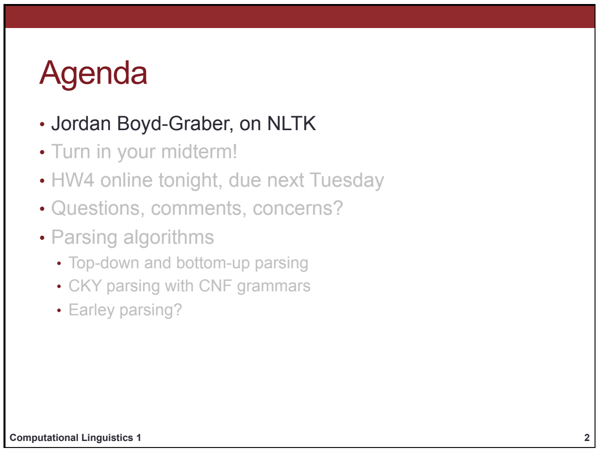
Computational Linguistics 1

CMSC/LING 723, LBSC 744

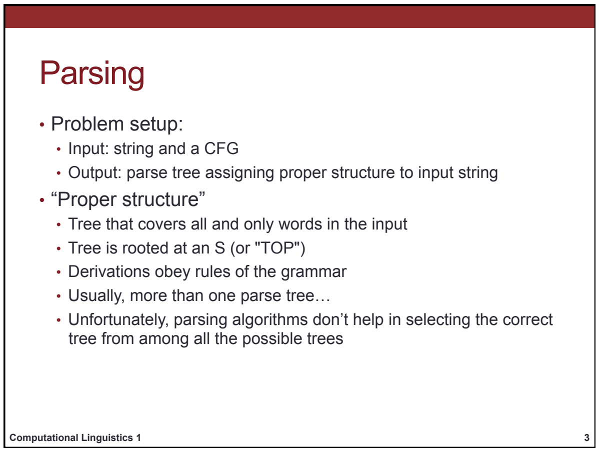
Kristy Hollingshead Seitz
Institute for Advanced Computer Studies
University of Maryland

Lecture 16: 25 October 2011



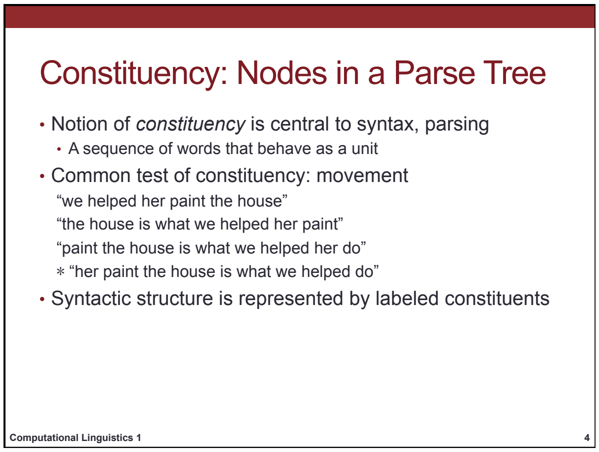
Agenda

- Jordan Boyd-Graber, on NLTK
- Turn in your midterm!
- HW4 online tonight, due next Tuesday
- Questions, comments, concerns?
- Parsing algorithms
  - Top-down and bottom-up parsing
  - CKY parsing with CNF grammars
  - Earley parsing?



Parsing

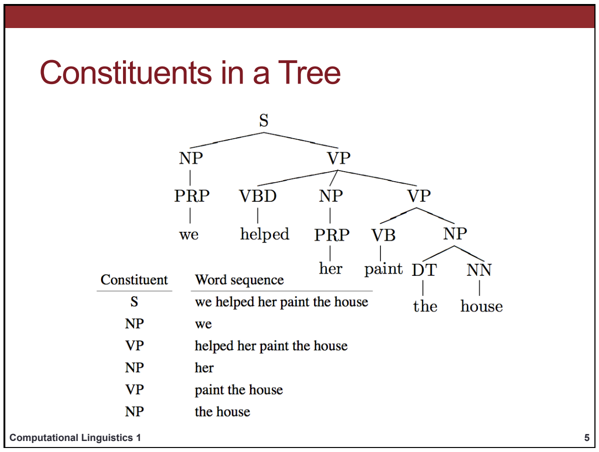
- Problem setup:
  - Input: string and a CFG
  - Output: parse tree assigning proper structure to input string
- "Proper structure"
  - Tree that covers all and only words in the input
  - Tree is rooted at an S (or "TOP")
  - Derivations obey rules of the grammar
  - Usually, more than one parse tree…
  - Unfortunately, parsing algorithms don't help in selecting the correct tree from among all the possible trees



Constituency: Nodes in a Parse Tree

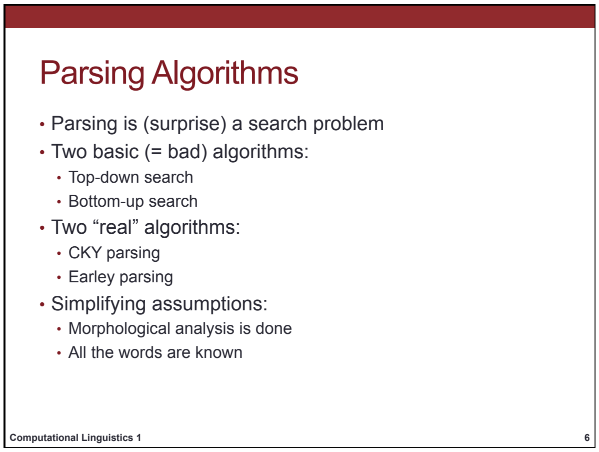
- Notion of *constituency* is central to syntax, parsing
  - A sequence of words that behave as a unit
- Common test of constituency: movement

  "we helped her paint the house"
  "the house is what we helped her paint"
  "paint the house is what we helped her do"
  * "her paint the house is what we helped do"
- Syntactic structure is represented by labeled constituents



Constituents in a Tree

Constituent	Word sequence
S	we helped her paint the house
NP	we
VP	helped her paint the house
NP	her
VP	paint the house
NP	the house



Parsing Algorithms

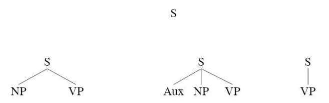
- Parsing is (surprise) a search problem
- Two basic (= bad) algorithms:
  - Top-down search
  - Bottom-up search
- Two "real" algorithms:
  - CKY parsing
  - Earley parsing
- Simplifying assumptions:
  - Morphological analysis is done
  - All the words are known

Top-Down Search

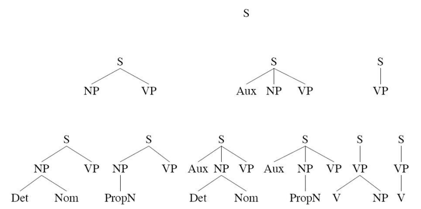
- Observation: trees must be rooted with an S node
- Parsing strategy:
  - Start at top with an S node
  - Apply rules to build out trees
  - Work down toward leaves

Top-Down Search

Top-Down Search



Top-Down Search



Problems with top-down

- Ambiguity
  - Can follow just one path
    - Requires backtracking, rebuilding structure
  - Might keep all around in parallel
    - Exponential in the length of the string
- Left-recursive grammars: NP → NP PP
  - Grammar transformation
- Probabilistic variants, with pruning, have been successful

Probabilistic Top-Down Parsing

- Keep a heap of candidate derivations, each of which follows a top-down search path
- Rank the analyses by some score, to work on the promising ones early
- Pop the topmost ranked analysis from the heap, and follow all top-down paths
- Push all new analyses onto the heap
- Collect successful parses and return the best one

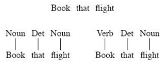
Bottom-Up Search

- Observation: trees must cover all input words
- Parsing strategy:
  - Start at the bottom with input words
  - Build structure based on grammar
  - Work up towards the root S

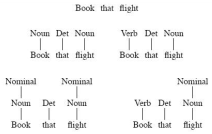
Bottom-Up Search

Book that flight

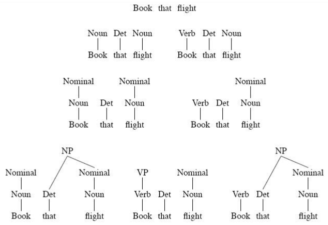
Bottom-Up Search



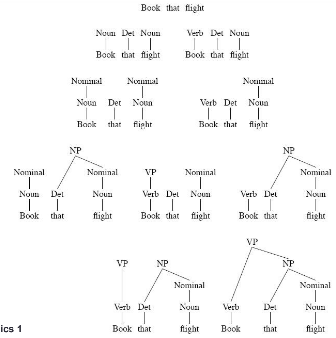
Bottom-Up Search



Bottom-Up Search



Bottom-Up Search



Top-Down vs Bottom-Up

- Top-down search
  - Only searches valid trees
  - But, considers trees that are not consistent with any of the words
  - Left-recursive grammars lead to non-termination NP → NP PP
  - Non-determinism
- Bottom-up search
  - Only builds trees consistent with the input
  - But, considers trees that don't lead anywhere
    - Without top-down guidance, can build a lot of structure that cannot be integrated with rest of string

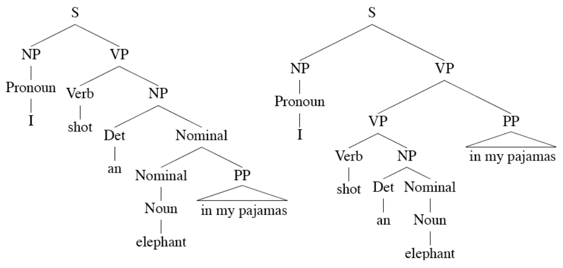
Parsing as Search

- Search involves controlling choices in the search space:
  - Which node to focus on in building structure
  - Which grammar rule to apply
- General strategy: backtracking
  - Make a choice, if it works out then fine
  - If not, then back up and make a different choice

Backtracking isn't enough!

- Ambiguity
- Shared sub-problems

Ambiguity



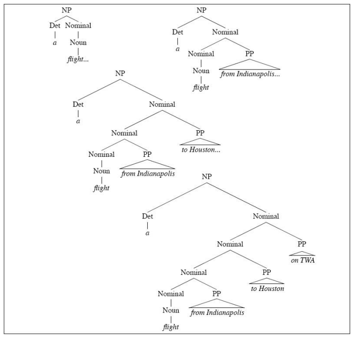
Shared Sub-Problems

- Observation: ambiguous parses still share sub-trees
- We don't want to redo work that's already been done
- Unfortunately, naïve backtracking leads to duplicate work

Shared Sub-Problems: Example

- Example: "A flight from Indianapolis to Houston on TWA"
- Assume a top-down parse making choices among the various nominal rules:
  - Nominal → Noun
  - Nominal → Nominal PP
- Statically choosing the rules in this order leads to lots of extra work...

Shared Sub-Problems: Example



Efficient Parsing

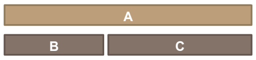
- Dynamic programming to the rescue!
- Intuition: store partial results in tables, thereby:
  - Avoiding repeated work on shared sub-problems
  - Efficiently storing ambiguous structures with shared sub-parts
- Two algorithms:
  - CKY: roughly, bottom-up
  - Earley: roughly, top-down

CYK Parsing

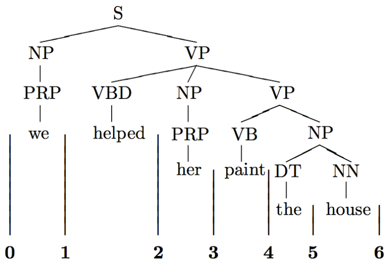
- Also referred to as "chart" parsing
- Related to Viterbi POS-tagging
- CKY parsing requires that the grammar consist of ε-free, binary rules = Chomsky Normal Form
- What if my treebank (or CFG) isn’t in CNF?

CKY Parsing: Intuition

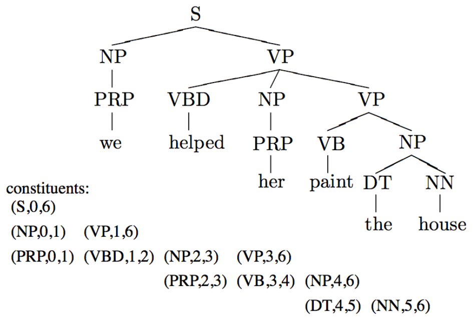
- Consider the rule D → w
  - Terminal (word) forms a constituent
  - Trivial to apply
- Consider the rule A → B C
  - If there is an A somewhere in the input then there must be a B followed by a C in the input
  - First, precisely define span [*i*, *j*]
  - If A spans from *i* to *j* in the input then there must be some *k* such that *i*<*k*<*j*
  - Easy to apply: we just need to try different values for *k*



Constituents and *Spans*

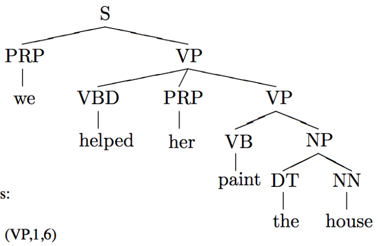


Constituents as Labeled Spans



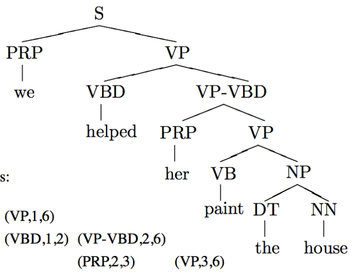
constituents:
(S,0,6)
(NP,0,1) (VP,1,6)
(PRP,0,1) (VBD,1,2) (NP,2,3) (VP,3,6)
(PRP,2,3) (VB,3,4) (NP,4,6)
(DT,4,5) (NN,5,6)

Labeled Spans, No Unaries



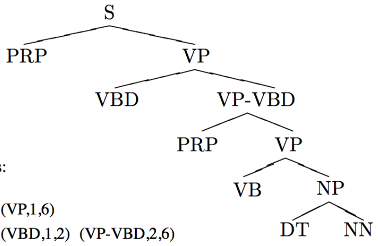
constituents:
(S,0,6)
(PRP,0,1) (VP,1,6)
(VBD,1,2) (PRP,2,3) (VP,3,6)
(VB,3,4) (NP,4,6)
(DT,4,5) (NN,5,6)

Labeled Spans in CNF



constituents:
(S,0,6)
(PRP,0,1) (VP,1,6)
(VBD,1,2) (VP-VBD,2,6)
(PRP,2,3) (VP,3,6)
(VB,3,4) (NP,4,6)
(DT,4,5) (NN,5,6)

Labeled Spans, No Lexical Items



constituents:
(S,0,6)
(PRP,0,1) (VP,1,6)
(VBD,1,2) (VP-VBD,2,6)
(PRP,2,3) (VP,3,6)
(VB,3,4) (NP,4,6)
(DT,4,5) (NN,5,6)

Chart Parsing, "Pseudocode"

- Initialize a chart with POS-tags (span length 1)
- For span length 2 to length of string
  - For all possible start and end points and all non-terminals
    1. Find the highest probability constituent with that label and span
    2. Keep a backtrace pointer
- Find the best analysis spanning the whole string
- Use backtrace pointers to output best parse

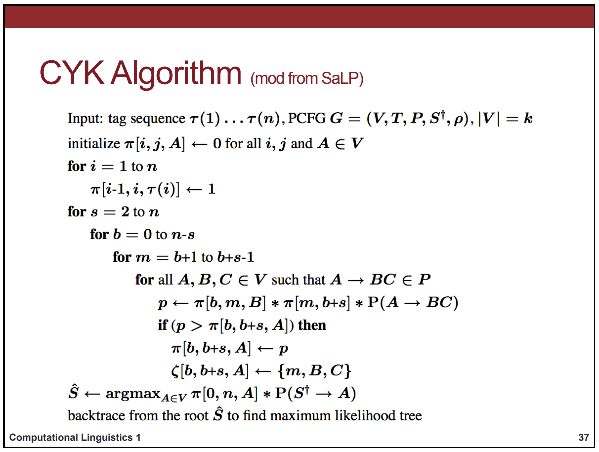
Labeled Spans, in CYK Chart

Span						
6	S					
5		VP				
4			VP-VBD			
3				VP		
2					NP	
1	PRP	VBD	PRP	VB	DT	NN

PCFG Notation (Refresher)

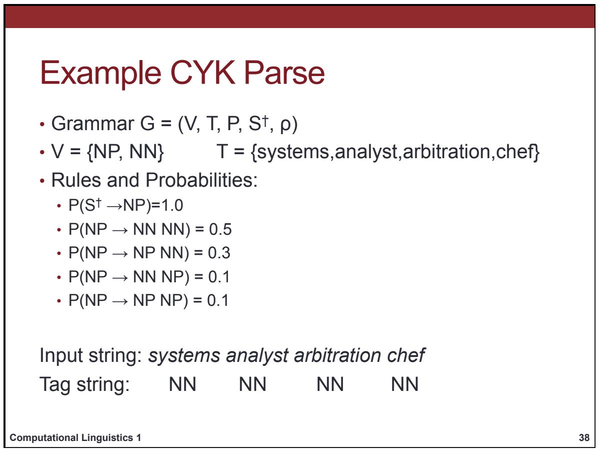
A PCFG G = (V,T,P,S†,ρ) consists of

- a set of non-terminal variables V
- a set of terminals T
- a set of rules P of the form A→α
- a special start symbol S† ∈ V
- a model ρ defining a conditional probability for every rule in P



CYK Algorithm (mod from SaLP)

```
Input: tag sequence $\tau(1) \ldots \tau(n)$, PCFG $G = (V, T, P, S^\dagger, \rho)$, $|V| = k$
initialize $\pi[i, j, A] \leftarrow 0$ for all $i, j$ and $A \in V$
for $i = 1$ to $n$
    $\pi[i\text{-}1, i, \tau(i)] \leftarrow 1$
for $s = 2$ to $n$
    for $b = 0$ to $n\text{-}s$
        for $m = b{+}1$ to $b{+}s\text{-}1$
            for all $A, B, C \in V$ such that $A \rightarrow BC \in P$
                $p \leftarrow \pi[b, m, B] * \pi[m, b{+}s] * \mathrm{P}(A \rightarrow BC)$
                if $(p > \pi[b, b{+}s, A])$ then
                    $\pi[b, b{+}s, A] \leftarrow p$
                    $\zeta[b, b{+}s, A] \leftarrow \{m, B, C\}$
$\hat{S} \leftarrow \mathrm{argmax}_{A \in V}\, \pi[0, n, A] * \mathrm{P}(S^\dagger \rightarrow A)$
backtrace from the root $\hat{S}$ to find maximum likelihood tree
```

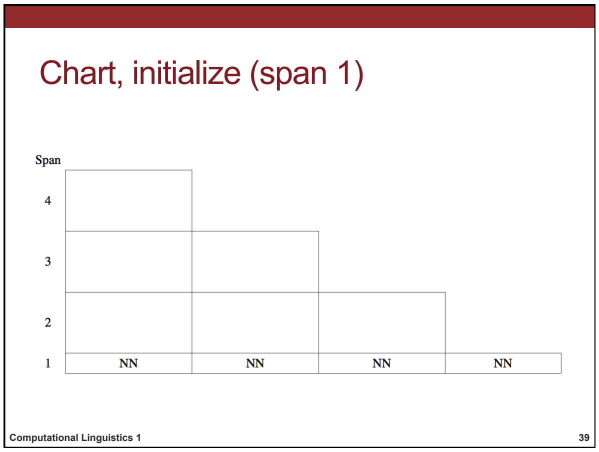


Example CYK Parse

- Grammar G = (V, T, P, S†, ρ)
- V = {NP, NN}   T = {systems,analyst,arbitration,chef}
- Rules and Probabilities:
  - P(S† →NP)=1.0
  - P(NP → NN NN) = 0.5
  - P(NP → NP NN) = 0.3
  - P(NP → NN NP) = 0.1
  - P(NP → NP NP) = 0.1

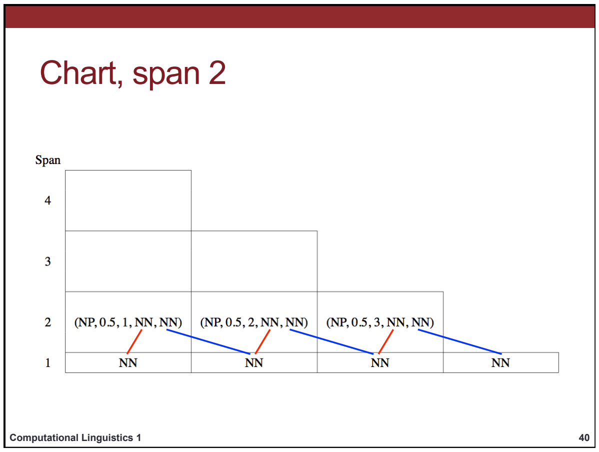
Input string: *systems analyst arbitration chef*

Tag string: NN NN NN NN



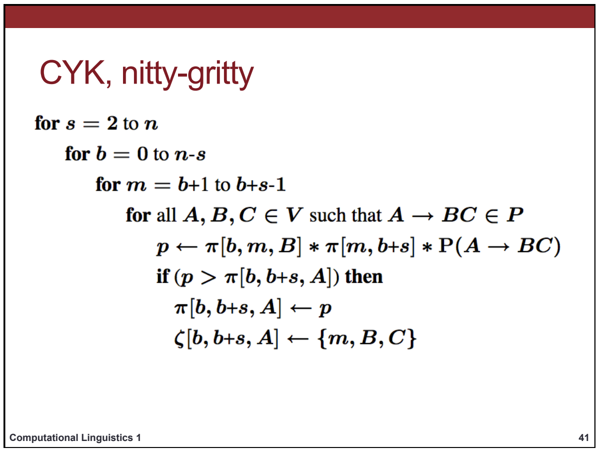
Chart, initialize (span 1)

Span				
4				
3				
2				
1	NN	NN	NN	NN



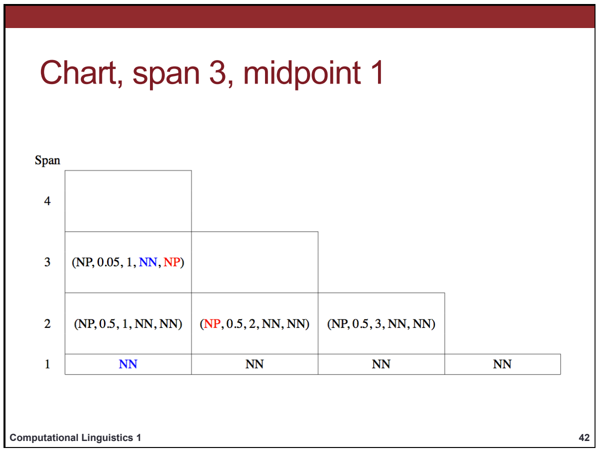
Chart, span 2

Span				
4				
3				
2	(NP, 0.5, 1, NN, NN)	(NP, 0.5, 2, NN, NN)	(NP, 0.5, 3, NN, NN)	
1	NN	NN	NN	NN



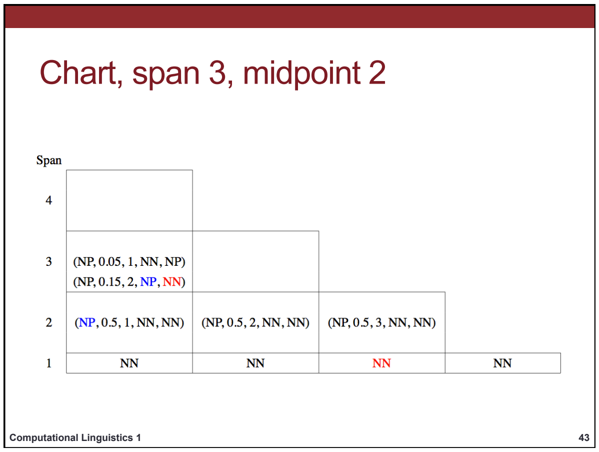
CYK, nitty-gritty

```
for $s = 2$ to $n$
    for $b = 0$ to $n\text{-}s$
        for $m = b{+}1$ to $b{+}s\text{-}1$
            for all $A, B, C \in V$ such that $A \rightarrow BC \in P$
                $p \leftarrow \pi[b, m, B] * \pi[m, b{+}s] * \mathrm{P}(A \rightarrow BC)$
                if $(p > \pi[b, b{+}s, A])$ then
                    $\pi[b, b{+}s, A] \leftarrow p$
                    $\zeta[b, b{+}s, A] \leftarrow \{m, B, C\}$
```



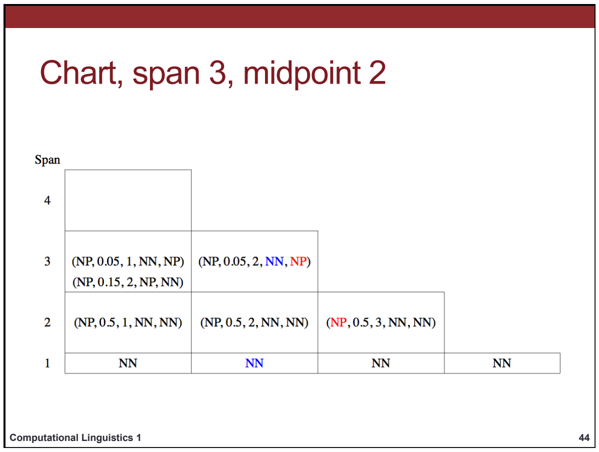
Chart, span 3, midpoint 1

Span				
4				
3	(NP, 0.05, 1, NN, NP)			
2	(NP, 0.5, 1, NN, NN)	(NP, 0.5, 2, NN, NN)	(NP, 0.5, 3, NN, NN)	
1	NN	NN	NN	NN



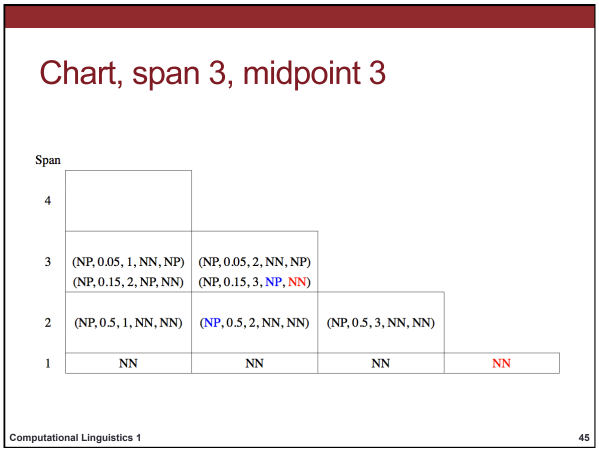
Chart, span 3, midpoint 2

Span				
4				
3	(NP, 0.05, 1, NN, NP) (NP, 0.15, 2, NP, NN)			
2	(NP, 0.5, 1, NN, NN)	(NP, 0.5, 2, NN, NN)	(NP, 0.5, 3, NN, NN)	
1	NN	NN	NN	NN



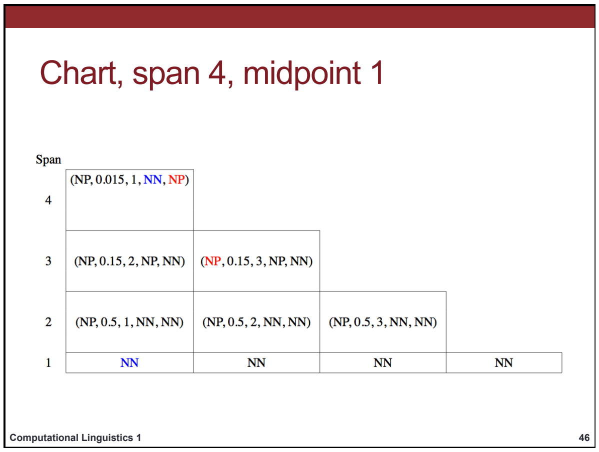
Chart, span 3, midpoint 2

Span				
4				
3	(NP, 0.05, 1, NN, NP) (NP, 0.15, 2, NP, NN)	(NP, 0.05, 2, NN, NP)		
2	(NP, 0.5, 1, NN, NN)	(NP, 0.5, 2, NN, NN)	(NP, 0.5, 3, NN, NN)	
1	NN	NN	NN	NN



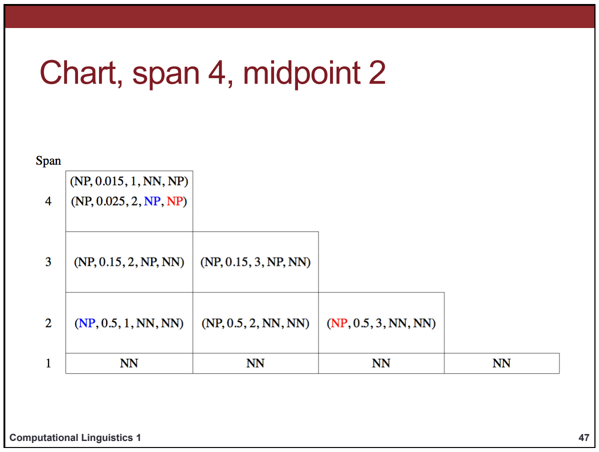
Chart, span 3, midpoint 3

Span				
4				
3	(NP, 0.05, 1, NN, NP) (NP, 0.15, 2, NP, NN)	(NP, 0.05, 2, NN, NP) (NP, 0.15, 3, NP, NN)		
2	(NP, 0.5, 1, NN, NN)	(NP, 0.5, 2, NN, NN)	(NP, 0.5, 3, NN, NN)	
1	NN	NN	NN	NN



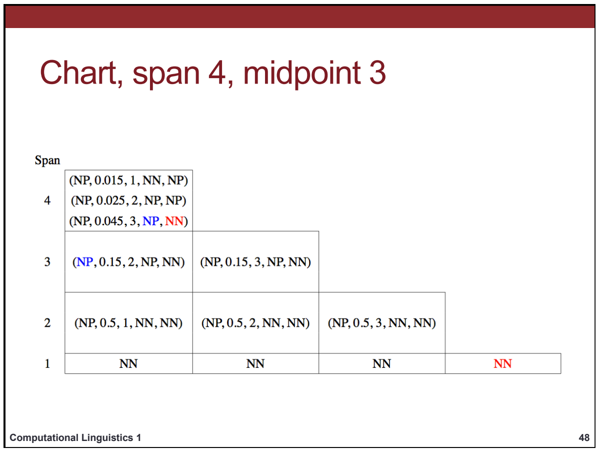
Chart, span 4, midpoint 1

Span				
4	(NP, 0.015, 1, NN, NP)			
3	(NP, 0.15, 2, NP, NN)	(NP, 0.15, 3, NP, NN)		
2	(NP, 0.5, 1, NN, NN)	(NP, 0.5, 2, NN, NN)	(NP, 0.5, 3, NN, NN)	
1	NN	NN	NN	NN



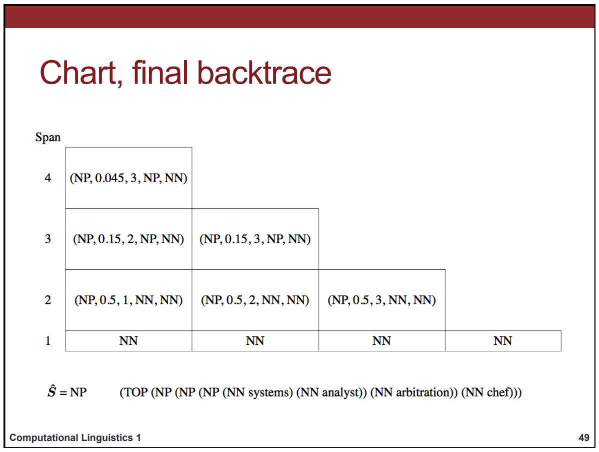
Chart, span 4, midpoint 2

Span				
4	(NP, 0.015, 1, NN, NP) (NP, 0.025, 2, NP, NP)			
3	(NP, 0.15, 2, NP, NN)	(NP, 0.15, 3, NP, NN)		
2	(NP, 0.5, 1, NN, NN)	(NP, 0.5, 2, NN, NN)	(NP, 0.5, 3, NN, NN)	
1	NN	NN	NN	NN



Chart, span 4, midpoint 3

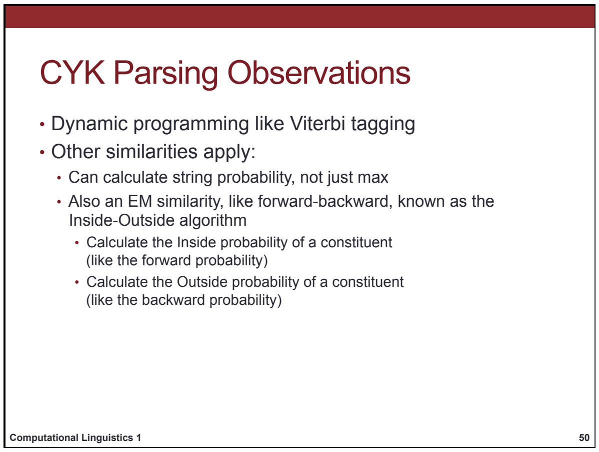
Span				
4	(NP, 0.015, 1, NN, NP) (NP, 0.025, 2, NP, NP) (NP, 0.045, 3, NP, NN)			
3	(NP, 0.15, 2, NP, NN)	(NP, 0.15, 3, NP, NN)		
2	(NP, 0.5, 1, NN, NN)	(NP, 0.5, 2, NN, NN)	(NP, 0.5, 3, NN, NN)	
1	NN	NN	NN	NN



Chart, final backtrace

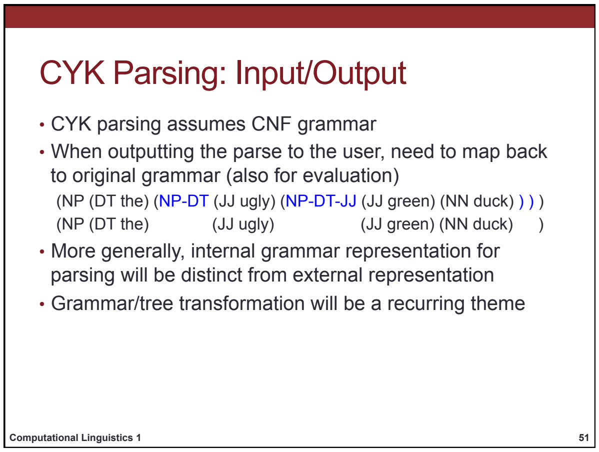
Span				
4	(NP, 0.045, 3, NP, NN)			
3	(NP, 0.15, 2, NP, NN)	(NP, 0.15, 3, NP, NN)		
2	(NP, 0.5, 1, NN, NN)	(NP, 0.5, 2, NN, NN)	(NP, 0.5, 3, NN, NN)	
1	NN	NN	NN	NN

$\hat{S}$ = NP (TOP (NP (NP (NP (NN systems) (NN analyst)) (NN arbitration)) (NN chef)))



CYK Parsing Observations

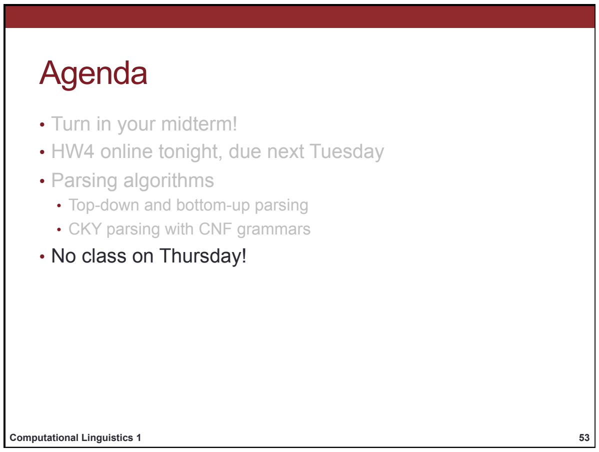
- Dynamic programming like Viterbi tagging
- Other similarities apply:
  - Can calculate string probability, not just max
  - Also an EM similarity, like forward-backward, known as the Inside-Outside algorithm
    - Calculate the Inside probability of a constituent (like the forward probability)
    - Calculate the Outside probability of a constituent (like the backward probability)



CYK Parsing: Input/Output

- CYK parsing assumes CNF grammar
- When outputting the parse to the user, need to map back to original grammar (also for evaluation)

  (NP (DT the) (NP-DT (JJ ugly) (NP-DT-JJ (JJ green) (NN duck))))
  (NP (DT the) (JJ ugly) (JJ green) (NN duck))
- More generally, internal grammar representation for parsing will be distinct from external representation
- Grammar/tree transformation will be a recurring theme



Agenda

- Turn in your midterm!
- HW4 online tonight, due next Tuesday
- Parsing algorithms
  - Top-down and bottom-up parsing
  - CKY parsing with CNF grammars
- No class on Thursday!