# ECE8813
# Statistical Natural Language Processing

## Lectures 24-25: Statistical Parsing

*Chin-Hui Lee*
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
chl@ece.gatech.edu

# Chunking and Grammar Induction

- Remember the IBM Story in mid-90's
- Chunking: recognizing higher level units of structure that allow us to compress our description of a sentence
- Grammar Induction: Explain the structure of chunks found over different sentences
- Parsing: can be considered as implementing chunking
  - http://en.wikipedia.org/wiki/Parsing
  - http://nlp.standford.edu/downloards/lex-parser.shtml

# Syntax and Parsing

- Why should we care?
  - Grammar checkers
  - Question answering
  - Information extraction
  - Machine translation
- Role of parsing in language analysis
  - For programming languages, everything is driven by parsing
  - For natural languages, many systems do things without parsing
    - Due to the lack of good parser.

# Parsing Goals

- The goal: develop grammars and parsers that are:
  - Accurate – produce good parses
  - Model optimal – find their models' actual best parses
  - Fast – seconds to parse long sentences
- Technology exists to get any two, but not all three
  - Exhaustive parsing – not fast
    - Chart Parsing [Earley 70]
  - Approximate parsing – not optimal
    - Beam parsing, [Collins 97, Charniak 01]
    - Best-First Parsing [Charniak et al. 98]
  - Always build right-branching structure – not accurate
- The problem involves both: learning and inference


*Center of Signal and Image Processing*
*Georgia Institute of Technology*

# Context-Free Grammars

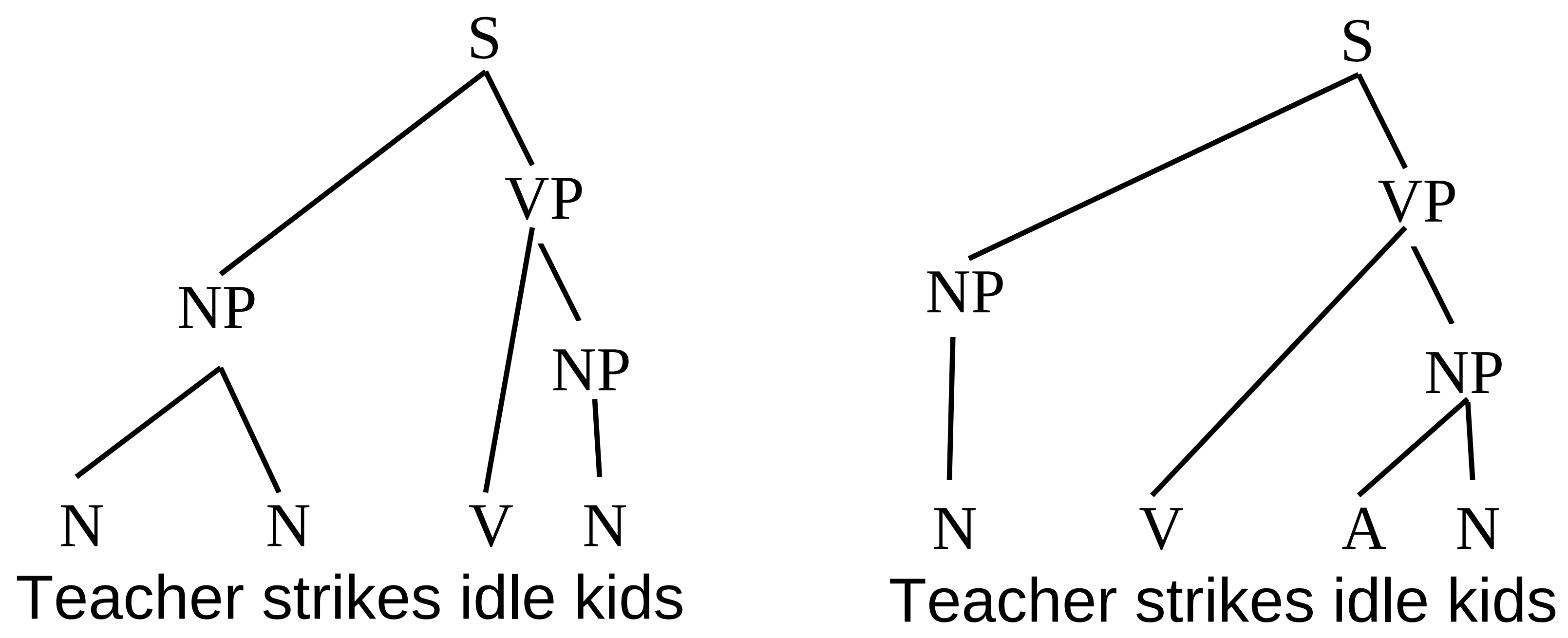
- A context free grammar consists of a set of phrase structure rules:
  - Examples
    - $S \rightarrow NP\ VP$
    - $N \rightarrow dog$
  - One left hand side symbol (non-terminal)
  - A sequence of right hand side symbols (terminals or non-terminals)
  - "Context-Free" means that the LHS symbol of a rule can be rewritten as the sequence of RHS symbols in any context

# Context Free Grammars and NLP

- Definitely not a good match!
  - Agreements
    - Fifi is/*are sleeping
  - Movements/empty categories
    - Who do you think Gary voted for?
  - Conjunctions
    - Kim and Dale/*yesterday
- However, almost all NL parsers has a CFG parser as the core

# Parsing

- Parsing is the process of taking a string and a grammar and returning parse tree(s) for that string



# Sentence-Types

- Declaratives: A plane left
  - $S \rightarrow NP\ VP$
- Imperatives: Leave!
  - $S \rightarrow VP$
- Yes-No Questions: Did the plane leave?
  - $S \rightarrow Aux\ NP\ VP$
- WH Questions: When did the plane leave?
  - $S \rightarrow WH\ Aux\ NP\ VP$

# Recursion

- We'll have to deal with rules such as the following where the non-terminal on the left also appears somewhere on the right (directly)
  - NP $\rightarrow$ NP PP [[The flight] [to Boston]]
  - VP $\rightarrow$ VP PP [[departed Miami] [at noon]]
- An example from ATIS
  - Flights from Denver
  - Flights from Denver to Miami
  - Flights from Denver to Miami in February
  - Flights from Denver to Miami in February on a Friday
  - Flights from Denver to Miami in February on a Friday under $300
  - Flights from Denver to Miami in February on a Friday under $300 with lunch

# Recursion

- Of course, this is what makes syntax interesting
  - [[Flights] [from Denver]]
  - [[[Flights] [from Denver]] [to Miami]]
  - [[[[Flights] [from Denver]] [to Miami]] [in February]]
  - [[[[[Flights] [from Denver]] [to Miami]] [in February]] [on a Friday]]
  - Etc.

# The Key Point

- VP $\rightarrow$ V NP
  - Only care that the thing after the verb is an NP
  - Doesn't have to know about the internal affairs of the NP
    - Flights from Denver
    - Flights from Denver to Miami
    - Flights from Denver to Miami in February
    - Flights from Denver to Miami in February on a Friday
    - Flights from Denver to Miami in February on a Friday under $300
    - Flights from Denver to Miami in February on a Friday under $300 with lunch

# CFG Parsing

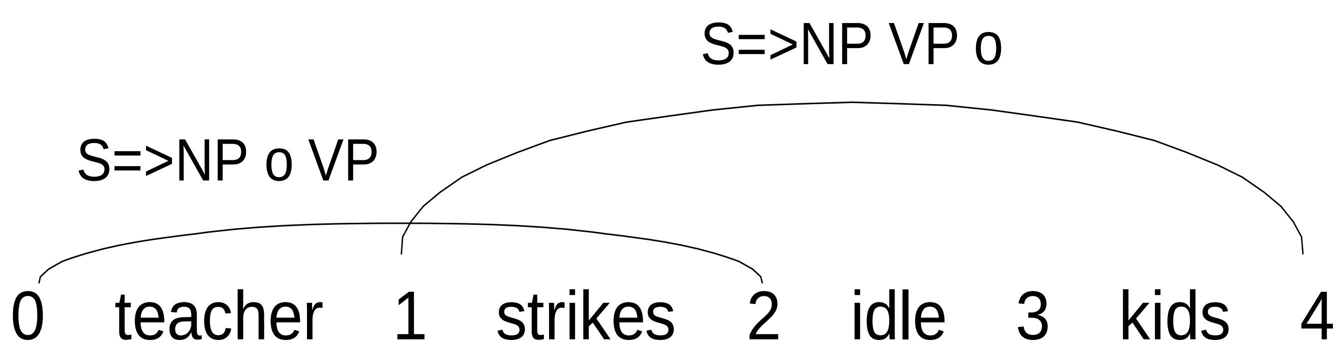
- Top down
  - Start from S, gradually expand rules to cover all the words
  - Usually involve search
- Bottom up
  - Start from words, gradually build up larger structures up to S
  - Usually involve dynamic programming

# Chart Parsing: Key Ideas

- Dynamic programming
  - Try everything, but never try the same thing more than once
- Ambiguity packing
  - Example: the NP "the book on the table by Chomsky", has two possible structures. However, if one of them can appear in a context, the other one can too
  - Stops the unnecessary propagation of ambiguities

# What is a Chart?

- A chart is a graph
  - Nodes represent word boundaries
  - There are two kinds of arcs
    - Active arcs: partially built phrases
    - Complete arcs: fully built phrases
  - Arcs are labeled with dot rules



# Example Arcs

- Arc: [0, 1] N => teacher •
  - [0, 1] is a noun
- Arc: [0, 1] S => NP • VP
  - We are trying to find a S, we've found the NP at [0,1]. We'll be looking for a VP from position 1
- Arc: [2, 4] S => NP • VP
  - We are trying to find a S, we've found the NP at [2,4]. We'll be looking for a VP from position 4
- Arc: [1, 4] VP => V NP •
  - We've found a VP at [1,4] that consists of a V and a NP
- Arc: [1, 4] VP => VP • PP
  - We are trying to find a VP, we've found the component VP at [1,4]. We'll be looking for a PP from position 4
- Arc: [0, 4] S => NP VP •
  - We've found a S at [0,4] that consists of a NP and a VP

# Chart Parsing: Initialization

- A chart has an agenda which keeps the complete arcs to be added to the chart
- The agenda is initialized with results of lexical look up
  - 0 teacher 1 strikes 2 idle 3 kids 4
    - [0, 1] N => teacher •
    - [1, 2] N => strikes •
    - [1, 2] V => strikes •
    - [2, 3] V => idle •
    - [2, 3] Adj => idle •
    - [3, 4] N => kids •

# Chart Parsing: Algorithm

```
while (!agenda.empty()) {
  arc = agenda.getFront();
  creatArcs(arc->lhs(), rules);
  foreach activeArc before arc {
    applyFundamentalRule(activeArc, arc);
  }
}
```

# Chart Parsing: Fundamental Rule

- Given
  - an active arc: *[a, b]* $X \to \ldots \bullet Y \ldots$; and
  - a complete arc: *[b, c]* $Y \to \ldots\ldots \bullet$

  create a new arc:
  - *[a, c]* $X \to \ldots Y \bullet \ldots$
- The new arc can be
  - complete (if nothing follows *Y* in $X \to \ldots \bullet Y \ldots$), or
  - active (otherwise)

# Chart Parsing Example

```
(def-cfg S
  (S  => NP VP)
  (N1 => Adj N1)
  (N1 => N)
  (N1 => N N)
  (NP => N1)
  (NP => Det N1)
  (N1 => N1 PP)
  (NP => Pron)
  (NP => Name)
  (VP => V)
  (VP => V NP)
  (VP => VP PP)
  (PP => P NP)
  )
```

```
(def-lexicon
  (teacher N)
  (strikes N V)
  (idle V Adj)
  (kids N)
  (she Pron)
  (him Pron)
  (in P)
  (the Det)
  (boy N)
  (park V N)
  (found V)
)
```

Input: teacher strikes idle kids

# Application of Derivation Rules

Arc: [0, 1] N => teacher •
Arc: [0, 1] N1 => N •
Arc: [0, 1] N1 => N • N
Arc: [0, 1] NP => N1 •
Arc: [0, 1] N1 => N1 • PP
Arc: [0, 1] S => NP • VP
Arc: [1, 2] N => strikes •
Arc: [1, 2] V => strikes •
Arc: [1, 2] N1 => N •
Arc: [1, 2] N1 => N • N
Arc: [0, 2] N1 => N N o
Arc: [1, 2] VP => V •
Arc: [1, 2] VP => V • NP
Arc: [1, 2] NP => N1 •
Arc: [1, 2] N1 => N1 • PP
Arc: [0, 2] NP => N1 •

Arc: [0, 2] N1 => N1 • PP
Arc: [1, 2] VP => VP • PP
Arc: [1, 2] S => NP • VP
Arc: [0, 2] S => NP • VP
Arc: [2, 3] V => idle •
Arc: [2, 3] Adj => idle •
Arc: [2, 3] VP => V •
Arc: [2, 3] VP => V • NP
Arc: [2, 3] N1 => Adj • N1
Arc: [2, 3] VP => VP • PP
Arc: [1, 3] S => NP VP •
Arc: [0, 3] S => NP VP •

Arc: [3, 4] N => kids •
Arc: [3, 4] N1 => N •
Arc: [3, 4] N1 => N • N
Arc: [3, 4] NP => N1 •
Arc: [3, 4] N1 => N1 • PP
Arc: [2, 4] N1 => Adj N1 •
Arc: [3, 4] S => NP • VP
Arc: [2, 4] VP => V NP •
Arc: [2, 4] NP => N1 •
Arc: [2, 4] N1 => N1 • PP
Arc: [2, 4] VP => VP • PP
Arc: [1, 4] S => NP VP •
Arc: [0, 4] S => NP VP •
Arc: [2, 4] S => NP • VP
Arc: [1, 4] VP => V NP •
Arc: [1, 4] VP => VP • PP
Arc: [0, 4] S => NP VP •

# Computational Complexity

$O(N^3G)$

- $N$ is the number of words in the input sentence
- $G$ is the total length of rules (measured by the number of symbols)
- It could be $O(N^3G^2)$ if grammar rules are not carefully organized (e.g., simply as a list)


Center of Signal and Image Processing
Georgia Institute of Technology

# Top-Down vs. Bottom-Up

- Top-down
  - Only searches for trees that can be answers
  - But suggests trees that are not consistent with the words
- Bottom-up
  - Only forms trees consistent with the words
  - Suggest trees that make no sense globally

# Computing String Probability

- a_dog saw a_cat with a_telescope

  1 2 3 4 5

| from\to | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | NP .21<br>N .3 | | S .441 | | S .00966 |
| 2 | | V 1 | VP .21 | | VP .046 |
| 3 | | | NP .35<br>N .5 | | NP .03 |
| 4 | | | | PREP 1 | PP .2 |
| 5 | | | | | N .2 |

- Create table *N* x *N* (*N* = length): cells might have more "lines"
- Initialize on diagonal, using $S \rightarrow a$ rules
- Recursively compute along diagonal towards upper right corner



*Center of Signal and Image Processing*
*Georgia Institute of Technology*
CSIP

# Language Model vs. Parsing Model

- Language model:
  - interested in string probability:

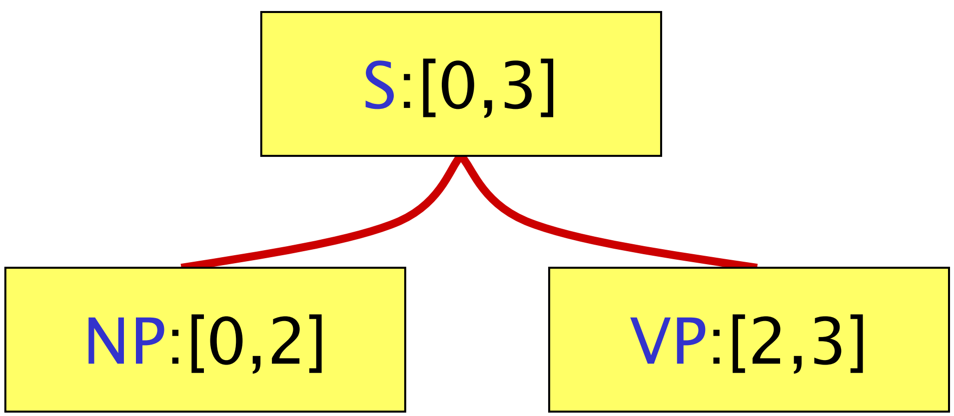
    $P(\mathrm{W})$ = probability definition using a formula such as

    $= \Pi_{\mathrm{i=1..n}}\, p(w_i|w_{i-2},w_{i-1})$ trigram language model

    $= \Sigma_{s \in S}\, p(W,s) = \Sigma_{s \in S}\, \Pi_{r \in s} r$ ; $r$ ~ rule used in parse tree
- Parsing model
  - conditional probability of tree given string: $P(\mathrm{s}|\mathrm{W}) = P(\mathrm{W,s}) / P(\mathrm{W}) = P(\mathrm{s}) / P(\mathrm{W})$ !! $P(\mathrm{W,s}) = P(\mathrm{s})$ !!
  - for argmax, just use $P(\mathrm{s})$ ($P(\mathrm{W})$ is constant)

# Parsing Complexity

- Time complexity of (general) CFG parsing is dominated by the number of traversals done
- Traversals represent the combination of two adjacent parse items into a larger one:




$= O(G^2 N^3)$

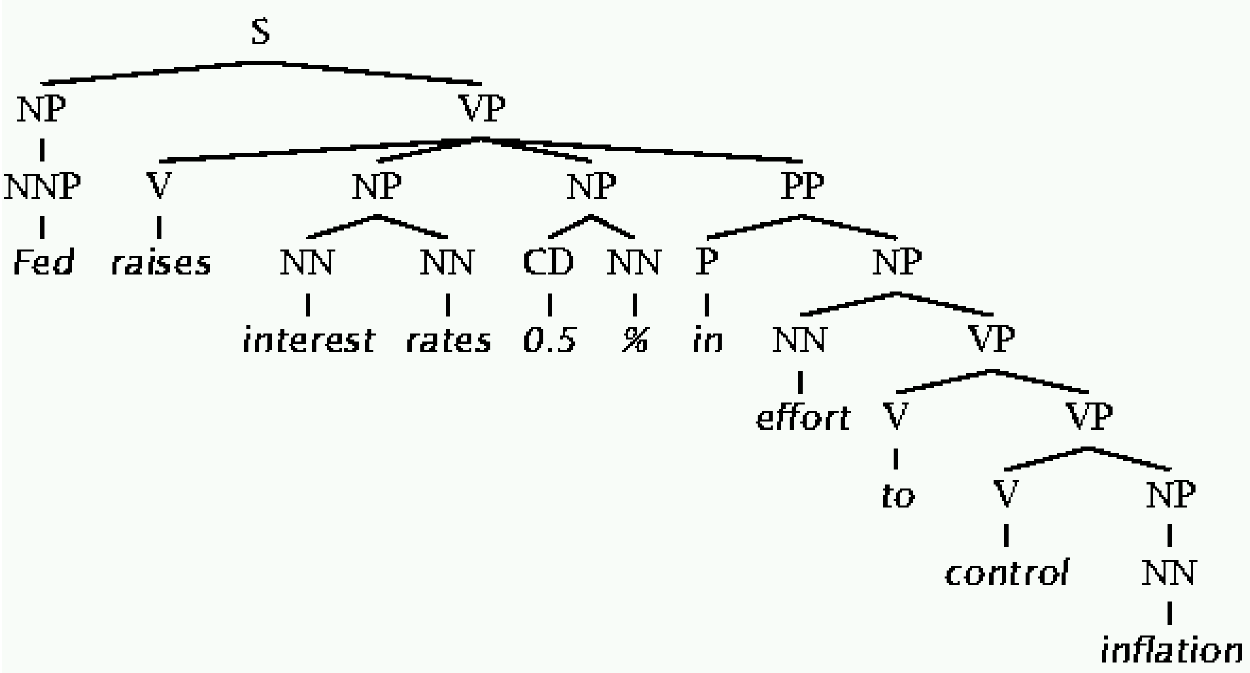
$G \approx 15{,}000$ $= 2 \times 10^8$

$N \approx 70$ $=343K$


*Center of Signal and Image Processing*
*Georgia Institute of Technology*

# Why is NL Understanding Difficult?

- Hidden structure of language is highly ambiguous
- Tree for: *Fed raises interest rates 0.5% in effort to control inflation* (*NYT* headline 5/17/00)



# Where Are the Ambiguities?

Part of speech ambiguities

*Syntactic attachment ambiguities*

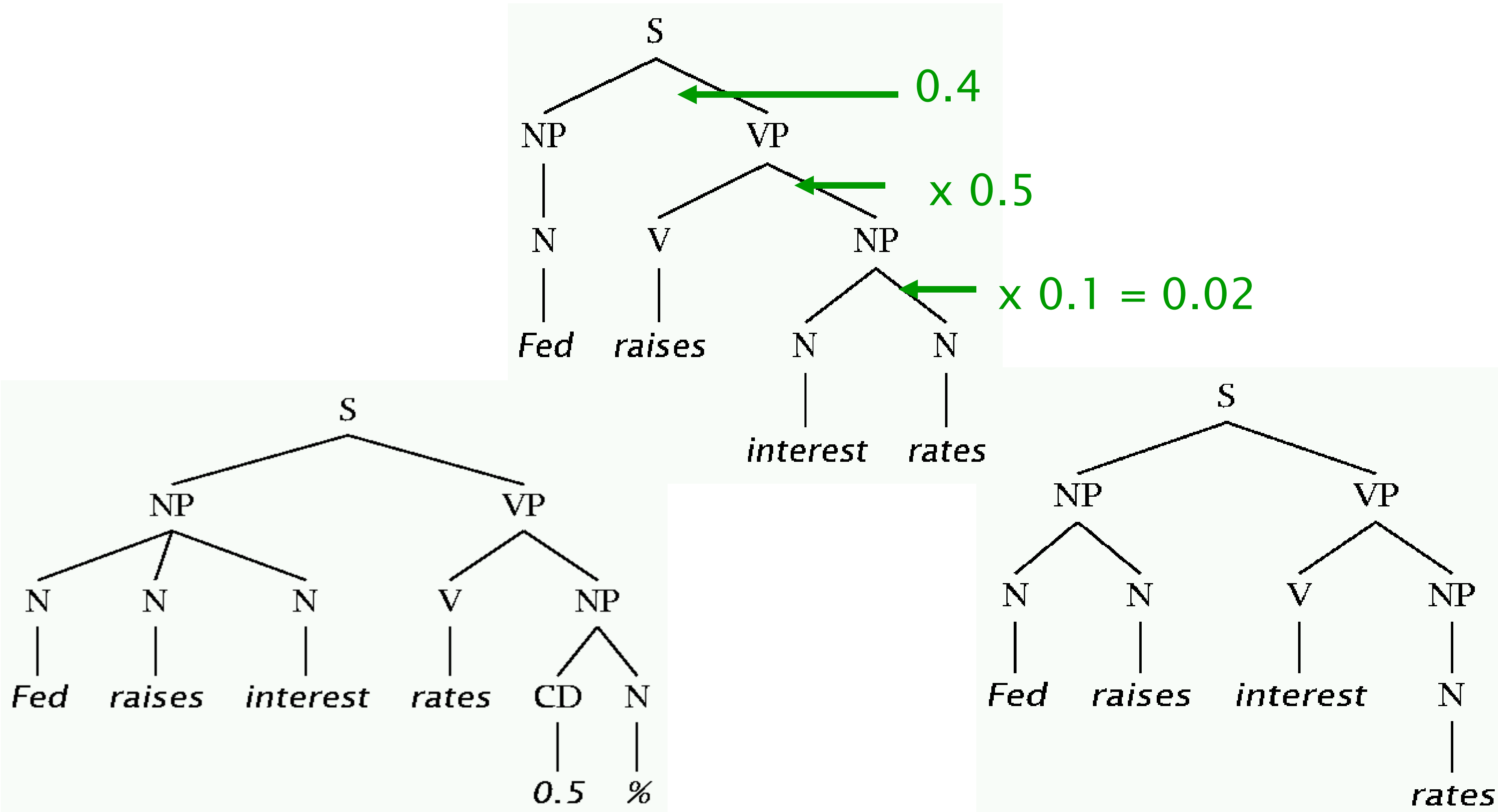
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | VB | | | | | |
| | VBZ | VBP | VBZ | | | | |
| NNP | NNS | NN | NNS | CD | NN | | |
| *Fed* | *raises* | *interest* | *rates* | *0.5* | *%* | *in* | *effort* |
| | | | | | | *to* | *control* |
| | | | | | | | *inflation* |

*Word sense ambiguities:* Fed → *"federal agent"*

interest → *a feeling of wanting to know or learn more*

*Semantic interpretation ambiguities above the word level*

# The Bad Effects of V/N Ambiguities



# Ambiguity of English: Newspaper Headlines

- Ban on Nude Dancing on Governor's Desk – *from a Georgia newspaper discussing current legislation*
- Juvenile Court to Try Shooting Defendant
- Teacher Strikes Idle Kids
- Stolen Painting Found by Tree
- Local High School Dropouts Cut in Half
- Red Tape Holds Up New Bridges
- China to orbit human on Oct. 15
- Moon wants to go to space

# Parsing for Disambiguation

- Probabilities for determining the sentence: choose sequence of words from a word lattice with highest probability (language model)
- Probabilities for speedier parsing: prune the search space of a parser
- Probabilities for choosing between parses: choose most likely among many parses of the input sentence

# Weakening the Independence Assumptions

- In PCFGs we make a number of independence assumptions
- Context: Humans make wide use of context
  - Context of who we are talking to, where we are, prior context of the conversation
  - Prior discourse context
  - People find semantically intuitive readings for sentences
- We need to incorporate these sources of information to build better parsers than PCFGs

# Weakening the Independence Assumptions

- Lexicalization: The PCFG independence assumptions do not take into consideration the particular words in the sentence
  - We need to include more information about the individual words when making decisions about the parse tree structure
- Structural Context: Certain types have location preferences in the parse tree
- In the PCFG case the way we derive (order of rewriting) the tree does not alter the tree probability

# Phrase Structure & Dependency Grammars

- In a dependency grammar, one word is the head of a sentence, and all other words are either a dependent of that word, or else dependent on some other word which connects to the head word through a series of dependencies
  - Lexicalized: Dependencies between words are taken care of
  - Gives a way of decomposing phrase structure rules

# Treebanks

- A collection of example parses by experts
- A commonly used treebank is the *Penn Treebank* *http://www.cis.upenn.edu/~treebank/*
- The induction problem is now that of extracting the grammatical knowledge that is implicit in the example parses
- Treebanks for other languages: Korean, Chinese

# PCFG Estimation (Charniak, 1996)

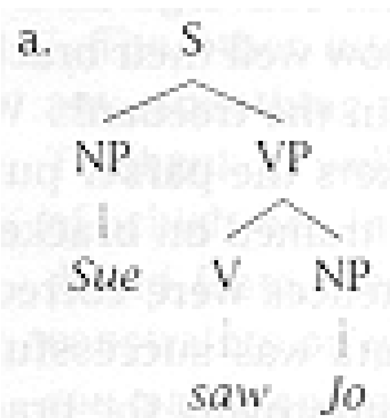
- Uses Penn Treebank POS and phrasal categories to induce a maximum likelihood based PCFG
  - by using the relative frequency of local trees as the estimates for rules
  - no attempt to do any smoothing or collapsing of rules
- Works surprisingly well: majority of parsing decisions are mundane and can be handled well by non-lexicalized PCFG

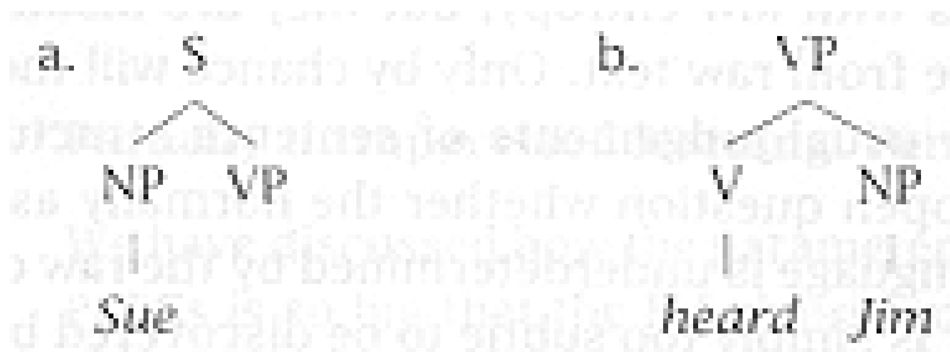
# Partially Unsupervised Learning
## (Pereira and Schabes, 1992)

- The parameter estimation space is too big for PCFGs that are of realistic sizes
- Some good practices:
  - Begin with a Chomsky normal form grammar with limited non-terminals and POS tags
  - Train on Penn treebank sentences
  - ignore the non-terminal labels, but use the treebank bracketing
  - Use a modified Inside-Outside algorithm constrained to consider parses that do not cross Penn-Treebank nodes

# Data Oriented Parsing

- Use whichever fragments of trees appear to be useful, can be multiple yet distinct parses
- Parse using Monte Carlo simulation methods
  - prob. is estimated by taking random samples of derivations



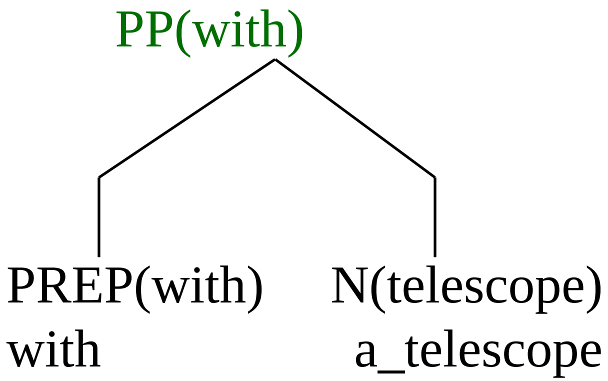



# History Based Grammars (HBG)

- All prior parse decisions could influence following parse decisions in the derivation
- (Black et al. 1993)
  - Use decision trees to decide which features in the derivational history were important in determining the expansion of the current node
  - Consider only nodes on a path to the root

# Once again, Lexicalization

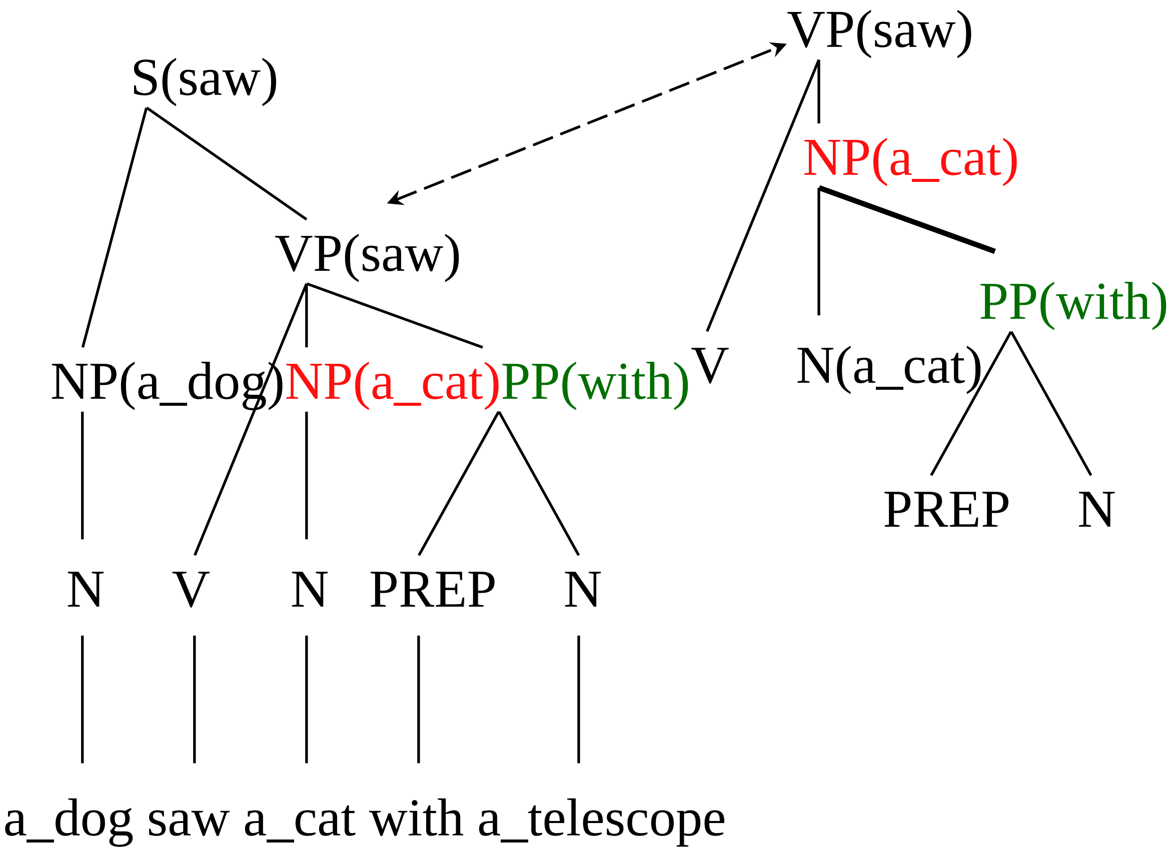
- Lexicalized parse tree (~ dependency tree+phrase labels)
- Ex. subtree:




- Pre-terminals (above leaves): assign the word below
- Recursive step (step up one level): (a) select node, (b) copy word up

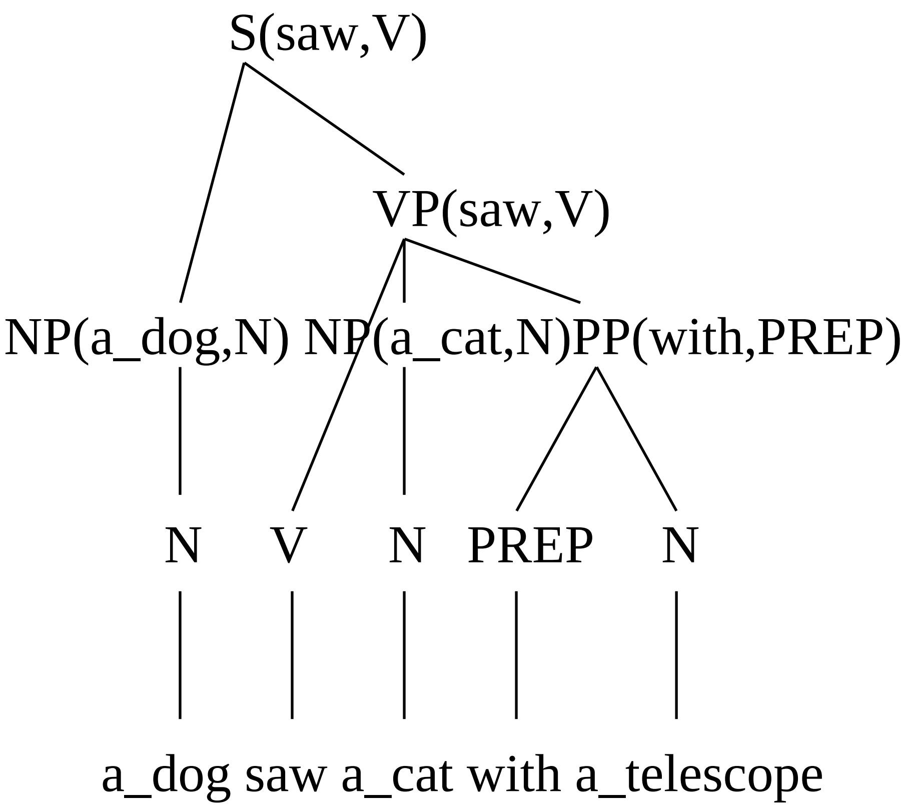
# Lexicalized Tree Example

- #1 S → NP VP
- #2 VP → V NP PP
- #3 VP → V NP
- #4 NP → N
- #5 NP → N PP
- #6 PP → PREP N
- #7 N → a_dog
- #8 N → a_cat
- #9 N → a_telescope
- #10 V → saw
- #11 PREP → with



# Using PoS Tags

- Head ~ word,tag






*Center of Signal and Image Processing*
*Georgia Institute of Technology*

# Conditioning

- Original PCFG: $P(\alpha B \gamma D \varepsilon ... | A)$
  - No "lexical" units (words)
- Introducing words:

$P(\ \alpha\ \mathrm{B}(\mathrm{head}_\mathrm{B})\ \gamma\ \mathrm{D}(\mathrm{head}_\mathrm{D})\ \varepsilon\ ...\ |\mathrm{A}(\mathrm{head}_\mathrm{A}))$

where $\mathrm{head}_\mathrm{A}$ is one of the heads on the left

e.g. rule VP(saw) $\rightarrow$ V(saw) NP(a_cat):

*P*(V(saw) NP(a_cat) | VP(saw))

# Independence Assumptions

- Too many rules
- Decompose:

$$P(\ \alpha\ \mathrm{B}(\mathrm{head}_{\mathrm{B}})\ \gamma\ \mathrm{D}(\mathrm{head}_{\mathrm{D}})\ \varepsilon\ \ldots\ |\mathrm{A}(\mathrm{head}_{\mathrm{A}})) =$$

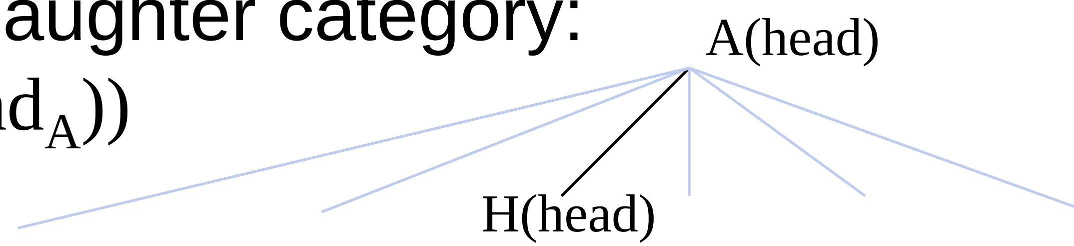
- In general (total independence):

$$P(\alpha|\mathrm{A}(\mathrm{head}_{\mathrm{A}})) \times P(\mathrm{B}(\mathrm{head}_{\mathrm{B}})|\mathrm{A}(\mathrm{head}_{\mathrm{A}})) \times \ldots \times P(\varepsilon|\mathrm{A}(\mathrm{head}_{\mathrm{A}}))$$

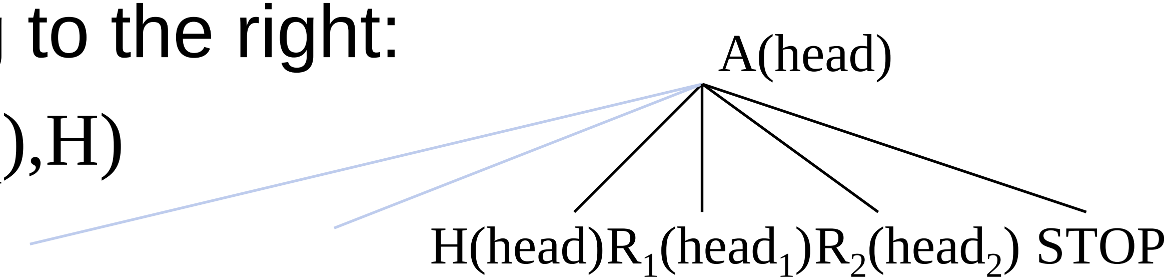
- Too much independent: need a compromise

# The Decomposition

- Order does not matter, use intuition ("linguistics")
- Select the head daughter category:
  $P_H(H(head_A)|A(head_A))$



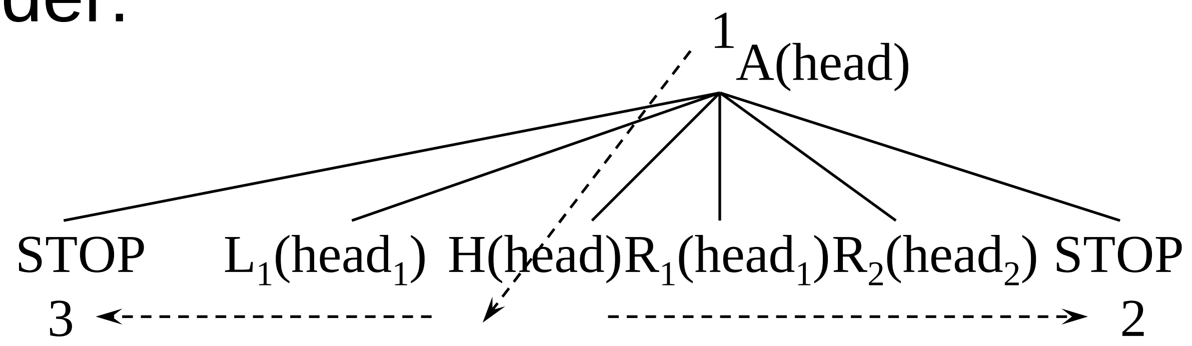

- Select everything to the right:
  $P_R(R_i(r_i) \mid A(head_A),H)$



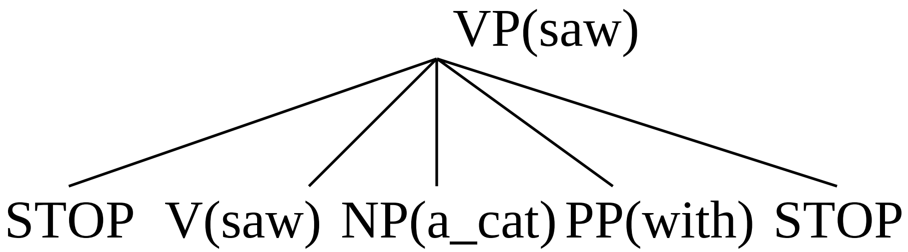

- Also, choose when to finish: $R_{m+1}(r_{m+1}) = STOP$
- Similarly, for the left direction: $P_L(L_i(l_i) \mid A(head_A),H)$

# Example Decomposition

- Order:




- Example:

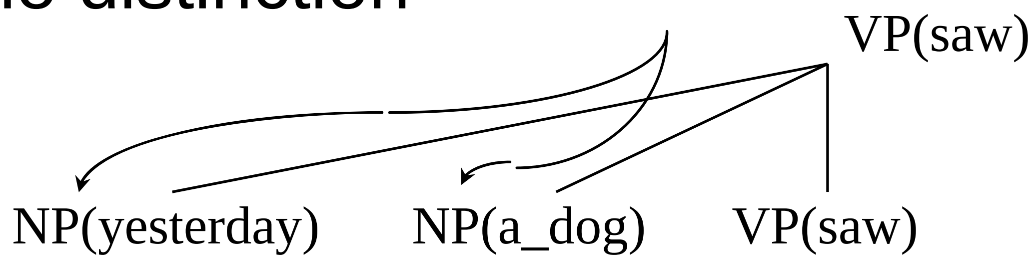


# More Conditioning: Distance

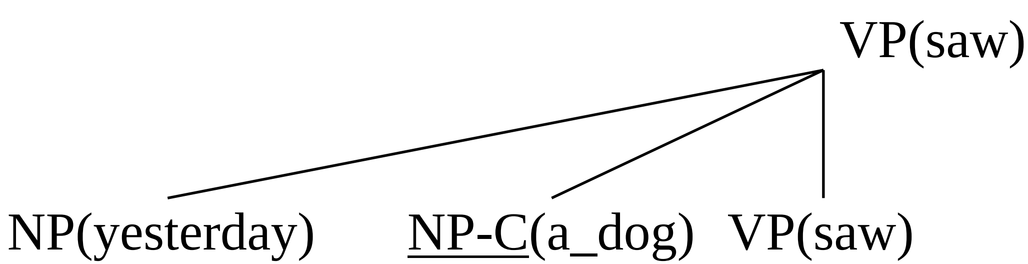
- Motivation:
  - close words tend to be dependents (or phrases) more likely
  - "walking on a sidewalk on a sunny day without looking on.."
- Number of words too detailed, though:
  - use more sophisticated (yet robust) distance measure $d_{r/l}$:
    - distinguish 0 and non-zero distance (2)
    - distinguish if verb is in-between the head and the constituent in question (2)
    - distinguish if there are commas in-between: 0, 1, 2, >2 commas (4)
    - total: 16 possibilities added: $P_R(R_i(r_i) \mid A(head_A),H,d_r)$
    - same to the left: $P_L(L_i(l_i) \mid A(head_A),H,d_l)$

# More Conditioning: Complement/Adjunct

- So far: no distinction



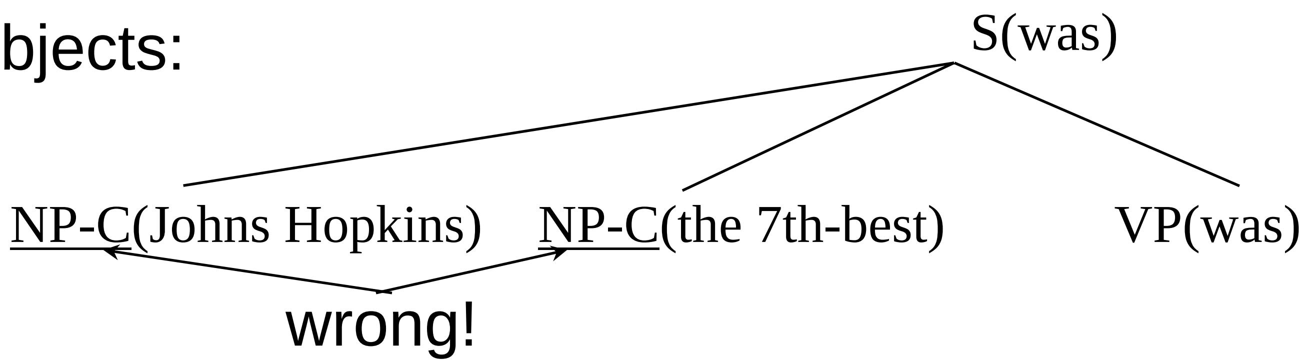

- ...but: time NP ≠ subject NP
- also, Subject NP cannot repeat... useful <u>during</u> parsing




[Must be added in training data]

# More Conditioning: Subcategorization

- The problem still not solved:
  - two subjects:




- Need: relation among complements
  - [linguistic observation: adjuncts can repeat freely.]
- Introduce:
  - Left & Right Subcategorization Frames (multisets)

# Inserting Subcategorization

- Use head probability as before:

  $P_H(H(head_A)|A(head_A))$

- Then, add left & right subcat frame:

  $P_{lc}(LC|\ A(head_A),H)$, $P_{rc}(RC|\ A(head_A),H)$

  LC, RC: list (multiset) of phrase labels (not words)

- Add them to context condition:

  (left) $P_L(L_i(l_i)\ |\ A(head_A),H,d_l,LC)$ [right: similar]

- LC/RC: “dynamic”: remove labels when generated

  $P(STOP|.....,LC) = 0$ if LC non-empty


Center of Signal and Image Processing
Georgia Institute of Technology
CSIP

# Smoothing

- Adding conditions... ~ adding parameters
- Sparse data problem as usual (head ~ <word,tag>!)
- Smooth (step-wise):

  $P_{\text{smooth-H}}(\text{H}(\text{head}_A)|\text{A}(\text{head}_A)) = w_1 P_{\text{H}}(\text{H}(\text{head}_A)|\text{A}(\text{head}_A)) + (1-w_1) P_{\text{smooth-H}}(\text{H}(\text{head}_A)|\text{A}(\text{tag}_A))$

  $P_{\text{smooth-H}}(\text{H}(\text{head}_A)|\text{A}(\text{tag}_A)) =$

  $w_2 P_{\text{H}}(\text{H}(\text{head}_A)|\text{A}(\text{tag}_A)) + (1-w_2) P_{\text{H}}(\text{H}(\text{head}_A)|\text{A})$
- Similarly, for $P_R$ and $P_L$

# Parsing Algorithm for a Lexicalized PCFG

- Bottom-up Chart parsing
  - Elements of a chart: a pair
    - <(from-position,to-position,label,head,distance), probability>
    - span - score -
  - Total probability = multiplying elementary probabilities

  $\rightarrow$ enables dynamic programming:
    - discard chart element with the same span but lower score.
- "Score" computation:
  - joining chart elements: [for 2]: $<e_1, p_1>$, $<e_2, p_2>$, $<e_n, p_n>$:
    - $P(e_{new}) = p_1$ ´ $p_2$ ´ ... ´ $p_n$ ´ $P_H(...)$ ´ $PP_R(...)$ ´ $PP_L(...)$;



Center of Signal and Image Processing
Georgia Institute of Technology

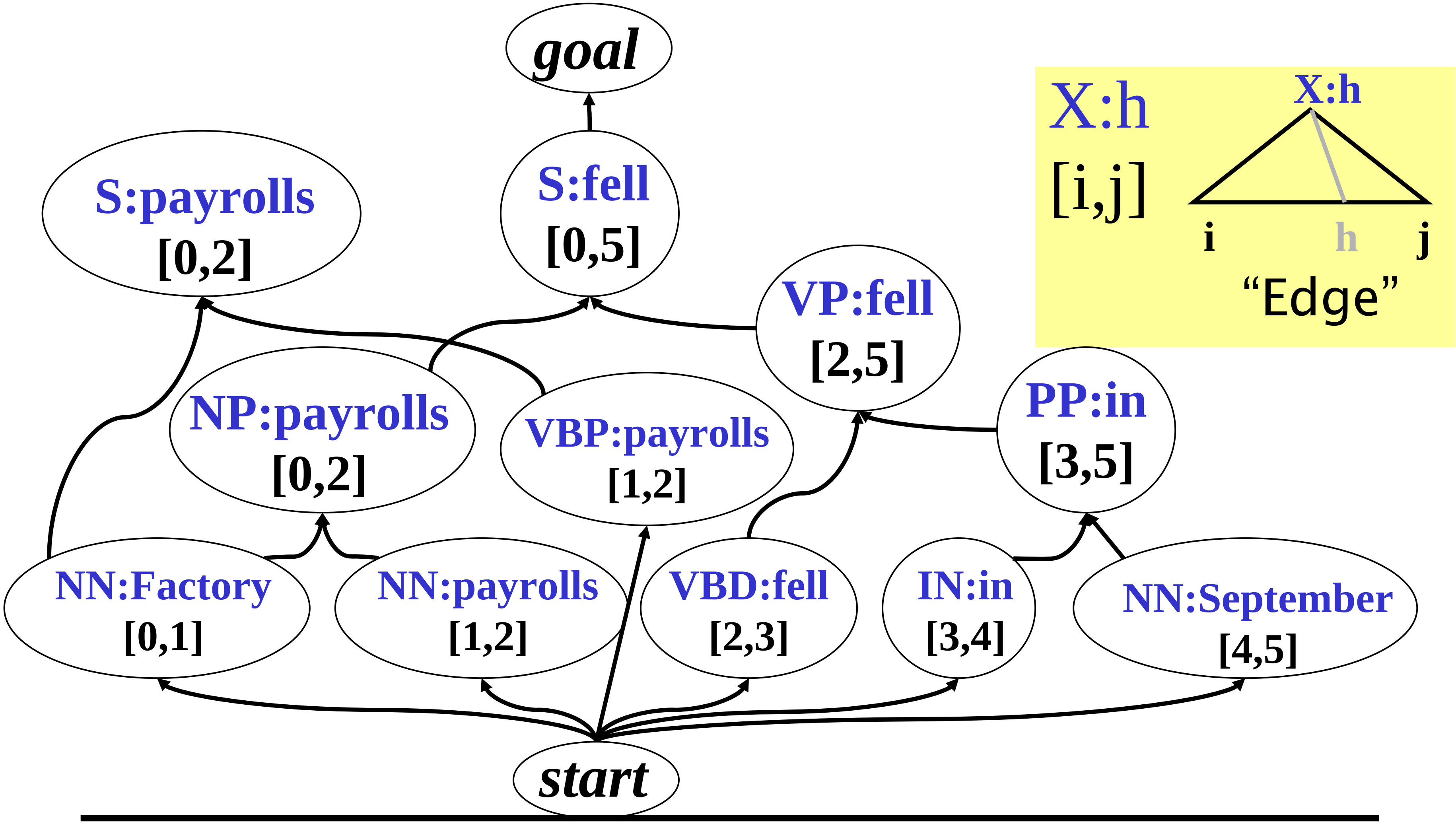
# Evaluation

- **Exact Match Criterion**: Compare parser performance with hand parses of sentences give 1 for exact match and 0 for any mistake
- **Parseval Measures**: Measure based on precision, recall and crossing brackets. Not very discriminating
- Partial Match Criterion
- Success in real tasks

# Equivalent Models

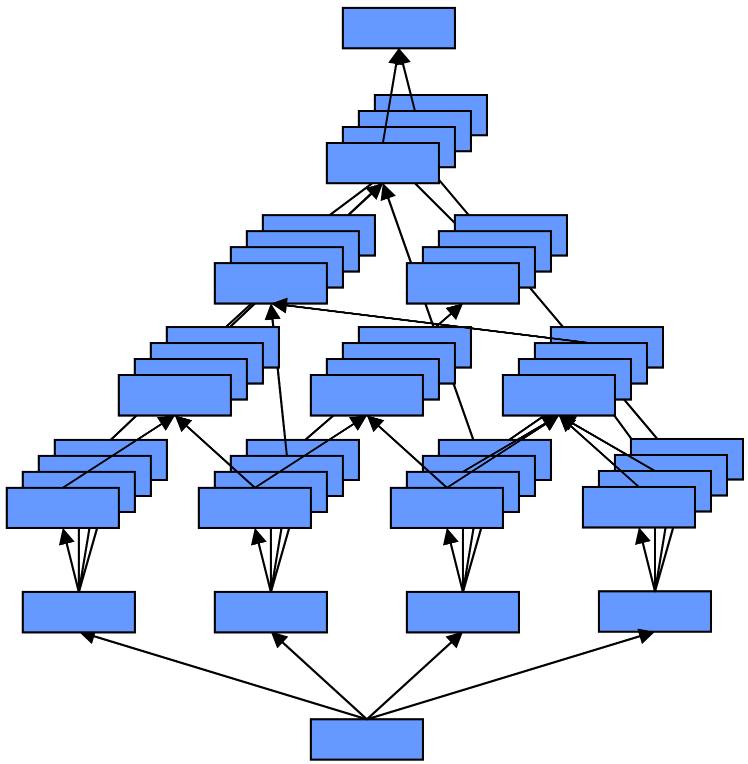
- Compare models in terms of what information is being used to condition the prediction of what Improving the Models by:
  - Remembering more of derivational history
  - Looking at bigger context in a phrase structure tree
  - Enriching the vocabulary of the tree in deterministic ways

# Parsing as Search



# CKY Parsing (Chart Parsing)

- In CKY parsing, we visit edges by span size:



- Guarantees correctness by working inside-out.
- Build all small bits before any larger bits that could possibly require them.
- Exhaustive: the goal is among the nodes with largest span size!

# What Can Go Wrong?

- We can build too many edges
  - Most edges that can be built, shouldn't
  - CKY builds them all!

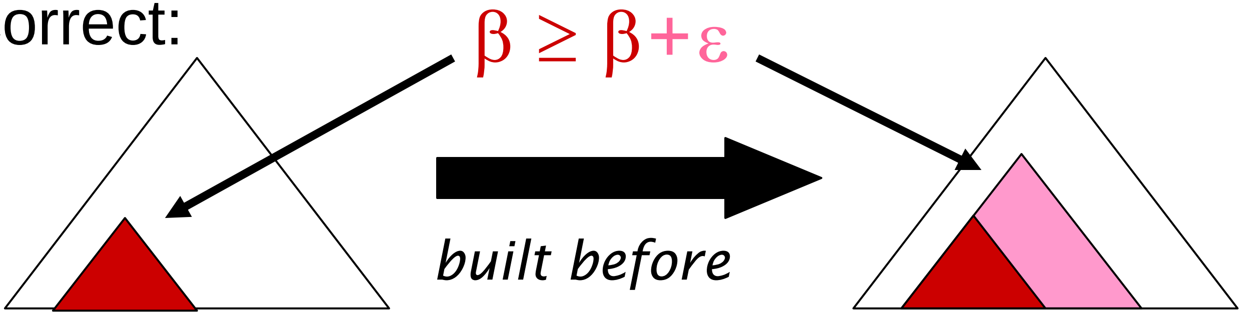
Speed: build promising edges first

- We can build in an bad order
  - Might find bad parses before good parses
  - Will trigger best-first propagation

Correctness: keep edges on the agenda until you're sure you've seen their best parse.

# Uniform-Cost Parsing

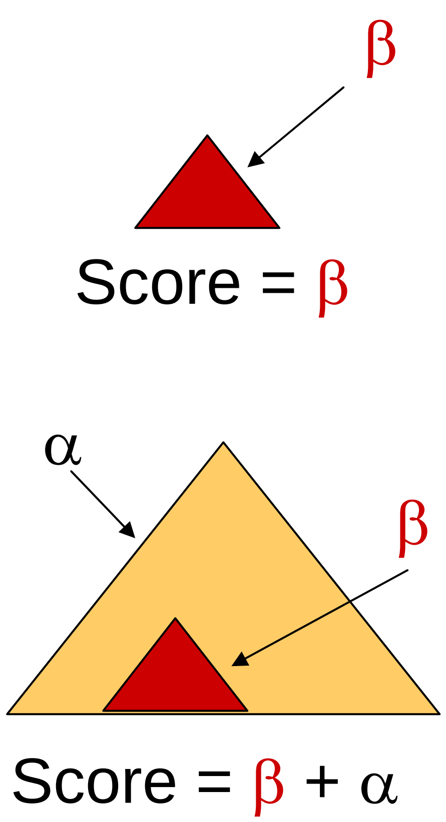
- We want to work on good parses inside-out
  - CKY does this synchronously, by span size
  - Uniform-cost orders edges by their best known score
- Why it's correct:




  - Adding structure incurs probability cost.
  - Trees have lower probability than their sub-parts.
- What makes things tricky:
  - We don't have a full graph to explore
  - The graph is built dynamically; correctness depends on the right bits of the graph being built before an edge is finished

# A* Search

- Problem with uniform-cost:
  - Even unlikely small edges have high score
  - We end up processing every small edge!
- Solution: A* Search
  - Small edges have to fit into a full parse
  - The smaller the edge, the more the full parse will cost
  - Consider both the cost to build (β) and the cost to complete (α)
- We figure out β during parsing
- We GUESS at α in advance (pre-processing)



# Results

- English, WSJ, Penn Treebank, 40k sentences

| | < 40Words | < 100 Words |
|---|---|---|
| – Labeled Recall: | 88.1% | 87.5% |
| – Labeled Precision: | 88.6% | 88.1% |
| – Crossing Brackets (avg): | 0.91 | 1.07 |
| – Sentences With 0 CBs: | 66.4% | 63.9% |

- Prague Dependency Treebank, 13k sentences:
  - Dependency Accuracy overall: 80.0%

    (~ unlabelled precision/recall)

# Summary

- Today's Class
  - Statistical Parsing
- Next Classes
  - Question Answering (last lecture)
  - Lab 6 due on 4/14
  - Final on 4/27 at 8:00-10:50
  - Project monitoring
    - Project Report due at midnight on 4/29 (or before 8am on 4/30)
  - Project Presentation on 4/16
    - In alphabetical order (15 minutes each)
- Reading Assignments
  - Manning and Schutze, Chapters 11-12