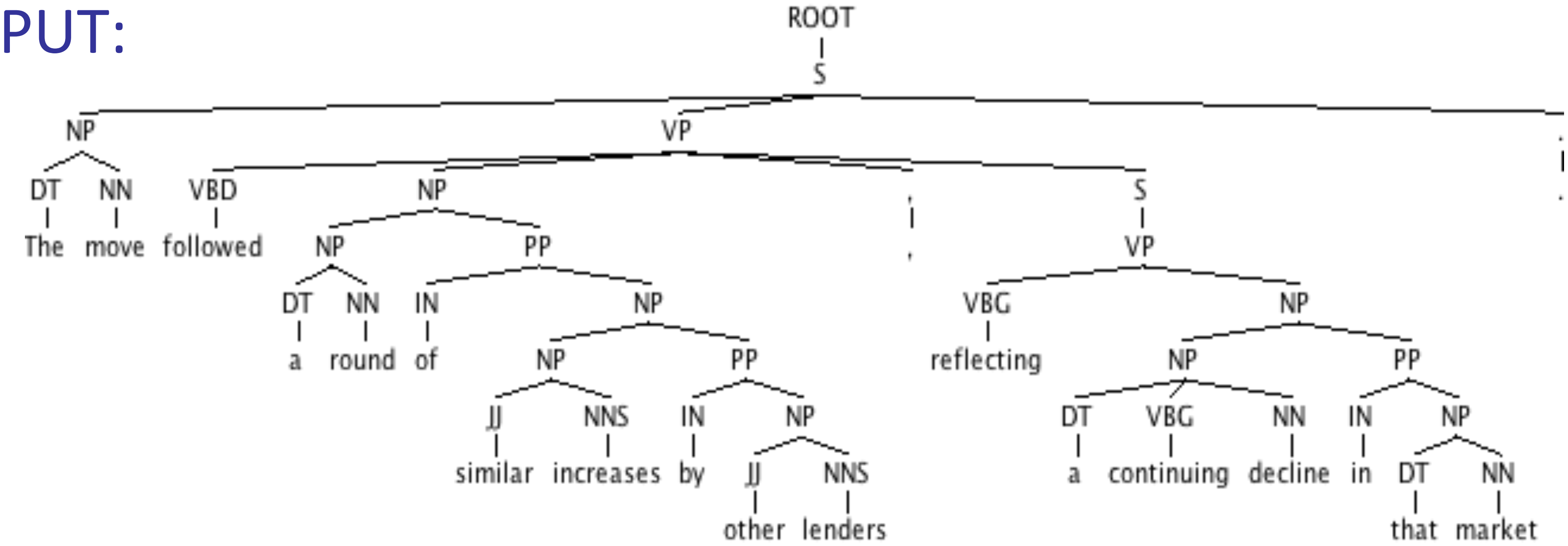
Slides: Ivan Titov – University of Edinburgh,
Chris Dyer – Deepmind

Announcements

- HW2 out
- Today: Sachin will give an overview of HW2
- Recitation on EM next week 10/12
- Recitation on HW2 the week after 10/19
- Yulia office hours
  - today: 3:30-4:00
  - next week Yulia is away, no office hours

Syntactic Parsing

- INPUT:
  - The move followed a round of similar increases by other lenders, reflecting a continuing decline in that market
- OUTPUT:

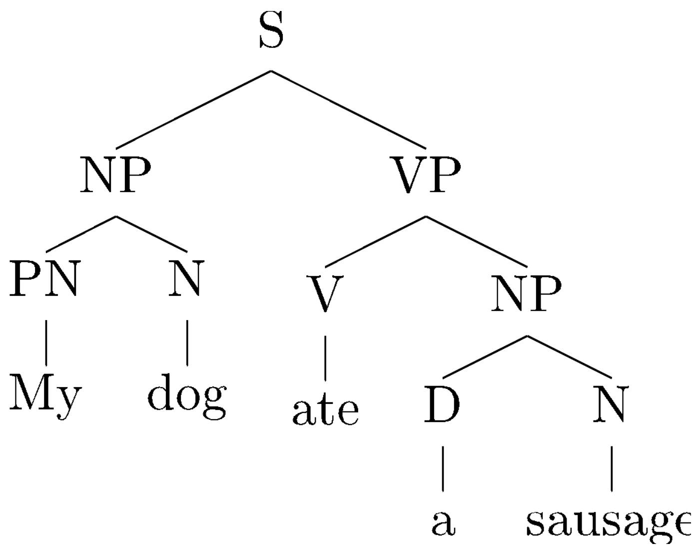


Context Free Grammar (CFG)

Grammar (CFG)		Lexicon
ROOT → S	NP → NP PP	NN → interest
S → NP VP	VP → VBP NP	NNS → raises
NP → DT NN	VP → VBP NP PP	VBP → interest
NP → NN NNS	PP → IN NP	VBZ → raises
		…

- Other grammar formalisms: LFG, HPSG, TAG, CCG…

Constituent trees



- Internal nodes correspond to phrases
  - S – a sentence
  - NP (Noun Phrase): My dog, a sandwich, lakes,..
  - VP (Verb Phrase): ate a sausage, barked, …
  - PP (Prepositional phrases): with a friend, in a car, …

- Nodes immediately above words are PoS tags (aka preterminals)
  - PN – pronoun
  - D – determiner
  - V – verb
  - N – noun
  - P – preposition

Parsing with CKY

lead	can	poison

0 1 2 3

$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

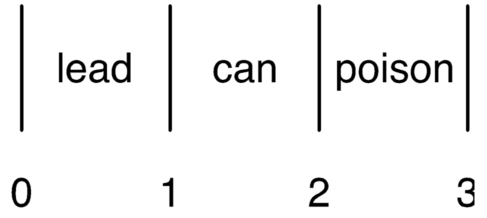
Inner rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules



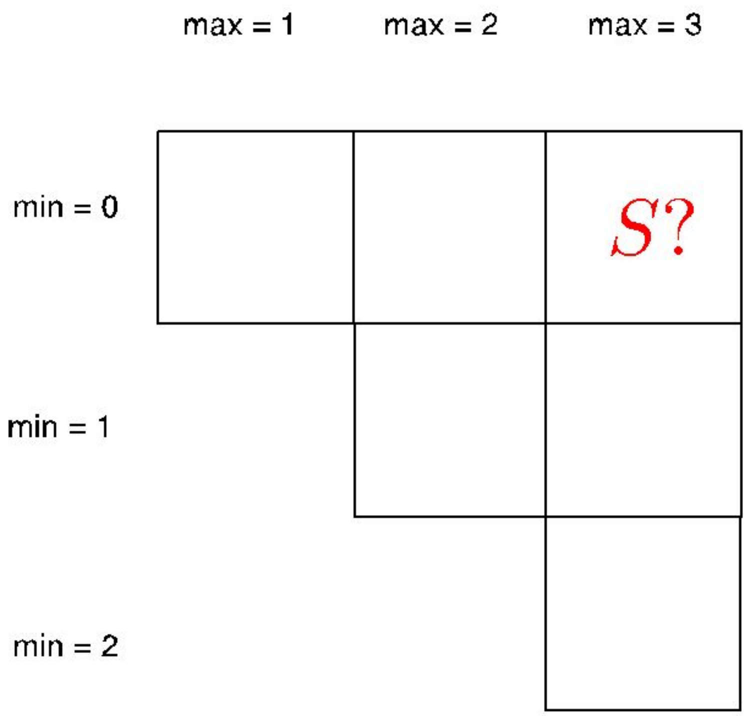


Chart (aka parsing triangle)

$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ \ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

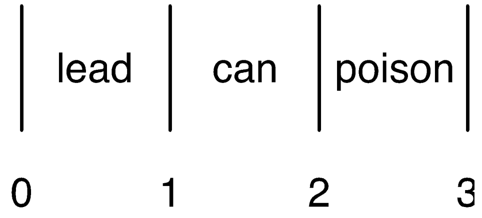
Inner rules

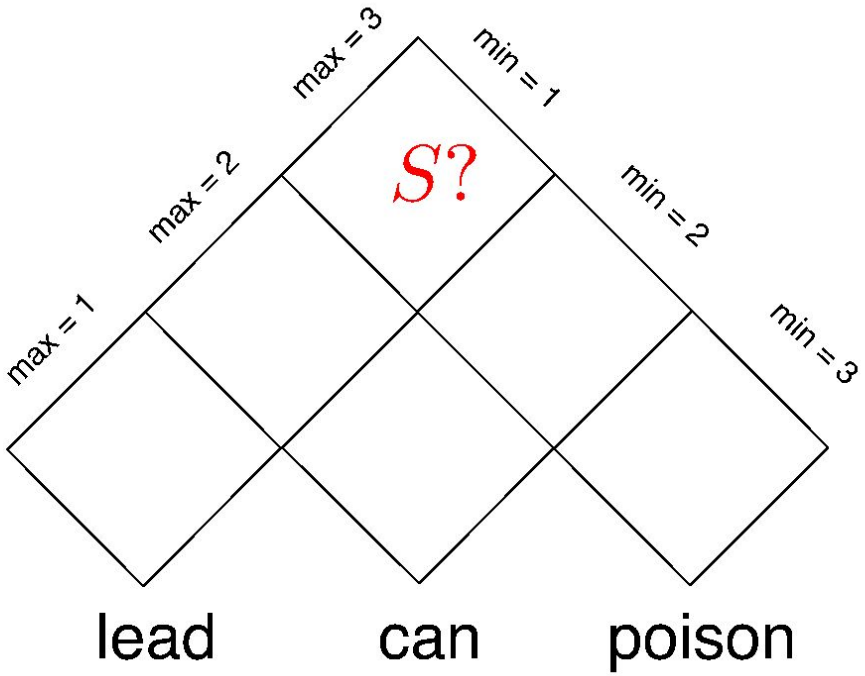
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \rightarrow NP \ \ VP$$

$$VP \rightarrow M \ \ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N \ NP$$

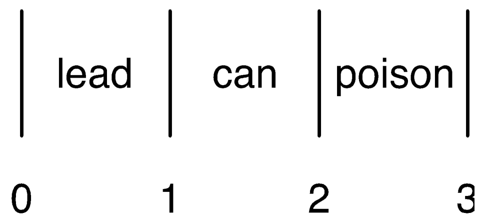
Inner rules

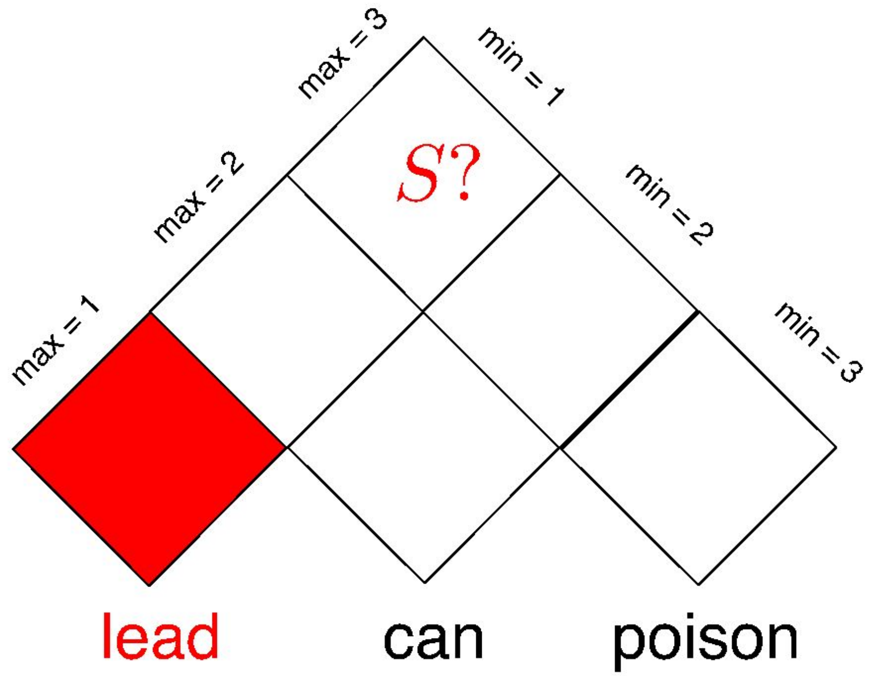
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \rightarrow NP \ VP$$

Inner rules

$$VP \rightarrow M \ V$$
$$VP \rightarrow V$$

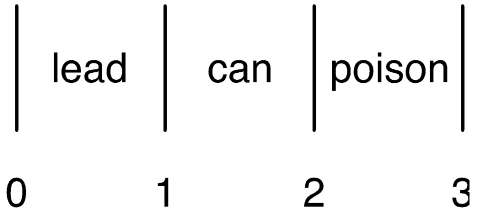
$$NP \rightarrow N$$
$$NP \rightarrow N \ NP$$

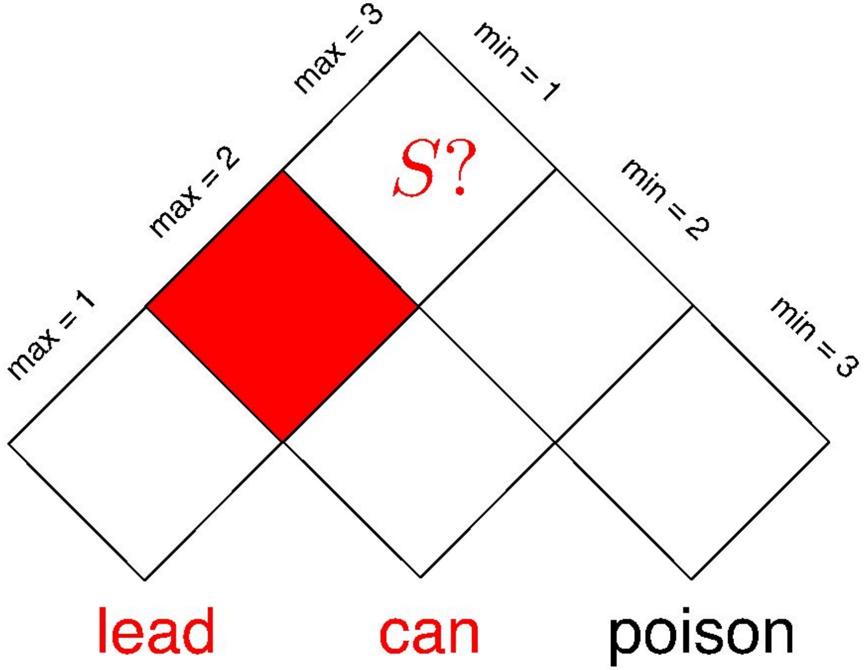
Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$





$$S \to NP \ \ VP$$

$$VP \to M \ \ V$$
$$VP \to V$$

$$NP \to N$$
$$NP \to N \ NP$$

Inner rules

$$N \to can$$
$$N \to lead$$
$$N \to poison$$

$$M \to can$$
$$M \to must$$

$$V \to poison$$
$$V \to lead$$

Preterminal rules

lead	can	poison

0 1 2 3

	max = 1	max = 2	max = 3
min = 0			$S?$
min = 1			
min = 2			

Inner rules

$$S \rightarrow NP \ \ VP$$

$$VP \rightarrow M \ \ V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

$$NP \rightarrow N \ NP$$

Preterminal rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

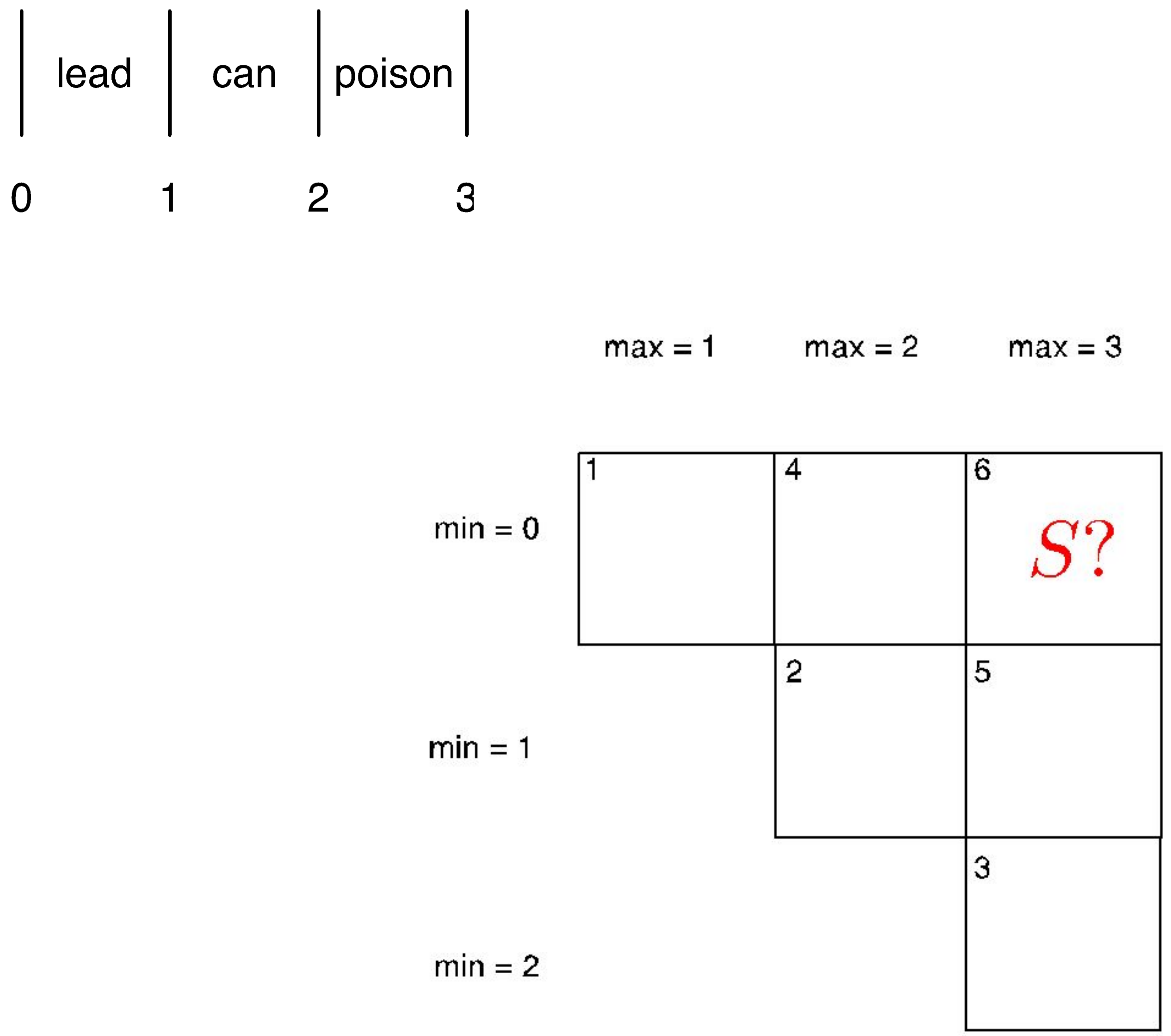
$$N \rightarrow poison$$

$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$



$$S \rightarrow NP \ VP$$

Inner rules

$$VP \rightarrow M \ V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

$$NP \rightarrow N \ NP$$

Preterminal rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

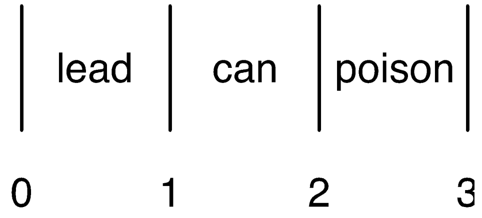
$$N \rightarrow poison$$

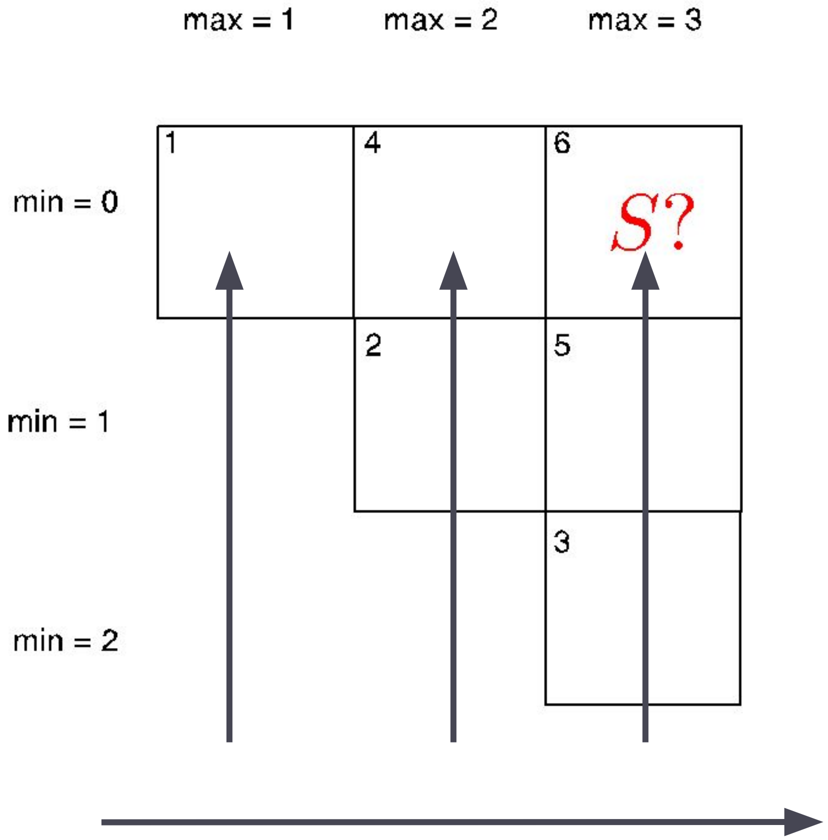
$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$





$$S \rightarrow NP \ VP$$

$$VP \rightarrow M \ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N \ NP$$

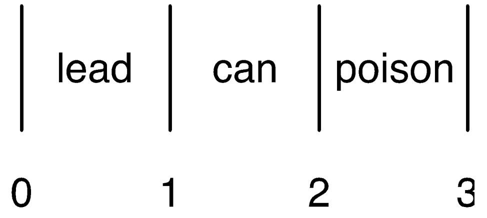
Inner rules

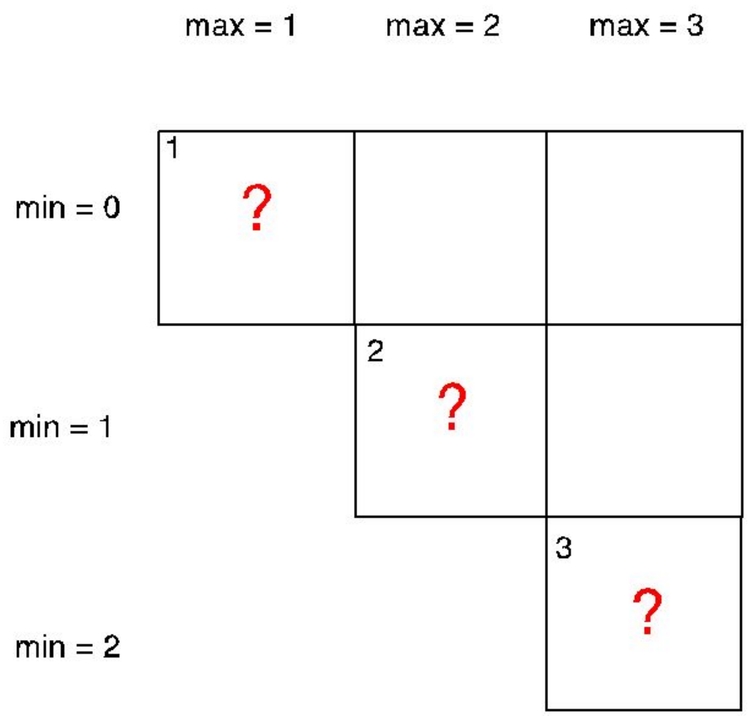
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

Preterminal rules





$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

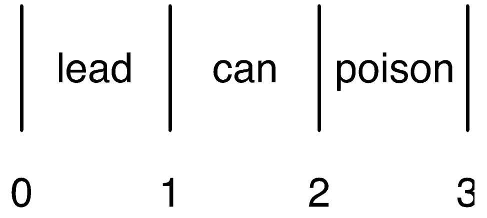
$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

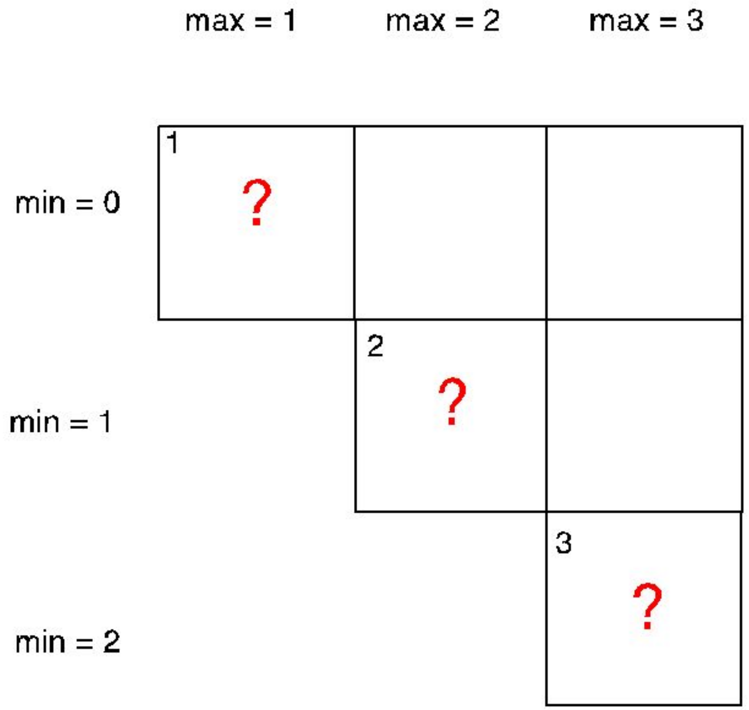
Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$





$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

lead	can	poison

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V		
min = 1		2 N, M	
min = 2			3 N, V

$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

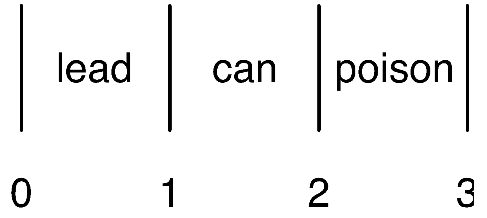
$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

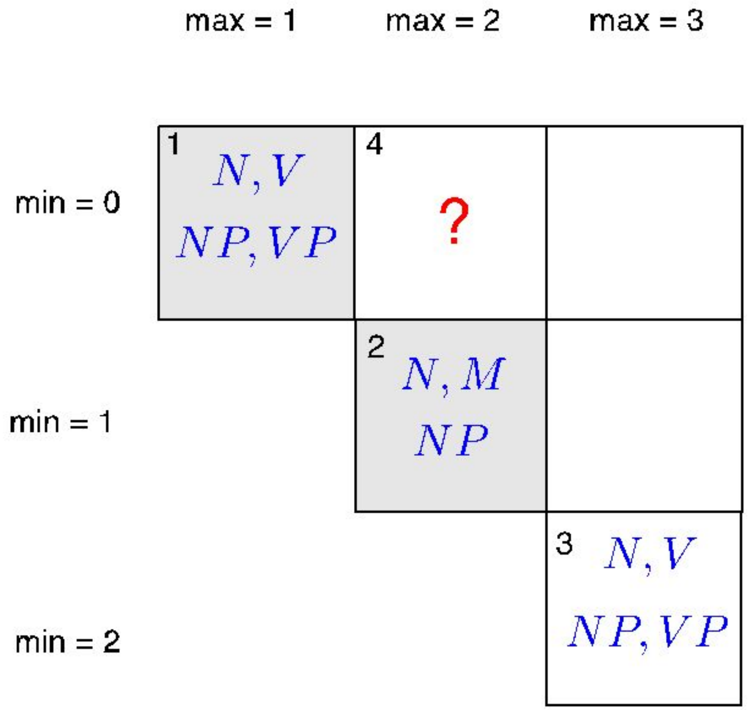
Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$





$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

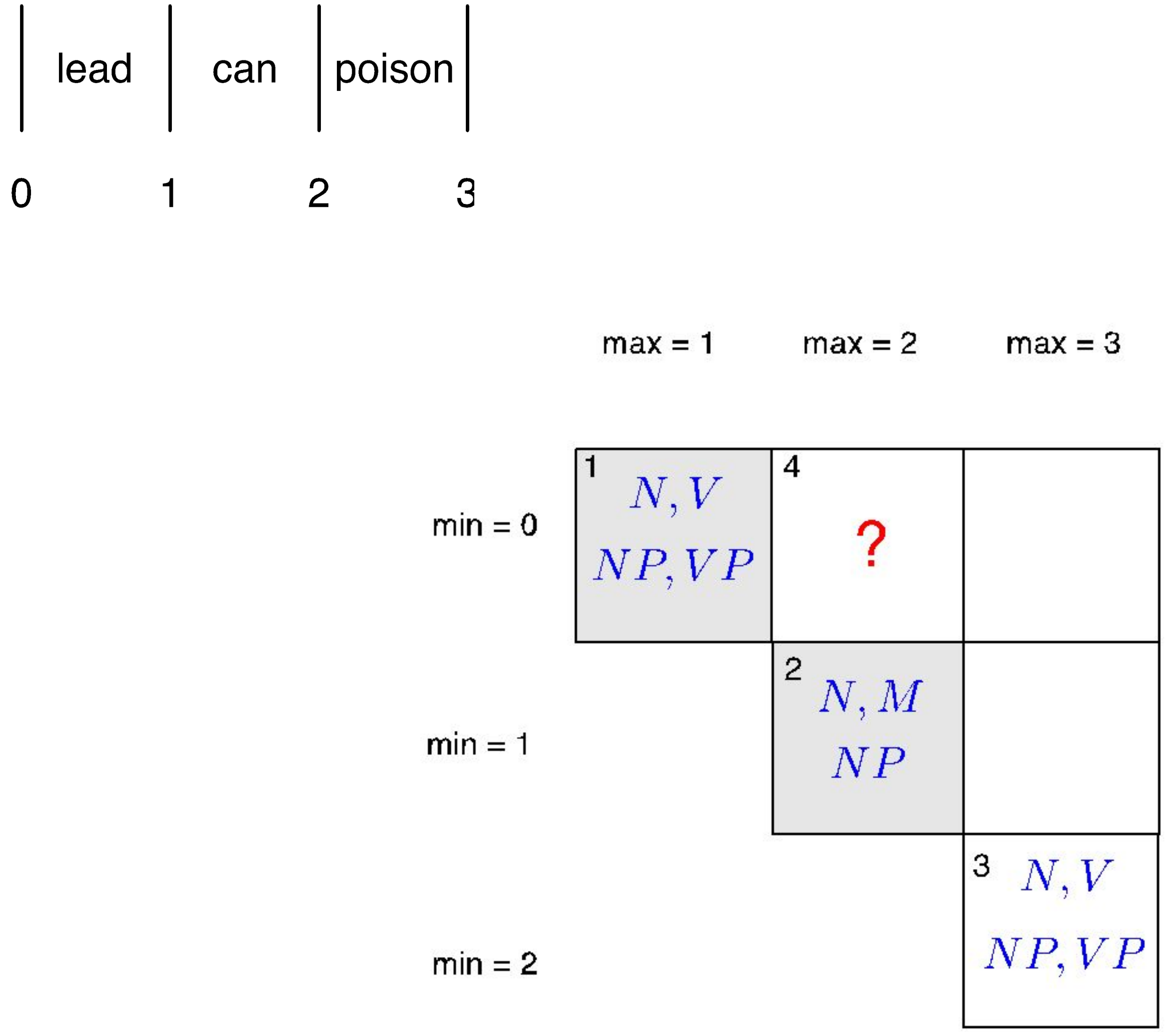
Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

$S \rightarrow NP\ VP$



Inner rules

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$\boxed{NP \rightarrow N\ NP}$

Preterminal rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

lead	can	poison

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	
min = 1		2 N, M NP	
min = 2			3 N, V NP, VP

$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$\boxed{NP \rightarrow N\ NP}$$

Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

lead	can	poison

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	
min = 1		2 N, M NP	5 ?
min = 2			3 N, V NP, VP

$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

| lead | can | poison |

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

$$\boxed{S \rightarrow NP\ VP}$$

Inner rules

$$\boxed{VP \rightarrow M\ V}$$
$$VP \rightarrow V$$

$$NP \rightarrow N$$
$$\boxed{NP \rightarrow N\ NP}$$

Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

$$M \rightarrow can$$
$$M \rightarrow must$$

$$V \rightarrow poison$$
$$V \rightarrow lead$$

lead	can	poison

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	6 ?
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

$$S \to NP \;\; VP$$

Inner rules

$$VP \to M \;\; V$$
$$VP \to V$$
$$NP \to N$$
$$NP \to N \; NP$$

Preterminal rules

$$N \to can$$
$$N \to lead$$
$$N \to poison$$
$$M \to can$$
$$M \to must$$
$$V \to poison$$
$$V \to lead$$

lead can poison

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	6 ?
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

$$S \rightarrow NP\ VP$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$
$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$
$$M \rightarrow can$$
$$M \rightarrow must$$
$$V \rightarrow poison$$
$$V \rightarrow lead$$

lead	can	poison

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	6 S, NP
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

mid=1

$$\boxed{S \rightarrow NP\ VP}$$

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$
$$NP \rightarrow N$$
$$\boxed{NP \rightarrow N\ NP}$$

Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$
$$M \rightarrow can$$
$$M \rightarrow must$$
$$V \rightarrow poison$$
$$V \rightarrow lead$$

$S \rightarrow NP\ VP$

lead	can	poison

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	6 S, NP S(?!)
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

mid=2

Inner rules

$$VP \rightarrow M\ V$$
$$VP \rightarrow V$$
$$NP \rightarrow N$$
$$NP \rightarrow N\ NP$$

Preterminal rules

$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$
$$M \rightarrow can$$
$$M \rightarrow must$$
$$V \rightarrow poison$$
$$V \rightarrow lead$$

| lead | can | poison |
0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1 N, V NP, VP	4 NP	6 S, NP $S(?!)$
min = 1		2 N, M NP	5 $S, VP,$ NP
min = 2			3 N, V NP, VP

$\boxed{S \rightarrow NP\ VP}$

Inner rules

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N\ NP$

Preterminal rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

Apparently the sentence is ambiguous for the grammar: (as the grammar overgenerates)

PCFGs

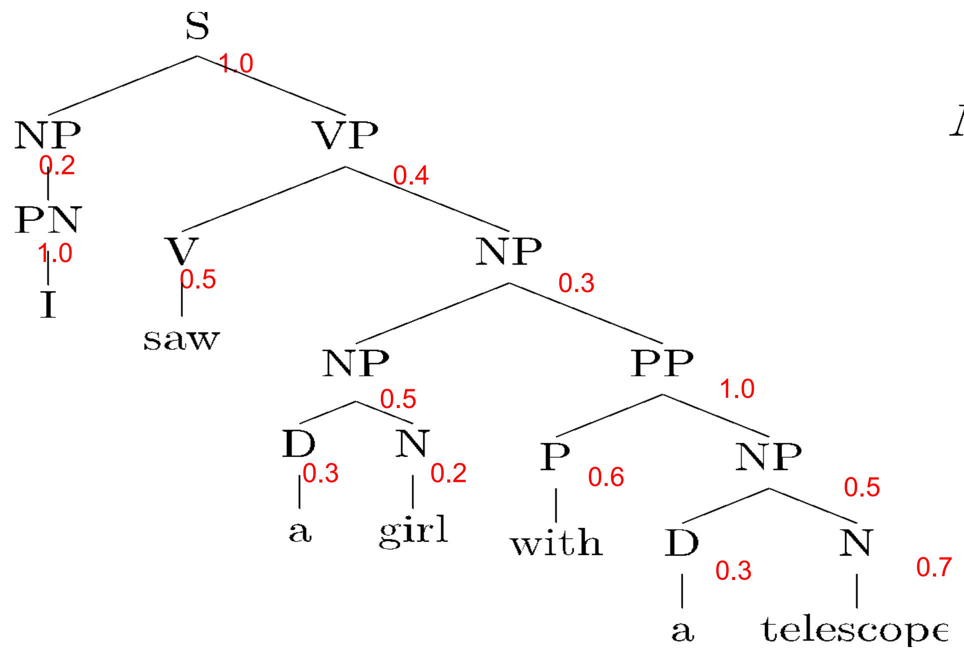
$$S \rightarrow NP \;\; VP \;\; 1.0$$

$$VP \rightarrow V \;\; 0.2$$
$$VP \rightarrow V \;\; NP \;\; 0.4$$
$$VP \rightarrow VP \;\; PP \;\; 0.4$$

$$NP \rightarrow NP \;\; PP \;\; 0.3$$
$$NP \rightarrow D \;\; N \;\; 0.5$$
$$NP \rightarrow PN \;\; 0.2$$

$$PP \rightarrow P \;\; NP \;\; 1.0$$

$$N \rightarrow girl \;\; 0.2$$
$$N \rightarrow telescope \;\; 0.7$$
$$N \rightarrow sandwich \;\; 0.1$$
$$PN \rightarrow I \;\; 1.0$$
$$V \rightarrow saw \;\; 0.5$$
$$V \rightarrow ate \;\; 0.5$$
$$P \rightarrow with \;\; 0.6$$
$$P \rightarrow in \;\; 0.4$$
$$D \rightarrow a \;\; 0.3$$
$$D \rightarrow the \;\; 0.7$$



$$p(T) = 1.0 \times 0.2 \times 1.0 \times 0.4 \times 0.5 \times 0.3 \times$$
$$0.5 \times 0.3 \times 0.2 \times 1.0 \times 0.6 \times 0.5 \times 0.3 \times 0.7$$
$$= 2.26 \times 10^{-5}$$

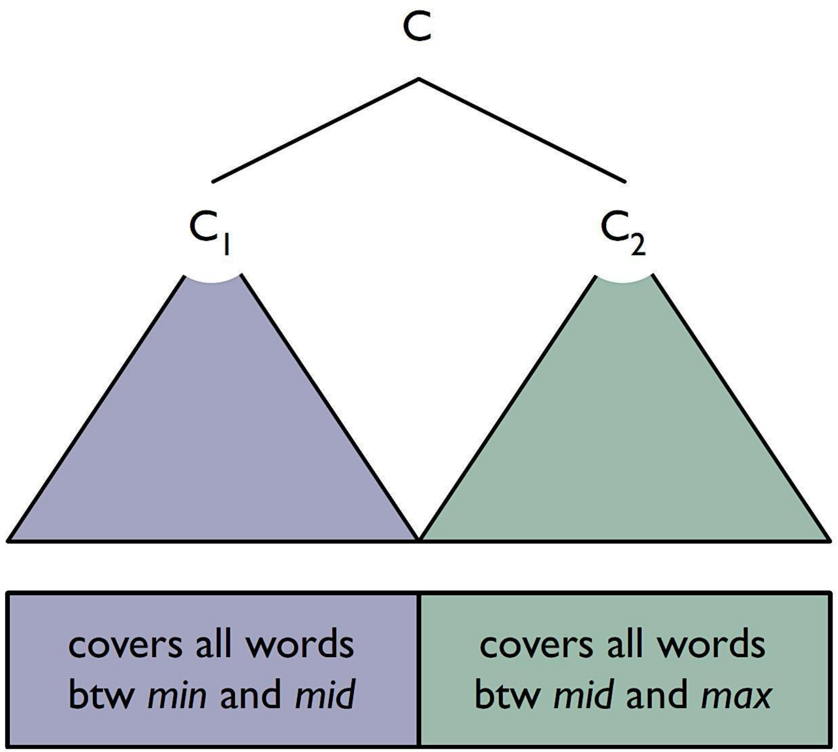
CKY with PCFGs

- Chart is represented by a 3d array of floats
  `chart[min][max][label]`
  - It stores probabilities for the most probable subtree with a given signature

- `chart[0][n][S]` will store the probability of the most probable full parse tree

Intuition

$$C \to C_1 \;\; C_2$$



For every C choose C_1, C_2 and mid such that

$$P(T_1) \times P(T_2) \times P(C \to C_1 C_2)$$

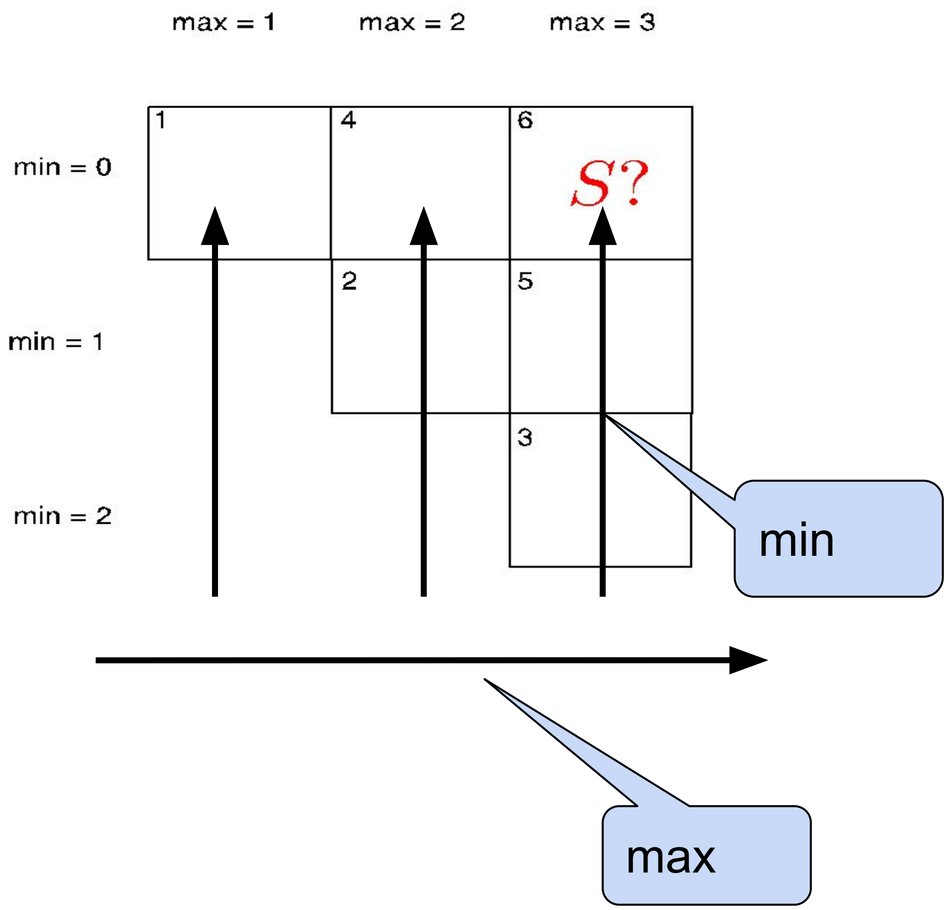
is maximal, where T_1 and T_2 are left and right subtrees.

Implementation: preterminal rules

```
for each wᵢ from left to right
  for each preterminal rule C -> wᵢ
    chart[i - 1][i][C] = p(C -> wᵢ)
```

Implementation: binary rules

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double best = undefined
        for each binary rule C -> C1 C2
          for each mid from min + 1 to max - 1
            double t1 = chart[min][mid][C1]
            double t2 = chart[mid][max][C2]
            double candidate = t1 * t2 * p(C -> C1 C2)
            if candidate > best then
              best = candidate
      chart[min][max][C] = best
```



Recovery of the tree

- For each signature we store backpointers to the elements from which it was built
  - start recovering from [0, n, S]

- What backpointers do we store?

Recovery of the tree

- For each signature we store backpointers to the elements from which it was built
  - start recovering from [0, n, S]

- What backpointers do we store?
  - rule
  - for binary rules, midpoint

Constraints on the grammar

- The basic CKY algorithm supports only rules in the Chomsky Normal Form (CNF):

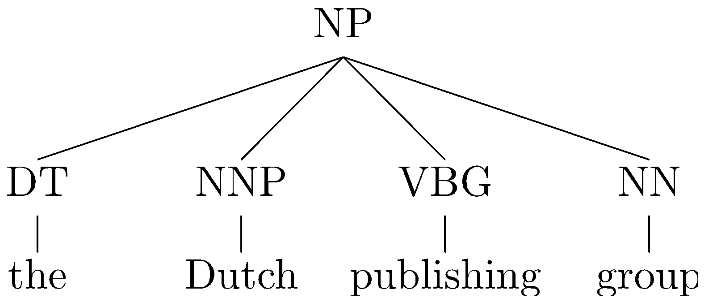
$$C \to x$$

$$C \to C_1 C_2$$

- Any CFG can be converted to an equivalent CNF
  - Equivalent means that they define the same language
  - However (syntactic) trees will look differently
  - It is possible to address it by defining such transformations that allows for easy reverse transformation

Transformation to CNF form: binarization

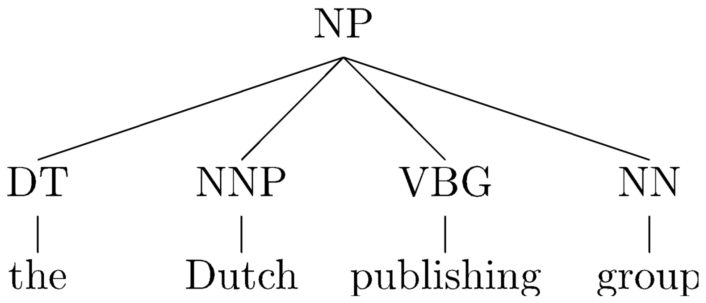
- Consider $NP \rightarrow DT \;\; NNP \;\; VBG \;\; NN$



- How do we get a set of binary rules which are equivalent?

Transformation to CNF form: binarization

- Consider $NP \rightarrow DT \;\; NNP \;\; VBG \;\; NN$

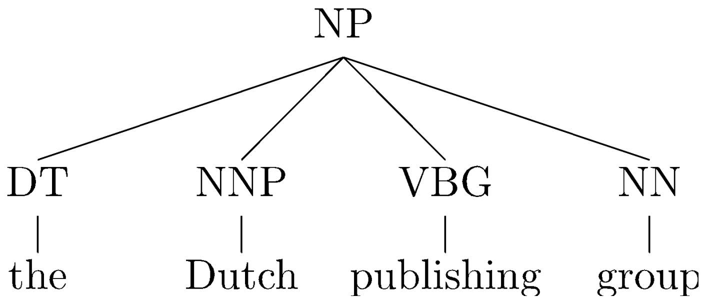


- How do we get a set of binary rules which are equivalent?

$$NP \rightarrow DT \;\; X$$
$$X \rightarrow NNP \;\; Y$$
$$Y \rightarrow VBG \;\; NN$$

Transformation to CNF form: binarization

- Consider $NP \rightarrow DT\ \ NNP\ \ VBG\ \ NN$



- How do we get a set of binary rules which are equivalent?

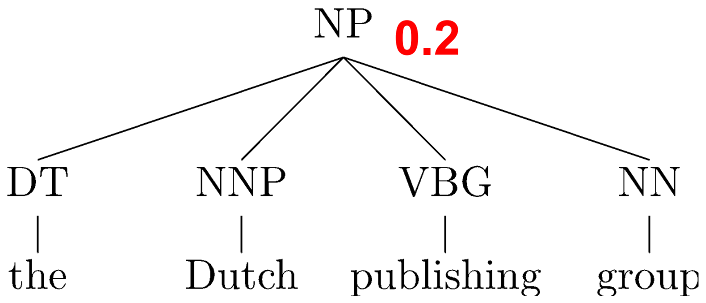
$$NP \rightarrow DT\ \ X$$
$$X \rightarrow NNP\ \ Y$$
$$Y \rightarrow VBG\ \ NN$$

- A more systematic way to refer to new non-terminals

$$NP \rightarrow DT\ \ @NP|DT$$
$$@NP|DT \rightarrow NNP\ \ @NP|DT\_NNP$$
$$@NP|DT\_NNP \rightarrow VBG\ \ NN$$

Transformation to CNF form: binarization

- Consider $NP \rightarrow DT \;\; NNP \;\; VBG \;\; NN$ **0.2**

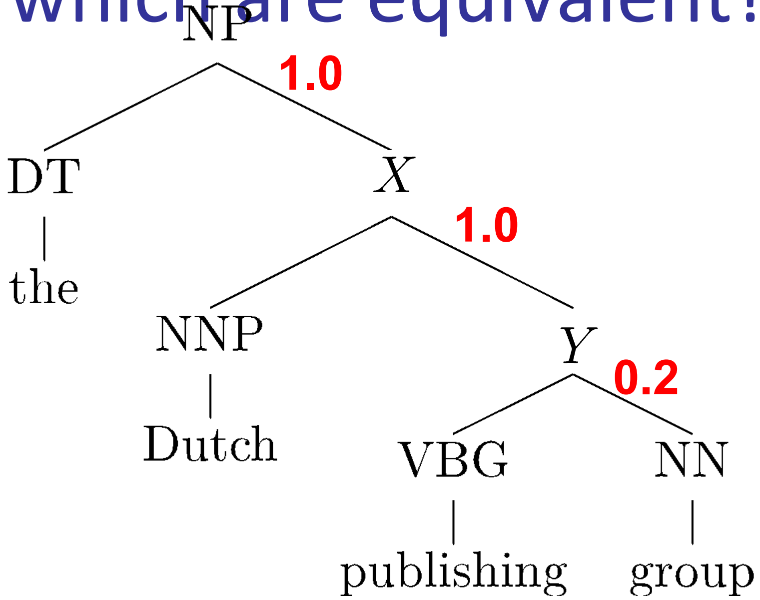


- How do we get a set of binary rules which are equivalent?

$NP \rightarrow DT \;\; X$ **1.0**

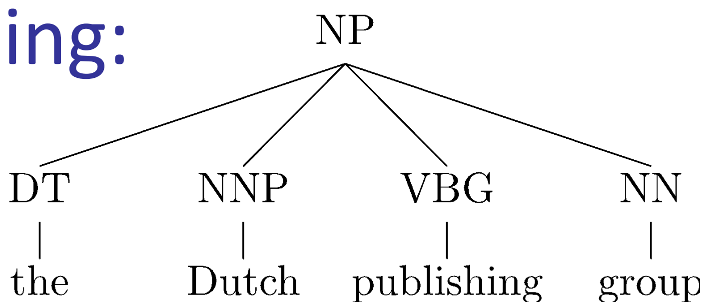
$X \rightarrow NNP \;\; Y$ **1.0**

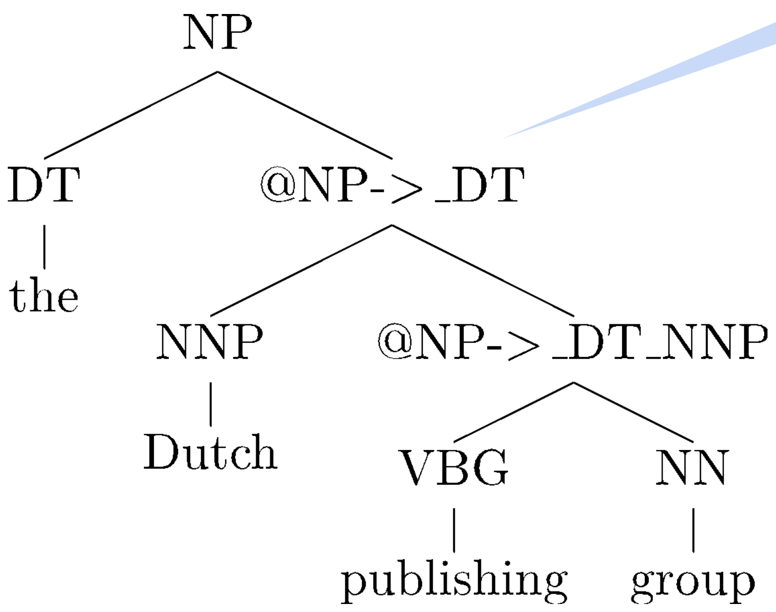
$Y \rightarrow VBG \;\; NN$ **0.2**



Transformation to CNF form: binarization

- Instead of binarizing tuples we can binarize trees on preprocessing:





Also known as **lossless Markovization** in the context of PCFGs

Can be easily reversed on postprocessing

Unary Rules

- CNF includes only two types of rules:

$$C \rightarrow x$$

$$C \rightarrow C_1 C_2$$

- What about unary rules:

$$C \rightarrow C_1$$

Unary Rules

CFG

A → X
B → X
C → X
…
X → C_1C_2

…

X → run
X → play
X → sleep
X → love

CNF

A → run,	B → run,	C → run,	X → run,
A → play,	B → play,	C → play,	X → play,
A → sleep,	B → sleep,	C → sleep,	X → sleep,
A → love	B → love	C → love	X → love
…	…	…	…
A → C_1C_2	B → C_1C_2	C → C_1C_2	X → C_1C_2

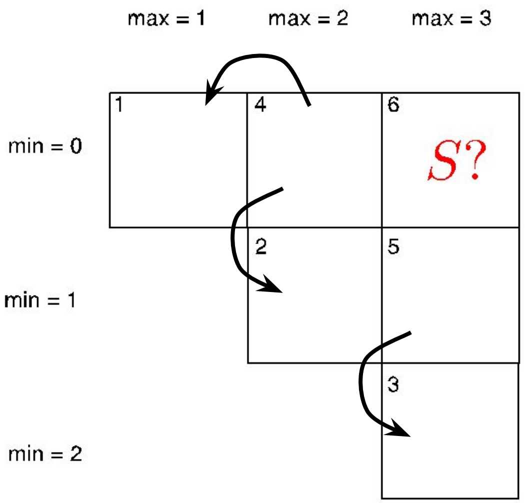
- explode the grammar
- make it hard to reverse

Unary rules

- How to integrate unary rules $C \longrightarrow C_1$?

```
for each max from 1 to n
  for each min from max - 1 down to 0
    // First, try all binary rules as before.
    ...
    // Then, try all unary rules.
    for each syntactic category C
      for each unary rule C → C1
        chart[min][max][C] = maximum (chart[min][max][C],
                                      chart[min][max][C1])
```

new bounds!



Unary closure

- What if the grammar contained 2 rules:

$$A \to B$$
$$B \to C$$

- But C can be derived from A by a chain of rules:

$$A \to B \to C$$

- One could support chains in the algorithm but it is easier to extend the grammar, to get the transitive closure

$$\begin{array}{c} A \to B \\ B \to C \end{array} \quad \Rightarrow \quad \begin{array}{c} A \to B \\ B \to C \\ A \to C \end{array}$$

Why unary closure

$A \rightarrow B$

$B \rightarrow C$

$A \rightarrow C$

```
// Then, try all unary rules.
for each syntactic category C
  for each unary rule C -> C1
    if chart[min][max][C1] then
      chart[min][max][C] = true
```

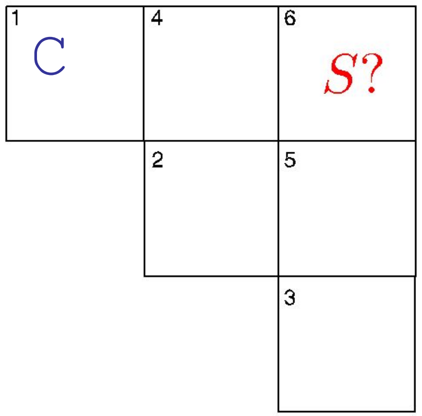
Why unary closure

$A \rightarrow B$

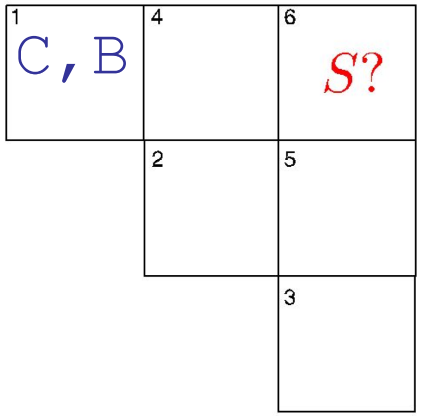
$B \rightarrow C$

$A \rightarrow C$

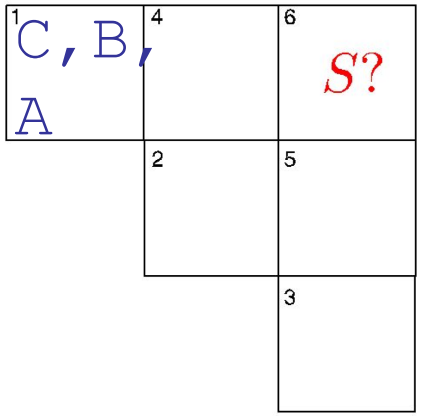
scenario 1



$B \rightarrow C$



$A \rightarrow B$



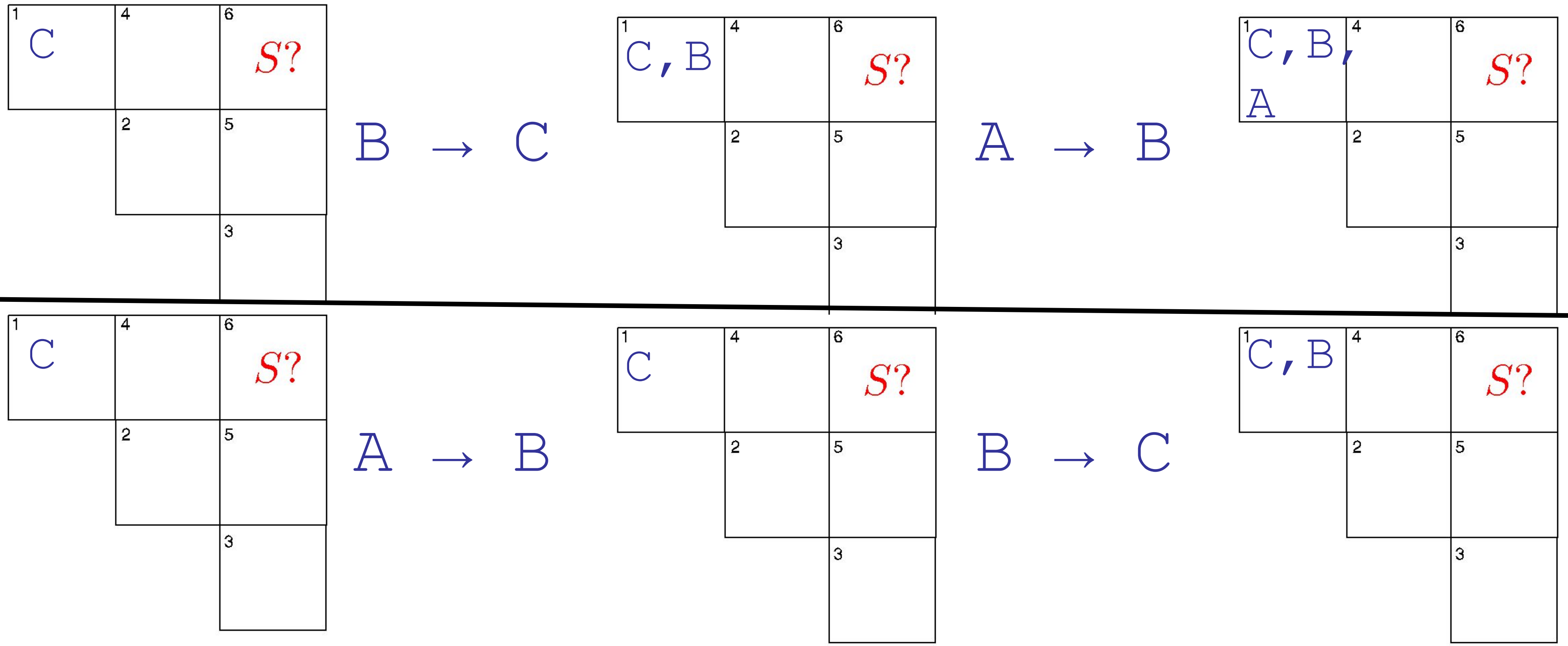
Why unary closure

$A \to B$

$B \to C$ ⇒ $A \to C$

scenario 1

scenario 2



Unary closure

- What if the grammar contained 2 rules:

$$A \to B$$
$$B \to C$$

- But C can be derived from A by a chain of rules:

$$A \to B \to C$$

- One could support chains in the algorithm but it is easier to extend the grammar, to get the transitive closure

$$\begin{array}{l} A \to B \\ B \to C \end{array} \quad \Rightarrow \quad \begin{array}{l} A \to B \\ B \to C \\ A \to C \end{array} \quad \begin{array}{l} A \to A \\ B \to B \\ C \to C \end{array}$$

Convenient for programming reasons in the PCFG case

Unary (reflexive transitive) closure

$$\begin{array}{ll} A \rightarrow B & 0.1 \\ B \rightarrow C & 0.2 \\ \ldots & \end{array} \quad \Rightarrow \quad \begin{array}{lr} A \rightarrow B & 0.1 \\ B \rightarrow C & 0.2 \\ A \rightarrow C & 0.2 \times 0.1 \end{array} \quad \begin{array}{lr} A \rightarrow A & 1 \\ B \rightarrow B & 1 \\ C \rightarrow C & 1 \end{array}$$

$$\ldots$$

Note that this is not a PCFG anymore as the rules do not sum to 1 for each parent

Unary (reflexive transitive) closure

The fact that the rule is composite needs to be stored to recover the true tree

$$
\begin{array}{ll}
A \rightarrow B & 0.1 \\
B \rightarrow C & 0.2 \\
\dots &
\end{array}
\Rightarrow
\begin{array}{ll}
A \rightarrow B & 0.1 \\
B \rightarrow C & 0.2 \\
A \rightarrow C & 0.2 \times 0.1 \\
& \dots
\end{array}
\quad
\begin{array}{ll}
A \rightarrow A & 1 \\
B \rightarrow B & 1 \\
C \rightarrow C & 1
\end{array}
$$

Note that this is not a PCFG anymore as the rules do not sum to 1 for each parent

Unary (reflexive transitive) closure

The fact that the rule is composite needs to be stored to recover the true tree

$$
\begin{array}{lr}
A \rightarrow B & 0.1 \\
B \rightarrow C & 0.2 \\
\ldots &
\end{array}
\quad \Rightarrow \quad
\begin{array}{lr}
A \rightarrow B & 0.1 \\
B \rightarrow C & 0.2 \\
A \rightarrow C & 0.2 \times 0.1
\end{array}
\quad
\begin{array}{lr}
A \rightarrow A & 1 \\
B \rightarrow B & 1 \\
C \rightarrow C & 1
\end{array}
$$

$\ldots$

Note that this is not a PCFG anymore as the rules do not sum to 1 for each parent

$$
\begin{array}{lr}
A \rightarrow B & 0.1 \\
B \rightarrow C & 0.2 \\
A \rightarrow C & 1.e-5
\end{array}
\quad \Rightarrow \quad
\begin{array}{lr}
A \rightarrow B & 0.1 \\
B \rightarrow C & 0.1 \\
A \rightarrow C & 0.02
\end{array}
\quad
\begin{array}{lr}
A \rightarrow A & 1 \\
B \rightarrow B & 1 \\
C \rightarrow C & 1
\end{array}
$$

What about loops, like: $A \rightarrow B \rightarrow A \rightarrow C$?

Recovery of the tree

- For each signature we store backpointers to the elements from which it was built
  - start recovering from [0, n, S]

- What do we store in backpointers?
  - rule
  - for binary rules, midpoint

- Be careful with unary rules
  - Basically you can assume that you always used an unary rule from the closure (but it could be the trivial one $C \to C$)

Speeding up the algorithm

- Basic pruning (roughly):
  - For every span (i,j) store only labels which have the probability at most N times smaller than the probability of the most probable label for this span
  - Check not all rules but only rules yielding subtree labels having non-zero probability
- Coarse-to-fine pruning
  - Parse with a smaller (simpler) grammar, and precompute (posterior) probabilities for each spans, and use only the ones with non-negligible probability from the previous grammar

Parsing evaluation

- Intrinsic evaluation:
  - Automatic: evaluate against annotation provided by human experts (gold standard) according to some predefined measure
  - Manual: … according to human judgment

- Extrinsic evaluation: score syntactic representation by comparing how well a system using this representation performs on some task
  - E.g., use syntactic representation as input for a semantic analyzer and compare results of the analyzer using syntax predicted by different parsers.

Standard evaluation setting in parsing

- Automatic intrinsic evaluation is used: parsers are evaluated against gold standard by provided by linguists
  - There is a standard split into the parts:
    - training set: used for estimation of model parameters
    - development set: used for tuning the model (initial experiments)
    - test set: final experiments to compare against previous work

Automatic evaluation of constituent parsers

- Exact match: percentage of trees predicted correctly
- Bracket score: scores how well individual phrases (and their boundaries) are identified

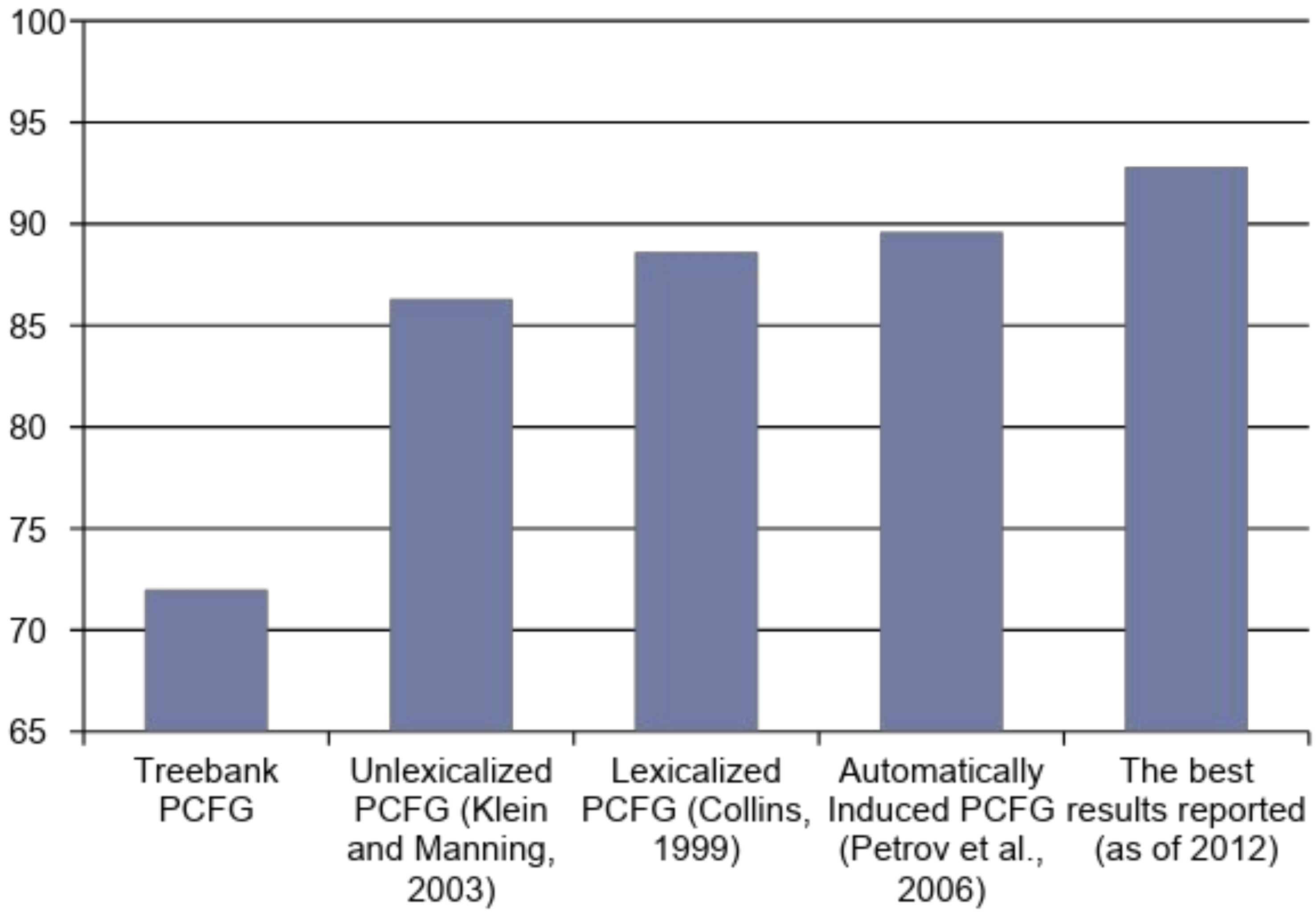
The most standard measure; we will focus on it

Brackets scores

Subtree signatures for CKY

- The most standard score is **bracket score**
- It regards a tree as a collection of brackets: $[min, max, C]$
- The set of brackets predicted by a parser is compared against the set of brackets in the tree annotated by a linguist
- **Precision, recall** and **F1** are used as scores

Preview: F1 bracket score



Estimating PCFGs

Estimating PCFGs

Associate probabilities with the rules : $p(X \rightarrow \alpha)$

$$\forall \ X \rightarrow \alpha \in R: \quad 0 \leq p(X \rightarrow \alpha) \leq 1$$

$$\forall X \in N: \quad \sum_{\alpha: X \rightarrow \alpha \in R} p(X \rightarrow \alpha) = 1$$

$S \rightarrow NP \ VP$	1.0	(NP A girl) (VP ate a sandwich)
$VP \rightarrow V$	0.2	
$VP \rightarrow V \ NP$	0.4	(VP ate) (NP a sandwich)
$VP \rightarrow VP \ PP$	0.4	(VP saw a girl) (PP with …)
$NP \rightarrow NP \ PP$	0.3	(NP a girl) (PP with ….)
$NP \rightarrow D \ N$	0.5	(D a) (N sandwich)
$NP \rightarrow PN$	0.2	
$PP \rightarrow P \ NP$	1.0	(P with) (NP with a sandwich)

$N \rightarrow girl$	0.2
$N \rightarrow telescope$	0.7
$N \rightarrow sandwich$	0.1
$PN \rightarrow I$	1.0
$V \rightarrow saw$	0.5
$V \rightarrow ate$	0.5
$P \rightarrow with$	0.6
$P \rightarrow in$	0.4
$D \rightarrow a$	0.3
$D \rightarrow the$	0.7

Estimating PCFGs: Intuition

- Probabilistic Regular Grammar

$$N^i \rightarrow w^j N^k$$

$$N^i \rightarrow w^j$$

Start state, N^1

Estimating PCFGs: Intuition

- Probabilistic Regular Grammar

$N^i \rightarrow w^j N^k$

$N^i \rightarrow w^j$

Start state, N^1

[Credit: Chris Manning]

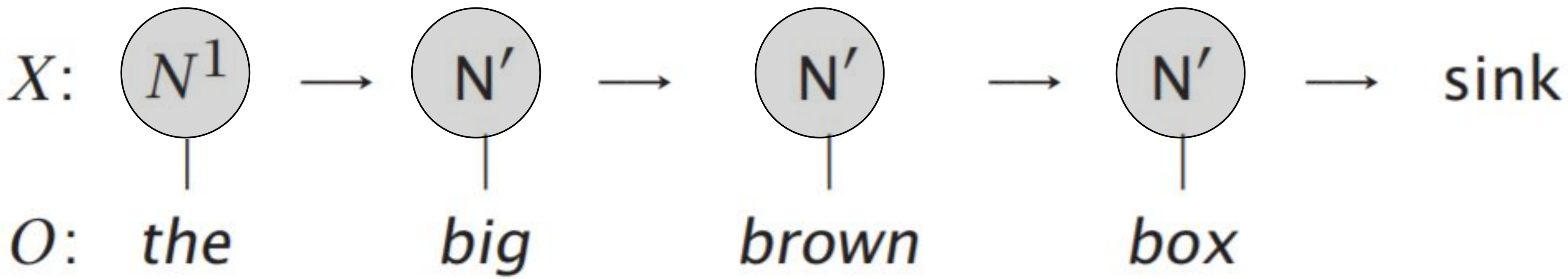
Estimating PCFGs: Intuition

- Probabilistic Regular Grammar

$$N^i \to w^j N^k$$

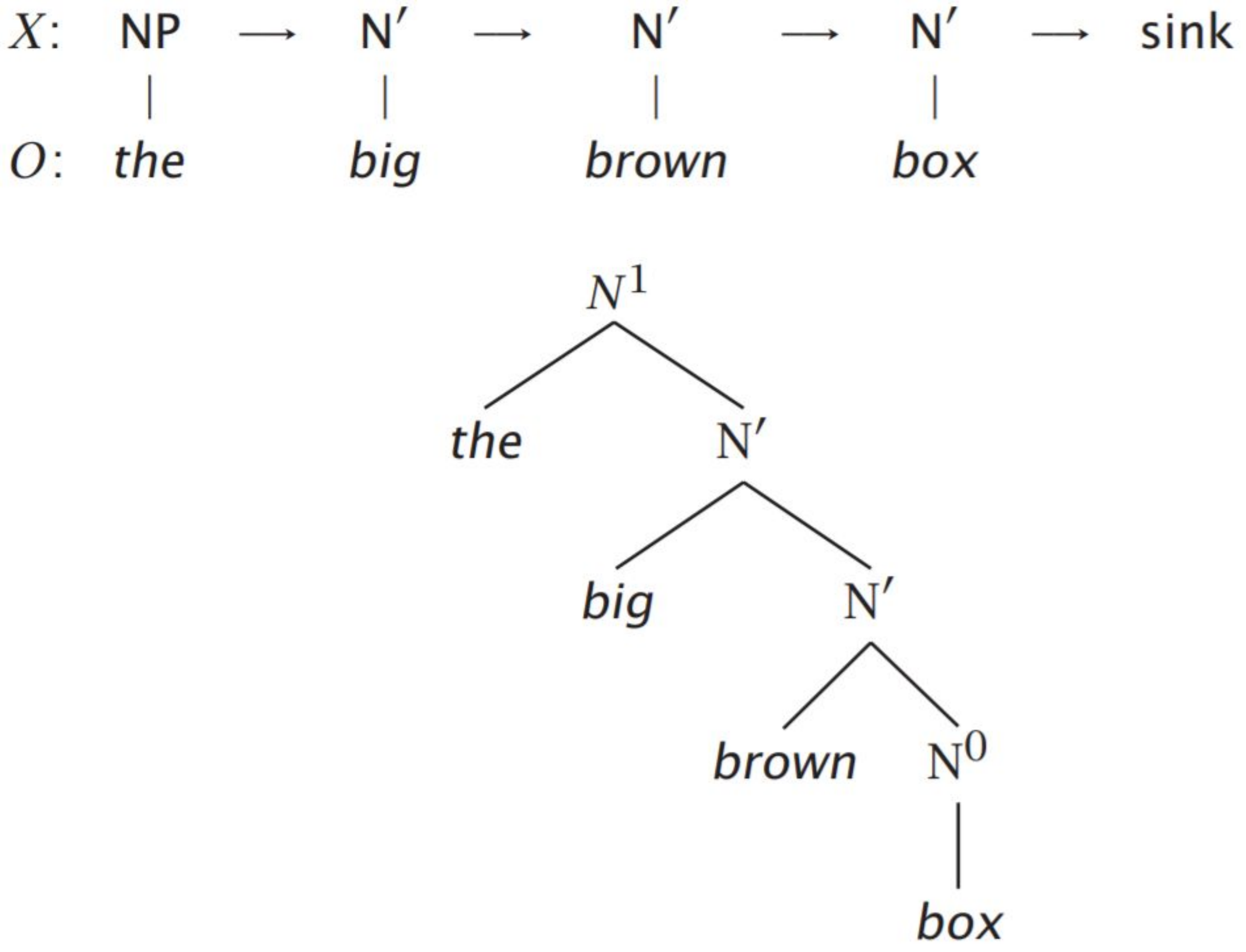
$$N^i \to w^j$$

Start state, N^1



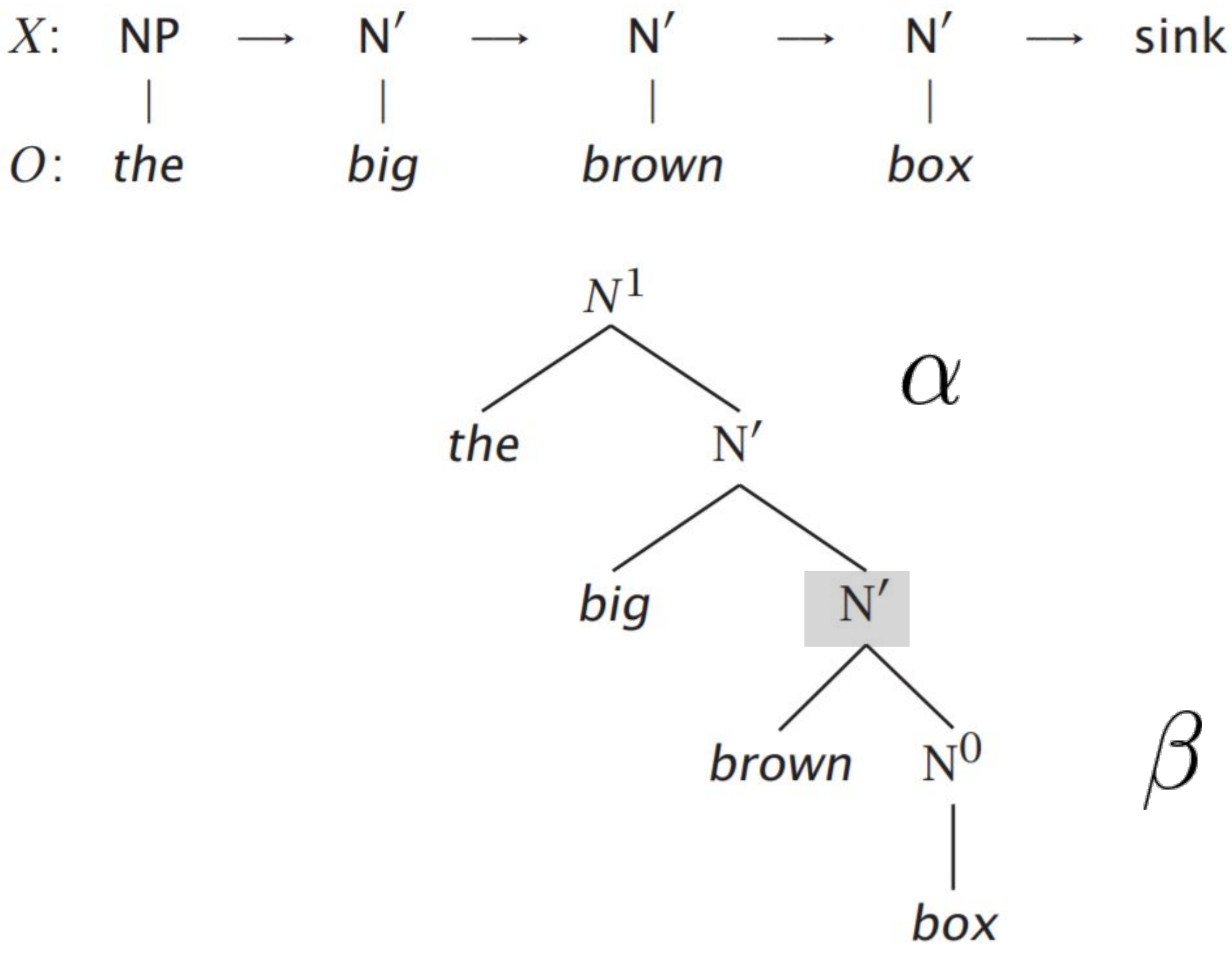
[Credit: Chris Manning]

Estimating PCFGs: Intuition



[Credit: Chris Manning]

Estimating PCFGs: Intuition



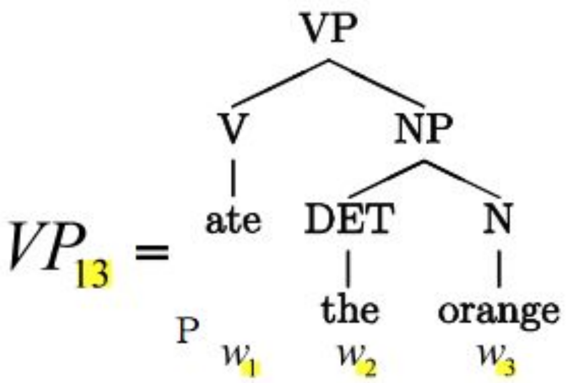
[Credit: Chris Manning]

Unsupervised estimation of PCFGs

- Notation
- Calculating inside probabilities
- Calculating outside probabilities
- The inside-outside algorithm (EM) - preview

Notation

- Non-terminal symbols (latent variables): $\{N^1, \ldots, N^n\}$
- Sentence (observed data): $\{w_1, \ldots, w_m\} = w_{1m}$
- N^j_{pq} denotes that N^j spans w_{pq} in the sentence



Inside probability

- Definition (compare with backward prob for HMMs):

$$\beta_j(p,q) = P(w_p, \ldots, w_q | N^j_{pq}, G) = P(N^j_{pq} \to w_{pq} | G)$$

- Computed recursively
  - Base case: $\beta_j(k,k) = P(w_k | N^j_{kk}, G) = P(N_j \to w_k | G)$
  - Induction:

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

The grammar is binarized

let's draw...

Implementation: PCFG parsing

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double best = undefined
        for each binary rule C -> C1 C2
          for each mid from min + 1 to max - 1
            double t1 = chart[min][mid][C1]
            double t2 = chart[mid][max][C2]
            double candidate = t1 * t2 * p(C -> C1 C2)
            if candidate > best then
              best = candidate
      chart[min][max][C] = best
```

Implementation: inside

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double total = 0.0
      for each binary rule C -> C1 C2
        for each mid from min + 1 to max - 1
          double t1 = chart[min][mid][C1]
          double t2 = chart[mid][max][C2]
          double candidate = t1 * t2 * p(C -> C1 C2)
          total = total + candidate
      chart[min][max][C] = total
```

Implementation: inside

```
for each max from 2 to n

  for each min from max - 2 down to 0

    for each syntactic category C
      double total = 0.0

      for each binary rule C -> C1 C2

        for each mid from min + 1 to max - 1

          double t1 = chart[min][mid][C1]

          double t2 = chart[mid][max][C2]

          double candidate = t1 * t2 * p(C -> C1 C2)

          total = total + candidate

      chart[min][max][C] = total
```

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

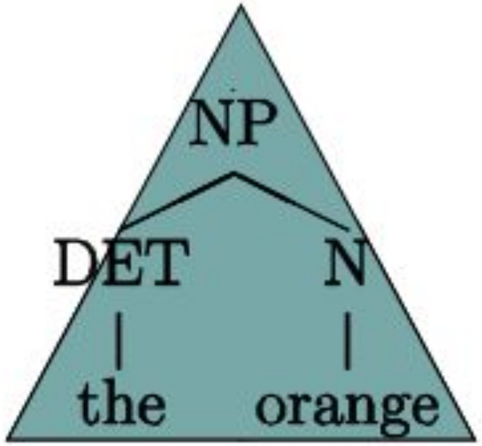
Implementation: inside

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      double total = 0.0
      for each binary rule C -> C1 C2
        for each mid from min + 1 to max - 1
          double t1 = chart[min][mid][C1]
          double t2 = chart[mid][max][C2]
          double candidate = t1 * t2 * p(C -> C1 C2)

          total = total + candidate

      chart[min][max][C] = total
```

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

Inside probability: example

NP→DET N	0.8	NP→N	0.2
DET→a	0.6	DET→the	0.4
N→apple	0.8	N→orange	0.2



$\beta_{DET}(1,1)$

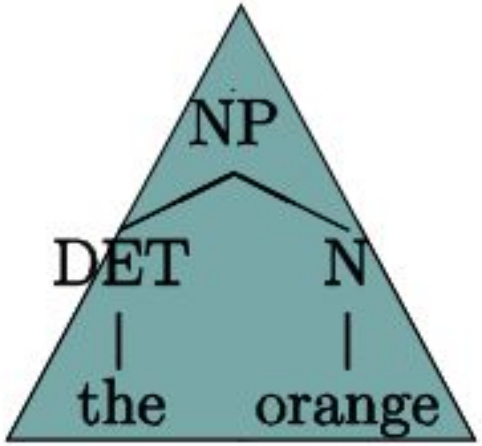
$\beta_{N}(2,2)$

$\beta_{NP}(1,2)$

$\beta_{NP}(1,2)$

Inside probability: example

NP→DET N	0.8	NP→N	0.2
DET→a	0.6	DET→the	0.4
N→apple	0.8	N→orange	0.2



$$\beta_{DET}(1,1) = P(the \mid DET_{11}, G) = P(DET \rightarrow the \mid G) = 0.4$$

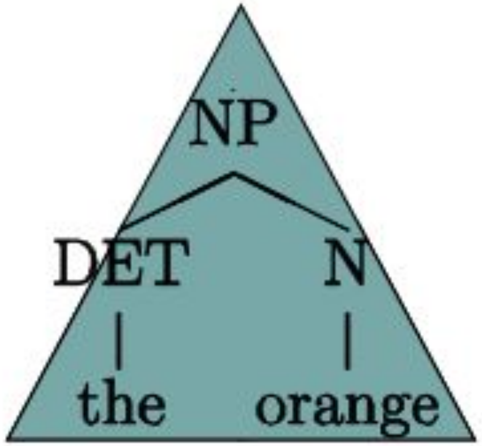
$$\beta_{N}(2,2)$$

$$\beta_{NP}(1,2)$$

$$\beta_{NP}(1,2)$$

Inside probability: example

NP→DET N	0.8	NP→N	0.2
DET→a	0.6	DET→the	0.4
N→apple	0.8	N→orange	0.2



$$\beta_{DET}(1,1) = P(the \mid DET_{11}, G) = P(DET \rightarrow the \mid G) = 0.4$$

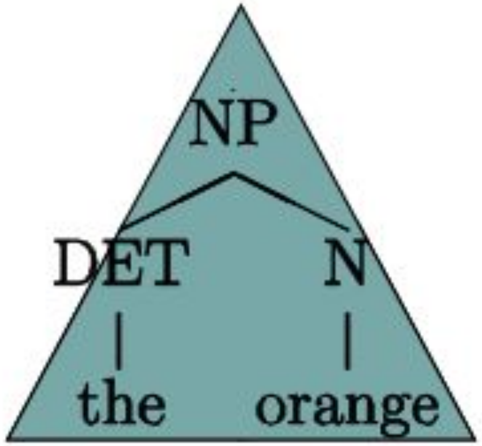
$$\beta_{N}(2,2) = P(N \rightarrow orange \mid G) = 0.2$$

$$\beta_{NP}(1,2)$$

$$\beta_{NP}(1,2)$$

Inside probability: example

NP→DET N	0.8	NP→N	0.2
DET→a	0.6	DET→the	0.4
N→apple	0.8	N→orange	0.2



$$\beta_{DET}(1,1) = P(the \mid DET_{11}, G) = P(DET \rightarrow the \mid G) = 0.4$$

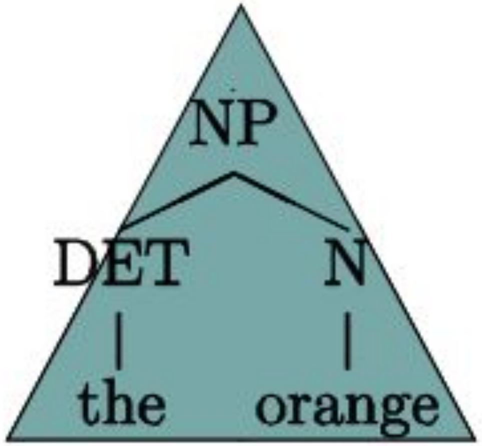
$$\beta_{N}(2,2) = P(N \rightarrow orange \mid G) = 0.2$$

$$\beta_{NP}(1,2) = P(NP \rightarrow DET \cdot N)\beta_{DET}(1,1)\beta_{N}(2,2)$$
$$= 0.8 \times 0.4 \times 0.2$$

$$\beta_{NP}(1,2) = 0.064$$

Inside probability: example

NP→DET N	0.8	NP→N	0.2
DET→a	0.6	DET→the	0.4
N→apple	0.8	N→orange	0.2



$$\beta_{DET}(1,1) = P(the \mid DET_{11}, G) = P(DET \rightarrow the \mid G) = 0.4$$

$$\beta_{N}(2,2) = P(N \rightarrow orange \mid G) = 0.2$$

$$\beta_{NP}(1,2) = P(NP \rightarrow DET \cdot N)\beta_{DET}(1,1)\beta_{N}(2,2)$$

$$= 0.8 \quad \times 0.4 \quad \times 0.2$$

$$\beta_{NP}(1,2) = 0.064$$

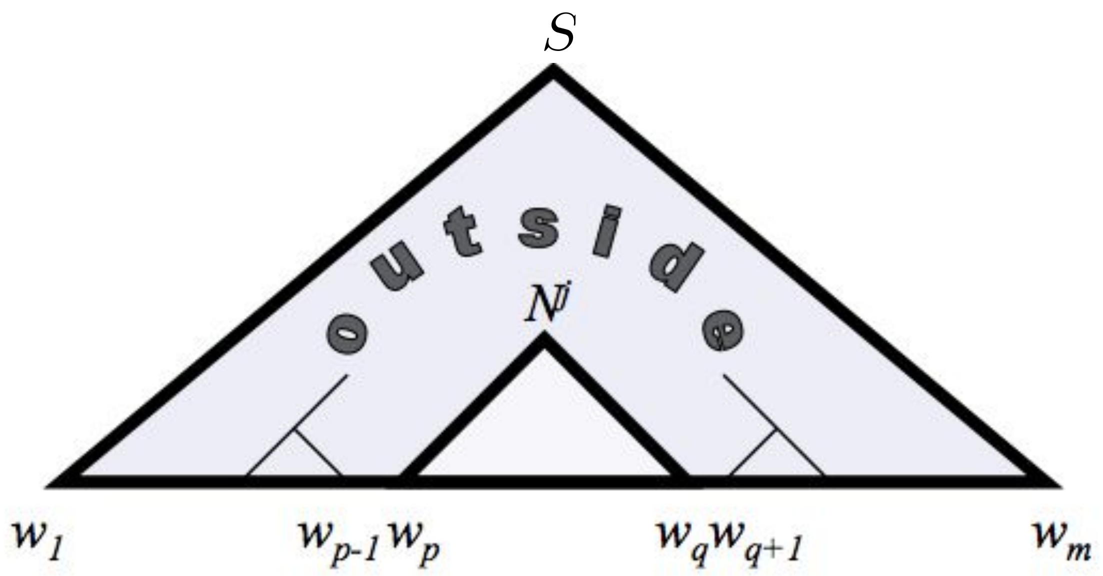
$$\beta_S(1,m) = P(S \rightarrow w_1, \ldots, w_m | G)$$

Outside probability

- Definition (compare with forward prob for HMMs):

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

- The joint probability of starting with *S*, generating words $w_1, \ldots, w_{p-1}$, the non terminal N^j and words $w_{q+1}, \ldots, w_m$.

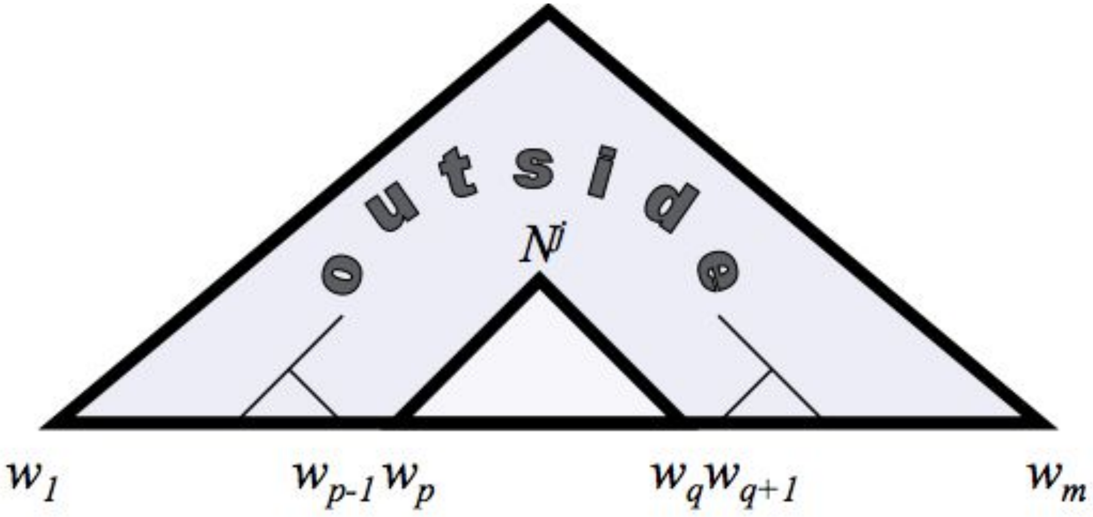


Calculating outside probability

- Computed recursively, base case

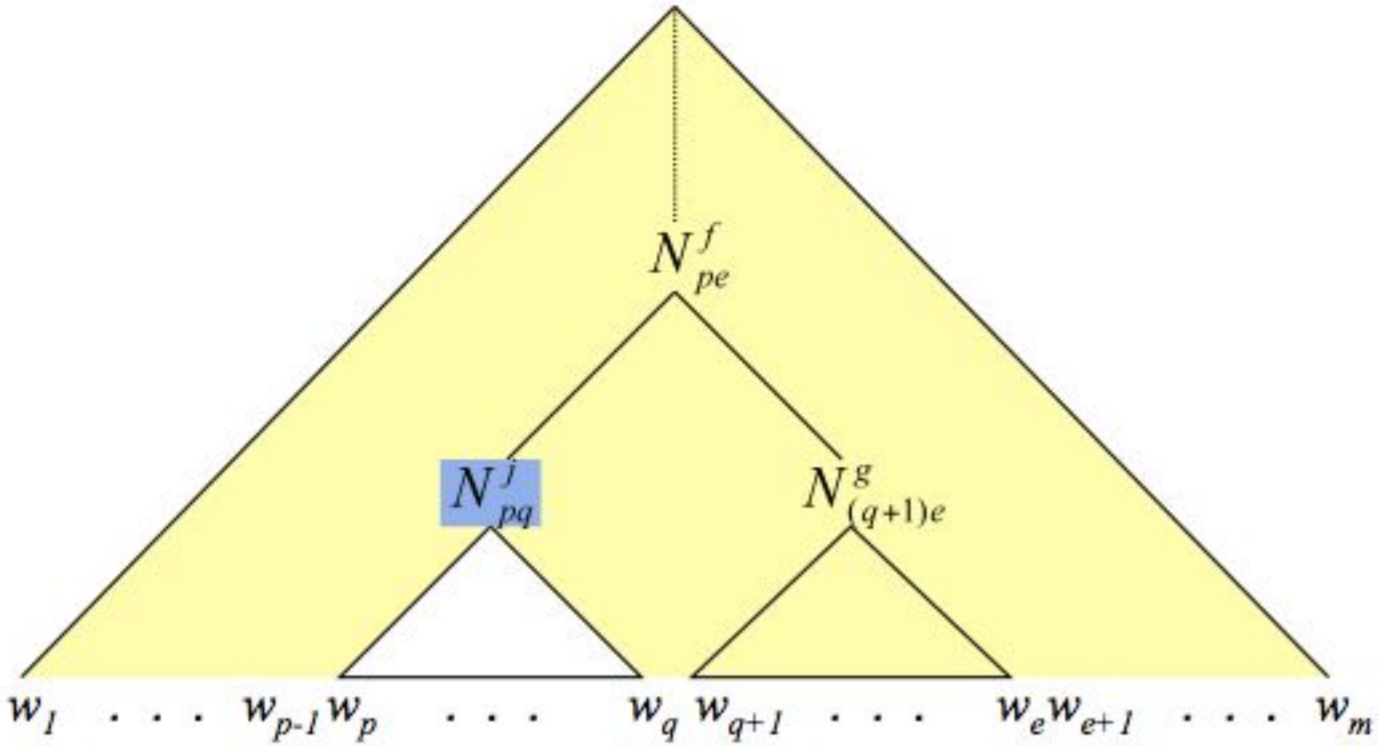
$$\alpha_1(1,m) = \alpha_S(1,m) = 1 \qquad\qquad \alpha_{j\neq 1}(1,m) = 0$$

- Induction?
- Intuition: N^j_{pq} must be either the L or R child of a parent node. We first consider the case when it is the L child.



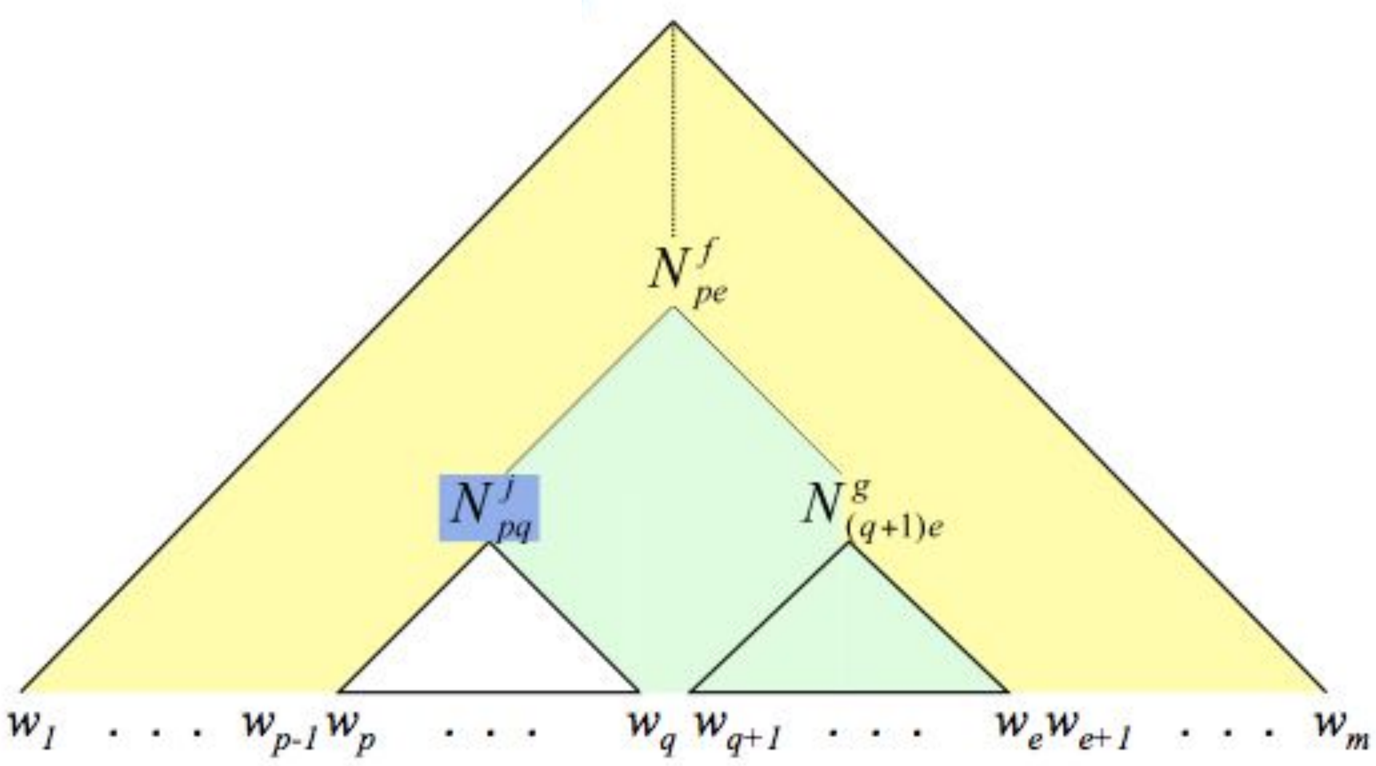
Calculating outside probability

- The yellow area is the probability we would like to calculate
  - How do we decompose it?



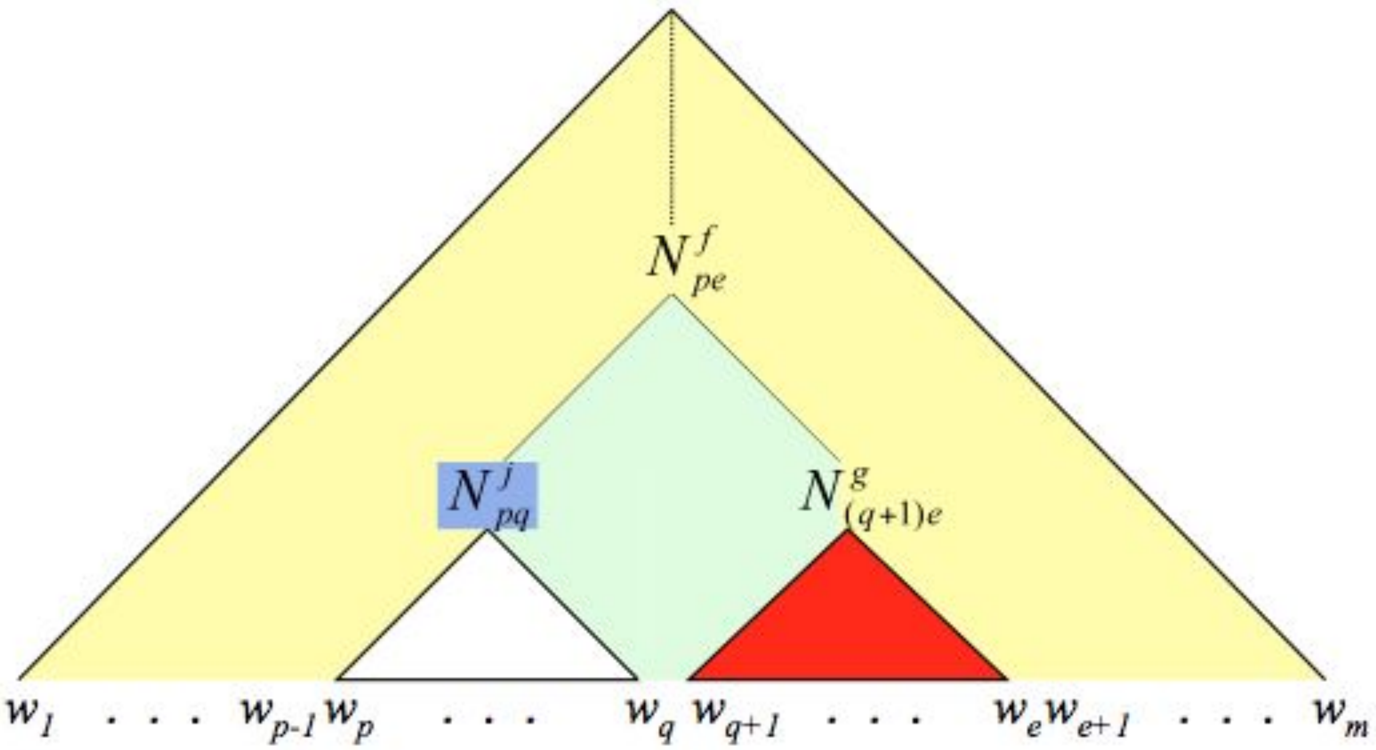
Calculating outside probability

- Step 1: We assume that N^f_{pe} is the parent of N^j_{pq}. Its outside probability, $\alpha_f(p, e)$ (represented by the yellow shading) is available recursively. But how do we compute the green part?



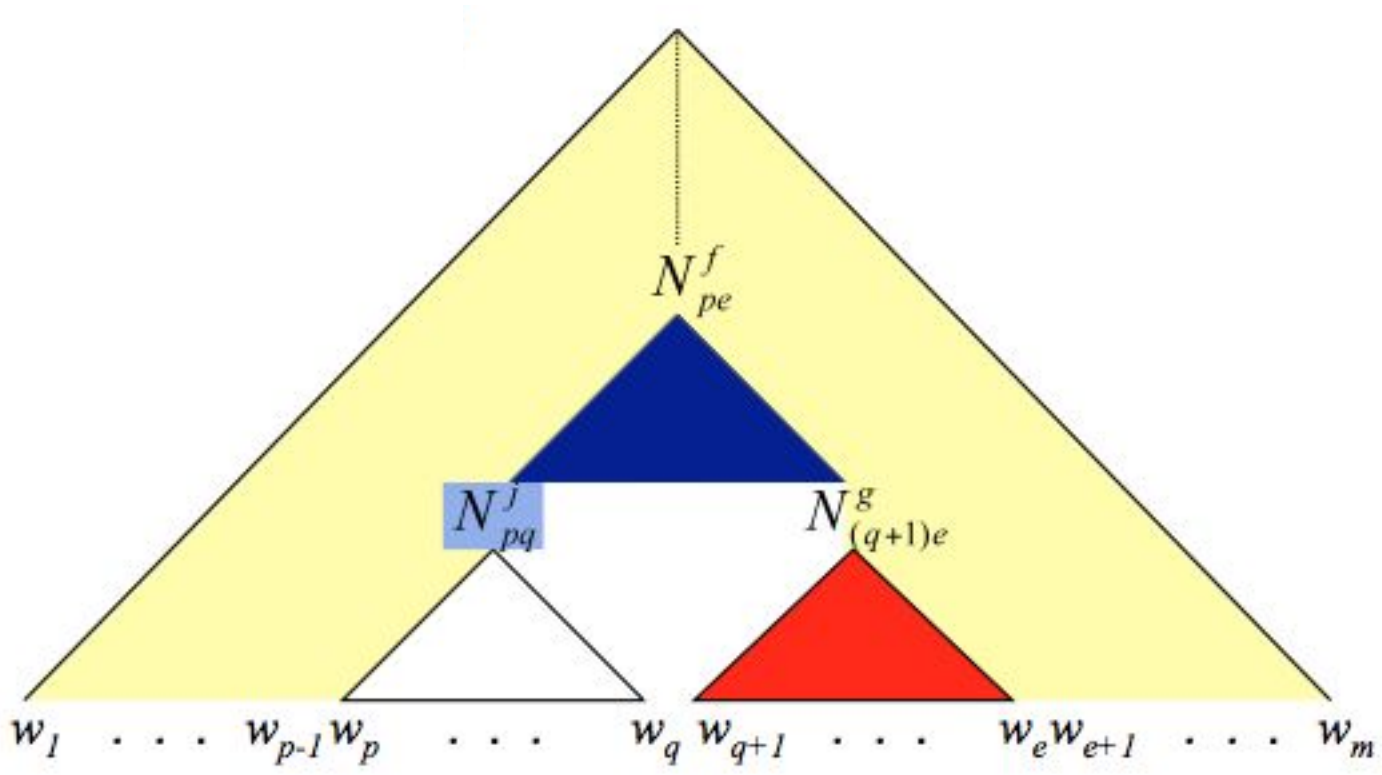
Calculating outside probability

- Step 1: The red shaded area is the inside probability for $N^g_{(q+1)\epsilon}$, i.e. $\beta_g(q+1, e)$



Calculating outside probability

- Step 3: The blue shaded area is just the production $N^f \rightarrow N^j N^g$, the corresponding probability $P(N^f \rightarrow N^j N^g | N^f, G)$

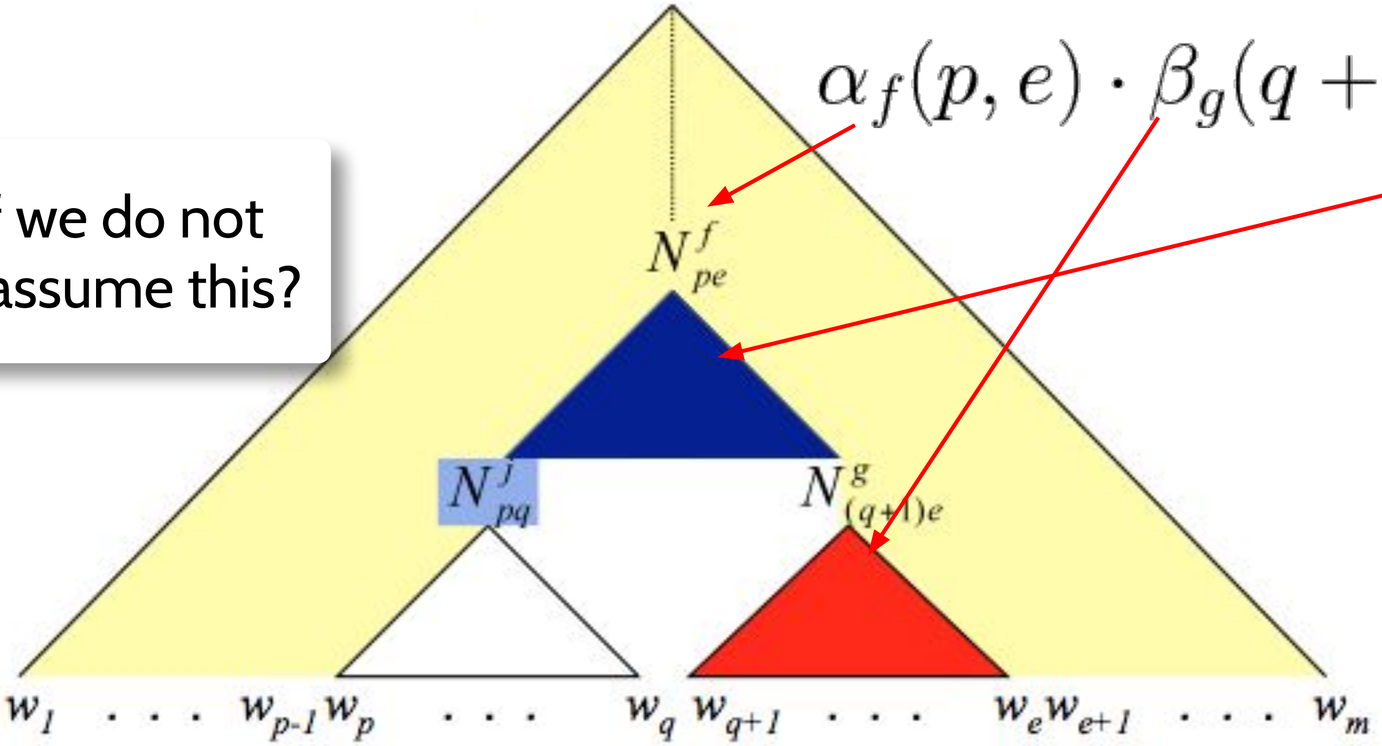


Calculating outside probability

- If we multiply the terms together, we have the joint probability corresponding to the yellow, red and blue areas, assuming N^j was the L child of N^f, and give fixed non-terminals *f* and *g*, as well as a fixed partition *e*

$$\alpha_f(p,e) \cdot \beta_g(q+1,e) \cdot P(N^f \rightarrow N^j N^g)$$

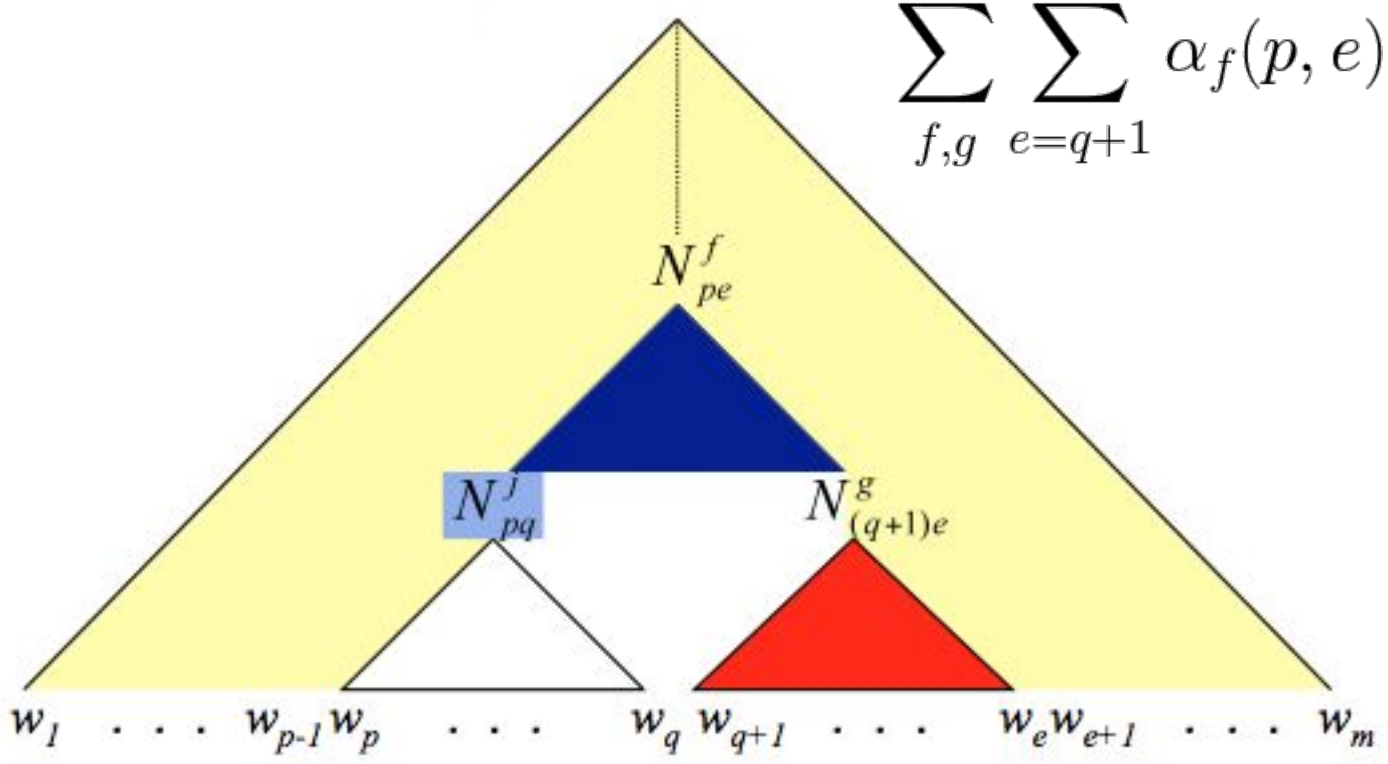
What if we do not want to assume this?



Calculating outside probability

- The joint probability corresponding to the yellow, red and blue areas, assuming N^j was the L child of some non-terminal:

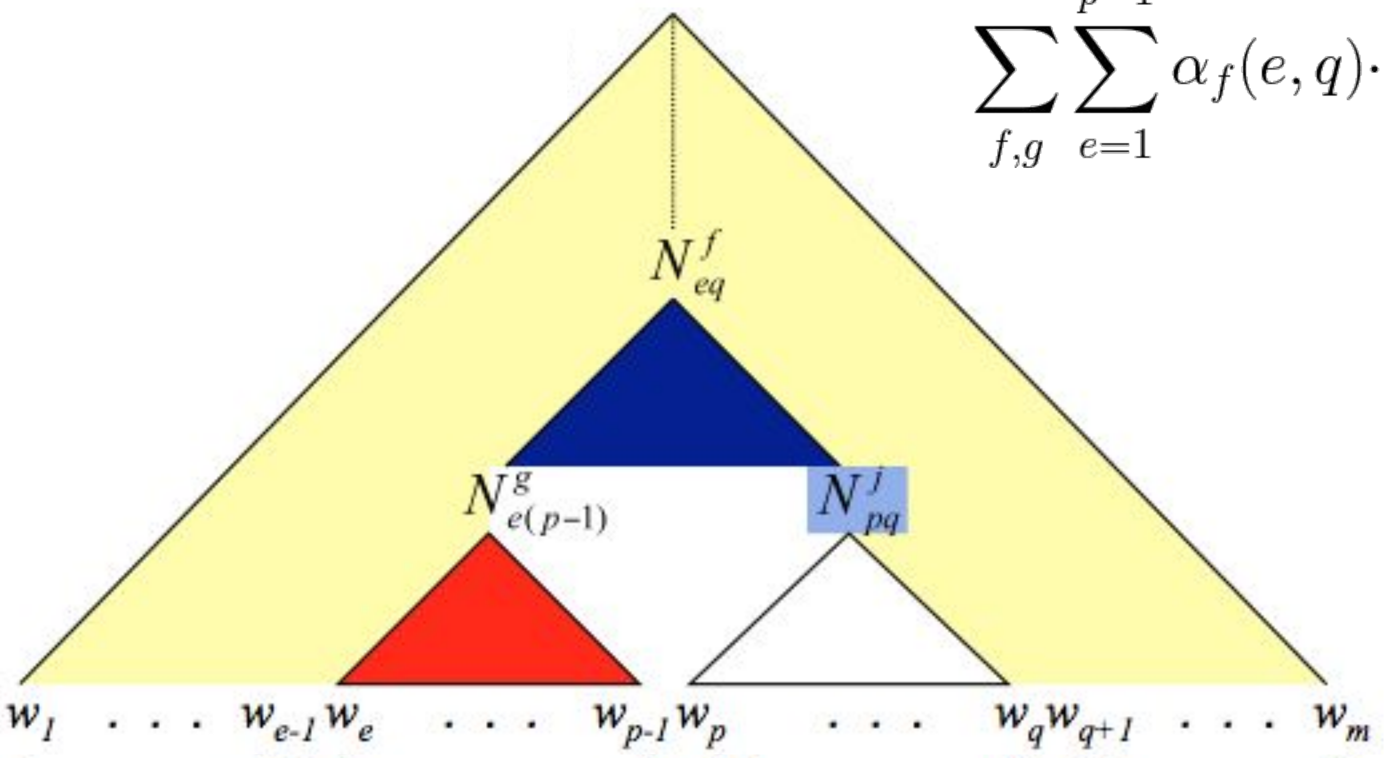
$$\sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p,e) \cdot \beta_g(q+1,e) \cdot P(N^f \rightarrow N^j N^g)$$



Calculating outside probability

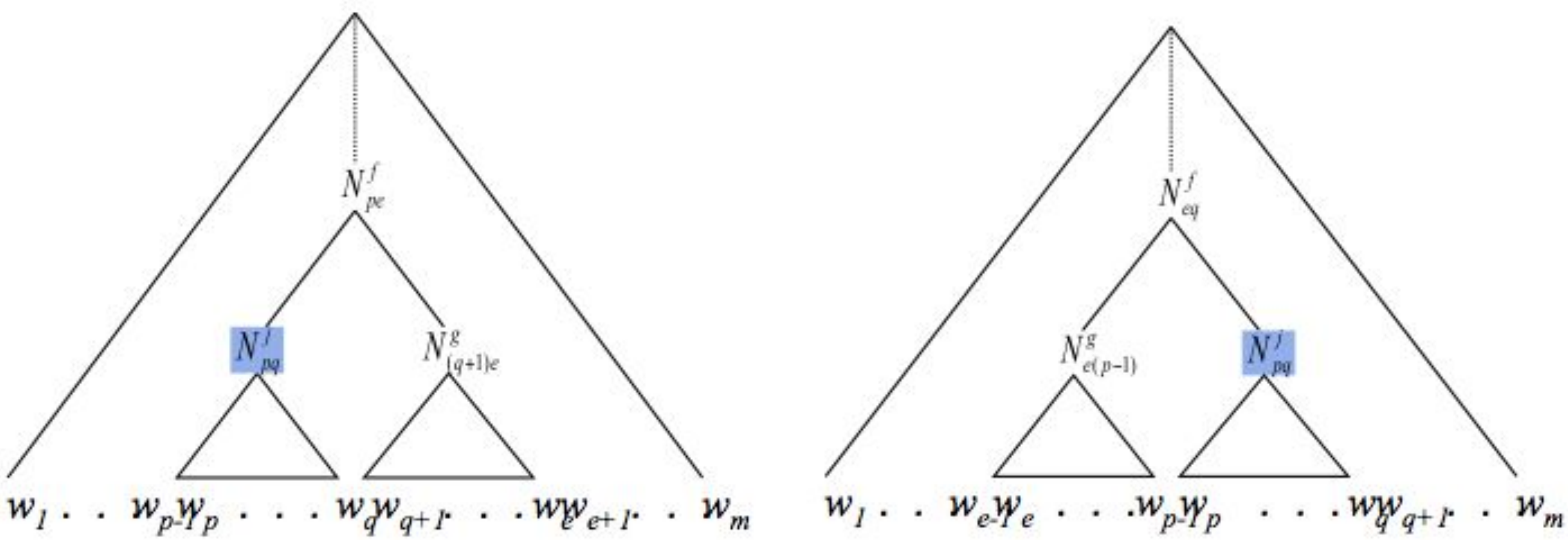
- The joint probability corresponding to the yellow, red and blue areas, assuming N^j was the R child of some non-terminal:

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



Calculating outside probability

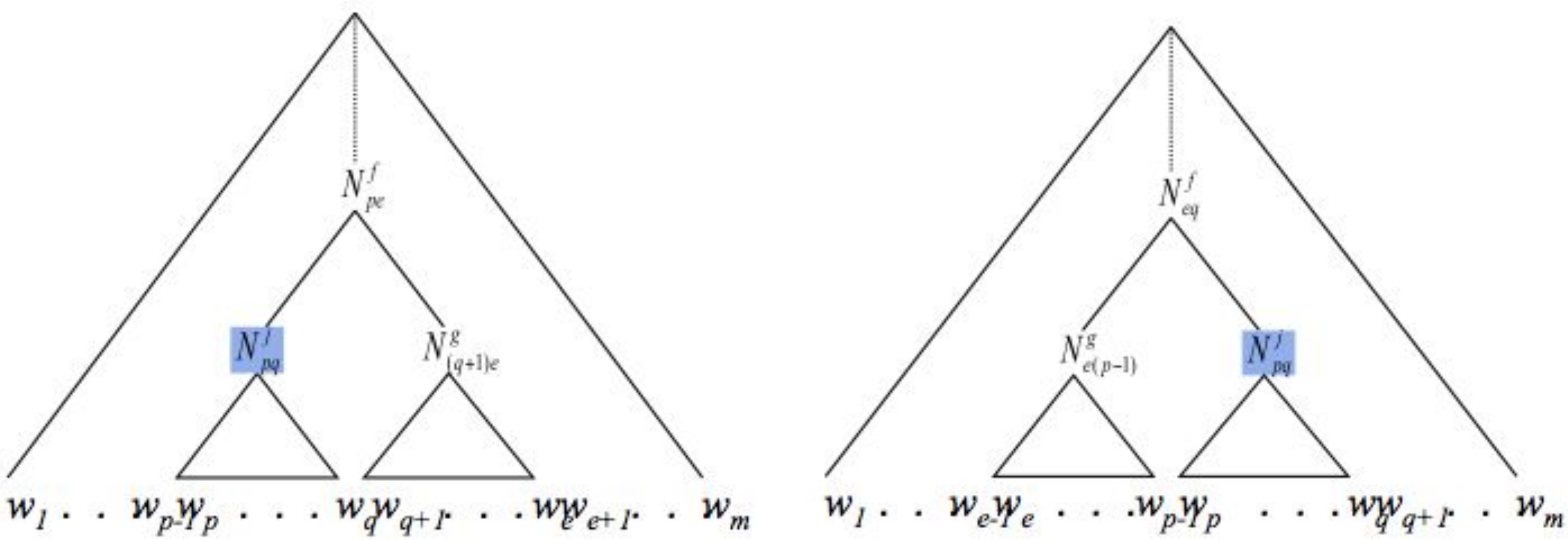
- The joint final joint probability (the sum over the L and R cases):



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

Calculating outside probability

- The joint final joint probability (the sum over the L and R cases):



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

Inside-outside algorithm

- For PCFGs we need to compute:

$$\theta^t = P(N^j \rightarrow N^r N^s | N^j)$$

EM

- Given two events, x and y, the maximum likelihood estimation (MLE) for their conditional probability is:

$$P(x \mid y) = \frac{count(x, y)}{count(x)}$$

- If they are observable, it’s easy to see what to do: just count the events in a representative corpus and use the MLE

EM

- What these are hidden variables that cannot be observed directly?

- Use a model μ and iteratively improve the model based on a corpus of observable data (O) generated by the hidden variables:

$$P_{\hat{\mu}}(x \mid y) = \frac{E_{\mu}[count(x, y) \mid O]}{E_{\mu}[count(x) \mid O]}$$

- It is worth noting that if you know how to calculate the numerator, the denominator is trivially derivable.

EM

- By updating μ and iterating, the model converges to at least a local maximum
- This can be proven, but I will not do it here.

The inside-outside algorithm

- Goal: estimate a model μ that is a PCFG (in Chomsky normal form) that characterizes a corpus of text.

- Required input:
  - Size of non-terminal vocabulary, *n*
  - At least one sentence to be modeled, O

The inside-outside algorithm

- Stated with the general schema described earlier, we seek to the MLE probabilities for productions in the grammar

$$P(N^j \rightarrow N^r N^s \mid N^j) = \frac{count(N^j \rightarrow N^r N^s, N^j)}{count(N^j)}$$

- (Observe that this would be trivially easy to calculate this with a treebank, since the non-terminals are observable in a treebank)

The inside-outside algorithm

- Since the non-terminals are not visible, we can use EM to estimate the probabilities iteratively:

$$P_{\hat{\mu}}(N^j \rightarrow N^r N^s \mid N^j) = \frac{E_\mu[count(N^j \rightarrow N^r N^s, N^j) \mid O]}{E_\mu[count(N^j) \mid O]}$$

To be continued...

- Next: recitation on EM