INF4820: Algorithms for Artificial Intelligence and Natural Language Processing

Chart Parsing

Stephan Oepen & Erik Velldal

Language Technology Group (LTG)

October 28, 2015

Overview

Last Time

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

Overview

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* occuring somewhere $\rightarrow P(A, B)$

Recall: Interpreting the Questions?

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

Recall: Interpreting the Questions?

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

Recall: Interpreting the Questions?

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

Recall: Joint and Conditional Probabilities

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

$$A \equiv \{w_{i-1} = \mathit{language}\} \;\; B \equiv \{w_i = \mathit{technology}\}$$

Recall: Interpreting the Questions?

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

Recall: Joint and Conditional Probabilities

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

$$A \equiv \{w_{i-1} = \mathit{language}\} \;\; B \equiv \{w_i = \mathit{technology}\}$$

Alternatively: A Complex Event

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

Recall: Syntactic Structures

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

Recall: Syntactic Structures

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

(4) Kim read [a very interesting book about grammar]$_{NP}$.
Kim read [it]$_{NP}$.

(5) Kim [read a book]$_{VP}$, [gave it to Sandy]$_{VP}$, and [left]$_{VP}$.

(6) [Interesting books about grammar] I like.

Examples from *Linguistic Fundamentals for NLP: 100 Essentials from Morphology and Syntax.* Bender (2013)

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.

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 }.

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 → NP VP

VP → V NP

VP → VP PP

PP → P NP

NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”

A Grossly Simplified Example

The Grammar of Spanish

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$VP \rightarrow VP\ PP$

$PP \rightarrow P\ NP$

$NP \rightarrow$ "nieve"

$NP \rightarrow$ "Juan"

$NP \rightarrow$ "Oslo"

$V \rightarrow$ "amó"

$P \rightarrow$ "en"

A Grossly Simplified Example

The Grammar of Spanish

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$VP \rightarrow VP\ PP$

$PP \rightarrow P\ NP$

$NP \rightarrow$ "nieve"

$NP \rightarrow$ "Juan"

$NP \rightarrow$ "Oslo"

$V \rightarrow$ "amó"

$P \rightarrow$ "en"

A Grossly Simplified Example

The Grammar of Spanish

$S \rightarrow NP\ VP$

$VP \rightarrow V\ NP$

$VP \rightarrow VP\ PP$

$PP \rightarrow P\ NP$

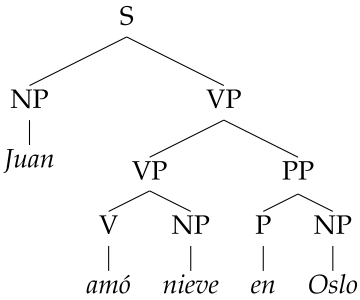
$NP \rightarrow$ “nieve”

$NP \rightarrow$ “Juan”

$NP \rightarrow$ “Oslo”

$V \rightarrow$ “amó”

$P \rightarrow$ “en”



Juan amó nieve en Oslo

A Grossly Simplified Example

The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

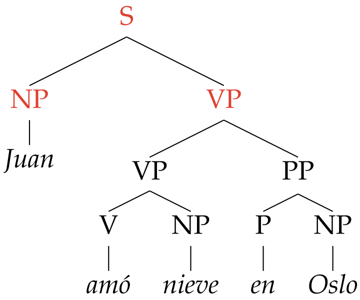
NP → "nieve"

NP → "Juan"

NP → "Oslo"

V → "amó"

P → "en"



Juan amó nieve en Oslo

A Grossly Simplified Example

The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

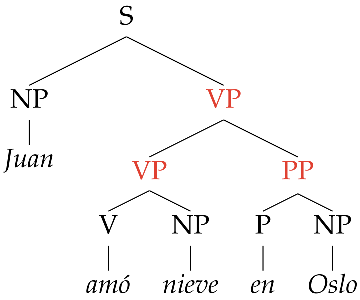
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”



Juan amó nieve en Oslo

A Grossly Simplified Example

The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

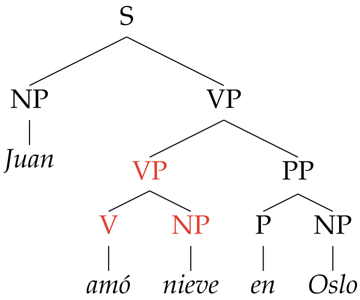
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”



Juan amó nieve en Oslo

A Grossly Simplified Example

The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

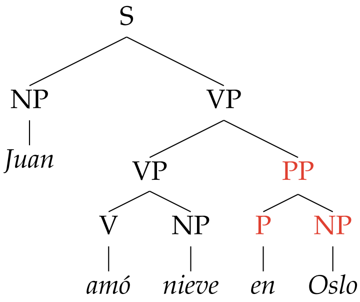
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”



Juan amó nieve en Oslo

A Grossly Simplified Example

The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

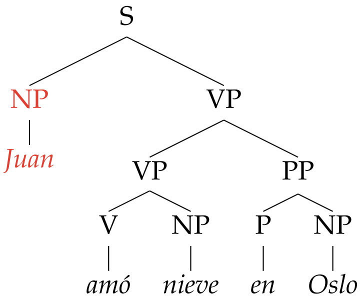
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”



Juan amó nieve en Oslo

A Grossly Simplified Example

The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

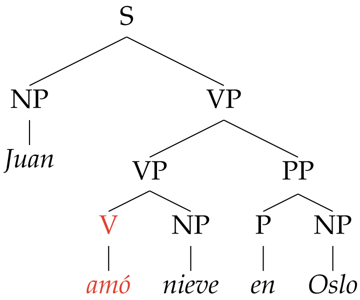
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”



Juan amó nieve en Oslo

A Grossly Simplified Example

The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

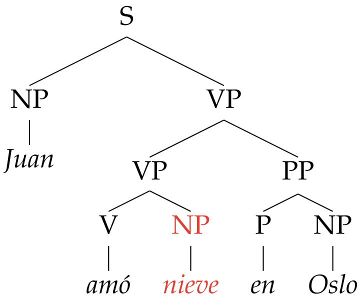
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”



Juan amó nieve en Oslo

A Grossly Simplified Example

The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

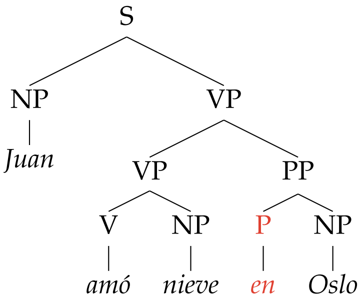
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”



Juan amó nieve en Oslo

A Grossly Simplified Example

The Grammar of Spanish

S → NP VP

VP → V NP

VP → VP PP

PP → P NP

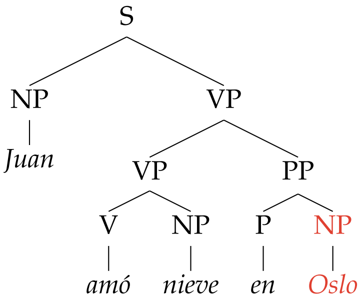
NP → “nieve”

NP → “Juan”

NP → “Oslo”

V → “amó”

P → “en”

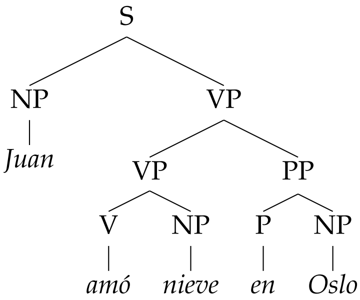


Juan amó nieve en Oslo

A Grossly Simplified Example

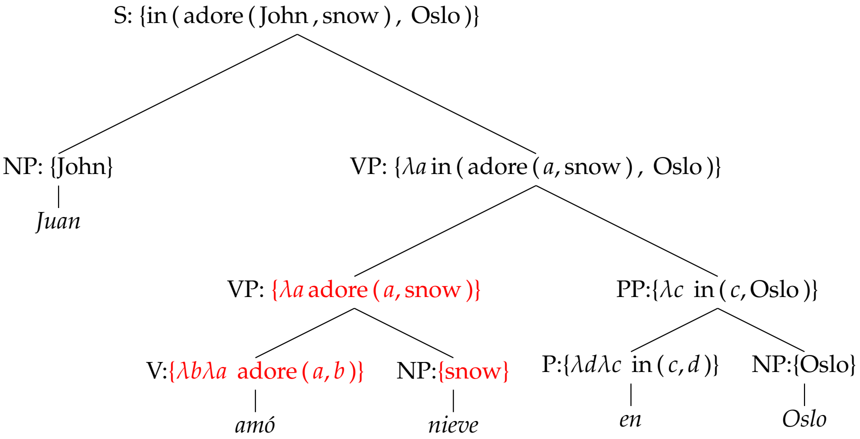
The Grammar of Spanish

$S \rightarrow NP\ VP$	$\{\, VP\,(\,NP\,)\,\}$
$VP \rightarrow V\ NP$	$\{\, V\,(\,NP\,)\,\}$
$VP \rightarrow VP\ PP$	$\{\, PP\,(\,VP\,)\,\}$
$PP \rightarrow P\ NP$	$\{\, P\,(\,NP\,)\,\}$
NP → “nieve”	$\{\,\text{snow}\,\}$
NP → “Juan”	$\{\,\text{John}\,\}$
NP → “Oslo”	$\{\,\text{Oslo}\,\}$
V → “amó”	$\{\,\lambda b \lambda a\ \text{adore}\,(\,a, b\,)\,\}$
P → “en”	$\{\,\lambda d \lambda c\ \text{in}\,(\,c, d\,)\,\}$



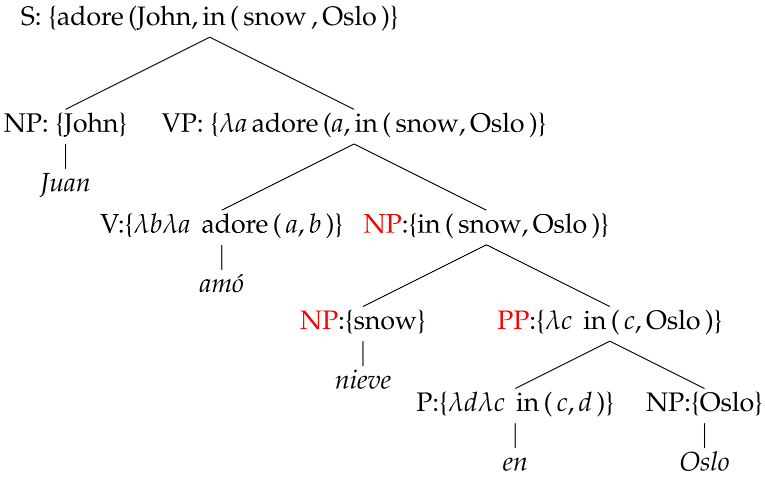
Juan amó nieve en Oslo

Meaning Composition (Still Very Simplified)



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

Another Interpretation



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

Context Free Grammars (CFGs)

- Formal system for modeling constituent structure.
- Defined in terms of a lexicon and a set of rules

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, . . .

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.

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: $G = \langle C, \Sigma, P, S \rangle$

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*),
  - $\{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{V}\}$

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*),
  - $\{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{V}\}$
- Σ is the vocabulary (aka *terminals*),
  - $\{\mathrm{Kim}, \mathrm{snow}, \mathrm{adores}, \mathrm{in}\}$

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

$$
\begin{array}{ll}
\mathrm{S} \rightarrow \mathrm{NP\ VP} & \mathrm{NP} \rightarrow \mathrm{Kim} \\
\mathrm{VP} \rightarrow \mathrm{V\ NP} & \mathrm{NP} \rightarrow \mathrm{snow} \\
 & \mathrm{V} \rightarrow \mathrm{adores}
\end{array}
$$

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

$$
\begin{array}{rcl}
\mathrm{S} \rightarrow \mathrm{NP\ VP} & \qquad & \mathrm{NP} \rightarrow \mathrm{Kim} \\
\mathrm{VP} \rightarrow \mathrm{V\ NP} & \qquad & \mathrm{NP} \rightarrow \mathrm{snow} \\
 & & \mathrm{V} \rightarrow \mathrm{adores}
\end{array}
$$

- $S \in C$ is the *start symbol*, a filter on complete results;

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

$$
\begin{array}{rcl}
\text{S} \rightarrow \text{NP VP} & \qquad & \text{NP} \rightarrow \text{Kim} \\
\text{VP} \rightarrow \text{V NP} & \qquad & \text{NP} \rightarrow \text{snow} \\
 & \qquad & \text{V} \rightarrow \text{adores}
\end{array}
$$

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

Generative Grammar

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 $w_i \in \Sigma$

Generative Grammar

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.

Generative Grammar

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.

Treebanks

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).

Treebanks

Generally

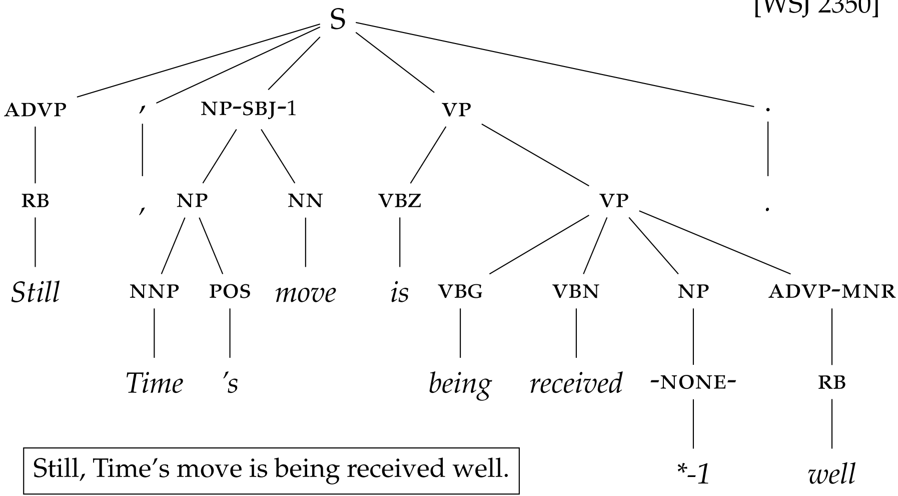
- A *treebank* is a corpus paired with 'gold-standard' (syntactic) analyses
- Can be created by manual annotation or selection among outputs from automated processing (plus correction).

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

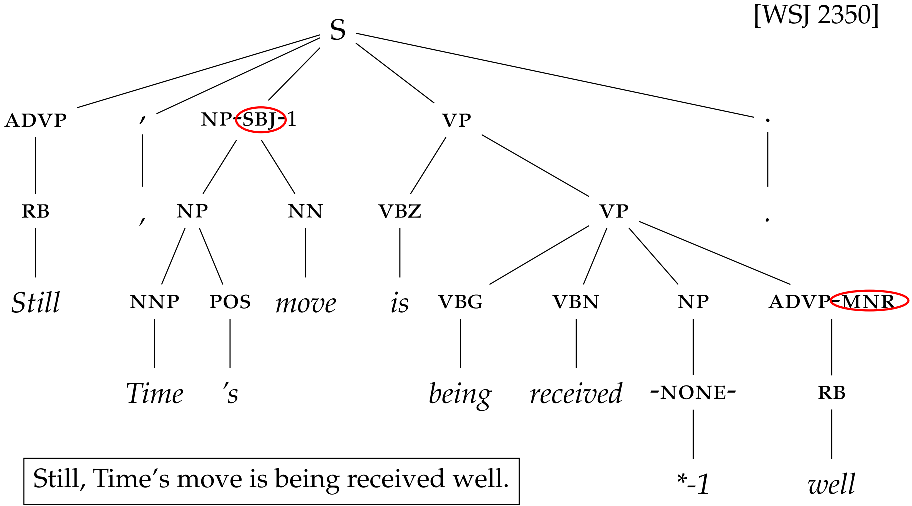
One Example from the Penn Treebank

[WSJ 2350]



Still, Time's move is being received well.

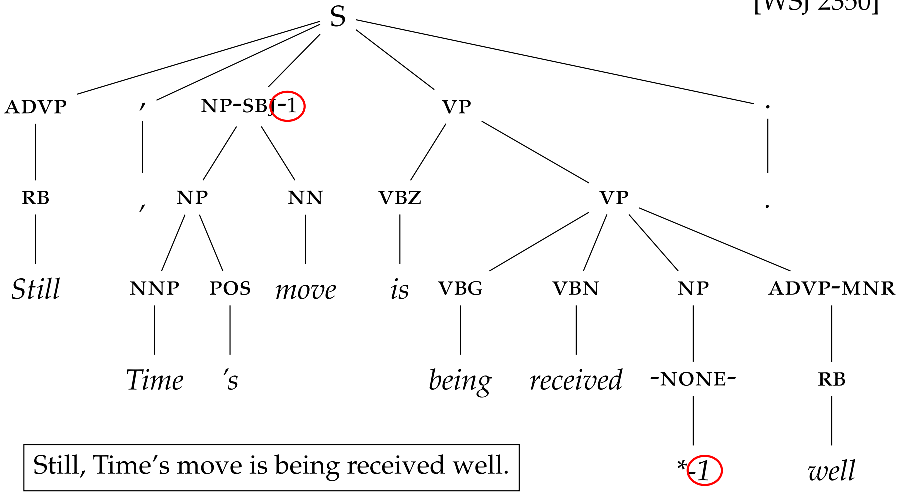
One Example from the Penn Treebank



Still, Time’s move is being received well.

One Example from the Penn Treebank

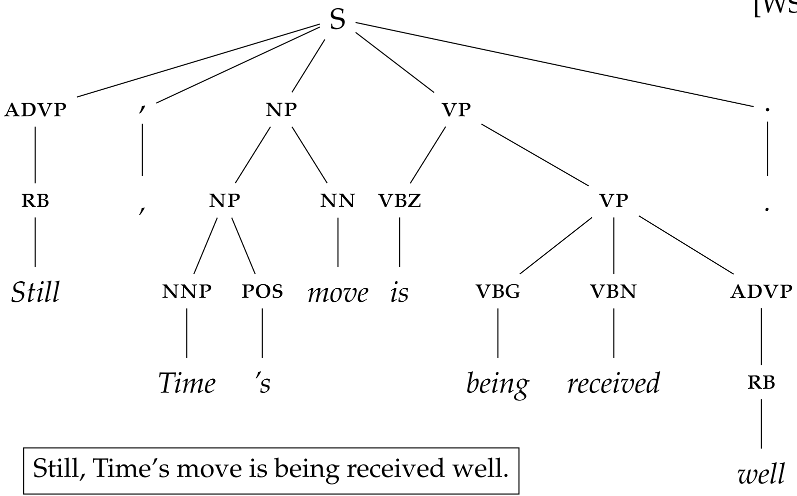
[WSJ 2350]



Still, Time’s move is being received well.

Elimination of Traces and Functions

[WSJ 2350]



Still, Time's move is being received well.

Probabilitic Context-Free Grammars

- We are interested, not just in which trees apply to a sentence, but also to which tree is most likely.

Probabilitic Context-Free Grammars

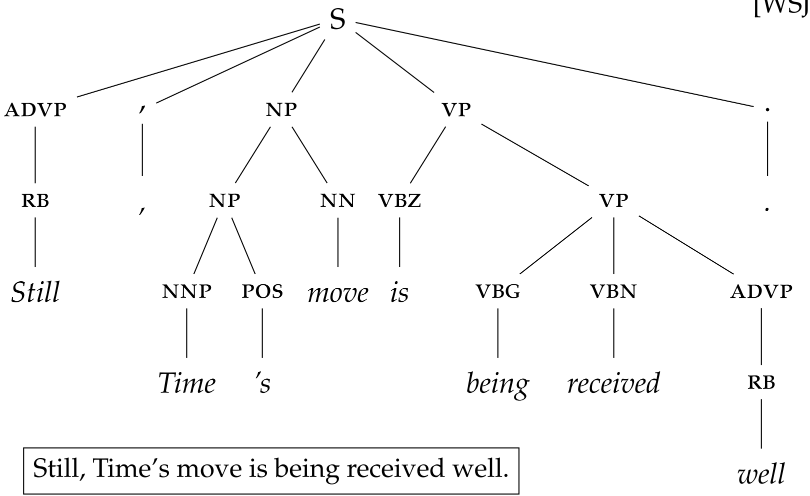
- We are interested, not just in which trees apply to a sentence, but also to which tree is most likely.
- Probabilistic context-free grammars (PCFGs) augment CFGs by adding probabilities to each production, e.g.
  - $S \rightarrow NP\ VP$ 0.6
  - $S \rightarrow NP\ VP\ PP$ 0.4
- These are conditional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

Probabilitic 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]



Still, Time's move is being received well.

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")))))
   (\. "."))
```

Estimating PCFGs (2/3)

RB $\rightarrow$ Still 1

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

Estimating PCFGs (2/3)

RB → Still	1
AVP → RB	1

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

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 $\rightarrow$ Still	1
AVP $\rightarrow$ RB	1
|,| $\rightarrow$,	1

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	1
|,| → ,	1
NNP → Time	1

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	1
|,| → ,	1
NNP → Time	1
POS → ’s	1

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 $\rightarrow$ Still	1
AVP $\rightarrow$ RB	1
|,| $\rightarrow$,	1
NNP $\rightarrow$ Time	1
POS $\rightarrow$'s	1
NP $\rightarrow$ NNP POS	1

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	1
|,| → ,	1
NNP → Time	1
POS → 's	1
NP → NNP POS	1
NN → move	1

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	1
|,| → ,	1
NNP → Time	1
POS → ’s	1
NP → NNP POS	1
NN → move	1
NP → NP NN	1

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	1
|,| → ,	1
NNP → Time	1
POS → 's	1
NP → NNP POS	1
NN → move	1
NP → NP NN	1
VBZ → is	1

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 $\rightarrow$ Still	1
AVP $\rightarrow$ RB	1
|,| $\rightarrow$,	1
NNP $\rightarrow$ Time	1
POS $\rightarrow$ ’s	1
NP $\rightarrow$ NNP POS	1
NN $\rightarrow$ move	1
NP $\rightarrow$ NP NN	1
VBZ $\rightarrow$ is	1
VBG $\rightarrow$ being	1

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	1
|,| → ,	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

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 $\rightarrow$ Still	1
AVP $\rightarrow$ RB	1
|,| $\rightarrow$,	1
NNP $\rightarrow$ Time	1
POS $\rightarrow$'s	1
NP $\rightarrow$ NNP POS	1
NN $\rightarrow$ move	1
NP $\rightarrow$ NP NN	1
VBZ $\rightarrow$ is	1
VBG $\rightarrow$ being	1
VBN $\rightarrow$ received	1
RB $\rightarrow$ well	1

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 $\rightarrow$ Still	1
AVP $\rightarrow$ RB	2
|,| $\rightarrow$,	1
NNP $\rightarrow$ Time	1
POS $\rightarrow$'s	1
NP $\rightarrow$ NNP POS	1
NN $\rightarrow$ move	1
NP $\rightarrow$ NP NN	1
VBZ $\rightarrow$ is	1
VBG $\rightarrow$ being	1
VBN $\rightarrow$ received	1
RB $\rightarrow$ well	1

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

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

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

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

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.

Estimating PCFGs (3/3)

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

$S \rightarrow$ ADVP |,| NP VP \\.	50	$S \rightarrow$ NP VP \\.	400
$S \rightarrow$ NP VP PP \\.	350	$S \rightarrow$ VP !	100
$S \rightarrow$ NP VP S \\.	200	$S \rightarrow$ NP VP	50

Estimating PCFGs (3/3)

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

S → ADVP |,| NP VP \\.	50	S → NP VP \\.	400
S → NP VP PP \\.	350	S → VP !	100
S → NP VP S \\.	200	S → NP VP	50

$$P(S \rightarrow ADVP \,|,|\, NP\ VP\ \backslash.) \approx \frac{C(S \rightarrow ADVP \,|,|\, NP\ VP\ \backslash.)}{C(S)}$$

Estimating PCFGs (3/3)

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

S $\rightarrow$ ADVP |,| NP VP \\.	50	S $\rightarrow$ NP VP \\.	400
S $\rightarrow$ NP VP PP \\.	350	S $\rightarrow$ VP !	100
S $\rightarrow$ NP VP S \\.	200	S $\rightarrow$ NP VP	50

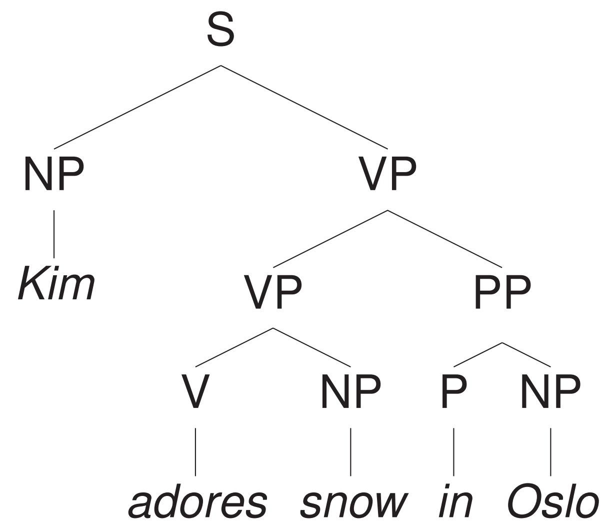
$$
\begin{aligned}
P(S \rightarrow ADVP\ |,|\ NP\ VP\ \backslash.) &\approx \frac{C(S \rightarrow ADVP\ |,|\ NP\ VP\ \backslash.)}{C(S)} \\
&= \frac{50}{1150} \\
&= 0.0435
\end{aligned}
$$

Parsing with CFGs: Moving to a Procedural View

S → NP VP
VP → V | V NP | VP PP
NP → NP PP
PP → P NP
NP → Kim | snow | Oslo
V → adores
P → 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.

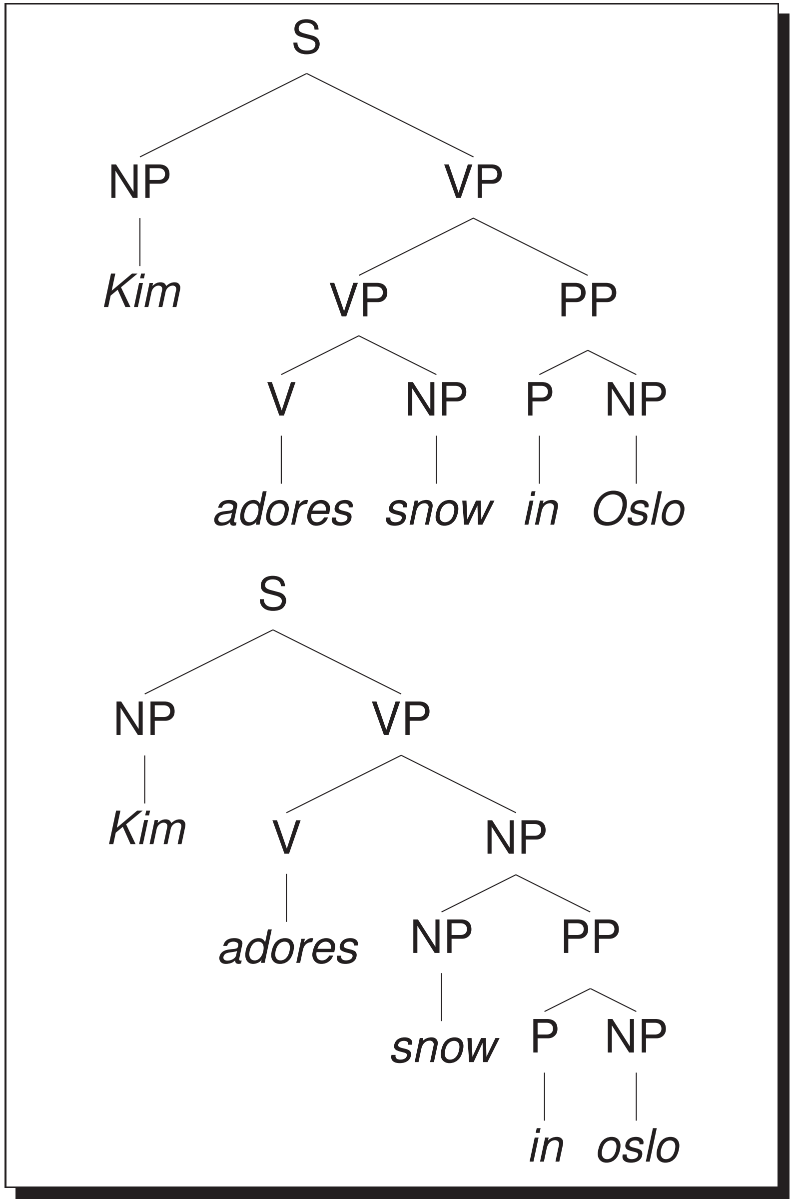


Parsing with CFGs: Moving to a Procedural View

S → NP VP
VP → V | V NP | VP PP
NP → NP PP
PP → P NP
NP → Kim | snow | Oslo
V → adores
P → 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.

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

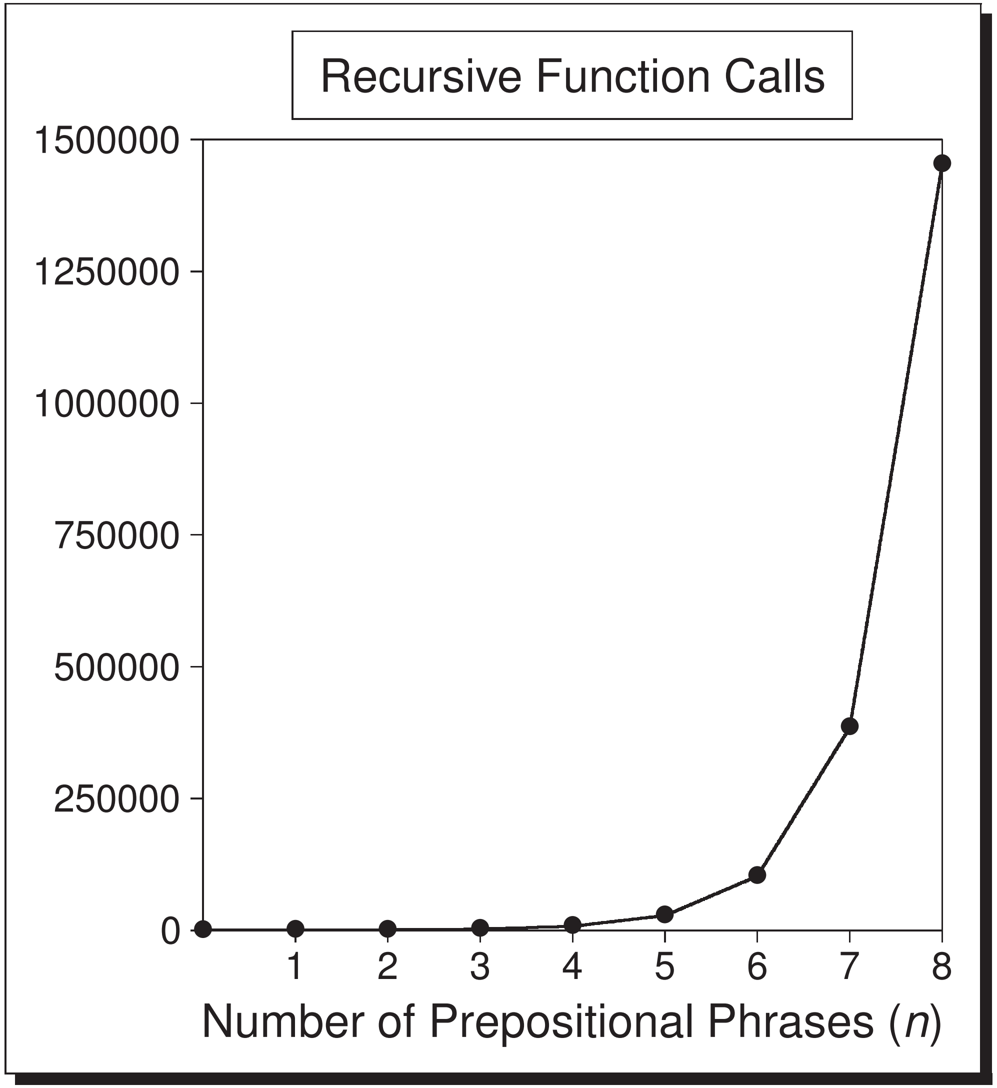
- Each result records a (partial) tree and remaining input to be parsed;
- complete results consume the full input string and are rooted in S;
- whenever a RHS is fully instantiated, a new tree is built and returned;
- all results at each level are combined and successively accumulated.

The Recursive Descent Parser

```lisp
(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))))
```

```lisp
(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

n	**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;

$\rightarrow$ assume bottom-up as basic search strategy for remainder of the course.

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;

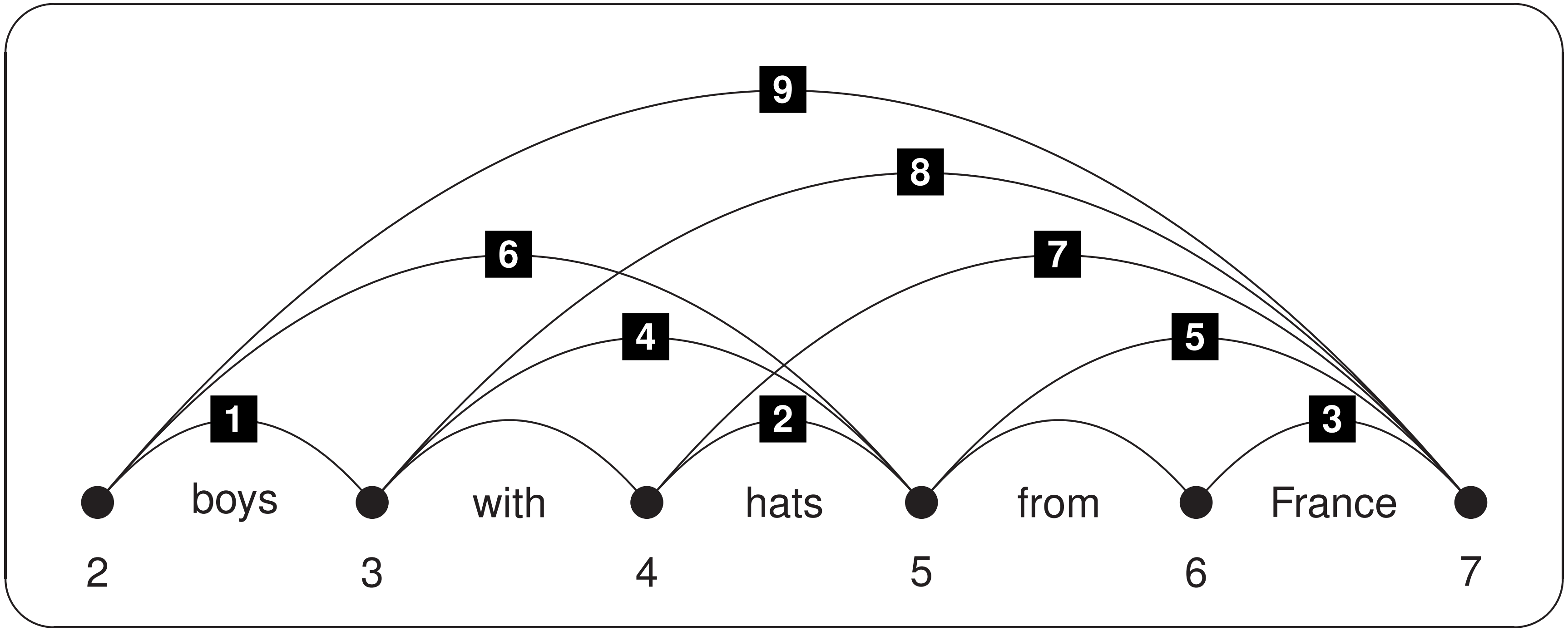
$\rightarrow$ assume bottom-up as basic search strategy for remainder of the course.

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: $\boxed{1} \mid \boxed{2} \mid \boxed{3} \mid \boxed{6} \mid \boxed{7} \mid \boxed{9}$; PPs: $\boxed{4} \mid \boxed{5} \mid \boxed{8}$; $\boxed{9} \equiv \boxed{1} + \boxed{8} \mid \boxed{6} + \boxed{5}$;
- *parse forest* — a single item represents multiple trees [Billot & Lang, 89].



The CKY (Cocke, Kasami, & Younger) Algorithm

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

$[0,2] \leftarrow [0,1] + [1,2]$
$\ldots$
$[0,5] \leftarrow [0,1] + [1,5]$
$[0,5] \leftarrow [0,2] + [2,5]$
$[0,5] \leftarrow [0,3] + [3,5]$
$[0,5] \leftarrow [0,4] + [4,5]$

	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).

Limitations of the CKY Algorithm

Built-In Assumptions

- *Chomsky Normal Form* grammars: $\alpha \to \beta_1\beta_2$ or $\alpha \to \gamma$ ($\beta_i \in C$, $\gamma \in \Sigma$);
- 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: $[1, 2, \text{VP} \to \text{V} \bullet \text{NP}]$;
- parser can fill in chart cells in *any* order and guarantee completeness.