

# **Introduction to Lexicalized Grammars for Natural Language Parsing**

Giorgio Satta

University of Padua

<http://www.dei.unipd.it/~satta/>

Position of **parsing** problem :

- Extract underlying/hidden information from input sentence
  - Phrase structure
  - Grammatical relations
- Pass to 'higher' layers
  - Information extraction
  - Question/answering
  - Machine translation

Models (+ lexicalized versions) :

- Finite automata/transducers
- Context-free grammars
- Dependency grammars
- Tree-adjoining grammars
- Synchronous context-free grammars

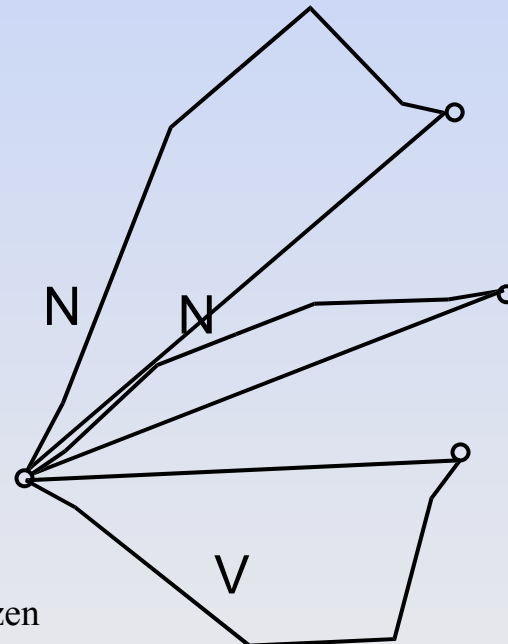
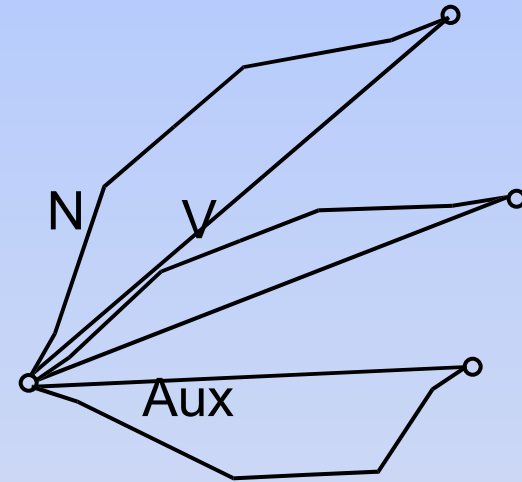
Finite automata (FA) are exploited in the following tasks :

- Part-of-speech tagging
- Named entity recognition
- Text normalization
- Chunking
- Shallow parsing
- Approximation of more expressive models

# Finite automata

Complexity :

- $M$  finite automaton
- $w$  string
- **Deterministic** case  
Process in time  $O(|w|)$
- **Nondeterministic** case  
Process in time  $O(|M||w|)$



Pros :

- `Nice' closure properties (union, intersection, complementation, concatenation, etc.)
- Processing efficiency
- Enrich with probabilities/weights

Cons :

- Limited generative power
- Lacks succinctness

# Context-free grammars

---

Context-free grammars (CFGs) provide hierarchical description of sentences, called **phrase structures**

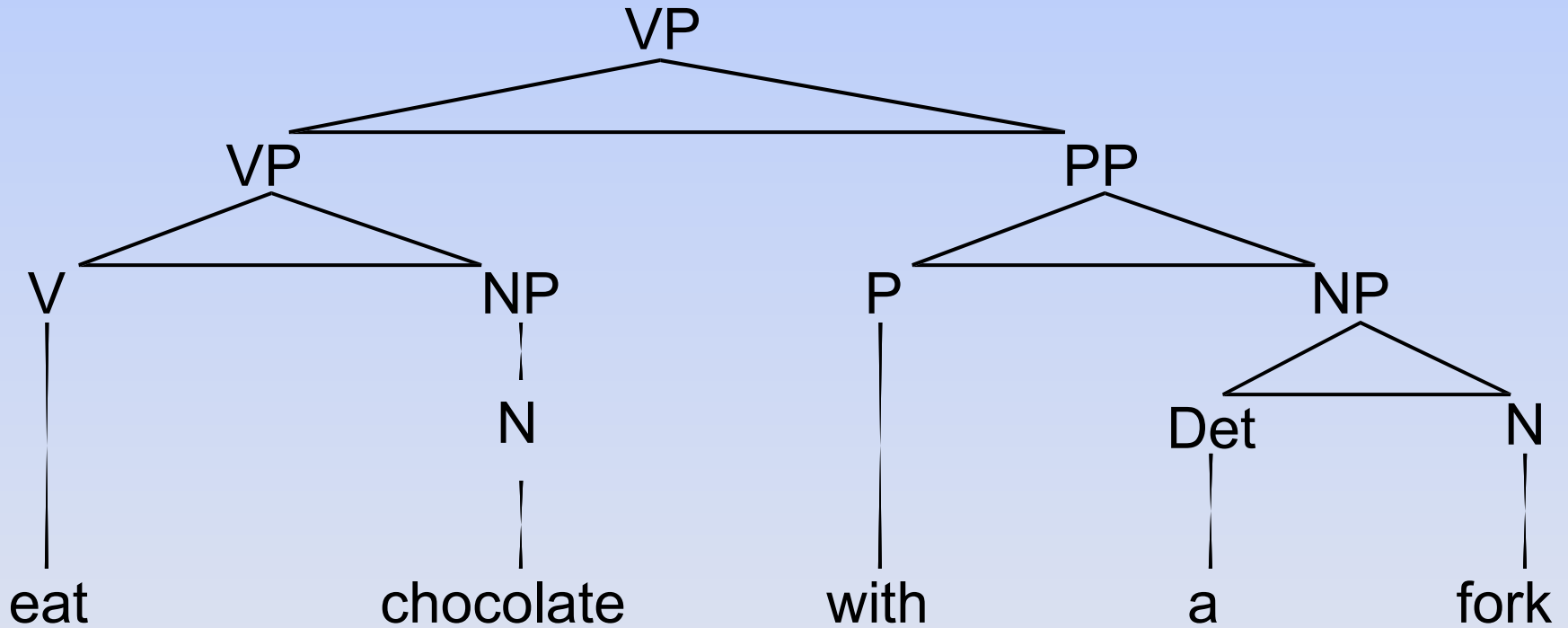
Example :

- $S \rightarrow NP VP$
- $NP \rightarrow NP PP \mid Det N \mid N$
- $VP \rightarrow VP PP \mid V NP, PP \rightarrow P NP$
- $N \rightarrow chocolate \mid fork$
- $V \rightarrow eat, Det \rightarrow a, P \rightarrow with$
- ...

# Context-free grammars

---

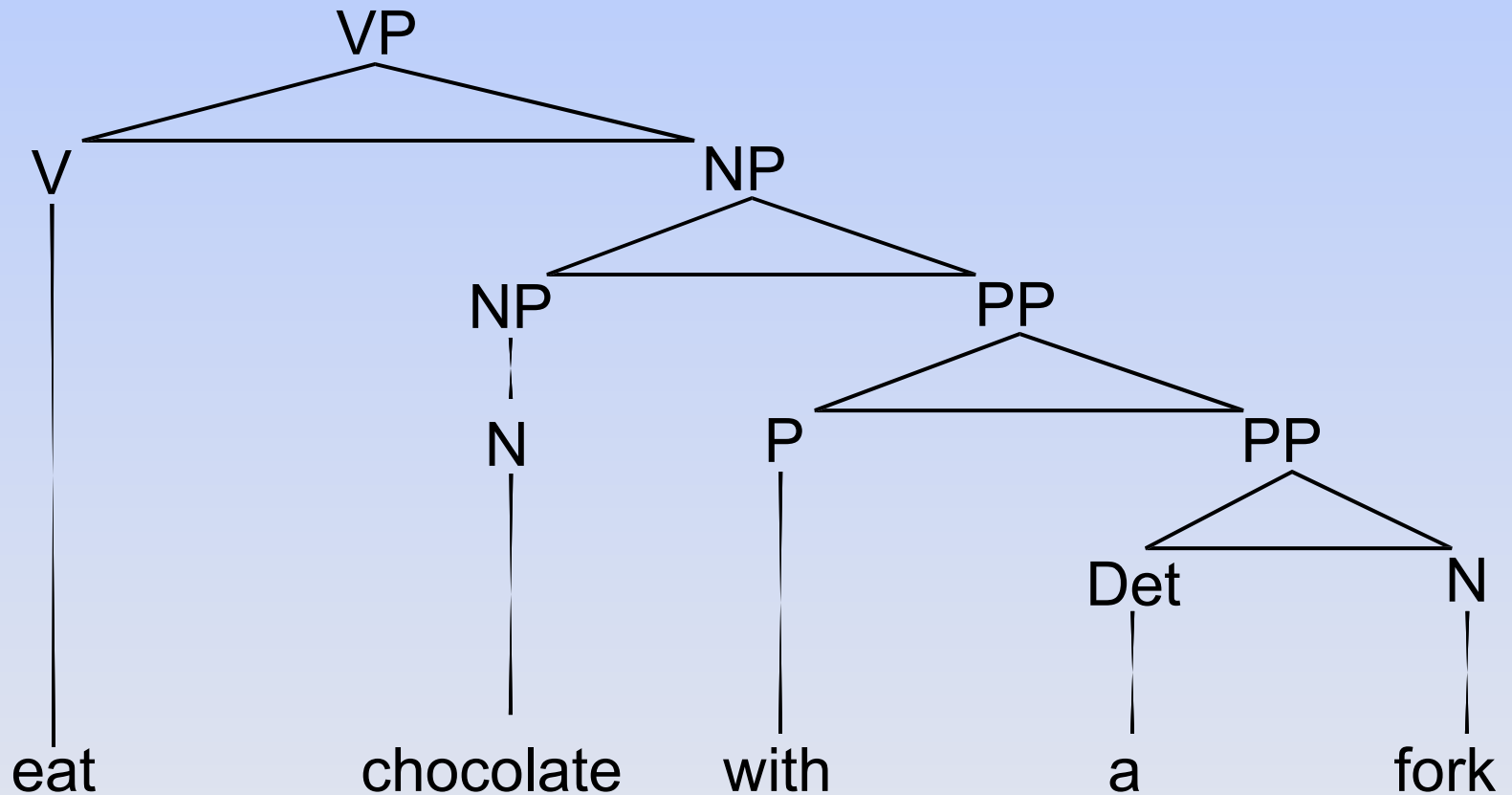
Example (cont'd) :



# Context-free grammars

---

Example (cont'd) :

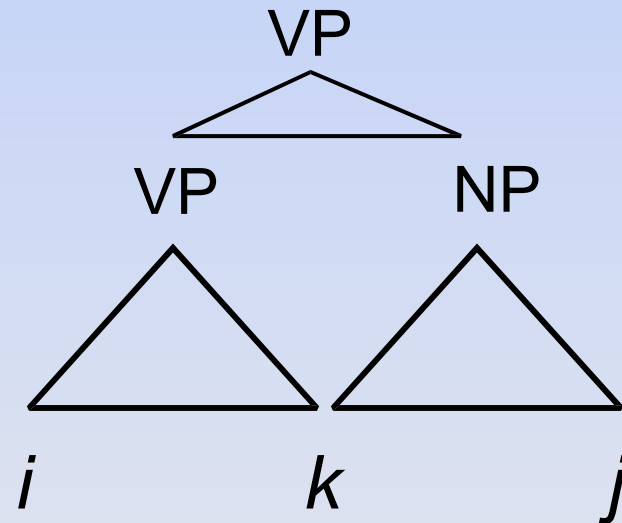


# Context-free grammars

---

Complexity :

- $G$  context-free grammar
- $w$  string
- Parse in time  $O(|G| |w|^3)$ 
  - Check all rules
  - Check all tuples  $i, k, j$



# Context-free grammars

---

Pros :

- Expressiveness
- Succinctness
- Enrich with probabilities/weights

Cons :

- Cannot express **syntactic preferences** that are sensitive to lexical words
- There is no control over **word selection**

# Context-free grammars

---

Syntactic preferences :

- I eat  $[_{NP} [_{NP} \text{chocolate}] [_{PP} \text{with strawberries}]]$
- I  $[_{VP} [_{VP} \text{eat} [_{NP} \text{chocolate}]] [_{PP} \text{with a fork}]]$
- ? I  $[_{VP} [_{VP} \text{eat} [_{NP} \text{chocolate}]] [_{PP} \text{with strawberries}]]$
- ? I eat  $[_{NP} [_{NP} \text{chocolate}] [_{PP} \text{with a fork}]]$

Word selection :

- Lexical
  - Nora convened the meeting
  - ? Nora convened the party
- Semantics
  - Peggy solved two puzzles
  - ? Peggy solved two goats
- World knowledge
  - Mary shelved some books
  - ? Mary shelved some cooks

# Lexicalized context-free grammars

In a **lexicalized CFG** (LCFG) each rule is specialized for one or more **lexical items**

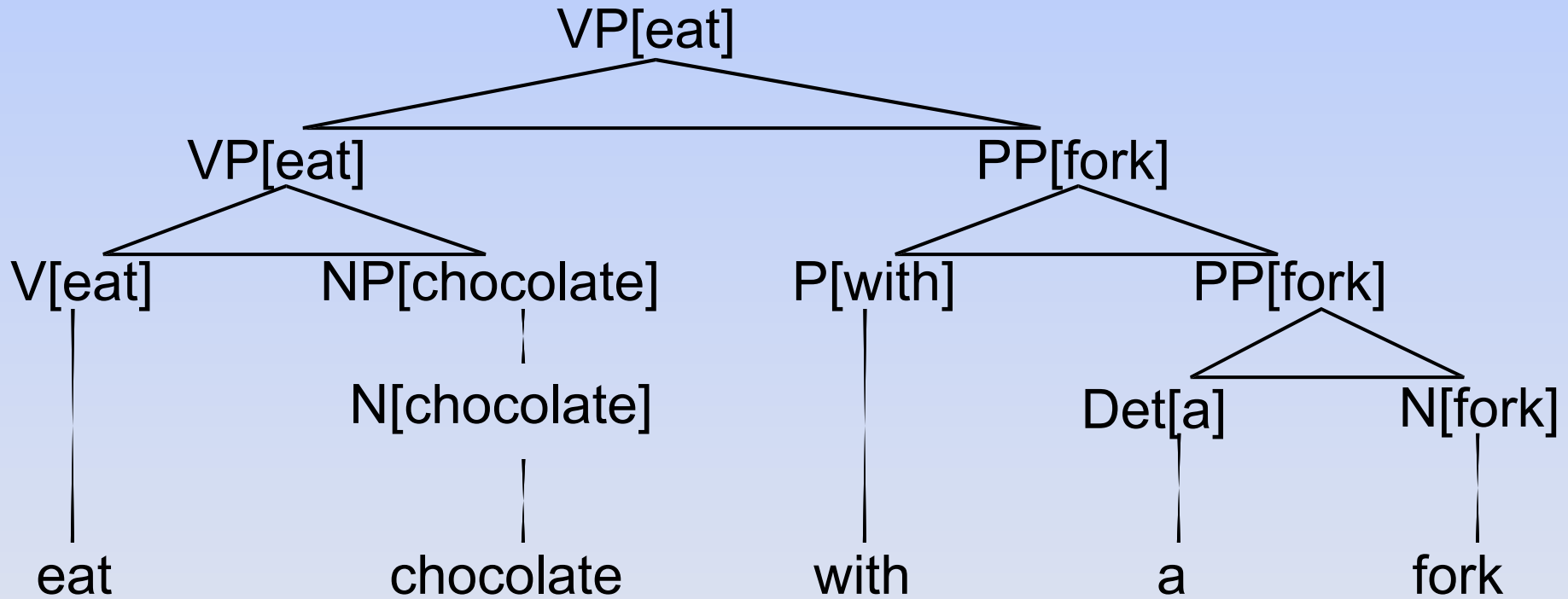
Example of 2-lexicalized CFG (2-LCFG) :

- $VP[eat] \rightarrow VP[eat] PP[fork]$
- $PP[fork] \rightarrow P[with] NP[fork]$
- $VP[eat] \rightarrow V[eat] NP[chocolate]$
- $V[eat] \rightarrow eat$
- $N[fork] \rightarrow fork$
- ...

# Lexicalized context-free grammars

---

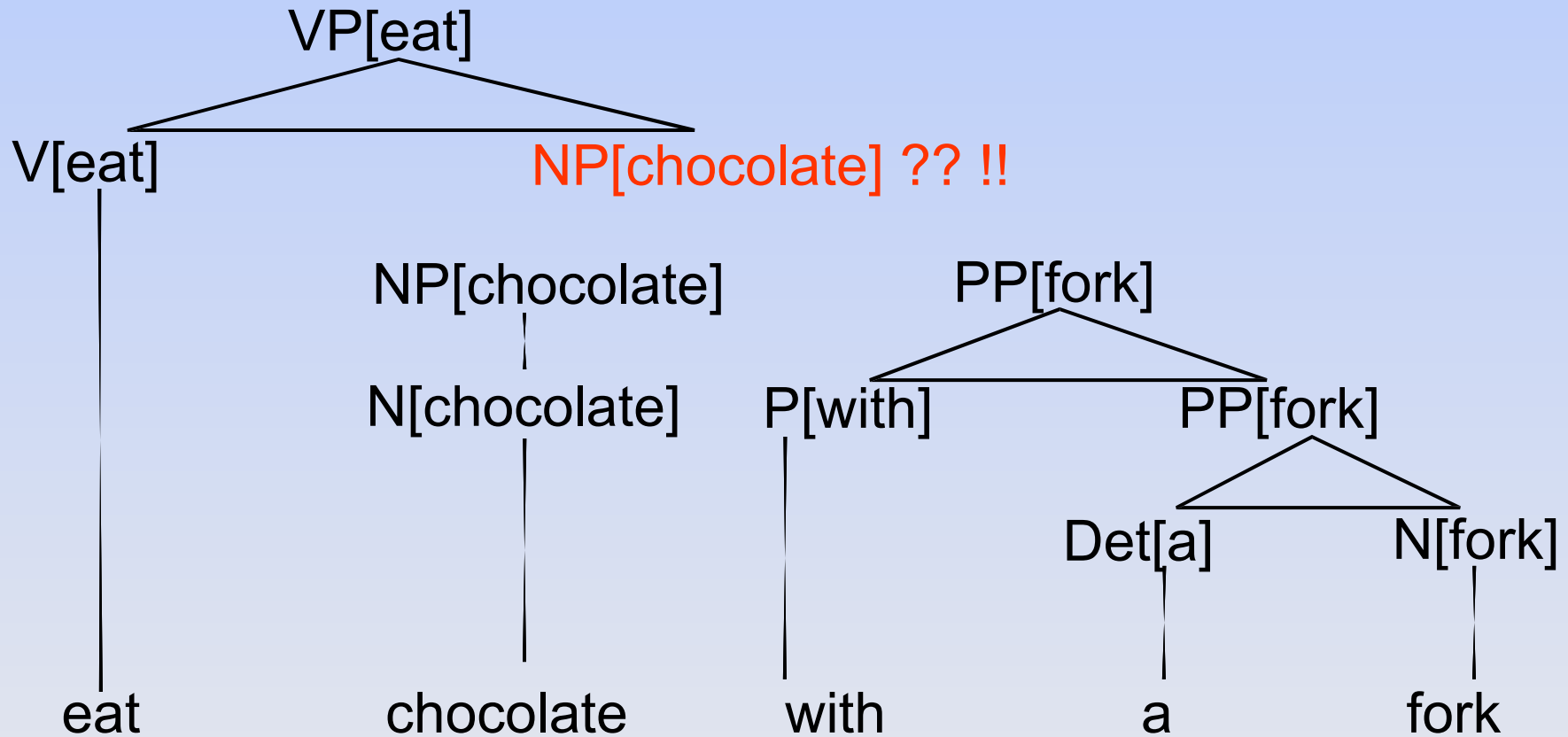
Example (cont'd) :



# Lexicalized context-free grammars

---

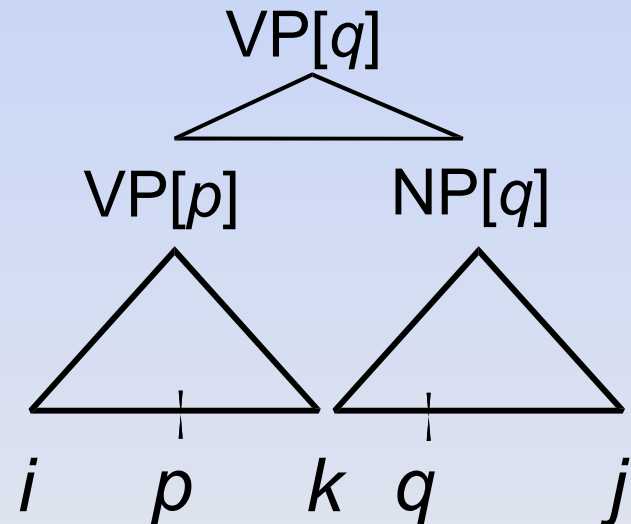
Example (cont'd) :



# Lexicalized context-free grammars

Complexity :

- $G$  2-lexicalized context-free grammar
- $w$  string
- Standard algorithm runs in time  $O(|u(G)||w|^5)$ 
  - $u(G)$  is the 'skeleton' CFG
  - Check all rules
  - Check all tuples  $i, k, j, p, q$
- **Improved** upper bound: time  $O(|G||w|^4)$



## **Probabilistic context-free grammars**

---

Each rule associated with a **probability**

$$0 \leq \Pr( A \rightarrow \alpha ) \leq 1$$

**Normalization** property

$$\sum_{\alpha} \Pr( A \rightarrow \alpha ) = 1, \text{ for every } A$$

Probability of a tree is the product of probabilities of all rules involved

## Probabilistic context-free grammars

---

Supervised learning empirically estimates probabilities from a tree sample (tree bank)

The estimator maximizes the likelihood of the sample :

$$\Pr( A \rightarrow \alpha ) = \frac{\#( A \rightarrow \alpha )}{\sum_{\alpha} \#( A \rightarrow \alpha )}$$

where # denotes the count over the sample

## Probabilistic context-free grammars

---

Unsupervised learning estimates probabilities from a sentence sample, again by likelihood maximization

$$\Pr(A \rightarrow \alpha) = \frac{\sum_{w \in C} f(w, C) \cdot E_{p(\cdot|w)} f(A \rightarrow \alpha)}{\sum_{w \in C} f(w, C) \cdot E_{p(\cdot|w)} f(A)}$$

The estimator cannot be expressed in closed form

A solution is found by means of the expectation maximization method, realized by the inside/outside algorithm

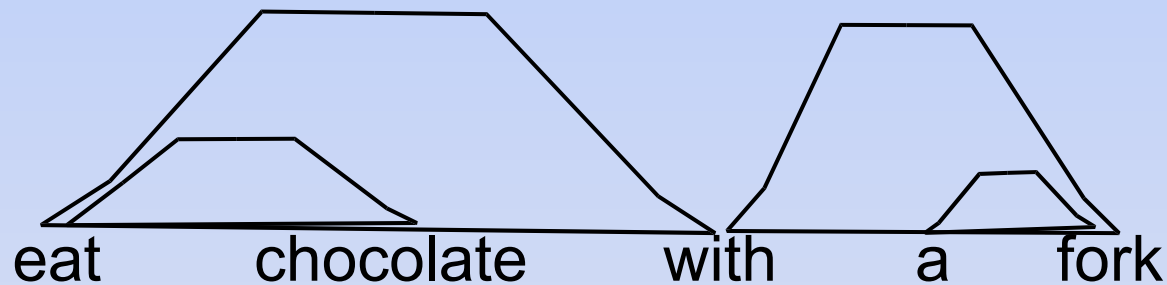
This algorithm only finds a local maximum

# Dependency grammars

---

Provide binary **dependency relation** for words in the sentence

Example :

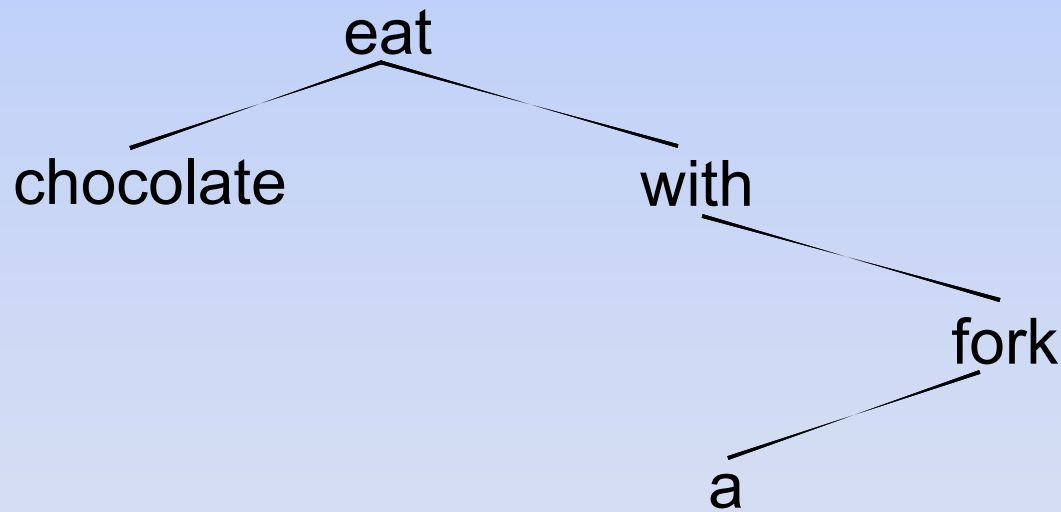


A dependency analysis is called **projective** if there is no crossing between any two links

# Dependency grammars

---

Can be viewed as a tree whose nodes are words, called **dependency tree**



- Related to phrase structure in case of projectivity
- No use of nonterminals

# Dependency grammars

---

Complexity :

- $G$  dependency grammar
- $w$  string
- Projective grammar: parse in time  $O(|G||w|^3)$ 
  - Triangularization methods
  - Spanning tree algorithms
  - Dynamic programming

# **Dependency grammars**

---

Pros :

- More efficient to learn
- In practice, more efficient to parse
- Enrich with probabilities/weights
- Getting a lot of attention now, so wait and see ...

Cons :

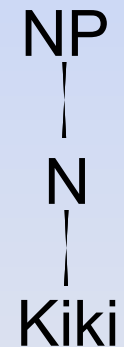
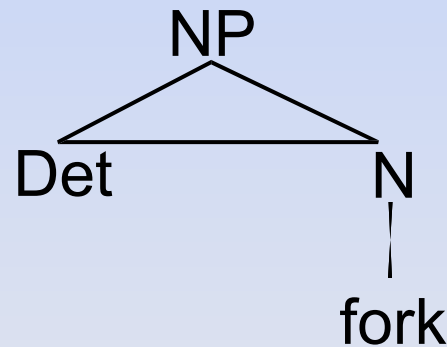
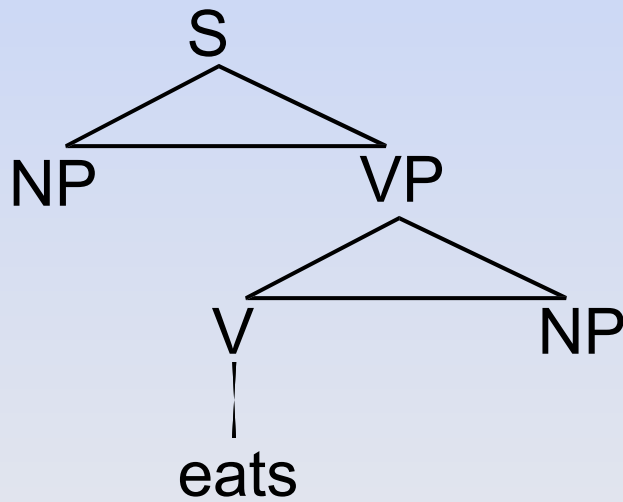
- Less informative than lexicalized phrase structures

# Lexicalized tree adjoining grammars

Lexicalized tree adjoining grammars (LTAGs) are based on tree rewriting

A grammar is composed of a finite set of trees, each specialized for some lexical item. Example:

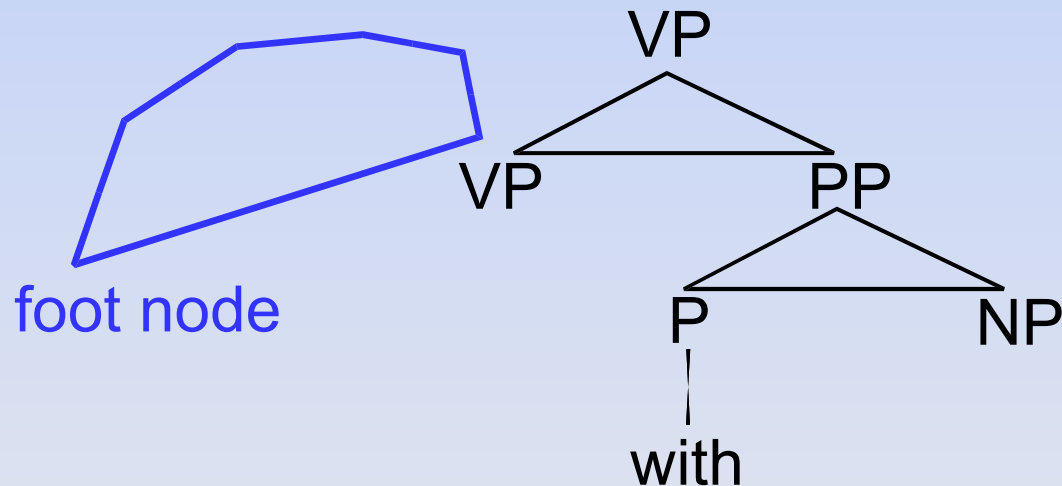
- Initial trees



# Lexicalized tree adjoining grammars

Example (cont'd) :

- **Auxiliary trees** with a special node called **foot node**, having the same label as the root node

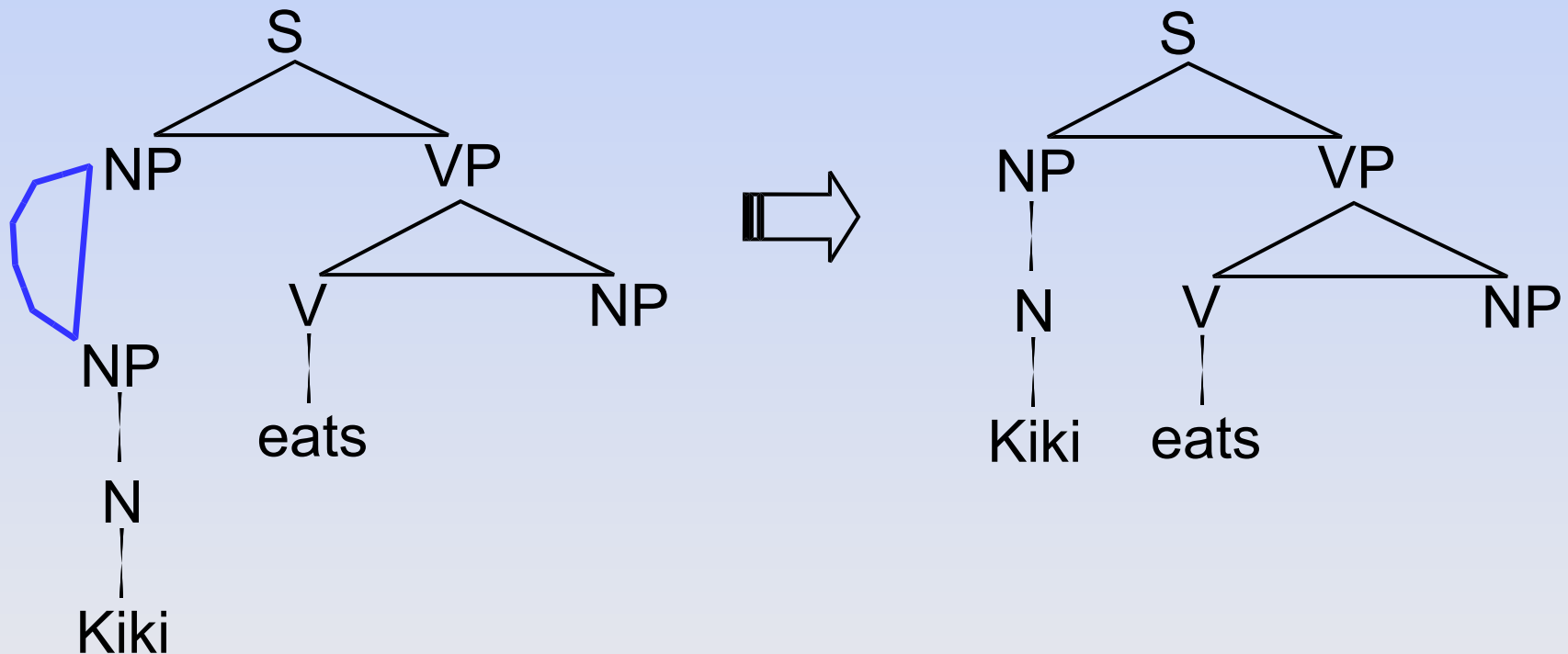


# Lexicalized tree adjoining grammars

Two basic operations define the rewriting process

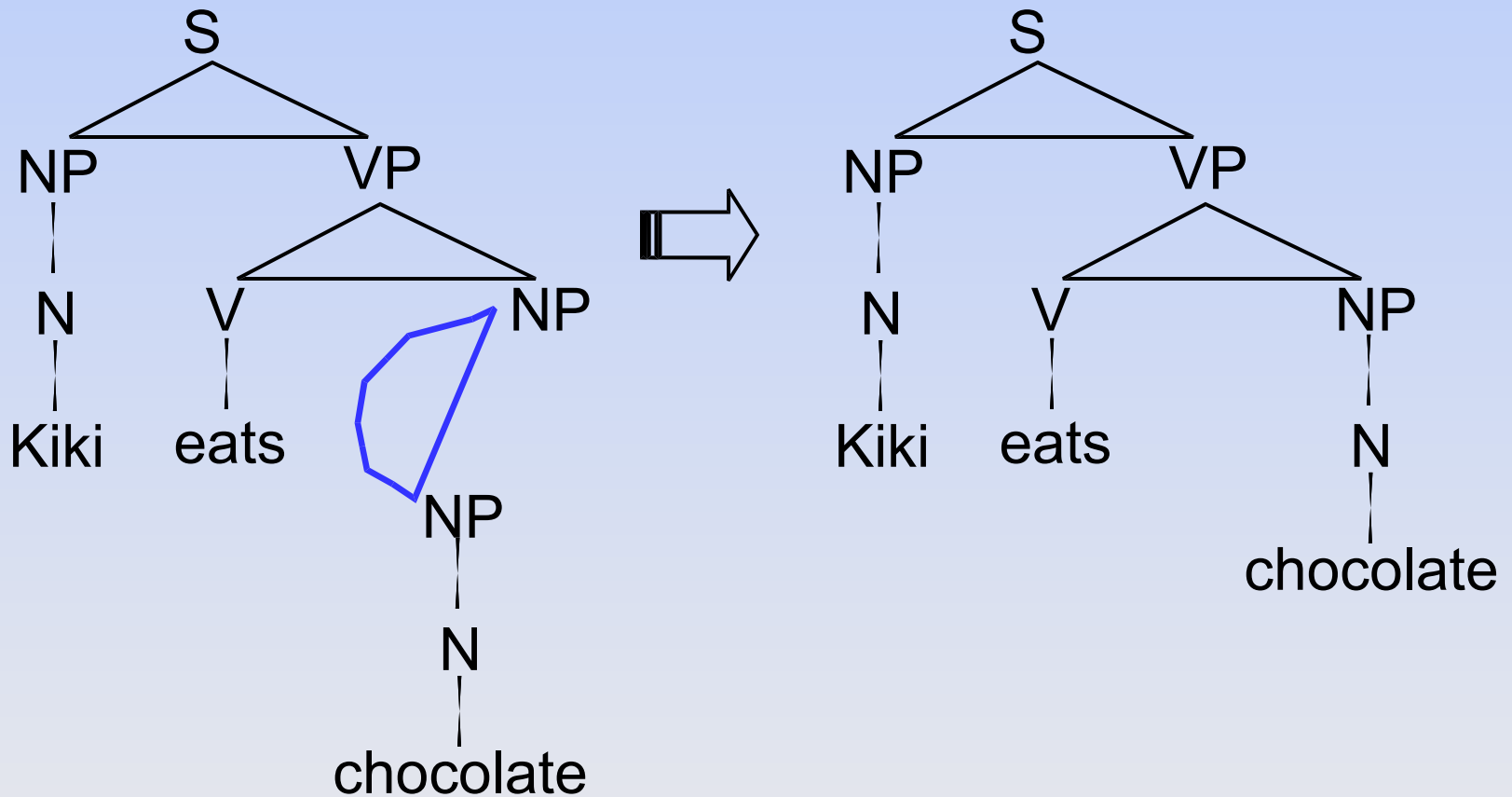
Example (cont'd) :

- Tree insertion



# Lexicalized tree adjoining grammars

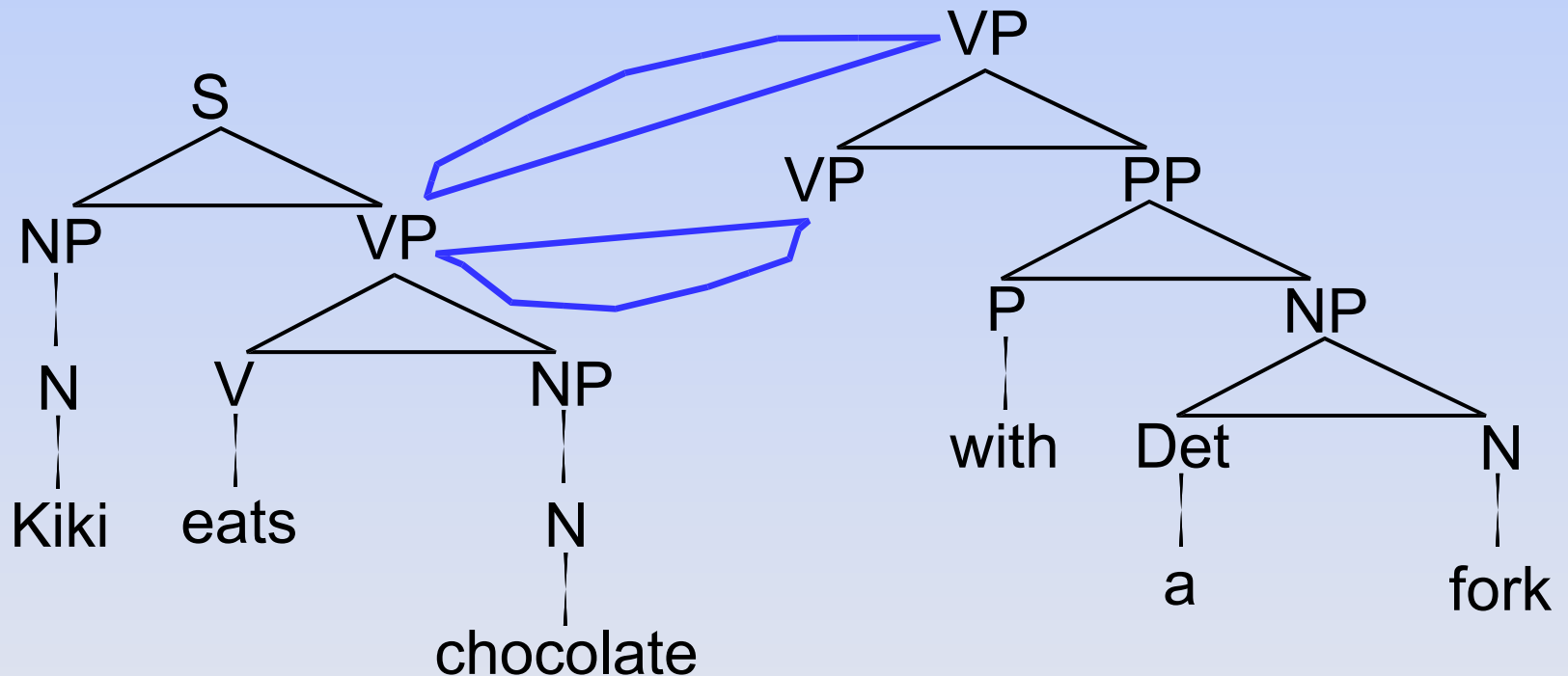
Example (cont'd) :



# Lexicalized tree adjoining grammars

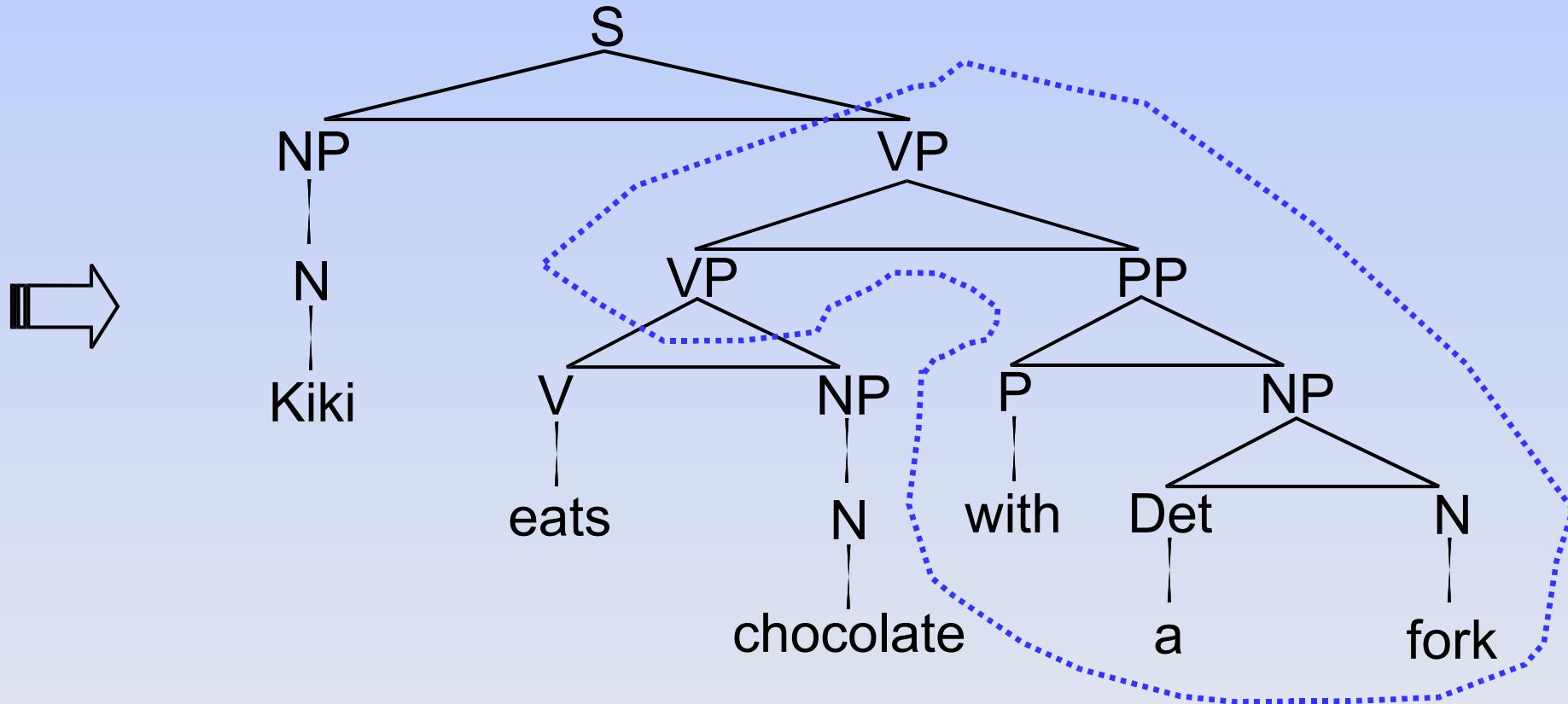
Example (cont'd) :

- Tree adjunction



# Lexicalized tree adjoining grammars

Example (cont'd) :



## Lexicalized tree adjoining grammars

Tree insertion does not increase the generative power of context-free grammars

Tree adjunction **increases** the generative power of tree adjoining grammars

The formal language

$$\{ a^n b^n c^n d^n \mid n \geq 1 \}$$

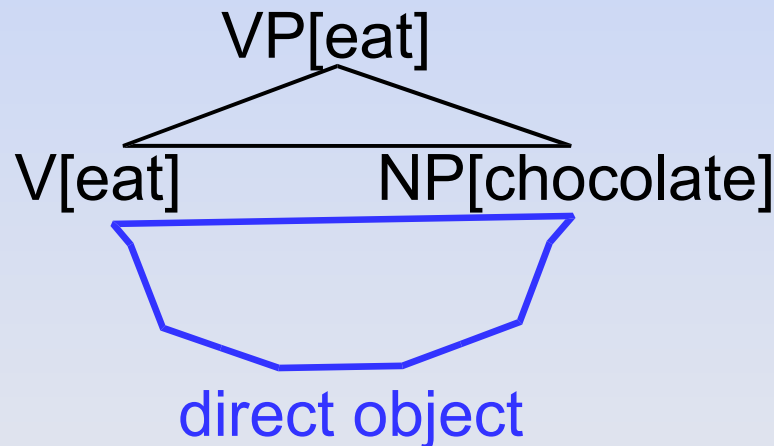
can be generated by a TAG (or LTAG) but not by a CFG

# Lexicalized tree adjoining grammars

An elementary object of a grammar defines the **domain of locality** of the formalism

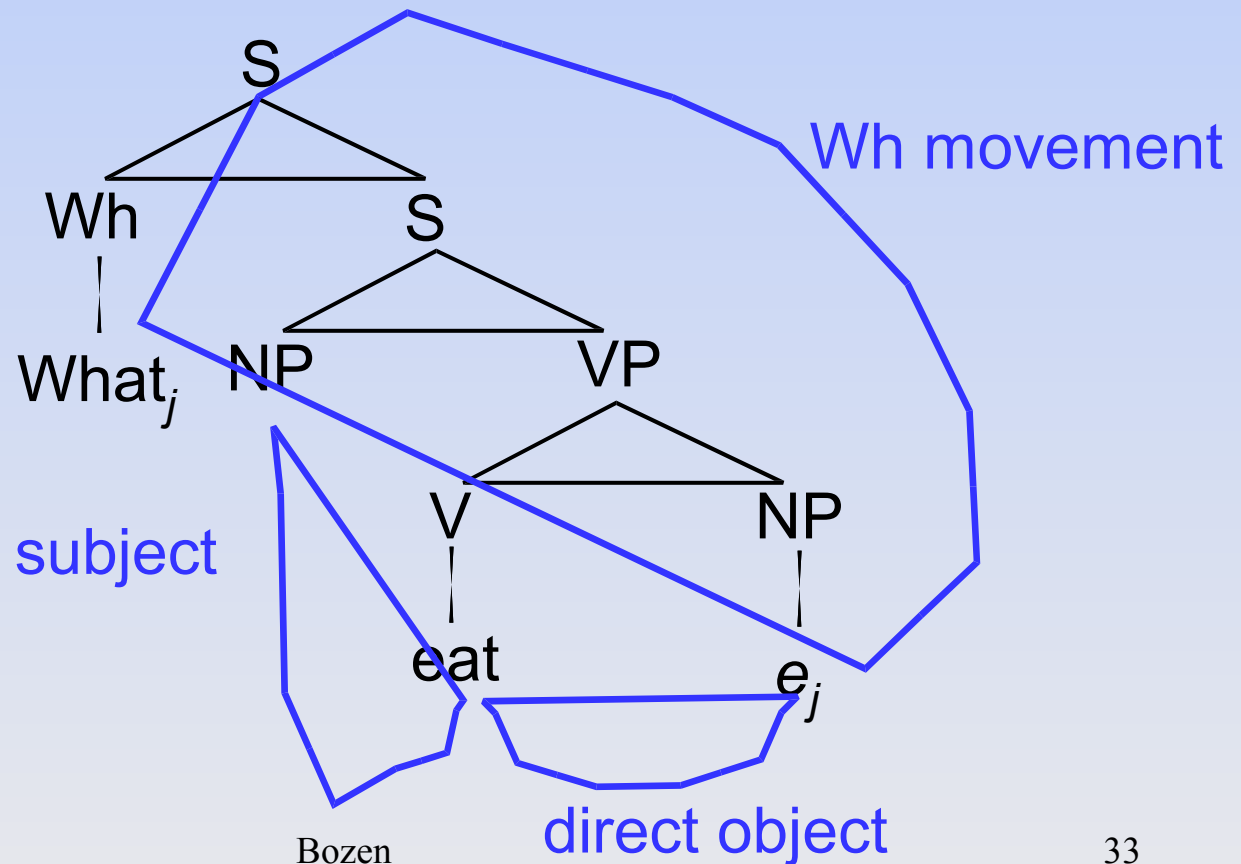
The domain of locality is used to encode the grammatical relations we need to extract when parsing

Context-free grammars :



# Lexicalized tree adjoining grammars

In tree adjoining grammars **several** grammatical relations pertaining to a single lexical element can be expressed within the same domain (aux removed in example)



# Lexicalized tree adjoining grammars

---

Complexity :

- $G$  tree adjoining grammar
- $w$  string

Parse in time :

- Most complex operation is adjunction
- $O(|G||w|^6)$ , unlexicalized
- $O(|G||w|^7)$ , 2-lexical case

# **Lexicalized Tree adjoining grammars**

---

Pros :

- Increase generative power of context-free grammars
- Extended domain of locality
- Enrich with probabilities/weights

Cons :

- Computationally demanding

# Synchronous context-free grammars

---

Synchronous context-free grammars (SCFGs) are used as translation models

A SCFG is based on three components:

- Context free grammar for source language
- Context free grammar for target language
- **Pairing relation** on the productions of the two grammars and on the nonterminals in their right-hand sides

# Synchronous context-free grammars

---

Example :

VB → PRP<sup>(1)</sup> VB1<sup>(2)</sup> VB2<sup>(3)</sup>

VB → PRP<sup>(1)</sup> VB2<sup>(3)</sup> VB1<sup>(2)</sup>

VB2 → VB<sup>(1)</sup> TO<sup>(2)</sup>

VB2 → TO<sup>(2)</sup> VB<sup>(1)</sup> ga

TO → TO<sup>(1)</sup> NN<sup>(2)</sup>

TO → NN<sup>(2)</sup> TO<sup>(1)</sup>

PRP → he

PRP → kare ha

VB1 → adores

VB1 → daisuki desu

VB → listening

VB → kiku no

TO → to

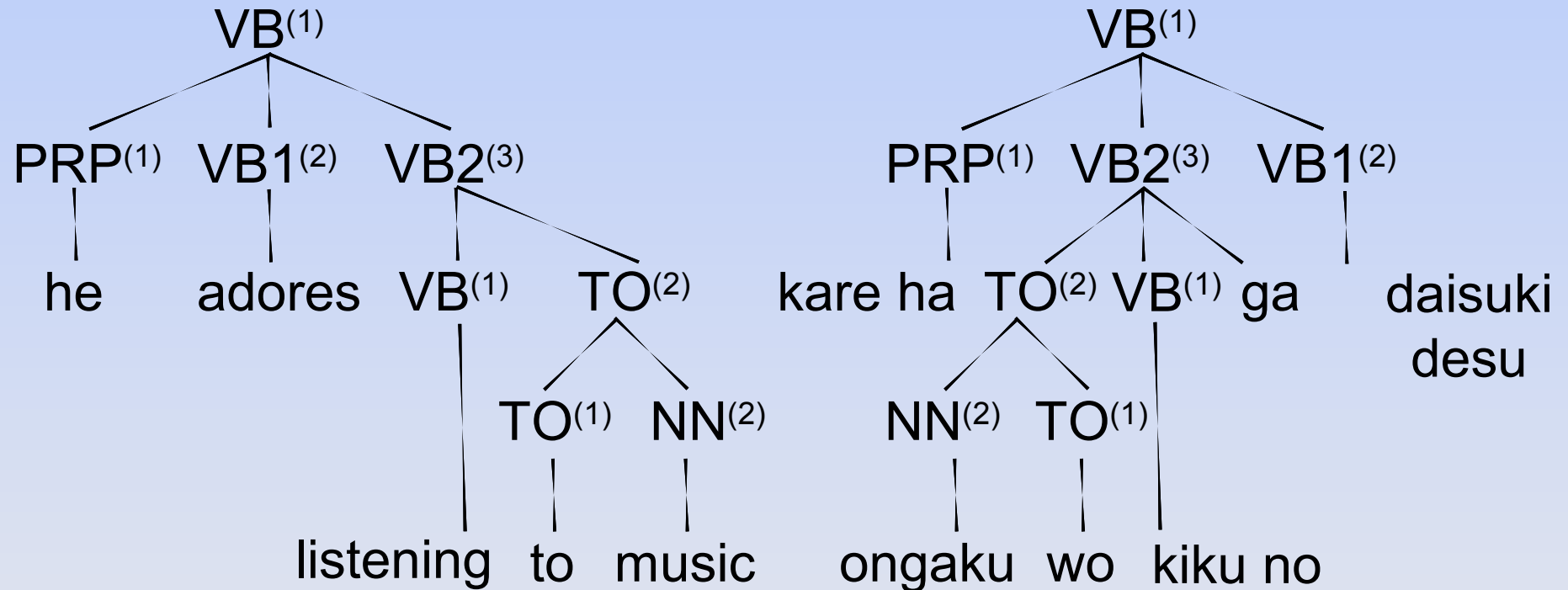
TO → wo

NN → music

NN → ongaku

# Synchronous context-free grammars

Example (cont'd):



## **Synchronous context-free grammars**

---

Synchronous CFGs generate **pairs** of trees/strings,  
where each component is a translation of the other

Synchronous CFGs can be extended with probabilities and  
can be empirically estimated from aligned corpora

Synchronous CFG have recently been used in alternative to  
finite transducers

## Synchronous context-free grammars

---

We can define two problems :

- **Parse** pair of strings  $u, v$
- **Translate** string  $u$  into set of all its string translations

Both problems are NP-hard

# **Synchronous context-free grammars**

---

Pros :

- More powerful than finite transducers for translations
- Can be extended with probabilities
- Higher accuracy

Cons :

- Intractable

# Conclusions

---

We have discussed several models/results for natural language parsing (+ lexicalized versions)

- Finite automata/transducers
- Context-free grammars
- Dependency grammars
- Tree-adjoining grammars
- Synchronous context-free grammars