



*INF4820: Algorithms for
Artificial Intelligence and
Natural Language Processing*

Chart Parsing

Stephan Oepen & Erik Velldal

Language Technology Group (LTG)

October 28, 2015

Last Time

- ▶ Mid-Way Evaluation
- ▶ Forward Algorithm
- ▶ Quiz & Bonus Points
- ▶ Syntactic Structure

Today

- ▶ Context-Free Grammar
- ▶ Treebanks
- ▶ Probabilistic CFGs
- ▶ Syntactic Parsing
 - ▶ Naïve: Recursive-Descent
 - ▶ Dynamic Programming: CKY

Recall: Question (2): Language Modelling



*Group members at the Language Technology Group
supervise a variety of topics for MSc projects
in natural language processing.
Many candidate projects are available on-line.
Please make contact with us.*

**(2) What is the probability of the bi-gram
language technology
when ignoring case and punctuation,
and using Laplace smoothing?**

Recall: Interpreting the Questions?



- ? *technology* following right after *language* $\rightarrow P(B|A)$
- ? *language technology* occurring somewhere $\rightarrow P(A, B)$
- ? ~~*language and technology* occurring somewhere~~ $\rightarrow P(A, B)$

Recall: Joint and Conditional Probabilities

$$P(A, B) = P(A) \times P(B|A)$$

$$A \equiv \{w_{i-1} = \text{language}\} \quad B \equiv \{w_i = \text{technology}\}$$

Alternatively: A Complex Event

$$A \equiv \{w_{i-1} = \text{language} \wedge w_i = \text{technology}\}$$

Constituency

- ▶ Words tends to lump together into groups that behave like single units: we call them *constituents*.
- ▶ *Constituency tests* give evidence for constituent structure:
 - ▶ interchangeable in similar syntactic environments.
 - ▶ can be co-ordinated
 - ▶ can be moved within a sentence as a unit

- (1) Kim read [a very interesting book about grammar]_{NP}.
Kim read [it]_{NP}.
- (2) Kim [read a book]_{VP}, [gave it to Sandy]_{VP}, and [left]_{VP}.
- (3) [Interesting books about grammar] I like.

Recall: Grammar Aids Understanding



Formal grammars describe a language, giving us a way to:

- ▶ judge or predict well-formedness

Kim was happy because _____ passed the exam.

Kim was happy because _____ final grade was an A.

- ▶ make explicit structural ambiguities

Have her report on my desk by Friday!

I like to eat sushi with { chopsticks | tuna }.

- ▶ derive abstract representations of meaning

Kim gave Sandy a book.

Kim gave a book to Sandy.

Sandy was given a book by Kim.

A Grossly Simplified Example



The Grammar of Spanish

$S \rightarrow NP VP$ $\{ VP(NP) \}$

$VP \rightarrow V NP$ $\{ V(NP) \}$

$VP \rightarrow VP PP$ $\{ PP(VP) \}$

$PP \rightarrow P NP$ $\{ P(NP) \}$

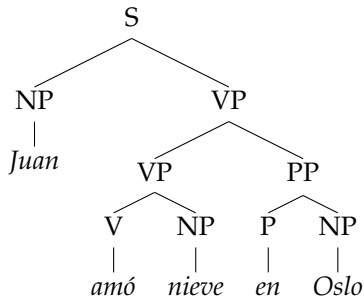
$NP \rightarrow \text{"nieve"}$ $\{ \text{snow} \}$

$NP \rightarrow \text{"Juan"}$ $\{ \text{John} \}$

$NP \rightarrow \text{"Oslo"}$ $\{ \text{Oslo} \}$

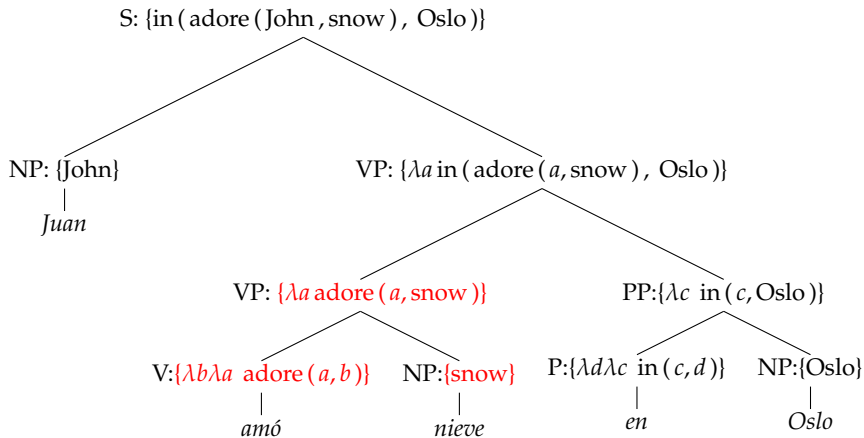
$V \rightarrow \text{"amó"}$ $\{ \lambda b \lambda a \text{ adore}(a, b) \}$

$P \rightarrow \text{"en"}$ $\{ \lambda d \lambda c \text{ in}(c, d) \}$



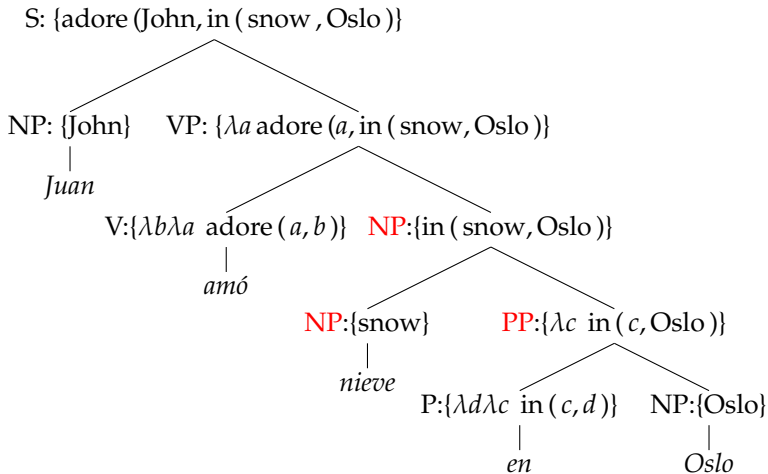
Juan amó nieve en Oslo

Meaning Composition (Still Very Simplified)



$VP \rightarrow V NP \quad \{ V(NP) \}$

Another Interpretation



$NP \rightarrow NP PP \{ PP (NP) \}$

Context Free Grammars (CFGs)



- ▶ Formal system for modeling constituent structure.
- ▶ Defined in terms of a lexicon and a set of rules
- ▶ Formal models of 'language' in a broad sense
 - ▶ natural languages, programming languages, communication protocols, . . .
- ▶ Can be expressed in the 'meta-syntax' of the Backus-Naur Form (BNF) formalism.
 - ▶ When looking up concepts and syntax in the Common Lisp HyperSpec, you have been reading (extended) BNF.
- ▶ Powerful enough to express sophisticated relations among words, yet in a computationally tractable way.

CFGs (Formally, this Time)



Formally, a CFG is a quadruple: $G = \langle C, \Sigma, P, S \rangle$

- ▶ C is the set of categories (aka *non-terminals*),
 - ▶ $\{S, NP, VP, V\}$
- ▶ Σ is the vocabulary (aka *terminals*),
 - ▶ $\{\text{Kim, snow, adores, in}\}$
- ▶ P is a set of category rewrite rules (aka *productions*)

$S \rightarrow NP VP$

$NP \rightarrow \text{Kim}$

$VP \rightarrow V NP$

$NP \rightarrow \text{snow}$

$V \rightarrow \text{adores}$

- ▶ $S \in C$ is the *start symbol*, a filter on complete results;
- ▶ for each rule $\alpha \rightarrow \beta_1, \beta_2, \dots, \beta_n \in P$: $\alpha \in C$ and $\beta_i \in C \cup \Sigma$

Top-down view of generative grammars:

- ▶ For a grammar G , the language \mathcal{L}_G is defined as the set of strings that can be derived from S .
- ▶ To derive w_1^n from S , we use the rules in P to recursively rewrite S into the sequence w_1^n where each $w_i \in \Sigma$
- ▶ The grammar is seen as **generating** strings.
- ▶ *Grammatical* strings are defined as strings that can be generated by the grammar.
- ▶ The 'context-freeness' of CFGs refers to the fact that we rewrite non-terminals without regard to the overall context in which they occur.

Generally

- ▶ A *treebank* is a corpus paired with ‘gold-standard’ (syntactic) analyses
- ▶ Can be created by manual annotation or selection among outputs from automated processing (plus correction).

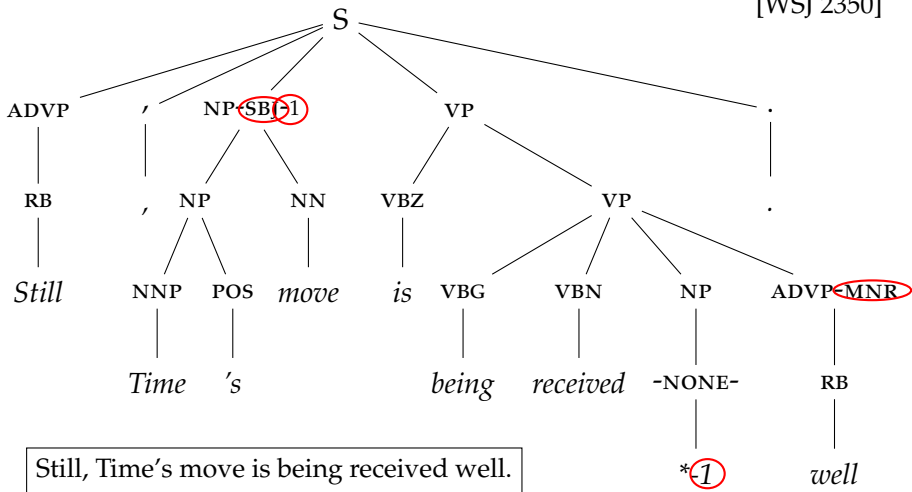
Penn Treebank (Marcus et al., 1993)

- ▶ About one million tokens of Wall Street Journal text
- ▶ Hand-corrected PoS annotation using 45 word classes
- ▶ Manual annotation with (somewhat) coarse constituent structure

One Example from the Penn Treebank



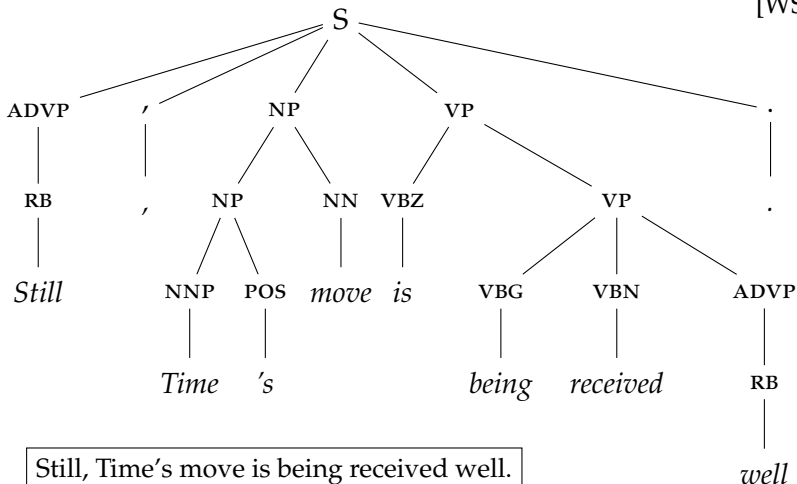
[WSJ 2350]



Elimination of Traces and Functions



[WSJ 2350]



Probabilistic Context-Free Grammars

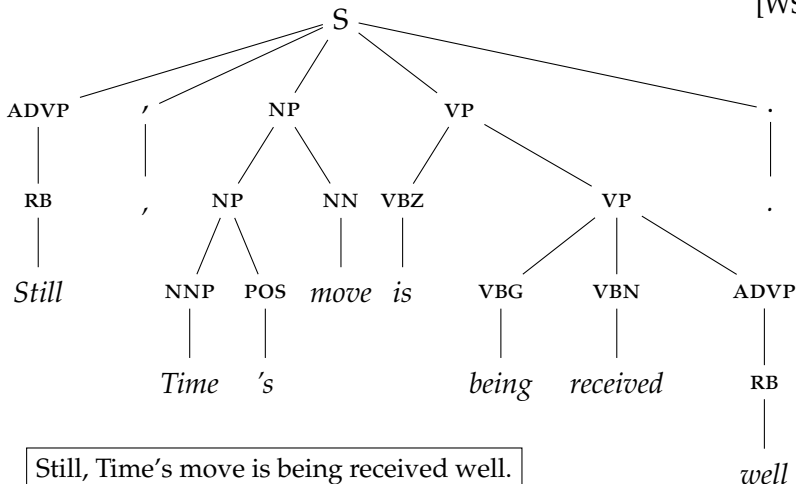


- ▶ We are interested, not just in which trees apply to a sentence, but also to which tree is **most likely**.
- ▶ Probabilistic context-free grammars (PCFGs) augment CFGs by adding probabilities to each production, e.g.
 - ▶ $S \rightarrow NP VP$ 0.6
 - ▶ $S \rightarrow NP VP PP$ 0.4
- ▶ These are conditional probabilities — the probability of the right hand side (RHS) given the left hand side (LHS)
 - ▶ $P(S \rightarrow NP VP) = P(NP VP|S)$
- ▶ We can learn these probabilities from a treebank, again using Maximum Likelihood Estimation.

Estimating PCFGs (1/3)



[WSJ 2350]



Estimating PCFGs (2/3)



```
(S
  (ADVP (RB "Still"))
  (|,| ",")
  (NP
    (NP (NNP "Time") (POS "'s"))
    (NN "move"))
  (VP
    (VBZ "is")
    (VP
      (VBG "being")
      (VP
        (VBN "received")
        (ADVP (RB "well"))))))
  (\. "."))
```

RB → Still	1
AVP → RB	2
, → ,	1
NNP → Time	1
POS → 's	1
NP → NNP POS	1
NN → move	1
NP → NP NN	1
VBZ → is	1
VBG → being	1
VBN → received	1
RB → well	1
VP → VBN ADVP	1
VP → VBG VP	1
\. → .	1
S → ADVP , NP VP \.	1
START → S	1

Estimating PCFGs (3/3)



Once we have counts of all the rules, we turn them into probabilities.

$S \rightarrow \text{ADVP } , \text{ NP VP } \backslash.$	50	$S \rightarrow \text{NP VP } \backslash.$	400
$S \rightarrow \text{NP VP PP } \backslash.$	350	$S \rightarrow \text{VP } !$	100
$S \rightarrow \text{NP VP S } \backslash.$	200	$S \rightarrow \text{NP VP}$	50

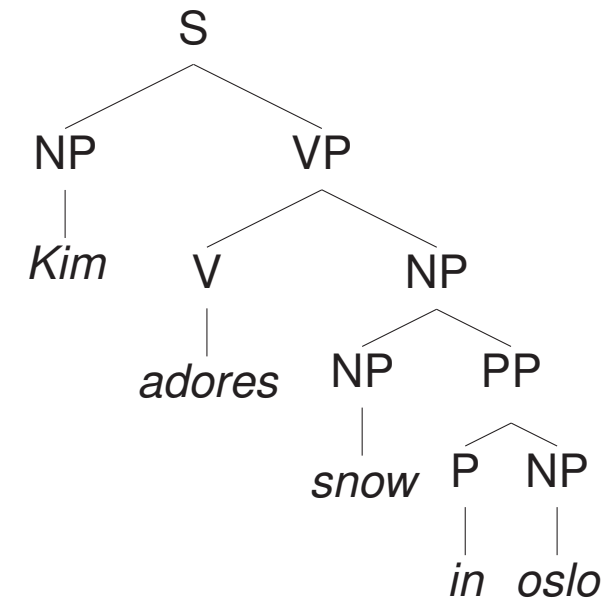
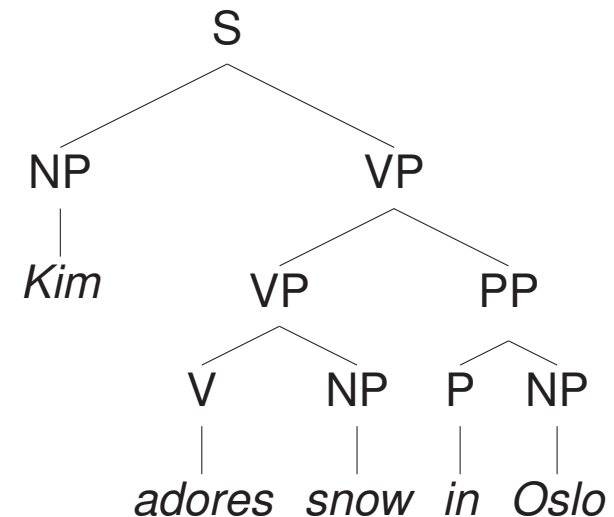
$$\begin{aligned} P(S \rightarrow \text{ADVP } |, | \text{ NP VP } \backslash.) &\approx \frac{C(S \rightarrow \text{ADVP } |, | \text{ NP VP } \backslash.)}{C(S)} \\ &= \frac{50}{1150} \\ &= 0.0435 \end{aligned}$$

Parsing with CFGs: Moving to a Procedural View

$S \rightarrow NP VP$
 $VP \rightarrow V \mid V NP \mid VP PP$
 $NP \rightarrow NP PP$
 $PP \rightarrow P NP$
 $NP \rightarrow Kim \mid snow \mid Oslo$
 $V \rightarrow adores$
 $P \rightarrow in$

All Complete Derivations

- are rooted in the start symbol S ;
- label internal nodes with categories $\in C$, leafs with words $\in \Sigma$;
- instantiate a grammar rule $\in P$ at each local subtree of depth one.



Recursive Descend: A Naïve Parsing Algorithm

Control Structure

- top-down: given a parsing goal α , use all grammar rules that rewrite α ;
- successively instantiate (extend) the right-hand sides of each rule;
- for each β_i in the RHS of each rule, recursively attempt to parse β_i ;
- termination: when α is a prefix of the input string, parsing succeeds.

(Intermediate) Results

- Each result records a (partial) tree and remaining input to be parsed;
- complete results consume the full input string and are rooted in S ;
- whenever a RHS is fully instantiated, a new tree is built and returned;
- all results at each level are combined and successively accumulated.



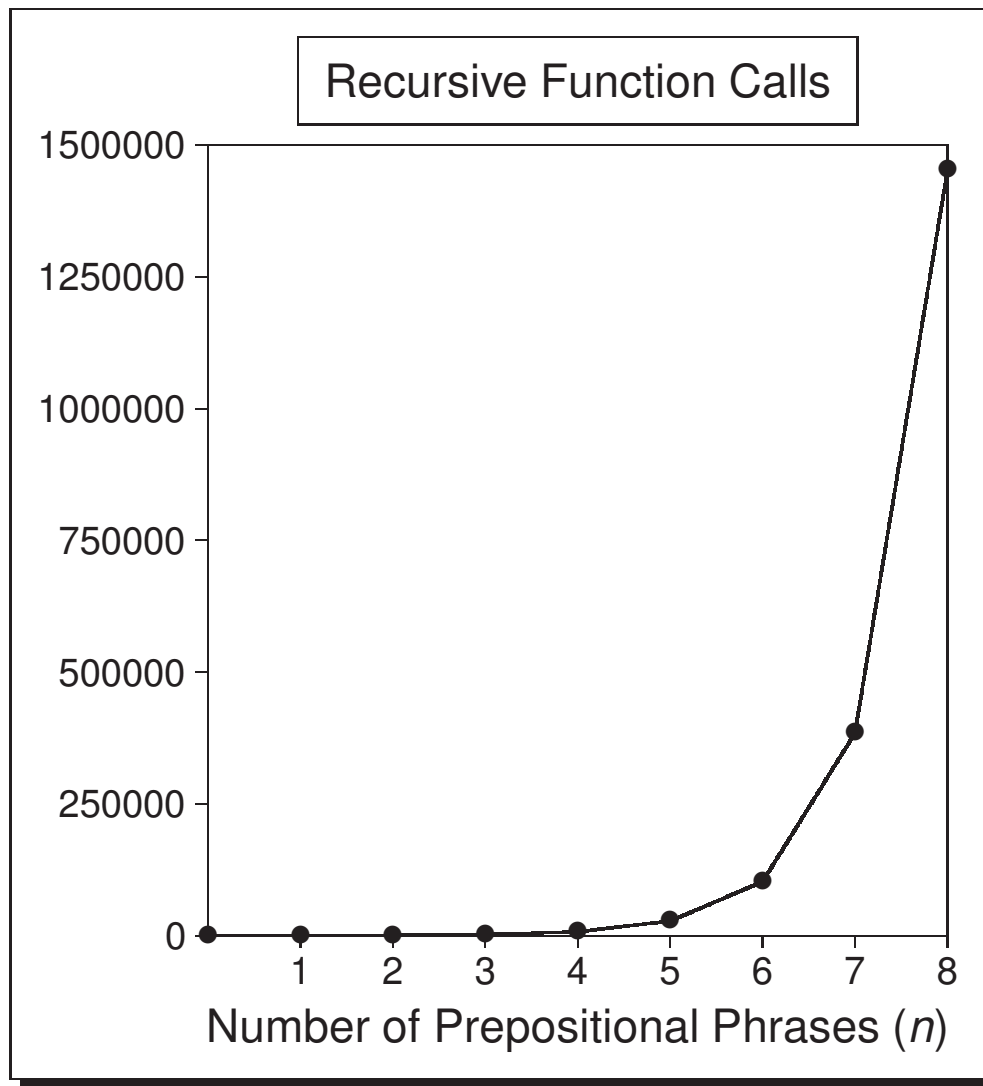
The Recursive Descent Parser

```
(defun parse (input goal)
  (if (equal (first input) goal)
      (let ((edge (make-edge :category (first input))))
        (list (make-parse :edge edge :input (rest input))))
      (loop
        for rule in (rules-deriving goal)
        append (extend-parse (rule-lhs rule) nil (rule-rhs rule) input))))
```

```
(defun extend-parse (goal analyzed unanalyzed input)
  (if (null unanalyzed)
      (let ((tree (cons goal analyzed)))
        (list (make-parse :tree tree :input input)))
      (loop
        for parse in (parse input (first unanalyzed))
        append (extend-parse
                  goal (append analyzed (list (parse-tree parse)))
                  (rest unanalyzed)
                  (parse-input parse)))))
```



Quantifying the Complexity of the Parsing Task



Kim adores snow (in Oslo)ⁿ

<i>n</i>	trees	calls
0	1	46
1	2	170
2	5	593
3	14	2,093
4	42	7,539
5	132	27,627
6	429	102,570
7	1430	384,566
8	4862	1,452,776
⋮	⋮	⋮



Top-Down vs. Bottom-Up Parsing

Top-Down (Goal-Oriented)

- Left recursion (e.g. a rule like ‘ $VP \rightarrow VP PP$ ’) causes infinite recursion;
 - search is uninformed by the (observable) input: can hypothesize many unmotivated sub-trees, assuming terminals (words) that are not present;
- assume bottom-up as basic search strategy for remainder of the course.

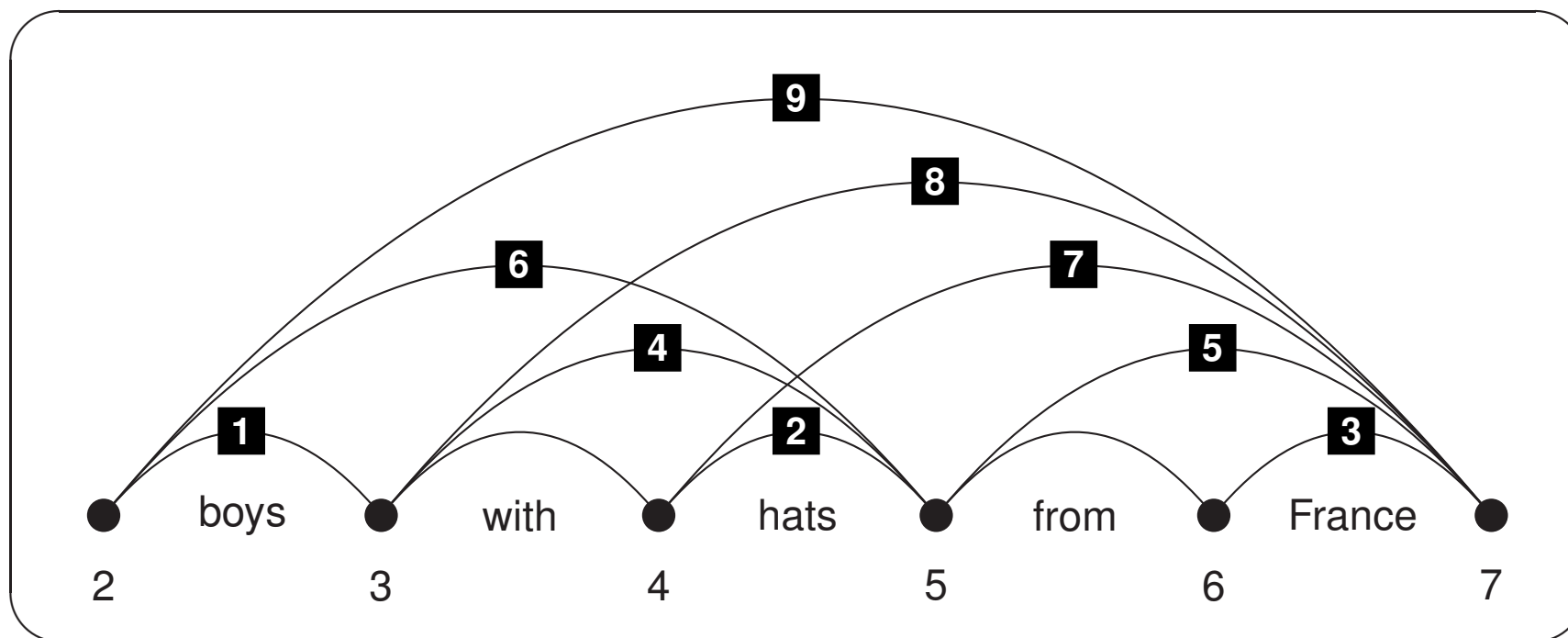
Bottom-Up (Data-Oriented)

- unary (left-recursive) rules (e.g. ‘ $NP \rightarrow NP$ ’) would still be problematic;
- lack of parsing goal: compute all possible derivations for, say, the input *adores snow*; however, it is ultimately rejected since it is not sentential;
- availability of partial analyses desirable for, at least, some applications.



A Key Insight: Local Ambiguity

- For many substrings, more than one way of deriving the same category;
- NPs: **1** | **2** | **3** | **6** | **7** | **9**; PPs: **4** | **5** | **8**; **9** \equiv **1** + **8** | **6** + **5**;
- *parse forest* — a single item represents multiple trees [Billot & Lang, 89].



The CKY (Cocke, Kasami, & Younger) Algorithm

```

for ( $0 \leq i < |input|$ ) do
   $chart_{[i,i+1]} \leftarrow \{\alpha \mid \alpha \rightarrow input_i \in P\};$ 
  for ( $1 \leq l < |input|$ ) do
    for ( $0 \leq i < |input| - l$ ) do
      for ( $1 \leq j \leq l$ ) do
        if ( $\alpha \rightarrow \beta_1 \beta_2 \in P \wedge \beta_1 \in chart_{[i,i+j]} \wedge \beta_2 \in chart_{[i+j,i+l+1]}$ ) then
           $chart_{[i,i+l+1]} \leftarrow chart_{[i,i+l+1]} \cup \{\alpha\};$ 

```

$[0,2] \leftarrow [0,1] + [1,2]$

...

$[0,5] \leftarrow [0,1] + [1,5]$

$[0,5] \leftarrow [0,2] + [2,5]$

$[0,5] \leftarrow [0,3] + [3,5]$

$[0,5] \leftarrow [0,4] + [4,5]$

	1	2	3	4	5
0	NP		S		S
1		V	VP		VP
2			NP		NP
3				P	PP
4					NP



Limitations of the CKY Algorithm

Built-In Assumptions

- *Chomsky Normal Form* grammars: $\alpha \rightarrow \beta_1\beta_2$ or $\alpha \rightarrow \gamma$ ($\beta_i \in C$, $\gamma \in \Sigma$);
- breadth-first (aka exhaustive): always compute all values for each cell;
- rigid control structure: bottom-up, left-to-right (one diagonal at a time).

Generalized Chart Parsing

- Liberate order of computation: no assumptions about earlier results;
- *active edges* encode partial rule instantiations, ‘waiting’ for additional (adjacent and passive) constituents to complete: $[1, 2, VP \rightarrow V \bullet NP]$;
- parser can fill in chart cells in *any* order and guarantee completeness.

