Natural Language Models and Interfaces

Part B, lecture 2

Ivan Titov

Institute for Logic, Language and Computation

Universiteit van Amsterdam

Today

- Parsing algorithms for CFGs
  - Recap, Chomsky Normal Form (CNF)
  - A dynamic programming algorithm for parsing (CKY)
  - Extension of CKY to support unary inner rules
- Parsing for PCFGs
  - Extension of CKY for parsing with PCFGs
- Parser evaluation (if we have time)

After this lecture you should be able to start working on the assignment step 2.1

Parsing

- Parsing is search through the space of all possible parses
  - e.g., we may want either any parse, all parses or the highest scoring parse (if PCFG):

$$\arg\max_{T \in G(x)} P(T)$$

The probability by the PCFG model

Set of all trees given by the grammar for the sentence *x*

- Bottom-up:
  - One starts from words and attempt to construct the full tree
- Top-down
  - Start from the start symbol and attempt to expand to get the sentence

CKY algorithm (aka CYK)

- Cocke-Kasami-Younger algorithm
  - Independently discovered in late 60s / early 70s
- An efficient bottom up parsing algorithm for (P)CFGs
  - can be used both for the recognition and parsing problems
- Very important in NLP (and beyond)

- We will start with the non-probabilistic version

Constraints on the grammar

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

$$C \to x$$

Unary preterminal rules (generation of words given PoS tags $N \to telescope$, $D \to the$, …)

$$C \to C_1 C_2$$

Binary inner rules (e.g., $S \to NP\ VP,\ NP \to D\ \ N$)

- Any CFG can be converted to an equivalent CNF
  - Equivalent means that they define the same language
  - However (syntactic) trees will look differently

Makes linguists unhappy

  - It is possible to address it but defining such transformations that allows for easy reverse transformation

Transformation to CNF form

- What one need to do to convert to CNF form
  - Get rid of empty (aka epsilon) productions: $C \rightarrow \epsilon$
  - Get rid of unary rules: $C \rightarrow C_1$
  - N-ary rules: $C \rightarrow C_1\ C_2 \dots C_n \quad (n > 2)$

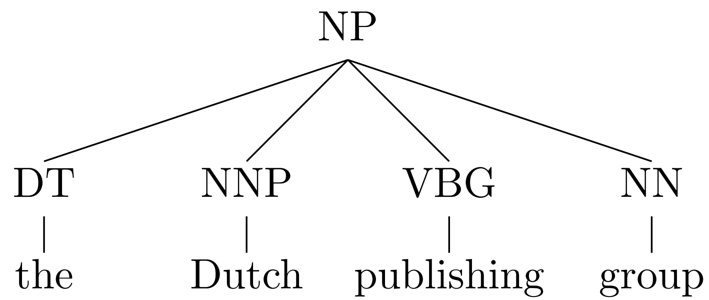
Generally not a problem as there are not empty production in the standard (postporcessed) treebanks

Not a problem, as our CKY algorithm will support unary rules

Crucial to process them, as required for efficient parsing

Transformation to CNF form: binarization

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



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

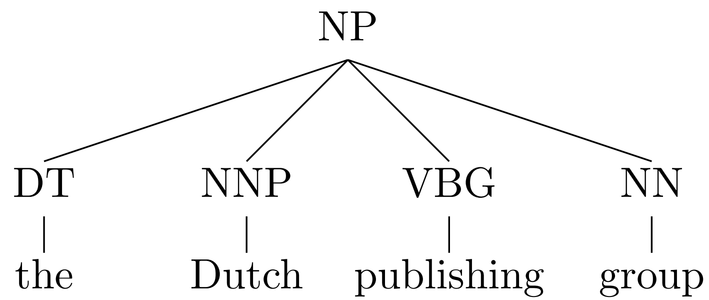
$$NP \rightarrow DT \;\; X$$
$$X \rightarrow NNP \;\; Y$$
$$Y \rightarrow VBG \;\; NN$$

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

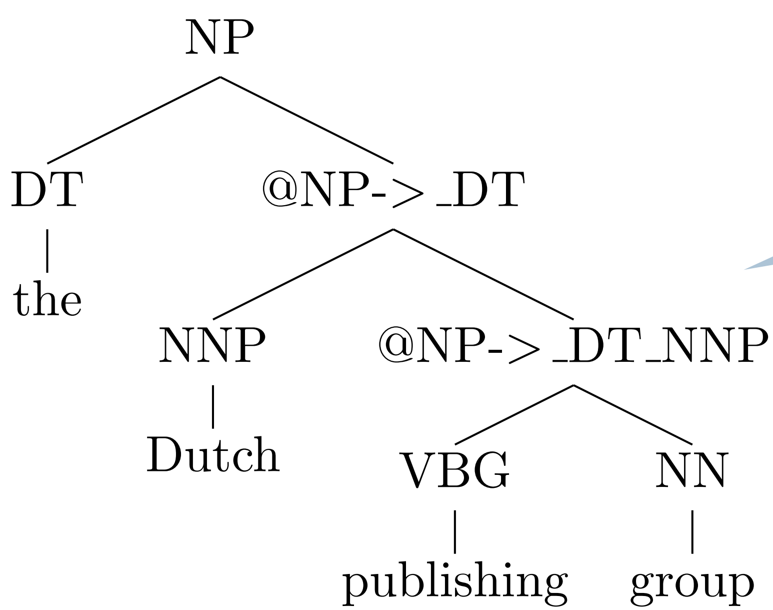
$$NP \rightarrow DT \;\; @NP|DT$$
$$@NP|DT \rightarrow NNP \;\; @NP|DT\_NNP$$
$$@NP|DT\_NNP \rightarrow VBG \;\; NN$$

Transformation to CNF form: binarization

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



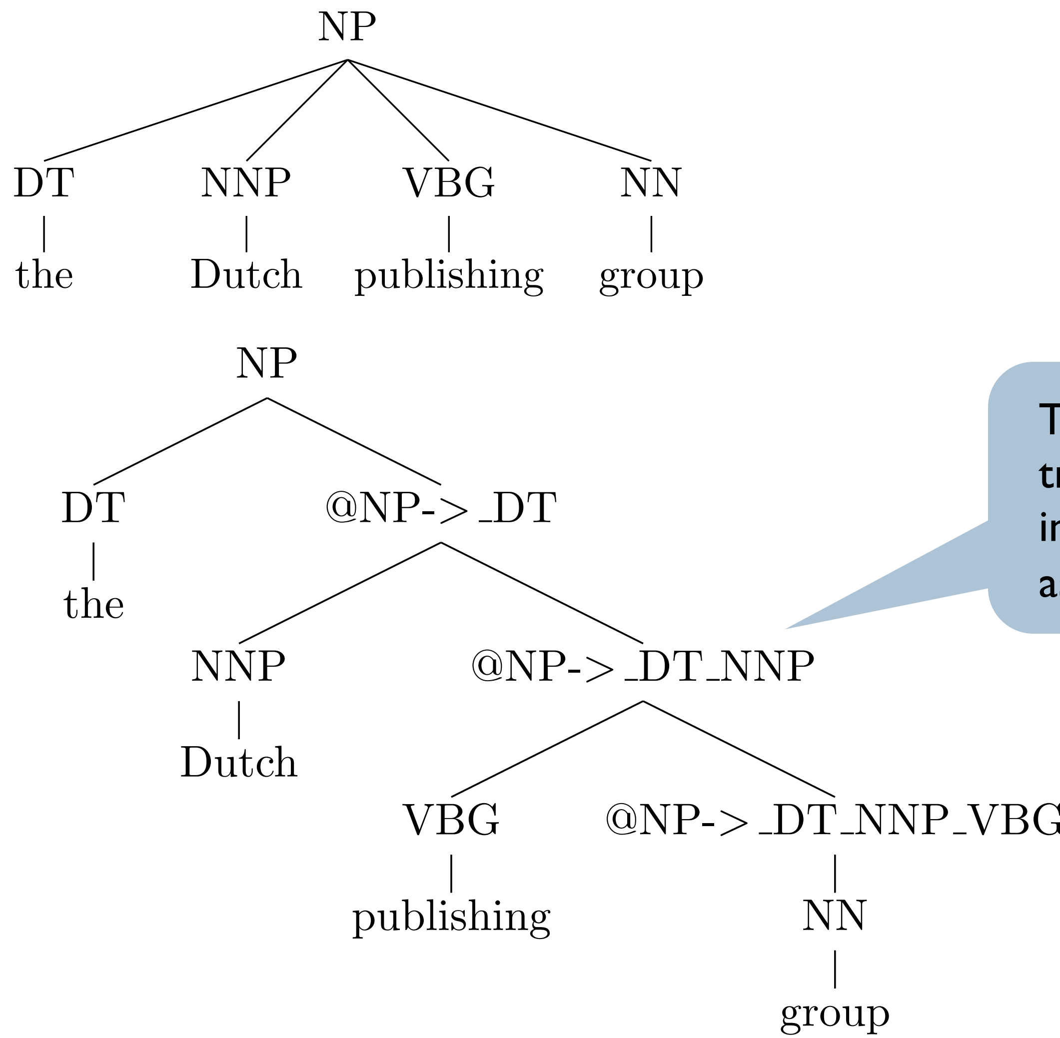
Also known as **lossless Markovization** in the context of PCFGs



Can be easily reversed on postprocessing

Transformation to CNF form: binarization

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



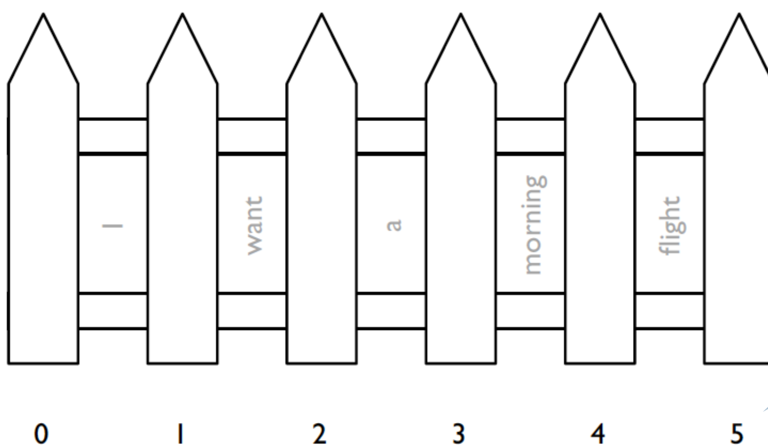
This is exactly the transformation used in the code of the assignment

CKY: Parsing task

[Some illustrations and slides in this lecture are from Marco Kuhlmann]

start symbol

- We a given
  - a grammar $G = (V, \Sigma, R, S)$
  - a sequence of words $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$
- Our goal is to produce a parse tree for $\boldsymbol{w}$
- We need an easy way to refer to substrings of $\boldsymbol{w}$



indices refer to fenceposts

span (i, j) refers to words between fenceposts *i* and *j*

Key problems

- Recognition problem: does the sentence belong to the language defined by CFG?
  - Is there a derivation which yields the sentence?
- Parsing problem: what is a derivation (tree) corresponding the sentence?
  - Probabilistic parsing: what is the most probable tree for the sentence?

Parsing one word

$$C \rightarrow w_i$$

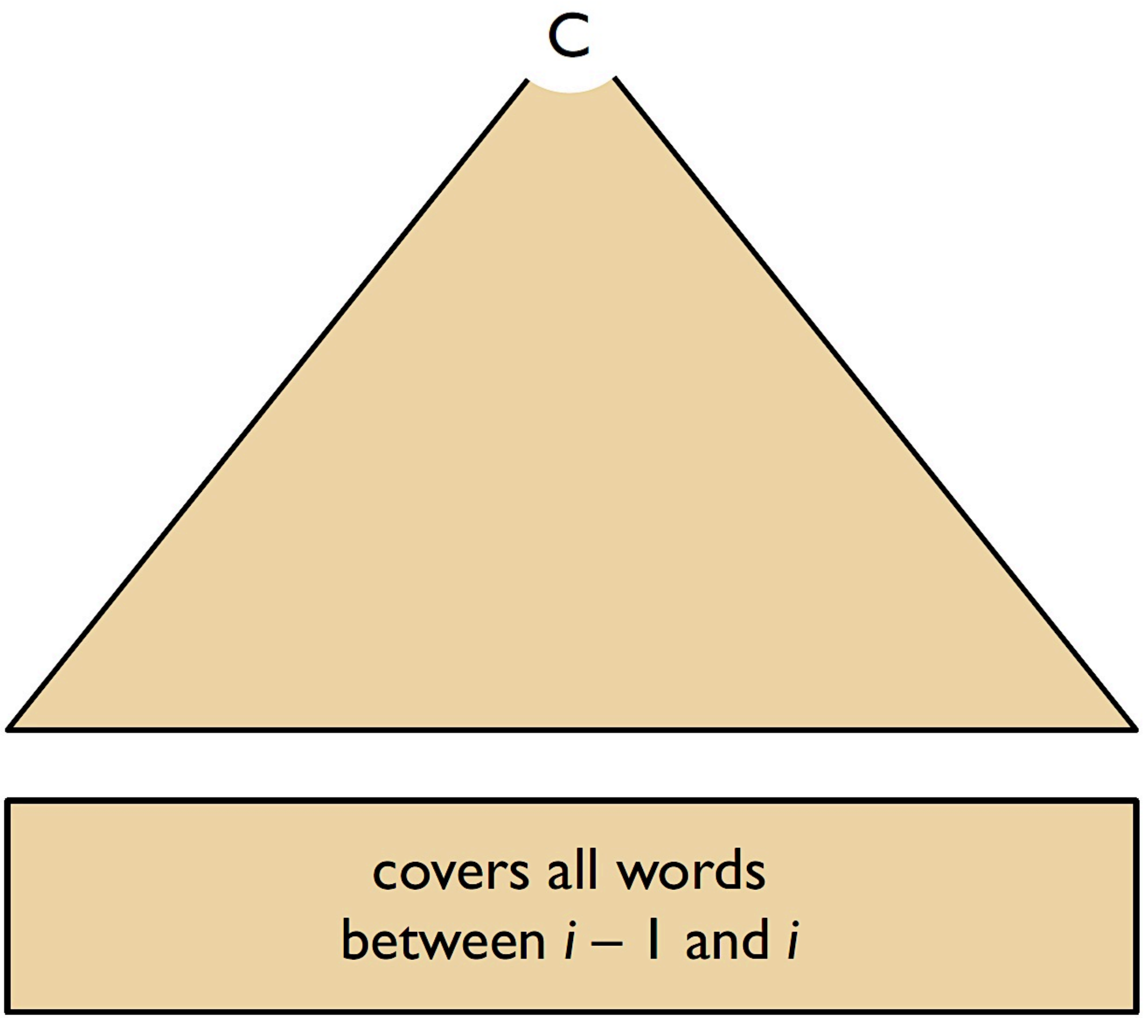
w_i

Parsing one word

$$C \rightarrow w_i$$

Parsing one word

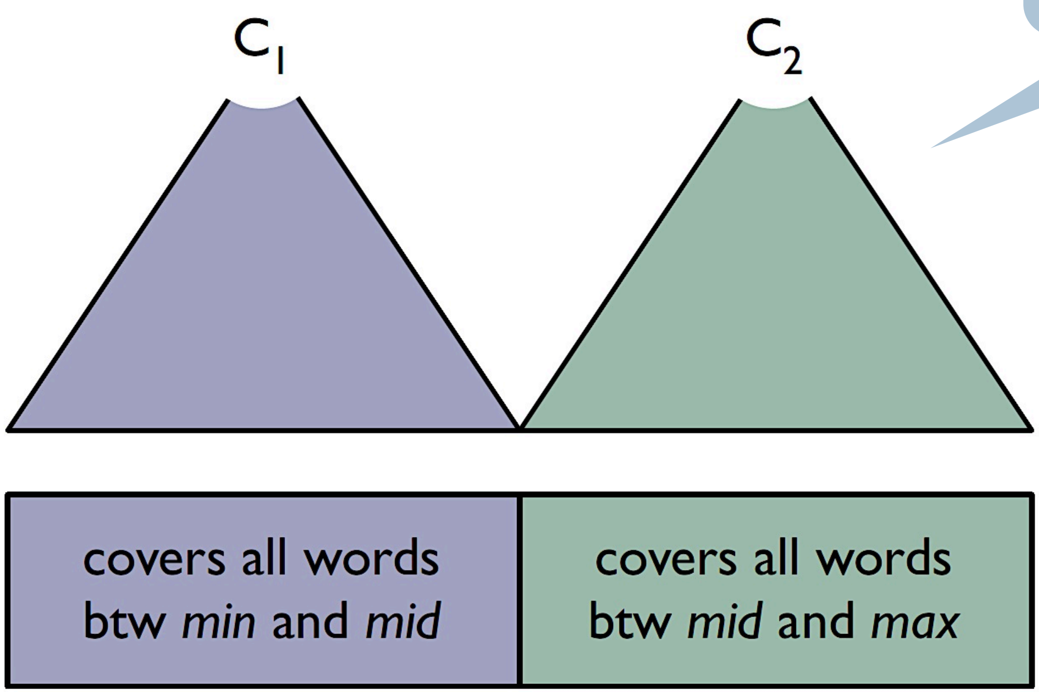
$C \to w_i$



Parsing longer spans

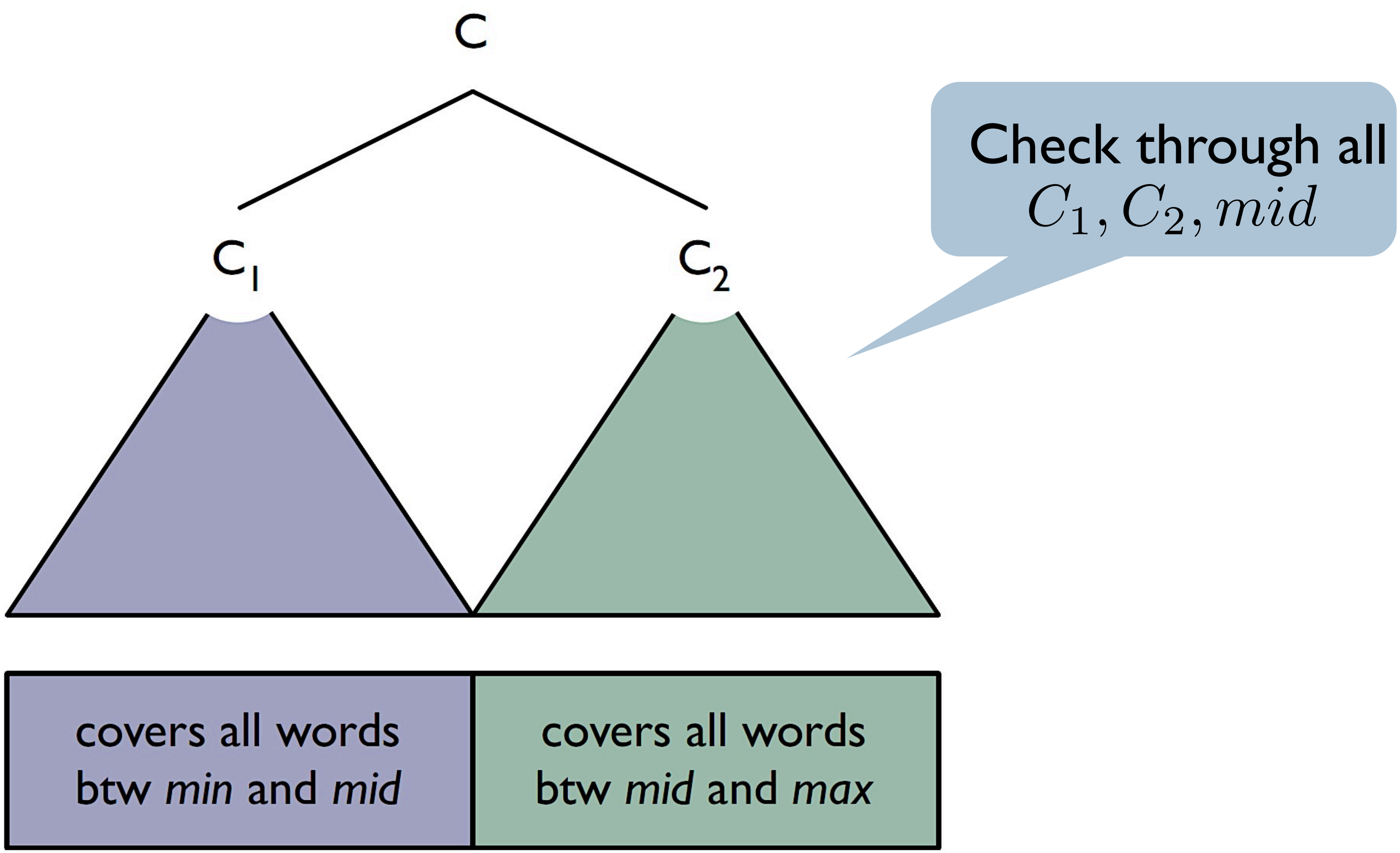
$$C \rightarrow C_1 \ \ C_2$$

Check through all C_1, C_2, mid

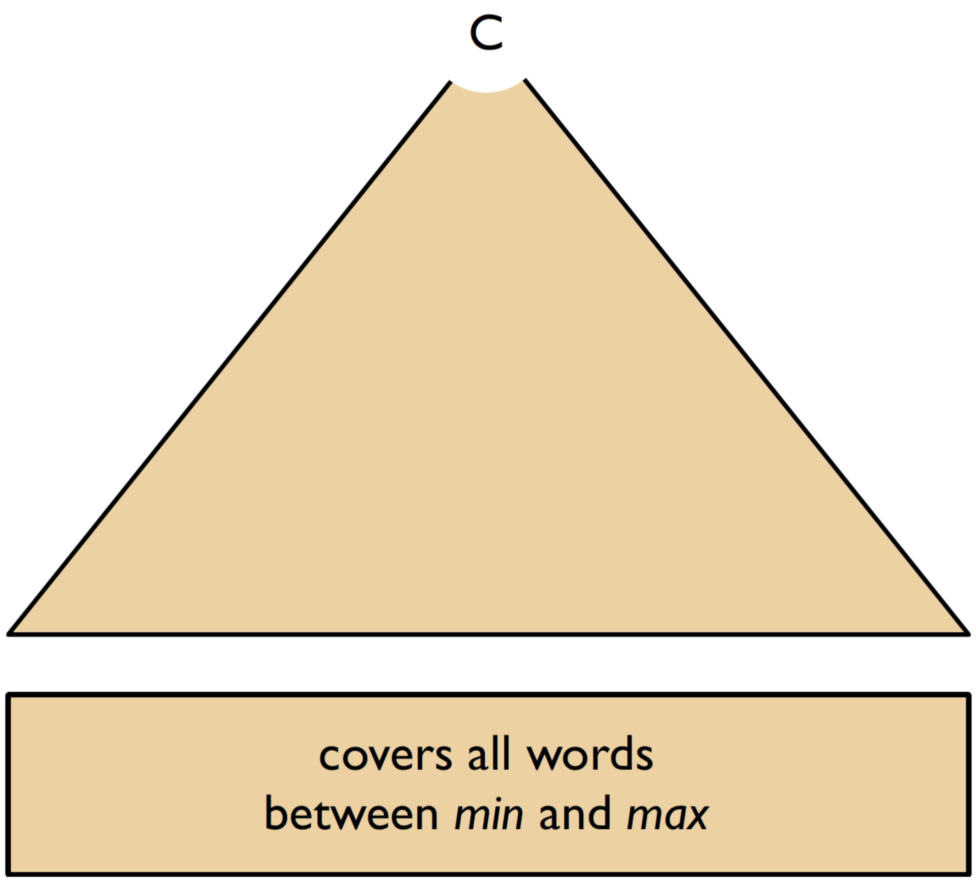


Parsing longer spans

$$C \rightarrow C_1 \;\; C_2$$



Parsing longer spans



Signatures

- Applications of rules is independent of inner structure of a parse tree
- We only need to know the corresponding span and the root label of the tree
  - Its signature $[min, max, C]$

Also known as an *edge*

CKY idea

- Compute for every span a set of admissible labels (may be empty for some spans)
  - Start from small trees (single words) and proceed to larger ones
- When done, check if *S* is among admissible labels for the whole sentence, if yes – the sentence belong to the language
  - That is if a tree with signature *[0, n, S]* exists
- Unary rules?

CKY in action

| lead | can | poison |

0 1 2 3

$$S \rightarrow NP\ VP$$

Inner rules

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

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

Preterminal rules

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

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

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

CKY in action

| lead | can | poison |

0 1 2 3

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

Chart (aka parsing triangle)

Inner rules

$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

$$NP \rightarrow N\ NP$$

Preterminal rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

$$N \rightarrow poison$$

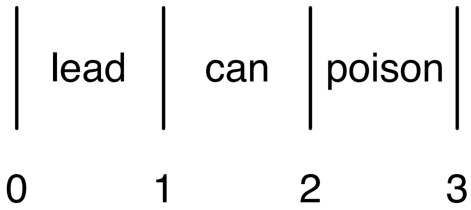
$$M \rightarrow can$$

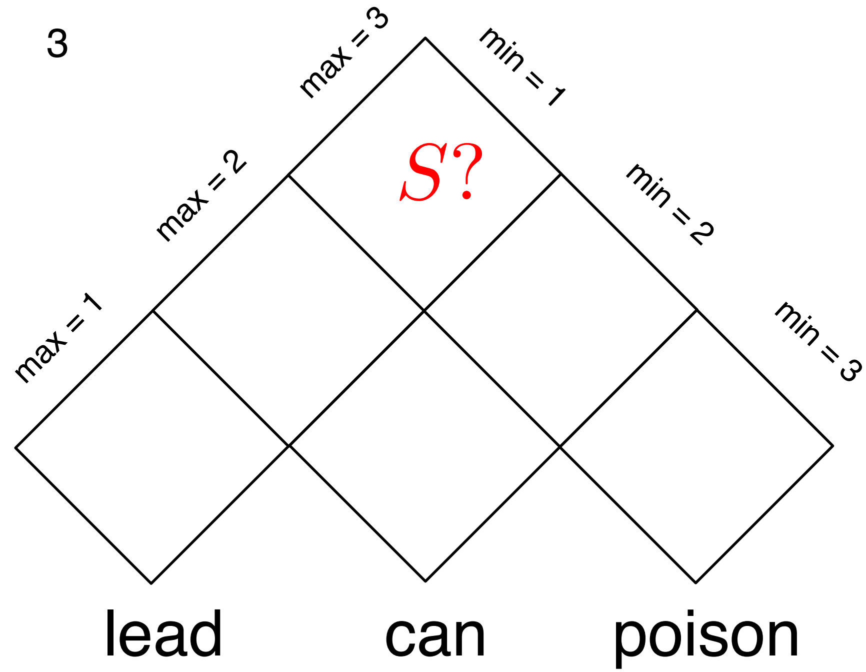
$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$

CKY in action





$S \rightarrow NP\ VP$

$VP \rightarrow M\ V$

$VP \rightarrow V$

$NP \rightarrow N$

$NP \rightarrow N\ NP$

Inner rules

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

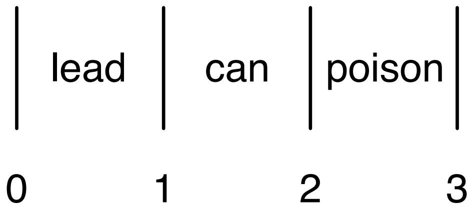
$M \rightarrow must$

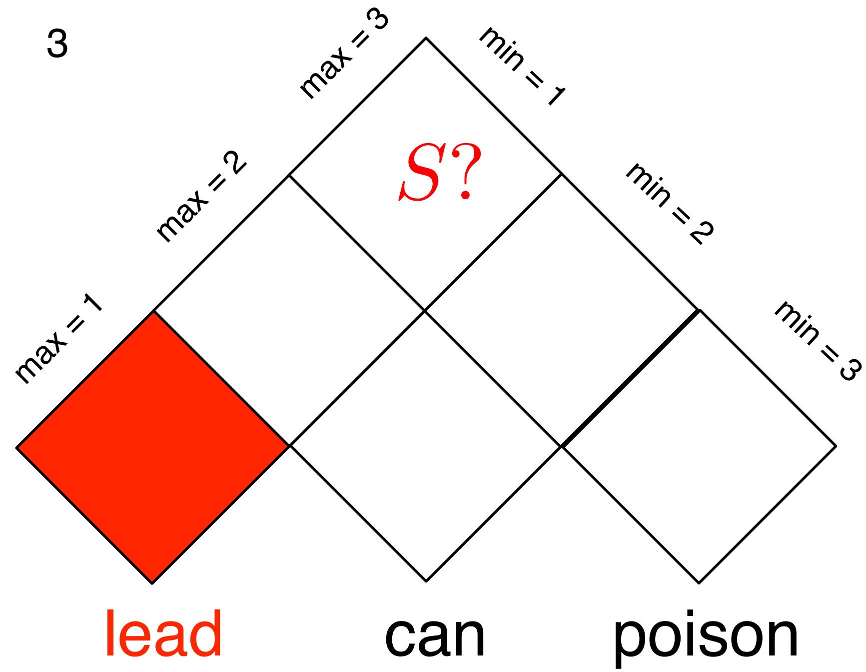
$V \rightarrow poison$

$V \rightarrow lead$

Preterminal rules

CKY in action





Inner rules

$$S \rightarrow NP \ \ VP$$

$$VP \rightarrow M \ \ V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

$$NP \rightarrow N \ NP$$

Preterminal rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

$$N \rightarrow poison$$

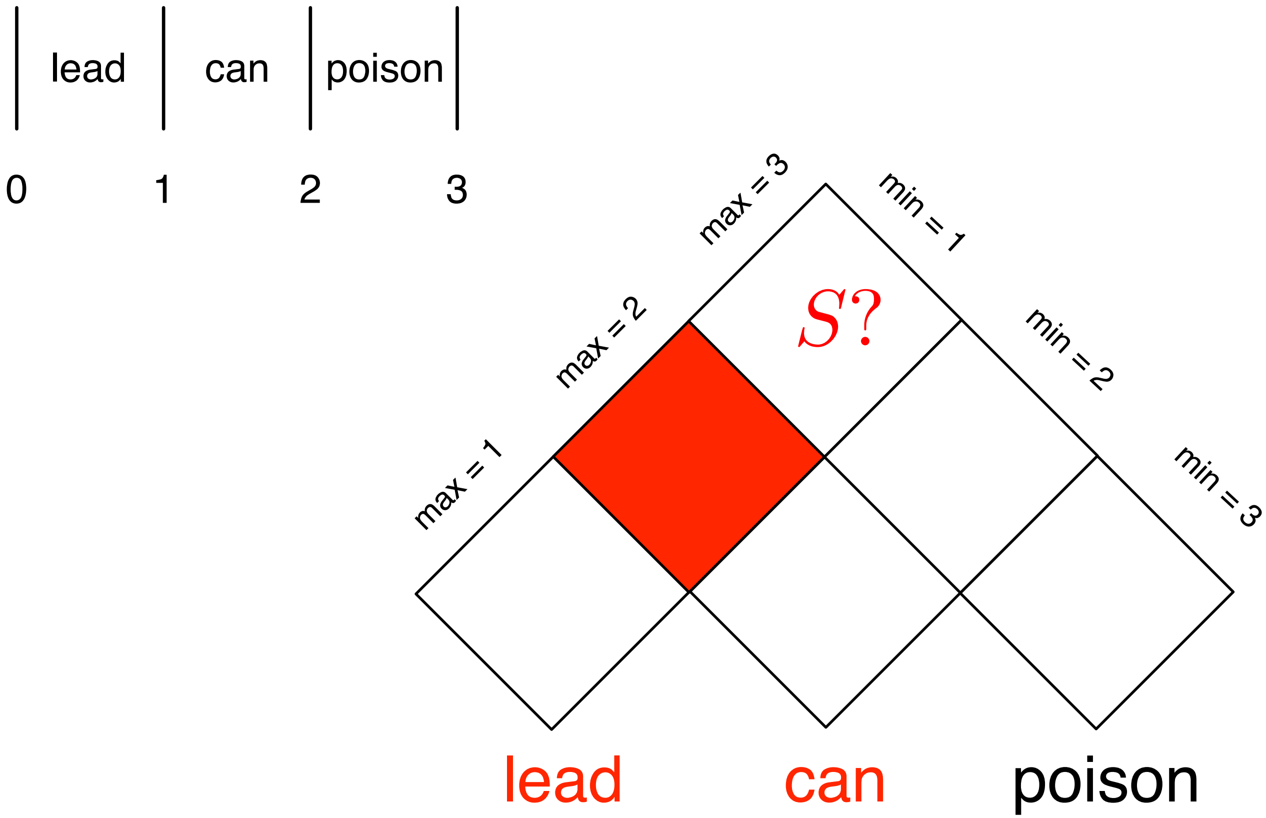
$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$

CKY in action



Inner rules

$S \to NP\ VP$

$VP \to M\ V$

$VP \to V$

$NP \to N$

$NP \to N\ NP$

Preterminal rules

$N \to can$

$N \to lead$

$N \to poison$

$M \to can$

$M \to must$

$V \to poison$

$V \to lead$

CKY in action

| lead | can | poison |

0 1 2 3

	max = 1	max = 2	max = 3
min = 0	1	4	6 $S?$
min = 1		2	5
min = 2			3

Inner rules

$$S \rightarrow NP\ VP$$

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

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

Preterminal rules

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

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

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

CKY in action

lead	can	poison

0 1 2 3

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

Inner rules

$$S \rightarrow NP \ VP$$

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

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

Preterminal rules

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

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

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

CKY in action

| lead | can | poison |

0 1 2 3

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

$$S \rightarrow NP\ VP$$

Inner rules

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

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

Preterminal rules

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

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

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

CKY in action

	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

Inner rules

$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

$$NP \rightarrow N\ NP$$

Preterminal rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

$$N \rightarrow poison$$

$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$

CKY in action

| lead | can | poison |

0 1 2 3

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

Inner rules

$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$

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

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

$$NP \rightarrow N\ NP$$

Preterminal rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

$$N \rightarrow poison$$

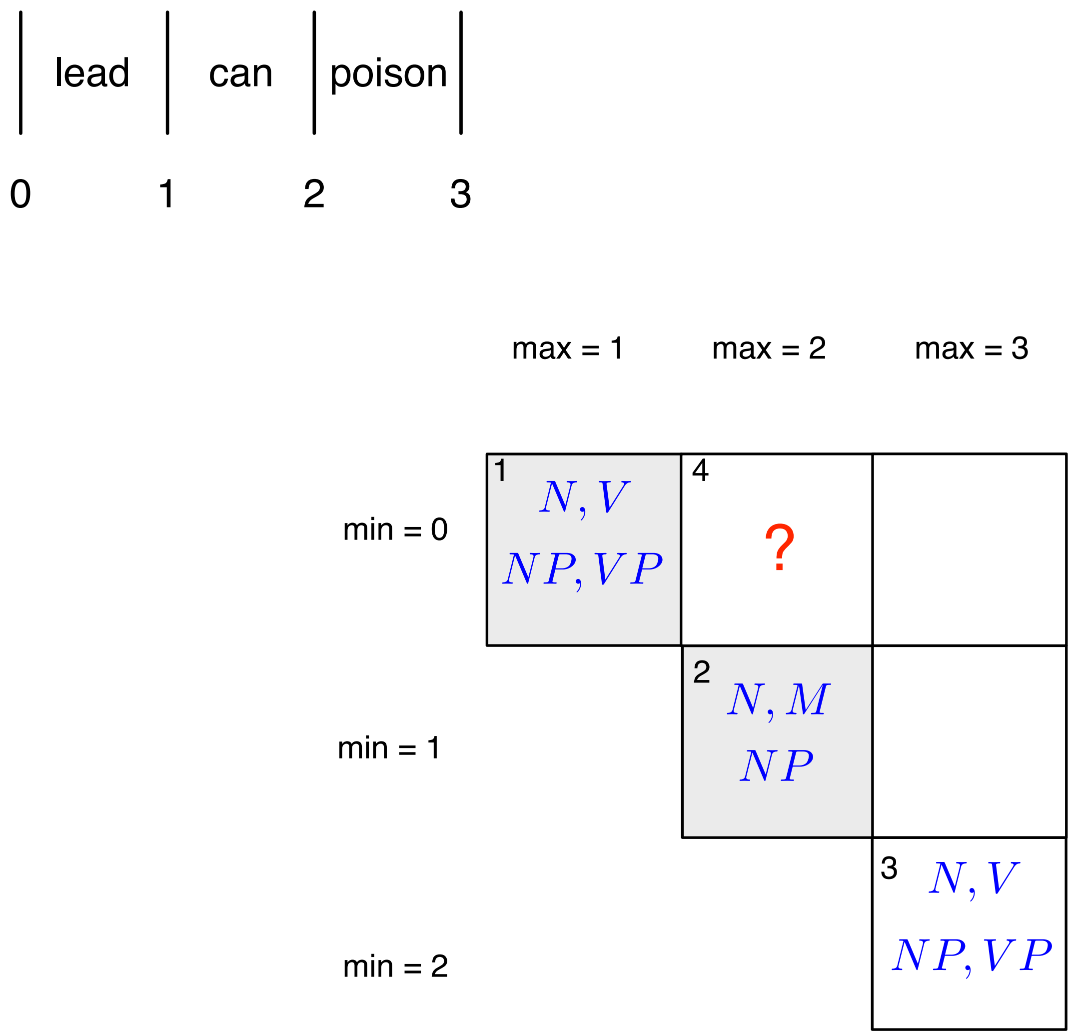
$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$

CKY in action



$$S \rightarrow NP \ VP$$

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

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

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

CKY in action

| lead | can | poison |

0 1 2 3

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

$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

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

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

CKY in action

| 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

Inner rules

$$S \rightarrow NP\ \ VP$$

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

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

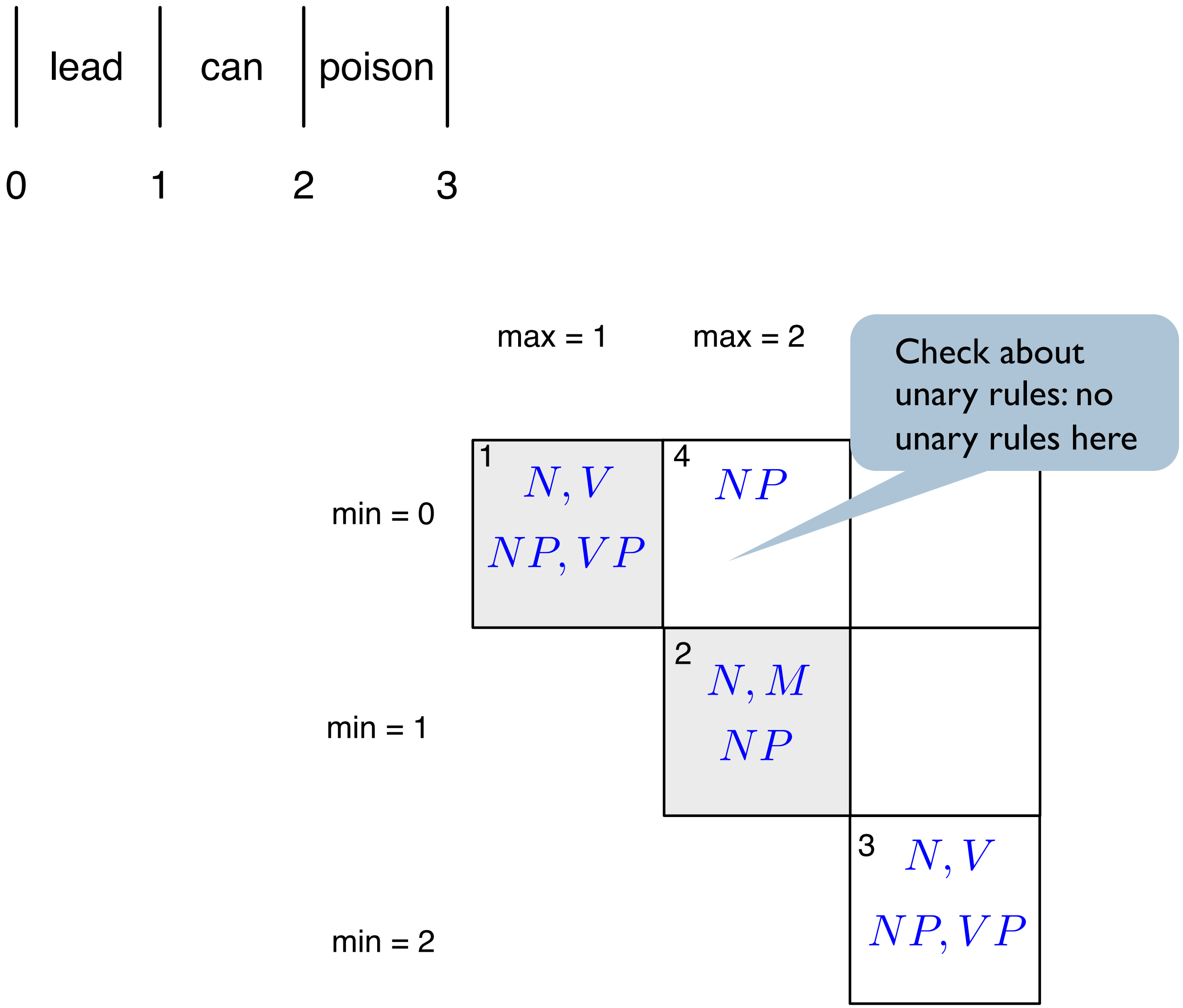
Preterminal rules

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

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

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

CKY in action



Inner rules

$$S \to NP\ VP$$

$$VP \to M\ V$$

$$VP \to V$$

$$NP \to N$$

$$NP \to N\ NP$$

Preterminal rules

$$N \to can$$

$$N \to lead$$

$$N \to poison$$

$$M \to can$$

$$M \to must$$

$$V \to poison$$

$$V \to lead$$

CKY in action

| 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

Inner rules

$$S \rightarrow NP\ VP$$

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

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

Preterminal rules

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

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

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

CKY in action

| 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

Inner rules

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

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

$$VP \rightarrow V$$

$$NP \rightarrow N$$

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

Preterminal rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

$$N \rightarrow poison$$

$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$

CKY in action

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

Check about unary rules: no unary rules here

Inner rules

$$S \rightarrow NP\ VP$$

$$VP \rightarrow M\ V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

$$NP \rightarrow N\ NP$$

Preterminal rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

$$N \rightarrow poison$$

$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$

CKY in action

| 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

Inner rules

$$S \rightarrow NP\ VP$$

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

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

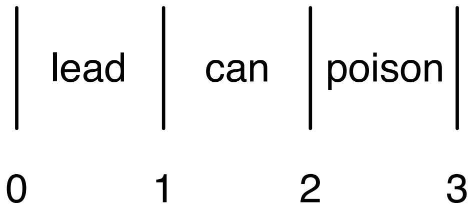
Preterminal rules

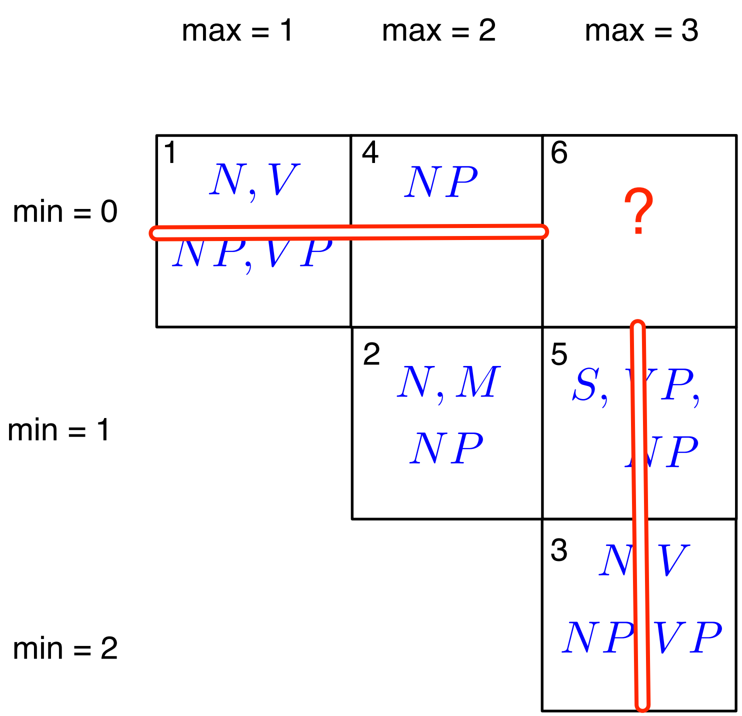
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$

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

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

CKY in action





Inner rules

$$S \rightarrow NP\ VP$$

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

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

Preterminal rules

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

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

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

CKY in action

| lead | can | poison |

0 1 2 3

$\mathrm{mid} = 1$

	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

Inner rules

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

$$VP \rightarrow M\ \ V$$

$$VP \rightarrow V$$

$$NP \rightarrow N$$

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

Preterminal rules

$$N \rightarrow can$$

$$N \rightarrow lead$$

$$N \rightarrow poison$$

$$M \rightarrow can$$

$$M \rightarrow must$$

$$V \rightarrow poison$$

$$V \rightarrow lead$$

CKY in action

| lead | can | poison |

0 1 2 3

$\mathrm{mid} = 2$

	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

$N \rightarrow can$

$N \rightarrow lead$

$N \rightarrow poison$

$M \rightarrow can$

$M \rightarrow must$

$V \rightarrow poison$

$V \rightarrow lead$

CKY in action

lead	can	poison

0 1 2 3

$$S \rightarrow NP \ VP$$

Inner rules

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

Preterminal rules

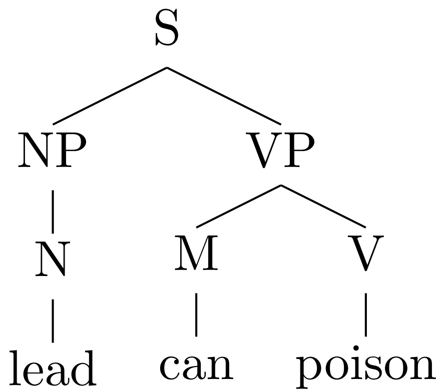
$$N \rightarrow can$$
$$N \rightarrow lead$$
$$N \rightarrow poison$$
$$M \rightarrow can$$
$$M \rightarrow must$$
$$V \rightarrow poison$$
$$V \rightarrow lead$$

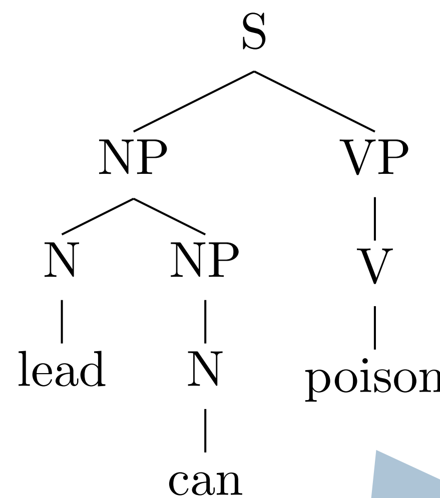
$\mathrm{mid} = 2$

	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
			3 N, V

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

Ambiguity





No subject-verb agreement, and *poison* used as an intransitive verb

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

CKY more formally

Here we assume that labels (C) are integer indices

- Chart can be represented by a Boolean array `chart[min][max][C]`
  - Relevant entries have $0 < min < max \leq n$
- `chart[min][max][C] = true` if the signature (min, max, C) is already added to the chart; `false` otherwise.

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

In the assignment code we use a class Chart but its access methods are similar

Implementation: preterminal rules

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

Implementation: binary rules

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      for each binary rule C -> C1 C2
        for each mid from min + 1 to max - 1
          if chart[min][mid][C1] and chart[mid][max][C2] then
            chart[min][max][C] = true
```

Implementation: unary rules

```
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
        if chart[min][max][C1] then
          chart[min][max][C] = true
```

new bounds!

But we forgot something!

Unary closure

- What if the grammar contained 2 rules:

  $$A \rightarrow B$$
  $$B \rightarrow C$$

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

  $$A \rightarrow B \rightarrow C$$

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

$$\begin{array}{l} A \rightarrow B \\ B \rightarrow C \end{array} \quad \Rightarrow \quad \begin{array}{ll} A \rightarrow B & A \rightarrow A \\ B \rightarrow C & B \rightarrow B \\ A \rightarrow C & C \rightarrow C \end{array}$$

Convenient for programming reasons in the PCFG case

Implementation: skeleton

```
// int n = number of words in the sequence

// int m = number of syntactic categories in the grammar

// int s = the (number of the) grammar's start symbol

boolean[][][] chart = new boolean[n + 1][n + 1][m]

// Recognize all parse trees built with with preterminal rules.

// Recognize all parse trees built with inner rules.

return chart[0][n][s]
```

Algorithm analysis

- Time complexity?

```
for each max from 2 to n
  for each min from max - 2 down to 0
    for each syntactic category C
      for each binary rule C -> C1 C2
        for each mid from min + 1 to max - 1
```

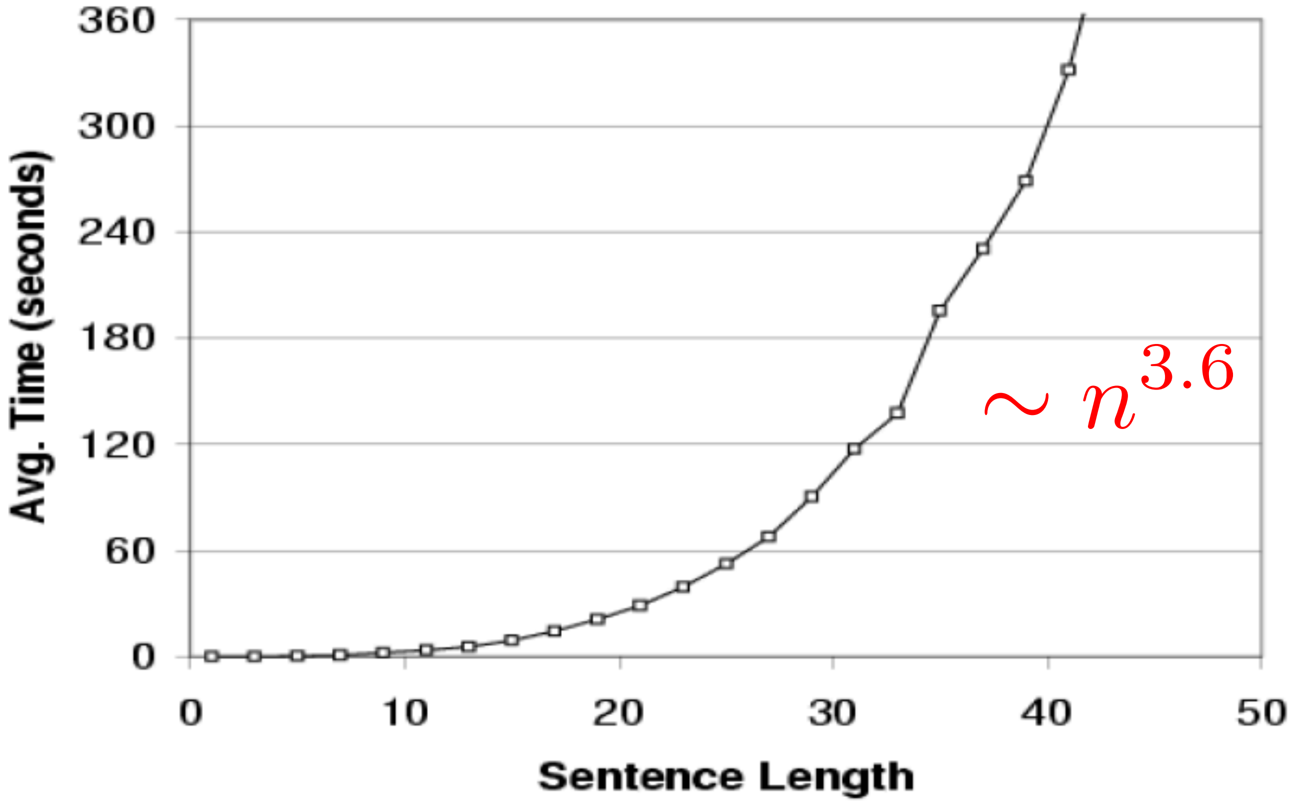
A few seconds for sentences under < 20 words for a non-optimized parser

- $\theta(n^3|R|)$, where $|R|$ is the number of rules in the grammar

- There exist algorithms with better asymptotical time complexity but the `constant' makes them slower in practice (in general)

Practical time complexity

- Time complexity? (for the PCFG version)



[Plot by Dan Klein]

Today

- Parsing algorithms for CFGs
  - Recap, Chomsky Normal Form (CNF)
  - A dynamic programming algorithm for parsing (CKY)
  - Extension of CKY to support unary inner rules
- Parsing for PCFGs
  - Extension of CKY for parsing with PCFGs
- Parser evaluation (if we have time)

Probabilistic parsing

- We discussed the recognition problem:
  - check if a sentence is parsable with a CFG
- Now we consider parsing with PCFGs
  - Recognition with PCFGs: what is the probability of the most probable parse tree?
  - Parsing with PCFGs: What is the most probable parse tree?

Distribution over trees

- Let us denote by $G(x)$ the set of derivations for the sentence x
- The probability distribution defines the scoring $P(T)$ over the trees $T \in G(x)$
- Finding the best parse for the sentence according to PCFG:

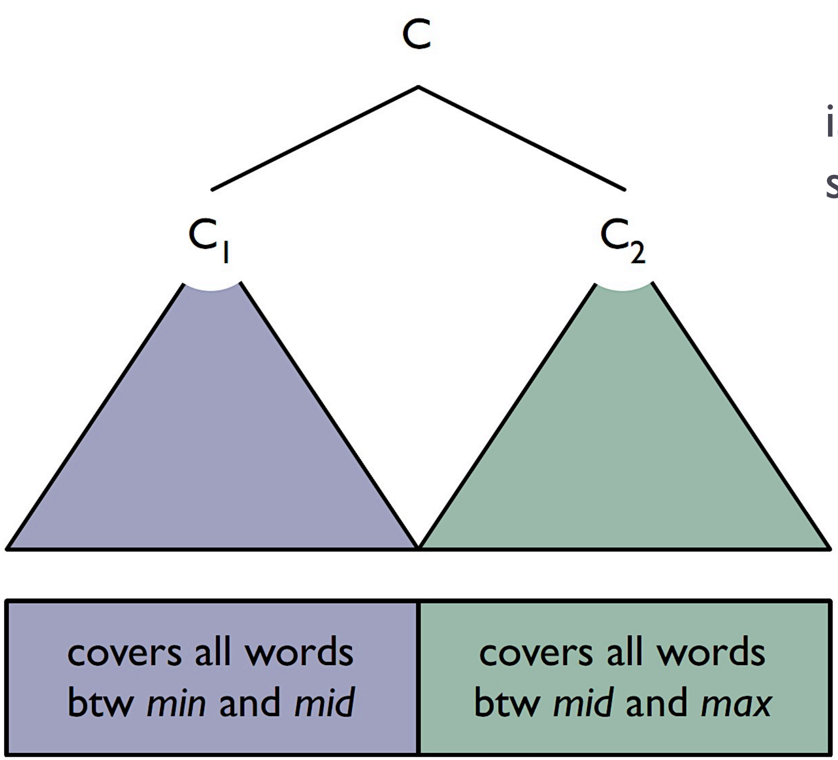
$$\arg\max_{T \in G(x)} P(T)$$

CKY with PCFGs

- Chart is represented by a double array `chart[min][max][C]`
  - 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

$$C \rightarrow C_1 \;\; C_2$$



For every C choose C_1, C_2 and mid such that

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

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
```

Unary (reflexive transitive) closure

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

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

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

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

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

Recovery of the tree

- For each signature we store backpointers to the elements from which it was built (e.g., rule and, for binary rules, midpoint)
  - start recovering from *[0, n, S]*

- 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 (approximate search)

- 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

Today

- Parsing algorithms for CFGs
  - Recap, Chomsky Normal Form (CNF)
  - A dynamic programming algorithm for parsing (CKY)
  - Extension of CKY to support unary inner rules
- Parsing for PCFGs
  - Extension of CKY for parsing with PCFGs
- Parser evaluation

Parser evaluation

Though has many drawbacks it is easier and allows to track state-of-the-art across many years

- 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

The most standard measure; we will focus on it

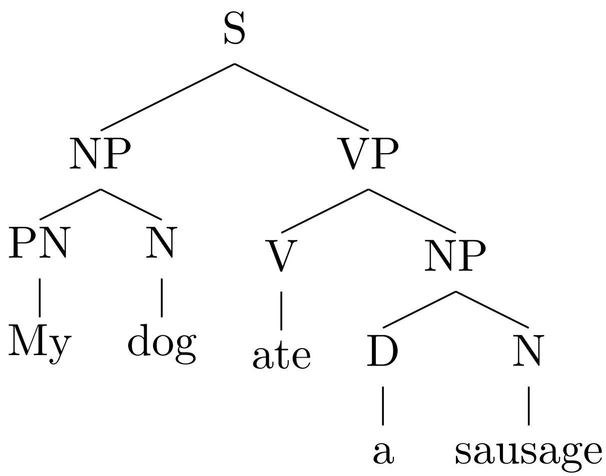
- Exact match: percentage of trees predicted correctly
- Bracket score: scores how well individual phrases (and their boundaries) are identified
- Crossing brackets: percentage of phrases boundaries crossing
- Dependency metrics: scores dependency structure corresponding to the constituent tree (percentage of correctly identified heads)

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

Bracketing notation



- The same tree as a bracketed sequence

```
(S
    (NP  (PN My) (N Dog) )
    (VP  (V ate)
           (NP (D a )  (N sausage) )
    )
)
```

Brackets scores

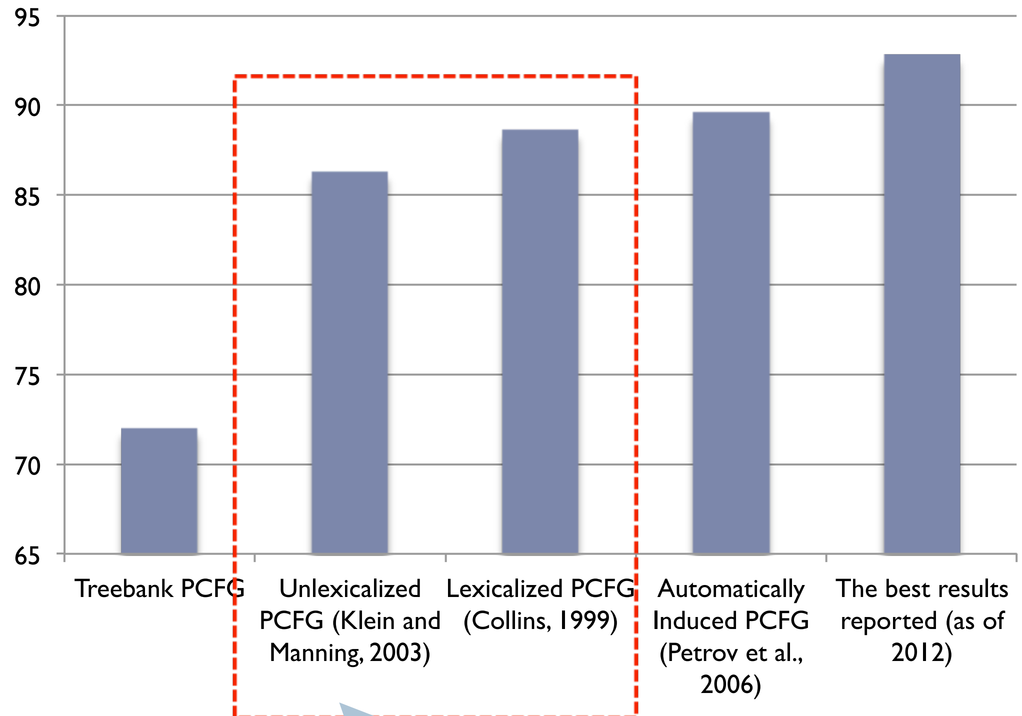
$$Pr = \frac{\text{number of brackets the parser and annotation agree on}}{\text{number of brackets predicted by the parser}}$$

$$Re = \frac{\text{number of brackets the parser and annotation agree on}}{\text{number of brackets in annotation}}$$

$$F1 = \frac{2 \times Pr \times Re}{Pr + Re}$$

Harmonic mean of precision and recall

Preview: F1 bracket score



We will introduce these models next time